kalles-fraktaler-2

Claude Heiland-Allen

2017–2019

Contents

1 32.mk . . . 4
2 64.mk . . . 4
3 64+.mk . . . 5
4 cl/common.cl . . . 5
5 cl/double_post.cl . . . 39
6 cl/double_post_m.cl . . . 39
7 cl/double_post_rc.cl . . . 40
8 cl/double_pre.cl . . . 40
9 cl/floatexp_post.cl . . . 41
10 cl/floatexp_post_m.cl . . . 41
11 cl/floatexp_post_rc.cl . . . 42
12 cl/floatexp_pre.cl . . . 42
13 cl/formula.xsl . . . 43
14 cl/opencl.cpp . . . 48
15 cl/opencl.h . . . 62
16 cl/opencl.xsl . . . 65
17 cl/s2c.sh . . . 66
18 cl/softfloat_post.cl . . . 66
19 cl/softfloat_post_m.cl . . . 67
20 cl/softfloat_post_rc.cl . . . 67
21 cl/softfloat_pre.cl . . . 67
22 common/barrier.h . . . 69
23 common/bitmap.cpp . . . 70
24 common/bitmap.h . . . 71
25 common/FolderBrowser.cpp . . . 71
26 common/FolderBrowser.h . . . 74
27 common/getimage.cpp . . . 75
28 common/getimage.h . . . 86
29 common/matrix.cpp . . . 87
30 common/matrix.h . . . 88
31 common/parallell.cpp . . . 89
32 common/parallell.h . . . 93
33 common/StringVector.cpp . . . 94
34 common/StringVector.h . . . 117
35 common/tooltip.cpp . . . 119
36 common/tooltip.h . . . 125
37 examples/examples.kfs . . . 125
38 examples/examples.sh . . . 126
39 examples/examples.xsl . . . 127
40 formula/common.cpp . . . 127
41 formula/formula.h . . . 140
42 formula/formula.xml . . . 144

43 formula/formula.xsl . . . 241
44 formula/generated/formula.h . . . 286
45 formula/generated/.gitignore . . . 289
46 fraktal_sft/CDecNumber.cpp . . . 289
47 fraktal_sft/CDecNumber.h . . . 291
48 fraktal_sft/CFixedFloat.cpp . . . 295
49 fraktal_sft/CFixedFloat.h . . . 295
50 fraktal_sft/check_for_update.cpp . . . 304
51 fraktal_sft/check_for_update.h . . . 305
52 fraktal_sft/cmdline.cpp . . . 306
53 fraktal_sft/cmdline.h . . . 311
54 fraktal_sft/colour.h . . . 312
55 fraktal_sft/complex.h . . . 315
56 fraktal_sft/cursor1.cur . . . 320
57 fraktal_sft/double_perturbation.cpp . . . 320
58 fraktal_sft/double_reference.cpp . . . 327
59 fraktal_sft/exr.cpp . . . 331
60 fraktal_sft/exr.h . . . 337
61 fraktal_sft/floatexp_approximation.cpp . . . 338
62 fraktal_sft/floatexp.h . . . 353
63 fraktal_sft/floatexp_perturbation.cpp . . . 363
64 fraktal_sft/floatexp_reference.cpp . . . 367
65 fraktal_sft/fraktal_sft.cpp . . . 375
66 fraktal_sft/fraktal_sft.h . . . 444
67 fraktal_sft/fraktal_sft.rc . . . 456
68 fraktal_sft/gradient.cpp . . . 472
69 fraktal_sft/gradient.h . . . 477
70 fraktal_sft/icon1.ico . . . 478
71 fraktal_sft/icon.ico . . . 478
71.1 Layer 0 . . . 478
71.2 Layer 1 . . . 479
71.3 Layer 2 . . . 479
71.4 Layer 3 . . . 480
71.5 Layer 4 . . . 480
71.6 Layer 5 . . . 481
71.7 Layer 6 . . . 481
71.8 Layer 7 . . . 482
71.9 Layer 8 . . . 482
72 fraktal_sft/itercount_array.h . . . 482
73 fraktal_sft/jpeg.cpp . . . 484
74 fraktal_sft/jpeg.h . . . 491
75 fraktal_sft/listbox.cpp . . . 491
76 fraktal_sft/listbox.h . . . 500
77 fraktal_sft/long_double_perturbation.cpp . . . 501
78 fraktal_sft/long_double_reference.cpp . . . 505
79 fraktal_sft/main_color.cpp . . . 514
80 fraktal_sft/main_color.h . . . 538
81 fraktal_sft/main.cpp . . . 539
82 fraktal_sft/main_examine.cpp . . . 649
83 fraktal_sft/main_examine.h . . . 656
84 fraktal_sft/main.h . . . 656
85 fraktal_sft/main_iterations.cpp . . . 657
86 fraktal_sft/main_iterations.h . . . 667

87 fraktal_sft/main_position.cpp . . . 668
88 fraktal_sft/main_position.h . . . 670
89 fraktal_sft/nanomb1.h . . . 671
90 fraktal_sft/nanomb1.inc . . . 671
91 fraktal_sft/nanomb1_perturbation.cpp . . . 682
92 fraktal_sft/nanomb1_reference.cpp . . . 684
93 fraktal_sft/nanomb2.h . . . 684
94 fraktal_sft/nanomb2.inc . . . 685
95 fraktal_sft/nanomb2_perturbation.cpp . . . 698
96 fraktal_sft/nanomb2_reference.cpp . . . 699
97 fraktal_sft/newton.cpp . . . 700
98 fraktal_sft/newton.h . . . 722
99 fraktal_sft/Parameter.cpp . . . 723
100 fraktal_sft/png.cpp . . . 732
101 fraktal_sft/png.h . . . 735
102 fraktal_sft/render.cpp . . . 736
103 fraktal_sft/resource.h . . . 751
104 fraktal_sft/scale_bitmap.cpp . . . 757
105 fraktal_sft/scale_bitmap.h . . . 759
106 fraktal_sft/Settings.cpp . . . 759
107 fraktal_sft/Settings.h . . . 764
108 fraktal_sft/tiff.cpp . . . 769
109 fraktal_sft/tiff.h . . . 771
110 .gitignore . . . 771
111 LICENSE.md . . . 772
112 Makefile . . . 784
113 native.mk . . . 788
114 openexr-2.4.0.patch . . . 788
115 prepare.sh . . . 790
116 preprocessor.hs . . . 794
117 README.md . . . 802
118 release.sh . . . 847
119 utils/kf-tile.cpp . . . 848
120 utils/resizeKFB.m . . . 853
121 utils/stratify.m . . . 854

1 32.mk

```
WINPREFIXPLUS ?= $(HOME)/win32
WINPREFIX ?= $(HOME)/win32
SIMD ?= 0
COMPILE ?= i686-w64-mingw32-g++
LINK ?= i686-w64-mingw32-g++
WINDRES ?= i686-w64-mingw32-windres
AR ?= i686-w64-mingw32-ar
AR2 ?= i686-w64-mingw32-ranlib
XSLTPROC ?= xsltproc
RM ?= rm -f
GCC ?= i686-w64-mingw32-gcc
```

2 64.mk

```
WINPREFIXPLUS ?= $(HOME)/win64
WINPREFIX ?= $(HOME)/win64
SIMD ?= 2
COMPILE ?= x86_64-w64-mingw32-g++
LINK ?= x86_64-w64-mingw32-g++
WINDRES ?= x86_64-w64-mingw32-windres
AR ?= x86_64-w64-mingw32-ar
AR2 ?= x86_64-w64-mingw32-ranlib
XSLTPROC ?= xsltproc
RM ?= rm -f
GCC ?= x86_64-w64-mingw32-gcc
```

3 64+.mk

```
WINPREFIXPLUS ?= $(HOME)/win64+
WINPREFIX ?= $(HOME)/win64
SIMD ?= 2
# require 'haswell' (Intel Haswell), 'bdver4' (AMD Excavator), or 'eden-x4' (VIA
    ↘ )
COMPILE ?= x86_64-w64-mingw32-g++ -mmmx -msse -msse2 -msse3 -mssse3 -msse4.1 -
    ↘ msse4.2 -mavx -mavx2
LINK ?= x86_64-w64-mingw32-g++
WINDRES ?= x86_64-w64-mingw32-windres
AR ?= x86_64-w64-mingw32-ar
AR2 ?= x86_64-w64-mingw32-ranlib
XSLTPROC ?= xsltproc
RM ?= rm -f
GCC ?= x86_64-w64-mingw32-gcc
```

4 cl/common.cl

```
#pragma OPENCL EXTENSION cl_khr_fp64: enable

#define assert(n) do{}while(0)

double d_add(const double a, const double b)
{
  return a + b;
}

double d_sub(const double a, const double b)
{
  return a - b;
}

double d_neg(const double a)
{
  return -a;
}

double d_abs(const double a)
{
  return a < 0.0 ? -a : a;
}

double d_imul(const int a, const double b)
```

```
{
  return a * b;
}

double d_muli(const double a, const int b)
{
  return a * b;
}

double d_mul(const double a, const double b)
{
  return a * b;
}

double d_pow(const double a, const int b)
{
  double r = 1.0;
  for (int i = 0; i < b; ++i)
  {
    r *= a;
  }
  return r;
}

int d_cmp(const double a, const double b)
{
  return ((int)(a > b)) - ((int)(a < b));
}

double d_diffabs(const double c, const double d)
{
  int s = d_cmp(c, 0);
  if (s > 0)
  {
    int t = d_cmp(c + d, 0.0);
    if (t >= 0)
    {
      return d;
    }
    else
    {
      return -d - 2 * c;
    }
  }
  else if (s < 0)
  {
    int t = d_cmp(c + d, 0.0);
    if (t > 0)
    {
      return d + 2 * c;
    }
    else
    {
      return -d;
    }
  }
  return d_abs(d);
```

```
}

typedef struct
{
  double val;
  long exp;
} floatexp;

double fe_double(floatexp f)
{
  if (f.exp < -1020)
  {
    return 0.0;
  }
  if (f.exp > 1020)
  {
    return f.val / 0.0;
  }
  return as_double((as_long(f.val) & 0x800FFFFFFFFFFFFFL) | ((f.exp + 1023) << 52));
}

floatexp fe_floatexp(const double val, const long exp)
{
  long f_exp = ((as_long(val) & 0x7FF0000000000000L) >> 52) - 1023;
  double f_val = as_double((as_long(val) & 0x800FFFFFFFFFFFFFL) | 0x3FF0000000000000L);
  floatexp fe = { f_val, f_exp + exp };
  return fe;
}

floatexp fe_abs(const floatexp f)
{
  floatexp fe = { d_abs(f.val), f.exp };
  return fe;
}

floatexp fe_neg(const floatexp f)
{
  floatexp fe = { -f.val, f.exp };
  return fe;
}

floatexp fe_sqr(const floatexp a)
{
  return fe_floatexp(a.val * a.val, a.exp << 1);
}


floatexp fe_muld(const floatexp a, const double b)
{
  return fe_floatexp(a.val * b, a.exp);
}

floatexp fe_dmul(const double a, const floatexp b)
{
  return fe_floatexp(a * b.val, b.exp);
```

```
}

floatexp fe_muli(const floatexp a, const int b)
{
  return fe_floatexp(a.val * b, a.exp);
}

floatexp fe_imul(const int a, const floatexp b)
{
  return fe_floatexp(a * b.val, b.exp);
}

floatexp fe_mul(const floatexp a, const floatexp b)
{
  return fe_floatexp(a.val * b.val, a.exp + b.exp);
}

floatexp fe_mul_2si(const floatexp a, const long b)
{
  floatexp fe = { a.val, a.exp + b };
  return fe;
}

floatexp fe_div(const floatexp a, const floatexp b)
{
  return fe_floatexp(a.val / b.val, a.exp - b.exp);
}

floatexp fe_div_2si(const floatexp a, const long b)
{
  floatexp fe = { a.val, a.exp - b };
  return fe;
}

floatexp fe_add(const floatexp a, const floatexp b)
{
  if (a.exp > b.exp)
  {
    floatexp c = { b.val, b.exp - a.exp };
    return fe_floatexp(a.val + fe_double(c), a.exp);
  }
  else
  {
    floatexp c = { a.val, a.exp - b.exp };
    return fe_floatexp(fe_double(c) + b.val, b.exp);
  }
}

floatexp fe_sub(const floatexp a, const floatexp b)
{
  return fe_add(a, fe_neg(b));
}

int fe_cmp(const floatexp a, const floatexp b)
{
  if (a.val > 0)
  {
```

```
195       if (b.val <= 0)
          {
            return 1;
          }
          else if (a.exp > b.exp)
200       {
            return 1;
          }
          else if (a.exp < b.exp)
          {
205         return -1;
          }
          else
          {
            return d_cmp(a.val, b.val);
210       }
        }
        else
        {
          if (b.val > 0)
215       {
            return -1;
          }
          else if (a.exp > b.exp)
          {
220         return -1;
          }
          else if (a.exp < b.exp)
          {
            return 1;
225       }
          else
          {
            return d_cmp(a.val, b.val);
          }
230     }
      }

      floatexp fe_diffabs(const floatexp c, const floatexp d)
      {
235     int s = d_cmp(c.val, 0.0);
        if (s > 0)
        {
          int t = d_cmp(fe_add(c, d).val, 0.0);
          if (t >= 0)
240       {
            return d;
          }
          else
          {
245         return fe_neg(fe_add(d, fe_mul_2si(c, 1)));
          }
        }
        else if (s < 0)
        {
250       int t = d_cmp(fe_add(c, d).val, 0.0);
          if (t > 0)
```

```
      {
        return fe_add(d, fe_mul_2si(c, 1));
      }
      else
      {
        return fe_neg(d);
      }
    }
    return fe_abs(d);
}

floatexp fe_pow(const floatexp a, int b)
{
  floatexp r = fe_floatexp(1.0, 0);
  for (int i = 0; i < b; ++i)
  {
    r = fe_mul(r, a);
  }
  return r;
}

typedef struct {
  uint se;
  uint m;
} softfloat;

#define SF_EXPONENT_BIAS ((1U << 30U) - 1U)

#define SF_MANTISSA_BITS 32

bool sf_sign_bit(const softfloat f)
{
  return !!(f.se & 0x80000000U);
}

uint sf_biased_exponent(const softfloat f)
{
  return f.se & 0x7FFFFFFFU;
}

uint sf_mantissa(const softfloat f)
{
  return f.m;
}

bool sf_is_zero(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0 &&
    sf_mantissa(f) == 0;
}

bool sf_is_denormal(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0 &&
    sf_mantissa(f) != 0;
```

```
}

bool sf_is_inf(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0x7FFFFFFFU &&
    sf_mantissa(f) == 0;
}

bool sf_is_nan(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0x7FFFFFFFU &&
    sf_mantissa(f) != 0;
}

bool sf_lt(const softfloat a, const softfloat b)
{
  if (sf_is_nan(a) || sf_is_nan(b))
  {
    return false;
  }
  else if (sf_sign_bit(a) && ! sf_sign_bit(b))
  {
    return true;
  }
  else if (! sf_sign_bit(a) && sf_sign_bit(b))
  {
    return false;
  }
  else if (sf_biased_exponent(a) > sf_biased_exponent(b))
  {
    return sf_sign_bit(a);
  }
  else if (sf_biased_exponent(a) < sf_biased_exponent(b))
  {
    return ! sf_sign_bit(a);
  }
  else if (sf_mantissa(a) > sf_mantissa(b))
  {
    return sf_sign_bit(a);
  }
  else if (sf_mantissa(a) < sf_mantissa(b))
  {
    return ! sf_sign_bit(a);
  }
  else
  {
    // equal
    return false;
  }
}

bool sf_gt(const softfloat a, const softfloat b)
{
  return sf_lt(b, a);
}
```

```
float sf_to_float(const softfloat f)
{
  if (sf_is_zero(f) || sf_is_denormal(f))
  {
    if (sf_sign_bit(f)) return -0.0f; else return 0.0f;
  }
  else if (sf_is_inf(f))
  {
    if (sf_sign_bit(f)) return -1.0f/0.0f; else return 1.0f/0.0f;
  }
  else if (sf_is_nan(f))
  {
    if (sf_sign_bit(f)) return -(0.0f/0.0f); else return 0.0f/0.0f;
  }
  else
  {
    float x = sf_mantissa(f);
    int e
      = convert_int_sat((long)(sf_biased_exponent(f))
      - (SF_EXPONENT_BIAS + SF_MANTISSA_BITS));
    if (sf_sign_bit(f)) return -ldexp(x, e); else return ldexp(x, e);
  }
}

double sf_to_double(const softfloat f)
{
  if (sf_is_zero(f) || sf_is_denormal(f))
  {
    if (sf_sign_bit(f)) return -0.0; else return 0.0;
  }
  else if (sf_is_inf(f))
  {
    if (sf_sign_bit(f)) return -1.0/0.0; else return 1.0/0.0;
  }
  else if (sf_is_nan(f))
  {
    if (sf_sign_bit(f)) return -(0.0/0.0); else return 0.0/0.0;
  }
  else
  {
    double x = sf_mantissa(f);
    int e
      = convert_int_sat((long)(sf_biased_exponent(f))
      - (SF_EXPONENT_BIAS + SF_MANTISSA_BITS));
    if (sf_sign_bit(f)) return -ldexp(x, e); else return ldexp(x, e);
  }
}

softfloat sf_from_float(const float x)
{
  if (isnan(x))
  {
    softfloat f = { ((uint)(!!signbit(x)) << 31) | 0x7FFFFFFFU, 0xFFFFFFFFU };
    return f;
  }
  else if (isinf(x))
```

```
  {
    softfloat f = { ((uint)(!!signbit(x)) << 31) | 0x7FFFFFFFU, 0U };
    return f;
  }
  else if (x == 0.0f)
  {
    softfloat f = { ((uint)(!!signbit(x)) << 31) | 0U, 0U };
    return f;
  }
  else
  {
    int e;
    float y = frexp(fabs(x), &e);
    float z = ldexp(y, SF_MANTISSA_BITS);
    uint mantissa = convert_uint_rtz(z);
    uint biased_e = convert_uint_sat(e + SF_EXPONENT_BIAS);
    assert(0 < biased_e);
    assert(biased_e < 0x7FFFFFFFU);
    assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
    softfloat f = { ((uint)(!!signbit(x)) << 31) | biased_e, mantissa };
    return f;
  }
}

softfloat sf_from_double(const double x)
{
  if (isnan(x))
  {
    softfloat f = { ((uint)(!!signbit(x)) << 31) | 0x7FFFFFFFU, 0xFFFFFFFFU };
    return f;
  }
  else if (isinf(x))
  {
    softfloat f = { ((uint)(!!signbit(x)) << 31) | 0x7FFFFFFFU, 0U };
    return f;
  }
  else if (x == 0.0)
  {
    softfloat f = { ((uint)(!!signbit(x)) << 31) | 0U, 0U };
    return f;
  }
  else
  {
    int e;
    double y = frexp(fabs(x), &e);
    double z = ldexp(y, SF_MANTISSA_BITS);
    uint mantissa = convert_uint_rtz(z); // rte might overflow rarely
    uint biased_e = convert_uint_sat(e + SF_EXPONENT_BIAS);
    assert(0 < biased_e);
    assert(biased_e < 0x7FFFFFFFU);
    assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
    softfloat f = { ((uint)(!!signbit(x)) << 31) | biased_e, mantissa };
    return f;
  }
}

softfloat sf_ldexp(const softfloat a, int e)
```

```
480 {
      if (sf_is_zero(a) || sf_is_inf(a) || sf_is_nan(a))
      {
        return a;
      }
485   else if (e >= (int)(0x7FFFFFFFU - sf_biased_exponent(a)))
      {
        // overflow to +/-infinity
        softfloat o = { (a.se & 0x80000000U) | 0x7FFFFFFFU, 0U };
        return o;
490   }
      else if ((int)(sf_biased_exponent(a)) + e <= 0)
      {
        // underfloat to 0
        softfloat o = { (a.se & 0x80000000U) | 0U, 0U };
495     return o;
      }
      else
      {
        softfloat o = { (a.se & 0x80000000U) | (sf_biased_exponent(a) + e), ↙
            ↳ sf_mantissa(a) };
500     return o;
      }
    }

    softfloat sf_from_floatexp(const floatexp x)
505 {
      return sf_ldexp(sf_from_double(x.val), convert_int_sat(x.exp));
    }

    softfloat sf_zero()
510 {
      softfloat o = { 0, 0 };
      return o;
    }

515 softfloat sf_one()
    {
      return sf_from_float(1.0f);
    }

520 softfloat sf_abs(const softfloat a)
    {
      softfloat o = { a.se & 0x7FFFFFFFU, a.m };
      return o;
    }
525
    softfloat sf_neg(const softfloat a)
    {
      softfloat o = { a.se ^ 0x80000000U, a.m };
      return o;
530 }

    softfloat sf_sqr(const softfloat a)
    {
      if (sf_biased_exponent(a) >= ((0x7FFFFFFFU >> 1) + (SF_EXPONENT_BIAS >> 1)))
535   {
```

```
      // overflow to +infinity
      softfloat o = { 0x7FFFFFFFU, sf_is_nan(a) ? 0xFFFFFFFFU : 0U };
      return o;
    }
    else if (sf_biased_exponent(a) <= (SF_EXPONENT_BIAS >> 1) + 1)
    {
      // underflow to +0
      // FIXME handle denormals
      softfloat o = { 0U, 0U };
      return o;
    }
    else
    {
      ulong m = a.m;
      uint mantissa = (m * m) >> SF_MANTISSA_BITS;
      uint biased_e = ((a.se & 0x7FFFFFFFU) << 1) - SF_EXPONENT_BIAS;
      if ((mantissa & 0x80000000U) == 0)
      {
        mantissa <<= 1;
        biased_e -= 1;
      }
      assert(0 < biased_e);
      assert(biased_e < 0x7FFFFFFFU);
      assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
      softfloat o = { biased_e, mantissa };
      return o;
    }
  }

  softfloat sf_mul(const softfloat a, const softfloat b)
  {
    if ( sf_is_nan(a) ||
         sf_is_nan(b) ||
         (sf_is_inf(a) && sf_is_zero(b)) ||
         (sf_is_zero(a) && sf_is_inf(b))
       )
    {
      // nan
      softfloat o = { ((a.se ^ b.se) & 0x80000000U) | 0x7FFFFFFFU, 0xFFFFFFFFU };
      return o;
    }
    else if (sf_is_zero(a) || sf_is_zero(b))
    {
      // zero
      softfloat o = { ((a.se ^ b.se) & 0x80000000U) | 0U, 0U };
      return o;
    }
    else if ((sf_biased_exponent(a) + sf_biased_exponent(b)) >= (0x7FFFFFFFU + ↙
        ↳ SF_EXPONENT_BIAS))
    {
      // overflow to +/-infinity
      softfloat o = { ((a.se ^ b.se) & 0x80000000U) | 0x7FFFFFFFU, 0U };
      return o;
    }
    else if ((sf_biased_exponent(a) + sf_biased_exponent(b)) <= (SF_EXPONENT_BIAS ↙
        ↳ + 1))
    {
```

```
    // underflow to +/-0
    // FIXME handle denormals
    softfloat o = { ((a.se ^ b.se) & 0x80000000U) | 0U, 0U };
    return o;
  }
  else
  {
    ulong ma = a.m;
    ulong mb = b.m;
    uint mantissa = (ma * mb) >> SF_MANTISSA_BITS;
    uint biased_e = ((a.se & 0x7FFFFFFFU) + (b.se & 0x7FFFFFFFU)) - SF_EXPONENT_BIAS;
    if ((mantissa & 0x80000000U) == 0)
    {
      mantissa <<= 1;
      biased_e -= 1;
    }
    assert(0 < biased_e);
    assert(biased_e < 0x7FFFFFFFU);
    assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
    softfloat o = { ((a.se ^ b.se) & 0x80000000U) | biased_e, mantissa };
    return o;
  }
}

softfloat sf_muli(const softfloat a, const int b)
{
  return sf_mul(a, sf_from_float(b));
}

softfloat sf_imul(const int a, const softfloat b)
{
  return sf_mul(sf_from_float(a), b);
}

softfloat sf_add_a_gt_b_gt_0(const softfloat a, const softfloat b)
{
  // same sign addition, |a| > |b| or same exponent
  uint ea = sf_biased_exponent(a);
  uint eb = sf_biased_exponent(b);
  ulong ma = sf_mantissa(a);
  ulong mb = sf_mantissa(b);
  assert(ea >= eb);
  assert(sf_sign_bit(a) == sf_sign_bit(b));
  ulong mantissa = ma + (mb >> (ea - eb));
  uint biased_e = ea;
  if (!! (mantissa & 0x100000000LLU))
  {
    biased_e += 1;
    mantissa >>= 1;
  }
  if (biased_e >= 0x7FFFFFFFU)
  {
    // overflow to +/-infinity
    softfloat o = { (a.se & 0x80000000U) | 0x7FFFFFFFU, 0U };
    return o;
  }
```

```
  assert(0 < biased_e);
  assert(biased_e < 0x7FFFFFFFU);
  assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
  assert((mantissa >> SF_MANTISSA_BITS) == 0U);
  softfloat o = { biased_e, mantissa };
  if (sf_sign_bit(a)) return sf_neg(o); else return o;
}

softfloat sf_add_a_gt_0_gt_b(const softfloat a, const softfloat b)
{
  // opposite sign addition, a > 0 > b, |a| > |b|
  uint ea = sf_biased_exponent(a);
  uint eb = sf_biased_exponent(b);
  ulong ma = sf_mantissa(a);
  ulong mb = sf_mantissa(b);
  assert(ea > eb);
  assert(! sf_sign_bit(a));
  assert(  sf_sign_bit(b));
  // a > 0 > b, |a| > |b|
  long smantissa = (ma << 1) - ((mb << 1) >> (ea - eb));
  assert(smantissa > 0);
  ulong mantissa = smantissa;
  long biased_e = ea - 1;
  int shift = ((int)(clz(mantissa))) - SF_MANTISSA_BITS;
  if (shift > 0)
  {
    mantissa <<= shift;
    biased_e -= shift;
  }
  else if (shift < 0)
  {
    mantissa >>= -shift;
    biased_e += -shift;
  }
  if (biased_e >= 0x7FFFFFFFU)
  {
    // overflow to +infinity, impossible?
    softfloat o = { 0x7FFFFFFFU, 0U };
    return o;
  }
  else if (biased_e <= 0)
  {
    // underflow to +0
    softfloat o = { 0U, 0U };
    return o;
  }
  assert(0 < biased_e);
  assert(biased_e < 0x7FFFFFFFU);
  assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
  assert((mantissa >> SF_MANTISSA_BITS) == 0U);
  softfloat o = { biased_e, (uint)(mantissa) };
  return o;
}

softfloat sf_add(const softfloat a, const softfloat b)
{
  if ( sf_is_nan(a) ||
```

```
        sf_is_nan(b) ||
705     (sf_is_inf(a) && sf_is_inf(b) && !!((a.se ^ b.se) & 0x80000000U))
      )
  {
    // nan
    softfloat o = { 0x7FFFFFFFU, 0xFFFFFFFFU };
710 return o;
  }
  else if (sf_is_zero(a))
  {
    return b;
715 }
  else if (sf_is_zero(b))
  {
    return a;
  }
720 else if (sf_is_inf(a))
  {
    return a;
  }
  else if (sf_is_inf(b))
725 {
    return b;
  }
  else if (((a.se ^ b.se) & 0x80000000U) == 0)
  {
730 // same sign addition
    uint ea = sf_biased_exponent(a);
    uint eb = sf_biased_exponent(b);
    if (ea > eb + SF_MANTISSA_BITS)
    {
735   return a;
    }
    else if (eb > ea + SF_MANTISSA_BITS)
    {
      return b;
740 }
    else if (ea >= eb)
    {
      return sf_add_a_gt_b_gt_0(a, b);
    }
745 else
    {
      return sf_add_a_gt_b_gt_0(b, a);
    }
  }
750 else
  {
    // opposite sign addition
    uint ea = sf_biased_exponent(a);
    uint eb = sf_biased_exponent(b);
755 if (ea > eb + SF_MANTISSA_BITS)
    {
      return a;
    }
    else if (eb > ea + SF_MANTISSA_BITS)
760 {
```

```
      return b;
    }
    else if (ea == eb)
    {
765   uint ma = sf_mantissa(a);
      uint mb = sf_mantissa(b);
      if (ma > mb)
      {
        uint mantissa = ma - mb;
770     uint shift = clz(mantissa);
        mantissa <<= shift;
        if (ea > shift)
        {
          uint biased_e = ea - shift;
775       assert(0 < biased_e);
          assert(biased_e < 0x7FFFFFFFU);
          assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
          softfloat o = { biased_e, mantissa };
          if (sf_sign_bit(a)) return sf_neg(o); else return o;
780     }
        else
        {
          // FIXME handle denormals
          softfloat o = { 0U, 0U };
785       return o;
        }
      }
      else if (mb > ma)
      {
790     uint mantissa = mb - ma;
        uint shift = clz(mantissa);
        mantissa <<= shift;
        if (eb > shift)
        {
795       uint biased_e = eb - shift;
          assert(0 < biased_e);
          assert(biased_e < 0x7FFFFFFFU);
          assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
          softfloat o = { biased_e, mantissa };
800       if (sf_sign_bit(b)) return sf_neg(o); else return o;
        }
        else
        {
          // FIXME handle denormals
805       softfloat o = { 0U, 0U };
          return o;
        }
      }
      else
810   {
        // cancels to 0
        softfloat o = { 0U, 0U };
        return o;
      }
815 }
    else if (ea > eb)
    {
```

```
      // |a| > |b|
      if (sf_sign_bit(a))
      {
        return sf_neg(sf_add_a_gt_0_gt_b(sf_neg(a), sf_neg(b)));
      }
      else
      {
        return sf_add_a_gt_0_gt_b(a, b);
      }
    }
    else
    {
      // |b| > |a|
      if (sf_sign_bit(b))
      {
        return sf_neg(sf_add_a_gt_0_gt_b(sf_neg(b), sf_neg(a)));
      }
      else
      {
        return sf_add_a_gt_0_gt_b(b, a);
      }
    }
  }
}

softfloat sf_sub(const softfloat a, const softfloat b)
{
  return sf_add(a, sf_neg(b));
}

typedef struct
{
  double re;
  double im;
} dcomplex;

double dc_norm(const dcomplex a)
{
  return a.re * a.re + a.im * a.im;
}

double dc_abs(const dcomplex a)
{
  return sqrt(dc_norm(a));
}

dcomplex dc_neg(const dcomplex a)
{
  dcomplex dc = { -a.re, -a.im };
  return dc;
}

dcomplex dc_add(const dcomplex a, const dcomplex b)
{
  dcomplex dc = { a.re + b.re, a.im + b.im };
  return dc;
}
```

```
875
dcomplex dc_sub(const dcomplex a, const dcomplex b)
{
  dcomplex dc = { a.re - b.re, a.im - b.im };
  return dc;
880 }

dcomplex dc_imul(const int a, const dcomplex b)
{
  dcomplex dc = { a * b.re, a * b.im };
885   return dc;
}

dcomplex dc_muli(const dcomplex a, const int b)
{
890   dcomplex dc = { a.re * b, a.im * b };
  return dc;
}

dcomplex dc_mul(const dcomplex a, const dcomplex b)
895 {
  dcomplex dc = { a.re * b.re - a.im * b.im, a.re * b.im + a.im * b.re };
  return dc;
}

900 dcomplex dc_ddiv(const double a, const dcomplex b)
{
  double b2 = dc_norm(b);
  dcomplex dc = { (a * b.re) / b2, (-a * b.im) / b2 };
  return dc;
905 }

dcomplex dc_div(const dcomplex a, const dcomplex b)
{
  double b2 = dc_norm(b);
910   dcomplex dc = { (a.re * b.re + a.im * b.im) / b2, (-a.re * b.im + a.im * b.re)↙
      ↪  / b2 };
  return dc;
}

dcomplex dc_pow(const dcomplex a, const int b)
915 {
  dcomplex r = { 1.0, 0.0 };
  for (int i = 0; i < b; ++i)
  {
    r = dc_mul(r, a);
920   }
  return r;
}

typedef struct
925 {
  floatexp re;
  floatexp im;
} fecomplex;

930 fecomplex fec_neg(const fecomplex a)
```

```
{
  fecomplex fec = { fe_neg(a.re), fe_neg(a.im) };
  return fec;
}

fecomplex fec_add(const fecomplex a, const fecomplex b)
{
  fecomplex fec = { fe_add(a.re, b.re), fe_add(a.im, b.im) };
  return fec;
}

fecomplex fec_sub(const fecomplex a, const fecomplex b)
{
  fecomplex fec = { fe_sub(a.re, b.re), fe_sub(a.im, b.im) };
  return fec;
}

fecomplex fec_imul(const int a, const fecomplex b)
{
  fecomplex fec = { fe_imul(a, b.re), fe_imul(a, b.im) };
  return fec;
}

fecomplex fec_muli(const fecomplex a, const int b)
{
  fecomplex fec = { fe_muli(a.re, b), fe_muli(a.im, b) };
  return fec;
}

fecomplex fec_mul(const fecomplex a, const fecomplex b)
{
  fecomplex fec = { fe_sub(fe_mul(a.re, b.re), fe_mul(a.im, b.im)), fe_add(↙
      ↳ fe_mul(a.re, b.im), fe_mul(a.im, b.re)) };
  return fec;
}

fecomplex fec_pow(const fecomplex a, const int b)
{
  fecomplex r = { fe_floatexp(1.0, 0), fe_floatexp(0.0, 0) };
  for (int i = 0; i < b; ++i)
  {
    r = fec_mul(r, a);
  }
  return r;
}

typedef struct
{
  softfloat re;
  softfloat im;
} sfcomplex;

sfcomplex sfc_neg(const sfcomplex a)
{
  sfcomplex sfc = { sf_neg(a.re), sf_neg(a.im) };
  return sfc;
}
```

```
sfcomplex sfc_add(const sfcomplex a, const sfcomplex b)
{
  sfcomplex sfc = { sf_add(a.re, b.re), sf_add(a.im, b.im) };
  return sfc;
}

sfcomplex sfc_sub(const sfcomplex a, const sfcomplex b)
{
  sfcomplex sfc = { sf_sub(a.re, b.re), sf_sub(a.im, b.im) };
  return sfc;
}

sfcomplex sfc_imul(const int a, const sfcomplex b)
{
  sfcomplex sfc = { sf_imul(a, b.re), sf_imul(a, b.im) };
  return sfc;
}

sfcomplex sfc_muli(const sfcomplex a, const int b)
{
  sfcomplex sfc = { sf_muli(a.re, b), sf_muli(a.im, b) };
  return sfc;
}

sfcomplex sfc_mul(const sfcomplex a, const sfcomplex b)
{
  sfcomplex sfc = { sf_sub(sf_mul(a.re, b.re), sf_mul(a.im, b.im)), sf_add(
      ↪ sf_mul(a.re, b.im), sf_mul(a.im, b.re)) };
  return sfc;
}

sfcomplex sfc_pow(const sfcomplex a, const int b)
{
  sfcomplex r = { sf_one(), sf_zero() };
  for (int i = 0; i < b; ++i)
  {
    r = sfc_mul(r, a);
  }
  return r;
}

typedef struct
{
  double cr;
  double ci;
  double xr;
  double xi;
  double dr;
  double di;
  double daa;
  double dab;
  double dba;
  double dbb;
  double test1;
  double test2;
  long antal;
```

```
  int bGlitch;
} p_status_d;

typedef struct
{
  floatexp cr;
  floatexp ci;
  floatexp xr;
  floatexp xi;
  floatexp dr;
  floatexp di;
  floatexp daa;
  floatexp dab;
  floatexp dba;
  floatexp dbb;
  double test1;
  double test2;
  long antal;
  int bGlitch;
} p_status_fe;

typedef struct
{
  softfloat cr;
  softfloat ci;
  softfloat xr;
  softfloat xi;
  softfloat dr;
  softfloat di;
  softfloat daa;
  softfloat dab;
  softfloat dba;
  softfloat dbb;
  softfloat test1;
  softfloat test2;
  long antal;
  int bGlitch;
} p_status_sf;


typedef struct __attribute__((packed))
{
  long BYTES;
  // for pixel -> parameter mapping
  int m_nX;
  int m_nY;
  uint JitterSeed;
  int JitterShape;
  double JitterScale;
  floatexp m_pixel_step_x;
  floatexp m_pixel_step_y;
  floatexp m_pixel_center_x;
  floatexp m_pixel_center_y;
  double m_C;
  double m_S;
  // for result -> output mapping
  long stride_y;
```

```
1100    long stride_x;
        long stride_offset;
        // for iteration control
        double m_nBailout;
        double m_nBailout2;
1105    double log_m_nPower;
        long m_nGlitchIter;
        long m_nMaxIter;
        long nMaxIter;
        long nMinIter;
1110    short m_bNoGlitchDetection;
        short derivatives;
        short m_bAddReference;
        short m_nSmoothMethod;
        double g_real;
1115    double g_imag;
        double g_FactorAR;
        double g_FactorAI;
        double m_epsilon;
        // for series approximation
1120    double m_dPixelSpacing;
        floatexp m_fPixelSpacing;
        long m_nMaxApproximation;
        int m_nApproxTerms;
        int approximation_type;
1125    // 130kB data follows
        floatexp m_APr[MAX_APPROX_TERMS + 1];
        floatexp m_APi[MAX_APPROX_TERMS + 1];
        floatexp m_APs_s[MAX_APPROX_TERMS + 1][MAX_APPROX_TERMS + 1];
        floatexp m_APs_t[MAX_APPROX_TERMS + 1][MAX_APPROX_TERMS + 1];
1130  } p_config;

      uint burtle_hash(uint a)
      {
          a = (a+0x7ed55d16) + (a<<12);
1135      a = (a^0xc761c23c) ^ (a>>19);
          a = (a+0x165667b1) + (a<<5);
          a = (a+0xd3a2646c) ^ (a<<9);
          a = (a+0xfd7046c5) + (a<<3);
          a = (a^0xb55a4f09) ^ (a>>16);
1140      return a;
      }

      // uniform in [0,1)
      double dither(uint x, uint y, uint c)
1145  {
        return
          burtle_hash(x +
          burtle_hash(y +
          burtle_hash(c))) / (double) (0x100000000LL);
1150  }

      void GetPixelOffset
      ( __global const p_config *g
      , const int i
1155  , const int j
      , double *x
```

```
, double *y
)
{
  uint c = g->JitterSeed;
  if (c != 0)
  {
    double s = g->JitterScale;
    double u = dither(i, j, 2 * c + 0);
    double v = dither(i, j, 2 * c + 1);
    switch (g->JitterShape)
    {
      default:
      case 0: // uniform
        {
          *x = s * (u - 0.5);
          *y = s * (v - 0.5);
        }
        break;
      case 1: // Gaussian
        {
          // FIXME cache slow trig functions for every pixel for every image?
          // https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform
          double r = 0 < u && u < 1 ? sqrt(-2 * log(u)) : 0;
          double t = 2 * 3.141592653589793 * v;
          s *= 0.5;
          *x = s * r * cos(t);
          *y = s * r * sin(t);
        }
        break;
    }
  }
  else
  {
    *x = 0.0;
    *y = 0.0;
  }
}

void GetPixelCoordinates
( __global const p_config *g
, const int i
, const int j
, floatexp *x
, floatexp *y
)
{
  double di = 0;
  double dj = 0;
  GetPixelOffset(g, i, j, &di, &dj);
  floatexp u = fe_muld(g->m_pixel_step_x, i - g->m_nX/2 + di);
  floatexp v = fe_muld(g->m_pixel_step_y, j - g->m_nY/2 + dj);
  *x = fe_add(g->m_pixel_center_x, fe_add(fe_muld(u, g->m_C), fe_muld(v, g->m_S)↙
      ↳ ));
  *y = fe_sub(g->m_pixel_center_y, fe_sub(fe_muld(u, g->m_S), fe_muld(v, g->m_C)↙
      ↳ ));
}
```

```
void GetPixelCoordinatesM
( __global const p_config *g
, const int i
, const int j
, floatexp *x
, floatexp *y
, floatexp *daa
, floatexp *dab
, floatexp *dba
, floatexp *dbb
)
{
  double di = 0;
  double dj = 0;
  GetPixelOffset(g, i, j, &di, &dj);
  floatexp u = fe_muld(g->m_pixel_step_x, i - g->m_nX/2 + di);
  floatexp v = fe_muld(g->m_pixel_step_y, j - g->m_nY/2 + dj);
  *x = fe_add(g->m_pixel_center_x, fe_add(fe_muld(u, g->m_C), fe_muld(v, g->m_S)
      ↳ ));
  *y = fe_sub(g->m_pixel_center_y, fe_sub(fe_muld(u, g->m_S), fe_muld(v, g->m_C)
      ↳ ));
  *daa = fe_muld(g->m_pixel_step_x,  g->m_C);
  *dab = fe_muld(g->m_pixel_step_y,  g->m_S);
  *dba = fe_muld(g->m_pixel_step_x, -g->m_S);
  *dbb = fe_muld(g->m_pixel_step_y,  g->m_C);
}

void DoApproximationD
( __global const p_config *g
, long *antal
, const floatexp D0r
, const floatexp D0i
, floatexp *oTDnr
, floatexp *oTDni
, floatexp *oTDDnr
, floatexp *oTDDni
)
{
  const floatexp zero = fe_floatexp(0.0, 0);
  if (g->m_nMaxApproximation)
  {
    floatexp TDnr = zero;
    floatexp TDni = zero;
    floatexp TDDnr = zero;
    floatexp TDDni = zero;
    for (int k = g->m_nApproxTerms - 1; k >= 0; --k)
    {
      floatexp tr = fe_add(fe_sub(fe_mul(TDnr, D0r), fe_mul(TDni, D0i)), g->
          ↳ m_APr[k]);
      floatexp ti = fe_add(fe_add(fe_mul(TDnr, D0i), fe_mul(TDni, D0r)), g->
          ↳ m_APi[k]);
      TDnr = tr;
      TDni = ti;
      tr = fe_add(fe_sub(fe_mul(TDDnr, D0r), fe_mul(TDDni, D0i)), fe_muli(g->
          ↳ m_APr[k], k + 1));
      ti = fe_add(fe_add(fe_mul(TDDnr, D0i), fe_mul(TDDni, D0r)), fe_muli(g->
          ↳ m_APi[k], k + 1));
```

```
      TDDnr = tr;
      TDDni = ti;
    }
    floatexp tr = fe_sub(fe_mul(TDnr, D0r), fe_mul(TDni, D0i));
    floatexp ti = fe_add(fe_mul(TDnr, D0i), fe_mul(TDni, D0r));
    TDnr = tr;
    TDni = ti;
    *antal = g->m_nMaxApproximation - 1;
    *oTDnr = TDnr;
    *oTDni = TDni;
    *oTDDnr = TDnr;
    *oTDDni = TDni;
  }
  else
  {
    *antal = 0;
    *oTDnr = D0r;
    *oTDni = D0i;
    *oTDDnr = fe_floatexp(1.0, 0);
    *oTDDni = zero;
  }
}

void DoApproximationM
( __global const p_config *g
, long *antal
, const floatexp a
, const floatexp b
, floatexp *x
, floatexp *y
, floatexp *dxa
, floatexp *dxb
, floatexp *dya
, floatexp *dyb
)
{
    const floatexp zero = fe_floatexp(0.0, 0);
    const floatexp one  = fe_floatexp(1.0, 0);
    if (g->approximation_type == 1)
    {
      floatexp dx, dy;
      DoApproximationD(g, antal, a, b, x, y, &dx, &dy);
      // Cauchy-Riemann
      *dxa = dx;
      *dxb = fe_neg(dy);
      *dya = dy;
      *dyb = dx;
    }
    else if (g->approximation_type == 2)
    {
      if (g->m_nMaxApproximation)
      {
        floatexp an[MAX_APPROX_TERMS + 1];
        floatexp bn[MAX_APPROX_TERMS + 1];
        floatexp dan[MAX_APPROX_TERMS + 1];
        floatexp dbn[MAX_APPROX_TERMS + 1];
        an[0] = one;
```

```
1320            bn[0] = one;
                dan[0] = zero;
                dbn[0] = zero;
                for (int i = 1; i <= g->m_nApproxTerms; ++i)
                {
1325              an[i] = fe_mul(an[i - 1], a);
                  bn[i] = fe_mul(bn[i - 1], b);
                  dan[i] = fe_muli(an[i - 1], i);
                  dbn[i] = fe_muli(bn[i - 1], i);
                }
1330            // x   = sum ApproxSeriesR2X[i][j]   a^i       b^j
                // dxa = sum ApproxSeriesR2X[i][j] i a^{i-1}   b^j
                // dxb = sum ApproxSeriesR2X[i][j]   a^i     j b^{j-1}
                floatexp totalX = zero;
                floatexp totalY = zero;
1335            floatexp totalDXA = zero;
                floatexp totalDXB = zero;
                floatexp totalDYA = zero;
                floatexp totalDYB = zero;
                for (int i = 0; i <= g->m_nApproxTerms; ++i)
1340            {
                  floatexp subtotalX = zero;
                  floatexp subtotalY = zero;
                  floatexp subtotalDXB = zero;
                  floatexp subtotalDYB = zero;
1345              for (int j = 0; j <= g->m_nApproxTerms - i; ++j)
                  {
                    subtotalX   = fe_add(subtotalX,   fe_mul(g->m_APs_s[i][j],  bn[j]));
                    subtotalY   = fe_add(subtotalY,   fe_mul(g->m_APs_t[i][j],  bn[j]));
                    subtotalDXB = fe_add(subtotalDXB, fe_mul(g->m_APs_s[i][j], dbn[j]));
1350                subtotalDYB = fe_add(subtotalDYB, fe_mul(g->m_APs_t[i][j], dbn[j]));
                  }
                  totalX   = fe_add(totalX,   fe_mul(subtotalX  , an[i]));
                  totalY   = fe_add(totalY,   fe_mul(subtotalY  , an[i]));
                  totalDXA = fe_add(totalDXA, fe_mul(subtotalX  , dan[i]));
1355              totalDYA = fe_add(totalDYA, fe_mul(subtotalY  , dan[i]));
                  totalDXB = fe_add(totalDXB, fe_mul(subtotalDXB, an[i]));
                  totalDYB = fe_add(totalDYB, fe_mul(subtotalDYB, an[i]));
                }
                *antal = g->m_nMaxApproximation - 1;
1360            *x   = totalX;
                *y   = totalY;
                *dxa = totalDXA;
                *dxb = totalDXB;
                *dya = totalDYA;
1365            *dyb = totalDYB;
              }
              else
              {
                *antal = 0;
1370            *x = a;
                *y = b;
                *dxa = one;
                *dxb = zero;
                *dya = zero;
1375            *dyb = one;
              }
```

```
    }
    else
    {
      *antal = 0;
      *x = a;
      *y = b;
      *dxa = one;
      *dxb = zero;
      *dya = zero;
      *dyb = one;
    }
}

#if 0
void DoApproximation
( __global const p_config *g
, const floatexp a
, const floatexp b
, floatexp *x
, floatexp *y
)
{
  floatexp an[MAX_APPROX_TERMS + 1];
  floatexp bn[MAX_APPROX_TERMS + 1];
  an[0] = 1;
  bn[0] = 1;
  for (int i = 1; i <= g->m_nApproxTerms; ++i)
  {
    an[i] = an[i - 1] * a;
    bn[i] = bn[i - 1] * b;
  }
  floatexp totalX = 0;
  floatexp totalY = 0;
  for (int i = 0; i <= g->m_nApproxTerms; ++i)
  {
    floatexp subtotalX = 0;
    floatexp subtotalY = 0;
    for (int j = 0; j <= g->m_nApproxTerms - i; ++j)
    {
      subtotalX += g->m_APs_s[i][j] * bn[j];
      subtotalY += g->m_APs_t[i][j] * bn[j];
    }
    totalX += subtotalX * an[i];
    totalY += subtotalY * an[i];
  }
  x = totalX;
  y = totalY;
}
#endif

#define PIXEL_UNEVALUATED (-2147483648LL)
#define ISFLOATOK(x) ((! isnan(x)) && (! isinf(x)))
#define SET_TRANS_GLITCH(x) (-1.0)

// forward declaration, defined per formula
void perturbation_double_loop
( __global const p_config   *g
```

```
    , __global const double     *m_db_dxr
1435 , __global const double     *m_db_dxi
    , __global const double     *m_db_z
    ,               p_status_d *l
    );

1440 // forward declaration, defined per formula
    void perturbation_floatexp_loop
    ( __global const p_config    *g
    , __global const floatexp    *m_db_dxr
    , __global const floatexp    *m_db_dxi
1445 , __global const double      *m_db_z
    ,               p_status_fe *l
    );

    // forward declaration, defined per formula
1450 void perturbation_softfloat_loop
    ( __global const p_config    *g
    , __global const softfloat   *m_db_dxr
    , __global const softfloat   *m_db_dxi
    , __global const softfloat   *m_db_z
1455 ,               p_status_sf *l
    );

    // entry point
    __kernel void perturbation_double
1460 ( __global const p_config *g // configuration including series approximation ↙
       ↳ coefficients
    , __global const double *m_db_dxr // reference orbit re
    , __global const double *m_db_dxi // reference orbit im
    , __global const double *m_db_z   // reference orbit (re^2+im^2)*↙
       ↳ glitch_threshold
    , __global       uint   *n1  // iteration count msb, may be null
1465 , __global       uint   *n0  // iteration count lsb
    , __global       float  *nf  // iteration count fractional part
    , __global       float  *dex // directional de x, may be null
    , __global       float  *dey // directional de y, may be null
    )
1470 {
      if (g->BYTES != sizeof(p_config)) return;
      int y = get_global_id(0);
      int x = get_global_id(1);
      long ix = y * g->stride_y + x * g->stride_x + g->stride_offset;
1475  long orig = 0;
      float origf = 0;
      if (n1)
      {
        orig |= ((long)(int)(n1[ix])) << 32;
1480    if (n0) orig |= n0[ix];
      }
      else
      {
        if (n0) orig = (long)(int)(n0[ix]); // sign extend
1485  }
      if (nf)
      {
        origf = nf[ix];
```

```
  }
1490  if (! g->m_bAddReference || orig == PIXEL_UNEVALUATED || origf < 0) // first ↙
        ↳ or fresh or glitch
  {
    const floatexp zero = fe_floatexp(0.0, 0);
    const floatexp one  = fe_floatexp(1.0, 0);
    long nMaxIter = g->m_nGlitchIter < g->m_nMaxIter ? g->m_nGlitchIter : g->↙
        ↳ m_nMaxIter;
1495    // FIXME TODO mirroring, incremental rendering, guessing
/*
  while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
    int64_t nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
    if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
1500  SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, w, h);
      continue;
    }
    if (GuessPixel(x, y, w, h))
      continue;
1505 */
    // Series approximation
    floatexp D0r = zero;
    floatexp D0i = zero;
    floatexp daa0 = one;
1510  floatexp dab0 = zero;
    floatexp dba0 = zero;
    floatexp dbb0 = one;
    GetPixelCoordinatesM(g, x, y, &D0r, &D0i, &daa0, &dab0, &dba0, &dbb0);
    long antal;
1515  floatexp TDnr;
    floatexp TDni;
    floatexp dxa1, dxb1, dya1, dyb1;
    DoApproximationM(g, &antal, D0r, D0i, &TDnr, &TDni, &dxa1, &dxb1, &dya1, &↙
        ↳ dyb1);
    floatexp TDDnr = dxa1;
1520  floatexp TDDni = dya1;
    double test1 = 0, test2 = 0;
    bool bGlitch = false;
    double dbD0r = fe_double(D0r);
    double dbD0i = fe_double(D0i);
1525  double Dr = fe_double(TDnr);
    double Di = fe_double(TDni);
    double daa = fe_double(daa0);
    double dab = fe_double(dab0);
    double dba = fe_double(dba0);
1530  double dbb = fe_double(dbb0);
    double dr = fe_double(TDDnr);
    double di = fe_double(TDDni);
    dr *= g->m_dPixelSpacing;
    di *= g->m_dPixelSpacing;
1535  // in
    p_status_d l =
      { dbD0r
      , dbD0i
      , Dr
1540  , Di
      , dr
      , di
```

```
      , daa
      , dab
      , dba
      , dbb
      , test1
      , test2
      , antal
      , bGlitch
      };
    // core per pixel calculation
    perturbation_double_loop(g, m_db_dxr, m_db_dxi, m_db_z, &l);
    // out
    Dr = l.xr;
    Di = l.xi;
    dr = l.dr;
    di = l.di;
    test1 = l.test1;
    test2 = l.test2;
    antal = l.antal;
    bGlitch = l.bGlitch;
    dcomplex de = { 0.0, 0.0 };
    if (g->derivatives)
    {
      dcomplex z = { Dr, Di };
      dcomplex dc = { dr, di };
      de = dc_ddiv(dc_abs(z) * log(dc_abs(z)), dc);
    }
    // output iteration data
    if (antal == g->m_nGlitchIter)
      bGlitch = true;
    if (antal >= g->m_nMaxIter){
      if (n1) n1[ix] = antal >> 32;
      if (n0) n0[ix] = antal;
      if (nf) nf[ix] = 0;
      if (dex) dex[ix] = 0;
      if (dey) dey[ix] = 0;
    }
    else{
      if (n1) n1[ix] = antal >> 32;
      if (n0) n0[ix] = antal;
      if (dex) dex[ix] = de.re;
      if (dey) dey[ix] = de.im;
      if (!bGlitch && (g->m_nSmoothMethod == 1 || g->m_nSmoothMethod == 2)){
        double div = sqrt(test1) - sqrt(test2);
        if (div != 0)
        {
          if (nf) nf[ix] = (sqrt(test1) - g->m_nBailout) / div;
        }
        else
        {
          if (nf) nf[ix] = 0;
        }
      }
      else if (!bGlitch && g->m_nSmoothMethod == 0){
        double t = log(log(sqrt(test1))) / g->log_m_nPower;
        if (!ISFLOATOK(t))
          t = 0;
```

```
1600          long i = floor(t);
              antal -= i;
              t -= i;
              if (n1) n1[ix] = antal >> 32;
              if (n0) n0[ix] = antal;
1605          if (nf) nf[ix] = t;
            }

            if (bGlitch && !g->m_bNoGlitchDetection){
              if (nf) nf[ix] = SET_TRANS_GLITCH(test1);
1610        }
          }
        }
      }

1615  // entry point
      __kernel void perturbation_floatexp
      ( __global const p_config *g // configuration including series approximation ↙
          ↳ coefficients
      , __global const floatexp *m_db_dxr // reference orbit re
      , __global const floatexp *m_db_dxi // reference orbit im
1620  , __global const double   *m_db_z   // reference orbit (re^2+im^2)*↙
          ↳ glitch_threshold
      , __global       uint     *n1  // iteration count msb, may be null
      , __global       uint     *n0  // iteration count lsb
      , __global       float    *nf  // iteration count fractional part
      , __global       float    *dex // directional de x, may be null
1625  , __global       float    *dey // directional de y, may be null
      )
      {
        if (g->BYTES != sizeof(p_config)) return;
        int y = get_global_id(0);
1630    int x = get_global_id(1);
        long ix = y * g->stride_y + x * g->stride_x + g->stride_offset;
        long orig = 0;
        float origf = 0;
        if (n1)
1635    {
          orig |= ((long)(int)(n1[ix])) << 32;
          if (n0) orig |= n0[ix];
        }
        else
1640    {
          if (n0) orig = (long)(int)(n0[ix]); // sign extend
        }
        if (nf)
        {
1645      origf = nf[ix];
        }
        if (! g->m_bAddReference || orig == PIXEL_UNEVALUATED || origf < 0) // first ↙
            ↳ or fresh or glitch
        {
          const floatexp zero = fe_floatexp(0.0, 0);
1650      const floatexp one  = fe_floatexp(1.0, 0);
          long nMaxIter = g->m_nGlitchIter < g->m_nMaxIter ? g->m_nGlitchIter : g->↙
              ↳ m_nMaxIter;
          // FIXME TODO mirroring, incremental rendering, guessing
```

```
/*
  while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
1655    int64_t nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
    if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
      SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, w, h);
      continue;
    }
1660    if (GuessPixel(x, y, w, h))
      continue;
*/
    // Series approximation
    floatexp D0r = zero;
1665    floatexp D0i = zero;
    floatexp daa0 = one;
    floatexp dab0 = zero;
    floatexp dba0 = zero;
    floatexp dbb0 = one;
1670    GetPixelCoordinatesM(g, x, y, &D0r, &D0i, &daa0, &dab0, &dba0, &dbb0);
    long antal;
    floatexp TDnr;
    floatexp TDni;
    floatexp dxa1, dxb1, dya1, dyb1;
1675    DoApproximationM(g, &antal, D0r, D0i, &TDnr, &TDni, &dxa1, &dxb1, &dya1, &↙
      ↳ dyb1);
    floatexp TDDnr = dxa1;
    floatexp TDDni = dya1;
    double test1 = 0, test2 = 0;
    bool bGlitch = false;
1680    // in
    p_status_fe l =
      { D0r
      , D0i
      , TDnr
1685      , TDni
      , fe_mul(TDDnr, g->m_fPixelSpacing)
      , fe_mul(TDDni, g->m_fPixelSpacing)
      , daa0
      , dab0
1690      , dba0
      , dbb0
      , test1
      , test2
      , antal
1695      , bGlitch
      };
    // core per pixel calculation
    perturbation_floatexp_loop(g, m_db_dxr, m_db_dxi, m_db_z, &l);
    // out
1700    floatexp Dr = l.xr;
    floatexp Di = l.xi;
    floatexp dr = l.dr;
    floatexp di = l.di;
    test1 = l.test1;
1705    test2 = l.test2;
    antal = l.antal;
    bGlitch = l.bGlitch;
    dcomplex de = { 0.0, 0.0 };
```

```
    if (g->derivatives)
    {
      dcomplex z = { fe_double(Dr), fe_double(Di) };
      dcomplex dc = { fe_double(dr), fe_double(di) };
      de = dc_ddiv(dc_abs(z) * log(dc_abs(z)), dc);
    }
    // output iteration data
    if (antal == g->m_nGlitchIter)
      bGlitch = true;
    if (antal >= g->m_nMaxIter){
      if (n1) n1[ix] = antal >> 32;
      if (n0) n0[ix] = antal;
      if (nf) nf[ix] = 0;
      if (dex) dex[ix] = 0;
      if (dey) dey[ix] = 0;
    }
    else{
      if (n1) n1[ix] = antal >> 32;
      if (n0) n0[ix] = antal;
      if (dex) dex[ix] = de.re;
      if (dey) dey[ix] = de.im;
      if (!bGlitch && (g->m_nSmoothMethod == 1 || g->m_nSmoothMethod == 2)){
        double div = sqrt(test1) - sqrt(test2);
        if (div != 0)
        {
          if (nf) nf[ix] = (sqrt(test1) - g->m_nBailout) / div;
        }
        else
        {
          if (nf) nf[ix] = 0;
        }
      }
      else if (!bGlitch && g->m_nSmoothMethod == 0){
        double t = log(log(sqrt(test1))) / g->log_m_nPower;
        if (!ISFLOATOK(t))
          t = 0;
        long i = floor(t);
        antal -= i;
        t -= i;
        if (n1) n1[ix] = antal >> 32;
        if (n0) n0[ix] = antal;
        if (nf) nf[ix] = t;
      }

      if (bGlitch && !g->m_bNoGlitchDetection){
        if (nf) nf[ix] = SET_TRANS_GLITCH(test1);
      }
    }
  }
}

// entry point
__kernel void perturbation_softfloat
( __global const p_config  *g // configuration including series approximation ↙
    ↳ coefficients
, __global const softfloat *m_db_dxr // reference orbit re
, __global const softfloat *m_db_dxi // reference orbit im
```

```
1765  , __global const softfloat *m_db_z   // reference orbit (re^2+im^2)*↙
        ↳ glitch_threshold
      , __global       uint      *n1  // iteration count msb, may be null
      , __global       uint      *n0  // iteration count lsb
      , __global       float     *nf  // iteration count fractional part
      , __global       float     *dex // directional de x, may be null
1770  , __global       float     *dey // directional de y, may be null
      )
      {
        if (g->BYTES != sizeof(p_config)) return;
        int y = get_global_id(0);
1775    int x = get_global_id(1);
        long ix = y * g->stride_y + x * g->stride_x + g->stride_offset;
        long orig = 0;
        float origf = 0;
        if (n1)
1780    {
          orig |= ((long)(int)(n1[ix])) << 32;
          if (n0) orig |= n0[ix];
        }
        else
1785    {
          if (n0) orig = (long)(int)(n0[ix]); // sign extend
        }
        if (nf)
        {
1790      origf = nf[ix];
        }
        if (! g->m_bAddReference || orig == PIXEL_UNEVALUATED || origf < 0) // first ↙
          ↳ or fresh or glitch
        {
          const floatexp zero = fe_floatexp(0.0, 0);
1795      const floatexp one  = fe_floatexp(1.0, 0);
          long nMaxIter = g->m_nGlitchIter < g->m_nMaxIter ? g->m_nGlitchIter : g->↙
            ↳ m_nMaxIter;
          // FIXME TODO mirroring, incremental rendering, guessing
      /*
        while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
1800      int64_t nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
          if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
            SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, w, h);
            continue;
          }
1805      if (GuessPixel(x, y, w, h))
            continue;
      */
          // Series approximation
          floatexp D0r = zero;
1810      floatexp D0i = zero;
          floatexp daa0 = one;
          floatexp dab0 = zero;
          floatexp dba0 = zero;
          floatexp dbb0 = one;
1815      GetPixelCoordinatesM(g, x, y, &D0r, &D0i, &daa0, &dab0, &dba0, &dbb0);
          long antal;
          floatexp TDnr;
          floatexp TDni;
```

```
    floatexp dxa1, dxb1, dya1, dyb1;
1820    DoApproximationM(g, &antal, D0r, D0i, &TDnr, &TDni, &dxa1, &dxb1, &dya1, &↙
            ↳ dyb1);
    floatexp TDDnr = dxa1;
    floatexp TDDni = dya1;
    double test1 = 0.0, test2 = 0.0;
    bool bGlitch = false;
1825    // in
    p_status_sf l =
      { sf_from_floatexp(D0r)
      , sf_from_floatexp(D0i)
      , sf_from_floatexp(TDnr)
1830      , sf_from_floatexp(TDni)
      , sf_from_floatexp(fe_mul(TDDnr, g->m_fPixelSpacing))
      , sf_from_floatexp(fe_mul(TDDni, g->m_fPixelSpacing))
      , sf_from_floatexp(daa0)
      , sf_from_floatexp(dab0)
1835      , sf_from_floatexp(dba0)
      , sf_from_floatexp(dbb0)
      , sf_from_double(test1)
      , sf_from_double(test2)
      , antal
1840      , bGlitch
      };
    // core per pixel calculation
    perturbation_softfloat_loop(g, m_db_dxr, m_db_dxi, m_db_z, &l);
    // out
1845    softfloat Dr = l.xr;
    softfloat Di = l.xi;
    softfloat dr = l.dr;
    softfloat di = l.di;
    test1 = sf_to_double(l.test1);
1850    test2 = sf_to_double(l.test2);
    antal = l.antal;
    bGlitch = l.bGlitch;
    dcomplex de = { 0.0, 0.0 };
    if (g->derivatives)
1855    {
      dcomplex z = { sf_to_double(Dr), sf_to_double(Di) };
      dcomplex dc = { sf_to_double(dr), sf_to_double(di) };
      de = dc_ddiv(dc_abs(z) * log(dc_abs(z)), dc);
    }
1860    // output iteration data
    if (antal == g->m_nGlitchIter)
      bGlitch = true;
    if (antal >= g->m_nMaxIter){
      if (n1) n1[ix] = antal >> 32;
1865      if (n0) n0[ix] = antal;
      if (nf) nf[ix] = 0;
      if (dex) dex[ix] = 0;
      if (dey) dey[ix] = 0;
    }
1870    else{
      if (n1) n1[ix] = antal >> 32;
      if (n0) n0[ix] = antal;
      if (dex) dex[ix] = de.re;
      if (dey) dey[ix] = de.im;
```

```
1875        if (!bGlitch && (g->m_nSmoothMethod == 1 || g->m_nSmoothMethod == 2)){
              double div = sqrt(test1) - sqrt(test2);
              if (div != 0)
              {
                if (nf) nf[ix] = (sqrt(test1) - g->m_nBailout) / div;
1880          }
              else
              {
                if (nf) nf[ix] = 0;
              }
1885        }
            else if (!bGlitch && g->m_nSmoothMethod == 0){
              double t = log(log(sqrt(test1))) / g->log_m_nPower;
              if (!ISFLOATOK(t))
                t = 0;
1890          long i = floor(t);
              antal -= i;
              t -= i;
              if (n1) n1[ix] = antal >> 32;
              if (n0) n0[ix] = antal;
1895          if (nf) nf[ix] = t;
            }

            if (bGlitch && !g->m_bNoGlitchDetection){
              if (nf) nf[ix] = SET_TRANS_GLITCH(test1);
1900        }
          }
        }
      }
```

5 cl/double_post.cl

```
// ...continued
      xr = xrn;
      xi = xin;
    }
5   l->antal = antal;
    l->test1 = test1;
    l->test2 = test2;
    l->xr = Xxr;
    l->xi = Xxi;
10  l->dr = 0;
    l->di = 0;
}
```

6 cl/double_post_m.cl

```
// ...continued
      xr = xrn;
      xi = xin;
      dxa = dxan;
5     dxb = dxbn;
      dya = dyan;
      dyb = dybn;
    }
    l->antal = antal;
```

```
  l->test1 = test1;
  l->test2 = test2;
  l->xr = Xxr;
  l->xi = Xxi;
  l->dr = (Xxr * dxa + Xxi * dya) / sqrt(test1);
  l->di = (Xxr * dxb + Xxi * dyb) / sqrt(test1);
}
```

7 cl/double_post_rc.cl

```
// ...continued
    xr = xrn;
    xi = xin;
    dr = drn;
    di = din;
  }
  l->antal = antal;
  l->test1 = test1;
  l->test2 = test2;
  l->xr = Xxr;
  l->xi = Xxi;
  l->dr = dr;
  l->di = di;
}
```

8 cl/double_pre.cl

```
void perturbation_double_loop
( __global const p_config   *g
, __global const double     *m_db_dxr
, __global const double     *m_db_dxi
, __global const double     *m_db_z
,                p_status_d *l
)
{
  const double Ar = g->g_FactorAR;
  const double Ai = g->g_FactorAI;
  bool no_g = g->g_real == 1.0 && g->g_imag == 1.0;
  const double cr = l->cr;
  const double ci = l->ci;
  double xr = l->xr;
  double xi = l->xi;
  // type C derivatives
  double dr = l->dr;
  double di = l->di;
  // type M derivatives
  double daa = l->daa;
  double dab = l->dab;
  double dba = l->dba;
  double dbb = l->dbb;
  double dxa = dr;
  double dxb = -di;
  double dya = di;
  double dyb = dr;
  // conditions
  double test1 = l->test1;
```

```
    double test2 = l->test2;
    long antal = l->antal;
    double Xxr = 0;
    double Xxi = 0;
    for (; antal < g->nMaxIter; antal++)
    {
      const double Xr = m_db_dxr[antal - g->nMinIter];
      const double Xi = m_db_dxi[antal - g->nMinIter];
      const double Xz = m_db_z  [antal - g->nMinIter];
      Xxr = Xr + xr;
      Xxi = Xi + xi;
      test2 = test1;
      if (no_g)
      {
        test1 = Xxr * Xxr + Xxi * Xxi;
      }
      else
      {
        test1 = g->g_real * Xxr * Xxr + g->g_imag * Xxi * Xxi;
      }
      if (test1 < Xz)
      {
        l->bGlitch = true;
        if (! g->m_bNoGlitchDetection)
          break;
      }
      if (test1 > g->m_nBailout2)
      {
        break;
      }
      const dcomplex X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi};
      double xrn = 0;
      double xin = 0;
      dcomplex xn = {0, 0};
      const dcomplex d = {dr, di}, d0 = {daa, dba}; // FIXME matrix derivatives
      double drn = 0, din = 0;
      dcomplex dn = {0, 0};
      double dxan = 0, dxbn = 0, dyan = 0, dybn = 0;
// continued...
```

9 cl/floatexp_post.cl

```
// ...continued
      xr = xrn;
      xi = xin;
    }
    l->antal = antal;
    l->test1 = test1;
    l->test2 = test2;
    l->xr = Xxr;
    l->xi = Xxi;
    l->dr = zero;
    l->di = zero;
}
```

10 cl/floatexp_post_m.cl

```
// ...continued
    xr = xrn;
    xi = xin;
    dxa = dxan;
    dxb = dxbn;
    dya = dyan;
    dyb = dybn;
  }
  l->antal = antal;
  l->test1 = test1;
  l->test2 = test2;
  l->xr = Xxr;
  l->xi = Xxi;
  l->dr = fe_double(fe_add(fe_mul(Xxr, dxa), fe_mul(Xxi, dya))) / sqrt(test1);
  l->di = fe_double(fe_add(fe_mul(Xxr, dxb), fe_mul(Xxi, dyb))) / sqrt(test1);
}
```

11 cl/floatexp_post_rc.cl

```
// ...continued
    xr = xrn;
    xi = xin;
    dr = drn;
    di = din;
  }
  l->antal = antal;
  l->test1 = test1;
  l->test2 = test2;
  l->xr = Xxr;
  l->xi = Xxi;
  l->dr = dr;
  l->di = di;
}
```

12 cl/floatexp_pre.cl

```
void perturbation_floatexp_loop
( __global const p_config    *g
, __global const floatexp    *m_db_dxr
, __global const floatexp    *m_db_dxi
, __global const double      *m_db_z
,                p_status_fe *l
)
{
  const floatexp zero = fe_floatexp(0.0, 0);
  const double Ar = g->g_FactorAR;
  const double Ai = g->g_FactorAI;
  bool no_g = g->g_real == 1.0 && g->g_imag == 1.0;
  const floatexp cr = l->cr;
  const floatexp ci = l->ci;
  floatexp xr = l->xr;
  floatexp xi = l->xi;
  // type C derivatives
  floatexp dr = l->dr;
  floatexp di = l->di;
  // type M derivatives
```

```
  floatexp daa = l->daa;
  floatexp dab = l->dab;
  floatexp dba = l->dba;
  floatexp dbb = l->dbb;
  floatexp dxa = dr;
  floatexp dxb = fe_neg(di);
  floatexp dya = di;
  floatexp dyb = dr;
  // conditions
  double test1 = l->test1;
  double test2 = l->test2;
  long antal = l->antal;
  floatexp Xxr = zero;
  floatexp Xxi = zero;
  for (; antal < g->nMaxIter; antal++)
  {
    const floatexp Xr = m_db_dxr[antal - g->nMinIter];
    const floatexp Xi = m_db_dxi[antal - g->nMinIter];
    const double   Xz = m_db_z  [antal - g->nMinIter];
    Xxr = fe_add(Xr, xr);
    Xxi = fe_add(Xi, xi);
    test2 = test1;
    if (no_g)
    {
      test1 = fe_double(fe_add(fe_sqr(Xxr), fe_sqr(Xxi)));
    }
    else
    {
      test1 = fe_double(fe_add(fe_dmul(g->g_real, fe_sqr(Xxr)), fe_dmul(g->↙
          ↳ g_imag, fe_sqr(Xxi))));
    }
    if (test1 < Xz)
    {
      l->bGlitch = true;
      if (! g->m_bNoGlitchDetection)
        break;
    }
    if (test1 > g->m_nBailout2)
    {
      break;
    }
    const fecomplex X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi};
    floatexp xrn = zero;
    floatexp xin = zero;
    fecomplex xn = {zero, zero};
    const fecomplex d = {dr, di}, d0 = {daa, dba}; // FIXME matrix derivatives
    floatexp drn = zero, din = zero;
    fecomplex dn = {zero, zero};
    floatexp dxan = zero, dxbn = zero, dyan = zero, dybn = zero;
// continued...
```

13 cl/formula.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" />
<xsl:template match="/">
```

```
5
    #include &lt;string&gt;
    #include &lt;sstream&gt;

    #include "common_cl.c"
10  #include "double_pre_cl.c"
    #include "double_post_cl.c"
    #include "double_post_rc_cl.c"
    #include "double_post_m_cl.c"
    #include "floatexp_pre_cl.c"
15  #include "floatexp_post_cl.c"
    #include "floatexp_post_rc_cl.c"
    #include "floatexp_post_m_cl.c"
    #include "softfloat_pre_cl.c"
    #include "softfloat_post_cl.c"
20  #include "softfloat_post_rc_cl.c"
    #include "softfloat_post_m_cl.c"

    #define STR(s) #s
    <xsl:for-each select="formulas/group/formula">
25  static const char *perturbation_opencl_double_<xsl:value-of select="../@type" />↙
        ↳ _<xsl:value-of select="@power" /> = STR(
    <xsl:choose>
    <xsl:when test="perturbation/@t='C'">
    #if DERIVATIVES == 1
    <xsl:choose>
30  <xsl:when test="derivative/@t='R' or derivative/@t='M'">
    @cld {
          <xsl:value-of select="derivative" />
        }
    </xsl:when>
35  <xsl:when test="derivative/@t='C'">
    @cldc {
          <xsl:value-of select="derivative" />
        }
        drn = dn.re; din = dn.im;
40  </xsl:when>
    </xsl:choose>
    #endif
    @cldc {
          <xsl:value-of select="perturbation" />
45      }
        xrn = xn.re; xin = xn.im;
    </xsl:when>
    <xsl:when test="perturbation/@t='R'">
    #if DERIVATIVES == 1
50  <xsl:choose>
    <xsl:when test="derivative/@t='C'">
    @cldc {
          <xsl:value-of select="derivative" />
        }
55      drn = dn.m_r; din = dn.m_i;
    </xsl:when>
    <xsl:when test="derivative/@t='R' or derivative/@t='M'">
    @cld {
          <xsl:value-of select="derivative" />
60      }
```

```
</xsl:when>
</xsl:choose>
#endif
@cld  {
        <xsl:value-of select="perturbation" />
      }
</xsl:when>
</xsl:choose>
);
</xsl:for-each>

<xsl:for-each select="formulas/group/formula">
static const char *perturbation_opencl_floatexp_<xsl:value-of select="../@type" />_<xsl:value-of select="@power" /> = STR(
<xsl:choose>
<xsl:when test="perturbation/@t='C'">
#if DERIVATIVES == 1
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@clfe   {
        <xsl:value-of select="derivative" />
        }
</xsl:when>
<xsl:when test="derivative/@t='C'">
@clfec  {
        <xsl:value-of select="derivative" />
        }
      drn = dn.re; din = dn.im;
</xsl:when>
</xsl:choose>
#endif
@clfec  {
        <xsl:value-of select="perturbation" />
        }
      xrn = xn.re; xin = xn.im;
</xsl:when>
<xsl:when test="perturbation/@t='R'">
#if DERIVATIVES == 1
<xsl:choose>
<xsl:when test="derivative/@t='C'">
@clfec  {
        <xsl:value-of select="derivative" />
        }
      drn = dn.m_r; din = dn.m_i;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@clfe   {
        <xsl:value-of select="derivative" />
        }
</xsl:when>
</xsl:choose>
#endif
@clfe   {
        <xsl:value-of select="perturbation" />
        }
</xsl:when>
</xsl:choose>
```

```
);
</xsl:for-each>

<xsl:for-each select="formulas/group/formula">
static const char *perturbation_opencl_softfloat_<xsl:value-of select="../@type" />_<xsl:value-of select="@power" /> = STR(
<xsl:choose>
<xsl:when test="perturbation/@t='C'">
#if DERIVATIVES == 1
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@clsf   {
        <xsl:value-of select="derivative" />
        }
</xsl:when>
<xsl:when test="derivative/@t='C'">
@clsfc  {
        <xsl:value-of select="derivative" />
        }
      drn = dn.re; din = dn.im;
</xsl:when>
</xsl:choose>
#endif
@clsfc  {
        <xsl:value-of select="perturbation" />
        }
      xrn = xn.re; xin = xn.im;
</xsl:when>
<xsl:when test="perturbation/@t='R'">
#if DERIVATIVES == 1
<xsl:choose>
<xsl:when test="derivative/@t='C'">
@clsfc  {
        <xsl:value-of select="derivative" />
        }
      drn = dn.m_r; din = dn.m_i;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@clsf   {
        <xsl:value-of select="derivative" />
        }
</xsl:when>
</xsl:choose>
#endif
@clsf   {
        <xsl:value-of select="perturbation" />
        }
</xsl:when>
</xsl:choose>
);
</xsl:for-each>

static const std::string perturbation_opencl_error = "#error unsupported fractal type and power\n";

extern std::string perturbation_opencl(int m_nFractalType, int m_nPower, int derivatives)
```

```
{
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
      <xsl:for-each select="formula">
        case <xsl:value-of select="@power" />:
        {
          std::ostringstream o;
          o &lt;&lt; perturbation_opencl_common;
          o &lt;&lt; perturbation_opencl_double_pre;
          o &lt;&lt; perturbation_opencl_double_<xsl:value-of select="../@type" />
            ↳ /><xsl:value-of select="@power" />;
          if (derivatives)
          {
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
            o &lt;&lt; perturbation_opencl_double_post_rc;
</xsl:when>
<xsl:when test="derivative/@t='M'">
            o &lt;&lt; perturbation_opencl_double_post_m;
</xsl:when>
</xsl:choose>
          }
          else
          {
            o &lt;&lt; perturbation_opencl_double_post;
          }
          o &lt;&lt; perturbation_opencl_floatexp_pre;
          o &lt;&lt; perturbation_opencl_floatexp_<xsl:value-of select="../@type
            ↳ " /><xsl:value-of select="@power" />;
          if (derivatives)
          {
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
            o &lt;&lt; perturbation_opencl_floatexp_post_rc;
</xsl:when>
<xsl:when test="derivative/@t='M'">
            o &lt;&lt; perturbation_opencl_floatexp_post_m;
</xsl:when>
</xsl:choose>
          }
          else
          {
            o &lt;&lt; perturbation_opencl_floatexp_post;
          }
          o &lt;&lt; perturbation_opencl_softfloat_pre;
          o &lt;&lt; perturbation_opencl_softfloat_<xsl:value-of select="../
            ↳ @type" /><xsl:value-of select="@power" />;
          if (derivatives)
          {
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
            o &lt;&lt; perturbation_opencl_softfloat_post_rc;
```

```
225 </xsl:when>
<xsl:when test="derivative/@t='M'">
            o &lt;&lt; perturbation_opencl_softfloat_post_m;
</xsl:when>
</xsl:choose>
230           }
              else
              {
                o &lt;&lt; perturbation_opencl_softfloat_post;
              }
235           return o.str();
            }
          </xsl:for-each>
          }
          break;
240     </xsl:for-each>
      }
      return perturbation_opencl_error;
}

245 </xsl:template>
</xsl:stylesheet>
```

14 cl/opencl.cpp

```
#ifdef KF_OPENCL

#include <assert.h>
#include <math.h>
 5 #include <string.h>

#include "../fraktal_sft/fraktal_sft.h"

const char *error_string(int err) {
10   switch (err) {
     case CL_SUCCESS:                            return 0;
     case CL_DEVICE_NOT_FOUND:                   return "Device not found.";
     case CL_DEVICE_NOT_AVAILABLE:               return "Device not available";
     case CL_COMPILER_NOT_AVAILABLE:             return "Compiler not available";
15   case CL_MEM_OBJECT_ALLOCATION_FAILURE:      return "Memory object allocation ↙
         ↳ failure";
     case CL_OUT_OF_RESOURCES:                   return "Out of resources";
     case CL_OUT_OF_HOST_MEMORY:                 return "Out of host memory";
     case CL_PROFILING_INFO_NOT_AVAILABLE:       return "Profiling information not ↙
         ↳ available";
     case CL_MEM_COPY_OVERLAP:                   return "Memory copy overlap";
20   case CL_IMAGE_FORMAT_MISMATCH:              return "Image format mismatch";
     case CL_IMAGE_FORMAT_NOT_SUPPORTED:         return "Image format not supported↙
         ↳ ";
     case CL_BUILD_PROGRAM_FAILURE:              return "Program build failure";
     case CL_MAP_FAILURE:                        return "Map failure";
     case CL_INVALID_VALUE:                      return "Invalid value";
25   case CL_INVALID_DEVICE_TYPE:                return "Invalid device type";
     case CL_INVALID_PLATFORM:                   return "Invalid platform";
     case CL_INVALID_DEVICE:                     return "Invalid device";
     case CL_INVALID_CONTEXT:                    return "Invalid context";
     case CL_INVALID_QUEUE_PROPERTIES:           return "Invalid queue properties";
```

```
30    case CL_INVALID_COMMAND_QUEUE:                return "Invalid command queue";
      case CL_INVALID_HOST_PTR:                     return "Invalid host pointer";
      case CL_INVALID_MEM_OBJECT:                   return "Invalid memory object";
      case CL_INVALID_IMAGE_FORMAT_DESCRIPTOR:      return "Invalid image format ↙
          ↳ descriptor";
      case CL_INVALID_IMAGE_SIZE:                   return "Invalid image size";
35    case CL_INVALID_SAMPLER:                      return "Invalid sampler";
      case CL_INVALID_BINARY:                       return "Invalid binary";
      case CL_INVALID_BUILD_OPTIONS:                return "Invalid build options";
      case CL_INVALID_PROGRAM:                      return "Invalid program";
      case CL_INVALID_PROGRAM_EXECUTABLE:           return "Invalid program executable↙
          ↳ ";
40    case CL_INVALID_KERNEL_NAME:                  return "Invalid kernel name";
      case CL_INVALID_KERNEL_DEFINITION:            return "Invalid kernel definition↙
          ↳ ";
      case CL_INVALID_KERNEL:                       return "Invalid kernel";
      case CL_INVALID_ARG_INDEX:                    return "Invalid argument index";
      case CL_INVALID_ARG_VALUE:                    return "Invalid argument value";
45    case CL_INVALID_ARG_SIZE:                     return "Invalid argument size";
      case CL_INVALID_KERNEL_ARGS:                  return "Invalid kernel arguments";
      case CL_INVALID_WORK_DIMENSION:               return "Invalid work dimension";
      case CL_INVALID_WORK_GROUP_SIZE:              return "Invalid work group size";
      case CL_INVALID_WORK_ITEM_SIZE:               return "Invalid work item size";
50    case CL_INVALID_GLOBAL_OFFSET:                return "Invalid global offset";
      case CL_INVALID_EVENT_WAIT_LIST:              return "Invalid event wait list";
      case CL_INVALID_EVENT:                        return "Invalid event";
      case CL_INVALID_OPERATION:                    return "Invalid operation";
      case CL_INVALID_GL_OBJECT:                    return "Invalid OpenGL object";
55    case CL_INVALID_BUFFER_SIZE:                  return "Invalid buffer size";
      case CL_INVALID_MIP_LEVEL:                    return "Invalid mip-map level";
      default: return "Unknown";
    }
  }
60
  void error_print(int err, int loc) {
    if (err == CL_SUCCESS) { return; }
    fprintf(stderr, "CL ERROR: %d %s (%d)\n", err, error_string(err), loc);
    exit(1);
65  }

  #define E(err) error_print(err, __LINE__)

  std::vector<cldevice> initialize_opencl()
70  {
    std::vector<cldevice> devices;
    int ok = clewInit();
    if (ok != CLEW_SUCCESS)
    {
75    return devices;
    }

    cl_platform_id platform_id[64];
    cl_uint platform_ids;
80  E(clGetPlatformIDs(64, &platform_id[0], &platform_ids));
    char buf[1024];
    for (cl_uint i = 0; i < platform_ids; ++i) {
      buf[0] = 0;
```

```
      E(clGetPlatformInfo(platform_id[i], CL_PLATFORM_VERSION, 1024, &buf[0], 0));
      std::string version(buf);
      buf[0] = 0;
      E(clGetPlatformInfo(platform_id[i], CL_PLATFORM_VENDOR, 1024, &buf[0], 0));
      std::string vendor(buf);

      cl_device_id device_id[64];
      cl_uint device_ids;
      E(clGetDeviceIDs(platform_id[i], CL_DEVICE_TYPE_ALL, 64, &device_id[0], &↙
          ↳ device_ids));
      for (cl_uint j = 0; j < device_ids; ++j)
      {
        buf[0] = 0;
        E(clGetDeviceInfo(device_id[j], CL_DEVICE_NAME, 1024, &buf[0], 0));
        std::string dname(buf);
        cldevice d = { platform_id[i], device_id[j], dname, vendor, version };
        devices.push_back(d);
      }
    }
    return devices;
  }

OpenCL::OpenCL(cl_platform_id platform_id0, cl_device_id device_id0)
: platform_id(platform_id0)
, device_id(device_id0)
, context(0)
, commands(0)
, program(0)
, config_bytes(0)
, config(0)
, n1_bytes(0), n1(0)
, n0_bytes(0), n0(0)
, nf_bytes(0), nf(0)
, dex_bytes(0), dex(0)
, dey_bytes(0), dey(0)
{
  //memset(this, 0, sizeof(*this));
  mutex = CreateMutex(0,0,0);
  platform_id = platform_id0;
  device_id = device_id0;
  // create context
  cl_context_properties properties[] =
    {
      CL_CONTEXT_PLATFORM, (cl_context_properties) platform_id
    , 0
    };
  cl_int err;
  context = clCreateContext(properties, 1, &device_id, NULL, NULL, &err);
  if (! context) { E(err); }
  commands = clCreateCommandQueue(context, device_id, 0, &err);
  if (! commands) { E(err); }
  config_bytes = sizeof(p_config);
  config = clCreateBuffer(context, CL_MEM_READ_ONLY, config_bytes, 0, &err); if ↙
      ↳ (! config) { E(err); }
}
```

```
OpenCL::~OpenCL()
{
  WaitForSingleObject(mutex, INFINITE);
  // FIXME TODO
  ReleaseMutex(mutex);
}

void OpenCL::lock()
{
  WaitForSingleObject(mutex, INFINITE);
}

void OpenCL::unlock()
{
  ReleaseMutex(mutex);
}

struct softfloat
{
  uint32_t se;
  uint32_t m;
};
#define SF_EXPONENT_BIAS ((1U << 30U) - 1U)
#define SF_MANTISSA_BITS 32

bool sf_sign_bit(const softfloat f)
{
  return !!(f.se & 0x80000000U);
}

uint32_t sf_biased_exponent(const softfloat f)
{
  return f.se & 0x7FFFFFFFU;
}

uint32_t sf_mantissa(const softfloat f)
{
  return f.m;
}

bool sf_is_zero(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0 &&
    sf_mantissa(f) == 0;
}

bool sf_is_denormal(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0 &&
    sf_mantissa(f) != 0;
}

bool sf_is_inf(const softfloat f)
{
  return
```

```
    sf_biased_exponent(f) == 0x7FFFFFFFU &&
    sf_mantissa(f) == 0;
}

bool sf_is_nan(const softfloat f)
{
  return
    sf_biased_exponent(f) == 0x7FFFFFFFU &&
    sf_mantissa(f) != 0;
}

softfloat sf_from_double(const double x)
{
  if (isnan(x))
  {
    softfloat f = { ((uint32_t)(!!signbit(x)) << 31) | 0x7FFFFFFFU, 0xFFFFFFFFU ↙
        ↳ };
    return f;
  }
  else if (isinf(x))
  {
    softfloat f = { ((uint32_t)(!!signbit(x)) << 31) | 0x7FFFFFFFU, 0U };
    return f;
  }
  else if (x == 0.0)
  {
    softfloat f = { ((uint32_t)(!!signbit(x)) << 31) | 0U, 0U };
    return f;
  }
  else
  {
    int e;
    double y = frexp(fabs(x), &e);
    double z = ldexp(y, SF_MANTISSA_BITS);
    uint32_t mantissa = (uint32_t) trunc(z); // rounding might overflow rarely
    uint32_t biased_e = e + SF_EXPONENT_BIAS; // always in range?
    assert(0 < biased_e);
    assert(biased_e < 0x7FFFFFFFU);
    assert((mantissa >> (SF_MANTISSA_BITS - 1)) == 1U);
    softfloat f = { ((uint32_t)(!!signbit(x)) << 31) | biased_e, mantissa };
    return f;
  }
}

softfloat sf_ldexp(const softfloat a, int e)
{
  if (sf_is_zero(a) || sf_is_inf(a) || sf_is_nan(a))
  {
    return a;
  }
  else if (e >= (int32_t)(0x7FFFFFFFU - sf_biased_exponent(a)))
  {
    // overflow to +/-infinity
    softfloat o = { (a.se & 0x80000000U) | 0x7FFFFFFFU, 0U };
    return o;
  }
  else if ((int)(sf_biased_exponent(a)) + e <= 0)
```

```
  {
    // underfloat to 0
    softfloat o = { (a.se & 0x80000000U) | 0U, 0U };
    return o;
  }
  else
  {
    softfloat o = { (a.se & 0x80000000U) | (sf_biased_exponent(a) + e), ↙
        ↳ sf_mantissa(a) };
    return o;
  }
}

softfloat sf_from_floatexp(const floatexp x)
{
  return sf_ldexp(sf_from_double(x.val), x.exp < INT_MIN ? INT_MIN : x.exp > ↙
      ↳ INT_MAX ? INT_MAX : x.exp);
}

softfloat sf_softfloat(const floatexp f)
{
  return sf_from_floatexp(f);
}

softfloat sf_softfloat(const double f)
{
  return sf_from_double(f);
}

template <typename T>
void OpenCL::run
(
  // for pixel -> parameter mapping
  int32_t m_nX,
  int32_t m_nY,
  uint32_t JitterSeed,
  int32_t JitterShape,
  double JitterScale,
  floatexp m_pixel_step_x,
  floatexp m_pixel_step_y,
  floatexp m_pixel_center_x,
  floatexp m_pixel_center_y,
  double m_C,
  double m_S,
  // for result -> output mapping
  int64_t stride_y,
  int64_t stride_x,
  int64_t stride_offset,
  // for iteration control
  double m_nBailout,
  double m_nBailout2,
  double log_m_nPower,
  int64_t m_nGlitchIter,
  int64_t m_nMaxIter,
  int64_t nMaxIter,
  int64_t nMinIter,
  int16_t m_bNoGlitchDetection,
```

```
    int16_t m_bAddReference,
    int16_t m_nSmoothMethod,
    double g_real,
    double g_imag,
    double g_FactorAR,
    double g_FactorAI,
    double m_epsilon,
    // for series approximation
    double m_dPixelSpacing,
    floatexp m_fPixelSpacing,
    int64_t m_nMaxApproximation,
    int32_t m_nApproxTerms,
    int32_t approximation_type,
    floatexp *APr,
    floatexp *APi,
    SeriesR2 *APs,

    // reference orbit
    const T *rx,
    const T *ry,
    const double *rz,
    size_t roffset,
    size_t rcount,

    // formula selection
    int type,
    int m_nFractalType,
    int m_nPower,
    int16_t derivatives,

    // output arrays
    uint32_t *n1_p,
    uint32_t *n0_p,
    float *nf_p,
    float *dex_p,
    float *dey_p
  )
  {
    assert(roffset < rcount);
    lock();

    // upload config
    p_config configdata =
      {
        sizeof(p_config),
        m_nX,
        m_nY,
        JitterSeed,
        JitterShape,
        JitterScale,
        m_pixel_step_x,
        m_pixel_step_y,
        m_pixel_center_x,
        m_pixel_center_y,
        m_C,
        m_S,
        stride_y,
```

```
        stride_x,
        stride_offset,
        m_nBailout,
        m_nBailout2,
        log_m_nPower,
        m_nGlitchIter,
        m_nMaxIter,
        nMaxIter,
        nMinIter,
        m_bNoGlitchDetection,
        derivatives,
        m_bAddReference,
        m_nSmoothMethod,
        g_real,
        g_imag,
        g_FactorAR,
        g_FactorAI,
        m_epsilon,
        m_dPixelSpacing,
        m_fPixelSpacing,
        m_nMaxApproximation,
        m_nApproxTerms,
        approximation_type
        // arrays go here
      };
    const floatexp zero(0);
    for (int i = 0; i < MAX_APPROX_TERMS + 1; ++i)
    {
      configdata.m_APr[i] = APr ? APr[i] : zero;
      configdata.m_APi[i] = APi ? APi[i] : zero;
      for (int j = 0; j < MAX_APPROX_TERMS + 1; ++j)
      {
        configdata.m_APs_s[i][j] = APs ? APs->s[i][j] : zero;
        configdata.m_APs_t[i][j] = APs ? APs->t[i][j] : zero;
      }
    }
    cl_event config_uploaded;
    E(clEnqueueWriteBuffer(commands, config, CL_FALSE, 0, sizeof(configdata), &↙
        ↳ configdata, 0, 0, &config_uploaded));

    // upload reference
    cl_int err;
    int64_t ref_bytes  = sizeof(rx[0]) * (rcount - roffset);
    int64_t ref_bytesz = sizeof(rz[0]) * (rcount - roffset);
    cl_mem refx = clCreateBuffer(context, CL_MEM_READ_ONLY, ref_bytes,  nullptr, &↙
        ↳ err); if (! refx) { E(err); }
    cl_mem refy = clCreateBuffer(context, CL_MEM_READ_ONLY, ref_bytes,  nullptr, &↙
        ↳ err); if (! refy) { E(err); }
    cl_mem refz = clCreateBuffer(context, CL_MEM_READ_ONLY, ref_bytesz, nullptr, &↙
        ↳ err); if (! refz) { E(err); }
    cl_event refx_uploaded, refy_uploaded, refz_uploaded;
    E(clEnqueueWriteBuffer(commands, refx, CL_FALSE, 0, ref_bytes,  &rx[roffset], ↙
        ↳ 0, 0, &refx_uploaded));
    E(clEnqueueWriteBuffer(commands, refy, CL_FALSE, 0, ref_bytes,  &ry[roffset], ↙
        ↳ 0, 0, &refy_uploaded));
    E(clEnqueueWriteBuffer(commands, refz, CL_FALSE, 0, ref_bytesz, &rz[roffset], ↙
        ↳ 0, 0, &refz_uploaded));
```

```
415     // reallocate output buffers if necessary
        {
          if (n1_p)
          {
            size_t bytes = sizeof(uint32_t) * m_nX * m_nY;
420         if (n1_bytes != bytes)
            {
              if (n1_bytes)
              {
                clReleaseMemObject(n1);
425           }
              n1_bytes = bytes;
              if (n1_bytes)
              {
                n1 = clCreateBuffer(context, CL_MEM_READ_WRITE, bytes, nullptr, &err);↙
                    ↳ if (! n1) { E(err); }
430           }
            }
          }
          else
          {
435         if (n1_bytes)
            {
              clReleaseMemObject(n1);
              n1_bytes = 0;
            }
440       }

          if (n0_p)
          {
            size_t bytes = sizeof(uint32_t) * m_nX * m_nY;
445         if (n0_bytes != bytes)
            {
              if (n0_bytes)
              {
                clReleaseMemObject(n0);
450           }
              n0_bytes = bytes;
              if (n0_bytes)
              {
                n0 = clCreateBuffer(context, CL_MEM_READ_WRITE, bytes, nullptr, &err);↙
                    ↳ if (! n0) { E(err); }
455           }
            }
          }
          else
          {
460         if (n0_bytes)
            {
              clReleaseMemObject(n0);
              n0_bytes = 0;
            }
465       }

          if (nf_p)
          {
```

```
      size_t bytes = sizeof(float) * m_nX * m_nY;
      if (nf_bytes != bytes)
      {
        if (nf_bytes)
        {
          clReleaseMemObject(nf);
        }
        nf_bytes = bytes;
        if (nf_bytes)
        {
          nf = clCreateBuffer(context, CL_MEM_READ_WRITE, bytes, nullptr, &err);
            ↳ if (! nf) { E(err); }
        }
      }
    }
    else
    {
      if (nf_bytes)
      {
        clReleaseMemObject(nf);
        nf_bytes = 0;
      }
    }

    if (dex_p)
    {
      size_t bytes = sizeof(float) * m_nX * m_nY;
      if (dex_bytes != bytes)
      {
        if (dex_bytes)
        {
          clReleaseMemObject(dex);
        }
        dex_bytes = bytes;
        if (dex_bytes)
        {
          dex = clCreateBuffer(context, CL_MEM_READ_WRITE, bytes, nullptr, &err)
            ↳ ; if (! dex) { E(err); }
        }
      }
    }
    else
    {
      if (dex_bytes)
      {
        clReleaseMemObject(dex);
        dex_bytes = 0;
      }
    }

    if (dey_p)
    {
      size_t bytes = sizeof(float) * m_nX * m_nY;
      if (dey_bytes != bytes)
      {
        if (dey_bytes)
        {
```

```
          clReleaseMemObject(dey);
        }
        dey_bytes = bytes;
        if (dey_bytes)
        {
          dey = clCreateBuffer(context, CL_MEM_READ_WRITE, bytes, nullptr, &err)
              ; if (! dey) { E(err); }
        }
      }
    }
    else
    {
      if (dey_bytes)
      {
        clReleaseMemObject(dey);
        dey_bytes = 0;
      }
    }
  }

  // upload buffers
  // FIXME synchronous, async would be better
  if (n1_p)  E(clEnqueueWriteBuffer(commands, n1,  CL_TRUE, 0, n1_bytes,  n1_p,
      0, 0, 0));
  if (n0_p)  E(clEnqueueWriteBuffer(commands, n0,  CL_TRUE, 0, n0_bytes,  n0_p,
      0, 0, 0));
  if (nf_p)  E(clEnqueueWriteBuffer(commands, nf,  CL_TRUE, 0, nf_bytes,  nf_p,
      0, 0, 0));
  if (dex_p) E(clEnqueueWriteBuffer(commands, dex, CL_TRUE, 0, dex_bytes, dex_p,
      0, 0, 0));
  if (dey_p) E(clEnqueueWriteBuffer(commands, dey, CL_TRUE, 0, dey_bytes, dey_p,
      0, 0, 0));

  // compile formula or retrieve kernel from cache
  cl_event formula_executed;
  clformula *formula = nullptr;
  for (int i = 0; i < formulas.size(); ++i)
  {
    if (   formulas[i].type == type
        && formulas[i].fractalType == m_nFractalType
        && formulas[i].power == m_nPower
        && formulas[i].derivatives == derivatives
       )
    {
      formula = &formulas[i];
    }
  }
  if (formula == nullptr)
  {
    // build program
    std::string source = perturbation_opencl(m_nFractalType, m_nPower,
        derivatives);
    std::string name =
      type == 0 ? "perturbation_double"    :
      type == 2 ? "perturbation_floatexp" :
      "#error unknown type\n";
    const char *src = source.c_str();
```

```
      program = clCreateProgramWithSource(context, 1, &src, 0, &err);
      if (! program) { E(err); }
      // FIXME synchronous program building, async would be better
      std::ostringstream options;
      options << "-Werror -D DERIVATIVES=" << (derivatives ? 1 : 0)
              << " -D MAX_APPROX_TERMS=" << MAX_APPROX_TERMS;
      err = clBuildProgram(program, 1, &device_id, options.str().c_str(), 0, 0);
      if (err != CL_SUCCESS) {
        char *buf = (char *) malloc(1000000);
        buf[0] = 0;
        E(clGetProgramBuildInfo(program, device_id, CL_PROGRAM_BUILD_LOG, 1000000,↙
            ↪ &buf[0], 0));
        fprintf(stderr, "source:\n%s\nbuild failed:\n%s\n", source.c_str(), buf);
        free(buf);
        E(err);
      }
      cl_kernel kernel = clCreateKernel(program, name.c_str(), &err);
      if (! kernel) { E(err); }
      clformula newformula = { type, m_nFractalType, m_nPower, derivatives, kernel↙
          ↪ };
      formulas.push_back(newformula);
      formula = &formulas[ssize_t(formulas.size()) - 1];
    }
    assert(formula);

    // execute formula
    E(clSetKernelArg(formula->kernel, 0, sizeof(cl_mem), &config));
    E(clSetKernelArg(formula->kernel, 1, sizeof(cl_mem), &refx));
    E(clSetKernelArg(formula->kernel, 2, sizeof(cl_mem), &refy));
    E(clSetKernelArg(formula->kernel, 3, sizeof(cl_mem), &refz));
    E(clSetKernelArg(formula->kernel, 4, sizeof(cl_mem), n1_p  ? &n1  : nullptr));
    E(clSetKernelArg(formula->kernel, 5, sizeof(cl_mem), n0_p  ? &n0  : nullptr));
    E(clSetKernelArg(formula->kernel, 6, sizeof(cl_mem), nf_p  ? &nf  : nullptr));
    E(clSetKernelArg(formula->kernel, 7, sizeof(cl_mem), dex_p ? &dex : nullptr));
    E(clSetKernelArg(formula->kernel, 8, sizeof(cl_mem), dey_p ? &dey : nullptr));
    size_t global[2] = { m_nY, m_nX };
    cl_event uploaded[4] =
      { config_uploaded
      , refx_uploaded
      , refy_uploaded
      , refz_uploaded
      };
    E(clEnqueueNDRangeKernel(commands, formula->kernel, 2, nullptr, global, ↙
        ↪ nullptr, 4, uploaded, &formula_executed));

    // download results
    // FIXME synchronous, async would be better
    if (n1_p)  E(clEnqueueReadBuffer(commands, n1,  CL_TRUE, 0, n1_bytes,  n1_p,  ↙
        ↪ 1, &formula_executed, 0));
    if (n0_p)  E(clEnqueueReadBuffer(commands, n0,  CL_TRUE, 0, n0_bytes,  n0_p,  ↙
        ↪ 1, &formula_executed, 0));
    if (nf_p)  E(clEnqueueReadBuffer(commands, nf,  CL_TRUE, 0, nf_bytes,  nf_p,  ↙
        ↪ 1, &formula_executed, 0));
    if (dex_p) E(clEnqueueReadBuffer(commands, dex, CL_TRUE, 0, dex_bytes, dex_p, ↙
        ↪ 1, &formula_executed, 0));
    if (dey_p) E(clEnqueueReadBuffer(commands, dey, CL_TRUE, 0, dey_bytes, dey_p, ↙
        ↪ 1, &formula_executed, 0));
```

```
    // clean up reference
    clReleaseMemObject(refz);
    clReleaseMemObject(refy);
    clReleaseMemObject(refx);

    unlock();
}

template void OpenCL::run<double>(
    // for pixel -> parameter mapping
    int32_t m_nX,
    int32_t m_nY,
    uint32_t JitterSeed,
    int32_t JitterShape,
    double JitterScale,
    floatexp m_pixel_step_x,
    floatexp m_pixel_step_y,
    floatexp m_pixel_center_x,
    floatexp m_pixel_center_y,
    double m_C,
    double m_S,
    // for result -> output mapping
    int64_t stride_y,
    int64_t stride_x,
    int64_t stride_offset,
    // for iteration control
    double m_nBailout,
    double m_nBailout2,
    double log_m_nPower,
    int64_t m_nGlitchIter,
    int64_t m_nMaxIter,
    int64_t nMaxIter,
    int64_t nMinIter,
    int16_t m_bNoGlitchDetection,
    int16_t m_bAddReference,
    int16_t m_nSmoothMethod,
    double g_real,
    double g_imag,
    double g_FactorAR,
    double g_FactorAI,
    double m_epsilon,
    // for series approximation
    double m_dPixelSpacing,
    floatexp m_fPixelSpacing,
    int64_t m_nMaxApproximation,
    int32_t m_nApproxTerms,
    int32_t approximation_type,
    floatexp *APr,
    floatexp *APi,
    SeriesR2 *APs,

    // reference orbit
    const double *rx,
    const double *ry,
    const double *rz,
    size_t roffset,
```

```
    size_t rcount,

    // formula selection
    int type,
    int m_nFractalType,
    int m_nPower,
    int16_t derivatives,

    // output arrays
    uint32_t *n1_p,
    uint32_t *n0_p,
    float *nf_p,
    float *dex_p,
    float *dey_p
);

template void OpenCL::run<floatexp>(
    // for pixel -> parameter mapping
    int32_t m_nX,
    int32_t m_nY,
    uint32_t JitterSeed,
    int32_t JitterShape,
    double JitterScale,
    floatexp m_pixel_step_x,
    floatexp m_pixel_step_y,
    floatexp m_pixel_center_x,
    floatexp m_pixel_center_y,
    double m_C,
    double m_S,
    // for result -> output mapping
    int64_t stride_y,
    int64_t stride_x,
    int64_t stride_offset,
    // for iteration control
    double m_nBailout,
    double m_nBailout2,
    double log_m_nPower,
    int64_t m_nGlitchIter,
    int64_t m_nMaxIter,
    int64_t nMaxIter,
    int64_t nMinIter,
    int16_t m_bNoGlitchDetection,
    int16_t m_bAddReference,
    int16_t m_nSmoothMethod,
    double g_real,
    double g_imag,
    double g_FactorAR,
    double g_FactorAI,
    double m_epsilon,
    // for series approximation
    double m_dPixelSpacing,
    floatexp m_fPixelSpacing,
    int64_t m_nMaxApproximation,
    int32_t m_nApproxTerms,
    int32_t approximation_type,
    floatexp *APr,
    floatexp *APi,
```

```
    SeriesR2 *APs,

    // reference orbit
    const floatexp *rx,
    const floatexp *ry,
    const double *rz,
    size_t roffset,
    size_t rcount,

    // formula selection
    int type,
    int m_nFractalType,
    int m_nPower,
    int16_t derivatives,

    // output arrays
    uint32_t *n1_p,
    uint32_t *n0_p,
    float *nf_p,
    float *dex_p,
    float *dey_p
);

#endif
```

15 cl/opencl.h

```
#ifdef KF_OPENCL
#ifndef KF_OPENCL_H
#define KF_OPENCL_H 1

#include <windows.h>
#include <stdint.h>
#include <string>
#include <vector>

#include "../fraktal_sft/floatexp.h"
struct SeriesR2;

#define CL_USE_DEPRECATED_OPENCL_1_2_APIS
#define clew_STATIC
#include "clew.h"

extern std::string perturbation_opencl
( int m_nFractalType
, int m_nPower
, int derivatives
);

typedef struct __attribute__((packed))
{
  size_t BYTES;
  // for pixel -> parameter mapping
  int32_t m_nX;
  int32_t m_nY;
  uint32_t JitterSeed;
  int32_t JitterShape;
```

```
  double JitterScale;
  floatexp m_pixel_step_x;
  floatexp m_pixel_step_y;
  floatexp m_pixel_center_x;
  floatexp m_pixel_center_y;
  double m_C;
  double m_S;
  // for result -> output mapping
  int64_t stride_y;
  int64_t stride_x;
  int64_t stride_offset;
  // for iteration control
  double m_nBailout;
  double m_nBailout2;
  double log_m_nPower;
  int64_t m_nGlitchIter;
  int64_t m_nMaxIter;
  int64_t nMaxIter;
  int64_t nMinIter;
  int16_t m_bNoGlitchDetection;
  int16_t derivatives;
  int16_t m_bAddReference;
  int16_t m_nSmoothMethod;
  double g_real;
  double g_imag;
  double g_FactorAR;
  double g_FactorAI;
  double m_epsilon;
  // for series approximation
  double m_dPixelSpacing;
  floatexp m_fPixelSpacing;
  int64_t m_nMaxApproximation;
  int32_t m_nApproxTerms;
  int32_t approximation_type;
  // 130kB data follows
  floatexp m_APr[MAX_APPROX_TERMS + 1];
  floatexp m_APi[MAX_APPROX_TERMS + 1];
  floatexp m_APs_s[MAX_APPROX_TERMS + 1][MAX_APPROX_TERMS + 1];
  floatexp m_APs_t[MAX_APPROX_TERMS + 1][MAX_APPROX_TERMS + 1];
} p_config;

struct cldevice
{
  cl_platform_id pid;
  cl_device_id did;
  std::string name;
  std::string vendor;
  std::string version;
};

struct clformula
{
  int type;
  int fractalType;
  int power;
  int derivatives;
  cl_kernel kernel;
```

```
};

struct OpenCL
{
private:

  cl_platform_id platform_id;
  cl_device_id device_id;
  cl_context context;
  cl_command_queue commands;
  cl_program program;

  // config (r)
  size_t config_bytes;
  cl_mem config;

  // pixel data (rw)
  size_t n1_bytes;  cl_mem n1;
  size_t n0_bytes;  cl_mem n0;
  size_t nf_bytes;  cl_mem nf;
  size_t dex_bytes; cl_mem dex;
  size_t dey_bytes; cl_mem dey;

  HANDLE mutex;

  // formula kernels
  std::vector<clformula> formulas;

public:

  OpenCL(cl_platform_id platform, cl_device_id device_id);
  ~OpenCL();

  void lock();
  void unlock();

  template <typename T>
  void run(
    // for pixel -> parameter mapping
    int32_t m_nX,
    int32_t m_nY,
    uint32_t JitterSeed,
    int32_t JitterShape,
    double JitterScale,
    floatexp m_pixel_step_x,
    floatexp m_pixel_step_y,
    floatexp m_pixel_center_x,
    floatexp m_pixel_center_y,
    double m_C,
    double m_S,
    // for result -> output mapping
    int64_t stride_y,
    int64_t stride_x,
    int64_t stride_offset,
    // for iteration control
    double m_nBailout,
    double m_nBailout2,
```

```
145         double log_m_nPower,
            int64_t m_nGlitchIter,
            int64_t m_nMaxIter,
            int64_t nMaxIter,
            int64_t nMinIter,
150         int16_t m_bNoGlitchDetection,
            int16_t m_bAddReference,
            int16_t m_nSmoothMethod,
            double g_real,
            double g_imag,
155         double g_FactorAR,
            double g_FactorAI,
            double m_epsilon,
            // for series approximation
            double m_dPixelSpacing,
160         floatexp m_fPixelSpacing,
            int64_t m_nMaxApproximation,
            int32_t m_nApproxTerms,
            int32_t approximation_type,
            floatexp *APr,
165         floatexp *APi,
            SeriesR2 *APs,

            // reference orbit
            const T *rx,
170         const T *ry,
            const double *rz,
            size_t roffset,
            size_t rcount,

175         // formula selection
            int type,
            int m_nFractalType,
            int m_nPower,
            int16_t derivatives,
180
            // output arrays
            uint32_t *n1_p,
            uint32_t *n0_p,
            float *nf_p,
185         float *dex_p,
            float *dey_p
        );
    };

190 std::vector<cldevice> initialize_opencl();

    #endif
    #endif
```

16 cl/opencl.xsl

```
    <?xml version="1.0" encoding="UTF-8"?>
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="text" />
    <xsl:template match="/">
5
```

```
#define P 0
#define P2 2
#define P3 3
#define P4 4
#define P5 5
#define P6 6
#define P7 7
#define P8 8
#define P9 9
#define P10 10

std::vector&lt;clformula&gt; compile_kernels(cl_program program)
{
  cl_int err;
  std::vector&lt;clformula&gt; fs;

<xsl:for-each select="formulas/group/formula">
  {
    cl_kernel kd = clCreateKernel(program, "perturbation_double_<xsl:value-of ↙
        ↳ select="../@type" />_<xsl:value-of select="@power" />", &amp;err);
    if (! kd) { E(err); }
    cl_kernel kfe = clCreateKernel(program, "perturbation_floatexp_<xsl:value-of↙
        ↳  select="../@type" />_<xsl:value-of select="@power" />", &amp;err);
    if (! kfe) { E(err); }
    clformula f = { <xsl:value-of select="../@type" />, P<xsl:value-of select="↙
        ↳ @power" />, kd, kfe };
    fs.push_back(f);
  }
</xsl:for-each>

  return fs;
}

</xsl:template>
</xsl:stylesheet>
```

17 cl/s2c.sh

```
#!/bin/bash
echo "/* machine-generated file, do not edit */"
echo "const char *$1 ="
sed 's|//.*||' |
sed 's|^|"|' |
sed 's|$|\\n"|'
echo ";"
```

18 cl/softfloat_post.cl

```
// ...continued
    xr = xrn;
    xi = xin;
  }
  l->antal = antal;
  l->test1 = test1;
  l->test2 = test2;
  l->xr = Xxr;
```

```
    l->xi = Xxi;
    l->dr = zero;
    l->di = zero;
  }
```

19 cl/softfloat_post_m.cl

```
  // ...continued
      xr = xrn;
      xi = xin;
      dxa = dxan;
      dxb = dxbn;
      dya = dyan;
      dyb = dybn;
    }
    l->antal = antal;
    l->test1 = test1;
    l->test2 = test2;
    l->xr = Xxr;
    l->xi = Xxi;
    l->dr = sf_to_double(sf_add(sf_mul(Xxr, dxa), sf_mul(Xxi, dya))) / sqrt(↙
        ↪ sf_to_double(test1));
    l->di = sf_to_double(sf_add(sf_mul(Xxr, dxb), sf_mul(Xxi, dyb))) / sqrt(↙
        ↪ sf_to_double(test1));
  }
```

20 cl/softfloat_post_rc.cl

```
  // ...continued
      xr = xrn;
      xi = xin;
      dr = drn;
      di = din;
    }
    l->antal = antal;
    l->test1 = test1;
    l->test2 = test2;
    l->xr = Xxr;
    l->xi = Xxi;
    l->dr = dr;
    l->di = di;
  }
```

21 cl/softfloat_pre.cl

```
  void perturbation_softfloat_loop
  ( __global const p_config    *g
  , __global const softfloat   *m_db_dxr
  , __global const softfloat   *m_db_dxi
  , __global const softfloat   *m_db_z
  ,               p_status_sf *l
  )
  {
    const softfloat zero = sf_zero();
    const softfloat Ar = sf_from_double(g->g_FactorAR);
    const softfloat Ai = sf_from_double(g->g_FactorAI);
```

```
  bool no_g = g->g_real == 1.0 && g->g_imag == 1.0;
  const softfloat cr = l->cr;
  const softfloat ci = l->ci;
  softfloat xr = l->xr;
  softfloat xi = l->xi;
  // type C derivatives
  softfloat dr = l->dr;
  softfloat di = l->di;
  // type M derivatives
  softfloat daa = l->daa;
  softfloat dab = l->dab;
  softfloat dba = l->dba;
  softfloat dbb = l->dbb;
  softfloat dxa = dr;
  softfloat dxb = sf_neg(di);
  softfloat dya = di;
  softfloat dyb = dr;
  softfloat g_real = sf_from_double(g->g_real);
  softfloat g_imag = sf_from_double(g->g_imag);
  softfloat m_nBailout2 = sf_from_double(g->m_nBailout2);
  // conditions
  softfloat test1 = l->test1;
  softfloat test2 = l->test2;
  long antal = l->antal;
  softfloat Xxr = zero;
  softfloat Xxi = zero;
  for (; antal < g->nMaxIter; antal++)
  {
    const softfloat Xr = m_db_dxr[antal - g->nMinIter];
    const softfloat Xi = m_db_dxi[antal - g->nMinIter];
    const softfloat Xz = m_db_z  [antal - g->nMinIter];
    Xxr = sf_add(Xr, xr);
    Xxi = sf_add(Xi, xi);
    test2 = test1;
    if (no_g)
    {
      test1 = sf_add(sf_sqr(Xxr), sf_sqr(Xxi));
    }
    else
    {
      test1 = sf_add(sf_mul(g_real, sf_sqr(Xxr)), sf_mul(g_imag, sf_sqr(Xxi)));
    }
    if (sf_lt(test1, Xz))
    {
      l->bGlitch = true;
      if (! g->m_bNoGlitchDetection)
        break;
    }
    if (sf_gt(test1, m_nBailout2))
    {
      break;
    }
    const sfcomplex X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi};
    softfloat xrn = zero;
    softfloat xin = zero;
    sfcomplex xn = {zero, zero};
    const sfcomplex d = {dr, di}, d0 = {daa, dba}; // FIXME matrix derivatives
```

```
    softfloat drn = zero, din = zero;
    sfcomplex dn = {zero, zero};
    softfloat dxan = zero, dxbn = zero, dyan = zero, dybn = zero;
// continued...
```

22 common/barrier.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef BARRIER_H
#define BARRIER_H 1

#include <windows.h>

// centralized barrier from
// https://www.jlab.org/hpc/papers/hpcasia07.pdf

class barrier
{

private:

  volatile LONG release;
  LONG padding1[63];
  volatile LONG counter;
  LONG padding2[63];
  LONG num_thread;
  bool yielding;

public:

  inline barrier(LONG n)
  : release(0), counter(n), num_thread(n)
  {
    SYSTEM_INFO sysinfo;
    GetSystemInfo(&sysinfo);
    yielding = LONG(sysinfo.dwNumberOfProcessors) < num_thread;
  };

  inline bool wait(volatile bool *stop)
```

```
  {
    LONG flag = release;
    LONG count = InterlockedDecrement(&counter);
    if (count == 0)
    {
      counter = num_thread;
      InterlockedIncrement(&release);
    }
    else
      while (flag == release)
      {
        if (*stop)
          return true;
        if (yielding)
          SwitchToThread();
      }
    return false;
  };

};

#endif
```

23 common/bitmap.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <iostream>
#include "bitmap.h"

// based on: https://source.winehq.org/git/wine.git/blob/39935↙
    ↪ fe5ad889d537d828cc82771bdb969cdb2d4:/dlls/gdi32/gdi_private.h#l561
static inline long long get_bitmap_stride(long long width, long long bpp)
{
  return ((width * bpp + 15LL) >> 3LL) & ~1LL;
}

// workaround https://bugs.winehq.org/show_bug.cgi?id=42727
// "obsolete 128MB bitmap size limit"
// see also: http://www.fractalforums.com/windows-fractal-software/windows-↙
```

```
    ↳ bitmap-size-test/
HBITMAP create_bitmap(HDC hdc, int width, int height)
{
  long long stride = get_bitmap_stride(width, 24LL);
  long long bytes = stride * height;
  if (bytes >= 2LL * 1024LL * 1024LL * 1024LL || bytes <= 0LL)
  {
    std::cerr << "ERROR image too large (" << bytes << " bytes)" << std::endl;
    return 0;
  }
  BITMAPINFO bmi;
  bmi.bmiHeader.biSize = sizeof(bmi);
  bmi.bmiHeader.biWidth = width;
  bmi.bmiHeader.biHeight = height;
  bmi.bmiHeader.biPlanes = 1;
  bmi.bmiHeader.biBitCount = 24;
  bmi.bmiHeader.biCompression = BI_RGB;
  bmi.bmiHeader.biSizeImage = bytes;
  bmi.bmiHeader.biXPelsPerMeter = 2835; // 72 dpi
  bmi.bmiHeader.biYPelsPerMeter = 2835; // 72 dpi
  bmi.bmiHeader.biClrUsed = 0;
  bmi.bmiHeader.biClrImportant = 0;
  return CreateDIBSection(hdc, &bmi, DIB_RGB_COLORS, 0, 0, 0);
}
```

24 common/bitmap.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef BITMAP_H
#define BITMAP_H 1

#include <windows.h>

HBITMAP create_bitmap(HDC hdc, int width, int height);

#endif
```

25 common/FolderBrowser.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

// Kalles Fraktaler 2
//
// © 2014 Karl Runmo ,runmo@hotmail.com
//
// COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT ↙
    ↘ WARRANTY
// OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ↙
    ↘ WARRANTIES
// THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR ↙
    ↘ PURPOSE
// OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE ↙
    ↘ COVERED
// CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU↙
    ↘  (NOT
// THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY ↙
    ↘ NECESSARY
// SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ↙
    ↘ ESSENTIAL
// PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER ↙
    ↘ EXCEPT UNDER
// THIS DISCLAIMER.

#include <shlobj.h>
#include "FolderBrowser.h"

char g_szFolder[MAX_PATH]={0};
int g_nInitialized;

#ifdef _WIN64
#define GCL_WNDPROC -24
#define GWL_WNDPROC -4
#endif

int WINAPI lpfnEdit(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
        if(uMsg==WM_DESTROY)
                GetWindowText(hWnd,g_szFolder,MAX_PATH);
        return CallWindowProc((WNDPROC)GetClassLongPtr(hWnd,GCLP_WNDPROC),hWnd,↙
```

```
            ↳ uMsg,wParam,lParam);
}

int WINAPI lpfnCallBack(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
        (void) lParam;
        HWND hEdit;

        if(uMsg==BFFM_INITIALIZED){
                hEdit=CreateWindowEx(WS_EX_CLIENTEDGE,"edit",NULL,WS_CHILD|↙
                    ↳ WS_VISIBLE|ES_AUTOHSCROLL|WS_TABSTOP,50,10,257,20,hWnd,(↙
                    ↳ HMENU)1221,GetModuleHandle(NULL),0);
                SetWindowLongPtr(hEdit,GWLP_WNDPROC,(LONG_PTR)lpfnEdit);
                SendMessage(hEdit,WM_SETFONT,(WPARAM)GetStockObject(↙
                    ↳ ANSI_VAR_FONT),MAKELPARAM(TRUE, 0));

                SendMessage(hWnd,BFFM_SETSELECTION ,1,(LPARAM) g_szFolder);
                SetWindowText(hEdit,g_szFolder);
                g_nInitialized=1;
                EnableWindow(GetDlgItem(hWnd,IDOK),TRUE);
        }
        if(uMsg==BFFM_SELCHANGED && g_nInitialized){
                SHGetPathFromIDList((LPITEMIDLIST)wParam,g_szFolder);
                SetWindowText(GetDlgItem(hWnd,1221),g_szFolder);
        }
        return 0;
}

intptr_t Browse(HWND hWnd, std::string &szFolder)
{
        if(szFolder[0])
                strcpy(g_szFolder,szFolder.c_str());
        BROWSEINFO bi={0};
        char szDisplayName[MAX_PATH];
        char szPathName[MAX_PATH];
        bi.hwndOwner=hWnd;
        bi.pszDisplayName=szDisplayName;
        bi.lpszTitle="Folder:";
        bi.ulFlags=BIF_RETURNONLYFSDIRS;
        bi.lpfn=(BFFCALLBACK)lpfnCallBack;
        bi.lParam=(LPARAM)szPathName;
        g_nInitialized=0;
        intptr_t nRet = (intptr_t)SHBrowseForFolderA(&bi);
        if(!nRet)
                return 0;
        szFolder = g_szFolder;
        return nRet;
}

int BrowseFile(HWND hwParent,BOOL bOpen,const char *szTitle,const char *szExt,↙
    ↳ std::string &szFile)
{
        char buffer[1024] = { 0 };
        strncpy(buffer, szFile.c_str(), sizeof(buffer));
        buffer[sizeof(buffer) - 1] = 0;
        OPENFILENAME ofn={sizeof(OPENFILENAME)};
        ofn.hInstance = GetModuleHandle(NULL);
```

```
	ofn.lpstrFile = buffer;
	ofn.lpstrTitle = szTitle;
	ofn.nMaxFile = sizeof(buffer);
	ofn.hwndOwner = hwParent;
	ofn.lpstrFilter = szExt;
	ofn.nFilterIndex = 1;
	if(bOpen){
		ofn.Flags = OFN_SHOWHELP | OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
		int ret = GetOpenFileName(&ofn);
		if (ret)
		{
			szFile = buffer;
		}
		return ret;
	}
	else{
		ofn.Flags = OFN_SHOWHELP | OFN_OVERWRITEPROMPT | OFN_HIDEREADONLY;
		if(GetSaveFileName(&ofn)){
			szFile = buffer;
			char *s = strrchr(buffer,'\\');
			if(!s)
				s = buffer;
			else
				s++;
			if(!strrchr(s,'.')){
				const char *e = szExt;
				e+=strlen(e)+1;
				if(*e){
					for(int i=1;i<(int)ofn.nFilterIndex;i++){
						e+=strlen(e)+1;
						if(!*e)
							break;
						e+=strlen(e)+1;
						if(!*e)
							break;
					}
				}
				if(strstr(e,"."))
					e = strstr(e,".");
				szFile += e;
			}
			return 1;
		}
		else
			return 0;
	}
}
```

26 common/FolderBrowser.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen
```

```
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #ifndef KF_FOLDERBROWSER_H
    #define KF_FOLDERBROWSER_H 1

    #include <cinttypes>
    #include <string>
25
    #include <windows.h>

    intptr_t Browse(HWND hWnd, std::string &szFolder);
    int BrowseFile(HWND hwParent,BOOL bOpen, const char *szTitle,const char *szExt,↙
        ↳ std::string &szFile);
30
    #endif
```

27 common/getimage.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Runmo
    Copyright (C) 2017-2018 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  // Kalles Fraktaler 2
    //
    // © 2014 Karl Runmo ,runmo@hotmail.com
    //
    // COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT ↙
        ↳ WARRANTY
25  // OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ↙
```

```
    ↪ WARRANTIES
// THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR ↙
    ↪ PURPOSE
// OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE ↙
    ↪ COVERED
// CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU↙
    ↪  (NOT
// THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY ↙
    ↪ NECESSARY
// SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ↙
    ↪ ESSENTIAL
// PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER ↙
    ↪ EXCEPT UNDER
// THIS DISCLAIMER.

#include "getimage.h"

#include <stdio.h>
#define sprintf_s snprintf
#include <comdef.h>

#include "bitmap.h"

HBITMAP GetImageFromData(char *szImgData,int nImgData)
{
        BITMAP bm;
        HBITMAP bmBmp, bmRet, bmBmpOld, bmRetOld;
        HDC dcBmp, dcRet, hDC;
        HRESULT hr;
        char *szData;
        IStream *pStream;
        IPicture *pPicture;
        HANDLE hMem;

        hMem = GlobalAlloc(GMEM_FIXED,nImgData);
        szData = (char*)GlobalLock(hMem);
        memcpy(szData,szImgData,nImgData);

        if(FAILED(hr = CreateStreamOnHGlobal(hMem,FALSE,&pStream))){
                GlobalFree(hMem);
                return NULL;
        }
        hr = OleLoadPicture(pStream,nImgData,TRUE,IID_IPicture,(void**)&pPicture↙
            ↪ );

        pStream->Release();
        GlobalFree(hMem);
        if(FAILED(hr))
                return NULL;

        if(FAILED(hr = pPicture->get_Handle((OLE_HANDLE*)&bmBmp))){
                pPicture->Release();
                return NULL;
        }
        GetObject(bmBmp,sizeof(BITMAP),&bm);

        hDC = GetDC(NULL);
```

```
75          bmRet = create_bitmap(hDC,bm.bmWidth,bm.bmHeight);
            dcBmp = CreateCompatibleDC(hDC);
            dcRet = CreateCompatibleDC(hDC);
            bmBmpOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
            bmRetOld = (HBITMAP)SelectObject(dcRet,bmRet);
80          BitBlt(dcRet,0,0,bm.bmWidth,bm.bmHeight,dcBmp,0,0,SRCCOPY);
            SelectObject(dcBmp,bmBmpOld);
            SelectObject(dcRet,bmRetOld);
            DeleteDC(dcBmp);
            DeleteDC(dcRet);
85          ReleaseDC(NULL,hDC);
            pPicture->Release();

            return bmRet;
    }
90  void UpsideBitmap(HBITMAP bmBitmap)
    {
            HDC hDC = GetDC(NULL);
            BYTE *lpBits, *lpBitsNew;
            int row;
95          BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
            if(!GetDIBits(hDC,bmBitmap,0,0,NULL,(LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                    Beep(1000,10);
            bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
            bmi.biBitCount = 24;
100         row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
            bmi.biSizeImage=row*bmi.biHeight;
            lpBitsNew = new BYTE[bmi.biSizeImage];
            lpBits = new BYTE[bmi.biSizeImage];
            if(!GetDIBits(hDC,bmBitmap,0,bmi.biHeight,lpBits,
105                         (LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                    Beep(1000,10);
            int nX, nY;
            for(nX=0;nX<bmi.biWidth;nX++)
                    for(nY=0;nY<bmi.biHeight;nY++){
110                         lpBitsNew[nX*3 + (bmi.biHeight-nY-1)*row] = lpBits[nX*3 ↙
                                ↳ + nY*row];
                            lpBitsNew[nX*3 + (bmi.biHeight-nY-1)*row + 1] = lpBits[↙
                                ↳ nX*3 + nY*row + 1];
                            lpBitsNew[nX*3 + (bmi.biHeight-nY-1)*row + 2] = lpBits[↙
                                ↳ nX*3 + nY*row + 2];
                    }
            if(!SetDIBits(hDC,bmBitmap,0,bmi.biHeight,lpBitsNew,
115                         (LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                    Beep(1000,10);
            delete lpBitsNew;
            delete lpBits;
    }
120 HBITMAP GetImage(const std::string &szFile)
    {
            BITMAP bm;
            HBITMAP bmBmp, bmRet, bmBmpOld, bmRetOld;
            HDC dcBmp, dcRet, hDC;
125         HRESULT hr;
            int nLen;
            char *szData;
            IStream *pStream;
```

```
	IPicture *pPicture;
	DWORD dwRead;
	HANDLE hFile, hMem;

	if((hFile = CreateFileA(szFile.c_str(),GENERIC_READ,0,NULL,OPEN_EXISTING,0,NULL))==INVALID_HANDLE_VALUE)
		return NULL;
	nLen = GetFileSize(hFile,NULL);
	hMem = GlobalAlloc(GMEM_FIXED,nLen);
	szData = (char*)GlobalLock(hMem);
	ReadFile(hFile,szData,nLen,&dwRead,NULL);
	CloseHandle(hFile);

	if(FAILED(hr = CreateStreamOnHGlobal(hMem,FALSE,&pStream))){
		GlobalFree(hMem);
		return (HBITMAP)LoadImageA(GetModuleHandle(NULL),szFile.c_str(),IMAGE_BITMAP,0,0,LR_LOADFROMFILE);
	}
	hr = OleLoadPicture(pStream,nLen,TRUE,IID_IPicture,(void**)&pPicture);

	pStream->Release();
	GlobalFree(hMem);
	if(FAILED(hr))
		return (HBITMAP)LoadImageA(GetModuleHandle(NULL),szFile.c_str(),IMAGE_BITMAP,0,0,LR_LOADFROMFILE);

	short nType;
	pPicture->get_Type(&nType);
	if(nType==2){
		short nType;
		long nWidth, nHeight;
		pPicture->get_Type(&nType);
		pPicture->get_Width(&nWidth);
		pPicture->get_Height(&nHeight);
		hDC = GetDC(NULL);
		HDC dcBmp = CreateCompatibleDC(hDC);
		HBITMAP bmBmp = create_bitmap(hDC,nWidth/4,nHeight/4);
		HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
		RECT r={0,0,nWidth/4,nHeight/4};
		FillRect(dcBmp,&r,(HBRUSH)GetStockObject(WHITE_BRUSH));
		pPicture->Render(dcBmp,0,0,nWidth/4,nHeight/4,0,nHeight,nWidth,-nHeight,&r);

		SelectObject(dcBmp,bmOld);
		DeleteDC(dcBmp);
		ReleaseDC(NULL,hDC);
		pPicture->Release();
		return bmBmp;
	}
	if(FAILED(hr = pPicture->get_Handle((OLE_HANDLE*)&bmBmp))){
		pPicture->Release();
		return (HBITMAP)LoadImageA(GetModuleHandle(NULL),szFile.c_str(),IMAGE_BITMAP,0,0,LR_LOADFROMFILE);
	}
	GetObject(bmBmp,sizeof(BITMAP),&bm);

	hDC = GetDC(NULL);
```

```
        bmRet = create_bitmap(hDC,bm.bmWidth,bm.bmHeight);
        dcBmp = CreateCompatibleDC(hDC);
        dcRet = CreateCompatibleDC(hDC);
        bmBmpOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
185     bmRetOld = (HBITMAP)SelectObject(dcRet,bmRet);
        BitBlt(dcRet,0,0,bm.bmWidth,bm.bmHeight,dcBmp,0,0,SRCCOPY);
        SelectObject(dcBmp,bmBmpOld);
        SelectObject(dcRet,bmRetOld);
        DeleteDC(dcBmp);
190     DeleteDC(dcRet);
        ReleaseDC(NULL,hDC);
        pPicture->Release();

        return bmRet;
195 }

HBITMAP GetImageFromResource(char *szResourceType, char *szResourceName,↙
    ↪ HINSTANCE hInstance)
{
        (void) hInstance;
200     HGLOBAL hGlobal;
        HRSRC hResource;
        if(!(hResource=FindResourceA(GetModuleHandle(NULL),szResourceName,↙
            ↪ szResourceType)))
                return NULL;
        if(!(hGlobal=LoadResource(GetModuleHandle(NULL),hResource)))
205             return NULL;
        int nLen = SizeofResource(GetModuleHandle(NULL),hResource);
        char *szData = (char*)GlobalLock(hGlobal);
        return GetImageFromData(szData,nLen);
}
210 void FillRectShade(HDC hDC, RECT r, int nR1, int nG1, int nB1,int nR2, int nG2, ↙
    ↪ int nB2,int nType)
{
        RECT rFill=r;
        HBRUSH br;
        int i, nStep = (r.right-r.left>r.bottom-r.top?r.right-r.left:r.bottom-r.↙
            ↪ top);
215     if(nType>3){
                HRGN rr = CreateRectRgn(r.left,r.top,r.right,r.bottom);
                int nWidth = r.right-r.left;
                int nHeight = r.bottom-r.top;
                r.left-=nWidth/4;
220             r.right+=nWidth/4;
                r.top-=nHeight/4;
                r.bottom+=nHeight/4;
                rFill = r;
                nStep = (r.right-r.left>r.bottom-r.top?r.right-r.left:r.bottom-r↙
                    ↪ .top);
225             for(i=0;i<=nStep;i++){
                        if((nType & 1)==0)
                                rFill.right = r.right - (r.right-r.left)*i/nStep↙
                                    ↪ ;
                        else
                                rFill.right = r.right;
230                     if((nType & 2)==0)
                                rFill.bottom = r.bottom - (r.bottom-r.top)*i/↙
```

```
                                        ↪ nStep;
                                else
                                        rFill.bottom = r.bottom;
                                HRGN rgn = CreateEllipticRgn(rFill.left,rFill.top,rFill.↙
                                    ↪ right,rFill.bottom);
235                             CombineRgn(rgn,rgn,rr,RGN_AND);
                                br = CreateSolidBrush(RGB(nR2 - (nR2-nR1)*i/nStep,nG2 - ↙
                                    ↪ (nG2-nG1)*i/nStep,nB2 - (nB2-nB1)*i/nStep));
                                FillRgn(hDC,rgn,br);
//                              FillRect(hDC,&rFill,br);
                                DeleteObject(br);
240                             DeleteObject(rgn);
                        }
                        return;
                }
                for(i=0;i<=nStep;i++){
245                     if((nType & 1)==0)
                                rFill.right = r.right - (r.right-r.left)*i/nStep;
                        else
                                rFill.right = r.right;
                        if((nType & 2)==0)
250                             rFill.bottom = r.bottom - (r.bottom-r.top)*i/nStep;
                        else
                                rFill.bottom = r.bottom;
                        br = CreateSolidBrush(RGB(nR2 - (nR2-nR1)*i/nStep,nG2 - (nG2-nG1↙
                            ↪ )*i/nStep,nB2 - (nB2-nB1)*i/nStep));
                        FillRect(hDC,&rFill,br);
255                     DeleteObject(br);
                }
}

COLORREF GetPixelDIB(HDC hDC, HBITMAP bmBitmap,int x, int y)
260 {
//      return GetPixel(hDC,x,y);
        static BYTE *lpBits=NULL;
        static BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
        static int row;
265
        if(!hDC){
                delete lpBits;
                lpBits=NULL;
                memset(&bmi,0,sizeof(BITMAPINFOHEADER));
270             bmi.biSize = sizeof(BITMAPINFOHEADER);
                return 0;
        }
        else if(!lpBits){
                if(!GetDIBits(hDC,bmBitmap,0,0,NULL,(LPBITMAPINFO)&bmi,↙
                    ↪ DIB_RGB_COLORS))
275                     Beep(100,10);
                bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
                bmi.biBitCount = 24;
                row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
                bmi.biSizeImage=row*bmi.biHeight;
280             lpBits = new BYTE[bmi.biSizeImage];
                if(!GetDIBits(hDC,bmBitmap,0,bmi.biHeight,lpBits,
                                (LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                        Beep(100,10);
```

```
        }
        if(x*3 + (bmi.biHeight-y-1)*row + 2 < (int)bmi.biSizeImage){
                int B = lpBits[x*3 + (bmi.biHeight-y-1)*row], G = lpBits[x*3 + (
                    ↪ bmi.biHeight-y-1)*row + 1], R = lpBits[x*3 + (bmi.biHeight
                    ↪ -y-1)*row + 2];
                return RGB(R,G,B);
        }
        return 0;
}
void SetPixelDIB(HDC hDC, HBITMAP bmBitmap,int x, int y,COLORREF col,BOOL
    ↪ bCreate)
{
//      return GetPixel(hDC,x,y);
        static BYTE *lpBits=NULL;
        static BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
        static int row;

        if(bCreate){
                if(!SetDIBits(hDC,bmBitmap,0,bmi.biHeight,lpBits,
                                (LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                        Beep(100,10);
                delete lpBits;
                lpBits=NULL;
                memset(&bmi,0,sizeof(BITMAPINFOHEADER));
                bmi.biSize = sizeof(BITMAPINFOHEADER);
                return;
        }
        else if(!lpBits){
                if(!GetDIBits(hDC,bmBitmap,0,0,NULL,(LPBITMAPINFO)&bmi,
                    ↪ DIB_RGB_COLORS))
                        Beep(100,10);
                bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
                bmi.biBitCount = 24;
                row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
                bmi.biSizeImage=row*bmi.biHeight;
                lpBits = new BYTE[bmi.biSizeImage];
                if(!GetDIBits(hDC,bmBitmap,0,bmi.biHeight,lpBits,
                                (LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                        Beep(100,10);
        }
        unsigned char *Colors = (unsigned char*)&col;
        if(x*3 + (bmi.biHeight-y-1)*row + 2 < (int)bmi.biSizeImage){
                lpBits[x*3 + (bmi.biHeight-y-1)*row] = Colors[2];
                lpBits[x*3 + (bmi.biHeight-y-1)*row + 1] = Colors[1];
                lpBits[x*3 + (bmi.biHeight-y-1)*row + 2] = Colors[0];
//              lpBits[x*3 + y*row] = Colors[2];
//              lpBits[x*3 + y*row + 1] = Colors[1];
//              lpBits[x*3 + y*row + 2] = Colors[0];
        }
}

void ResizeBitmap(HBITMAP *bmBitmap,int nWidth,int nHeight,int nNewWidth,int
    ↪ nNewHeight,BOOL bHalftone,__int64 *pnData)
{
        int row, rowNew;
        if(pnData)
                *pnData=0;
```

```
	BITMAP bb;
	GetObject(*bmBitmap,sizeof(BITMAP),&bb);
	if(nWidth==-1)
340		nWidth = bb.bmWidth;
	if(nHeight==-1)
		nHeight = bb.bmHeight;

	BYTE *lpBits, *lpBitsNew;
345	BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
	BITMAPINFOHEADER bmiNew={sizeof(BITMAPINFOHEADER)};

	HDC hDC = GetDC(NULL);
	HDC dcNew = CreateCompatibleDC(hDC);
350	HBITMAP bmNew = create_bitmap(hDC,nNewWidth,nNewHeight);
	HBITMAP bmOld = (HBITMAP)SelectObject(dcNew,bmNew);

	if(!GetDIBits(dcNew,bmNew,0,0,NULL,(LPBITMAPINFO)&bmiNew,DIB_RGB_COLORS)
	    ↪ )
		Beep(1000,10);
355	bmiNew.biCompression=bmiNew.biClrUsed=bmiNew.biClrImportant=0;
	bmiNew.biBitCount = 24;
	rowNew = ((((bmiNew.biWidth*(DWORD)bmiNew.biBitCount)+31)&~31) >> 3);
	bmiNew.biSizeImage=rowNew*bmiNew.biHeight;
	lpBitsNew = new BYTE[bmiNew.biSizeImage];
360
	if(!GetDIBits(hDC,*bmBitmap,0,0,NULL,(LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
		Beep(1000,10);
	bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
	bmi.biBitCount = 24;
365	row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
	bmi.biSizeImage=row*bmi.biHeight;
	lpBits = new BYTE[bmi.biSizeImage];
	if(!GetDIBits(hDC,*bmBitmap,0,bmi.biHeight,lpBits,
			(LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
370		Beep(1000,10);
	int nXStart,nYStart;
	int nXNew, nYNew;
	if(bHalftone){
		for(nXNew=0;nXNew<nNewWidth;nXNew++)
375			for(nYNew=0;nYNew<nNewHeight;nYNew++){
				int Colors[3]={0};
				int nDiv=0;
				for(nXStart=nXNew*nWidth/nNewWidth;nXStart<(
				    ↪ nXNew+1)*nWidth/nNewWidth;nXStart++)
					for(nYStart=nYNew*nHeight/nNewHeight;
					    ↪ nYStart<(nYNew+1)*nHeight/
					    ↪ nNewHeight;nYStart++)
380						if(nXStart*3 + nYStart*row + 2 <
						    ↪ (int)bmi.biSizeImage){
							nDiv++;
							Colors[0]+=lpBits[
							    ↪ nXStart*3 +
							    ↪ nYStart*row];
							Colors[1]+=lpBits[
							    ↪ nXStart*3 +
							    ↪ nYStart*row + 1];
```

```
                                        Colors[2]+=lpBits[↙
                                            ↳ nXStart*3 + ↙
                                            ↳ nYStart*row + 2];
385                                 }
                                if(!nDiv){
                                    nXStart=nXNew*nWidth/nNewWidth;
                                    nYStart=nYNew*nHeight/nNewHeight;
                                    Colors[0]+=lpBits[nXStart*3 + nYStart*↙
                                        ↳ row];
390                                 Colors[1]+=lpBits[nXStart*3 + nYStart*↙
                                        ↳ row + 1];
                                    Colors[2]+=lpBits[nXStart*3 + nYStart*↙
                                        ↳ row + 2];
                                    nDiv++;
                                    if(nXNew && nXStart==(nXNew-1)*nWidth/↙
                                        ↳ nNewWidth){
                                        nXNew--;
395                                     Colors[0]+=lpBitsNew[nXNew*3 + ↙
                                            ↳ nYNew*rowNew];
                                        Colors[1]+=lpBitsNew[nXNew*3 + ↙
                                            ↳ nYNew*rowNew + 1];
                                        Colors[2]+=lpBitsNew[nXNew*3 + ↙
                                            ↳ nYNew*rowNew + 2];
                                        nXNew++;
                                        nDiv++;
400                                 }
                                    if(nYNew && nYStart==(nYNew+1)*nHeight/↙
                                        ↳ nNewHeight){
                                        nYNew--;
                                        Colors[0]+=lpBitsNew[nXNew*3 + ↙
                                            ↳ nYNew*rowNew];
                                        Colors[1]+=lpBitsNew[nXNew*3 + ↙
                                            ↳ nYNew*rowNew + 1];
405                                     Colors[2]+=lpBitsNew[nXNew*3 + ↙
                                            ↳ nYNew*rowNew + 2];
                                        nYNew++;
                                        nDiv++;
                                    }
                                if(nXNew*3 + nYNew*rowNew + 2 < (int)bmiNew.↙
                                    ↳ biSizeImage){
410                                     lpBitsNew[nXNew*3 + nYNew*rowNew↙
                                            ↳ ] = Colors[0]/nDiv;
                                        lpBitsNew[nXNew*3 + nYNew*rowNew↙
                                            ↳  + 1] = Colors[1]/nDiv;
                                        lpBitsNew[nXNew*3 + nYNew*rowNew↙
                                            ↳  + 2] = Colors[2]/nDiv;
                                        if(pnData)
                                            *pnData+=lpBitsNew[nXNew↙
                                                ↳ *3 + nYNew*rowNew↙
                                                ↳ ]+lpBitsNew[nXNew↙
                                                ↳ *3 + nYNew*rowNew ↙
                                                ↳ + 1]+lpBitsNew[↙
                                                ↳ nXNew*3 + nYNew*↙
                                                ↳ rowNew + 2];
415                                 }
                                }
                                else{
```

```
                                                Colors[0]/=nDiv;
                                                Colors[1]/=nDiv;
                                                Colors[2]/=nDiv;
                                                if(nXNew*3 + nYNew*rowNew + 2 < (int)
                                                    ↪ bmiNew.biSizeImage){
                                                        lpBitsNew[nXNew*3 + nYNew*rowNew
                                                            ↪ ] = Colors[0];
                                                        lpBitsNew[nXNew*3 + nYNew*rowNew
                                                            ↪  + 1] = Colors[1];
                                                        lpBitsNew[nXNew*3 + nYNew*rowNew
                                                            ↪  + 2] = Colors[2];
                                                        if(pnData)
                                                                *pnData+=lpBitsNew[nXNew
                                                                    ↪ *3 + nYNew*rowNew
                                                                    ↪ ]+lpBitsNew[nXNew
                                                                    ↪ *3 + nYNew*rowNew
                                                                    ↪ + 1]+lpBitsNew[
                                                                    ↪ nXNew*3 + nYNew*
                                                                    ↪ rowNew + 2];
                                                }
                                        }
                                }
        }
        else{
                for(nXNew=0;nXNew<nNewWidth;nXNew++)
                        for(nYNew=0;nYNew<nNewHeight;nYNew++){
                                nXStart=nXNew*nWidth/nNewWidth;
                                nYStart=nYNew*nHeight/nNewHeight;
                                lpBitsNew[nXNew*3 + nYNew*rowNew] = lpBits[
                                    ↪ nXStart*3 + nYStart*row];
                                lpBitsNew[nXNew*3 + nYNew*rowNew + 1] = lpBits[
                                    ↪ nXStart*3 + nYStart*row+1];
                                lpBitsNew[nXNew*3 + nYNew*rowNew + 2] = lpBits[
                                    ↪ nXStart*3 + nYStart*row+2];
                                if(pnData)
                                        *pnData+=lpBitsNew[nXNew*3 + nYNew*
                                            ↪ rowNew]+lpBitsNew[nXNew*3 + nYNew*
                                            ↪ rowNew + 1]+lpBitsNew[nXNew*3 +
                                            ↪ nYNew*rowNew + 2];
                        }
        }
        if(!SetDIBits(hDC,bmNew,0,bmiNew.biHeight,lpBitsNew,
                        (LPBITMAPINFO)&bmiNew,DIB_RGB_COLORS))
                Beep(1000,10);
        delete [] lpBits;
        delete [] lpBitsNew;
        SelectObject(dcNew,bmOld);
        DeleteObject(*bmBitmap);
        *bmBitmap = bmNew;
        DeleteDC(dcNew);
        ReleaseDC(NULL,hDC);
        return;
}

void SkuggadRect(HDC pDC, RECT wr, BOOL bUpp,BOOL bDark, int nCorner)
{
```

```
        HPEN wp=CreatePen(PS_SOLID,1,bDark?RGB(192,192,192):RGB(255,255,255)),
460                 gp=CreatePen(PS_SOLID,1,bDark?RGB(0,0,0):RGB(128,128,128)), op;

        // Rita vänstra och övre med ena färgen
        if(bUpp)
                op = (HPEN)SelectObject(pDC,gp);
465     else
                op = (HPEN)SelectObject(pDC,wp);
        MoveToEx(pDC,wr.left+nCorner,wr.bottom-1,NULL);
        LineTo(pDC,wr.right-1-nCorner,wr.bottom-1);
        LineTo(pDC,wr.right-1,wr.bottom-1-nCorner);
470     LineTo(pDC,wr.right-1,wr.top+nCorner);
        LineTo(pDC,wr.right-1-nCorner,wr.top);

        // Rita högra och undre med andra färgen
        if(bUpp)
475             SelectObject(pDC,wp);
        else
                SelectObject(pDC,gp);
        MoveToEx(pDC,wr.right-1-nCorner,wr.top,NULL);
        LineTo(pDC,wr.left+nCorner,wr.top);
480     LineTo(pDC,wr.left,wr.top+nCorner);
        LineTo(pDC,wr.left,wr.bottom-1-nCorner);
        LineTo(pDC,wr.left+nCorner,wr.bottom-1);

        // Frigör GDI-minne
485     SelectObject(pDC,op);
        DeleteObject(wp);
        DeleteObject(gp);
}

490 void SkuggadCirkle(HDC pDC, RECT wr, BOOL bUpp,BOOL bDark)
{
        HPEN wp=CreatePen(PS_SOLID,1,bDark?RGB(192,192,192):RGB(255,255,255)),
                gp=CreatePen(PS_SOLID,1,bDark?RGB(0,0,0):RGB(128,128,128)), op;

495     // Rita vänstra och övre med ena färgen
        if(bUpp)
                op = (HPEN)SelectObject(pDC,gp);
        else
                op = (HPEN)SelectObject(pDC,wp);
500     Arc(pDC,wr.left,wr.top,wr.right,wr.bottom,wr.left,wr.bottom,wr.right,wr.↙
            ↪ top);
/*      MoveToEx(pDC,wr.left+nCorner,wr.bottom-1,NULL);
        LineTo(pDC,wr.right-1-nCorner,wr.bottom-1);
        LineTo(pDC,wr.right-1,wr.bottom-1-nCorner);
        LineTo(pDC,wr.right-1,wr.top+nCorner);
505     LineTo(pDC,wr.right-1-nCorner,wr.top);
*/
        // Rita högra och undre med andra färgen
        if(bUpp)
                SelectObject(pDC,wp);
510     else
                SelectObject(pDC,gp);
        Arc(pDC,wr.left,wr.top,wr.right,wr.bottom,wr.right,wr.top,wr.left,wr.↙
            ↪ bottom);
/*      MoveToEx(pDC,wr.right-1-nCorner,wr.top,NULL);
```

```
        LineTo(pDC,wr.left+nCorner,wr.top);
        LineTo(pDC,wr.left ,wr.top+nCorner);
        LineTo(pDC,wr.left ,wr.bottom-1-nCorner);
        LineTo(pDC,wr.left+nCorner,wr.bottom-1);
*/
        // Frigör GDI-minne
        SelectObject(pDC,op);
        DeleteObject(wp);
        DeleteObject(gp);
}
```

28 common/getimage.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_GETIMAGE_H
#define KF_GETIMAGE_H 1

#include <string>

#include <windows.h>

HBITMAP GetImageFromData(char *szImgData,int nImgData);
HBITMAP GetImage(const std::string &szFile);
void FillRectShade(HDC hDC, RECT r, int nR1, int nG1, int nB1,int nR2, int nG2, ↙
    ↳ int nB2,int nType=0);
void SkuggadRect(HDC pDC, RECT wr, BOOL bUpp,BOOL bDark, int nCorner=0);
void SkuggadCirkle(HDC pDC, RECT wr, BOOL bUpp,BOOL bDark);

HBITMAP GetImageFromResource(char *szResourceType, char *szResourceName,↙
    ↳ HINSTANCE hInstance);

// Denna funktion skall anropas med hDC=NULL för att frigöra minnet
COLORREF GetPixelDIB(HDC hDC, HBITMAP bmBitmap,int x, int y);
// Denna funktion skall avslutas med bCreate=TRUE för att skapa bilden och ↙
    ↳ frigöra minnet
void SetPixelDIB(HDC hDC, HBITMAP bmBitmap,int x, int y,COLORREF col,BOOL ↙
    ↳ bCreate);

void ResizeBitmap(HBITMAP *bmBitmap,int nWidth,int nHeight,int nNewWidth,int ↙
```

```
    ↳ nNewHeight,BOOL bHalftone=TRUE,__int64 *pnData=NULL);

#endif
```

29 common/matrix.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "matrix.h"

#include <cmath>

mat2 polar_composition(const polar2 &P)
{
  using std::cos;
  using std::sin;
  using glm::transpose;
  const double r = P.rotate;
  const double a = P.stretch_angle;
  const mat2 R( cos(r), -sin(r), sin(r), cos(r) );
  const mat2 S( P.stretch_factor, 0, 0, 1/P.stretch_factor );
  const mat2 T( cos(a), -sin(a), sin(a), cos(a) );
  return R * T * S * transpose(T) * P.scale;
}

polar2 polar_decomposition(const mat2 &M)
{
  using std::sqrt;
  using std::abs;
  using std::atan2;
  using glm::inverse;
  using glm::transpose;
  using glm::determinant;
  const double scale = sqrt(abs(determinant(M)));
  if (scale != 0)
  {
    const mat2 A = M / scale;
    const mat2 B = A + inverse(transpose(A));
    const double b = sqrt(abs(determinant(B)));
    const mat2 V = B / b;
```

```
    const double rotate = atan2(V[1][0], V[0][0]);
    const mat2 P = (transpose(A) * A + mat2(1,0,0,1)) / b;
    // [U,D] = eig(P);
    const double pa = P[0][0];
    const double pb = P[0][1];
    const double pc = P[1][0];
    const double pd = P[1][1];
    const double ptr2 = (pa + pd) / 2;
    const double pdet = pa * pd - pb * pc;
    const double pdisc = ptr2 * ptr2 - pdet;
    if (pdisc > 0)
    {
      const double d1 = ptr2 + sqrt(pdisc);
      const double d2 = ptr2 - sqrt(pdisc);
      double ua, ub, uc, ud;
      if (pb != 0)
      {
        ua = pb;
        ub = pb;
        uc = d1 - pa;
        ud = d2 - pa;
      }
      else if (pc != 0)
      {
        ua = d1 - pd;
        ub = d2 - pd;
        uc = pc;
        ud = pc;
      }
      else
      {
        ua = 1;
        ub = 0;
        uc = 0;
        ud = 1;
      }
      double stretch_factor = d1;
      double stretch_angle = atan2(uc, ua);
      (void) ub;
      (void) ud;
      if (stretch_factor < 1)
      {
        stretch_factor = 1 / stretch_factor;
        stretch_angle += 1.5707963267948966;
      }
      return polar2(scale, rotate, stretch_factor, stretch_angle);
    }
  }
  return polar2();
}
```

30 common/matrix.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen
```

```
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #ifndef MATRIX_H
    #define MATRIX_H 1

    #include <glm/glm.hpp>

25  typedef glm::dvec2 vec2;
    typedef glm::dmat2 mat2;

    struct polar2
    {
30    double scale, rotate, stretch_factor, stretch_angle;
      polar2(double s, double r, double sf, double sa)
      : scale(s)
      , rotate(r)
      , stretch_factor(sf)
35    , stretch_angle(sa)
      { };
      polar2() : polar2(1, 0, 1, 0) { };
    };

40  mat2 polar_composition(const polar2 &P);
    polar2 polar_decomposition(const mat2 &M);

    #endif
```

31 common/parallell.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Runmo
    Copyright (C) 2017-2019 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
```

```
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  // Kalles Fraktaler 2
    //
    // © 2014 Karl Runmo ,runmo@hotmail.com
    //
    // COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT ↙
        ↳ WARRANTY
25  // OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ↙
        ↳ WARRANTIES
    // THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR ↙
        ↳ PURPOSE
    // OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE ↙
        ↳ COVERED
    // CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU↙
        ↳  (NOT
    // THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY ↙
        ↳ NECESSARY
30  // SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ↙
        ↳ ESSENTIAL
    // PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER ↙
        ↳ EXCEPT UNDER
    // THIS DISCLAIMER.
    #include <windows.h>
    #include "parallell.h"
35  #include <mpfr.h>

    ULONG WINAPI Parallell_ThExecute(LPVOID pParameter)
    {
            CParallell::EXECUTE *pE = (CParallell::EXECUTE *)pParameter;
40          SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
            SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
    #ifndef _DEBUG
    try{
    #endif
45          pE->lpfnExecute(pE->pParameter);
    #ifndef _DEBUG
    }catch(...){
    }
    #endif
50          SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_END);
            SetEvent(pE->hDone);
            mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
            return 0;
    }
55
    CParallell::CParallell(int nParallell, int nTimeout,int nInitWait)
    {
            m_dwStackSize = 0;
            m_nParallell = nParallell;
60          m_nTimeout = nTimeout;
            m_nInitWait = nInitWait;
            m_ppExecute = NULL;
            m_nExecute = 0;
            m_lpfnTotalDone = NULL;
```

```
65          m_pDone = NULL;
    }
    CParallell::~CParallell()
    {
            Reset();
70  }
    void CParallell::Reset()
    {
            int i;
            for(i=0;i<m_nExecute;i++)
75                  delete m_ppExecute[i];
            free(m_ppExecute);
            m_nExecute=0;
            m_ppExecute=NULL;
    }
80
    void CParallell::SetTotalDone(LPEXECUTE lpfnTotalDone,LPVOID pDone)
    {
            m_lpfnTotalDone = lpfnTotalDone;
            m_pDone = pDone;
85  }
    int CParallell::AddFunction(LPEXECUTE lpfnExecute,LPVOID pParameter,LPEXECUTE ↙
        ↪ lpfnDone)
    {
            int i = m_nExecute++;
            m_ppExecute = (EXECUTE**)realloc(m_ppExecute,sizeof(EXECUTE*)*m_nExecute↙
                ↪ );
90          m_ppExecute[i] = new EXECUTE;
            m_ppExecute[i]->hThread = NULL;
            m_ppExecute[i]->hDone = CreateEvent(NULL,0,0,NULL);
            m_ppExecute[i]->lpfnExecute = lpfnExecute;
            m_ppExecute[i]->lpfnDone = lpfnDone;
95          m_ppExecute[i]->pParameter = pParameter;
            return m_nExecute;
    }

    void CParallell::SetStackSize(DWORD dwStackSize)
100 {
            m_dwStackSize = dwStackSize;
    }
    int CParallell::Execute()
    {
105         int i, j;
            DWORD dw;
            for(i=0;i<m_nParallell && i<m_nExecute;i++){
                    m_ppExecute[i]->hThread = CreateThread(NULL,m_dwStackSize,↙
                        ↪ Parallell_ThExecute,m_ppExecute[i],0,&dw);
                    Sleep(m_nInitWait);
110                 while(m_ppExecute[i]->hThread==INVALID_HANDLE_VALUE || ↙
                        ↪ m_ppExecute[i]->hThread==NULL){
                            m_ppExecute[i]->hThread = CreateThread(NULL,↙
                                ↪ m_dwStackSize,Parallell_ThExecute,m_ppExecute[i↙
                                ↪ ],0,&dw);
                            Sleep(20);
                    }
            }
115         for(j=0;j<m_nExecute;j++){
```

```
                if(WaitForSingleObject(m_ppExecute[j]->hDone,m_nTimeout)==↙
                    ↳ WAIT_TIMEOUT)
                        TerminateThread(m_ppExecute[j]->hThread,0);
                if(m_ppExecute[j]->lpfnDone)
                        m_ppExecute[j]->lpfnDone(m_ppExecute[j]->pParameter);
                CloseHandle(m_ppExecute[j]->hThread);
                CloseHandle(m_ppExecute[j]->hDone);
                if(i<m_nExecute){
                        m_ppExecute[i]->hThread = CreateThread(NULL,↙
                            ↳ m_dwStackSize,Parallell_ThExecute,m_ppExecute[i↙
                            ↳ ],0,&dw);
                        while(m_ppExecute[i]->hThread==INVALID_HANDLE_VALUE ||↙
                            ↳ m_ppExecute[i]->hThread==NULL){
                                m_ppExecute[i]->hThread = CreateThread(NULL,↙
                                    ↳ m_dwStackSize,Parallell_ThExecute,↙
                                    ↳ m_ppExecute[i],0,&dw);
                                Sleep(20);
                        }
                        i++;
                }
        }
        if(m_lpfnTotalDone)
                m_lpfnTotalDone(m_pDone);
        return 1;
}

int CParallell::ExecuteNoOrder()
{
        int i, j;
        DWORD dw;
        HANDLE *hDone = new HANDLE[m_nParallell];
        int *nIndex = new int[m_nParallell];
        int nEnd=m_nParallell;
        for(i=0;i<m_nParallell && i<m_nExecute;i++){
                hDone[i]=m_ppExecute[i]->hDone;
                nIndex[i]=i;
                m_ppExecute[i]->hThread = CreateThread(NULL,m_dwStackSize,↙
                    ↳ Parallell_ThExecute,m_ppExecute[i],0,&dw);
                Sleep(m_nInitWait);
                while(m_ppExecute[i]->hThread==INVALID_HANDLE_VALUE ||↙
                    ↳ m_ppExecute[i]->hThread==NULL){
                        m_ppExecute[i]->hThread = CreateThread(NULL,↙
                            ↳ m_dwStackSize,Parallell_ThExecute,m_ppExecute[i↙
                            ↳ ],0,&dw);
                        Sleep(20);
                }
                nEnd--;
        }
        j=0;
        while(j<m_nExecute){
                int nWRet;
                if((nWRet=WaitForMultipleObjects(m_nParallell-nEnd,hDone,FALSE,↙
                    ↳ m_nTimeout))==WAIT_TIMEOUT){
                        nWRet=0;
                        TerminateThread(m_ppExecute[nIndex[nWRet]]->hThread,0);
                }
                else{
```

```
                        nWRet-=WAIT_OBJECT_0;
                }
                int k = nIndex[nWRet];
                if(m_ppExecute[k]->lpfnDone)
                        m_ppExecute[k]->lpfnDone(m_ppExecute[k]->pParameter);
                CloseHandle(m_ppExecute[k]->hThread);
                CloseHandle(m_ppExecute[k]->hDone);
                if(i<m_nExecute){
                        m_ppExecute[i]->hThread = CreateThread(NULL,
                            ↪ m_dwStackSize,Parallell_ThExecute,m_ppExecute[i
                            ↪ ],0,&dw);
                        while(m_ppExecute[i]->hThread==INVALID_HANDLE_VALUE ||
                            ↪ m_ppExecute[i]->hThread==NULL){
                                m_ppExecute[i]->hThread = CreateThread(NULL,
                                    ↪ m_dwStackSize,Parallell_ThExecute,
                                    ↪ m_ppExecute[i],0,&dw);
                                Sleep(20);
                        }
                        nIndex[nWRet] = i;
                        hDone[nWRet] = m_ppExecute[i]->hDone;
                        i++;
                }
                else{
                        for(;nWRet<m_nParallell-1;nWRet++){
                                hDone[nWRet]=hDone[nWRet+1];
                                nIndex[nWRet]=nIndex[nWRet+1];
                        }
                        nEnd++;
                }

                j++;
        }
        if(m_lpfnTotalDone)
                m_lpfnTotalDone(m_pDone);

        delete[] hDone;
        delete[] nIndex;
        return 1;
}
```

32 common/parallell.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
```

```
You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

typedef int (*LPEXECUTE)(LPVOID pParameter);

ULONG WINAPI Parallell_ThExecute(LPVOID pParameter);

class CParallell
{
        struct EXECUTE{
                HANDLE hThread;
                HANDLE hDone;
                LPEXECUTE lpfnExecute;
                LPEXECUTE lpfnDone;
                LPVOID pParameter;
        }**m_ppExecute;
        int m_nExecute;
        LPEXECUTE m_lpfnTotalDone;
        LPVOID m_pDone;

        int m_nParallell;
        int m_nTimeout;
        int m_nInitWait;
        DWORD m_dwStackSize;

public:
        CParallell(int nParallell, int nTimeout=INFINITE,int nInitWait=0);
        ~CParallell();

        void SetTotalDone(LPEXECUTE lpfnTotalDone,LPVOID pDone);
        int AddFunction(LPEXECUTE lpfnExecute, LPVOID pParameter, LPEXECUTE
            ↳ lpfnDone=NULL);
        int Execute();
        int ExecuteNoOrder();
        void Reset();
        void SetStackSize(DWORD dwStackSize);

        friend ULONG WINAPI Parallell_ThExecute(LPVOID pParameter);
};
```

33 common/StringVector.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
```

```
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  // Kalles Fraktaler 2
    //
    // © 2014 Karl Rummo ,runmo@hotmail.com
    //
    // COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT ↙
        ↘ WARRANTY
25  // OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ↙
        ↘ WARRANTIES
    // THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR ↙
        ↘ PURPOSE
    // OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE ↙
        ↘ COVERED
    // CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU↙
        ↘  (NOT
    // THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY ↙
        ↘ NECESSARY
30  // SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ↙
        ↘ ESSENTIAL
    // PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER ↙
        ↘ EXCEPT UNDER
    // THIS DISCLAIMER.
    //#include "stdafx.h"
    #include <windows.h>
35  #include "StringVector.h"

    int g_Allocated_StringTable=0;
    int g_nAllocatedNum=0;
    int g_nTestCount=0;
40  IMalloc *pAlloc=NULL;
    #define HASHOFFS2 1

    union Header{
            struct{
45                  Header *ptr;
                    unsigned int size;
            }s;
            double x[1];
    };
50  static Header base;
    static Header *freep=NULL;
    static Header trailer;
    #define NALLOC 1048576

55  HANDLE g_hSTMutex=NULL;

    BOOL g_StringTableMemCorrupted=FALSE;
    class CSTMutex
    {
60  public:
            CSTMutex()
            {
                    g_hSTMutex = CreateMutex(NULL,0,NULL);
```

```
        }
        ~CSTMutex()
        {
                CloseHandle(g_hSTMutex);
        }
}g_STMutex;
void kr_free(void *ap,BOOL bNoCheck=0)
{
        Header *bp, *p;
        BOOL bCheckFree=0;

        if(!ap)
                return;
        bp = (Header*)ap-1;
        if(!bNoCheck && memcmp(&trailer,bp+bp->s.size-1,sizeof(Header))){
                char szDmp[256];
                GetModuleFileName(GetModuleHandle(NULL),szDmp,sizeof(szDmp));
                strcat(szDmp,".dmp");
                DWORD dw;
                HANDLE hFile = CreateFile(szDmp,GENERIC_WRITE,0,NULL,↙
                    ↳ CREATE_ALWAYS,0,NULL);
                if(hFile!=INVALID_HANDLE_VALUE){
                        wsprintf(szDmp,"Trailer overwritten. Size: %d, Data:\r\n↙
                            ↳ \r\n",bp->s.size*sizeof(Header));
                        WriteFile(hFile,szDmp,strlen(szDmp),&dw,NULL);
                        WriteFile(hFile,bp,bp->s.size*sizeof(Header),&dw,NULL);
                        CloseHandle(hFile);
                }
        }

        WaitForSingleObject(g_hSTMutex,INFINITE);
        for(p=freep;!(bp>p && bp<p->s.ptr);p=p->s.ptr){
                if(p>=p->s.ptr && (bp>p || bp<p->s.ptr))
                        break;
        }
        memset(ap,0xcc,(bp->s.size-1)*sizeof(Header));
        if(bp+bp->s.size==p->s.ptr){
                bp->s.size+=p->s.ptr->s.size;
                bp->s.ptr=p->s.ptr->s.ptr;
                if(bp->s.size==NALLOC)
                        bCheckFree=1;
        }
        else
                bp->s.ptr=p->s.ptr;
        if(p+p->s.size==bp){
                p->s.size+=bp->s.size;
                p->s.ptr=bp->s.ptr;
                if(p->s.size==NALLOC)
                        bCheckFree=1;
        }
        else
                p->s.ptr=bp;
        freep=p;
        if(bNoCheck || !bCheckFree){
                ReleaseMutex(g_hSTMutex);
                return;
        }
```

```
        p = &base;
120     Header *prevp=NULL;
        do{
                if(p->s.size==NALLOC){
                        if(prevp){
                                prevp->s.ptr = p->s.ptr;
125                             if(freep==p)
                                        freep=prevp;
                        }
                        else
                                freep=NULL;
130                     free(p);
                        g_Allocated_StringTable--;
                        p=prevp;
                        break;
                }
135             prevp = p;
                p=p->s.ptr;
        }while(p!=&base);
        ReleaseMutex(g_hSTMutex);
}
140 static Header *morecore(unsigned int nu)
{
        char *cp;
        Header *up;

145     if(nu<NALLOC)
                nu = NALLOC;
        cp = (char*)malloc(nu*sizeof(Header));
        g_Allocated_StringTable++;
        if(cp==NULL)
150             return NULL;
        up = (Header*)cp;
        up->s.size=nu;
        kr_free((void*)(up+1),1);
        return freep;
155 }
int g_nMaxTries=0;
void *kr_malloc(unsigned int nbytes)
{
        g_nTestCount++;
160     Header *p, *prevp;
        unsigned int nunits;
        nunits = (nbytes+sizeof(Header)-1)/sizeof(Header) + 2;
        if(nunits>=NALLOC)
                Beep(1000,10);
165     WaitForSingleObject(g_hSTMutex,INFINITE);
        if((prevp=freep)==NULL){
                memset(&trailer,1,sizeof(Header));
                base.s.ptr = freep = prevp = &base;
                base.s.size = 0;
170     }
        int nTries=0;
        for(p=prevp->s.ptr;;prevp=p,p=p->s.ptr){
                if(p->s.size>=nunits){
                        if(p->s.size-nunits<3)
175                             nunits = p->s.size;
```

```
                        if(p->s.size==nunits)
                                prevp->s.ptr = p->s.ptr;
                        else{
                                p->s.size-=nunits;
                                p+=p->s.size;
                                p->s.size = nunits;
                        }
                        freep = prevp;
                        ReleaseMutex(g_hSTMutex);
                        memset(p+p->s.size-1,1,sizeof(Header));
                        if(nTries>g_nMaxTries)
                                g_nMaxTries=nTries;
                        return (void*)(p+1);
                }
                nTries++;
                if(p==freep){
                        if((p=morecore(nunits))==NULL){
                                ReleaseMutex(g_hSTMutex);
                                return NULL;
                        }
                }
        }
        ReleaseMutex(g_hSTMutex);
}

void *realloc_stringtable(void *p,size_t n)
{
#ifdef _KR_MALLOC_
        if(p){
                Header *ph = (Header *)p;
                ph--;
                int nPrevSize = (ph->s.size-2)*sizeof(Header);
                if(nPrevSize>=n)
                        return p;
        }
        void *pr = kr_malloc(n+(p && n+n<NALLOC*sizeof(Header)?n:0));
        if(p){
                Header *ph = (Header *)p;
                ph--;
                int nPrevSize = ph->s.size*sizeof(Header);
                memcpy(pr,p,nPrevSize<n?nPrevSize:n);
                kr_free(p);
        }
        return pr;
#endif

        if(!pAlloc)
                CoGetMalloc(1,&pAlloc);
        void *pp=NULL;
        while(!pp){
                if(p){
                        size_t a = *((int*)((char*)p-sizeof(int)));
                        if(a>100000000)
                                Beep(1000,10);

                        char *TEST = (char*)p;
                        if(TEST[a]!='A' || TEST[a+1]!='A'){
```

```
#ifdef _DEBUG
                                        MessageBox(NULL,"Memory corrupted","Error",MB_OK↙
                                            ↳ );
#endif
                                        g_StringTableMemCorrupted=TRUE;
                                }

                                if(a>=n)
                                        return p;
                                if(n>1024)
                                        n<<=1;
                                else
                                        n<<=2;
                                pp = pAlloc->Alloc(n+sizeof(int)+8);
                                if(pp){
                                        memcpy((char*)pp+sizeof(int),p,a);
                                        pAlloc->Free((char*)p-sizeof(int));
                                        g_Allocated_StringTable-=a;
                                }
                        }
                        else{
                                g_nAllocatedNum++;
                /*              if(n>1024)
                                        n<<=1;
                                else
                                        n<<=2;
                */              pp = pAlloc->Alloc(n+sizeof(int)+8);
                        }
                        if(!pp){
                                pAlloc->HeapMinimize();
                        }
                }
                char *TEST = (char*)pp;
                TEST+=sizeof(int);
                TEST[n]='A';
                TEST[n+1]='A';

                *((int*)((char*)pp)) = n;
                g_Allocated_StringTable+=n;

                if(g_nAllocatedNum%100000==0)
                        pAlloc->HeapMinimize();

                return (char*)pp+sizeof(int);
}
void free_stringtable(void *p)
{
#ifdef _KR_MALLOC_
                kr_free(p);
                return;
#endif

                if(p){
                        size_t a = *((int*)((char*)p-sizeof(int)));
//                      if(a>1000000)
//                              Beep(1000,10);
```

```
                char *TEST = (char*)p;
290             if(TEST[a]!='A' || TEST[a+1]!='A'){
#ifdef _DEBUG
                                MessageBox(NULL,"Memory corrupted","Error",MB_OK↙
                                    ↳ );
#endif
                        g_StringTableMemCorrupted=TRUE;
295             }

                g_Allocated_StringTable -= *((int*)((char*)p-sizeof(int)));
                pAlloc->Free((char*)p-sizeof(int));
        }
300 }
    CStringVektor::CStringVektor()
        : m_pszStrings(NULL)
        , m_pnStrings(NULL)
        , m_pnIndexValues(NULL)
305     , m_nStrings(0)
    {
        nullstr[0] = 0;
    }
    CStringVektor::~CStringVektor()
310 {
        Clean();
    }
    void CStringVektor::Clean()
    {
315     int i;
        for(i=0;i<m_nStrings;i++)
                delete[] m_pszStrings[i];
        free_stringtable(m_pszStrings);
        m_nStrings=0;
320     m_pszStrings=NULL;
        free_stringtable(m_pnStrings);
        m_pnStrings=NULL;
        free_stringtable(m_pnIndexValues);
        m_pnIndexValues=NULL;
325 }
    int CStringVektor::MakeIndex(const char *szString, int nLenght)
    {
        __int64 nIndex=0, i;
        for(i=0;i<nLenght;i++)
330             nIndex = (217*nIndex+szString[i]);
        return (int)nIndex;
    }

    int CStringVektor::AddInt(intptr_t nVal)
335 {
        char szTmp[80];
        snprintf(szTmp, 80, "%" PRIdPTR, nVal);
        return AddString(szTmp);
    }
340 int CStringVektor::AddString(const char *szString,int nSize)
    {
        if(!szString)
                szString = "";
        int i=m_nStrings++;
```

```
345             m_pnStrings = (int*)realloc_stringtable(m_pnStrings,sizeof(int)*↙
                    ↳ m_nStrings);
                m_pnIndexValues = (int*)realloc_stringtable(m_pnIndexValues,sizeof(int)*↙
                    ↳ m_nStrings);
                if(nSize==-1)
                        m_pnStrings[i] = strlen(szString);
                else
350                     m_pnStrings[i] = nSize;
                m_pszStrings = (char**)realloc_stringtable(m_pszStrings,sizeof(char*)*↙
                    ↳ m_nStrings);
                m_pszStrings[i] = new char[m_pnStrings[i]+1];
                if(m_pnStrings[i])
                        memcpy(m_pszStrings[i],szString,m_pnStrings[i]);
355             m_pszStrings[i][m_pnStrings[i]]=0;
                if(m_pnStrings[i])
                        m_pnIndexValues[i] = MakeIndex(szString, m_pnStrings[i]);
                else
                        m_pnIndexValues[i]=0;
360             return 1;
        }
        int CStringVektor::InsertString(int nIndex, const char *szString,int nSize)
        {
                if(nIndex>=m_nStrings)
365                     return AddString(szString,nSize);
                if(!szString || nIndex<0 || nIndex>=m_nStrings)
                        return 0;
                int i;
                m_nStrings++;
370             if(nSize==-1)
                        nSize = strlen(szString);
                m_pnStrings = (int*)realloc_stringtable(m_pnStrings,sizeof(int)*↙
                    ↳ m_nStrings);
                m_pnIndexValues = (int*)realloc_stringtable(m_pnIndexValues,sizeof(int)*↙
                    ↳ m_nStrings);
                m_pszStrings = (char**)realloc_stringtable(m_pszStrings,sizeof(char*)*↙
                    ↳ m_nStrings);
375             for(i=m_nStrings-1;i>nIndex;i--){
                        m_pnStrings[i] = m_pnStrings[i-1];
                        m_pnIndexValues[i] = m_pnIndexValues[i-1];
                        m_pszStrings[i] = m_pszStrings[i-1];
                }
380             m_pnStrings[i] = nSize;
                m_pszStrings[i] = new char[m_pnStrings[i]+1];
                if(m_pnStrings[i])
                        memcpy(m_pszStrings[i],szString,m_pnStrings[i]);
                m_pszStrings[i][m_pnStrings[i]]=0;
385             if(m_pnStrings[i])
                        m_pnIndexValues[i] = MakeIndex(szString, m_pnStrings[i]);
                else
                        m_pnIndexValues[i]=0;
                return 1;
390     }

        int CStringVektor::DeleteString(int nIndex)
        {
                if(nIndex<0 || nIndex>=m_nStrings)
395                     return 0;
```

```
        delete[] m_pszStrings[nIndex];
        m_nStrings--;
        for(int i=nIndex;i<m_nStrings;i++){
                m_pnStrings[i] = m_pnStrings[i+1];
                m_pnIndexValues[i] = m_pnIndexValues[i+1];
                m_pszStrings[i] = m_pszStrings[i+1];
        }
        return 1;
}
int CStringVektor::SetString(int nIndex, const char *szString, int nSize)
{
        if(nIndex<0 || nIndex>=m_nStrings || !szString)
                return 0;
        char *szTmp = m_pszStrings[nIndex];
        if(nSize==-1)
                m_pnStrings[nIndex] = strlen(szString);
        else
                m_pnStrings[nIndex] = nSize;
        m_pszStrings[nIndex] = new char[m_pnStrings[nIndex]+1];
        memcpy(m_pszStrings[nIndex],szString,m_pnStrings[nIndex]+1);
        m_pszStrings[nIndex][m_pnStrings[nIndex]]=0;
        m_pnIndexValues[nIndex] = MakeIndex(szString, m_pnStrings[nIndex]);
        delete[] szTmp;

        return 1;
}
int CStringVektor::AppendString(int nIndex, const char *szString, int nSize)
{
        if(nIndex<0 || nIndex>=m_nStrings || !szString)
                return 0;
        char *szTmp = m_pszStrings[nIndex];
        int nPrevSize = m_pnStrings[nIndex];
        if(nSize==-1)
                nSize = strlen(szString);
        m_pnStrings[nIndex] += nSize;

        m_pszStrings[nIndex] = new char[m_pnStrings[nIndex]+1];
        memcpy(m_pszStrings[nIndex],szTmp,nPrevSize+1);
        m_pszStrings[nIndex][nPrevSize]=0;
        memcpy(m_pszStrings[nIndex]+nPrevSize,szString,nSize+1);
        m_pszStrings[nIndex][m_pnStrings[nIndex]]=0;
        m_pnIndexValues[nIndex] = MakeIndex(m_pszStrings[nIndex], m_pnStrings[↙
            ↳ nIndex]);
        delete[] szTmp;

        return 1;
}
char *CStringVektor::GetString(int nIndex,int *pnSize)
{
        if(nIndex<0 || nIndex>=m_nStrings){
                if(pnSize)
                        *pnSize=0;
                return nullstr;
        }
        if(pnSize)
                *pnSize = m_pnStrings[nIndex];
        return m_pszStrings[nIndex]?m_pszStrings[nIndex]:nullstr;
```

```
}
int CStringVektor::GetIndex(int nIndex)
{
        if(nIndex<0 || nIndex>=m_nStrings)
                return -1;
        return m_pnIndexValues[nIndex];
}
int CStringVektor::GetCount()
{
        return m_nStrings;
}
int CStringVektor::GetLength(int nIndex)
{
        if(nIndex<0 || nIndex>=m_nStrings)
                return -1;
        return m_pnStrings[nIndex];
}
int CStringVektor::SetCount(int nCount)
{
        int i;
        if(nCount<m_nStrings){
                for(i=nCount;i<m_nStrings;i++)
                        delete[] m_pszStrings[i];
        }
        else{
                m_pszStrings = (char**)realloc_stringtable(m_pszStrings,sizeof(↙
                    ↳ char*)*nCount);
                m_pnStrings = (int*)realloc_stringtable(m_pnStrings,sizeof(int)*↙
                    ↳ nCount);
                m_pnIndexValues = (int*)realloc_stringtable(m_pnIndexValues,↙
                    ↳ sizeof(int)*nCount);
                for(i=m_nStrings;i<nCount;i++){
                        m_pszStrings[i] = new char[1];
                        *m_pszStrings[i]=0;
                        m_pnStrings[i]=0;
                        m_pnIndexValues[i]=-1;
                }
        }
        return (m_nStrings = nCount);
}
char *CStringVektor::operator [] (int nIndex)
{
        return GetString(nIndex);
}
CStringVektor &CStringVektor::operator =(CStringVektor &s)
{
        Clean();
        int i, n;
        for(i=0;i<s.GetCount();i++){
                const char *sz = s.GetString(i,&n);
                AddString(sz,n);
        }
        return *this;
}

int CStringVektor::MoveString(int nFrom, int nTo)
{
```

```
        if(nFrom<0 || nFrom>=m_nStrings)
                return 0;
        if(nTo<0 || nTo>=m_nStrings)
                return 0;
        char *szTmpStr = m_pszStrings[nFrom];
        m_pszStrings[nFrom] = m_pszStrings[nTo];
        m_pszStrings[nTo] = szTmpStr;
        int nTmp = m_pnStrings[nFrom];
        m_pnStrings[nFrom] = m_pnStrings[nTo];
        m_pnStrings[nTo] = nTmp;
        nTmp = m_pnIndexValues[nFrom];
        m_pnIndexValues[nFrom] = m_pnIndexValues[nTo];
        m_pnIndexValues[nTo] = nTmp;
        return 1;
}
char *CStringVektor::ToText(const char *szFieldSep)
{
        int nLen=0;
        char *szRet=NULL;
        int j, nSize;
        int nFieldSep=strlen(szFieldSep);

        for(j=0;j<m_nStrings;j++){
                GetString(j,&nSize);
                nLen+=nSize;
                if(j<m_nStrings-1)
                        nLen+=nFieldSep;
        }
        nLen++;
        szRet = new char[nLen];
        nLen=0;
        for(j=0;j<m_nStrings;j++){
                strcpy(szRet+nLen,GetString(j,&nSize));
                nLen+=nSize;
                if(j<m_nStrings-1){
                        strcpy(szRet+nLen,szFieldSep);
                        nLen+=nFieldSep;
                }
        }
        szRet[nLen]=0;
        return szRet;
}
void CStringVektor::DeleteToText(char *szToText)
{
        delete[] szToText;
}
int CStringVektor::FindString(const char *szString,int nLen)
{
        if(nLen==-1)
                nLen = strlen(szString);
        int index = MakeIndex(szString,nLen);
        int i;
        for(i=0;i<m_nStrings;i++)
                if(m_pnIndexValues[i]==index && !strncmp(m_pszStrings[i],↙
                    ↳ szString,nLen) && m_pnStrings[i]==nLen)
                        return i;
        return -1;
```

```
}

CStringTable::CStringTable()
	: m_pVektors(NULL)
	, m_nVektors(0)
	, m_nRowSize(0)
	, m_pnOrders(NULL)
	, m_pnHash(NULL)
	, m_nHash(0)
	, m_nHashColumn(-1)
	, m_nLastHash(-1)
	, m_bHashDirty(FALSE)
{
}
CStringTable::CStringTable(const char *szData,const char *szFieldSep, const char *szRowSep,BOOL bApo)
	: m_pVektors(NULL)
	, m_nVektors(0)
	, m_nRowSize(0)
	, m_pnOrders(NULL)
	, m_pnHash(NULL)
	, m_nHash(0)
	, m_nHashColumn(-1)
	, m_nLastHash(-1)
	, m_bHashDirty(FALSE)
{
	nullstr[0] = 0;
	SplitString(szData,szFieldSep,szRowSep,bApo);
}

CStringTable::~CStringTable()
{
	int i;
	for(i=0;i<m_nVektors;i++)
		delete m_pVektors[i];
	free_stringtable(m_pVektors);
	m_nVektors=0;
	m_pVektors=NULL;
	delete[] m_pnHash;
	m_pnHash = NULL;
}
int CStringTable::AddRow(char **pszRow, int nRow)
{
	if(nRow>m_nRowSize)
		m_nRowSize = nRow;
	int j, i = AddRow()-1;
	for(j=0;j<nRow;j++)
		m_pVektors[i]->AddString(pszRow[j]);
	return m_nVektors;
}
int CStringTable::AddRow()
{
	int i = m_nVektors++;
	m_pVektors = (CStringVektor**)realloc_stringtable(m_pVektors,sizeof(CStringVektor*)*m_nVektors);
	m_pVektors[i] = new CStringVektor();
	return m_nVektors;
```

```
}
int CStringTable::InsertRow(int nRow)
{
        if(nRow<0 || nRow>m_nVektors)
                return 0;

        m_bHashDirty=1;
/*      if(m_pnHash)
                delete m_pnHash;
        m_pnHash=NULL;
        m_nHash = -1;
        m_nHashColumn = -1;
*/
        if(nRow==m_nVektors){
                AddRow();
                return 1;
        }
        int i;
        m_nVektors++;
        m_pVektors = (CStringVektor**)realloc_stringtable(m_pVektors,sizeof(
            ↪ CStringVektor*)*m_nVektors);
        for(i=m_nVektors-1;i>nRow;i--)
                m_pVektors[i] = m_pVektors[i-1];
        m_pVektors[nRow] = new CStringVektor();
        return 1;
}
int CStringTable::AddString(int nRow, const char *szString, int nString)
{
        if(nRow<0 || nRow>=m_nVektors)
                return 0;
        m_pVektors[nRow]->AddString(szString,nString);
        if(m_nRowSize<m_pVektors[nRow]->GetCount())
                m_nRowSize = m_pVektors[nRow]->GetCount();

        if(m_pVektors[nRow]->GetCount()==m_nHashColumn+1){
                if(GetCount()<m_nHash-10000){
                        int j;
                        j = (unsigned int)m_pVektors[nRow]->GetIndex(
                            ↪ m_nHashColumn)%m_nHash;
                        while(m_pnHash[j]!=-1){
                                j = (j+HASHOFFS2)%m_nHash;
                        }
                        m_pnHash[j] = nRow;
                }
                else
                        BuildHash(m_nHashColumn);
        }
        return 1;
}
int CStringTable::AddString(int nRow, const std::string &szString, int nString)
{
        char *s = strdup(szString.c_str());
        int ret = AddString(nRow, s, nString);
        free(s);
        return ret;
}
```

```
int CStringTable::AddInt(int nRow, intptr_t nVal)
{
        char szTmp[80];
675     snprintf(szTmp, 80, "%" PRIdPTR, nVal);
        return AddString(nRow,szTmp);
}
int CStringTable::DeleteRow(int nRow)
{
680     if(nRow<0 || nRow>=m_nVektors)
                return 0;
        m_bHashDirty=1;
        CStringVektor *p = m_pVektors[nRow];
        m_nVektors--;
685     for(;nRow<m_nVektors;nRow++)
                m_pVektors[nRow] = m_pVektors[nRow+1];
        delete p;
        return 1;
}
690 int CStringTable::DeleteColumn(int nColumn)
{
        if(nColumn<=m_nHashColumn){
                m_bHashDirty=1;
/*              if(m_pnHash)
695                     delete m_pnHash;
                m_pnHash=NULL;
                m_nHash = -1;
                m_nHashColumn = -1;
*/      }
700     int i;
        for(i=0;i<m_nVektors;i++)
                m_pVektors[i]->DeleteString(nColumn);
        return 1;
}
705 void CStringTable::Reset()
{
        while(m_nVektors)
                DeleteRow(m_nVektors-1);
        if(m_nHashColumn!=-1)
710             BuildHash(m_nHashColumn);
}
int CStringTable::SetString(int nRow, int nIndex, char *szString, int nString)
{
        if(nRow<0 || nRow>=m_nVektors)
715             return 0;

        if(m_nHashColumn==nIndex){
                m_bHashDirty=1;
/*              if(m_pnHash)
720                     delete m_pnHash;
                m_pnHash=NULL;
                m_nHash = -1;
                m_nHashColumn = -1;
*/      }
725     return m_pVektors[nRow]->SetString(nIndex,szString,nString);
}
int CStringTable::AppendString(int nRow, int nIndex, char *szString, int nString↙
    ↳ )
```

```
{
        if(nRow<0 || nRow>=m_nVektors)
                return 0;

        if(m_nHashColumn==nIndex){
                m_bHashDirty=1;
/*              if(m_pnHash)
                        delete m_pnHash;
                m_pnHash=NULL;
                m_nHash = -1;
                m_nHashColumn = -1;
*/      }
        return m_pVektors[nRow]->AppendString(nIndex,szString,nString);
}
int CStringTable::SetInt(int nRow, int nIndex, intptr_t nVal)
{
        char szTmp[80];
        snprintf(szTmp, 80, "%" PRIdPTR, nVal);
        return SetString(nRow,nIndex,szTmp);
}
char *CStringTable::GetString(int nRow, int nIndex,int *pnSize)
{
        if(nRow<0 || nRow>=m_nVektors)
                return nullstr;
        return m_pVektors[nRow]->GetString(nIndex,pnSize);
}
int CStringTable::GetCount()
{
        return m_nVektors;
}
int CStringTable::GetRowSize(int nRow)
{
        if(nRow==-1)
                return m_nRowSize;
        if(nRow<0 || nRow>=m_nVektors)
                return 0;
        return m_pVektors[nRow]->GetCount();
}

CStringVektor &CStringTable::operator [] (int nRow)
{
        if(nRow<0 || nRow>=m_nVektors)
                return m_err;
        return *m_pVektors[nRow];
}
CStringTable &CStringTable::operator = (CStringTable &st)
{
        Reset();
        int i, j, n = 0;
        for(i=0;i<st.GetCount();i++){
                AddRow();
                for(j=0;j<st.GetRowSize(i);j++){
                        char *sz = st.GetString(i,j,&n);
                        AddString(i,sz,n);
                }
        }
        return *this;
```

```
785 }
    CStringTable::CStringTable(CStringTable &st)
            : m_pVektors(NULL)
            , m_nVektors(0)
            , m_nRowSize(0)
790         , m_pnOrders(NULL)
            , m_pnHash(NULL)
            , m_nHash(0)
            , m_nHashColumn(-1)
            , m_nLastHash(-1)
795         , m_bHashDirty(FALSE)
    {
            *this = st;
    }

800 int CStringTable::ReadCSV(char *szFileName)
    {
            DWORD dw;
            HANDLE hFile = CreateFileA(szFileName,GENERIC_READ,FILE_SHARE_READ|↙
                ↳ FILE_SHARE_WRITE,NULL,OPEN_EXISTING,0,NULL);
            if(hFile==INVALID_HANDLE_VALUE)
805                 return 0;
            int nLen = GetFileSize(hFile,NULL);
            char *szData = new char[nLen+1];
            ReadFile(hFile,szData,nLen,&dw,NULL);
            CloseHandle(hFile);
810         szData[nLen]=0;

            AddRow();
            SplitString(szData,";","\r\n");
            delete[] szData;
815         return 1;
    }

    char *stristr(char *src, char *find)
    {
820         int i;

            while(*src){
                    if(toupper(*src)==toupper(*find)){
                            for(i=1;find[i] && toupper(src[i])==toupper(find[i]);i↙
                                ↳ ++);
825                         if(find[i]==0)
                                    return src;
                    }
                    src++;
            }
830         return NULL;
    }

    int CStringTable::Compare(int a, int b, int nColumn, int nType, int nOrder)
    {
835         int ret;

            int nCmp=0;
            if(nType==0)
                    nCmp = stricmp(m_pVektors[a]->GetString(nColumn),m_pVektors[b]->↙
```

```
                        ↪ GetString(nColumn));
840             else{
                        if(atof(m_pVektors[a]->GetString(nColumn))<atof(m_pVektors[b]->↙
                            ↪ GetString(nColumn)))
                                nCmp = -1;
                        else if(atof(m_pVektors[a]->GetString(nColumn))>atof(m_pVektors[↙
                            ↪ b]->GetString(nColumn)))
                                nCmp = 1;
845             }
                if(nOrder){
                        if(nCmp==0 && m_pnOrders[a]>=m_pnOrders[b])
                                nCmp=-1;
                        if(nCmp<0)
850                             ret=1;
                        else
                                ret=0;
                }
                else{
855                     if(nCmp==0 && m_pnOrders[a]>=m_pnOrders[b])
                                nCmp=1;
                        if(nCmp>0)
                                ret=1;
                        else
860                             ret=0;
                }

                return ret;
        }
865
        void CStringTable::Swap(int a,  int b)
        {
                CStringVektor *tmp = m_pVektors[a];
                m_pVektors[a] = m_pVektors[b];
870             m_pVektors[b] = tmp;

                if(m_pnOrders){
                        int nTmp = m_pnOrders[a];
                        m_pnOrders[a] = m_pnOrders[b];
875                     m_pnOrders[b] = nTmp;
                }
        }
        void CStringTable::M3QSort(int nColumn,int nOrder,int nType,int left, int right)
        {
880             if(!m_nVektors || left>=m_nVektors)
                        return;
                BOOL bInit=FALSE;
                if(right==-1)
                        right = m_nVektors-1;
885             int i, j, z, cut;
                if(left==right)
                        return;
                i=left;
                j=right;
890             z=(right+left)/2;
                cut = 0;

                if(!m_pnOrders){
```

```
                bInit=TRUE;
895             m_pnOrders = new int[m_nVektors];
                for(int i=0;i<m_nVektors;i++)
                        m_pnOrders[i] = i;
        }

900     if(Compare(left,z,nColumn,nType,nOrder))
                Swap(left, z);
        if(Compare(z,right,nColumn,nType,nOrder)){
                Swap(z, right);
                if(Compare(left,z,nColumn,nType,nOrder))
905                     Swap(left, z);
        }

        cut = z;
        do{
910             while(Compare(cut,i,nColumn,nType,nOrder)){
                        i=i+1;
                        if(i==right) break;
                }
                while(Compare(j,cut,nColumn,nType,nOrder)){
915                     j=j-1;
                        if(j==left) break;
                }
                if(i<=j){
                        if(j!=i)
920                             Swap(i,j);
                        i=i+1;
                        j=j-1;
                }
        }while(i<=j);
925     if(left < j)
                M3QSort(nColumn,nOrder,nType,left,j);
        if(i < right)
                M3QSort(nColumn,nOrder,nType,i,right);

930     if(bInit){
                delete[] m_pnOrders;
                m_pnOrders=NULL;
        }
}
935 int CStringTable::FindString(int nColumn, const char *szString, int nLength)
{
        if(!m_nVektors || !szString)
                return -1;
        if(nLength==-1)
940             nLength = strlen(szString);
        int nIndexValue = m_pVektors[0]->MakeIndex(szString,nLength);
        int i;
        for(i=0;i<m_nVektors;i++)
                if(nIndexValue==m_pVektors[i]->GetIndex(nColumn)){
945                     if(!strncmp(szString,m_pVektors[i]->GetString(nColumn),↙
                            ↳ nLength) && (int)strlen(m_pVektors[i]->GetString(↙
                            ↳ nColumn))==nLength)
                                return i;
                }
        return -1;
```

```
}
int CStringTable::FindStringN(int nColumn, const char *szString, int nLength)
{
	if(!m_nVektors)
		return -1;
	if(nLength==-1)
		nLength = strlen(szString);
	int i;
	for(i=0;i<m_nVektors;i++)
		if(!strnicmp(m_pVektors[i]->GetString(nColumn),szString,nLength)
		    )
			return i;
	return -1;
}
int CStringTable::Save(char *szFile)
{
	int i, j;
	DWORD dw;
	HANDLE hFile = CreateFileA(szFile,GENERIC_WRITE,0,NULL,CREATE_ALWAYS,0,
	    NULL);
	if(hFile==INVALID_HANDLE_VALUE)
		return 0;
	WriteFile(hFile,&m_nVektors,sizeof(m_nVektors),&dw,NULL);
	for(i=0;i<m_nVektors;i++){
		int nLen = m_pVektors[i]->GetCount();
		WriteFile(hFile,&nLen,sizeof(nLen),&dw,NULL);
		for(j=0;j<m_pVektors[i]->GetCount();j++){
			char *sz = m_pVektors[i]->GetString(j,&nLen);
			WriteFile(hFile,&nLen,sizeof(nLen),&dw,NULL);
			WriteFile(hFile,sz,nLen,&dw,NULL);
		}
	}
	CloseHandle(hFile);
	return 1;
}
int CStringTable::Load(char *szFile)
{
	int i, j, nLen;
	DWORD dw;
	HANDLE hFile = CreateFileA(szFile,GENERIC_READ,0,NULL,OPEN_EXISTING,0,
	    NULL);
	if(hFile==INVALID_HANDLE_VALUE)
		return 0;
	while(m_nVektors)
		DeleteRow(0);
	int nRows;
	ReadFile(hFile,&nRows,sizeof(nRows),&dw,NULL);
	for(i=0;i<nRows;i++){
		ReadFile(hFile,&nLen,sizeof(nLen),&dw,NULL);
		AddRow();
		for(j=0;j<nLen;j++){
			int nStringSize;
			ReadFile(hFile,&nStringSize,sizeof(nStringSize),&dw,NULL
			    );
			char *sz = new char[nStringSize+1];
			ReadFile(hFile,sz,nStringSize,&dw,NULL);
			sz[nStringSize]=0;
```

```
                        AddString(i,sz,nStringSize);
                        delete[] sz;
                }
        }
        CloseHandle(hFile);
        return 1;
}

int CStringTable::SplitString(const char *szData,const char *szFieldSep, const char *szRowSep,BOOL bApo)
{
        if(!szData)
                return 0;
        AddRow();
        int q, p;
        int nFieldSep=0;
        if(szFieldSep)
                nFieldSep = strlen(szFieldSep);
        int nRowSep=0;
        if(szRowSep)
                nRowSep = strlen(szRowSep);
        BOOL bInsideApo=0;
        BOOL bRecordStart=TRUE;
        for(p=q=0;szData[q];q++){
                if(bApo && bRecordStart && szData[q]=='"'){
                        bInsideApo=TRUE;
                        bRecordStart=FALSE;
                }
                else if(bInsideApo && szData[q]=='"' && szData[q+1]!='"')
                        bInsideApo=FALSE;
                if(bInsideApo)
                        continue;

                if(!szData[q] || (nFieldSep && !strncmp(&szData[q],szFieldSep,nFieldSep)) ||
                                (nRowSep && !strncmp(&szData[q],szRowSep,nRowSep))){
                        if(bApo && szData[p]=='"')
                                AddString(GetCount()-1,&szData[p+1],q-p-2);
                        else
                                AddString(GetCount()-1,&szData[p],q-p);
                        if(nRowSep && !strncmp(&szData[q],szRowSep,nRowSep)){
                                q+=nRowSep-1;
                                AddRow();
                        }
                        else
                                q+=nFieldSep-1;
                        p=q+1;
                        bRecordStart=TRUE;
                }
        }
        if(szData[p])
                AddString(GetCount()-1,&szData[p]);
        else if(!GetRowSize(GetCount()-1))
                DeleteRow(GetCount()-1);
        return 1;
}
```

```
char *CStringTable::ToText(const char *szFieldSep, const char *szRowSep)
{
        int nLen=0;
1060    char *szRet=NULL;
        int i, j, k, nSize;
        int nFieldSep=strlen(szFieldSep);
        int nRowSep=strlen(szRowSep);

1065    for(i=0;i<GetCount();i++){
                for(j=0;j<GetRowSize(i);j++){
                        m_pVektors[i]->GetString(j,&nSize);
                        nLen+=nSize;
                        if(j<GetRowSize(i)-1)
1070                            nLen+=nFieldSep;
                        else
                                nLen+=nRowSep;
                }
        }
1075    nLen++;
        szRet = new char[nLen];
        *szRet=0;
        nLen=0;
        for(i=0;i<GetCount();i++){
1080            for(j=0;j<GetRowSize(i);j++){
                        strcpy(&szRet[nLen],m_pVektors[i]->GetString(j,&nSize));
                        for(k=0;k<nSize;k++)
                                if(!szRet[nLen+k])
                                        szRet[nLen+k]=' ';
1085                    nLen+=nSize;
                        if(j<GetRowSize(i)-1){
                                strcpy(&szRet[nLen],szFieldSep);
                                nLen+=nFieldSep;
                        }
1090                    else{
                                strcpy(&szRet[nLen],szRowSep);
                                nLen+=nRowSep;
                        }
                }
1095    }
        return szRet;
}
void CStringTable::DeleteToText(char *szToText)
{
1100    delete[] szToText;
}

int CStringTable::MoveCol(int nFrom, int nTo)
{
1105    int i;
        for(i=0;i<GetCount();i++)
                m_pVektors[i]->MoveString(nFrom,nTo);
        return 1;
}
1110 int CStringTable::MoveRow(int nFrom, int nTo)
{
        if(nFrom<0 || nFrom>=m_nVektors || nTo<0 || nTo>=m_nVektors)
```

```
                return 0;
        CStringVektor *pTmp = m_pVektors[nFrom];
        m_pVektors[nFrom] = m_pVektors[nTo];
        m_pVektors[nTo] = pTmp;
        return 1;
}

int CStringTable::BuildHash(int nColumn,int nItems)
{
        if(m_pnHash)
                delete[] m_pnHash;
        m_nHash = nItems;
        m_nHashColumn = nColumn;
        m_nLastHash = -1;
        if(m_nHash<GetCount()+10)
                m_nHash = GetCount()*2;
        if(m_nHash<64919)
                m_nHash=64919;
        else{
                m_nHash+=64919;
//#ifdef REAL_PRIME_VALUE
                int q;
                while(1){
                        int nHalf = m_nHash/2;
                        for(q=2;q<=nHalf;q++)
                                if(m_nHash%q==0)
                                        break;
                        if(q>nHalf)
                                break;
                        m_nHash++;
                }
//#endif
        }
        m_pnHash = new int[m_nHash];
        memset(m_pnHash,-1,sizeof(int)*m_nHash);
        int i, j;
#ifdef _DEBUG
        int nFirst, nKrock=0, nKrock2=0;
#endif
        for(i=0;i<GetCount();i++){
#ifdef _DEBUG
                nFirst=1;
#endif
                j = (unsigned int)m_pVektors[i]->GetIndex(nColumn)%m_nHash;
                while(m_pnHash[j]!=-1){
#ifdef _DEBUG
                        if(nFirst){
                                nFirst=0;
                                nKrock++;
                        }
                        nKrock2++;
#endif
                        j = (j+HASHOFFS2)%m_nHash;
                }
                m_pnHash[j] = i;
        }
```

```
1170            m_bHashDirty=FALSE;
                return 0;
        }
        int CStringTable::FindStringHash(int nColumn, char *szString,int nString,int ↙
            ↪ nPrevHit)
        {
1175            if(!m_pnHash)
                        return -1;
                if(!m_pVektors)
                        return -1;
                if(nColumn!=m_nHashColumn)
1180                    return -1;
                if(m_bHashDirty)
                        BuildHash(m_nHashColumn);
                if(nString==-1)
                        nString = strlen(szString);
1185            int nTest;
                if(nPrevHit!=-1){
                        if(m_nLastHash==-1)
                                return -1;
                        int j = m_nLastHash;
                        if(j<0 || j>=m_nHash || m_pnHash[j]!=nPrevHit)
1190                            return -1;
                        while(m_pnHash[j]!=-1){
                                do{
                                        j = (j+HASHOFFS2)%m_nHash;
                                }while(m_pnHash[j]!=-1 && (strncmp(szString,m_pVektors[↙
                                    ↪ m_pnHash[j]]->GetString(nColumn,&nTest),nString) ↙
                                    ↪ || nString!=nTest));
                                m_nLastHash = j;
                                return m_pnHash[j];
                        }
                        m_nLastHash = j;
1200                    return m_pnHash[j];
                }
                int nIndexValue = m_pVektors[0]->MakeIndex(szString,nString);
                int j = (unsigned int)nIndexValue%m_nHash;
                while(m_pnHash[j]!=-1 && nIndexValue!=m_pVektors[m_pnHash[j]]->GetIndex(↙
                    ↪ nColumn)){
1205                j = (j+HASHOFFS2)%m_nHash;
                }
                while(m_pnHash[j]!=-1 && (strncmp(szString,m_pVektors[m_pnHash[j]]->↙
                    ↪ GetString(nColumn,&nTest),nString) || nString!=nTest)){
                        j = (j+HASHOFFS2)%m_nHash;
                }
1210            m_nLastHash = j;
                return m_pnHash[j];
        }
        int CStringTable::GetHashColumn()
        {
1215            return m_nHashColumn;
        }

        int CStringTable::FindStringBinary(int nColumn, char *szString, int nLength)
        {
1220            int m = 0;
                int n = GetCount();
```

```
        int k = n/2;
        int t;
        if(nLength==-1)
1225            nLength = strlen(szString);
        while((t = strncmp(GetString(k,nColumn),szString,nLength))){
            if(t<0)
                m = k;
            else
1230                n = k;
            if(m==n)
                return -1;
            k = (n-m)/2 + m;
        }
1235    return k;
}
```

34 common/StringVector.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15
You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20  #ifndef __STRINGVECTOR_H__
    #define __STRINGVECTOR_H__

    #include <string>
    #include <cinttypes>
25
    class CStringVektor
    {
        char **m_pszStrings;
        int *m_pnStrings;
30      int *m_pnIndexValues;
        int m_nStrings;
        char nullstr[1];
    public:
        CStringVektor();
35      ~CStringVektor();
        void Clean();
        int AddInt(intptr_t nVal);
        int AddString(const char *szString,int nSize=-1);
        int InsertString(int nIndex, const char *szString,int nSize=-1);
```

```
40          int DeleteString(int nIndex);
            int SetString(int nIndex, const char *szString, int nSize=-1);
            int AppendString(int nIndex, const char *szString,int nSize=-1);
            char *GetString(int nIndex,int *pnSize=NULL);
            int GetCount();
45          int SetCount(int nCount);
            char *operator [] (int nIndex);
            CStringVektor &operator =(CStringVektor &s);
            int MakeIndex(const char *szString, int nLenght);
            int GetIndex(int nIndex);
50
            int MoveString(int nFrom, int nTo);
            int GetLength(int nIndex);
            char *ToText(const char *szFieldSep);
            void DeleteToText(char *szToText);
55          int FindString(const char *szString, int nLen=-1);
    };
    class CStringTable
    {
            CStringVektor **m_pVektors;
60          CStringVektor m_err;
            int m_nVektors;
            int m_nRowSize;
            int *m_pnOrders;
            int *m_pnHash;
65          int m_nHash;
            int m_nHashColumn;
            int m_nLastHash;
            BOOL m_bHashDirty;
            char nullstr[1];
70  public:
            CStringTable();
            CStringTable(CStringTable &stCopy);
            CStringTable(const char *szData,const char *szFieldSep, const char *↙
                ↳ szRowSep,BOOL bApo=FALSE);
            ~CStringTable();
75          int AddRow(char **pszRow, int nRow);
            int AddRow();
            int InsertRow(int nRow);
            int AddString(int nRow, const char *szString, int nString=-1);
            int AddString(int nRow, const std::string &szString, int nString=-1);
80          int AddInt(int nRow, intptr_t nVal);
            int DeleteRow(int nRow);
            int DeleteColumn(int nColumn);
            void Reset();
            int SetString(int nRow, int nIndex, char *szString, int nString=-1);
85          int AppendString(int nRow, int nIndex, char *szString,int nString=-1);
            int SetInt(int nRow, int nIndex, intptr_t nVal);
            char* GetString(int nRow, int nIndex,int *pnSize=NULL);
            int GetCount();
            int GetRowSize(int nRow=-1);
90
            CStringVektor &operator [] (int nRow);
            CStringTable &operator = (CStringTable &st);
            int ReadCSV(char *szFileName);

95          int Compare(int a, int b, int nColumn, int nType, int nOrder);
```

```
        void Swap(int a,  int b);
        void M3QSort(int nColumn,int nOrder=0,int nType=0,int left=0, int right
            =-1);

        int FindString(int nColumn, const char *szString, int nLength=-1);
        int FindStringN(int nColumn, const char *szString, int nLength=-1);
        int Save(char *szFile);
        int Load(char *szFile);

        int BuildHash(int nColumn,int nItems=-1);
        // Must be a hash build before this function is called!!
        int FindStringHash(int nColumn, char *szString,int nString=-1,int
            nPrevHit=-1);
        int GetHashColumn();

        // The table must be sorted with order 0 on this column!!
        int FindStringBinary(int nColumn, char *szString, int nLength=-1);

        int SplitString(const char *szData,const char *szFieldSep, const char *
            szRowSep,BOOL bApo=FALSE);
        char *ToText(const char *szFieldSep="\t", const char *szRowSep="\r\n");
        void DeleteToText(char *szToText);

        int MoveCol(int nFrom, int nTo);
        int MoveRow(int nFrom, int nTo);
};

// stristr
//
// Why is this function not included in the standard library?
//
// 000821 KR Commented
//
char *stristr(char *src, char *find);

#endif//__STRINGVECTOR_H__
```

35 common/tooltip.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/
```

```
20  #include <windows.h>
    #include <stdio.h>
    #include <string.h>
    #include <memory.h>
    #include "tooltip.h"
25  #include "StringVector.h"
    #include <cinttypes>
    #include "bitmap.h"

    HWND hToolTip;
30  HWND hSkugga;
    HFONT hfFont;
    BOOL bFirst=TRUE;
    LPGETTEXT g_lpfnGetText;
    LPARAM g_lParam;
35  BOOL g_bCenter=0;
    CStringTable g_stWindows;

    #ifdef _WIN64
    #define GCL_WNDPROC -24
40  #define GWL_WNDPROC -4
    #define GWL_USERDATA -21
    #endif

    typedef void (__stdcall *lpfnSetLayeredWindowAttributes)(HWND,int,int,int);
45  lpfnSetLayeredWindowAttributes g_lpfnSetLayeredWindowAttributes=NULL;

    BOOL IsActiveApp(HWND hWnd)
    {
            BOOL bActive=FALSE;
50
            hWnd=GetParent(hWnd);
            while(hWnd && !bActive){
                    if(GetActiveWindow()==hWnd)
                            bActive=TRUE;
55                  else
                            hWnd=GetParent(hWnd);
            }
            return bActive;
    }
60
    void SetRegions(HWND hWnd)
    {
            HRGN hRgn1, hRgn2, hRgn3;
            RECT r;
65          POINT p[3];
            GetWindowRect(hWnd,&r);
            hRgn1 = CreateRoundRectRgn(0,10,r.right-r.left+1,r.bottom-r.top+1,35,35)↙
                ↳ ;
            p[0].x=0;           p[0].y=0;
            p[1].x=3;           p[1].y=20;
70          p[2].x=15;          p[2].y=15;
            hRgn2 = CreatePolygonRgn(p,3,WINDING);
            hRgn3 = CreateRectRgn(0,0,0,0);
            CombineRgn(hRgn3,hRgn1,hRgn2,RGN_OR);
            SetWindowRgn(hWnd,hRgn3,TRUE);
```

```
75          DeleteObject(hRgn1);
            DeleteObject(hRgn2);
}

char szTemp[15120];
80 long WINAPI SkuggaProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
        PAINTSTRUCT ps;
        HDC hDC;
        RECT r;
85 //      int i;
        HBRUSH br;
        HBITMAP bmo, bm;

        if(uMsg==WM_CREATE){
90              SetWindowLong( hWnd, GWL_EXSTYLE, GetWindowLong(hWnd, ↙
                    ↳ GWL_EXSTYLE) | 0x00080000/*WS_EX_LAYERED*/);
                g_lpfnSetLayeredWindowAttributes(hWnd, 0, 100, 2/*LWA_ALPHA*/);
        }
        else if(uMsg==WM_PAINT){
                if(g_lpfnSetLayeredWindowAttributes)
95                      br=CreateSolidBrush(RGB(0,0,0));
                else
                        br=CreateSolidBrush(RGB(111,111,111));
                GetClientRect(hWnd,&r);
                BeginPaint(hWnd,&ps);
100             hDC = CreateCompatibleDC(ps.hdc);
                bm=create_bitmap(ps.hdc,r.right,r.bottom);
                bmo=(HBITMAP)SelectObject(hDC,bm);
                if(g_lpfnSetLayeredWindowAttributes){
                        int a = r.bottom;
105                     int nCol = 180;
                        for(;r.bottom>a/2;r.bottom--){
                                DeleteObject(br);
                                br = CreateSolidBrush(RGB(nCol,nCol,nCol));
                                nCol-=20;
110                             FillRect(ps.hdc,&r,br);
                                r.right--;
                                r.left++;
                                r.top++;
                                if(!nCol)
115                                     break;
                        }
                }
                else{
                        FillRect(hDC,&r,br);
120                     BitBlt(ps.hdc,0,0,r.right,r.bottom,hDC,0,0,SRCAND);
                }
                SelectObject(hDC,bmo);
                DeleteObject(bm);
                DeleteObject(br);
125             DeleteDC(hDC);
                EndPaint(hWnd,&ps);
/*              GetClientRect(hWnd,&r);
                hDC=BeginPaint(hWnd,&ps);
                for(i=-r.bottom;i<r.right;i+=2){
130                     MoveToEx(hDC,i,0,NULL);
```

```
                        LineTo(hDC, i+r.bottom, r.bottom);
                }
                EndPaint(hWnd,&ps);
*/      }

        return DefWindowProc(hWnd,uMsg,wParam,lParam);
}
BOOL g_bShow=FALSE;
int g_nPrevIndex=-1;
long WINAPI ToolTipProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
        char szTemp[15120];
        PAINTSTRUCT ps;
        HDC hDC;
        HFONT hfOld;
        RECT r;
        HRGN rgn;

        if(uMsg==WM_PAINT){
                SendMessage(hWnd,WM_GETTEXT,sizeof(szTemp),(LPARAM)szTemp);
                hDC=BeginPaint(hWnd,&ps);
                hfOld=(HFONT)SelectObject(hDC,(HFONT)GetStockObject(↙
                    ↳ ANSI_VAR_FONT));
                SetTextAlign(hDC,TA_LEFT|TA_TOP);
                SetBkMode(hDC,TRANSPARENT);
                GetClientRect(hWnd,&r);
                r.top=14;
                DrawText(hDC,szTemp,strlen(szTemp),&r,DT_CENTER|DT_TOP|↙
                    ↳ DT_EXPANDTABS);
                SelectObject(hDC,hfOld);
                rgn = CreateRectRgn(0,0,0,0);
                GetWindowRgn(hWnd,rgn);
                OffsetRgn(rgn,-1,-1);
                FrameRgn(hDC,rgn,(HBRUSH)GetStockObject(BLACK_BRUSH),1,1);
                DeleteObject(rgn);
                GetClientRect(hWnd,&r);
                MoveToEx(ps.hdc,0,r.bottom-1,NULL);
                LineTo(ps.hdc,r.right,r.bottom-1);
                EndPaint(hWnd,&ps);
        }
        if(uMsg==WM_TIMER){
                POINT p;
                GetCursorPos(&p);
                char szTmp[256];
                HWND hw = WindowFromPoint(p);
                snprintf(szTmp,256,"%" PRIdPTR,(intptr_t)hw);
                int n=-1;
//              if(IsActiveApp(hWnd))
                        n = g_stWindows.FindStringHash(0,szTmp);
                char *sz;
                if(n!=-1){
                        sz = g_lpfnGetText(GetDlgCtrlID(hw),atoi(g_stWindows[n↙
                            ↳ ][2]));
                        if(sz && *sz)
                                strcpy(szTemp,sz);
                        else
                                n=-1;
```

```
185                     }
                        if(g_nPrevIndex!=-1 && g_nPrevIndex!=n)
                                n=-1;
                        if(n==-1){
                                if(g_bShow){
190                                     g_nPrevIndex=-1;
                                        g_bShow=FALSE;
                                        ShowWindow(hToolTip,SW_HIDE);
                                        ShowWindow(hSkugga,SW_HIDE);
                                }
195                             return DefWindowProc(hWnd,uMsg,wParam,lParam);
                        }

                        if(g_nPrevIndex==-1){
                                g_bShow=TRUE;
200                             g_nPrevIndex=n;

                                hDC=GetDC(hToolTip);
                                hfOld=(HFONT)SelectObject(hDC,hfFont);
                                memset(&r,0,sizeof(RECT));
205                             DrawText(hDC,szTemp,strlen(szTemp),&r,DT_CALCRECT);
                                SelectObject(hDC,hfOld);
                                ReleaseDC(hToolTip,hDC);

                                RECT wr;
210                             GetClientRect(GetDesktopWindow(),&wr);
                                if(p.x+15+r.right+12>wr.right)
                                        p.x=wr.right-15-r.right-20;
                                if(p.y+15+r.bottom+12>wr.bottom)
                                        p.y=wr.bottom-15-r.bottom-30;
215                             SetWindowText(hToolTip,szTemp);
                                SetWindowPos(hSkugga,HWND_TOPMOST,p.x+15+10,p.y+15+4,r.↙
                                    ↪ right+20-4,r.bottom+20,SWP_NOACTIVATE);
                                SetRegions(hSkugga);
                                ShowWindow(hSkugga,SW_SHOWNA);
                                SetWindowPos(hToolTip,HWND_TOPMOST,p.x+15,p.y+15,r.right↙
                                    ↪ +20,r.bottom+20,SWP_NOACTIVATE);
220                             SetRegions(hToolTip);
                                ShowWindow(hToolTip,SW_SHOWNA);
                        }
                }

225             return DefWindowProc(hWnd,uMsg,wParam,lParam);
        }
        void ErrorText(DWORD dwError)
        {
                TCHAR szMsgBuf[500];
230
                dwError = GetLastError();
                FormatMessage(
                        FORMAT_MESSAGE_FROM_SYSTEM,
                        NULL,
235                     dwError,
                        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                        szMsgBuf,
                        500,
                        NULL );
```

```
240             MessageBox(NULL,szMsgBuf," Error",MB_ICONSTOP|MB_OK);
        }
        void CollectObjects(HWND hwNext,HWND hwParent,LPARAM lParam)
        {
                hwNext=GetWindow(hwNext,GW_CHILD);
245             while(hwNext){
                        g_stWindows.AddRow();
                        g_stWindows.AddInt(g_stWindows.GetCount()-1,(intptr_t)hwNext);
                        g_stWindows.AddInt(g_stWindows.GetCount()-1,(intptr_t)hwParent);
                        g_stWindows.AddInt(g_stWindows.GetCount()-1,(intptr_t)lParam);
250                     CollectObjects(hwNext,hwParent,lParam);
                        hwNext=GetWindow(hwNext,GW_HWNDNEXT);
                }
        }

255     BOOL InitToolTip(HWND hWnd,HINSTANCE hInst,LPGETTEXT lpfnGetText,LPARAM lParam,↙
            ↳ BOOL bCenter,char *szFont, int nFont)
        {
                if(!g_lpfnSetLayeredWindowAttributes){
                        HINSTANCE hMod = LoadLibrary("User32.dll");
                        g_lpfnSetLayeredWindowAttributes =
260                             (lpfnSetLayeredWindowAttributes)GetProcAddress(hMod,"↙
                                    ↳ SetLayeredWindowAttributes");
                }
                g_bCenter = bCenter;
                g_lpfnGetText = lpfnGetText;
                g_lParam = lParam;
265             WNDCLASS wc;
                HBRUSH brGul;
                if(szFont){
                        hfFont = CreateFont(nFont,0,0,0,0,0,0,0,0,0,0,0,0,szFont);
                }
270             else
                        hfFont = (HFONT)GetStockObject(ANSI_VAR_FONT);

                if(bFirst){
                        bFirst=FALSE;
275                     brGul=CreateSolidBrush(RGB(255,255,200));
                        memset(&wc,0,sizeof(WNDCLASS));
                        wc.lpfnWndProc=(WNDPROC)ToolTipProc;
                        wc.hInstance=(HINSTANCE)hInst;
                        wc.hCursor= LoadCursor(NULL,IDC_ARROW);
280                     wc.hbrBackground=brGul;
                        wc.lpszClassName="TOOLTIP";
                        HWND hwParent = hWnd;
                        while(GetParent(hwParent))
                                hwParent = GetParent(hwParent);
285                     if(RegisterClass(&wc))
                                hToolTip=CreateWindow("TOOLTIP","",WS_POPUP|WS_BORDER↙
                                    ↳ ,0,0,120,120,hwParent,NULL,(HINSTANCE)hInst,0);
                        //SetWindowPos(hToolTip,HWND_TOPMOST,0,0,0,0,SWP_NOMOVE|↙
                            ↳ SWP_NOSIZE);
                        wc.lpfnWndProc=(WNDPROC)SkuggaProc;
                        wc.hbrBackground=NULL;
290                     wc.lpszClassName="SKUGGA";
                        if(RegisterClass(&wc))
                                hSkugga=CreateWindow("SKUGGA","",WS_POPUP,0,0,120,120,↙
```

```
                                        ↪ hwParent,NULL,hInst,0);
                        //SetWindowPos(hSkugga,hToolTip,0,0,0,0,SWP_NOMOVE|SWP_NOSIZE);
                }
295             CollectObjects(hWnd,hWnd,lParam);
                g_stWindows.BuildHash(0);
                SetTimer(hToolTip,0,100,NULL);
                return TRUE;
        }
300     void ExitToolTip(HWND hWnd)
        {
                int i;
                for(i=0;i<g_stWindows.GetCount();i++){
                        intptr_t h = 0;
305                     sscanf(g_stWindows[i][1], "%" SCNdPTR, &h);
                        if(h==(intptr_t)hWnd)
                                g_stWindows.DeleteRow(i--);
                }
        }
```

36 common/tooltip.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen
 5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15
You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20  #ifndef __TOOLTIP_H__
    #define __TOOLTIP_H__

    typedef char * (*LPGETTEXT)(int nID,LPARAM lParam);
    BOOL InitToolTip(HWND hWnd,HINSTANCE hInst,LPGETTEXT lpfnGetText,LPARAM lParam,↙
        ↪ BOOL bCenter=FALSE,char *szFont=NULL, int nFont=0);
25  long WINAPI ButtonProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam);
    void ExitToolTip(HWND hWnd);

    #endif//__TOOLTIP_H__
```

37 examples/examples.kfs

```
ZoomSize: 4
MaxReferences: 10000
GlitchLowTolerance: 1
```

```
ApproxLowTolerance: 1
AutoApproxTerms: 1
ApproxTerms: 10
WindowWidth: 512
WindowHeight: 512
WindowTop: 249
WindowLeft: 698
WindowBottom: 582
WindowRight: 524
ImageWidth: 512
ImageHeight: 512
ThreadsPerCore: 1
AnimateZoom: 1
ArbitrarySize: 1
ReuseReference: 0
AutoSolveGlitches: 1
Guessing: 0
SolveGlitchNear: 0
NoApprox: 0
Mirror: 0
LongDoubleAlways: 0
FloatExpAlways: 0
AutoIterations: 1
ShowGlitches: 0
NoReuseCenter: 0
IsolatedGlitchNeighbourhood: 0
JitterSeed: 1
JitterShape: 0
JitterScale: 1
Derivatives: 0
```

38 examples/examples.sh

```
#!/bin/bash
rm examples.html
xsltproc examples.xsl ../formula/formula.xml | tee /dev/stderr |
while read FractalType Power Name
do
    stem="${FractalType}-${Power}"
    echo >> examples.html "<a href='formula/${stem}.kfr' title='${Name} (power ${Power})'><img src='formula/${stem}.png' alt='${Name} (power ${Power})' /></a>"
    cat <<EOF | sed -s 's/$/\r/g' > "${stem}.kfr"
Re: 0
Im: 0
Zoom: 1
Iterations: 1000
IterDiv: 0.010000
SmoothMethod: 0
ColorMethod: 7
Differences: 3
ColorOffset: 0
Rotate: 0.000000
Ratio: 360.000000
Colors: 255,255,255,128,0,64,160,0,0,192,128,0,64,128,0,0,255,255,64,128,255,0,0,255,
```

```
InteriorColor: 0,0,0,
Smooth: 1
MultiColor: 0
BlendMC: 0
MultiColors:
Power: ${Power}
FractalType: ${FractalType}
Slopes: 1
SlopePower: 50
SlopeRatio: 20
SlopeAngle: 45
imag: 1
real: 1
SeedR: 0
SeedI: 0
FactorAR: 1
FactorAI: 0
EOF
    ../kf.exe -s examples.kfs -l "${stem}.kfr" -p "${stem}.png" --log warn
    pngtopnm -text "${stem}.txt" "${stem}.png" > "${stem}.ppm"
    pnmscale -reduce 4 "${stem}.ppm" |
    pnmtopng -text "${stem}.txt" -force -interlace -compression 9 > "${stem}.png
    "
    rm "${stem}.ppm" "${stem}.txt"
done
```

39 examples/examples.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" />
<xsl:template match="/"><xsl:for-each select="formulas/group/formula">
<xsl:value-of select="../@type" /><xsl:text> </xsl:text><xsl:value-of select="
    @power" /><xsl:text> </xsl:text><xsl:value-of select="../@name" /><xsl:
    text>
</xsl:text>
</xsl:for-each></xsl:template></xsl:stylesheet>
```

40 formula/common.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
```

```
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <windows.h>
#include <stdint.h>
#include "../fraktal_sft/floatexp.h"
#include "formula.h"

template <typename T> T sgn(const T &a)
{
  return (a > 0) - (a < 0);
}

#if 0

static inline double d_add(const double a, const double b)
{
  return a + b;
}

static inline double d_addi(const double a, const int b)
{
  return a + b;
}

static inline double d_sub(const double a, const double b)
{
  return a - b;
}

static inline double d_isub(const int a, const double b)
{
  return a - b;
}

static inline double d_neg(const double a)
{
  return -a;
}

static inline double d_abs(const double a)
{
  return a < 0.0 ? -a : a;
}

static inline double d_sgn(const double a)
{
  return (a > 0.0) - (a < 0.0);
}

static inline double d_imul(const int a, const double b)
{
  return a * b;
}

static inline double d_muli(const double a, const int b)
{
```

```
  return a * b;
}

static inline double d_mul(const double a, const double b)
{
  return a * b;
}

static inline double d_pow(const double a, const int b)
{
  double r = 1.0;
  for (int i = 0; i < b; ++i)
  {
    r *= a;
  }
  return r;
}

static inline int d_cmp(const double a, const double b)
{
  return ((int)(a > b)) - ((int)(a < b));
}

static inline double d_diffabs(const double c, const double d)
{
  double cd = c + d;
  if (c >= 0.0)
  {
    if (cd >= 0.0)
    {
      return d;
    }
    else
    {
      return -d - 2.0 * c;
    }
  }
  else
  {
    if (cd > 0.0)
    {
      return d + 2.0 * c;
    }
    else
    {
      return -d;
    }
  }
}

static inline long double ld_add(const long double a, const long double b)
{
  return a + b;
}

static inline long double ld_addi(const long double a, const int b)
{
```

```
  return a + b;
}

static inline long double ld_sub(const long double a, const long double b)
{
  return a - b;
}

static inline long double ld_isub(const int a, const long double b)
{
  return a - b;
}

static inline long double ld_neg(const long double a)
{
  return -a;
}

static inline long double ld_abs(const long double a)
{
  return a < 0.0 ? -a : a;
}

static inline long double ld_sgn(const long double a)
{
  return (a > 0.0) - (a < 0.0);
}

static inline long double ld_imul(const int a, const long double b)
{
  return a * b;
}

static inline long double ld_muli(const long double a, const int b)
{
  return a * b;
}

static inline long double ld_mul(const long double a, const long double b)
{
  return a * b;
}

static inline long double ld_pow(const long double a, const int b)
{
  long double r = 1.0;
  for (int i = 0; i < b; ++i)
  {
    r *= a;
  }
  return r;
}

static inline int ld_cmp(const long double a, const long double b)
{
  return ((int)(a > b)) - ((int)(a < b));
}
```

```
static inline long double ld_diffabs(const long double c, const long double d)
{
  long double cd = c + d;
  if (c >= 0.0L)
  {
    if (cd >= 0.0L)
    {
      return d;
    }
    else
    {
      return - d - 2.0L * c;
    }
  }
  else
  {
    if (cd > 0.0L)
    {
      return d + 2.0L * c;
    }
    else
    {
      return -d;
    }
  }
}

static inline int64_t as_long(double d)
{
  union {
    double d;
    int64_t i;
  } u;
  u.d = d;
  return u.i;
}

static inline double as_double(int64_t i)
{
  union {
    double d;
    int64_t i;
  } u;
  u.i = i;
  return u.d;
}

static inline double fe_double(floatexp f)
{
  if (f.exp < -1020)
  {
    return 0.0;
  }
  if (f.exp > 1020)
  {
    return f.val / 0.0;
```

```
245   }
      return as_double((as_long(f.val) & 0x800FFFFFFFFFFFFFLL) | ((f.exp + 1023) << ↙
          ↳ 52));
    }

    static inline floatexp fe_floatexp(const double val, const int64_t exp)
250 {
      int64_t f_exp = ((as_long(val) & 0x7FF0000000000000LL) >> 52) - 1023;
      double f_val = as_double((as_long(val) & 0x800FFFFFFFFFFFFFLL) | 0↙
          ↳ x3FF0000000000000LL);
      floatexp fe(f_val, f_exp + exp, 0);
      return fe;
255 }

    static inline floatexp fe_abs(const floatexp f)
    {
      floatexp fe(d_abs(f.val), f.exp, 0);
260   return fe;
    }

    static inline floatexp fe_sgn(const floatexp a)
    {
265   return floatexp((a.val > 0.0) - (a.val < 0.0));
    }

    static inline floatexp fe_neg(const floatexp f)
    {
270   floatexp fe(-f.val, f.exp, 0);
      return fe;
    }

    static inline floatexp fe_muli(const floatexp a, const int b)
275 {
      return fe_floatexp(a.val * b, a.exp);
    }

    static inline floatexp fe_imul(const int a, const floatexp b)
280 {
      return fe_floatexp(a * b.val, b.exp);
    }

    static inline floatexp fe_mul(const floatexp a, const floatexp b)
285 {
      return fe_floatexp(a.val * b.val, a.exp + b.exp);
    }

    static inline floatexp fe_mul_2si(const floatexp a, const int64_t b)
290 {
      floatexp fe(a.val, a.exp + b, 0);
      return fe;
    }

295 static inline floatexp fe_div(const floatexp a, const floatexp b)
    {
      return fe_floatexp(a.val / b.val, a.exp - b.exp);
    }
```

```
300 static inline floatexp fe_div_2si(const floatexp a, const int64_t b)
    {
      floatexp fe(a.val, a.exp - b, 0);
      return fe;
    }
305
    static inline floatexp fe_add(const floatexp a, const floatexp b)
    {
      if (a.exp > b.exp)
      {
310     floatexp c(b.val, b.exp - a.exp, 0);
        return fe_floatexp(a.val + fe_double(c), a.exp);
      }
      else
      {
315     floatexp c(a.val, a.exp - b.exp, 0);
        return fe_floatexp(fe_double(c) + b.val, b.exp);
      }
    }

320 static inline floatexp fe_addi(const floatexp a, const int b)
    {
      return fe_add(a, fe_floatexp(double(b), 0));
    }

325 static inline floatexp fe_sub(const floatexp a, const floatexp b)
    {
      return fe_add(a, fe_neg(b));
    }

330 static inline floatexp fe_isub(const int a, const floatexp b)
    {
      return fe_addi(fe_neg(b), a);
    }

335 static inline int fe_cmp(const floatexp a, const floatexp b)
    {
      if (a.val > 0)
      {
        if (b.val <= 0)
340     {
          return 1;
        }
        else if (a.exp > b.exp)
        {
345       return 1;
        }
        else if (a.exp < b.exp)
        {
          return -1;
350     }
        else
        {
          return d_cmp(a.val, b.val);
        }
355   }
      else
```

```
  {
    if (b.val > 0)
    {
      return -1;
    }
    else if (a.exp > b.exp)
    {
      return -1;
    }
    else if (a.exp < b.exp)
    {
      return 1;
    }
    else
    {
      return d_cmp(a.val, b.val);
    }
  }
}

static inline floatexp fe_diffabs(const floatexp c, const floatexp d)
{
  floatexp cd = fe_add(c, d);
  if (c.val >= 0.0)
  {
    if (cd.val >= 0.0)
    {
      return d;
    }
    else
    {
      return fe_neg(fe_add(d, fe_mul_2si(c, 1)));
    }
  }
  else
  {
    if (cd.val > 0.0)
    {
      return fe_add(d, fe_mul_2si(c, 1));
    }
    else
    {
      return fe_neg(d);
    }
  }
}

static inline floatexp fe_pow(const floatexp a, int b)
{
  floatexp r = fe_floatexp(1.0, 0);
  for (int i = 0; i < b; ++i)
  {
    r = fe_mul(r, a);
  }
  return r;
}
```

```
typedef struct
{
  double re;
  double im;
} dcomplex;

static inline dcomplex dc_neg(const dcomplex a)
{
  dcomplex dc = { -a.re, -a.im };
  return dc;
}

static inline dcomplex dc_add(const dcomplex a, const dcomplex b)
{
  dcomplex dc = { a.re + b.re, a.im + b.im };
  return dc;
}

static inline dcomplex dc_addi(const dcomplex a, const int b)
{
  dcomplex dc = { a.re + b, a.im };
  return dc;
}

static inline dcomplex dc_sub(const dcomplex a, const dcomplex b)
{
  dcomplex dc = { a.re - b.re, a.im - b.im };
  return dc;
}

static inline dcomplex dc_isub(const int a, const dcomplex b)
{
  dcomplex dc = { a - b.re, - b.im };
  return dc;
}

static inline dcomplex dc_imul(const int a, const dcomplex b)
{
  dcomplex dc = { a * b.re, a * b.im };
  return dc;
}

static inline dcomplex dc_muli(const dcomplex a, const int b)
{
  dcomplex dc = { a.re * b, a.im * b };
  return dc;
}

static inline dcomplex dc_mul(const dcomplex a, const dcomplex b)
{
  dcomplex dc = { a.re * b.re - a.im * b.im, a.re * b.im + a.im * b.re };
  return dc;
}

static inline dcomplex dc_pow(const dcomplex a, const int b)
{
  dcomplex r = { 1.0, 0.0 };
```

```
  for (int i = 0; i < b; ++i)
  {
    r = dc_mul(r, a);
  }
  return r;
}

typedef struct
{
  long double re;
  long double im;
} ldcomplex;

static inline ldcomplex ldc_neg(const ldcomplex a)
{
  ldcomplex dc = { -a.re, -a.im };
  return dc;
}

static inline ldcomplex ldc_add(const ldcomplex a, const ldcomplex b)
{
  ldcomplex dc = { a.re + b.re, a.im + b.im };
  return dc;
}

static inline ldcomplex ldc_addi(const ldcomplex a, const int b)
{
  ldcomplex ldc = { a.re + b, a.im };
  return ldc;
}

static inline ldcomplex ldc_sub(const ldcomplex a, const ldcomplex b)
{
  ldcomplex dc = { a.re - b.re, a.im - b.im };
  return dc;
}

static inline ldcomplex ldc_isub(const int a, const ldcomplex b)
{
  ldcomplex dc = { a - b.re, - b.im };
  return dc;
}

static inline ldcomplex ldc_imul(const int a, const ldcomplex b)
{
  ldcomplex dc = { a * b.re, a * b.im };
  return dc;
}

static inline ldcomplex ldc_muli(const ldcomplex a, const int b)
{
  ldcomplex dc = { a.re * b, a.im * b };
  return dc;
}

static inline ldcomplex ldc_mul(const ldcomplex a, const ldcomplex b)
{
```

```
  ldcomplex dc = { a.re * b.re - a.im * b.im, a.re * b.im + a.im * b.re };
  return dc;
}

static inline ldcomplex ldc_pow(const ldcomplex a, const int b)
{
  ldcomplex r = { 1.0, 0.0 };
  for (int i = 0; i < b; ++i)
  {
    r = ldc_mul(r, a);
  }
  return r;
}

typedef struct
{
  floatexp re;
  floatexp im;
} fecomplex;

static inline fecomplex fec_neg(const fecomplex a)
{
  fecomplex fec = { fe_neg(a.re), fe_neg(a.im) };
  return fec;
}

static inline fecomplex fec_add(const fecomplex a, const fecomplex b)
{
  fecomplex fec = { fe_add(a.re, b.re), fe_add(a.im, b.im) };
  return fec;
}

static inline fecomplex fec_addi(const fecomplex a, const int b)
{
  fecomplex fec = { fe_addi(a.re, b), a.im };
  return fec;
}

static inline fecomplex fec_sub(const fecomplex a, const fecomplex b)
{
  fecomplex fec = { fe_sub(a.re, b.re), fe_sub(a.im, b.im) };
  return fec;
}

static inline fecomplex fec_isub(const int a, const fecomplex b)
{
  fecomplex fec = { fe_isub(a, b.re), fe_neg(b.im) };
  return fec;
}

static inline fecomplex fec_imul(const int a, const fecomplex b)
{
  fecomplex fec = { fe_imul(a, b.re), fe_imul(a, b.im) };
  return fec;
}

static inline fecomplex fec_muli(const fecomplex a, const int b)
```

```
585 {
      fecomplex fec = { fe_muli(a.re, b), fe_muli(a.im, b) };
      return fec;
    }

590 static inline fecomplex fec_mul(const fecomplex a, const fecomplex b)
    {
      fecomplex fec = { fe_sub(fe_mul(a.re, b.re), fe_mul(a.im, b.im)), fe_add(↙
          ↳ fe_mul(a.re, b.im), fe_mul(a.im, b.re)) };
      return fec;
    }
595
    static inline fecomplex fec_pow(const fecomplex a, const int b)
    {
      fecomplex r = { fe_floatexp(1.0, 0), fe_floatexp(0.0, 0) };
      for (int i = 0; i < b; ++i)
600   {
        r = fec_mul(r, a);
      }
      return r;
    }
605
    static inline bool gezero(const double &x) { return x >= 0.0; }
    static inline bool gtzero(const double &x) { return x >  0.0; }
    static inline bool gezero(const long double &x) { return x >= 0.0; }
    static inline bool gtzero(const long double &x) { return x >  0.0; }
610 static inline bool gezero(const floatexp &x) { return x.val >= 0.0; }
    static inline bool gtzero(const floatexp &x) { return x.val >  0.0; }

    static inline long double ConvertFromFixedFloat(const CFixedFloat &f)
    {
615   using std::ldexp;
      signed long int e = 0;
      long double l = mpfr_get_ld_2exp(&e, f.m_f.backend().data(), MPFR_RNDN);
      l = ldexp(l, e);
      return l;
620 }

    #endif

    static inline double mpfr_get(const mpfr_t x, const double &t, mpfr_rnd_t rnd)
625 {
      (void) t;
      return mpfr_get_d(x, rnd);
    }

630 static inline long double mpfr_get(const mpfr_t x, const long double &t, ↙
        ↳ mpfr_rnd_t rnd)
    {
      (void) t;
      return mpfr_get_ld(x, rnd);
    }
635
    static inline floatexp mpfr_get(const mpfr_t x, const floatexp &t, mpfr_rnd_t ↙
        ↳ rnd)
    {
      (void) t;
```

```
  signed long int e = 0;
  double d = mpfr_get_d_2exp(&e, x, rnd);
  return floatexp(d, e);
}


static inline bool all(const int2 &i) {
  return (((i[0] && i[1])));
}

static inline bool any(const int2 &i) {
  return (((i[0] || i[1])));
}


static inline bool all(const int4 &i) {
  return (((i[0] && i[1]) && (i[2] && i[3])));
}

static inline bool any(const int4 &i) {
  return (((i[0] || i[1]) || (i[2] || i[3])));
}


static inline bool all(const int8 &i) {
  return (((i[0] && i[1]) && (i[2] && i[3])) &&
          ((i[4] && i[5]) && (i[6] && i[7])));
}

static inline bool any(const int8 &i) {
  return (((i[0] || i[1]) || (i[2] || i[3])) ||
          ((i[4] || i[5]) || (i[6] || i[7])));
}


static inline bool all(const int16 &i) {
  return (((i[0] && i[1]) && (i[2] && i[3])) &&
          ((i[4] && i[5]) && (i[6] && i[7]))) &&
          (((i[8] && i[9]) && (i[10] && i[11])) &&
          ((i[12] && i[13]) && (i[14] && i[15])));
}

static inline bool any(const int16 &i) {
  return (((i[0] || i[1]) || (i[2] || i[3])) ||
          ((i[4] || i[5]) || (i[6] || i[7]))) ||
          (((i[8] || i[9]) || (i[10] || i[11])) ||
          ((i[12] || i[13]) || (i[14] || i[15])));
}

template <typename T>
static inline T abs(const T &x) { return x < 0.0 ? -x : x; }

template<typename S, typename T>
static inline T diffabs(const S &c, const T &d)
{
  const T cd = c + d;
  const T c2d = 2.0 * c + d;
```

```
  return c >= 0.0 ? cd >= 0.0 ? d : -c2d : cd > 0.0 ? c2d : -d;
}
```

41 formula/formula.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_FORMULA_H
#define KF_FORMULA_H 1

#include <cstdint>

class CFixedFloat;
class floatexp;

// https://fractalforums.org/fractal-mathematics-and-new-theories/28/perturbation-theory/487/msg3170#msg3170
// |2w'(w+z)+1|/|delta0|+|w|(|w+2z|+|w|+2|z|)<epsilon/h
template <typename R>
static inline R mag(const R &x, const R &y)
{
  using std::abs;
  return abs(x) + abs(y);
}

template <typename R>
static inline bool type_0_power_2_pixel_has_glitched(R cr, R ci, R zr, R zi, R Zr, R Zi, R dr, R di, R e, R h)
{
  R Zzr = Zr + zr;
  R Zzi = Zi + zi;
  R a = mag(2 * (dr * Zzr - di * Zzi) + 1, 2 * (dr * Zzi + di * Zzr));
  R b = mag(cr, ci) + mag(zr, zi) * (mag(zr + 2 * Zr, zi + 2 * Zi) + mag(zr, zi) + 2 * mag(Zr, Zi));
  return a * h < b * e;
}

template <typename R, typename T> inline R broadcast(T x) { return R(x); }

typedef int64_t int2 __attribute__ ((vector_size (16)));
```

```
50  typedef double double2 __attribute__ ((vector_size (16)));
    template <> inline double2 broadcast<double2,double>(double x) { double2 r = { x↙
        ↳ , x }; return r; }

    typedef int64_t int4 __attribute__ ((vector_size (32)));
    typedef double double4 __attribute__ ((vector_size (32)));
55  template <> inline double4 broadcast<double4,double>(double x) { double4 r = { x↙
        ↳ , x, x, x }; return r; }

    typedef int64_t int8 __attribute__ ((vector_size (64)));
    typedef double double8 __attribute__ ((vector_size (64)));
    template <> inline double8 broadcast<double8,double>(double x) { double8 r = { x↙
        ↳ , x, x, x, x, x, x, x }; return r; }
60
    typedef int64_t int16 __attribute__ ((vector_size (128)));
    typedef double double16 __attribute__ ((vector_size (128)));
    template <> inline double16 broadcast<double16,double>(double x) { double16 r = ↙
        ↳ { x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x }; return r; }

65  // reference

    template <typename T>
    bool reference
      ( const int m_nFractalType, const int m_nPower
70    , T *m_db_dxr, T *m_db_dxi, double *m_db_z
      , bool &m_bStop, int64_t &m_nRDone, int64_t &m_nGlitchIter, int64_t &↙
          ↳ m_nMaxIter
      , const CFixedFloat &Cr, const CFixedFloat &Ci
      , const double g_SeedR, const double g_SeedI
      , const double g_FactorAR, const double g_FactorAI
75    , const double terminate, const double g_real, const double g_imag
      , const bool m_bGlitchLowTolerance
      , int64_t &antal, double &test1, double &test2
      , double &Xxr, double &Xxi
      );
80
    // reference with derivatives

    template <typename S, typename T>
    bool reference
85    ( const int m_nFractalType, const int m_nPower
      , T *m_db_dxr, T *m_db_dxi, double *m_db_z
      , bool &m_bStop, int64_t &m_nRDone, int64_t &m_nGlitchIter, int64_t &↙
          ↳ m_nMaxIter
      , const CFixedFloat &Cr0, const CFixedFloat &Ci0
      , const double g_SeedR, const double g_SeedI
90    , const double g_FactorAR, const double g_FactorAI
      , const double terminate, const double g_real, const double g_imag
      , const bool m_bGlitchLowTolerance
      , int64_t &antal, double &test1, double &test2
      , double &Xxr, double &Xxi
95    , S &dr0, S &di0
      , const S &daa, const S &dab, const S &dba, const S &dbb
      );

    // perturbation
100
```

```
template <typename T>
bool perturbation
  ( const int m_nFractalType, const int m_nPower
  , const T *m_dxr, const T *m_dxi, const double *m_db_z
  , int64_t &antal, double &test1, double &test2, bool &bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , T &xr, T &xi
  , const T &cr, const T &ci
  );

// perturbation with derivatives

template <typename T>
bool perturbation
  ( const int m_nFractalType, const int m_nPower
  , const T *m_dxr, const T *m_dxi, const double *m_db_z
  , int64_t &antal, double &test1, double &test2, bool &bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , T &xr, T &xi
  , const T &cr, const T &ci
  , T &dr, T &di
  , const T &e, const T &h
  , const T &daa, const T &dab, const T &dba, const T &dbb
  );

// perturbation with SIMD

template <typename intN, typename doubleN>
bool perturbation
  ( const int m_nFractalType, const int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , intN &antal, doubleN &test1, doubleN &test2, intN &bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , doubleN &xr, doubleN &xi
  , const doubleN &cr, const doubleN &ci
  , const int64_t chunksize
  );

// perturbation with SIMD and derivatives

template <typename intN, typename doubleN>
bool perturbation
  ( const int m_nFractalType, const int m_nPower
  , const double *m_dxr, const double *m_dxi, const double *m_db_z
  , intN &antal, doubleN &test1, doubleN &test2, intN &bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , doubleN &xr, doubleN &xi
  , const doubleN &cr, const doubleN &ci
  , doubleN &dr, doubleN &di
```

```
    , const double e, const double h
    , const doubleN &daa, const doubleN &dab, const doubleN &dba, const doubleN &dbb
    , const int64_t chunksize
    );

// perturbation with scaling

template <typename T>
bool perturbation
    ( const int m_nFractalType, const int m_nPower
    , const T *m_db_dxr, const T *m_db_dxi, const double *m_db_z
    , int64_t &antal, double &test1, double &test2, bool &bGlitch
    , const double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , T &xr, T &xi
    , const T &cr, const T &ci
    , const T &s, const T &S
    );

// perturbation with SIMD and scaling

template <typename intN, typename doubleN>
bool perturbation
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , intN &antal0, doubleN &test10, doubleN &test20, intN &bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , doubleN &xr00, doubleN &xi00
    , const doubleN &cr0, const doubleN &ci0
    , const int64_t chunksize
    , const double s, const double S
    );

// perturbation with derivatives and scaling

template <typename D, typename Z>
bool perturbation
    ( int m_nFractalType, int m_nPower
    , const Z *m_db_dxr, const Z *m_db_dxi, const double *m_db_z
    , int64_t &antal0, double &test10, double &test20, bool &bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , Z &xr0, Z &xi0
    , const Z &cr, const Z &ci
    , D &dr0, D &di0
    , const D &e, const D &h
    , const D &daa, const D &dab, const D &dba, const D &dbb
    , const Z &s, const Z &S
    );

// miscellaneous
```

```
bool scaling_supported(const int m_nFractalType, const int m_nPower, const bool derivatives);
void combo5_addstrings(HWND hWnd, const int combo);
int validate_power_for_fractal_type(const int m_nFractalType, const int m_nPower);
void update_power_dropdown_for_fractal_type(HWND hWnd, const int combo, const int m_nFractalType, const int m_nPower);

#endif
```

42 formula/formula.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program. If not, see <https://www.gnu.org/licenses/>.
-->
<formulas>
  <group type="0" name="Mandelbrot">

    <!--

      IMPORTANT NOTE:

      Mandelbrot power 2 has separate threaded implementations for reference
      calculations, which need to be updated by hand when you change the glitch
      threshold (or the reference formula). You can find those implementations
      in these files:
        - fraktal_sft/long_double_reference.cpp
        - fraktal_sft/floatexp_reference.cpp

      Mandelbrot power 2 and 3 have separate scaled (long) double
      implementations for perturbation calculations. You can find them in:
        - fraktal_sft/double_perturbation.cpp
        - fraktal_sft/long_double_perturbation.cpp

      Mandelbrot power 2 and 3 have separate implementations for reference
      calculations with scaled (long) double (that use long double or floatexp
      for derivative to avoid overflow), which need to be updated by hand too.
      You can find those implementations in:
        - fraktal_sft/double_reference.cpp
        - fraktal_sft/long_double_reference.cpp

```

```
      Mandelbrot power > 10 is implemented separately in various files.

      Mandelbrot series approximation is in:
        - fraktal_sft/floatexp_approximation.cpp

    -->
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = Xr2 - Xi2 + Cr;
        Xin = sqr(Xr + Xi) - Xr2 - Xi2 + Ci;
      </reference>
      <perturbation t="R">
        xrn = (2 * Xr + xr) * xr - (2 * Xi + xi) * xi + cr;
        xin = 2 * (Xxr * xi + Xi * xr) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * dxa)) + (( - 2 * (Xxi * dya)) + daa));
        dxbn = ((2 * (Xxr * dxb)) + (( - 2 * (Xxi * dyb)) + dab));
        dyan = ((2 * (Xxi * dxa)) + ((2 * (Xxr * dya)) + dba));
        dybn = ((2 * (Xxi * dxb)) + ((2 * (Xxr * dyb)) + dbb));
      </derivative>
      <scaled t="R">
        xrn = (2 * Xr + xr * s) * xr - (2 * Xi + xi * s) * xi + cr;
        xin = 2 * ((Xr + xr * s) * xi + Xi * xr) + ci;
      </scaled>
    </formula>

    <formula power="3" glitch="0.000001">
      <reference t="R">
        Xrn = Xr * (Xr2 - 3 * Xi2) + Cr;
        Xin = (3 * Xr2 - Xi2) * Xi + Ci;
      </reference>
      <perturbation t="R">
        xrn = 3 * Xr2 * xr - 6 * Xr * Xi * xi - 3 * Xi2 * xr + 3 * Xr * xr2 - 3
* Xr * xi2 - 6 * Xi * xr * xi + xr * xr2 - 3 * xr * xi2 + cr;
        xin = 3 * Xr2 * xi + 6 * Xr * Xi * xr - 3 * Xi2 * xi + 6 * Xr * xr * xi
+ 3 * Xi * xr2 - 3 * Xi * xi2 + 3 * xr2 * xi - xi * xi2 + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((3 * (Xxr * (Xxr * dxa))) + (( - 3 * (Xxi * (Xxi * dxa))) + ((
- 6 * (Xxi * (Xxr * dya))) + daa)));
        dxbn = ((3 * (Xxr * (Xxr * dxb))) + (( - 3 * (Xxi * (Xxi * dxb))) + ((
- 6 * (Xxi * (Xxr * dyb))) + dab)));
        dyan = ((3 * (Xxr * (Xxr * dya))) + (( - 3 * (Xxi * (Xxi * dya))) + ((6
* (Xxi * (Xxr * dxa))) + dba)));
        dybn = ((3 * (Xxr * (Xxr * dyb))) + (( - 3 * (Xxi * (Xxi * dyb))) + ((6
* (Xxi * (Xxr * dxb))) + dbb)));
      </derivative>
      <scaled t="R">
        xrn = 3 * Xr2 * xr - 6 * Xr * Xi * xi - 3 * Xi2 * xr + 3 * Xr * xr2 * s
- 3 * Xr * xi2 * s - 6 * Xi * xr * xi * s + xr2 * xr * s * s - 3 * xr * xi2 * s
* s + cr;
        xin = 3 * Xr2 * xi + 6 * Xr * Xi * xr - 3 * Xi2 * xi + 6 * Xr * xr * xi
* s + 3 * Xi * xr2 * s - 3 * Xi * xi2 * s + 3 * xr2 * xi * s * s - xi2 * xi * s
* s + ci;
      </scaled>
    </formula>
```

```
    <formula power="4" glitch="0.00001">
      <reference t="C">
        Xn = (X ^ 4) + C;
      </reference>
      <perturbation t="C">
        xn = 4 * (X ^ 3) * x + 6 * (X ^ 2) * (x ^ 2) + 4 * X * (x ^ 3) + (x ^
4) + c;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dya)))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dyb)))) + dab))));
        dyan = ((4 * (Xxr * (Xxr * (Xxr * dya)))) + (( - 4 * (Xxi * (Xxi * (Xxi
* dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dya)))) + ((12 * (Xxi * (Xxr * (Xxr
* dxa)))) + dba))));
        dybn = ((4 * (Xxr * (Xxr * (Xxr * dyb)))) + (( - 4 * (Xxi * (Xxi * (Xxi
* dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dyb)))) + ((12 * (Xxi * (Xxr * (Xxr
* dxb)))) + dbb))));
      </derivative>
    </formula>

    <formula power="5" glitch="0.0001">
      <reference t="C">
        Xn = (X ^ 5) + C;
      </reference>
      <perturbation t="C">
        xn = 5 * (X ^ 4) * x + 10 * (X ^ 3) * (x ^ 2) + 10 * (X ^ 2) * (x ^ 3)
+ 5 * X * (x ^ 4) + (x ^ 5) + c;
      </perturbation>
      <derivative t="M">
        dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dxa))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dxa))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dya))))) + ((20
* (Xxi * (Xxi * (Xxi * (Xxr * dya))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dxa))))) + daa)))));
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dxb))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dxb))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dyb))))) + ((20
* (Xxi * (Xxi * (Xxi * (Xxr * dyb))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dxb))))) + dab)))));
        dyan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dya))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dxa))))) + (( - 20
* (Xxi * (Xxi * (Xxi * (Xxr * dxa))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dya))))) + dba)))));
        dybn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dyb))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dxb))))) + (( - 20
* (Xxi * (Xxi * (Xxi * (Xxr * dxb))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dyb))))) + dbb)))));
      </derivative>
    </formula>

    <formula power="6" glitch="0.0001">
      <reference t="C">
        Xn = (X ^ 6) + C;
      </reference>
```

```
160       <perturbation t="C">
            xn = 6 * (X ^ 5) * x + 15 * (X ^ 4) * (x ^ 2) + 20 * (X ^ 3) * (x ^ 3)
    + 15 * (X ^ 2) * (x ^ 4) + 6 * X * (x ^ 5) + (x ^ 6) + c;
          </perturbation>
          <derivative t="C">
165         dn = 6 * d * (Xx ^ 5) + d0;
          </derivative>
        </formula>

        <formula power="7" glitch="0.001">
          <reference t="C">
170         Xn = (X ^ 7) + C;
          </reference>
          <perturbation t="C">
            xn = 7 * (X ^ 6) * x + 21 * (X ^ 5) * (x ^ 2) + 35 * (X ^ 4) * (x ^ 3)
    + 35 * (X ^ 3) * (x ^ 4) + 21 * (X ^ 2) * (x ^ 5) + 7 * X * (x ^ 6) + (x ^ 7) +
    c;
          </perturbation>
          <derivative t="C">
            dn = 7 * d * (Xx ^ 6) + d0;
180       </derivative>
        </formula>

        <formula power="8" glitch="0.001">
          <reference t="C">
185         Xn = (X ^ 8) + C;
          </reference>
          <perturbation t="C">
            xn = 8 * (X ^ 7) * x + 28 * (X ^ 6) * (x ^ 2) + 56 * (X ^ 5) * (x ^ 3)
    + 70 * (X ^ 4) * (x ^ 4) + 56 * (X ^ 3) * (x ^ 5) + 28 * (X ^ 2) * (x ^ 6) + 8
190 * X * (x ^ 7) + (x ^ 8) + c;
          </perturbation>
          <derivative t="C">
            dn = 8 * d * (Xx ^ 7) + d0;
          </derivative>
195     </formula>

        <formula power="9" glitch="0.01">
          <reference t="C">
            Xn = (X ^ 9) + C;
200       </reference>
          <perturbation t="C">
            xn = 9 * (X ^ 8) * x + 36 * (X ^ 7) * (x ^ 2) + 84 * (X ^ 6) * (x ^ 3)
    + 126 * (X ^ 5) * (x ^ 4) + 126 * (X ^ 4) * (x ^ 5) + 84 * (X ^ 3) * (x ^ 6) +
    36 * (X ^ 2) * (x ^ 7) + 9 * X * (x ^ 8) + (x ^ 9) + c;
205       </perturbation>
          <derivative t="C">
            dn = 9 * d * (Xx ^ 8) + d0;
          </derivative>
        </formula>
210
        <formula power="10" glitch="0.01">
          <reference t="C">
            Xn = (X ^ 10) + C;
          </reference>
215       <perturbation t="C">
            xn = 10 * (X ^ 9) * x + 45 * (X ^ 8) * (x ^ 2) + 120 * (X ^ 7) * (x ^
```

```
3) + 210 * (X ^ 6) * (x ^ 4) + 252 * (X ^ 5) * (x ^ 5) + 210 * (X ^ 4) * (x ^
6) + 120 * (X ^ 3) * (x ^ 7) + 45 * (X ^ 2) * (x ^ 8) + 10 * X * (x ^ 9) + (x ^
10) + c;
      </perturbation>
      <derivative t="C">
        dn = 10 * d * (Xx ^ 9) + d0;
      </derivative>
    </formula>

  </group>
  <group type="1" name="Burning Ship">

    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = Xr2 - Xi2 + Cr;
        Xin = abs(2 * Xr * Xi) + Ci;
      </reference>
      <perturbation t="R">
        xrn = (2 * Xr + xr) * xr - (2 * Xi + xi) * xi + cr;
        xin = 2 * diffabs(Xr * Xi, Xr * xi + xr * Xi + xr * xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * dxa)) + (( - 2 * (Xxi * dya)) + daa));
        dxbn = ((2 * (Xxr * dxb)) + (( - 2 * (Xxi * dyb)) + dab));
        dyan = ((2 * (sgn(Xxi) * (Xxi * (dxa * sgn(Xxr))))) + ((2 * (sgn(Xxi) *
(Xxr * (dya * sgn(Xxr))))) + dba));
        dybn = ((2 * (sgn(Xxi) * (Xxi * (dxb * sgn(Xxr))))) + ((2 * (sgn(Xxi) *
(Xxr * (dyb * sgn(Xxr))))) + dbb));
      </derivative>
<!--
neither of these work near on-axis miniships (Xi underflowed already)
more details here: <https://fractalforums.org/f/15/t/3041>
v1
      <scaled t="R">
        xrn = (2 * Xr + xr * s) * xr - (2 * Xi + xi * s) * xi + cr;
        xin = 2 * diffabs(Xr * S * Xi, Xr * xi + xr * (Xi + xi * s)) + ci;
      </scaled>
v2
      <scaled t="R">
        xrn = (2 * Xr + xr * s) * xr - (2 * Xi + xi * s) * xi + cr;
        xin = 2 * diffabs(abs(Xr) &lt; abs(Xi) ? (Xr * S) * Xi : Xr * (S * Xi),
Xr * xi + xr * Xi + (abs(xr) &gt; abs(xi) ? (xr * s) * xi : xr * (s * xi))) +
ci;
      </scaled>
-->
    </formula>

    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = (Xr2 - 3 * Xi2) * abs(Xr) + Cr;
        Xin = (3 * Xr2 - Xi2) * abs(Xi) + Ci;
      </reference>
      <perturbation t="R">
        xrn = (Xr2 - 3 * Xi2) * diffabs(Xr, xr) + (2 * Xr * xr + xr2 - 6 * Xi *
xi - 3 * xi2) * abs(Xr + xr) + cr;
        xin = (3 * Xr2 - Xi2) * diffabs(Xi, xi) + (6 * Xr * xr + 3 * xr2 - 2 *
Xi * xi - xi2) * abs(Xi + xi) + ci;
```

```
      </perturbation>
      <derivative t="M">
        dxan = ((3 * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (dxa *
sgn(Xxr)))))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (dxa *
sgn(Xxr)))))) + (( - 6 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxr * (dya *
sgn(Xxr)))))) + daa)));
        dxbn = ((3 * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (dxb *
sgn(Xxr)))))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (dxb *
sgn(Xxr)))))) + (( - 6 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxr * (dyb *
sgn(Xxr)))))) + dab)));
        dyan = ((3 * (sgn(Xxi) * (sgn(Xxr) * (Xxr * (Xxr * (dya *
sgn(Xxr)))))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (dya *
sgn(Xxi)))))) + ((6 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (dxa *
sgn(Xxr)))))) + dba)));
        dybn = ((3 * (sgn(Xxi) * (sgn(Xxr) * (Xxr * (Xxr * (dyb *
sgn(Xxr)))))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (dyb *
sgn(Xxi)))))) + ((6 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (dxb *
sgn(Xxr)))))) + dbb)));
      </derivative>
    </formula>

    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2 + Cr;
        Xin = 4 * abs(Xr * Xi) * (Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr * Xr * Xr * xr + 6 * Xr * Xr * xr * xr + 4 * Xr * xr * xr
* xr + xr * xr * xr * xr + 4 * Xi * Xi * Xi * xi + 6 * Xi * Xi * xi * xi + 4 *
Xi * xi * xi * xi + xi * xi * xi * xi - 12 * Xr * Xr * Xi * xi - 6 * Xr * Xr *
xi * xi - 12 * Xr * xr * Xi * Xi - 24 * Xr * xr * Xi * xi - 12 * Xr * xr * xi *
xi - 6 * xr * xr * Xi * Xi - 12 * xr * xr * Xi * xi - 6 * xr * xr * xi * xi +
cr;
        xin = 4 * (Xr * Xr - Xi * Xi) * diffabs(Xr * Xi, Xr * xi + xr * Xi + xr
* xi) + 4 * abs(Xr * Xi + Xr * xi + xr * Xi + xr * xi) * (2 * xr * Xr + xr * xr
- 2 * xi * Xi - xi * xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi *
(dya * sgn(Xxi))))))))) + ((4 * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr
* (Xxr * (dxa * sgn(Xxr))))))))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr)
* (Xxi * (Xxi * (Xxr * (dxa * sgn(Xxr)))))))) + (( - 12 * (sgn(Xxi) *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dya * sgn(Xxr))))))))) + daa))));
        dxbn = ((4 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi *
(dyb * sgn(Xxi))))))))) + ((4 * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr
* (Xxr * (dxb * sgn(Xxr))))))))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr)
* (Xxi * (Xxi * (Xxr * (dxb * sgn(Xxr)))))))) + (( - 12 * (sgn(Xxi) *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dyb * sgn(Xxr))))))))) + dab))));
        dyan = ((4 * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr *
(dya * sgn(Xxr))))))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (dxa * sgn(Xxr))))))))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dya * sgn(Xxr))))))))) + ((12 * (sgn(Xxi) *
(sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr))))))))) + dba))));
        dybn = ((4 * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr *
(dyb * sgn(Xxr))))))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (dxb * sgn(Xxr))))))))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dyb * sgn(Xxr))))))))) + ((12 * (sgn(Xxi) *
```

```
(sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + dbb))));
      </derivative>
    </formula>

    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr) * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2) + Cr;
        Xin = abs(Xi) * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr, xr) * (Xr * Xr * Xr * Xr - 10 * Xr * Xr * Xi * Xi + 5
* Xi * Xi * Xi * Xi) + abs(Xr + xr) * (4 * Xr * Xr * Xr * xr + 6 * Xr * Xr * xr
* xr + 4 * Xr * xr * xr * xr + xr * xr * xr * xr  - 20 * Xr * Xr * Xi * xi - 10
* Xr * Xr * xi * xi - 20 * Xr * xr * Xi * Xi - 40 * Xr * xr * Xi * xi - 20 * Xr
* xr * xi * xi - 10 * xr * xr * Xi * Xi - 20 * xr * xr * Xi * xi - 10 * xr * xr
* xi * xi + 20 * Xi * Xi * Xi * xi + 30 * Xi * Xi * xi * xi + 20 * Xi * xi * xi
* xi + 5 * xi * xi * xi * xi) + cr;
        xin = diffabs(Xi, xi) * (5 * Xr * Xr * Xr * Xr - 10 * Xr * Xr * Xi * Xi
+ Xi * Xi * Xi * Xi) + abs(Xi + xi) * (20 * Xr * Xr * Xr * xr + 30 * Xr * Xr *
xr * xr + 20 * Xr * xr * xr * xr + 5 * xr * xr * xr * xr - 20 * Xr * Xr * Xi *
xi - 10 * Xr * Xr * xi * xi - 20 * Xr * xr * Xi * Xi - 40 * Xr * xr * Xi * xi -
20 * Xr * xr * xi * xi - 10 * xr * xr * Xi * Xi - 20 * xr * xr * Xi * xi - 10 *
xr * xr * xi * xi  + 4 * Xi * Xi * Xi * xi + 6 * Xi * Xi * xi * xi + 4 * Xi *
xi * xi * xi + xi * xi * xi * xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((5 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))))))) + ((5 * (sgn(Xxr) * (sgn(Xxr) *
(sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))))) +
(( - 20 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr *
(Xxr * (dya * sgn(Xxr)))))))))) + ((20 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr * (dya * sgn(Xxr)))))))))) + (( - 30 *
(sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr *
(dxa * sgn(Xxr)))))))))) + daa)))));
        dxbn = ((5 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))))))) + ((5 * (sgn(Xxr) * (sgn(Xxr) *
(sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))))) +
(( - 20 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr *
(Xxr * (dyb * sgn(Xxr)))))))))) + ((20 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr * (dyb * sgn(Xxr)))))))))) + (( - 30 *
(sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr *
(dxb * sgn(Xxr)))))))))) + dab)))));
        dyan = ((5 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (Xxi * (dya * sgn(Xxi)))))))))) + ((5 * (sgn(Xxi) * (sgn(Xxr) *
(sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxr)))))))))) +
((20 * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr *
(Xxr * (dxa * sgn(Xxr)))))))))) + (( - 20 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi)
* (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxa * sgn(Xxr)))))))))) + (( - 30 *
(sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr *
(dya * sgn(Xxr)))))))))) + dba)))));
        dybn = ((5 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (Xxi * (dyb * sgn(Xxi)))))))))) + ((5 * (sgn(Xxi) * (sgn(Xxr) *
(sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxr)))))))))) +
((20 * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr *
(Xxr * (dxb * sgn(Xxr)))))))))) + (( - 20 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi)
* (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxb * sgn(Xxr)))))))))) + (( - 30 *
(sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr *
```

```
(dyb * sgn(Xxr))))))))))) + dbb)))));
      </derivative>
    </formula>

  </group>
  <group type="2" name="Buffalo">

    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = abs(Xr2 - Xi2) + Cr;
        Xin = Ci - 2 * abs(Xr * Xi);
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr * Xr - Xi * Xi, 2 * Xr * xr + xr * xr - 2 * Xi * xi -
xi * xi) + cr;
        xin = ci - 2 * diffabs(Xr * Xi, Xr * xi + xr * Xi + xr * xi);
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))
+ (( - 2 * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))) + daa));
        dxbn = ((2 * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))
+ (( - 2 * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))) + dab));
        dyan = ((2 * (sgn(Xxi) * (Xxi * (dxa * sgn(Xxr))))) + ((2 * (sgn(Xxi) *
(Xxr * (dya * sgn(Xxr))))) + dba));
        dybn = ((2 * (sgn(Xxi) * (Xxi * (dxb * sgn(Xxr))))) + ((2 * (sgn(Xxi) *
(Xxr * (dyb * sgn(Xxr))))) + dbb));
      </derivative>
    </formula>

    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = abs((Xr2 - Xi2 * 3) * Xr) + Cr;
        Xin = abs((Xr2 * 3 - Xi2) * Xi) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr * (Xr * Xr - 3 * Xi * Xi), xr * (3 * Xr * Xr + xr *
xr) + 3 * Xr * (xr  * xr - 2 * Xi * xi - xi * xi) - 3 * xr * (Xi * Xi + 2 * Xi
* xi + xi * xi)) + cr;
        xin = diffabs(Xi * (3 * Xr * Xr - Xi * Xi), 3 * Xi * (2 * Xr * xr + xr
* xr - xi * xi) + 3 * xi * (Xr * Xr + 2 * Xr * xr + xr * xr) - xi * (3 * Xi *
Xi + xi * xi)) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((3 * (Xxr * (Xxr * (dxa * sgn((( - 3 * (Xxi * (Xxi * Xxr))) +
(Xxr * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dxa * sgn((( - 3 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dya *
sgn((( - 3 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))))) + daa)));
        dxbn = ((3 * (Xxr * (Xxr * (dxb * sgn((( - 3 * (Xxi * (Xxi * Xxr))) +
(Xxr * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dxb * sgn((( - 3 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dyb *
sgn((( - 3 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))))) + dab)));
        dyan = ((3 * (Xxr * (Xxr * (dya * sgn((( - 1 * (Xxi * (Xxi * Xxi))) +
(3 * (Xxi * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dya * sgn((( - 1 *
(Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + ((6 * (Xxi * (Xxr *
(dxa * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr))))))))) +
dba)));
        dybn = ((3 * (Xxr * (Xxr * (dyb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) +
```

```
445 (3 * (Xxi * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dyb * sgn((( - 1 *
    (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + ((6 * (Xxi * (Xxr *
    (dxb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) +
    dbb)));
      </derivative>
450   </formula>

    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2) + Cr;
455     Xin = abs(4 * Xr * Xi * (Xr2 - Xi2)) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
    + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
460 xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2 * Xi * xi - 6 * Xr2 * xi2 - 12 *
    Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 * Xr * xr * xi2 - 6 * xr2 * Xi2 -
    12 * xr2 * Xi * xi - 6 * xr2 * xi2) + cr;
        xin = diffabs(4 * Xr2 * Xr * Xi - 4 * Xr * Xi2 * Xi,  - 4 * xr * Xi2 *
    Xi - 12 * xi * Xr * Xi2 - 12 * xr * xi * Xi2 + 12 * xr * Xr2 * Xi - 12 * xi2 *
465 Xr * Xi + 12 * xr2 * Xr * Xi - 12 * xr * xi2 * Xi + 4 * xr2 * xr * Xi + 4 * xi
    * Xr2 * Xr + 12 * xr * xi * Xr2 - 4 * xi2 * xi * Xr + 12 * xr2 * xi * Xr - 4 *
    xr * xi2 * xi + 4 * xr2 * xr * xi) + ci;
      </perturbation>
      <derivative t="M">
470     dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
    Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi *
    Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi
475 * Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi
    * Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
480 Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi *
    Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi
    * Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
485 Xxr)))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi
    * Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
    Xxr)))))))))) + dab))));
        dyan = ((4 * (Xxr * (Xxr * (Xxr * (dya * sgn((( - 4 * (Xxi * (Xxi *
    (Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr)))))))))) + (( - 4 * (Xxi *
490 (Xxi * (Xxi * (dxa * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi *
    (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dya * sgn((( - 4
    * (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr)))))))))) +
    ((12 * (Xxi * (Xxr * (Xxr * (dxa * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) +
    (4 * (Xxi * (Xxr * (Xxr * Xxr)))))))))) + dba))));
495     dybn = ((4 * (Xxr * (Xxr * (Xxr * (dyb * sgn((( - 4 * (Xxi * (Xxi *
    (Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr)))))))))) + (( - 4 * (Xxi *
    (Xxi * (Xxi * (dxb * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi *
    (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dyb * sgn((( - 4
    * (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr)))))))))) +
500 ((12 * (Xxi * (Xxr * (Xxr * (dxb * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) +
    (4 * (Xxi * (Xxr * (Xxr * Xxr)))))))))) + dbb))));
```

```
      </derivative>
    </formula>

    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)) + Cr;
        Xin = abs(Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2)) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(5 * Xr * Xi2 * Xi2 - 10 * Xr2 * Xr * Xi2 + Xr2 * Xr2 *
Xr, 20 * Xr * xi * Xi2 * Xi - 30 * Xr2 * xr * Xi2 + 30 * Xr * xi2 * Xi2 - 30 *
Xr * xr2 * Xi2 - 20 * Xr2 * Xr * xi * Xi - 60 * Xr2 * xr * xi * Xi + 20 * Xr *
xi2 * xi * Xi - 60 * Xr * xr2 * xi * Xi + 5 * Xr2 * Xr2 * xr - 10 * Xr2 * Xr *
xi2 + 10 * Xr2 * Xr * xr2 - 30 * Xr2 * xr * xi2 + 10 * Xr2 * xr2 * xr + 5 * Xr
* xi2 * xi2 - 30 * Xr * xr2 * xi2 + 5 * Xr * xr2 * xr2 + 5 * xr * Xi2 * Xi2 +
20 * xr * xi * Xi2 * Xi + 30 * xr * xi2 * Xi2 - 10 * xr2 * xr * Xi2 + 20 * xr *
xi2 * xi * Xi - 20 * xr2 * xr * xi * Xi + 5 * xr * xi2 * xi2 - 10 * xr2 * xr *
xi2 + xr2 * xr2 * xr) + cr;
        xin = diffabs(Xi2 * Xi2 * Xi - 10 * Xi2 * Xi * Xr2 + 5 * Xi * Xr2 *
Xr2, 5 * Xi2 * Xi2 * xi - 20 * Xi2 * Xi * xr * Xr + 10 * Xi2 * Xi * xi2 - 10 *
Xi2 * Xi * xr2 - 30 * Xi2 * xi * Xr2 - 60 * Xi2 * xr * xi * Xr + 10 * Xi2 * xi2
* xi - 30 * Xi2 * xr2 * xi + 20 * Xi * xr * Xr2 * Xr - 30 * Xi * xi2 * Xr2 + 30
* Xi * xr2 * Xr2 - 60 * Xi * xr * xi2 * Xr + 20 * Xi * xr2 * xr * Xr + 5 * Xi *
xi2 * xi2 - 30 * Xi * xr2 * xi2 + 5 * Xi * xr2 * xr2 + 5 * xi * Xr2 * Xr2 + 20
* xi * xr * Xr2 * Xr - 10 * xi2 * xi * Xr2 + 30 * xi * xr2 * Xr2 - 20 * xi2 *
xi * xr * Xr + 20 * xi * xr2 * xr * Xr + xi2 * xi2 * xi - 10 * xi2 * xi * xr2 +
5 * xi * xr2 * xr2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxa * sgn(((5 * (Xxi * (Xxi *
(Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr
* (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr *
(dxa * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi *
(Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 20
* (Xxi * (Xxr * (Xxr * (Xxr * (dya * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi *
Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr
* (Xxr * Xxr)))))))))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * (dya * sgn(((5 *
(Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr *
Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 30 * (Xxi * (Xxi
* (Xxr * (Xxr * (dxa * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10
* (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + daa)))));
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxb * sgn(((5 * (Xxi * (Xxi *
(Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr
* (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr *
(dxb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi *
(Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 20
* (Xxi * (Xxr * (Xxr * (Xxr * (dyb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi *
Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr
* (Xxr * Xxr)))))))))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * (dyb * sgn(((5 *
(Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr *
Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 30 * (Xxi * (Xxi
* (Xxr * (Xxr * (dxb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10
* (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + dab)))));
        dyan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi
* (Xxi * Xxi)))) + (( - 10 * (Xxi * (Xxi * (Xxi * (Xxr * Xxr))))) + (5 * (Xxi *
```

```
    (Xxr * (Xxr * (Xxr * Xxr)))))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr * (dya
560 * sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (Xxi * (Xxi * (Xxi *
    (Xxr * Xxr))))) + (5 * (Xxi * (Xxr * (Xxr * (Xxr * Xxr)))))))))))))) + ((20 *
    (Xxi * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) +
    (( - 10 * (Xxi * (Xxi * (Xxi * (Xxr * Xxr))))) + (5 * (Xxi * (Xxr * (Xxr * (Xxr
    * Xxr)))))))))))))) + (( - 20 * (Xxi * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi *
565 (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (Xxi * (Xxi * (Xxi * (Xxr * Xxr))))) +
    (5 * (Xxi * (Xxr * (Xxr * (Xxr * Xxr)))))))))))))) + (( - 30 * (Xxi * (Xxi *
    (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (Xxi
    * (Xxi * (Xxi * (Xxr * Xxr))))) + (5 * (Xxi * (Xxr * (Xxr * (Xxr *
    Xxr)))))))))))))) + dba)))));
570         dybn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi
    * (Xxi * Xxi)))) + (( - 10 * (Xxi * (Xxi * (Xxi * (Xxr * Xxr))))) + (5 * (Xxi *
    (Xxr * (Xxr * (Xxr * Xxr)))))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr * (dyb
    * sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (Xxi * (Xxi * (Xxi *
    (Xxr * Xxr))))) + (5 * (Xxi * (Xxr * (Xxr * (Xxr * Xxr)))))))))))))) + ((20 *
575 (Xxi * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) +
    (( - 10 * (Xxi * (Xxi * (Xxi * (Xxr * Xxr))))) + (5 * (Xxi * (Xxr * (Xxr * (Xxr
    * Xxr)))))))))))))) + (( - 20 * (Xxi * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi *
    (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (Xxi * (Xxi * (Xxi * (Xxr * Xxr))))) +
    (5 * (Xxi * (Xxr * (Xxr * (Xxr * Xxr)))))))))))))) + (( - 30 * (Xxi * (Xxi *
580 (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (Xxi
    * (Xxi * (Xxi * (Xxr * Xxr))))) + (5 * (Xxi * (Xxr * (Xxr * (Xxr *
    Xxr)))))))))))))) + dbb)))));
      </derivative>
    </formula>
585

  </group>
  <group type="3" name="Celtic">

    <formula power="2" glitch="0.0000001">
590   <reference t="R">
        Xrn = abs(Xr2 - Xi2) + Cr;
        Xin = Xr * Xi * 2 + Ci;
      </reference>
      <perturbation t="R">
595     xrn = diffabs(Xr * Xr - Xi * Xi, (2 * Xr + xr) * xr - (2 * Xi + xi) *
    xi) + cr;
        xin = 2 * Xr * xi + 2 * xr * (Xi + xi) + ci;
      </perturbation>
      <derivative t="M">
600     dxan = ((2 * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))
    + (( - 2 * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + daa));
        dxbn = ((2 * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))
    + (( - 2 * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + dab));
        dyan = ((2 * (Xxi * dxa)) + ((2 * (Xxr * dya)) + dba));
605     dybn = ((2 * (Xxi * dxb)) + ((2 * (Xxr * dyb)) + dbb));
      </derivative>
    </formula>

    <formula power="3" glitch="0.0000001">
610   <reference t="R">
        Xrn = abs((Xr2 - (Xi2 * 3)) * Xr) + Cr;
        Xin = ((Xr2 * 3) - Xi2) * Xi + Ci;
      </reference>
      <perturbation t="R">
615     xrn = diffabs(Xr * (Xr2 - 3 * Xi2), xr * (3 * Xr2 + xr2) + 3 * Xr *
```

```
(xr2 - 2 * Xi * xi - xi2) - 3 * xr * (Xi2 + 2 * Xi * xi + xi2)) + cr;
        xin = 3 * Xi * (2 * Xr * xr + xr2 - xi2) + 3 * xi * (Xr2 + 2 * Xr * xr
+ xr2) - xi * (xi2 + 3 * Xi2) + ci;
      </perturbation>
      <derivative t='M'>
        dxan = ((3 * (Xxr * (Xxr * (dxa * sgn((( - 3 * (Xxi * (Xxi * Xxr))) +
(Xxr * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dxa * sgn((( - 3 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dya *
sgn((( - 3 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + daa)));
        dxbn = ((3 * (Xxr * (Xxr * (dxb * sgn((( - 3 * (Xxi * (Xxi * Xxr))) +
(Xxr * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dxb * sgn((( - 3 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dyb *
sgn((( - 3 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + dab)));
        dyan = ((3 * (Xxr * (Xxr * dya))) + (( - 3 * (Xxi * (Xxi * dya))) + ((6
* (Xxi * (Xxr * dxa))) + dba)));
        dybn = ((3 * (Xxr * (Xxr * dyb))) + (( - 3 * (Xxi * (Xxi * dyb))) + ((6
* (Xxi * (Xxr * dxb))) + dbb)));
      </derivative>
    </formula>

    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = 4 * Xr * Xi * (Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2  - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12
* xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = 12 * Xr2 * Xi * xr + 12 * Xr * Xi * xr2 - 12 * Xr * Xi2 * xi - 12
* Xr * Xi * xi2 + 4 * Xr2 * Xr * xi + 12 * Xr2 * xi * xr + 12 * Xr * xi * xr2 -
4 * Xr * xi2 * xi + 4 * xr2 * xr * Xi - 4 * xr * Xi2 * Xi - 12 * xr * Xi2 * xi
- 12 * xr * Xi * xi2 + 4 * xr2 * xr * xi - 4 * xr * xi2 * xi + ci;
      </perturbation>
      <derivative t='M'>
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + dab))));
        dyan = ((4 * (Xxr * (Xxr * (Xxr * dya)))) + (( - 4 * (Xxi * (Xxi * (Xxi
* dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dya)))) + ((12 * (Xxi * (Xxr * (Xxr
```

```
* dxa)))) + dba))));
        dybn = ((4 * (Xxr * (Xxr * (Xxr * dyb)))) + (( - 4 * (Xxi * (Xxi * (Xxi
675 * dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dyb)))) + ((12 * (Xxi * (Xxr * (Xxr
* dxb)))) + dbb))));
      </derivative>
    </formula>

680   <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)) + Cr;
        Xin = Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2) + Ci;
      </reference>
685   <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 * Xr - 10 * Xr2 * Xr * Xi2 + 5 * Xr * Xi2 *
Xi2, 20 * Xr * xi * Xi2 * Xi - 30 * Xr2 * xr * Xi2 + 30 * Xr * xi2 * Xi2 - 30 *
Xr * xr2 * Xi2 - 20 * Xr2 * Xr * xi * Xi - 60 * Xr2 * xr * xi * Xi + 20 * Xr *
xi2 * xi * Xi - 60 * Xr * xr2 * xi * Xi + 5 * Xr2 * Xr2 * xr - 10 * Xr2 * Xr *
690 xi2 + 10 * Xr2 * Xr * xr2 - 30 * Xr2 * xr * xi2 + 10 * Xr2 * xr2 * xr + 5 * Xr
* xi2 * xi2 - 30 * Xr * xr2 * xi2 + 5 * Xr * xr2 * xr2 + 5 * xr * Xi2 * Xi2 +
20 * xr * xi * Xi2 * Xi + 30 * xr * xi2 * Xi2 - 10 * xr2 * xr * Xi2 + 20 * xr *
xi2 * xi * Xi - 20 * xr2 * xr * xi * Xi + 5 * xr * xi2 * xi2 - 10 * xr2 * xr *
xi2 + xr2 * xr2 * xr) + cr;
695     xin = 20 * Xi * Xr2 * Xr * xr + 30 * Xi * Xr2 * xr2 + 20 * Xi * Xr *
xr2 * xr + 5 * Xi * xr2 * xr2 - 30 * Xi2 * Xr2 * xi - 30 * Xi * Xr2 * xi2 - 20
* Xi2 * Xi * Xr * xr - 60 * Xi2 * Xr * xr * xi - 60 * Xi * Xr * xr * xi2 - 10 *
Xi2 * Xi * xr2 - 30 * Xi2 * xr2 * xi - 30 * Xi * xr2 * xi2 + 5 * Xi2 * Xi2 * xi
+ 10 * Xi2 * Xi * xi2 + 10 * Xi2 * xi2 * xi + 5 * Xi * xi2 * xi2 + 5 * xi * Xr2
700 * Xr2 + 20 * xi * Xr2 * Xr * xr + 30 * xi * Xr2 * xr2 + 20 * xi * Xr * xr2 * xr
+ 5 * xi * xr2 * xr2 - 10 * xi2 * xi * Xr2 - 20 * xi2 * xi * Xr * xr - 10 * xi2
* xi * xr2 + xi2 * xi2 * xi + ci;
      </perturbation>
      <derivative t="M">
705     dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxa * sgn(((5 * (Xxi * (Xxi *
(Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr
* (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr *
(dxa * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi *
(Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 20
710 * (Xxi * (Xxr * (Xxr * (Xxr * (dya * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi *
Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr
* (Xxr * Xxr)))))))))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * (dya * sgn(((5 *
(Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr *
Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 30 * (Xxi * (Xxi
715 * (Xxr * (Xxr * (dxa * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10
* (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + daa)))));
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxb * sgn(((5 * (Xxi * (Xxi *
(Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr
720 * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr *
(dxb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi *
(Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 20
* (Xxi * (Xxr * (Xxr * (Xxr * (dyb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi *
Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr
725 * (Xxr * Xxr)))))))))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * (dyb * sgn(((5 *
(Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr *
Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 30 * (Xxi * (Xxi
* (Xxr * (Xxr * (dxb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10
* (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr *
```

```
Xxr))))))))))))) + dab)))));
        dyan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dya))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dxa))))) + (( - 20
* (Xxi * (Xxi * (Xxi * (Xxr * dxa))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dya))))) + dba)))));
        dybn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dyb))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dxb))))) + (( - 20
* (Xxi * (Xxi * (Xxi * (Xxr * dxb))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dyb))))) + dbb)))));
      </derivative>
    </formula>

  </group>
  <group type="4" name="Mandelbar">

    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = Xr2 - Xi2 + Cr;
        Xin = Ci - Xr * Xi * 2;
      </reference>
      <perturbation t="R">
        xrn = 2 * Xr * xr + xr2 - xi2 - 2 * xi * Xi + cr;
        xin = ci - 2 * (Xr * xi + xr * Xi + xr * xi);
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxi * dya)) + ((2 * (Xxr * dxa)) + daa));
        dxbn = (( - 2 * (Xxi * dyb)) + ((2 * (Xxr * dxb)) + dab));
        dyan = (( - 2 * (Xxi * dxa)) + (( - 2 * (Xxr * dya)) + dba));
        dybn = (( - 2 * (Xxi * dxb)) + (( - 2 * (Xxr * dyb)) + dbb));
      </derivative>
    </formula>

    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = Cr - (Xr2 - Xi2 * 3) * Xr;
        Xin = (Xr2 * 3 - Xi2) * Xi + Ci;
      </reference>
      <perturbation t="R">
        xrn = cr - xr * (3 * Xr2 + xr2) + 3 * Xr * (xi2 + 2 * Xi * xi - xr2) +
3 * xr * (Xi2 + 2 * Xi * xi + xi2);
        xin = 6 * Xr * Xi * xr + 3 * Xi * xr2 - 3 * Xi2 * xi - 3 * Xi * xi2 + 3
* Xr2 * xi + 6 * Xr * xr * xi + 3 * xr2 * xi - xi2 * xi + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((3 * (Xxr * (Xxr * dxa))) + (( - 3 * (Xxi * (Xxi * dxa))) + ((
- 6 * (Xxi * (Xxr * dya))) + daa)));
        dxbn = ((3 * (Xxr * (Xxr * dxb))) + (( - 3 * (Xxi * (Xxi * dxb))) + ((
- 6 * (Xxi * (Xxr * dyb))) + dab)));
        dyan = (( - 3 * (Xxr * (Xxr * dya))) + ((3 * (Xxi * (Xxi * dya))) + ((
- 6 * (Xxi * (Xxr * dxa))) + dba)));
        dybn = (( - 3 * (Xxr * (Xxr * dyb))) + ((3 * (Xxi * (Xxi * dyb))) + ((
- 6 * (Xxi * (Xxr * dxb))) + dbb)));
      </derivative>
    </formula>

    <formula power="4" glitch="0.00001">
      <reference t="R">
```

```
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = Ci - 4 * Xr * Xi * (Xr2 - Xi2);
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
+ 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
* Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
        xin = ( - 12) * Xr2 * Xi * xr - 12 * Xr * Xi * xr2 + 12 * Xr * Xi2 * xi
+ 12 * Xr * Xi * xi2 - 4 * Xr2 * Xr * xi - 12 * Xr2 * xi * xr - 12 * Xr * xi *
xr2 + 4 * Xr * xi2 * xi - 4 * xr2 * xr * Xi + 4 * xr * Xi2 * Xi + 12 * xr * Xi2
* xi + 12 * xr * Xi * xi2 - 4 * xr2 * xr * xi + 4 * xr * xi2 * xi + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dya)))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dyb)))) + dab))));
        dyan = (( - 4 * (Xxr * (Xxr * (Xxr * dya)))) + ((4 * (Xxi * (Xxi * (Xxi
* dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr * dxa)))) + ((12 * (Xxi * (Xxi * (Xxr
* dya)))) + dba))));
        dybn = (( - 4 * (Xxr * (Xxr * (Xxr * dyb)))) + ((4 * (Xxi * (Xxi * (Xxi
* dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr * dxb)))) + ((12 * (Xxi * (Xxi * (Xxr
* dyb)))) + dbb))));
      </derivative>
    </formula>

    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = Xr * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)  + Cr;
        Xin = Ci - Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2);
      </reference>
      <perturbation t="R">
        xrn = 5 * Xr2 * Xr2 * xr + 10 * Xr2 * Xr * xr2 + 10 * Xr2 * xr2 * xr +
5 * Xr * xr2 * xr2 - 20 * Xr2 * Xr * Xi * xi - 10 * Xr2 * Xr * xi2 - 30 * Xr2 *
xr * Xi2 - 60 * Xr2 * xr * Xi * xi - 30 * Xr2 * xr * xi2 - 30 * Xr * xr2 * Xi2
- 60 * Xr * xr2 * Xi * xi - 30 * Xr * xr2 * xi2 + 20 * Xr * Xi2 * Xi * xi + 30
* Xr * Xi2 * xi2 + 20 * Xr * Xi * xi2 * xi + 5 * Xr * xi2 * xi2 + xr2 * xr2 *
xr - 10 * xr2 * xr * Xi2 - 20 * xr2 * xr * Xi * xi - 10 * xr2 * xr * xi2 + 5 *
xr * Xi2 * Xi2 + 20 * xr * Xi2 * Xi * xi + 30 * xr * Xi2 * xi2 + 20 * xr * Xi *
xi2 * xi + 5 * xr * xi2 * xi2 + cr;
        xin = ( - 20) * Xi * Xr2 * Xr * xr - 30 * Xi * Xr2 * xr2 - 20 * Xi * Xr
* xr2 * xr - 5 * Xi * xr2 * xr2 + 30 * Xi2 * Xr2 * xi + 30 * Xi * Xr2 * xi2 +
20 * Xi2 * Xi * Xr * xr + 60 * Xi2 * Xr * xr * xi + 60 * Xi * Xr * xr * xi2 +
10 * Xi2 * Xi * xr2 + 30 * Xi2 * xr2 * xi + 30 * Xi * xr2 * xi2 - 5 * Xi2 * Xi2
* xi - 10 * Xi2 * Xi * xi2 - 10 * Xi2 * xi2 * xi - 5 * Xi * xi2 * xi2 - 5 * xi
* Xr2 * Xr2 - 20 * xi * Xr2 * Xr * xr - 30 * xi * Xr2 * xr2 - 20 * xi * Xr *
xr2 * xr - 5 * xi * xr2 * xr2 + 10 * xi2 * xi * Xr2 + 20 * xi2 * xi * Xr * xr +
10 * xi2 * xi * xr2 - xi2 * xi2 * xi + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dxa))))) + ((5 * (Xxr * (Xxr *
(Xxr * (Xxr * dxa))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dya))))) + (( - 20
* (Xxi * (Xxr * (Xxr * (Xxr * dya))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
dxa))))) + daa)))));
```

```
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dxb))))) + ((5 * (Xxr * (Xxr *
845 (Xxr * (Xxr * dxb))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dyb))))) + (( - 20
    * (Xxi * (Xxr * (Xxr * (Xxr * dyb))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr *
    dxb))))) + dab)))));
        dyan = (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + (( - 5 * (Xxr *
    (Xxr * (Xxr * (Xxr * dya))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dxa))))) +
850 (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dxa))))) + ((30 * (Xxi * (Xxi * (Xxr *
    (Xxr * dya))))) + dba)))));
        dybn = (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + (( - 5 * (Xxr *
    (Xxr * (Xxr * (Xxr * dyb))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dxb))))) +
    (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dxb))))) + ((30 * (Xxi * (Xxi * (Xxr *
855 (Xxr * dyb))))) + dbb)))));
      </derivative>
    </formula>

  </group>
860 <group type="5" name="Mandelbar Celtic">
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = abs(Xr2 - Xi2) + Cr;
        Xin = Ci - Xr * Xi * 2;
865   </reference>
      <perturbation t="R">
        xrn = diffabs(Xr * Xr - Xi * Xi, 2 * Xr * xr + xr2 - 2 * Xi * xi - xi2)
    + cr;
        xin = (Xr * xi + xr * Xi + xr * xi) * ( - 2) + ci;
870   </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))
    + (( - 2 * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))) + daa));
        dxbn = ((2 * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))
875 + (( - 2 * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))) + dab));
        dyan = (( - 2 * (Xxi * dxa)) + (( - 2 * (Xxr * dya)) + dba));
        dybn = (( - 2 * (Xxi * dxb)) + (( - 2 * (Xxr * dyb)) + dbb));
      </derivative>
    </formula>
880 </group>

  <group type="6" name="Perpendicular Mandelbrot">
    <formula power="2" glitch="0.0000001">
      <reference t="R">
885     Xrn = Xr2 - Xi2 + Cr;
        Xin = Ci - abs(Xr) * Xi * 2;
      </reference>
      <perturbation t="R">
        xrn = 2 * Xr * xr + xr2 - xi2 - 2 * xi * Xi + cr;
890     xin = abs(Xr + xr) * xi * ( - 2) + diffabs(Xr, xr) * Xi * ( - 2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * dxa)) + (( - 2 * (Xxi * dya)) + daa));
        dxbn = ((2 * (Xxr * dxb)) + (( - 2 * (Xxi * dyb)) + dab));
895     dyan = (dba + (( - 2 * (Xxi * (dxa * sgn(Xxr)))) + ( - 2 * (abs(Xxr) *
    dya))));
        dybn = (dbb + (( - 2 * (Xxi * (dxb * sgn(Xxr)))) + ( - 2 * (abs(Xxr) *
    dyb))));
      </derivative>
900 </formula>
```

```
  </group>

  <group type="7" name="Perpendicular Burning Ship">
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = Xr2 - Xi2 + Cr;
        Xin = Ci - Xr * abs(Xi) * 2;
      </reference>
      <perturbation t="R">
        xrn = 2 * Xr * xr + xr2 - xi2 - 2 * xi * Xi + cr;
        xin = diffabs(Xi, xi) * Xr * ( - 2) + xr * abs((Xi + xi)) * ( - 2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * dxa)) + (( - 2 * (Xxi * dya)) + daa));
        dxbn = ((2 * (Xxr * dxb)) + (( - 2 * (Xxi * dyb)) + dab));
        dyan = (dba + (( - 2 * (Xxr * (dya * sgn(Xxi)))) + ( - 2 * (abs(Xxi) *
dxa))));
        dybn = (dbb + (( - 2 * (Xxr * (dyb * sgn(Xxi)))) + ( - 2 * (abs(Xxi) *
dxb))));
      </derivative>
    </formula>
  </group>

  <group type="8" name="Perpendicular Celtic">
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = abs(Xr2 - Xi2) + Cr;
        Xin = Ci - abs(Xr) * Xi * 2;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr * Xr - Xi * Xi, 2 * Xr * xr + xr2 - 2 * Xi * xi - xi2)
+ cr;
        xin = abs(Xr + xr)  * xi * ( - 2) + diffabs(Xr, xr) * Xi * ( - 2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))
+ (( - 2 * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + daa));
        dxbn = ((2 * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))))))
+ (( - 2 * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + dab));
        dyan = (dba + (( - 2 * (Xxi * (dxa * sgn(Xxr)))) + ( - 2 * (abs(Xxr) *
dya))));
        dybn = (dbb + (( - 2 * (Xxi * (dxb * sgn(Xxr)))) + ( - 2 * (abs(Xxr) *
dyb))));
      </derivative>
    </formula>
  </group>

  <group type="9" name="Perpendicular Buffalo">
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = abs(Xr2 - Xi2) + Cr;
        Xin = Ci - Xr * abs(Xi) * 2;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr * Xr - Xi * Xi, 2 * Xr * xr + xr2 - 2 * Xi * xi - xi2)
+ cr;
        xin = diffabs(Xi, xi) * Xr * ( - 2) + xr * abs((Xi + xi)) * ( - 2) + ci;
```

```
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + (( - 2 * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + daa));
        dxbn = ((2 * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + (( - 2 * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))) + dab));
        dyan = (dba + (( - 2 * (Xxr * (dya * sgn(Xxi)))) + ( - 2 * (abs(Xxi) * dxa))));
        dybn = (dbb + (( - 2 * (Xxr * (dyb * sgn(Xxi)))) + ( - 2 * (abs(Xxi) * dxb))));
      </derivative>
    </formula>
  </group>

  <group type="10" name="Cubic Quasi Burning Ship">
    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = (Xr2 - (Xi2 * 3)) * abs(Xr) + Cr;
        Xin = Ci - abs(((Xr2 * 3) - Xi2) * Xi);
      </reference>
      <perturbation t="R">
        xrn = (Xr2 - 3 * Xi2) * diffabs(Xr, xr) + (2 * xr * Xr + xr2 - 6 * Xi * xi - 3 * xi2) * abs(Xr + xr) + cr;
        xin = ci - diffabs(Xi * (3 * Xr2 - Xi2), 3 * Xi * (2 * Xr * xr + xr2 - xi2) + 3 * xi * (Xr2 + 2 * Xr * xr + xr2) - xi * (3 * Xi2 + xi2));
      </perturbation>
      <derivative t="M">
        dxan = ((Xxr * (Xxr * (dxa * sgn(Xxr)))) + ((2 * (abs(Xxr) * (Xxr * dxa))) + (( - 3 * (Xxi * (Xxi * (dxa * sgn(Xxr))))) + (( - 6 * (abs(Xxr) * (Xxi * dya))) + daa))));
        dxbn = ((Xxr * (Xxr * (dxb * sgn(Xxr)))) + ((2 * (abs(Xxr) * (Xxr * dxb))) + (( - 3 * (Xxi * (Xxi * (dxb * sgn(Xxr))))) + (( - 6 * (abs(Xxr) * (Xxi * dyb))) + dab))));
        dyan = (( - 3 * (Xxr * (Xxr * (dya * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + ((3 * (Xxi * (Xxi * (dya * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dxa * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + dba)));
        dybn = (( - 3 * (Xxr * (Xxr * (dyb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + ((3 * (Xxi * (Xxi * (dyb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dxb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + dbb)));
      </derivative>
    </formula>
  </group>

  <group type="11" name="Cubic Partial BS Real">
    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = (Xr2 - (Xi2 * 3)) * abs(Xr) + Cr;
        Xin = (Xr2 * 3 - Xi2) * Xi + Ci;
      </reference>
      <perturbation t="R">
        xrn = (Xr2 - 3 * Xi2) * diffabs(Xr, xr) + (2 * xr * Xr + xr2 - 6 * Xi * xi - 3 * xi2) * abs(Xr + xr) + cr;
        xin = 6 * Xr * (Xi * xr + xr * xi) + 3 * Xi * (xr2 - xi2) + 3 * xi *
```

```
(Xr2 - Xi2) + xi * (3 * xr2 - xi2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((Xxr * (Xxr * (dxa * sgn(Xxr)))) + ((2 * (abs(Xxr) * (Xxr *
dxa))) + (( - 3 * (Xxi * (Xxi * (dxa * sgn(Xxr))))) + (( - 6 * (abs(Xxr) * (Xxi
* dya))) + daa))));
        dxbn = ((Xxr * (Xxr * (dxb * sgn(Xxr)))) + ((2 * (abs(Xxr) * (Xxr *
dxb))) + (( - 3 * (Xxi * (Xxi * (dxb * sgn(Xxr))))) + (( - 6 * (abs(Xxr) * (Xxi
* dyb))) + dab))));
        dyan = ((3 * (Xxr * (Xxr * dya))) + (( - 3 * (Xxi * (Xxi * dya))) + ((6
* (Xxi * (Xxr * dxa))) + dba)));
        dybn = ((3 * (Xxr * (Xxr * dyb))) + (( - 3 * (Xxi * (Xxi * dyb))) + ((6
* (Xxi * (Xxr * dxb))) + dbb)));
      </derivative>
    </formula>
  </group>

  <group type="12" name="Cubic Partial BS Imag">
    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = (Xr2 - Xi2 * 3) * Xr + Cr;
        Xin = ((Xr2 * 3) - Xi2) * abs(Xi) + Ci;
      </reference>
      <perturbation t="R">
        xrn = 3 * Xr2 * xr + 3 * Xr * xr2 - 6 * Xr * Xi * xi - 3 * Xr * xi2 +
xr * xr2 - 3 * Xi2 * xr - 6 * Xi * xr * xi - 3 * xr * xi2 + cr;
        xin = (3 * Xr2 - Xi2) * diffabs(Xi, xi) + (6 * xr * Xr + 3 * xr2 - 2 *
Xi * xi - xi2) * abs(Xi + xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((3 * (Xxr * (Xxr * dxa))) + (( - 3 * (Xxi * (Xxi * dxa))) + ((
- 6 * (Xxi * (Xxr * dya))) + daa)));
        dxbn = ((3 * (Xxr * (Xxr * dxb))) + (( - 3 * (Xxi * (Xxi * dxb))) + ((
- 6 * (Xxi * (Xxr * dyb))) + dab)));
        dyan = (( - 1 * (Xxi * (Xxi * (dya * sgn(Xxi))))) + (( - 2 * (abs(Xxi)
* (Xxi * dya))) + ((3 * (Xxr * (Xxr * (dya * sgn(Xxi))))) + ((6 * (abs(Xxi) *
(Xxr * dxa))) + dba))));
        dybn = (( - 1 * (Xxi * (Xxi * (dyb * sgn(Xxi))))) + (( - 2 * (abs(Xxi)
* (Xxi * dyb))) + ((3 * (Xxr * (Xxr * (dyb * sgn(Xxi))))) + ((6 * (abs(Xxi) *
(Xxr * dxb))) + dbb))));
      </derivative>
    </formula>
  </group>

  <group type="13" name="Cubic Flying Squirrel (Buffalo Imag)">
    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = (Xr2 - Xi2 * 3) * Xr + Cr;
        Xin = abs(((Xr2 * 3) - Xi2) * Xi) + Ci;
      </reference>
      <perturbation t="R">
        xrn = 3 * Xr2 * xr + 3 * Xr * xr2 - 6 * Xr * Xi * xi - 3 * Xr * xi2 +
xr * xr2 - 3 * Xi2 * xr - 6 * Xi * xr * xi - 3 * xr * xi2 + cr;
        xin = diffabs(Xi * (3 * Xr2 - Xi2), 3 * Xi * (2 * Xr * xr + xr2 - xi2)
+ 3 * xi * (Xr2 + 2 * Xr * xr + xr2) - xi * (3 * Xi2 + xi2)) + ci;
      </perturbation>
      <derivative t="M">
```

```
        dxan = ((3 * (Xxr * (Xxr * dxa))) + (( - 3 * (Xxi * (Xxi * dxa))) + ((
- 6 * (Xxi * (Xxr * dya))) + daa)));
        dxbn = ((3 * (Xxr * (Xxr * dxb))) + (( - 3 * (Xxi * (Xxi * dxb))) + ((
- 6 * (Xxi * (Xxr * dyb))) + dab)));
        dyan = ((3 * (Xxr * (Xxr * (dya * sgn((( - 1 * (Xxi * (Xxi * Xxi))) +
(3 * (Xxi * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dya * sgn((( - 1 *
(Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + ((6 * (Xxi * (Xxr *
(dxa * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr))))))))) +
dba)));
        dybn = ((3 * (Xxr * (Xxr * (dyb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) +
(3 * (Xxi * (Xxr * Xxr)))))))) + (( - 3 * (Xxi * (Xxi * (dyb * sgn((( - 1 *
(Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr)))))))) + ((6 * (Xxi * (Xxr *
(dxb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (3 * (Xxi * (Xxr * Xxr))))))))) +
dbb)));
      </derivative>
    </formula>
  </group>

  <group type="14" name="Cubic Quasi Perpendicular">
    <formula power="3" glitch="0.0000001">
      <reference t="R">
        Xrn = (Xr2 - Xi2 * 3) * abs(Xr) + Cr;
        Xin = Ci - abs(Xr2 * 3 - Xi2) * Xi;
      </reference>
      <perturbation t="R">
        xrn = (Xr2 - 3 * Xi2) * diffabs(Xr, xr) + (2 * xr * Xr + xr2 - 6 * Xi *
xi - 3 * xi2) * abs(Xr + xr) + cr;
        xin = ci - diffabs(3 * Xr2 - Xi2, 6 * Xr * xr + 3 * xr2 - 2 * Xi * xi -
xi2) * Xi - abs(3 * Xr2 + 6 * Xr * xr + 3 * xr2 - Xi2 - 2 * Xi * xi - xi2) * xi;
      </perturbation>
      <derivative t="M">
        dxan = ((Xxr * (Xxr * (dxa * sgn(Xxr)))) + ((2 * (abs(Xxr) * (Xxr *
dxa))) + (( - 3 * (Xxi * (Xxi * (dxa * sgn(Xxr))))) + (( - 6 * (abs(Xxr) * (Xxi
* dya))) + daa))));
        dxbn = ((Xxr * (Xxr * (dxb * sgn(Xxr)))) + ((2 * (abs(Xxr) * (Xxr *
dxb))) + (( - 3 * (Xxi * (Xxi * (dxb * sgn(Xxr))))) + (( - 6 * (abs(Xxr) * (Xxi
* dyb))) + dab))));
        dyan = (dba + ((2 * (Xxi * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) + (3
* (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi))
+ (3 * (Xxr * Xxr)))))))) + ( - 1 * (abs((( - 1 * (Xxi * Xxi)) + (3 * (Xxr *
Xxr)))) * dya)))));
        dybn = (dbb + ((2 * (Xxi * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (3
* (Xxr * Xxr)))))))) + (( - 6 * (Xxi * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi))
+ (3 * (Xxr * Xxr)))))))) + ( - 1 * (abs((( - 1 * (Xxi * Xxi)) + (3 * (Xxr *
Xxr)))) * dyb)))));
      </derivative>
    </formula>
  </group>

  <group type="15" name="4th Burning Ship Partial Imag">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = 4 * Xr * abs(Xi) * (Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
```

```
    + 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
1130 * Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
    Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
          xin = abs(Xi + xi) * (12 * Xr2 * xr + 12 * Xr * xr2 + 4 * xr2 * xr - 4
    * xr * Xi2 - 8 * xi * Xr * Xi - 8 * xr * xi * Xi - 4 * xi2 * Xr - 4 * xr * xi2)
    + diffabs(Xi, xi) * (4 * Xr2 * Xr - 4 * Xr * Xi2) + ci;
1135      </perturbation>
          <derivative t="M">
            dxan = ((4 * (Xxr * (Xxr * (Xxr * dxa)))) + ((4 * (sgn(Xxi) * (sgn(Xxi)
    * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dya * sgn(Xxi)))))))))) + (( - 12 *
    (abs((Xxi * Xxi)) * (Xxr * dxa))) + (( - 12 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr *
1140 (dya * sgn(Xxi)))))))) + daa))));
            dxbn = ((4 * (Xxr * (Xxr * (Xxr * dxb)))) + ((4 * (sgn(Xxi) * (sgn(Xxi)
    * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dyb * sgn(Xxi)))))))))) + (( - 12 *
    (abs((Xxi * Xxi)) * (Xxr * dxb))) + (( - 12 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr *
    (dyb * sgn(Xxi)))))))) + dab))));
1145        dyan = (dba + ((4 * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxi)))))) + ((12 *
    (abs(Xxi) * (Xxr * (Xxr * dxa)))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
    (Xxi * (Xxr * (dya * sgn(Xxi)))))))) + ( - 4 * (abs((Xxi * (Xxi * Xxi))) *
    dxa))))));
            dybn = (dbb + ((4 * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxi)))))) + ((12 *
1150 (abs(Xxi) * (Xxr * (Xxr * dxb)))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
    (Xxi * (Xxr * (dyb * sgn(Xxi)))))))) + ( - 4 * (abs((Xxi * (Xxi * Xxi))) *
    dxb))))));
          </derivative>
        </formula>
1155  </group>

      <group type="16" name="4th Burning Ship Partial Real">
        <formula power="4" glitch="0.00001">
          <reference t="R">
1160        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
            Xin = 4 * abs(Xr) * Xi * (Xr2 - Xi2) + Ci;
          </reference>
          <perturbation t="R">
            xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
1165 + 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
    * Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
    Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
            xin = abs(Xr + xr) * (4 * Xr2 * xi + 8 * Xr * xr * Xi + 8 * Xr * xr *
    xi + 4 * xr2 * Xi + 4 * xr2 * xi - 12 * xi * Xi2 - 12 * xi2 * Xi - 4 * xi2 *
1170 xi) + diffabs(Xr, xr) * (4 * Xr2 * Xi - 4 * Xi2 * Xi) + ci;
          </perturbation>
          <derivative t="M">
            dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (sgn(Xxr) * (sgn(Xxr)
    * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))))) + (( - 12 *
1175 (abs((Xxr * Xxr)) * (Xxi * dya))) + (( - 12 * (sgn(Xxr) * (Xxi * (Xxi * (Xxr *
    (dxa * sgn(Xxr)))))))) + daa))));
            dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (sgn(Xxr) * (sgn(Xxr)
    * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))))) + (( - 12 *
    (abs((Xxr * Xxr)) * (Xxi * dyb))) + (( - 12 * (sgn(Xxr) * (Xxi * (Xxi * (Xxr *
1180 (dxb * sgn(Xxr)))))))) + dab))));
            dyan = (dba + (( - 4 * (Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))) + (( -
    12 * (abs(Xxr) * (Xxi * (Xxi * dya)))) + ((12 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
    (Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + (4 * (abs((Xxr * (Xxr * Xxr))) *
    dya))))));
1185        dybn = (dbb + (( - 4 * (Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))) + (( -
```

```
12 * (abs(Xxr) * (Xxi * (Xxi * dyb)))) + ((12 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + (4 * (abs((Xxr * (Xxr * Xxr))) *
dyb))))));
      </derivative>
    </formula>
  </group>

  <group type="17" name="4th Burning Ship Partial Real Mbar">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = Ci - 4 * abs(Xr) * Xi * (Xr2 - Xi2);
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
+ 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
* Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
        xin = abs(Xr + xr) * (12 * Xi2 * xi + 12 * Xi * xi2 + 4 * xi2 * xi - 8
* xr * Xr * Xi - 4 * xr2 * Xi - 4 * xi * Xr2 - 8 * xr * xi * Xr - 4 * xr2 * xi)
+ diffabs(Xr, xr) * (4 * Xi2 * Xi - 4 * Xr2 * Xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (sgn(Xxr) * (sgn(Xxr)
* (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + (( - 12 *
(abs((Xxr * Xxr)) * (Xxi * dya))) + (( - 12 * (sgn(Xxr) * (Xxi * (Xxi * (Xxr *
(dxa * sgn(Xxr))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (sgn(Xxr) * (sgn(Xxr)
* (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + (( - 12 *
(abs((Xxr * Xxr)) * (Xxi * dyb))) + (( - 12 * (sgn(Xxr) * (Xxi * (Xxi * (Xxr *
(dxb * sgn(Xxr))))))) + dab))));
        dyan = (dba + ((4 * (Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))) + ((12 *
(abs(Xxr) * (Xxi * (Xxi * dya)))) + (( - 12 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + ( - 4 * (abs((Xxr * (Xxr * Xxr))) *
dya))))));
        dybn = (dbb + ((4 * (Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))) + ((12 *
(abs(Xxr) * (Xxi * (Xxi * dyb)))) + (( - 12 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + ( - 4 * (abs((Xxr * (Xxr * Xxr))) *
dyb))))));
      </derivative>
    </formula>
  </group>

  <group type="18" name="4th Celtic Burning Ship Partial Imag">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = 4 * Xr * abs(Xi) * (Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12 *
xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = abs(Xi + xi) * (12 * Xr2 * xr + 12 * Xr * xr2 + 4 * xr2 * xr - 4
* xr * Xi2 - 8 * xi * Xr * Xi - 8 * xr * xi * Xi - 4 * xi2 * Xr - 4 * xr * xi2)
```

```
+ diffabs(Xi, xi) * (4 * Xr2 * Xr - 4 * Xr * Xi2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxr * (Xxr * (Xxr *
Xxr))) + (( - 6 * (abs((Xxi * Xxi)) * (Xxr * Xxr))) + abs((Xxi * (Xxi * (Xxi *
Xxi)))))))))))) + ((4 * (sgn(((Xxr * (Xxr * (Xxr * Xxr))) + (( - 6 * (abs((Xxi
* Xxi)) * (Xxr * Xxr))) + abs((Xxi * (Xxi * (Xxi * Xxi)))))))) * (sgn(Xxi) *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dya * sgn(Xxi)))))))))) + (( - 12
* (abs((Xxi * Xxi)) * (sgn(((Xxr * (Xxr * (Xxr * Xxr))) + (( - 6 * (abs((Xxi *
Xxi)) * (Xxr * Xxr))) + abs((Xxi * (Xxi * (Xxi * Xxi)))))))) * (Xxr * dxa)))) +
(( - 12 * (sgn(((Xxr * (Xxr * (Xxr * Xxr))) + (( - 6 * (abs((Xxi * Xxi)) * (Xxr
* Xxr))) + abs((Xxi * (Xxi * (Xxi * Xxi)))))))) * (sgn(Xxi) * (Xxi * (Xxr * (Xxr
* (dya * sgn(Xxi)))))))) + daa))));
        dxbn = ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxr * (Xxr * (Xxr *
Xxr))) + (( - 6 * (abs((Xxi * Xxi)) * (Xxr * Xxr))) + abs((Xxi * (Xxi * (Xxi *
Xxi)))))))))))) + ((4 * (sgn(((Xxr * (Xxr * (Xxr * Xxr))) + (( - 6 * (abs((Xxi
* Xxi)) * (Xxr * Xxr))) + abs((Xxi * (Xxi * (Xxi * Xxi)))))))) * (sgn(Xxi) *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dyb * sgn(Xxi)))))))))) + (( - 12
* (abs((Xxi * Xxi)) * (sgn(((Xxr * (Xxr * (Xxr * Xxr))) + (( - 6 * (abs((Xxi *
Xxi)) * (Xxr * Xxr))) + abs((Xxi * (Xxi * (Xxi * Xxi)))))))) * (Xxr * dxb)))) +
(( - 12 * (sgn(((Xxr * (Xxr * (Xxr * Xxr))) + (( - 6 * (abs((Xxi * Xxi)) * (Xxr
* Xxr))) + abs((Xxi * (Xxi * (Xxi * Xxi)))))))) * (sgn(Xxi) * (Xxi * (Xxr * (Xxr
* (dyb * sgn(Xxi)))))))) + dab))));
        dyan = (dba + ((4 * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxi)))))) + ((12 *
(abs(Xxi) * (Xxr * (Xxr * dxa)))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxr * (dya * sgn(Xxi)))))))) + ( - 4 * (abs((Xxi * (Xxi * Xxi))) *
dxa))))));
        dybn = (dbb + ((4 * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxi)))))) + ((12 *
(abs(Xxi) * (Xxr * (Xxr * dxb)))) + (( - 12 * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxr * (dyb * sgn(Xxi)))))))) + ( - 4 * (abs((Xxi * (Xxi * Xxi))) *
dxb))))));
      </derivative>
    </formula>
  </group>

  <group type="19" name="4th Celtic Burning Ship Partial Real">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = 4 * abs(Xr) * Xi * (Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12 *
xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = abs(Xr + xr) * (4 * Xr2 * xi + 8 * Xr * xr * Xi + 8 * Xr * xr *
xi + 4 * xr2 * Xi + 4 * xr2 * xi - 12 * xi * Xi2 - 12 * xi2 * Xi - 4 * xi2 *
xi) + diffabs(Xr, xr) * (4 * Xr2 * Xi - 4 * Xi2 * Xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi * Xxi))) + abs((Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + ((4 * (sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 6 * (abs((Xxr
* Xxr)) * (Xxi * Xxi))) + abs((Xxr * (Xxr * (Xxr * Xxr))))))) * (sgn(Xxr) *
(sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))))) + (( - 12
```

```
* (abs((Xxr * Xxr)) * (sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 6 * (abs((Xxr *
Xxr)) * (Xxi * Xxi))) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (Xxi * dya)))) +
(( - 12 * (sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi
* Xxi))) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr
* (dxa * sgn(Xxr)))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi * Xxi))) + abs((Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + ((4 * (sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 6 * (abs((Xxr
* Xxr)) * (Xxi * Xxi))) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (sgn(Xxr) *
(sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))))) + (( - 12
* (abs((Xxr * Xxr)) * (sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 6 * (abs((Xxr *
Xxr)) * (Xxi * Xxi))) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (Xxi * dyb)))) +
(( - 12 * (sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi
* Xxi))) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr
* (dxb * sgn(Xxr)))))))) + dab))));
        dyan = (dba + (( - 4 * (Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))) + (( -
12 * (abs(Xxr) * (Xxi * (Xxi * dya)))) + ((12 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + (4 * (abs((Xxr * (Xxr * Xxr))) *
dya))))));
        dybn = (dbb + (( - 4 * (Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))) + (( -
12 * (abs(Xxr) * (Xxi * (Xxi * dyb)))) + ((12 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + (4 * (abs((Xxr * (Xxr * Xxr))) *
dyb))))));
      </derivative>
    </formula>
  </group>

  <group type="20" name="4th Celtic Burning Ship Partial Real Mbar">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = Ci - 4 * abs(Xr) * Xi * (Xr2 - Xi2);
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12 *
xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = abs(Xr + xr) * (12 * Xi2 * xi + 12 * Xi * xi2 + 4 * xi2 * xi - 8
* xr * Xr * Xi - 4 * xr2 * Xi - 4 * xi * Xr2 - 8 * xr * xi * Xr - 4 * xr2 * xi)
+ diffabs(Xr, xr) * (4 * Xi2 * Xi - 4 * Xr2 * Xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (sgn((abs((Xxi * (Xxi * (Xxi * Xxi)))) + ((abs((Xxi * (Xxi
* (Xxr * Xxr)))) * ( - 6)) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (sgn(Xxi) *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dya * sgn(Xxi)))))))))) + ((4 *
(sgn((abs((Xxi * (Xxi * (Xxi * Xxi)))) + ((abs((Xxi * (Xxi * (Xxr * Xxr)))) * (
- 6)) + abs((Xxr * (Xxr * (Xxr * Xxr)))))))) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr)
* (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))))) + (( - 12 * (sgn((abs((Xxi *
(Xxi * (Xxi * Xxi)))) + ((abs((Xxi * (Xxi * (Xxr * Xxr)))) * ( - 6)) + abs((Xxr
* (Xxr * (Xxr * Xxr)))))))) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi *
(Xxr * (dxa * sgn(Xxr)))))))))) + (( - 12 * (sgn((abs((Xxi * (Xxi * (Xxi *
Xxi)))) + ((abs((Xxi * (Xxi * (Xxr * Xxr)))) * ( - 6)) + abs((Xxr * (Xxr * (Xxr
* Xxr)))))))) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dya *
sgn(Xxr)))))))))) + daa))));
        dxbn = ((4 * (sgn((abs((Xxi * (Xxi * (Xxi * Xxi)))) + ((abs((Xxi * (Xxi
```

```
* (Xxr * Xxr)))) * ( - 6)) + abs((Xxr * (Xxr * (Xxr * Xxr))))))) * (sgn(Xxi) *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dyb * sgn(Xxi))))))))) + ((4 *
(sgn((abs((Xxi * (Xxi * (Xxi * Xxi)))) + ((abs((Xxi * (Xxi * (Xxr * Xxr)))) * (
- 6)) + abs((Xxr * (Xxr * (Xxr * Xxr))))))) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr)
* (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr))))))))) + (( - 12 * (sgn((abs((Xxi *
(Xxi * (Xxi * Xxi)))) + ((abs((Xxi * (Xxi * (Xxr * Xxr)))) * ( - 6)) + abs((Xxr
* (Xxr * (Xxr * Xxr))))))) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi *
(Xxr * (dxb * sgn(Xxr))))))))) + (( - 12 * (sgn((abs((Xxi * (Xxi * (Xxi *
Xxi)))) + ((abs((Xxi * (Xxi * (Xxr * Xxr)))) * ( - 6)) + abs((Xxr * (Xxr * (Xxr
* Xxr))))))) * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dyb *
sgn(Xxr))))))))) + dab))));
        dyan = (( - 4 * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr
* (dya * sgn(Xxr))))))))) + ((4 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (dxa * sgn(Xxr))))))))) + ((12 * (sgn(Xxi) * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dya * sgn(Xxr))))))))) + (( - 12 * (sgn(Xxi)
* (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr))))))))) +
dba))));
        dybn = (( - 4 * (sgn(Xxi) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr
* (dyb * sgn(Xxr))))))))) + ((4 * (sgn(Xxi) * (sgn(Xxi) * (sgn(Xxi) * (Xxi *
(Xxi * (Xxi * (dxb * sgn(Xxr))))))))) + ((12 * (sgn(Xxi) * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dyb * sgn(Xxr))))))))) + (( - 12 * (sgn(Xxi)
* (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr))))))))) +
dbb))));
      </derivative>
    </formula>
  </group>

  <group type="21" name="4th Buffalo Partial Imag">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = abs(4 * Xr * Xi * (Xr2 - Xi2)) + Ci;
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
+ 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
* Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
        xin = diffabs(4 * Xr2 * Xr * Xi - 4 * Xr * Xi2 * Xi,  - 4 * xr * Xi2 *
Xi - 12 * xi * Xr * Xi2 - 12 * xr * xi * Xi2 + 12 * xr * Xr2 * Xi - 12 * xi2 *
Xr * Xi + 12 * xr2 * Xr * Xi - 12 * xr * xi2 * Xi + 4 * xr2 * xr * Xi + 4 * xi
* Xr2 * Xr + 12 * xr * xi * Xr2 - 4 * xi2 * xi * Xr + 12 * xr2 * xi * Xr - 4 *
xr * xi2 * xi + 4 * xr2 * xr * xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dya)))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dyb)))) + dab))));
        dyan = ((4 * (Xxr * (Xxr * (Xxr * (dya * sgn((( - 4 * (Xxi * (Xxi *
(Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr))))))))))) + (( - 4 * (Xxi *
(Xxi * (Xxi * (dxa * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi *
(Xxr * (Xxr * Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dya * sgn((( - 4
* (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr))))))))))) +
((12 * (Xxi * (Xxr * (Xxr * (dxa * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) +
```

```
(4 * (Xxi * (Xxr * (Xxr * Xxr))))))))))) + dba))));
        dybn = ((4 * (Xxr * (Xxr * (Xxr * (dyb * sgn((( - 4 * (Xxi * (Xxi *
(Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr))))))))))) + (( - 4 * (Xxi *
(Xxi * (Xxi * (dxb * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi *
(Xxr * (Xxr * Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dyb * sgn((( - 4
* (Xxi * (Xxi * (Xxi * Xxr)))) + (4 * (Xxi * (Xxr * (Xxr * Xxr))))))))))) +
((12 * (Xxi * (Xxr * (Xxr * (dxb * sgn((( - 4 * (Xxi * (Xxi * (Xxi * Xxr)))) +
(4 * (Xxi * (Xxr * (Xxr * Xxr))))))))))) + dbb))));
      </derivative>
    </formula>
  </group>

  <group type="22" name="4th Celtic Mbar">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = Ci - 4 * Xr * Xi * (Xr2 - Xi2);
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12 *
xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = ci - (12 * Xr2 * Xi * xr + 12 * Xr * Xi * xr2 - 12 * Xr * Xi2 *
xi - 12 * Xr * Xi * xi2 + 4 * Xr2 * Xr * xi + 12 * Xr2 * xi * xr + 12 * Xr * xi
* xr2 - 4 * Xr * xi2 * xi + 4 * xr2 * xr * Xi - 4 * xr * Xi2 * Xi - 12 * xr *
Xi2 * xi - 12 * xr * Xi * xi2 + 4 * xr2 * xr * xi - 4 * xr * xi2 * xi);
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + dab))));
        dyan = (( - 4 * (Xxr * (Xxr * (Xxr * dya)))) + ((4 * (Xxi * (Xxi * (Xxi
* dxa)))) + ((12 * (Xxi * (Xxi * (Xxr * dya)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dxa)))) + dba))));
        dybn = (( - 4 * (Xxr * (Xxr * (Xxr * dyb)))) + ((4 * (Xxi * (Xxi * (Xxi
* dxb)))) + ((12 * (Xxi * (Xxi * (Xxr * dyb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dxb)))) + dbb))));
      </derivative>
    </formula>
  </group>
```

```
  <group type="23" name="4th False Quasi Perpendicular">
    <formula power="4" glitch="0.00001">
      <reference t="R">
1475    Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = Ci - 4 * Xr * Xi * abs(Xr2 - Xi2);
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
1480 + 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
* Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
        xin = ( - (4 * Xr * Xi)) * diffabs(Xr2 - Xi2,  - 2 * xi * Xi + 2 * xr *
Xr - xi2 + xr2) - (4 * Xr * xi + 4 * xr * Xi + 4 * xr * xi) * abs( - Xi2 - 2 *
1485 xi * Xi + Xr2 + 2 * xr * Xr - xi2 + xr2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxa)))) + (( - 12 * (Xxr * (Xxr * dya))) + (( - 24 * (Xxi * (Xxr * dxa))) +
1490 daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxr * (Xxr * dyb))) + (( - 24 * (Xxi * (Xxr * dxb))) +
dab))));
        dyan = (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi *
1495 dxa))) + (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dya))) +
(( - 8 * (Xxi * (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr *
Xxr))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dya * sgn((( - 1 * (Xxi * Xxi)) +
(Xxr * Xxr)))))))) + dba))));
        dybn = (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi *
1500 dxb))) + (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dyb))) +
(( - 8 * (Xxi * (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr *
Xxr))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dyb * sgn((( - 1 * (Xxi * Xxi)) +
(Xxr * Xxr)))))))) + dbb))));
      </derivative>
1505  </formula>
  </group>

  <group type="24" name="4th False Quasi Heart">
    <formula power="4" glitch="0.00001">
1510    <reference t="R">
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = 4 * Xr * Xi * abs(Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
1515    xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
+ 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
* Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2 + cr;
        xin = (4 * Xr * Xi) * diffabs(Xr2 - Xi2,  - 2 * xi * Xi + 2 * xr * Xr -
1520 xi2 + xr2) + (4 * Xr * xi + 4 * xr * Xi + 4 * xr * xi) * abs( - Xi2 - 2 * xi *
Xi + Xr2 + 2 * xr * Xr - xi2 + xr2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
1525 dxa)))) + (( - 12 * (Xxr * (Xxr * dya))) + (( - 24 * (Xxi * (Xxr * dxa))) +
daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
```

```
dxb)))) + (( - 12 * (Xxr * (Xxr * dyb))) + (( - 24 * (Xxi * (Xxr * dxb))) +
dab))));
        dyan = ((4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi * dxa)))
+ ((4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dya))) + ((8 * (Xxi
* (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) + (( - 8
* (Xxi * (Xxi * (Xxr * (dya * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) +
dba))));
        dybn = ((4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi * dxb)))
+ ((4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dyb))) + ((8 * (Xxi
* (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) + (( - 8
* (Xxi * (Xxi * (Xxr * (dyb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) +
dbb))));
      </derivative>
    </formula>
  </group>

  <group type="25" name="4th Celtic False Quasi Perpendicular">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = ( - 4) * Xr * Xi * abs(Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12 *
xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = ( - (4 * Xr * Xi)) * diffabs(Xr2 - Xi2,  - 2 * xi * Xi + 2 * xr *
Xr - xi2 + xr2) - (4 * Xr * xi + 4 * xr * Xi + 4 * xr * xi) * abs( - Xi2 - 2 *
xi * Xi + Xr2 + 2 * xr * Xr - xi2 + xr2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) +
((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 *
(Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxr *
(Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr)))
+ (Xxr * (Xxr * (Xxr * Xxr))))))))) + (( - 24 * (Xxi * (Xxr * (dxa * sgn(((Xxi
* (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) +
((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 *
(Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxr *
(Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr)))
+ (Xxr * (Xxr * (Xxr * Xxr))))))))) + (( - 24 * (Xxi * (Xxr * (dxb * sgn(((Xxi
* (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))) + dab))));
        dyan = (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi *
dxa))) + (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dya))) +
(( - 8 * (Xxi * (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr *
Xxr)))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dya * sgn((( - 1 * (Xxi * Xxi)) +
(Xxr * Xxr)))))))) + dba))));
        dybn = (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi *
dxb))) + (( - 4 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dyb))) +
(( - 8 * (Xxi * (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr *
```

```
Xxr))))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dyb * sgn((( - 1 * (Xxi * Xxi)) +
(Xxr * Xxr))))))))) + dbb))));
      </derivative>
    </formula>
  </group>

  <group type="26" name="4th Celtic False Quasi Heart">
    <formula power="4" glitch="0.00001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = 4 * Xr * Xi * abs(Xr2 - Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2, 4 * Xr2 * Xr * xr
+ 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2 + 4 * Xi2 * Xi * xi + 6 * Xi2 *
xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * xr * Xr * Xi2 - 6 * xr2 * Xi2 - 12 *
xi * Xr2 * Xi - 24 * xr * xi * Xr * Xi - 12 * xr2 * xi * Xi - 6 * xi2 * Xr2 -
12 * xr * xi2 * Xr - 6 * xr2 * xi2) + cr;
        xin = (4 * Xr * Xi) * diffabs(Xr2 - Xi2,  - 2 * xi * Xi + 2 * xr * Xr -
xi2 + xr2) + (4 * Xr * xi + 4 * xr * Xi + 4 * xr * xi) * abs( - Xi2 - 2 * xi *
Xi + Xr2 + 2 * xr * Xr - xi2 + xr2) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) +
((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 *
(Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxr *
(Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr)))
+ (Xxr * (Xxr * (Xxr * Xxr))))))))) + (( - 24 * (Xxi * (Xxr * (dxa * sgn(((Xxi
* (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) +
((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 *
(Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxr *
(Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr)))
+ (Xxr * (Xxr * (Xxr * Xxr))))))))) + (( - 24 * (Xxi * (Xxr * (dxb * sgn(((Xxi
* (Xxi * (Xxi * Xxi))) + (( - 12 * (Xxi * (Xxr * Xxr))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + dab))));
        dyan = ((4 * (abs(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi * dxa)))
+ ((4 * (abs(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dya))) + ((8 * (Xxi
* (Xxr * (Xxr * (dxa * sgn(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) + (( - 8
* (Xxi * (Xxi * (Xxr * (dya * sgn(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) +
dba))));
        dybn = ((4 * (abs(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi * dxb)))
+ ((4 * (abs(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dyb))) + ((8 * (Xxi
* (Xxr * (Xxr * (dxb * sgn(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) + (( - 8
* (Xxi * (Xxi * (Xxr * (dyb * sgn(( - 1 * (Xxi * Xxi)) + (Xxr * Xxr)))))))) +
dbb))));
      </derivative>
    </formula>
  </group>

  <group type="27" name="5th Burning Ship Partial">
    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr) * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)  + Cr;
```

```
        Xin = Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2)  + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr, xr) * (Xr2 * Xr2 -  10 * Xr2 * Xi2 + 5 * Xi2 * Xi2) +
abs(Xr + xr) * (4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 *
xr2  - 20 * Xr2 * Xi * xi - 10 * Xr2 * xi2 - 20 * Xr * xr * Xi2 - 40 * Xr * xr
* Xi * xi - 20 * Xr * xr * xi2 - 10 * xr2 * Xi2 - 20 * xr2 * Xi * xi - 10 * xr2
* xi2 + 20 * Xi2 * Xi * xi + 30 * Xi2 * xi2 + 20 * Xi * xi2 * xi + 5 * xi2 *
xi2) + cr;
        xin = 20 * Xi * Xr2 * Xr * xr + 30 * Xi * Xr2 * xr2 + 20 * Xi * Xr *
xr2 * xr + 5 * Xi * xr2 * xr2 - 30 * Xi2 * Xr2 * xi - 30 * Xi * Xr2 * xi2 - 20
* Xi2 * Xi * Xr * xr - 60 * Xi2 * Xr * xr * xi - 60 * Xi * Xr * xr * xi2 - 10 *
Xi2 * Xi * xr2 - 30 * Xi2 * xr2 * xi - 30 * Xi * xr2 * xi2 + 5 * Xi2 * Xi2 * xi
+ 10 * Xi2 * Xi * xi2 + 10 * Xi2 * xi2 * xi + 5 * Xi * xi2 * xi2 + 5 * xi * Xr2
* Xr2 + 20 * xi * Xr2 * Xr * xr + 30 * xi * Xr2 * xr2 + 20 * xi * Xr * xr2 * xr
+ 5 * xi * xr2 * xr2 - 10 * xi2 * xi * Xr2 - 20 * xi2 * xi * Xr * xr - 10 * xi2
* xi * xr2 + xi2 * xi2 * xi + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))) + ((5 *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr *
(dxa * sgn(Xxr))))))))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi * (Xxi * dya))))) +
(( - 20 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * dya))) + (( - 30 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + daa)))));
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))) + ((5 *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr *
(dxb * sgn(Xxr))))))))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi * (Xxi * dyb))))) +
(( - 20 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * dyb))) + (( - 30 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + dab)))));
        dyan = (dba + ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + (( - 20 *
(sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxa * sgn(Xxr))))))) + ((20 *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (dxa *
sgn(Xxr))))))))) + (( - 30 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * dya)))) + (5 *
(abs((Xxr * (Xxr * (Xxr * Xxr)))) * dya)))))));
        dybn = (dbb + ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + (( - 20 *
(sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxb * sgn(Xxr))))))) + ((20 *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (dxb *
sgn(Xxr))))))))) + (( - 30 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * dyb)))) + (5 *
(abs((Xxr * (Xxr * (Xxr * Xxr)))) * dyb)))))));
      </derivative>
    </formula>
  </group>

  <group type="28" name="5th Burning Ship Partial Mbar">
    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr) * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)  + Cr;
        Xin = Ci - Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2);
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr, xr) * (Xr2 * Xr2 -  10 * Xr2 * Xi2 + 5 * Xi2 * Xi2) +
abs(Xr + xr) * (4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 *
xr2  - 20 * Xr2 * Xi * xi - 10 * Xr2 * xi2 - 20 * Xr * xr * Xi2 - 40 * Xr * xr
* Xi * xi - 20 * Xr * xr * xi2 - 10 * xr2 * Xi2 - 20 * xr2 * Xi * xi - 10 * xr2
* xi2 + 20 * Xi2 * Xi * xi + 30 * Xi2 * xi2 + 20 * Xi * xi2 * xi + 5 * xi2 *
xi2) + cr;
        xin = ci - (20 * Xi * Xr2 * Xr * xr + 30 * Xi * Xr2 * xr2 + 20 * Xi *
```

```
      Xr * xr2 * xr + 5 * Xi * xr2 * xr2 - 30 * Xi2 * Xr2 * xi - 30 * Xi * Xr2 * xi2
1700  - 20 * Xi2 * Xi * Xr * xr - 60 * Xi2 * Xr * xr * xi - 60 * Xi * Xr * xr * xi2 -
      10 * Xi2 * Xi * xr2 - 30 * Xi2 * xr2 * xi - 30 * Xi * xr2 * xi2 + 5 * Xi2 * Xi2
      * xi + 10 * Xi2 * Xi * xi2 + 10 * Xi2 * xi2 * xi + 5 * Xi * xi2 * xi2 + 5 * xi
      * Xr2 * Xr2 + 20 * xi * Xr2 * Xr * xr + 30 * xi * Xr2 * xr2 + 20 * xi * Xr *
      xr2 * xr + 5 * xi * xr2 * xr2 - 10 * xi2 * xi * Xr2 - 20 * xi2 * xi * Xr * xr -
1705  10 * xi2 * xi * xr2 + xi2 * xi2 * xi);
        </perturbation>
        <derivative t="M">
          dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))) + ((5 *
      (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr *
1710  (dxa * sgn(Xxr))))))))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi * (Xxi * dya))))) +
      (( - 20 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * dya))) + (( - 30 * (sgn(Xxr) *
      (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + daa)))));
          dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))) + ((5 *
      (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr *
1715  (dxb * sgn(Xxr))))))))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi * (Xxi * dyb))))) +
      (( - 20 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * dyb))) + (( - 30 * (sgn(Xxr) *
      (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + dab)))));
          dyan = (dba + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + ((20 *
      (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxa * sgn(Xxr))))))) + (( - 20 *
1720  (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (dxa *
      sgn(Xxr))))))))) + ((30 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * dya)))) + ( - 5 *
      (abs((Xxr * (Xxr * (Xxr * Xxr)))) * dya)))))));
          dybn = (dbb + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + ((20 *
      (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxb * sgn(Xxr))))))) + (( - 20 *
1725  (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (dxb *
      sgn(Xxr))))))))) + ((30 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * dyb)))) + ( - 5 *
      (abs((Xxr * (Xxr * (Xxr * Xxr)))) * dyb)))))));
        </derivative>
      </formula>
1730  </group>

      <group type="29" name="5th Celtic Mbar">
        <formula power="5" glitch="0.0001">
          <reference t="R">
1735        Xrn = abs(Xr * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)) + Cr;
            Xin = Ci - Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2);
          </reference>
          <perturbation t="R">
            xrn = diffabs(5 * Xr * Xi2 * Xi2 - 10 * Xr2 * Xr * Xi2 + Xr2 * Xr2 *
1740  Xr, 20 * Xr * xi * Xi2 * Xi - 30 * Xr2 * xr * Xi2 + 30 * Xr * xi2 * Xi2 - 30 *
      Xr * xr2 * Xi2 - 20 * Xr2 * Xr * xi * Xi - 60 * Xr2 * xr * xi * Xi + 20 * Xr *
      xi2 * xi * Xi - 60 * Xr * xr2 * xi * Xi + 5 * Xr2 * Xr2 * xr - 10 * Xr2 * Xr *
      xi2 + 10 * Xr2 * Xr * xr2 - 30 * Xr2 * xr * xi2 + 10 * Xr2 * xr2 * xr + 5 * Xr
      * xi2 * xi2 - 30 * Xr * xr2 * xi2 + 5 * Xr * xr2 * xr2 + 5 * xr * Xi2 * Xi2 +
1745  20 * xr * xi * Xi2 * Xi + 30 * xr * xi2 * Xi2 - 10 * xr2 * xr * Xi2 + 20 * xr *
      xi2 * xi * Xi - 20 * xr2 * xr * xi * Xi + 5 * xr * xi2 * xi2 - 10 * xr2 * xr *
      xi2 + xr2 * xr2 * xr) + cr;
            xin = ci - (20 * Xi * Xr2 * Xr * xr + 30 * Xi * Xr2 * xr2 + 20 * Xi *
      Xr * xr2 * xr + 5 * Xi * xr2 * xr2 - 30 * Xi2 * Xr2 * xi - 30 * Xi * Xr2 * xi2
1750  - 20 * Xi2 * Xi * Xr * xr - 60 * Xi2 * Xr * xr * xi - 60 * Xi * Xr * xr * xi2 -
      10 * Xi2 * Xi * xr2 - 30 * Xi2 * xr2 * xi - 30 * Xi * xr2 * xi2 + 5 * Xi2 * Xi2
      * xi + 10 * Xi2 * Xi * xi2 + 10 * Xi2 * xi2 * xi + 5 * Xi * xi2 * xi2 + 5 * xi
      * Xr2 * Xr2 + 20 * xi * Xr2 * Xr * xr + 30 * xi * Xr2 * xr2 + 20 * xi * Xr *
      xr2 * xr + 5 * xi * xr2 * xr2 - 10 * xi2 * xi * Xr2 - 20 * xi2 * xi * Xr * xr -
1755  10 * xi2 * xi * xr2 + xi2 * xi2 * xi);
```

```
      </perturbation>
      <derivative t='M'>
        dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxa * sgn(((5 * (Xxi * (Xxi *
(Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr
* (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr *
(dxa * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi *
(Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 20
* (Xxi * (Xxr * (Xxr * (Xxr * (dya * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi *
Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr
* (Xxr * Xxr)))))))))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * (dya * sgn(((5 *
(Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr *
Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 30 * (Xxi * (Xxi
* (Xxr * (Xxr * (dxa * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10
* (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + daa)))));
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxb * sgn(((5 * (Xxi * (Xxi *
(Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr
* (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + ((5 * (Xxr * (Xxr * (Xxr * (Xxr *
(dxb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi *
(Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 20
* (Xxi * (Xxr * (Xxr * (Xxr * (dyb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi *
Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr
* (Xxr * Xxr)))))))))))) + ((20 * (Xxi * (Xxi * (Xxi * (Xxr * (dyb * sgn(((5 *
(Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr *
Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr * Xxr)))))))))))) + (( - 30 * (Xxi * (Xxi
* (Xxr * (Xxr * (dxb * sgn(((5 * (Xxi * (Xxi * (Xxi * (Xxi * Xxr))))) + (( - 10
* (Xxi * (Xxi * (Xxr * (Xxr * Xxr))))) + (Xxr * (Xxr * (Xxr * (Xxr *
Xxr)))))))))))) + dab)))));
        dyan = (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + (( - 5 * (Xxr *
(Xxr * (Xxr * (Xxr * dya))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dxa)))))
+ ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dxa))))) + ((30 * (Xxi * (Xxi * (Xxr *
(Xxr * dya))))) + dba)))));
        dybn = (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + (( - 5 * (Xxr *
(Xxr * (Xxr * (Xxr * dyb))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dxb)))))
+ ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dxb))))) + ((30 * (Xxi * (Xxi * (Xxr *
(Xxr * dyb))))) + dbb)))));
      </derivative>
    </formula>
  </group>

  <group type="30" name="5th Quasi Burning Ship (BS/Buffalo Hybrid)">
    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr) * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)  + Cr;
        Xin = Ci - abs(Xi * (5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2));
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr, xr) * (Xr2 * Xr2 -  10 * Xr2 * Xi2 + 5 * Xi2 * Xi2) +
abs(Xr + xr) * (4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 *
xr2  - 20 * Xr2 * Xi * xi - 10 * Xr2 * xi2 - 20 * Xr * xr * Xi2 - 40 * Xr * xr
* Xi * xi - 20 * Xr * xr * xi2 - 10 * xr2 * Xi2 - 20 * xr2 * Xi * xi - 10 * xr2
* xi2 + 20 * Xi2 * Xi * xi + 30 * Xi2 * xi2 + 20 * Xi * xi2 * xi + 5 * xi2 *
xi2) + cr;
        xin = ci - diffabs(Xi2 * Xi2 * Xi - 10 * Xi2 * Xi * Xr2 + 5 * Xi * Xr2
* Xr2, 5 * Xi2 * Xi2 * xi - 20 * Xi2 * Xi * xr * Xr + 10 * Xi2 * Xi * xi2 - 10
* Xi2 * Xi * xr2 - 30 * Xi2 * xi * Xr2 - 60 * Xi2 * xr * xi * Xr + 10 * Xi2 *
xi2 * xi - 30 * Xi2 * xr2 * xi + 20 * Xi * xr * Xr2 * Xr - 30 * Xi * xi2 * Xr2
```

```
+ 30 * Xi * xr2 * Xr2 - 60 * Xi * xr * xi2 * Xr + 20 * Xi * xr2 * xr * Xr + 5 *
Xi * xi2 * xi2 - 30 * Xi * xr2 * xi2 + 5 * Xi * xr2 * xr2 + 5 * xi * Xr2 * Xr2
+ 20 * xi * xr * Xr2 * Xr - 10 * xi2 * xi * Xr2 + 30 * xi * xr2 * Xr2 - 20 *
xi2 * xi * xr * Xr + 20 * xi * xr2 * xr * Xr + xi2 * xi2 * xi - 10 * xi2 * xi *
xr2 + 5 * xi * xr2 * xr2);
      </perturbation>
      <derivative t="M">
        dxan = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxa * sgn(Xxr)))))) + ((5 *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr *
(dxa * sgn(Xxr))))))))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi * (Xxi * dya))))) +
(( - 20 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * dya))) + (( - 30 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + daa)))));
        dxbn = ((5 * (Xxi * (Xxi * (Xxi * (Xxi * (dxb * sgn(Xxr)))))) + ((5 *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr *
(dxb * sgn(Xxr))))))))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi * (Xxi * dyb))))) +
(( - 20 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * dyb))) + (( - 30 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + dab)))));
        dyan = (dba + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi *
(Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi * (Xxi *
Xxi)))) + (5 * (abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))))))))) + ((20 *
(sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi
* (Xxi * Xxi)))) + (5 * (abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxa * sgn(Xxr)))))))) + (( - 20 *
(sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi
* (Xxi * Xxi)))) + (5 * (abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))) *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (dxa *
sgn(Xxr))))))))))) + ((30 * (abs((Xxr * Xxr)) * (sgn(((Xxi * (Xxi * (Xxi * (Xxi
* Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * Xxi)))) + (5 *
(abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))) * (Xxi * (Xxi * dya))))) + ( - 5 *
(abs((Xxr * (Xxr * (Xxr * Xxr)))) * (sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) +
(( - 10 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * Xxi)))) + (5 * (abs((Xxr * (Xxr *
(Xxr * Xxr)))) * Xxi))))) * dya)))))))));
        dybn = (dbb + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi *
(Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi * (Xxi *
Xxi)))) + (5 * (abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))))))))) + ((20 *
(sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi
* (Xxi * Xxi)))) + (5 * (abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (dxb * sgn(Xxr)))))))) + (( - 20 *
(sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi
* (Xxi * Xxi)))) + (5 * (abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))) *
(sgn(Xxr) * (sgn(Xxr) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (dxb *
sgn(Xxr))))))))))) + ((30 * (abs((Xxr * Xxr)) * (sgn(((Xxi * (Xxi * (Xxi * (Xxi
* Xxi)))) + (( - 10 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * Xxi)))) + (5 *
(abs((Xxr * (Xxr * (Xxr * Xxr)))) * Xxi))))) * (Xxi * (Xxi * dyb))))) + ( - 5 *
(abs((Xxr * (Xxr * (Xxr * Xxr)))) * (sgn(((Xxi * (Xxi * (Xxi * (Xxi * Xxi)))) +
(( - 10 * (abs((Xxr * Xxr)) * (Xxi * (Xxi * Xxi)))) + (5 * (abs((Xxr * (Xxr *
(Xxr * Xxr)))) * Xxi))))) * dyb)))))))));
      </derivative>
    </formula>
  </group>

  <group type="31" name="5th Quasi Perpendicular">
    <formula power="5" glitch="0.0001">
      <reference t="R">
        Xrn = abs(Xr) * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)  + Cr;
        Xin = Ci - Xi * abs(5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2);
      </reference>
```

```
1870        <perturbation t="R">
              xrn = diffabs(Xr, xr) * (Xr2 * Xr2 -  10 * Xr2 * Xi2 + 5 * Xi2 * Xi2) +
      abs(Xr + xr) * (4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 *
      xr2  - 20 * Xr2 * Xi * xi - 10 * Xr2 * xi2 - 20 * Xr * xr * Xi2 - 40 * Xr * xr
      * Xi * xi - 20 * Xr * xr * xi2 - 10 * xr2 * Xi2 - 20 * xr2 * Xi * xi - 10 * xr2
1875  * xi2 + 20 * Xi2 * Xi * xi + 30 * Xi2 * xi2 + 20 * Xi * xi2 * xi + 5 * xi2 *
      xi2) + cr;
              xin = ( - Xi) * diffabs(5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2, 4 *
      xi * Xi2 * Xi - 20 * xr * Xr * Xi2 + 6 * xi2 * Xi2 - 10 * xr2 * Xi2 - 20 * xi *
      Xr2 * Xi - 40 * xr * xi * Xr * Xi + 4 * xi2 * xi * Xi - 20 * xr2 * xi * Xi + 20
1880  * xr * Xr2 * Xr - 10 * xi2 * Xr2 + 30 * xr2 * Xr2 - 20 * xr * xi2 * Xr + 20 *
      xr2 * xr * Xr + xi2 * xi2 - 10 * xr2 * xi2 + 5 * xr2 * xr2) - xi * abs(Xi2 *
      Xi2 + 4 * xi * Xi2 * Xi - 10 * Xr2 * Xi2 - 20 * xr * Xr * Xi2 + 6 * xi2 * Xi2 -
      10 * xr2 * Xi2 - 20 * xi * Xr2 * Xi - 40 * xr * xi * Xr * Xi + 4 * xi2 * xi *
      Xi - 20 * xr2 * xi * Xi + 5 * Xr2 * Xr2 + 20 * xr * Xr2 * Xr - 10 * xi2 * Xr2 +
1885  30 * xr2 * Xr2 - 20 * xr * xi2 * Xr + 20 * xr2 * xr * Xr + xi2 * xi2 - 10 * xr2
      * xi2 + 5 * xr2 * xr2) + ci;
            </perturbation>
            <derivative t="M">
              dxan = ((Xxr * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))) + ((4 *
1890  (abs(Xxr) * (Xxr * (Xxr * (Xxr * dxa))))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi *
      (dxa * sgn(Xxr)))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * (dxa *
      sgn(Xxr)))))) + (( - 20 * (abs(Xxr) * (Xxi * (Xxi * (Xxr * dxa))))) + (( - 20
      * (abs(Xxr) * (Xxi * (Xxr * (Xxr * dya))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi *
      (Xxi * dya))))) + daa)))))));
1895          dxbn = ((Xxr * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))) + ((4 *
      (abs(Xxr) * (Xxr * (Xxr * (Xxr * dxb))))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi *
      (dxb * sgn(Xxr)))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * (dxb *
      sgn(Xxr)))))) + (( - 20 * (abs(Xxr) * (Xxi * (Xxi * (Xxr * dxb))))) + (( - 20
      * (abs(Xxr) * (Xxi * (Xxr * (Xxr * dyb))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi *
1900  (Xxi * dyb))))) + dab)))))));
              dyan = (dba + (( - 4 * (Xxi * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi *
      (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5 * (Xxr *
      (Xxr * (Xxr * Xxr))))))))))))) + ((20 * (Xxi * (Xxi * (Xxr * (Xxr * (dya *
      sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5
1905  * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr *
      (dxa * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr *
      Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + ((20 * (Xxi * (Xxi *
      (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi *
      (Xxr * Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + ( - 1 *
1910  (abs(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5
      * (Xxr * (Xxr * (Xxr * Xxr))))))) * dya)))))));
              dybn = (dbb + (( - 4 * (Xxi * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi *
      (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5 * (Xxr *
      (Xxr * (Xxr * Xxr))))))))))))) + ((20 * (Xxi * (Xxi * (Xxr * (Xxr * (dyb *
1915  sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5
      * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr *
      (dxb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr *
      Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + ((20 * (Xxi * (Xxi *
      (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi *
1920  (Xxr * Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + ( - 1 *
      (abs(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5
      * (Xxr * (Xxr * (Xxr * Xxr))))))) * dyb)))))));
            </derivative>
          </formula>
1925    </group>
```

```
  <group type="32" name="5th Quasi Heart">
    <formula power="5" glitch="0.00001">
      <reference t="R">
1930          Xrn = abs(Xr) * (Xr2 * Xr2 - 10 * Xr2 * Xi2 + 5 * Xi2 * Xi2)  + Cr;
          Xin = Xi * abs(5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2)  + Ci;
      </reference>
      <perturbation t="R">
          xrn = diffabs(Xr, xr) * (Xr2 * Xr2 -  10 * Xr2 * Xi2 + 5 * Xi2 * Xi2) +
1935  abs(Xr + xr) * (4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 *
      xr2  - 20 * Xr2 * Xi * xi - 10 * Xr2 * xi2 - 20 * Xr * xr * Xi2 - 40 * Xr * xr
      * Xi * xi - 20 * Xr * xr * xi2 - 10 * xr2 * Xi2 - 20 * xr2 * Xi * xi - 10 * xr2
      * xi2 + 20 * Xi2 * Xi * xi + 30 * Xi2 * xi2 + 20 * Xi * xi2 * xi + 5 * xi2 *
      xi2) + cr;
1940          xin = Xi * diffabs(5 * Xr2 * Xr2 - 10 * Xr2 * Xi2 + Xi2 * Xi2, 4 * xi *
      Xi2 * Xi - 20 * xr * Xr * Xi2 + 6 * xi2 * Xi2 - 10 * xr2 * Xi2 - 20 * xi * Xr2
      * Xi - 40 * xr * xi * Xr * Xi + 4 * xi2 * xi * Xi - 20 * xr2 * xi * Xi + 20 *
      xr * Xr2 * Xr - 10 * xi2 * Xr2 + 30 * xr2 * Xr2 - 20 * xr * xi2 * Xr + 20 * xr2
      * xr * Xr + xi2 * xi2 - 10 * xr2 * xi2 + 5 * xr2 * xr2) + xi * abs(Xi2 * Xi2 +
1945  4 * xi * Xi2 * Xi - 10 * Xr2 * Xi2 - 20 * xr * Xr * Xi2 + 6 * xi2 * Xi2 - 10 *
      xr2 * Xi2 - 20 * xi * Xr2 * Xi - 40 * xr * xi * Xr * Xi + 4 * xi2 * xi * Xi -
      20 * xr2 * xi * Xi + 5 * Xr2 * Xr2 + 20 * xr * Xr2 * Xr - 10 * xi2 * Xr2 + 30 *
      xr2 * Xr2 - 20 * xr * xi2 * Xr + 20 * xr2 * xr * Xr + xi2 * xi2 - 10 * xr2 *
      xi2 + 5 * xr2 * xr2) + ci;
1950      </perturbation>
      <derivative t="M">
          dxan = ((Xxr * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))) + ((4 *
      (abs(Xxr) * (Xxr * (Xxr * (Xxr * dxa))))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi *
      (dxa * sgn(Xxr))))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * (dxa *
1955  sgn(Xxr)))))) + (( - 20 * (abs(Xxr) * (Xxi * (Xxi * (Xxr * dxa))))) + (( - 20
      * (abs(Xxr) * (Xxi * (Xxr * (Xxr * dya))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi *
      (Xxi * dya))))) + daa)))))));
          dxbn = ((Xxr * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))) + ((4 *
      (abs(Xxr) * (Xxr * (Xxr * (Xxr * dxb))))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi *
1960  (dxb * sgn(Xxr))))))) + (( - 10 * (Xxi * (Xxi * (Xxr * (Xxr * (dxb *
      sgn(Xxr)))))) + (( - 20 * (abs(Xxr) * (Xxi * (Xxi * (Xxr * dxb))))) + (( - 20
      * (abs(Xxr) * (Xxi * (Xxr * (Xxr * dyb))))) + ((20 * (abs(Xxr) * (Xxi * (Xxi *
      (Xxi * dyb))))) + dab)))))));
          dyan = (dba + ((4 * (Xxi * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi
1965  * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5 * (Xxr * (Xxr *
      (Xxr * Xxr))))))))))) + (( - 20 * (Xxi * (Xxi * (Xxr * (Xxr * (dya *
      sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5
      * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr *
      (dxa * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr *
1970  Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + (( - 20 * (Xxi * (Xxi *
      (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi *
      (Xxr * Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + (abs(((Xxi * (Xxi
      * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5 * (Xxr * (Xxr *
      (Xxr * Xxr))))))) * dya))))));
1975          dybn = (dbb + ((4 * (Xxi * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi
      * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5 * (Xxr * (Xxr *
      (Xxr * Xxr))))))))))))) + (( - 20 * (Xxi * (Xxi * (Xxr * (Xxr * (dyb *
      sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5
      * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr *
1980  (dxb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr *
      Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + (( - 20 * (Xxi * (Xxi *
      (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi *
      (Xxr * Xxr)))) + (5 * (Xxr * (Xxr * (Xxr * Xxr))))))))))))) + (abs(((Xxi * (Xxi
```

```
* (Xxi * Xxi))) + (( - 10 * (Xxi * (Xxi * (Xxr * Xxr)))) + (5 * (Xxr * (Xxr *
1985 (Xxr * Xxr))))))) * dyb))))));
      </derivative>
    </formula>
  </group>

1990   <group type="33" name="SimonBrot 4th">
    <formula power="4" glitch="0.000001">
      <reference t="R">
        Xrn = Xi2 * abs(Xi2) - 4 * Xi * abs(Xr * Xi) * Xr - Xr2 * abs(Xi2) -
Xi2 * abs(Xr2) + Xr2 * abs(Xr2) + Cr;
1995         Xin = Ci - 2 * Xr * Xi * abs(Xi2) - 2 * Xi2 * abs(Xr * Xi) + 2 * Xr2 *
abs(Xr * Xi) + 2 * Xr * Xi * abs(Xr2);
      </reference>
      <perturbation t="R">
        xrn = (Xi2) * diffabs(Xi2,2 * xi * Xi + xi2) - 4 * Xi * Xr * diffabs(Xr
2000 * Xi,Xr * xi + xr * Xi + xr * xi) - Xr2 * diffabs(Xi2,2 * xi * Xi + xi2) - Xi2
* diffabs(Xr2,2 * Xr * xr + xr2) + Xr2 * diffabs(Xr2,2 * Xr * xr + xr2) + (2 *
xi * Xi + xi2) * abs(Xi2 + 2 * xi * Xi + xi2) - 4 * (Xi * xr + xi * Xr + xi *
xr) * abs(Xr * Xi + Xr * xi + xr * Xi + xr * xi) - (2 * Xr * xr + xr2) *
abs(Xi2 + 2 * xi * Xi + xi2) - (2 * xi * Xi + xi2) * abs(Xr2 + 2 * Xr * xr +
2005 xr2) + (2 * Xr * xr + xr2) * abs(Xr2 + 2 * Xr * xr + xr2) + cr;
        xin = 2 * Xr2 * diffabs(Xr * Xi,Xr * xi + xr * Xi + xr * xi) + 2 * Xr *
Xi * diffabs(Xr2,2 * Xr * xr + xr2) - 2 * Xr * Xi * diffabs(Xi2,2 * xi * Xi +
xi2) - 2 * Xi2 * diffabs(Xr * Xi,Xr * xi + xr * Xi + xr * xi) + 2 * (2 * Xr *
xr + xr2) * abs(Xr * Xi + Xr * xi + xr * Xi + xr * xi)  + 2 * (Xr * xi + xr *
2010 Xi + xr * xi) * abs(Xr2 + 2 * Xr * xr + xr2) - 2 * (Xr * xi + xr * Xi + xr *
xi) * abs(Xi2 + 2 * xi * Xi + xi2) - 2 * (2 * xi * Xi + xi2) * abs(Xr * Xi + Xr
* xi + xr * Xi + xr * xi)  + ci;
      </perturbation>
      <derivative t="M">
2015         dxan = ((2 * (abs((Xxr * Xxr)) * (Xxr * dxa))) + (( - 2 * (abs((Xxi *
Xxi)) * (Xxr * dxa))) + ((2 * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxa *
sgn(Xxr)))))))) + (( - 2 * (abs((Xxr * Xxr)) * (Xxi * dya))) + ((2 * (abs((Xxi *
Xxi)) * (Xxi * dya))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dya *
sgn(Xxi)))))))) + (( - 2 * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (dxa *
2020 sgn(Xxr)))))))) + ((2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dya * sgn(Xxi))))))) +
(( - 4 * (abs((Xxi * Xxr)) * (Xxi * dxa))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxr
* dya))) + (( - 4 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dxa * sgn(Xxr))))))) + ((
- 4 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dya * sgn(Xxr)))))))) + daa))))))))))));
        dxbn = ((2 * (abs((Xxr * Xxr)) * (Xxr * dxb))) + (( - 2 * (abs((Xxi *
2025 Xxi)) * (Xxr * dxb))) + ((2 * (sgn(Xxr) * (Xxr * (Xxr * (Xxr * (dxb *
sgn(Xxr)))))))) + (( - 2 * (abs((Xxr * Xxr)) * (Xxi * dyb))) + ((2 * (abs((Xxi *
Xxi)) * (Xxi * dyb))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dyb *
sgn(Xxi)))))))) + (( - 2 * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (dxb *
sgn(Xxr)))))))) + ((2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dyb * sgn(Xxi))))))) +
2030 (( - 4 * (abs((Xxi * Xxr)) * (Xxi * dxb))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxr
* dyb))) + (( - 4 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dxb * sgn(Xxr))))))) + ((
- 4 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dyb * sgn(Xxr)))))))) + dab))))))))))));
        dyan = ((2 * (abs((Xxr * Xxr)) * (Xxr * dya))) + (( - 2 * (abs((Xxi *
Xxi)) * (Xxr * dya))) + ((2 * (abs((Xxr * Xxr)) * (Xxi * dxa))) + ((2 *
2035 (sgn(Xxi) * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxr))))))) + (( - 2 * (abs((Xxi *
Xxi)) * (Xxi * dxa))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dxa *
sgn(Xxr)))))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dya *
sgn(Xxr)))))))) + ((2 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr))))))) +
((4 * (abs((Xxi * Xxr)) * (Xxr * dxa))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxi *
2040 dya))) + ((4 * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr))))))) + (( - 4
```

```
* (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dya * sgn(Xxi)))))) + dba))))))))))));
        dybn = ((2 * (abs((Xxr * Xxr)) * (Xxr * dyb))) + (( - 2 * (abs((Xxi *
Xxi)) * (Xxr * dyb))) + ((2 * (abs((Xxr * Xxr)) * (Xxi * dxb))) + ((2 *
(sgn(Xxi) * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxr))))))) + (( - 2 * (abs((Xxi *
Xxi)) * (Xxi * dxb))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (dxb *
sgn(Xxr))))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dyb *
sgn(Xxr))))))) + ((2 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr))))))) +
((4 * (abs((Xxi * Xxr)) * (Xxr * dxb))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxi *
dyb))) + ((4 * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr))))))) + (( - 4
* (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dyb * sgn(Xxi)))))) + dbb))))))))))));
      </derivative>
    </formula>
  </group>

  <group type="34" name="4th Imag Quasi Perpendicular / Heart">
    <formula power="4" glitch="0.000001">
      <reference t="R">
        Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
        Xin = 4 * Xr * abs(Xi * Xr2 - Xi * Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
+ 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
* Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2  + cr;
        xin = 4 * (xr) * abs( -  Xi2 * Xi - 3 * xi * Xi2 + Xr2 * Xi + 2 * xr *
Xr * Xi - 3 * xi2 * Xi + xr2 * Xi + xi * Xr2 + 2 * xr * xi * Xr - xi2 * xi +
xr2 * xi) + 4 * Xr * diffabs( -  Xi2 * Xi + Xr2 * Xi, - 3 * xi * Xi2 + 2 * xr *
Xr * Xi - 3 * xi2 * Xi + xr2 * Xi + xi * Xr2 + 2 * xr * xi * Xr - xi2 * xi +
xr2 * xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dya)))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dyb)))) + dab))));
        dyan = (dba + ((4 * (Xxr * (Xxr * (Xxr * (dya * sgn((( - 1 * (Xxi *
(Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxr * (Xxr * (dxa *
sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))))))) + (( - 12 * (Xxi
* (Xxi * (Xxr * (dya * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr *
Xxr))))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))
* dxa))))));
        dybn = (dbb + ((4 * (Xxr * (Xxr * (Xxr * (dyb * sgn((( - 1 * (Xxi *
(Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxr * (Xxr * (dxb *
sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))))))) + (( - 12 * (Xxi
* (Xxi * (Xxr * (dyb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr *
Xxr))))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))
* dxb))))));
      </derivative>
    </formula>
  </group>

  <group type="35" name="4th Real Quasi Perpendicular">
    <formula power="4" glitch="0.000001">
      <reference t="R">
```

```
          Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
          Xin = Ci - 4 * Xi * abs(Xr * Xr2 - Xr * Xi2);
        </reference>
        <perturbation t="R">
          xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
      + 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
      * Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
      Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2  + cr;
          xin = ( - 4) * Xi * diffabs(Xr2 * Xr - Xr * Xi2, - xr * Xi2 - 2 * xi *
      Xr * Xi - 2 * xr * xi * Xi + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2
      + xr2 * xr) - 4 * xi * abs( -  Xr * Xi2 - xr * Xi2 - 2 * xi * Xr * Xi - 2 * xr
      * xi * Xi + Xr2 * Xr + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2 + xr2
      * xr) + ci;
        </perturbation>
        <derivative t="M">
          dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
      dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
      * dya)))) + daa))));
          dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
      dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
      * dyb)))) + dab))));
          dyan = (dba + ((4 * (Xxi * (Xxi * (Xxi * (dxa * sgn((( - 1 * (Xxi *
      (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dya *
      sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 12 * (Xxi
      * (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
      Xxr)))))))) + ( - 4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
      Xxr)))) * dya))))));
          dybn = (dbb + ((4 * (Xxi * (Xxi * (Xxi * (dxb * sgn((( - 1 * (Xxi *
      (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dyb *
      sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 12 * (Xxi
      * (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
      Xxr)))))))) + ( - 4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
      Xxr)))) * dyb))))));
        </derivative>
      </formula>
    </group>

    <group type="36" name="4th Real Quasi Heart">
      <formula power="4" glitch="0.000001">
        <reference t="R">
          Xrn = Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2  + Cr;
          Xin = 4 * Xi * abs(Xr * Xr2 - Xr * Xi2) + Ci;
        </reference>
        <perturbation t="R">
          xrn = 4 * Xr2 * Xr * xr + 6 * Xr2 * xr2 + 4 * Xr * xr2 * xr + xr2 * xr2
      + 4 * Xi2 * Xi * xi + 6 * Xi2 * xi2 + 4 * Xi * xi2 * xi + xi2 * xi2 - 12 * Xr2
      * Xi * xi - 6 * Xr2 * xi2 - 12 * Xr * xr * Xi2 - 24 * Xr * xr * Xi * xi - 12 *
      Xr * xr * xi2 - 6 * xr2 * Xi2 - 12 * xr2 * Xi * xi - 6 * xr2 * xi2  + cr;
          xin = 4 * Xi * diffabs(Xr2 * Xr - Xr * Xi2, - xr * Xi2 - 2 * xi * Xr *
      Xi - 2 * xr * xi * Xi + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2 + xr2
      * xr) + 4 * xi * abs( -  Xr * Xi2 - xr * Xi2 - 2 * xi * Xr * Xi - 2 * xr * xi *
      Xi + Xr2 * Xr + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2 + xr2 * xr) +
      ci;
        </perturbation>
        <derivative t="M">
          dxan = ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
      dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
```

```
* dya)))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dyb)))) + dab))));
        dyan = (dba + (( - 4 * (Xxi * (Xxi * (Xxi * (dxa * sgn((( - 1 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 8 * (Xxi * (Xxi * (Xxr * (dya
* sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + ((12 * (Xxi
* (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr)))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))
* dya))))));
        dybn = (dbb + (( - 4 * (Xxi * (Xxi * (Xxi * (dxb * sgn((( - 1 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 8 * (Xxi * (Xxi * (Xxr * (dyb
* sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + ((12 * (Xxi
* (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr)))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))
* dyb))))));
      </derivative>
    </formula>
  </group>

  <group type="37" name="4th Celtic Imag Quasi Perpendicular / Heart">
    <formula power="4" glitch="0.000001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = 4 * Xr * abs(Xi * Xr2 - Xi * Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2,4 * Xi2 * Xi * xi -
12 * Xi2 * xr * Xr + 6 * Xi2 * xi2 - 6 * Xi2 * xr2 - 12 * Xr2 * Xi * xi - 24 *
Xr * Xi * xr * xi + 4 * xi2 * xi * Xi - 12 * xi * Xi * xr2 + 4 * Xr2 * Xr * xr
- 6 * Xr2 * xi2 + 6 * Xr2 * xr2 - 12 * xi2 * Xr * xr + 4 * xr2 * xr * Xr + xi2
* xi2 - 6 * xi2 * xr2 + xr2 * xr2)  + cr;
        xin = 4 * xr * abs( -  Xi2 * Xi - 3 * xi * Xi2 + Xr2 * Xi + 2 * xr * Xr
* Xi - 3 * xi2 * Xi + xr2 * Xi + xi * Xr2 + 2 * xr * xi * Xr - xi2 * xi + xr2 *
xi) + 4 * Xr * diffabs( -  Xi2 * Xi + Xr2 * Xi, - 3 * xi * Xi2 + 2 * xr * Xr *
Xi - 3 * xi2 * Xi + xr2 * Xi + xi * Xr2 + 2 * xr * xi * Xr - xi2 * xi + xr2 *
xi) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr)))))))))) + dab))));
```

```
        dyan = (dba + ((4 * (Xxr * (Xxr * (Xxr * (dya * sgn((( - 1 * (Xxi *
(Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxr * (Xxr * (dxa *
sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr)))))))))) + (( - 12 * (Xxi
* (Xxi * (Xxr * (dya * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr *
Xxr)))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))
* dxa))))));
        dybn = (dbb + ((4 * (Xxr * (Xxr * (Xxr * (dyb * sgn((( - 1 * (Xxi *
(Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxr * (Xxr * (dxb *
sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr)))))))))) + (( - 12 * (Xxi
* (Xxi * (Xxr * (dyb * sgn((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr *
Xxr)))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxi))) + (Xxi * (Xxr * Xxr))))
* dxb))))));
      </derivative>
    </formula>
  </group>

  <group type="38" name="4th Celtic Real Quasi Perpendicular">
    <formula power="4" glitch="0.000001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = Ci - 4 * Xi * abs(Xr * Xr2 - Xr * Xi2);
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2,4 * Xi2 * Xi * xi -
12 * Xi2 * xr * Xr + 6 * Xi2 * xi2 - 6 * Xi2 * xr2 - 12 * Xr2 * Xi * xi - 24 *
Xr * Xi * xr * xi + 4 * xi2 * xi * Xi - 12 * xi * Xi * xr2 + 4 * Xr2 * Xr * xr
- 6 * Xr2 * xi2 + 6 * Xr2 * xr2 - 12 * xi2 * Xr * xr + 4 * xr2 * xr * Xr + xi2
* xi2 - 6 * xi2 * xr2 + xr2 * xr2)  + cr;
        xin = ( - 4) * Xi * diffabs(Xr2 * Xr - Xr * Xi2, - xr * Xi2 - 2 * xi *
Xr * Xi - 2 * xr * xi * Xi + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2
+ xr2 * xr) - 4 * xi * abs( -  Xr * Xi2 - xr * Xi2 - 2 * xi * Xr * Xi - 2 * xr
* xi * Xi + Xr2 * Xr + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2 + xr2
* xr) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + dab))));
        dyan = (dba + ((4 * (Xxi * (Xxi * (Xxi * (dxa * sgn((( - 1 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dya *
sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxi
* (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
```

```
Xxr))))))))) + ( - 4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr)))) * dya))))));
        dybn = (dbb + ((4 * (Xxi * (Xxi * (Xxi * (dxb * sgn((( - 1 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))))))) + ((8 * (Xxi * (Xxi * (Xxr * (dyb *
sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))))) + (( - 12 * (Xxi
* (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr))))))))) + ( - 4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr)))) * dyb))))));
      </derivative>
    </formula>
  </group>

  <group type="39" name="4th Celtic Real Quasi Heart">
    <formula power="4" glitch="0.000001">
      <reference t="R">
        Xrn = abs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2)  + Cr;
        Xin = 4 * Xi * abs(Xr * Xr2 - Xr * Xi2) + Ci;
      </reference>
      <perturbation t="R">
        xrn = diffabs(Xr2 * Xr2 + Xi2 * Xi2 - 6 * Xr2 * Xi2,4 * Xi2 * Xi * xi -
12 * Xi2 * xr * Xr + 6 * Xi2 * xi2 - 6 * Xi2 * xr2 - 12 * Xr2 * Xi * xi - 24 *
Xr * Xi * xr * xi + 4 * xi2 * xi * Xi - 12 * xi * Xi * xr2 + 4 * Xr2 * Xr * xr
- 6 * Xr2 * xi2 + 6 * Xr2 * xr2 - 12 * xi2 * Xr * xr + 4 * xr2 * xr * Xr + xi2
* xi2 - 6 * xi2 * xr2 + xr2 * xr2)  + cr;
        xin = 4 * Xi * diffabs(Xr2 * Xr - Xr * Xi2, - xr * Xi2 - 2 * xi * Xr *
Xi - 2 * xr * xi * Xi + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2 + xr2
* xr) + 4 * xi * abs( -  Xr * Xi2 - xr * Xi2 - 2 * xi * Xr * Xi - 2 * xr * xi *
Xi + Xr2 * Xr + 3 * xr * Xr2 - xi2 * Xr + 3 * xr2 * Xr - xr * xi2 + xr2 * xr) +
ci;
      </perturbation>
      <derivative t="M">
        dxan = ((4 * (Xxi * (Xxi * (Xxi * (dya * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxa * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dya * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + daa))));
        dxbn = ((4 * (Xxi * (Xxi * (Xxi * (dyb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + ((4 * (Xxr * (Xxr * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi *
Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxi * (Xxr * (dxb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + (( - 12 * (Xxi * (Xxr * (Xxr * (dyb * sgn(((Xxi * (Xxi * (Xxi
* Xxi))) + (( - 6 * (Xxi * (Xxi * (Xxr * Xxr)))) + (Xxr * (Xxr * (Xxr *
Xxr))))))))))) + dab))));
        dyan = (dba + (( - 4 * (Xxi * (Xxi * (Xxi * (dxa * sgn((( - 1 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 8 * (Xxi * (Xxi * (Xxr * (dya
* sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + ((12 * (Xxi
* (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr))))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))
* dya))))));
        dybn = (dbb + (( - 4 * (Xxi * (Xxi * (Xxi * (dxb * sgn((( - 1 * (Xxi *
(Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + (( - 8 * (Xxi * (Xxi * (Xxr * (dyb
```

```
* sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr)))))))) + ((12 * (Xxi
* (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr *
Xxr)))))))) + (4 * (abs((( - 1 * (Xxi * (Xxi * Xxr))) + (Xxr * (Xxr * Xxr))))
* dyb))))));
      </derivative>
    </formula>
  </group>

  <group type="40" name="SimonBrot 6th">
    <formula power="6" glitch="0.0001">
      <reference t="R">
        Xrn = 3 * abs(Xr2 * Xi) * Xi2 * Xi - abs(Xi2 * Xi) * Xi2 * Xi + 9 *
abs(Xr * Xi2) * Xr * Xi2 - 3 * abs(Xr2 * Xr) * Xr * Xi2 + 3 * abs(Xi2 * Xi) *
Xr2 * Xi - 9 * abs(Xr2 * Xi) * Xr2 * Xi - 3 * abs(Xr * Xi2) * Xr2 * Xr +
abs(Xr2 * Xr) * Xr2 * Xr + Cr;
        Xin = 3 * Xi2 * Xi * abs(Xr * Xi2) - abs(Xr2 * Xr) * Xi2 * Xi - 9 *
abs(Xr2 * Xi) * Xi2 * Xr - 9 * abs(Xr * Xi2) * Xi * Xr2 + 3 * abs(Xr2 * Xr) *
Xi * Xr2 + 3 * abs(Xi2 * Xi) * Xr * Xi2 - abs(Xi2 * Xi) * Xr2 * Xr + 3 * abs(Xi
* Xr2) * Xr2 * Xr + Ci;
      </reference>
      <perturbation t="R">
        xrn= 3 * diffabs(Xr2 * Xi,Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr * xr *
xi + xr2 * Xi + xr2 * xi) * (Xi2 * Xi) - diffabs(Xi2 * Xi,3 * Xi2 * xi + 3 * Xi
* xi2 + xi2 * xi) * (Xi2 * Xi) + 9 * diffabs(Xr * Xi2,2 * Xr * Xi * xi + Xr *
xi2 + xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) * (Xr * Xi2) - 3 * diffabs(Xr2 *
Xr,3 * Xr2 * xr + 3 * Xr * xr2 + xr2 * xr) * (Xr * Xi2) + 3 * diffabs(Xi2 *
Xi,3 * Xi2 * xi + 3 * Xi * xi2 + xi2 * xi) * (Xr2 * Xi) - 9 * diffabs(Xr2 *
Xi,Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) * (Xr2
* Xi) - 3 * diffabs(Xr * Xi2,2 * Xr * Xi * xi + Xr * xi2 + xr * Xi2 + 2 * xr *
Xi * xi + xr * xi2) * (Xr2 * Xr) + diffabs(Xr2 * Xr,3 * Xr2 * xr + 3 * Xr * xr2
+ xr2 * xr) * (Xr2 * Xr)          + 3 * abs(Xr2 * Xi + Xr2 * xi + 2 * Xr * xr *
Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) * (3 * Xi2 * xi + 3 * Xi * xi2 +
xi2 * xi) - abs(Xi2 * Xi + 3 * Xi2 * xi + 3 * Xi * xi2 + xi2 * xi) * (3 * Xi2 *
xi + 3 * Xi * xi2 + xi2 * xi) + 9 * abs(Xr * Xi2 + 2 * Xr * Xi * xi + Xr * xi2
+ xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) * (2 * Xr * Xi * xi + Xr * xi2 + xr *
Xi2 + 2 * xr * Xi * xi + xr * xi2) - 3 * abs(Xr2 * Xr + 3 * Xr2 * xr + 3 * Xr *
xr2 + xr2 * xr) * (2 * Xr * Xi * xi + Xr * xi2 + xr * Xi2 + 2 * xr * Xi * xi +
xr * xi2) + 3 * abs(Xi2 * Xi + 3 * Xi2 * xi + 3 * Xi * xi2 + xi2 * xi) * (Xr2 *
xi + 2 * Xr * xr * Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) - 9 * abs(Xr2 *
Xi + Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) *
(Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) - 3 *
abs(Xr * Xi2 + 2 * Xr * Xi * xi + Xr * xi2 + xr * Xi2 + 2 * xr * Xi * xi + xr *
xi2) * (3 * Xr2 * xr + 3 * Xr * xr2 + xr2 * xr) + abs(Xr2 * Xr + 3 * Xr2 * xr +
3 * Xr * xr2 + xr2 * xr) * (3 * Xr2 * xr + 3 * Xr * xr2 + xr2 * xr)          +
cr;
        xin= 3 * diffabs(Xr * Xi2,2 * Xr * Xi * xi + Xr * xi2 + xr * Xi2 + 2 *
xr * Xi * xi + xr * xi2) * (Xi2 * Xi) - diffabs(Xr2 * Xr,3 * Xr2 * xr + 3 * Xr
* xr2 + xr2 * xr) * (Xi2 * Xi) - 9 * diffabs(Xr2 * Xi,Xr2 * xi + 2 * Xr * xr *
Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) * (Xr * Xi2) - 9 * diffabs(Xr *
Xi2,2 * Xr * Xi * xi + Xr * xi2 + xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) *
(Xr2 * Xi) + 3 * diffabs(Xr2 * Xr,3 * Xr2 * xr + 3 * Xr * xr2 + xr2 * xr) *
(Xr2 * Xi) + 3 * diffabs(Xi2 * Xi,3 * Xi2 * xi + 3 * Xi * xi2 + xi2 * xi) * (Xr
* Xi2) - diffabs(Xi2 * Xi,3 * Xi2 * xi + 3 * Xi * xi2 + xi2 * xi) * (Xr2 * Xr)
+ 3 * diffabs(Xr2 * Xi,Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr * xr * xi + xr2 *
Xi + xr2 * xi) * (Xr2 * Xr)          + 3 * abs(Xr * Xi2 + 2 * Xr * Xi * xi + Xr
* xi2 + xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) * (3 * Xi2 * xi + 3 * Xi * xi2
+ xi2 * xi) - abs(Xr2 * Xr + 3 * Xr2 * xr + 3 * Xr * xr2 + xr2 * xr) * (3 * Xi2
```

```
* xi + 3 * Xi * xi2 + xi2 * xi) - 9 * abs(Xr2 * Xi + Xr2 * xi + 2 * Xr * xr *
Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) * (2 * Xr * Xi * xi + Xr * xi2 +
2385 xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) - 9 * abs(Xr * Xi2 + 2 * Xr * Xi * xi +
Xr * xi2 + xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) * (Xr2 * xi + 2 * Xr * xr *
Xi + 2 * Xr * xr * xi + xr2 * Xi + xr2 * xi) + 3 * abs(Xr2 * Xr + 3 * Xr2 * xr
+ 3 * Xr * xr2 + xr2 * xr) * (Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr * xr * xi +
xr2 * Xi + xr2 * xi) + 3 * abs(Xi2 * Xi + 3 * Xi2 * xi + 3 * Xi * xi2 + xi2 *
2390 xi) * (2 * Xr * Xi * xi + Xr * xi2 + xr * Xi2 + 2 * xr * Xi * xi + xr * xi2) -
abs(Xi2 * Xi + 3 * Xi2 * xi + 3 * Xi * xi2 + xi2 * xi) * (3 * Xr2 * xr + 3 * Xr
* xr2 + xr2 * xr) + 3 * abs(Xr2 * Xi + Xr2 * xi + 2 * Xr * xr * Xi + 2 * Xr *
xr * xi + xr2 * Xi + xr2 * xi) * (3 * Xr2 * xr + 3 * Xr * xr2 + xr2 * xr)
  + ci;
2395       </perturbation>
       <derivative t='M'>
         dxan = ((3 * (abs((Xxi * (Xxi * Xxi))) * (Xxr * (Xxr * dya)))) + ((3 *
(abs((Xxr * (Xxr * Xxr))) * (Xxr * (Xxr * dxa)))) + ((3 * (sgn(Xxr) * (sgn(Xxr)
* (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr))))))))) + (( - 3 *
2400 (abs((Xxi * (Xxi * Xxi))) * (Xxi * (Xxi * dya)))) + (( - 3 * (abs((Xxr * (Xxr *
Xxr))) * (Xxi * (Xxi * dxa)))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))))) + ((3 * (sgn(Xxi) * (sgn(Xxr) *
(Xxi * (Xxi * (Xxi * (Xxr * (Xxr * (dya * sgn(Xxr))))))))) + (( - 3 *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * (dya *
2405 sgn(Xxi))))))))) + (( - 6 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * (Xxr * dya))))
+ ((6 * (abs((Xxi * (Xxi * Xxi))) * (Xxi * (Xxr * dxa)))) + (( - 6 * (sgn(Xxi)
* (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxr)))))))))) + ((6
* (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dxa *
sgn(Xxr))))))))) + (( - 9 * (abs((Xxi * (Xxi * Xxr))) * (Xxr * (Xxr * dxa))))
2410 + (( - 9 * (abs((Xxi * (Xxr * Xxr))) * (Xxr * (Xxr * dya)))) + (( - 9 *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * (dya *
sgn(Xxr))))))))) + ((9 * (abs((Xxi * (Xxi * Xxr))) * (Xxi * (Xxi * dxa)))) +
((9 * (abs((Xxi * (Xxr * Xxr))) * (Xxi * (Xxi * dya)))) + (( - 9 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))))) + ((9 *
2415 (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dxa *
sgn(Xxr))))))))) + ((9 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr *
(Xxr * (dya * sgn(Xxi)))))))))) + ((18 * (abs((Xxi * (Xxi * Xxr))) * (Xxi *
(Xxr * dya)))) + (( - 18 * (abs((Xxi * (Xxr * Xxr))) * (Xxi * (Xxr * dxa)))) +
(( - 18 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (Xxr * (dxa *
2420 sgn(Xxr))))))))) + ((18 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr *
(Xxr * (dya * sgn(Xxr)))))))) + daa))))))))))))))))))))));
         dxbn = ((3 * (abs((Xxi * (Xxi * Xxi))) * (Xxr * (Xxr * dyb)))) + ((3 *
(abs((Xxr * (Xxr * Xxr))) * (Xxr * (Xxr * dxb)))) + ((3 * (sgn(Xxr) * (sgn(Xxr)
* (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr))))))))) + (( - 3 *
2425 (abs((Xxi * (Xxi * Xxi))) * (Xxi * (Xxi * dyb)))) + (( - 3 * (abs((Xxr * (Xxr *
Xxr))) * (Xxi * (Xxi * dxb)))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr))))))))) + ((3 * (sgn(Xxi) * (sgn(Xxr) *
(Xxi * (Xxi * (Xxi * (Xxr * (Xxr * (dyb * sgn(Xxr))))))))) + (( - 3 *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * (dyb *
2430 sgn(Xxi))))))))) + (( - 6 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * (Xxr * dyb))))
+ ((6 * (abs((Xxi * (Xxi * Xxi))) * (Xxi * (Xxr * dxb)))) + (( - 6 * (sgn(Xxi)
* (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxr)))))))))) + ((6
* (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dxb *
sgn(Xxr))))))))) + (( - 9 * (abs((Xxi * (Xxi * Xxr))) * (Xxr * (Xxr * dxb))))
2435 + (( - 9 * (abs((Xxi * (Xxr * Xxr))) * (Xxr * (Xxr * dyb)))) + (( - 9 *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * (dyb *
sgn(Xxr))))))))) + ((9 * (abs((Xxi * (Xxi * Xxr))) * (Xxi * (Xxi * dxb)))) +
((9 * (abs((Xxi * (Xxr * Xxr))) * (Xxi * (Xxi * dyb)))) + (( - 9 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))))) + ((9 *
```

```
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dxb *
sgn(Xxr))))))))) + ((9 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr *
(Xxr * (dyb * sgn(Xxi))))))))) + ((18 * (abs((Xxi * (Xxi * Xxr))) * (Xxi *
(Xxr * dyb)))) + (( - 18 * (abs((Xxi * (Xxr * Xxr))) * (Xxi * (Xxr * dxb)))) +
(( - 18 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxr * (Xxr * (Xxr * (dxb *
sgn(Xxr)))))))))) + ((18 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr *
(Xxr * (dyb * sgn(Xxr))))))))) + dab))))))))))))))))))))));
        dyan = ((3 * (abs((Xxr * (Xxr * Xxr))) * (Xxr * (Xxr * dya)))) + (( - 3
* (abs((Xxi * (Xxi * Xxi))) * (Xxr * (Xxr * dxa)))) + ((3 * (sgn(Xxi) *
(sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxr))))))))) + (( -
3 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * (Xxi * dya)))) + ((3 * (abs((Xxi * (Xxi
* Xxi))) * (Xxi * (Xxi * dxa)))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi
* (Xxr * (Xxr * (Xxr * (dya * sgn(Xxi))))))))) + (( - 3 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr))))))))) + ((3 *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * (dxa *
sgn(Xxr))))))))) + ((6 * (abs((Xxi * (Xxi * Xxi))) * (Xxi * (Xxr * dya)))) +
((6 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * (Xxr * dxa)))) + ((6 * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dya * sgn(Xxr))))))))) + ((6 *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * (dxa *
sgn(Xxr))))))))) + (( - 9 * (abs((Xxi * (Xxi * Xxr))) * (Xxr * (Xxr * dya))))
+ ((9 * (abs((Xxi * (Xxi * Xxr))) * (Xxi * (Xxi * dya)))) + ((9 * (abs((Xxi *
(Xxr * Xxr))) * (Xxr * (Xxr * dxa)))) + ((9 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (Xxr * (Xxr * (dxa * sgn(Xxr))))))))) + (( - 9 * (abs((Xxi *
(Xxr * Xxr))) * (Xxi * (Xxi * dxa)))) + (( - 9 * (sgn(Xxi) * (sgn(Xxr) * (Xxi *
(Xxi * (Xxr * (Xxr * (Xxr * (dya * sgn(Xxr))))))))) + ((9 * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dya * sgn(Xxi))))))))) + (( -
9 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * (dxa *
sgn(Xxr))))))))) + (( - 18 * (abs((Xxi * (Xxi * Xxr))) * (Xxi * (Xxr * dxa))))
+ (( - 18 * (abs((Xxi * (Xxr * Xxr))) * (Xxi * (Xxr * dya)))) + (( - 18 *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (Xxr * (Xxr * (dya *
sgn(Xxr))))))))) + (( - 18 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr
* (Xxr * (dxa * sgn(Xxr))))))))) + dba))))))))))))))))))))));
        dybn = ((3 * (abs((Xxr * (Xxr * Xxr))) * (Xxr * (Xxr * dyb)))) + (( - 3
* (abs((Xxi * (Xxi * Xxi))) * (Xxr * (Xxr * dxb)))) + ((3 * (sgn(Xxi) *
(sgn(Xxr) * (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxr))))))))) + (( -
3 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * (Xxi * dyb)))) + ((3 * (abs((Xxi * (Xxi
* Xxi))) * (Xxi * (Xxi * dxb)))) + (( - 3 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi
* (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxi))))))))) + (( - 3 * (sgn(Xxr) *
(sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr))))))))) + ((3 *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * (dxb *
sgn(Xxr))))))))) + ((6 * (abs((Xxi * (Xxi * Xxi))) * (Xxi * (Xxr * dyb)))) +
((6 * (abs((Xxr * (Xxr * Xxr))) * (Xxi * (Xxr * dxb)))) + ((6 * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dyb * sgn(Xxr))))))))) + ((6 *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * (dxb *
sgn(Xxr))))))))) + (( - 9 * (abs((Xxi * (Xxi * Xxr))) * (Xxr * (Xxr * dyb))))
+ ((9 * (abs((Xxi * (Xxi * Xxr))) * (Xxi * (Xxi * dyb)))) + ((9 * (abs((Xxi *
(Xxr * Xxr))) * (Xxr * (Xxr * dxb)))) + ((9 * (sgn(Xxr) * (sgn(Xxr) * (Xxi *
(Xxr * (Xxr * (Xxr * (Xxr * (dxb * sgn(Xxr))))))))) + (( - 9 * (abs((Xxi *
(Xxr * Xxr))) * (Xxi * (Xxi * dxb)))) + (( - 9 * (sgn(Xxi) * (sgn(Xxr) * (Xxi *
(Xxi * (Xxr * (Xxr * (Xxr * (dyb * sgn(Xxr))))))))) + ((9 * (sgn(Xxi) *
(sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * (dyb * sgn(Xxi))))))))) + (( -
9 * (sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * (dxb *
sgn(Xxr))))))))) + (( - 18 * (abs((Xxi * (Xxi * Xxr))) * (Xxi * (Xxr * dxb))))
+ (( - 18 * (abs((Xxi * (Xxr * Xxr))) * (Xxi * (Xxr * dyb)))) + (( - 18 *
(sgn(Xxi) * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (Xxr * (Xxr * (dyb *
sgn(Xxr))))))))) + (( - 18 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxi * (Xxi * (Xxr
* (Xxr * (dxb * sgn(Xxr))))))))) + dbb))))))))))))))))))))));
```

```
      </derivative>
    </formula>
  </group>

  <group type="41" name="HPDZ Buffalo">
    <formula power="2" glitch="0.0001">
      <reference t="R">
        Xrn = (Xr2 - Xi2) - abs(Xr) + Cr;
        Xin = abs(Xr * Xi) * 2 - abs(Xi) + Ci;
      </reference>
      <perturbation t="R">
        xrn = xr * (2 * Xr + xr) - xi * (xi + 2 * Xi) - diffabs(Xr,xr) + cr;
        xin = diffabs(Xr * Xi,Xr * xi + xr * Xi + xr * xi) * 2 - diffabs(Xi,xi)
+ ci;
      </perturbation>
      <derivative t="M">
        dxan = (daa + ((2 * (Xxr * dxa)) + (( - 2 * (Xxi * dya)) + ( - 1 * (dxa
* sgn(Xxr))))));
        dxbn = (dab + ((2 * (Xxr * dxb)) + (( - 2 * (Xxi * dyb)) + ( - 1 * (dxb
* sgn(Xxr))))));
        dyan = (dba + ((2 * (sgn(Xxi) * (Xxi * (dxa * sgn(Xxr))))) + ((2 *
(sgn(Xxi) * (Xxr * (dya * sgn(Xxr))))) + ( - 1 * (dya * sgn(Xxi))))));
        dybn = (dbb + ((2 * (sgn(Xxi) * (Xxi * (dxb * sgn(Xxr))))) + ((2 *
(sgn(Xxi) * (Xxr * (dyb * sgn(Xxr))))) + ( - 1 * (dyb * sgn(Xxi))))));
      </derivative>
    </formula>
  </group>

  <group type="42" name="TheRedshiftRider 1: a * z^2 + z^3 + c">
    <formula power="3" glitch="0.0001">
      <reference t="C">
        Xn = A * (X^2) + (X^3) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + x2) + 3 * X2 * x + 3 * X * x2 + x2 * x + c;
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + ((3 * (Xxr * (Xxr * dxa))) + (( - 3 * (Xxi * (Xxi * dxa))) + ((
- 6 * (Xxi * (Xxr * dya))) + daa)))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + ((3 * (Xxr * (Xxr * dxb))) + (( - 3 * (Xxi * (Xxi * dxb))) + ((
- 6 * (Xxi * (Xxr * dyb))) + dab)))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + ((3 * (Xxr * (Xxr * dya))) + (( - 3 * (Xxi * (Xxi * dya))) +
((6 * (Xxi * (Xxr * dxa))) + dba)))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + ((3 * (Xxr * (Xxr * dyb))) + (( - 3 * (Xxi * (Xxi * dyb))) +
((6 * (Xxi * (Xxr * dxb))) + dbb)))))));
      </derivative>
    </formula>
  </group>
```

```
  <group type="43" name="TheRedshiftRider 2: a * z^2 - z^3 + c">
    <formula power="3" glitch="0.0001">
      <reference t="C">
        Xn = A * (X^2) - (X^3) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + x2) - 3 * X2 * x - 3 * X * x2 - x2 * x + c;
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + (( - 3 * (Xxr * (Xxr * dxa))) + ((3 * (Xxi * (Xxi * dxa))) +
((6 * (Xxi * (Xxr * dya))) + daa)))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + (( - 3 * (Xxr * (Xxr * dxb))) + ((3 * (Xxi * (Xxi * dxb))) +
((6 * (Xxi * (Xxr * dyb))) + dab)))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + (( - 3 * (Xxr * (Xxr * dya))) + ((3 * (Xxi * (Xxi * dya))) + ((
- 6 * (Xxi * (Xxr * dxa))) + dba)))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + (( - 3 * (Xxr * (Xxr * dyb))) + ((3 * (Xxi * (Xxi * dyb))) + ((
- 6 * (Xxi * (Xxr * dxb))) + dbb)))))));
      </derivative>
    </formula>
  </group>

  <group type="44" name="TheRedshiftRider 3: 2 * z^2 - z^3 + c">
    <formula power="3" glitch="0.0001">
      <reference t="C">
        Xn = 2 * (X^2) - (X^3) + C;
      </reference>
      <perturbation t="C">
        xn = 2 * (2 * X * x + x2) - 3 * X2 * x - 3 * X * x2 - x2 * x + c;
      </perturbation>
      <derivative t="M">
        dxan = (( - 3 * (Xxr * (Xxr * dxa))) + ((3 * (Xxi * (Xxi * dxa))) + ((4
* (Xxr * dxa)) + (( - 4 * (Xxi * dya)) + ((6 * (Xxi * (Xxr * dya))) + daa)))));
        dxbn = (( - 3 * (Xxr * (Xxr * dxb))) + ((3 * (Xxi * (Xxi * dxb))) + ((4
* (Xxr * dxb)) + (( - 4 * (Xxi * dyb)) + ((6 * (Xxi * (Xxr * dyb))) + dab)))));
        dyan = (( - 3 * (Xxr * (Xxr * dya))) + ((3 * (Xxi * (Xxi * dya))) + ((4
* (Xxi * dxa)) + ((4 * (Xxr * dya)) + (( - 6 * (Xxi * (Xxr * dxa))) + dba)))));
        dybn = (( - 3 * (Xxr * (Xxr * dyb))) + ((3 * (Xxi * (Xxi * dyb))) + ((4
* (Xxi * dxb)) + ((4 * (Xxr * dyb)) + (( - 6 * (Xxi * (Xxr * dxb))) + dbb)))));
      </derivative>
    </formula>
  </group>

  <group type="45" name="TheRedshiftRider 4: a * z^2 + z^4 + c">
    <formula power="4" glitch="0.0001">
      <reference t="C">
        Xn = A * (X^2) + (X^4) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + x2) + 4 * (X^3) * x + 6 * (X^2) * x2 + 4 * X * x2
```

```
* x + (x2^2) + c;
      </perturbation>
      <derivative t='M'>
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
2615 g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + ((4 * (Xxi * (Xxi * (Xxi * dya)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxa)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dya)))) + daa)))))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
2620 g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + ((4 * (Xxi * (Xxi * (Xxi * dyb)))) + ((4 * (Xxr * (Xxr * (Xxr *
dxb)))) + (( - 12 * (Xxi * (Xxi * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr
* dyb)))) + dab)))))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
2625 g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + ((4 * (Xxr * (Xxr * (Xxr * dya)))) + (( - 4 * (Xxi * (Xxi *
(Xxi * dxa)))) + ((12 * (Xxi * (Xxr * (Xxr * dxa)))) + (( - 12 * (Xxi * (Xxi *
(Xxr * dya)))) + dba)))))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
2630 g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + ((4 * (Xxr * (Xxr * (Xxr * dyb)))) + (( - 4 * (Xxi * (Xxi *
(Xxi * dxb)))) + ((12 * (Xxi * (Xxr * (Xxr * dxb)))) + (( - 12 * (Xxi * (Xxi *
(Xxr * dyb)))) + dbb)))))))));
      </derivative>
2635   </formula>
  </group>

  <group type="46" name="TheRedshiftRider 5: a * z^2 - z^4 + c">
    <formula power="4" glitch="0.0001">
2640   <reference t="C">
        Xn = A * (X^2) - (X^4) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + x2) - 4 * (X^3) * x - 6 * (X^2) * x2 - 4 * X * x2
2645 * x - (x2^2) + c;
      </perturbation>
      <derivative t='M'>
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
2650 g_FactorAR))) + (( - 4 * (Xxi * (Xxi * (Xxi * dya)))) + (( - 4 * (Xxr * (Xxr *
(Xxr * dxa)))) + ((12 * (Xxi * (Xxi * (Xxr * dxa)))) + ((12 * (Xxi * (Xxr *
(Xxr * dya)))) + daa)))))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
2655 g_FactorAR))) + (( - 4 * (Xxi * (Xxi * (Xxi * dyb)))) + (( - 4 * (Xxr * (Xxr *
(Xxr * dxb)))) + ((12 * (Xxi * (Xxi * (Xxr * dxb)))) + ((12 * (Xxi * (Xxr *
(Xxr * dyb)))) + dab)))))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
2660 g_FactorAR))) + (( - 4 * (Xxr * (Xxr * (Xxr * dya)))) + ((4 * (Xxi * (Xxi *
(Xxi * dxa)))) + (( - 12 * (Xxi * (Xxr * (Xxr * dxa)))) + ((12 * (Xxi * (Xxi *
(Xxr * dya)))) + dba)))))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
2665 g_FactorAR))) + (( - 4 * (Xxr * (Xxr * (Xxr * dyb)))) + ((4 * (Xxi * (Xxi *
(Xxi * dxb)))) + (( - 12 * (Xxi * (Xxr * (Xxr * dxb)))) + ((12 * (Xxi * (Xxi *
(Xxr * dyb)))) + dbb)))))))));
```

```
      </derivative>
    </formula>
  </group>

  <group type="47" name="TheRedshiftRider 6: a * z^2 + z^5 + c">
    <formula power="5" glitch="0.0001">
      <reference t="C">
        Xn = A * (X^2) + (X^5) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + x2) + 5 * (X^4) * x + 10 * (X^3) * x2 + 10 *
(X^2) * x2 * x + 5 * X * (x2^2) + (x2 * x2 * x) + c;
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dxa))))) + ((5 * (Xxr * (Xxr
* (Xxr * (Xxr * dxa))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dya))))) +
((20 * (Xxi * (Xxi * (Xxi * (Xxr * dya))))) + (( - 30 * (Xxi * (Xxi * (Xxr *
(Xxr * dxa))))) + daa)))))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dxb))))) + ((5 * (Xxr * (Xxr
* (Xxr * (Xxr * dxb))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dyb))))) +
((20 * (Xxi * (Xxi * (Xxi * (Xxr * dyb))))) + (( - 30 * (Xxi * (Xxi * (Xxr *
(Xxr * dxb))))) + dab)))))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + ((5 * (Xxr * (Xxr
* (Xxr * (Xxr * dya))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dxa))))) + (( -
20 * (Xxi * (Xxi * (Xxi * (Xxr * dxa))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr
* dya))))) + dba)))))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + ((5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + ((5 * (Xxr * (Xxr
* (Xxr * (Xxr * dyb))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dxb))))) + (( -
20 * (Xxi * (Xxi * (Xxi * (Xxr * dxb))))) + (( - 30 * (Xxi * (Xxi * (Xxr * (Xxr
* dyb))))) + dbb)))))))));
      </derivative>
    </formula>
  </group>

  <group type="48" name="TheRedshiftRider 7: a * z^2 - z^5 + c">
    <formula power="5" glitch="0.0001">
      <reference t="C">
        Xn = A * (X^2) - (X^5) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + x2) - 5 * (X^4) * x - 10 * (X^3) * x2 - 10 *
(X^2) * x2 * x - 5 * X * (x2^2) - (x2 * x2 * x) + c;
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dxa))))) + (( - 5 * (Xxr *
(Xxr * (Xxr * (Xxr * dxa))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dya))))) +
(( - 20 * (Xxi * (Xxi * (Xxi * (Xxr * dya))))) + ((30 * (Xxi * (Xxi * (Xxr *
```

```
(Xxr * dxa))))) + daa)))))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dxb))))) + (( - 5 * (Xxr *
(Xxr * (Xxr * (Xxr * dxb))))) + ((20 * (Xxi * (Xxr * (Xxr * (Xxr * dyb))))) +
(( - 20 * (Xxi * (Xxi * (Xxi * (Xxr * dyb))))) + ((30 * (Xxi * (Xxi * (Xxr *
(Xxr * dxb))))) + dab)))))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dya))))) + (( - 5 * (Xxr *
(Xxr * (Xxr * (Xxr * dya))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dxa)))))
+ ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dxa))))) + ((30 * (Xxi * (Xxi * (Xxr *
(Xxr * dya))))) + dba)))))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + (( - 5 * (Xxi * (Xxi * (Xxi * (Xxi * dyb))))) + (( - 5 * (Xxr *
(Xxr * (Xxr * (Xxr * dyb))))) + (( - 20 * (Xxi * (Xxr * (Xxr * (Xxr * dxb)))))
+ ((20 * (Xxi * (Xxi * (Xxi * (Xxr * dxb))))) + ((30 * (Xxi * (Xxi * (Xxr *
(Xxr * dyb))))) + dbb)))))))));
      </derivative>
    </formula>
  </group>

  <group type="49" name="TheRedshiftRider 8: a * z^2 + z^6 + c">
    <formula power="6" glitch="0.0001">
      <reference t="ℂ">
        Xn = A * (X^2) + (X^6) + C;
      </reference>
      <perturbation t="ℂ">
        xn = A * (2 * X * x + x2) + 6 * (X^5) * x + 15 * (X^4) * x2 + 20 *
(X^3) * (x2 * x) + 15 * (X^2) * (x2^2) + 6 * X * (x2 * x2 * x) + (x2^3) + c;
      </perturbation>
      <derivative t="𝕄">
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + ((6 * (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * dxa)))))) + (( - 6 *
(Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dya)))))) + (( - 30 * (Xxi * (Xxr * (Xxr *
(Xxr * (Xxr * dya)))))) + ((30 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * dxa)))))) +
((60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dya)))))) + (( - 60 * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * dxa)))))) + daa)))))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + ((6 * (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * dxb)))))) + (( - 6 *
(Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dyb)))))) + (( - 30 * (Xxi * (Xxr * (Xxr *
(Xxr * (Xxr * dyb)))))) + ((30 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * dxb)))))) +
((60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dyb)))))) + (( - 60 * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * dxb)))))) + dab)))))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + ((6 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dxa)))))) + ((6 * (Xxr
* (Xxr * (Xxr * (Xxr * (Xxr * dya)))))) + ((30 * (Xxi * (Xxi * (Xxi * (Xxi *
(Xxr * dya)))))) + ((30 * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * dxa)))))) + (( -
60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dxa)))))) + (( - 60 * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * dya)))))) + dba)))))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + ((6 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dxb)))))) + ((6 * (Xxr
```

```
* (Xxr * (Xxr * (Xxr * (Xxr * dyb)))))) + ((30 * (Xxi * (Xxi * (Xxi * (Xxi *
(Xxr * dyb)))))) + ((30 * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr * dxb)))))) + (( -
60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dxb)))))) + (( - 60 * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * dyb)))))) + dbb)))))))))));
      </derivative>
    </formula>
  </group>

  <group type="50" name="TheRedshiftRider 9: a * z^2 - z^6 + c">
    <formula power="6" glitch="0.0001">
      <reference t="C">
        Xn = A * (X^2) - (X^6) + C;
      </reference>
      <perturbation t="C">
        xn = A * (2 * X * x + (x^2)) - 6 * (X^5) * x - 15 * (X^4) * (x^2) - 20
* (X^3) * (x^3) - 15 * (X^2) * (x^4) - 6 * X * (x^5) - (x^6) + c;
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxr * (dya * g_FactorAI))) + ((2 * (Xxr * (dxa *
g_FactorAR))) + (( - 2 * (Xxi * (dxa * g_FactorAI))) + (( - 2 * (Xxi * (dya *
g_FactorAR))) + (( - 6 * (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * dxa)))))) + ((6 *
(Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dya)))))) + ((30 * (Xxi * (Xxr * (Xxr * (Xxr
* (Xxr * dya)))))) + (( - 30 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * dxa)))))) +
(( - 60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dya)))))) + ((60 * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * dxa)))))) + daa)))))))))));
        dxbn = (( - 2 * (Xxr * (dyb * g_FactorAI))) + ((2 * (Xxr * (dxb *
g_FactorAR))) + (( - 2 * (Xxi * (dxb * g_FactorAI))) + (( - 2 * (Xxi * (dyb *
g_FactorAR))) + (( - 6 * (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * dxb)))))) + ((6 *
(Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dyb)))))) + ((30 * (Xxi * (Xxr * (Xxr * (Xxr
* (Xxr * dyb)))))) + (( - 30 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxr * dxb)))))) +
(( - 60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dyb)))))) + ((60 * (Xxi * (Xxi *
(Xxr * (Xxr * (Xxr * dxb)))))) + dab)))))))))));
        dyan = ((2 * (Xxr * (dxa * g_FactorAI))) + ((2 * (Xxr * (dya *
g_FactorAR))) + (( - 2 * (Xxi * (dya * g_FactorAI))) + ((2 * (Xxi * (dxa *
g_FactorAR))) + (( - 6 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dxa)))))) + (( - 6
* (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * dya)))))) + (( - 30 * (Xxi * (Xxi * (Xxi *
(Xxi * (Xxr * dya)))))) + (( - 30 * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr *
dxa)))))) + ((60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dxa)))))) + ((60 * (Xxi *
(Xxi * (Xxr * (Xxr * (Xxr * dya)))))) + dba)))))))))));
        dybn = ((2 * (Xxr * (dxb * g_FactorAI))) + ((2 * (Xxr * (dyb *
g_FactorAR))) + (( - 2 * (Xxi * (dyb * g_FactorAI))) + ((2 * (Xxi * (dxb *
g_FactorAR))) + (( - 6 * (Xxi * (Xxi * (Xxi * (Xxi * (Xxi * dxb)))))) + (( - 6
* (Xxr * (Xxr * (Xxr * (Xxr * (Xxr * dyb)))))) + (( - 30 * (Xxi * (Xxi * (Xxi *
(Xxi * (Xxr * dyb)))))) + (( - 30 * (Xxi * (Xxr * (Xxr * (Xxr * (Xxr *
dxb)))))) + ((60 * (Xxi * (Xxi * (Xxi * (Xxr * (Xxr * dxb)))))) + ((60 * (Xxi *
(Xxi * (Xxr * (Xxr * (Xxr * dyb)))))) + dbb)))))))))));
      </derivative>
    </formula>
  </group>

  <group type="51" name="SimonBrot2 4th">
    <formula power="4" glitch="0.0001">
      <reference t="R">
        Xrn = (Xr2 - Xi2) * abs(Xr2 - Xi2) - 2 * Xr * Xi * abs(2 * Xr * Xi) +
Cr;
        Xin = (Xr2 - Xi2) * abs(2 * Xr * Xi) + 2 * Xr * Xi * abs(Xr2 - Xi2) +
Ci;
```

```
      </reference>
      <perturbation t="R">
        xrn = (Xr2 - Xi2) * diffabs(Xr2 - Xi2,2 * Xr * xr + xr2 - 2 * Xi * xi -
    xi2) + (2 * Xr * xr + xr2 - 2 * Xi * xi - xi2) * abs(Xr2 + 2 * Xr * xr + xr2 -
    Xi2 - 2 * Xi * xi - xi2)  - (2 * Xr * Xi) * diffabs(2 * Xr * Xi,2 * Xr * xi +
    2 * xr * Xi + 2 * xr * xi) - (2 * Xr * xi + 2 * xr * Xi + 2 * xr * xi) * abs(2
    * Xr * Xi + 2 * Xr * xi + 2 * xr * Xi + 2 * xr * xi)  + cr;
        xin = (Xr2 - Xi2) * diffabs(2 * Xr * Xi,2 * Xr * xi + 2 * xr * Xi + 2 *
    xr * xi) + (2 * Xr * xr + xr2 - 2 * Xi * xi - xi2) * abs(2 * Xr * Xi + 2 * Xr *
    xi + 2 * xr * Xi + 2 * xr * xi)  + (2 * Xr * Xi) * diffabs(Xr2 - Xi2,2 * Xr *
    xr + xr2 - 2 * Xi * xi - xi2) + (2 * Xr * xi + 2 * xr * Xi + 2 * xr * xi) *
    abs(Xr2 - Xi2 + 2 * Xr * xr + xr2 - 2 * Xi * xi - xi2)  + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxr * (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) +
    (Xxr * Xxr)))))))) + ((2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr *
    dxa))) + (( - 2 * (Xxi * (Xxi * (Xxr * (dxa * sgn((( - 1 * (Xxi * Xxi)) + (Xxr
    * Xxr)))))))) + (( - 2 * (Xxi * (Xxr * (Xxr * (dya * sgn((( - 1 * (Xxi * Xxi))
    + (Xxr * Xxr)))))))) + (( - 2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) *
    (Xxi * dya))) + ((2 * (Xxi * (Xxi * (Xxi * (dya * sgn((( - 1 * (Xxi * Xxi)) +
    (Xxr * Xxr)))))))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxi * dxa))) + (( - 4 *
    (abs((Xxi * Xxr)) * (Xxr * dya))) + (( - 4 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr *
    (dxa * sgn(Xxr)))))))) + (( - 4 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dya *
    sgn(Xxr)))))) + daa)))))))));
        dxbn = ((2 * (Xxr * (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) +
    (Xxr * Xxr)))))))) + ((2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr *
    dxb))) + (( - 2 * (Xxi * (Xxi * (Xxr * (dxb * sgn((( - 1 * (Xxi * Xxi)) + (Xxr
    * Xxr)))))))) + (( - 2 * (Xxi * (Xxr * (Xxr * (dyb * sgn((( - 1 * (Xxi * Xxi))
    + (Xxr * Xxr)))))))) + (( - 2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) *
    (Xxi * dyb))) + ((2 * (Xxi * (Xxi * (Xxi * (dyb * sgn((( - 1 * (Xxi * Xxi)) +
    (Xxr * Xxr)))))))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxi * dxb))) + (( - 4 *
    (abs((Xxi * Xxr)) * (Xxr * dyb))) + (( - 4 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr *
    (dxb * sgn(Xxr)))))))) + (( - 4 * (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dyb *
    sgn(Xxr)))))) + dab)))))))));
        dyan = ((2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi * dxa)))
    + ((2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dya))) + ((2 *
    (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dxa * sgn(Xxr)))))))) + ((2 * (sgn(Xxi) * (Xxr
    * (Xxr * (Xxr * (dya * sgn(Xxr)))))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxi
    * (dxa * sgn(Xxr)))))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dya *
    sgn(Xxr)))))))) + ((4 * (abs((Xxi * Xxr)) * (Xxr * dxa))) + (( - 4 * (abs((Xxi *
    Xxr)) * (Xxi * dya))) + ((4 * (Xxi * (Xxr * (Xxr * (dxa * sgn((( - 1 * (Xxi *
    Xxi)) + (Xxr * Xxr)))))))) + (( - 4 * (Xxi * (Xxi * (Xxr * (dya * sgn((( - 1 *
    (Xxi * Xxi)) + (Xxr * Xxr)))))))) + dba)))))))));
        dybn = ((2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxi * dxb)))
    + ((2 * (abs((( - 1 * (Xxi * Xxi)) + (Xxr * Xxr))) * (Xxr * dyb))) + ((2 *
    (sgn(Xxi) * (Xxi * (Xxr * (Xxr * (dxb * sgn(Xxr)))))))) + ((2 * (sgn(Xxi) * (Xxr
    * (Xxr * (Xxr * (dyb * sgn(Xxr)))))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxi
    * (dxb * sgn(Xxr)))))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxi * (Xxr * (dyb *
    sgn(Xxr)))))))) + ((4 * (abs((Xxi * Xxr)) * (Xxr * dxb))) + (( - 4 * (abs((Xxi *
    Xxr)) * (Xxi * dyb))) + ((4 * (Xxi * (Xxr * (Xxr * (dxb * sgn((( - 1 * (Xxi *
    Xxi)) + (Xxr * Xxr)))))))) + (( - 4 * (Xxi * (Xxi * (Xxr * (dyb * sgn((( - 1 *
    (Xxi * Xxi)) + (Xxr * Xxr)))))))) + dbb)))))))));
      </derivative>
    </formula>
  </group>

  <group type="52" name="General Quadratic Minus">
```

```
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = Xr2 - Xi2 + Cr;
        Xin = 2 * Ar * Xr * Xi + Ai * Xr2 + Ci;
      </reference>
      <perturbation t="R">
        xrn = 2 * xr * Xr + xr * xr - 2 * Xi * xi - xi * xi + cr;
        xin = 2 * Ar * (Xr * xi + xr * Xi + xr * xi) + Ai * (2 * xr * Xr + xr * xr) + ci;
      </perturbation>
      <derivative t="M">
        dxan = (( - 2 * (Xxi * dya)) + (2 * (Xxr * dxa))) + daa;
        dxbn = (( - 2 * (Xxi * dyb)) + (2 * (Xxr * dxb))) + dab;
        dyan = ((2 * (Xxi * (dxa * g_FactorAR))) + ((2 * (Xxr * (dxa * g_FactorAI))) + (2 * (Xxr * (dya * g_FactorAR))))) + dba;
        dybn = ((2 * (Xxi * (dxb * g_FactorAR))) + ((2 * (Xxr * (dxb * g_FactorAI))) + (2 * (Xxr * (dyb * g_FactorAR))))) + dbb;
      </derivative>
    </formula>
  </group>

  <group type="53" name="General Quadratic Plus">
    <formula power="2" glitch="0.0000001">
      <reference t="R">
        Xrn = Xr2 + Xi2 + Cr;
        Xin = 2 * Ar * Xr * Xi + Ai * Xr2 + Ci;
      </reference>
      <perturbation t="R">
        xrn = 2 * xr * Xr + xr * xr + 2 * Xi * xi + xi * xi + cr;
        xin = 2 * Ar * (Xr * xi + xr * Xi + xr * xi) + Ai * (2 * xr * Xr + xr * xr) + ci;
      </perturbation>
      <derivative t="M">
        dxan = ((2 * (Xxi * dya)) + (2 * (Xxr * dxa))) + daa;
        dxbn = ((2 * (Xxi * dyb)) + (2 * (Xxr * dxb))) + dab;
        dyan = ((2 * (Xxi * (dxa * g_FactorAR))) + ((2 * (Xxr * (dxa * g_FactorAI))) + (2 * (Xxr * (dya * g_FactorAR))))) + dba;
        dybn = ((2 * (Xxi * (dxb * g_FactorAR))) + ((2 * (Xxr * (dxb * g_FactorAI))) + (2 * (Xxr * (dyb * g_FactorAR))))) + dbb;
      </derivative>
    </formula>
  </group>

  <group type="54" name="Mothbrot 2nd 1x1">
    <formula power="2" glitch="1.0000000000000004e-7">
      <reference t='R'>
        Xrn = (Cr + (( - 1 * (Xi * abs(Xi))) + (Xr * abs(Xr))));
        Xin = (Ci + ((Xi * abs(Xr)) + (Xr * abs(Xi))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((Xr * diffabs(Xr,xr)) + (( - 1 * ((Xi * diffabs(Xi,xi)) + (xi * abs((Xi + xi))))) + (xr * abs((Xr + xr))))));
        xin = (ci + ((Xi * diffabs(Xr,xr)) + ((Xr * diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))));
      </perturbation>
      <derivative t='M'>
        dxan = (daa + ((Xxr * (dxa * sgn(Xxr))) + ((dxa * abs(Xxr)) + ( - 1 *
```

```
((Xxi * (dya * sgn(Xxi))) + (dya * abs(Xxi)))))));
        dxbn = (dab + ((Xxr * (dxb * sgn(Xxr))) + ((dxb * abs(Xxr)) + ( - 1 *
((Xxi * (dyb * sgn(Xxi))) + (dyb * abs(Xxi)))))));
        dyan = (dba + ((Xxi * (dxa * sgn(Xxr))) + ((Xxr * (dya * sgn(Xxi))) +
((dxa * abs(Xxi)) + (dya * abs(Xxr))))));
        dybn = (dbb + ((Xxi * (dxb * sgn(Xxr))) + ((Xxr * (dyb * sgn(Xxi))) +
((dxb * abs(Xxi)) + (dyb * abs(Xxr))))));
      </derivative>
    </formula>
  </group>

  <group type="55" name="Mothbrot 3rd 1x2">
    <formula power="3" glitch="1.0000000000000004e-6">
      <reference t='R'>
        Xrn = (Cr + (( - 1 * (abs(Xi) * (Xr * abs(Xi)))) + (( - 2 * (abs(Xi) *
(Xi * abs(Xr)))) + (abs(Xr) * (Xr * abs(Xr))))));
        Xin = (Ci + ((abs(Xr) * (Xi * abs(Xr))) + (( - 1 * (abs(Xi) * (Xi *
abs(Xi)))) + (2 * (abs(Xi) * (Xr * abs(Xr)))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs(Xr) * ((Xr * diffabs(Xr,xr)) + (xr * abs((Xr +
xr))))) + ((abs((Xr + xr)) * (Xr * diffabs(Xr,xr))) + (( - 1 * ((abs(Xi) * ((Xr
* diffabs(Xi,xi)) + (xr * abs((Xi + xi))))) + (abs((Xi + xi)) * (diffabs(Xi,xi)
* (Xr + xr))))) + (( - 2 * ((abs(Xi) * ((Xi * diffabs(Xr,xr)) + (xi * abs((Xr +
xr))))) + (abs((Xr + xr)) * (diffabs(Xi,xi) * (Xi + xi))))) + (abs((Xr + xr)) *
(xr * diffabs(Xr,xr)))))))));
        xin = (ci + ((abs(Xr) * ((Xi * diffabs(Xr,xr)) + (xi * abs((Xr +
xr))))) + ((abs((Xr + xr)) * (diffabs(Xr,xr) * (Xi + xi))) + (( - 1 * ((abs(Xi)
* ((Xi * diffabs(Xi,xi)) + (xi * abs((Xi + xi))))) + (abs((Xi + xi)) *
(diffabs(Xi,xi) * (Xi + xi))))) + (2 * ((abs(Xi) * ((Xr * diffabs(Xr,xr)) + (xr
* abs((Xr + xr))))) + (abs((Xr + xr)) * (diffabs(Xi,xi) * (Xr + xr)))))))));
      </perturbation>
      <derivative t='M'>
        dxan = (daa + ((abs(Xxr) * (dxa * abs(Xxr))) + (( - 1 * (abs(Xxi) *
(dxa * abs(Xxi)))) + (( - 2 * (sgn(Xxi) * ((Xxi * (dya * abs(Xxr))) + (Xxr *
(dya * abs(Xxi)))))) + ((2 * (sgn(Xxr) * (Xxr * (dxa * abs(Xxr))))) + ( - 2 *
((sgn(Xxr) * (Xxi * (dxa * abs(Xxi)))) + (abs(Xxi) * (dya * abs(Xxr)))))))));
        dxbn = (dab + ((abs(Xxr) * (dxb * abs(Xxr))) + (( - 1 * (abs(Xxi) *
(dxb * abs(Xxi)))) + (( - 2 * (sgn(Xxi) * ((Xxi * (dyb * abs(Xxr))) + (Xxr *
(dyb * abs(Xxi)))))) + ((2 * (sgn(Xxr) * (Xxr * (dxb * abs(Xxr))))) + ( - 2 *
((sgn(Xxr) * (Xxi * (dxb * abs(Xxi)))) + (abs(Xxi) * (dyb * abs(Xxr)))))))));
        dyan = (dba + (( - 1 * (abs(Xxi) * (dya * abs(Xxi)))) + ((2 *
((abs(Xxi) * (dxa * abs(Xxr))) + ((sgn(Xxi) * (Xxr * (dya * abs(Xxr)))) +
(sgn(Xxr) * ((Xxi * (dxa * abs(Xxr))) + (Xxr * (dxa * abs(Xxi)))))))) + (( - 2
* (sgn(Xxi) * (Xxi * (dya * abs(Xxi))))) + (abs(Xxr) * (dya * abs(Xxr)))))));
        dybn = (dbb + (( - 1 * (abs(Xxi) * (dyb * abs(Xxi)))) + ((2 *
((abs(Xxi) * (dxb * abs(Xxr))) + ((sgn(Xxi) * (Xxr * (dyb * abs(Xxr)))) +
(sgn(Xxr) * ((Xxi * (dxb * abs(Xxr))) + (Xxr * (dxb * abs(Xxi)))))))) + (( - 2
* (sgn(Xxi) * (Xxi * (dyb * abs(Xxi))))) + (abs(Xxr) * (dyb * abs(Xxr)))))));
      </derivative>
    </formula>
  </group>

  <group type="56" name="Mothbrot 3rd 2x1">
    <formula power="3" glitch="1.0000000000000004e-6">
      <reference t='R'>
        Xrn = (Cr + (( - 1 * ((Xi * Xi) * abs(Xr))) + (( - 2 * (Xi * (Xr *
```

```
abs(Xi)))) + ((Xr * Xr) * abs(Xr)))));
        Xin = (Ci + (( - 1 * ((Xi * Xi) * abs(Xi))) + ((2 * (Xi * (Xr *
abs(Xr)))) + ((Xr * Xr) * abs(Xi)))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + (((Xr * Xr) * diffabs(Xr,xr)) + (( - 1 * (((Xi * Xi) *
diffabs(Xr,xr)) + ((xi * xi) * abs((Xr + xr))))) + (( - 2 * (xi * (xr * abs((Xi
+ xi))))) + ((2 * (Xr * (xr * abs((Xr + xr))))) + (( - 2 * ((Xi * ((Xr *
diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))) + (Xr * (xi
* abs((Xi + xi)))))) + ((xr * xr) * abs((Xr + xr)))))))));
        xin = (ci + (((Xr * Xr) * diffabs(Xi,xi)) + (( - 1 * (((Xi * Xi) *
diffabs(Xi,xi)) + ((xi * xi) * abs((Xi + xi))))) + ((2 * ((Xi * ((Xr *
diffabs(Xr,xr)) + (xr * abs((Xr + xr))))) + ((Xr * ((xi * abs((Xr + xr))) + (xr
* abs((Xi + xi))))) + (xi * (xr * abs((Xr + xr)))))))) + (( - 2 * (Xi * (xi *
abs((Xi + xi))))) + ((xr * xr) * abs((Xi + xi)))))))));
      </perturbation>
      <derivative t='M'>
        dxan = (((Xxr * Xxr) * (dxa * sgn(Xxr))) + (( - 1 * ((Xxi * Xxi) * (dxa
* sgn(Xxr)))) + (( - 2 * (Xxr * (dya * abs(Xxi)))) + ((2 * (Xxr * (dxa *
abs(Xxr)))) + (( - 2 * (Xxi * ((Xxr * (dya * sgn(Xxi))) + ((dxa * abs(Xxi)) +
(dya * abs(Xxr)))))) + daa)))));
        dxbn = (((Xxr * Xxr) * (dxb * sgn(Xxr))) + (( - 1 * ((Xxi * Xxi) * (dxb
* sgn(Xxr)))) + (( - 2 * (Xxr * (dyb * abs(Xxi)))) + ((2 * (Xxr * (dxb *
abs(Xxr)))) + (( - 2 * (Xxi * ((Xxr * (dyb * sgn(Xxi))) + ((dxb * abs(Xxi)) +
(dyb * abs(Xxr)))))) + dab)))));
        dyan = (((Xxr * Xxr) * (dya * sgn(Xxi))) + (( - 1 * ((Xxi * Xxi) * (dya
* sgn(Xxi)))) + ((2 * (Xxr * ((dxa * abs(Xxi)) + (dya * abs(Xxr))))) + (( - 2 *
(Xxi * (dya * abs(Xxi)))) + ((2 * (Xxi * ((Xxr * (dxa * sgn(Xxr))) + (dxa *
abs(Xxr))))) + dba)))));
        dybn = (((Xxr * Xxr) * (dyb * sgn(Xxi))) + (( - 1 * ((Xxi * Xxi) * (dyb
* sgn(Xxi)))) + ((2 * (Xxr * ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))) + (( - 2 *
(Xxi * (dyb * abs(Xxi)))) + ((2 * (Xxi * ((Xxr * (dxb * sgn(Xxr))) + (dxb *
abs(Xxr))))) + dbb)))));
      </derivative>
    </formula>
  </group>

  <group type="57" name="Mothbrot 4th 1x3">
    <formula power="4" glitch="1.0000000000000003e-5">
      <reference t='R'>
        Xrn = (Cr + ((abs((Xi * Xi)) * (Xi * abs(Xi))) + (( - 1 * ((abs((Xi *
Xi)) * (Xr * abs(Xr))) + (abs((Xr * Xr)) * (Xi * abs(Xi))))) + (( - 2 *
(abs((Xi * Xr)) * ((Xi * abs(Xr)) + (Xr * abs(Xi))))) + (abs((Xr * Xr)) * (Xr *
abs(Xr)))))));
        Xin = (Ci + ((abs((Xr * Xr)) * (Xi * abs(Xr))) + (( - 1 * (abs((Xi *
Xi)) * ((Xi * abs(Xr)) + (Xr * abs(Xi))))) + ((2 * (abs((Xi * Xr)) * (Xr *
abs(Xr)))) + (( - 2 * (abs((Xi * Xr)) * (Xi * abs(Xi)))) + (abs((Xr * Xr)) *
(Xr * abs(Xi)))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs((Xi * Xi)) * ((Xi * diffabs(Xi,xi)) + (xi * abs((Xi +
xi))))) + ((abs((Xr * Xr)) * (Xr * diffabs(Xr,xr))) + ((abs((Xi + xi)) *
(diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))) * (Xi + xi))) + ((abs((Xr +
xr)) * (diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))) * (Xr + xr))) + (( - 1
* ((abs((Xi * Xi)) * ((Xr * diffabs(Xr,xr)) + (xr * abs((Xr + xr))))) +
((abs((Xr * Xr)) * ((Xi * diffabs(Xi,xi)) + (xi * abs((Xi + xi))))) + ((abs((Xi
+ xi)) * (diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))) * (Xi + xi))) +
```

```
(abs((Xr + xr)) * (diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))) * (Xr +
xr))))))) + (( - 2 * ((abs((Xi * Xr)) * ((Xi * diffabs(Xr,xr)) + ((Xr *
diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))))) +
((abs((Xi + xi)) * (diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))) * (Xr +
xr))) + (abs((Xr + xr)) * (diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))) *
(Xi + xi)))))) + (abs((Xr * Xr)) * (xr * abs((Xr + xr)))))))))));
        xin = (ci + ((abs((Xr * Xr)) * ((Xi * diffabs(Xr,xr)) + ((Xr *
diffabs(Xi,xi)) + (xi * abs((Xr + xr)))))) + ((abs((Xi + xi)) * (diffabs((Xr *
Xr),((2 * (Xr * xr)) + (xr * xr))) * (Xr + xr))) + ((abs((Xr + xr)) *
(diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))) * (Xi + xi))) + (( - 1 *
((abs((Xi * Xi)) * ((Xi * diffabs(Xr,xr)) + ((Xr * diffabs(Xi,xi)) + ((xi *
abs((Xr + xr))) + (xr * abs((Xi + xi)))))))) + ((abs((Xi + xi)) * (diffabs((Xi *
Xi),((2 * (Xi * xi)) + (xi * xi))) * (Xr + xr))) + (abs((Xr + xr)) *
(diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))) * (Xi + xi)))))) + ((2 *
((abs((Xi * Xr)) * (xr * abs((Xr + xr)))) + (abs((Xr + xr)) * (xr * diffabs((Xi
* Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + (( - 2 * ((abs((Xi * Xr)) * (xi *
abs((Xi + xi)))) + (abs((Xi + xi)) * (xi * diffabs((Xi * Xr),((Xi * xr) + (xi *
(Xr + xr))))))))) + ((2 * ((abs((Xi * Xr)) * (Xr * diffabs(Xr,xr))) + (abs((Xr +
xr)) * (Xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + (( - 2 *
((abs((Xi * Xr)) * (Xi * diffabs(Xi,xi))) + (abs((Xi + xi)) * (Xi * diffabs((Xi
* Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + (abs((Xr * Xr)) * (xr * abs((Xi +
xi)))))))))))));
      </perturbation>
      <derivative t='M'>
        dxan = (daa + ((abs((Xxi * Xxi)) * (dya * abs(Xxi))) + ((abs((Xxr *
Xxr)) * (dxa * abs(Xxr))) + ((sgn(Xxi) * (Xxi * (dya * abs((Xxi * Xxi))))) +
((sgn(Xxr) * (Xxr * (dxa * abs((Xxr * Xxr))))) + (( - 1 * ((abs((Xxi * Xxi)) *
(dxa * abs(Xxr))) + ((sgn(Xxi) * (Xxi * (dya * abs((Xxr * Xxr))))) + (sgn(Xxr)
* (Xxr * (dxa * abs((Xxi * Xxi)))))))) + (( - 2 * ((abs((Xxi * Xxr)) * ((dxa *
abs(Xxi)) + (dya * abs(Xxr)))) + (sgn(Xxi) * ((Xxr * (dya * abs((Xxi * Xxr))))
+ (sgn(Xxr) * ((Xxr * Xxr) * (dya * abs(Xxi)))))))) + ((2 * ((sgn(Xxr) *
sgn(Xxr)) * ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 2 * (((sgn(Xxi) *
sgn(Xxi)) * (Xxi * (Xxr * (dya * abs(Xxr))))) + ((sgn(Xxi) * (sgn(Xxr) * (((Xxi
* Xxi) * (dxa * abs(Xxr))) + (Xxi * (Xxr * ((dxa * abs(Xxi)) + (dya *
abs(Xxr)))))))) + ((sgn(Xxr) * (Xxi * (dxa * abs((Xxi * Xxr))))) + ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * (dxa * abs(Xxi)))))))) + ((2 * ((sgn(Xxi) *
sgn(Xxi)) * ((Xxi * Xxi) * (dya * abs(Xxi))))) + ( - 1 * (abs((Xxr * Xxr)) *
(dya * abs(Xxi))))))))))))));
        dxbn = (dab + ((abs((Xxi * Xxi)) * (dyb * abs(Xxi))) + ((abs((Xxr *
Xxr)) * (dxb * abs(Xxr))) + ((sgn(Xxi) * (Xxi * (dyb * abs((Xxi * Xxi))))) +
((sgn(Xxr) * (Xxr * (dxb * abs((Xxr * Xxr))))) + (( - 1 * ((abs((Xxi * Xxi)) *
(dxb * abs(Xxr))) + ((sgn(Xxi) * (Xxi * (dyb * abs((Xxr * Xxr))))) + (sgn(Xxr)
* (Xxr * (dxb * abs((Xxi * Xxi)))))))) + (( - 2 * ((abs((Xxi * Xxr)) * ((dxb *
abs(Xxi)) + (dyb * abs(Xxr)))) + (sgn(Xxi) * ((Xxr * (dyb * abs((Xxi * Xxr))))
+ (sgn(Xxr) * ((Xxr * Xxr) * (dyb * abs(Xxi)))))))) + ((2 * ((sgn(Xxr) *
sgn(Xxr)) * ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 2 * (((sgn(Xxi) *
sgn(Xxi)) * (Xxi * (Xxr * (dyb * abs(Xxr))))) + ((sgn(Xxi) * (sgn(Xxr) * (((Xxi
* Xxi) * (dxb * abs(Xxr))) + (Xxi * (Xxr * ((dxb * abs(Xxi)) + (dyb *
abs(Xxr)))))))) + ((sgn(Xxr) * (Xxi * (dxb * abs((Xxi * Xxr))))) + ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * (dxb * abs(Xxi)))))))) + ((2 * ((sgn(Xxi) *
sgn(Xxi)) * ((Xxi * Xxi) * (dyb * abs(Xxi))))) + ( - 1 * (abs((Xxr * Xxr)) *
(dyb * abs(Xxi))))))))))))));
        dyan = (dba + ((abs((Xxr * Xxr)) * (dxa * abs(Xxi))) + ((sgn(Xxi) *
(Xxr * (dya * abs((Xxr * Xxr))))) + ((sgn(Xxr) * (Xxi * (dxa * abs((Xxr *
Xxr))))) + (( - 1 * ((abs((Xxi * Xxi)) * ((dxa * abs(Xxi)) + (dya * abs(Xxr))))
+ ((sgn(Xxi) * (Xxr * (dya * abs((Xxi * Xxi))))) + (sgn(Xxr) * (Xxi * (dxa *
abs((Xxi * Xxi)))))))) + (( - 2 * (abs((Xxi * Xxr)) * (dya * abs(Xxi)))) + ((2
```

```
* ((abs((Xxi * Xxr)) * (dxa * abs(Xxr))) + ((sgn(Xxi) * (sgn(Xxr) * ((Xxr *
3125 Xxr) * (dya * abs(Xxr))))) + (sgn(Xxr) * (Xxr * (dxa * abs((Xxi * Xxr)))))))) +
(( - 2 * ((sgn(Xxi) * (Xxi * (dya * abs((Xxi * Xxr))))) + (((sgn(Xxi) *
sgn(Xxi)) * (Xxi * (Xxr * (dya * abs(Xxi))))) + (sgn(Xxi) * (sgn(Xxr) * (Xxi *
(Xxr * (dya * abs(Xxi)))))))) + ((2 * ((sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr *
(dxa * abs(Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * (Xxr * (dxa *
3130 abs(Xxr)))) + ((Xxr * Xxr) * (dxa * abs(Xxi))))))) + (( - 2 * (((sgn(Xxi) *
sgn(Xxi)) * ((Xxi * Xxi) * (dya * abs(Xxr)))) + (sgn(Xxi) * (sgn(Xxr) * ((Xxi *
Xxi) * (dxa * abs(Xxi)))))) + (abs((Xxr * Xxr)) * (dya * abs(Xxr)))))))))))));
        dybn = (dbb + ((abs((Xxr * Xxr)) * (dxb * abs(Xxi))) + ((sgn(Xxi) *
(Xxr * (dyb * abs((Xxr * Xxr))))) + ((sgn(Xxr) * (Xxi * (dxb * abs((Xxr *
3135 Xxr))))) + (( - 1 * ((abs((Xxi * Xxi)) * ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))
+ ((sgn(Xxi) * (Xxr * (dyb * abs((Xxi * Xxi))))) + (sgn(Xxr) * (Xxi * (dxb *
abs((Xxi * Xxi)))))))) + (( - 2 * (abs((Xxi * Xxr)) * (dyb * abs(Xxi)))) + ((2
* ((abs((Xxi * Xxr)) * (dxb * abs(Xxr))) + ((sgn(Xxi) * (sgn(Xxr) * ((Xxr *
Xxr) * (dyb * abs(Xxr))))) + (sgn(Xxr) * (Xxr * (dxb * abs((Xxi * Xxr)))))))) +
3140 (( - 2 * ((sgn(Xxi) * (Xxi * (dyb * abs((Xxi * Xxr))))) + (((sgn(Xxi) *
sgn(Xxi)) * (Xxi * (Xxr * (dyb * abs(Xxi))))) + (sgn(Xxi) * (sgn(Xxr) * (Xxi *
(Xxr * (dyb * abs(Xxi)))))))) + ((2 * ((sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr *
(dxb * abs(Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * (Xxr * (dxb *
abs(Xxr)))) + ((Xxr * Xxr) * (dxb * abs(Xxi))))))) + (( - 2 * (((sgn(Xxi) *
3145 sgn(Xxi)) * ((Xxi * Xxi) * (dyb * abs(Xxr)))) + (sgn(Xxi) * (sgn(Xxr) * ((Xxi *
Xxi) * (dxb * abs(Xxi)))))) + (abs((Xxr * Xxr)) * (dyb * abs(Xxr)))))))))))));
      </derivative>
    </formula>
  </group>
3150

  <group type="58" name="Mothbrot 4th 2x2 (aka SimonBrot 4th)">
    <formula power="4" glitch="1.0000000000000003e-5">
      <reference t='R'>
        Xrn = (Cr + ((abs(Xi) * ((Xi * Xi) * abs(Xi))) + (( - 1 * ((abs(Xi) *
3155 ((Xr * Xr) * abs(Xi))) + (abs(Xr) * ((Xi * Xi) * abs(Xr))))) + (( - 4 *
(abs(Xi) * (Xi * (Xr * abs(Xr))))) + (abs(Xr) * ((Xr * Xr) * abs(Xr)))))));
        Xin = (Ci + (( - 2 * (abs(Xi) * (Xi * (Xr * abs(Xi))))) + ((2 *
(abs(Xr) * (Xi * (Xr * abs(Xr))))) + (( - 2 * (abs(Xi) * ((Xi * Xi) *
abs(Xr)))) + (2 * (abs(Xi) * ((Xr * Xr) * abs(Xr)))))));
3160   </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs(Xi) * (((Xi * Xi) * diffabs(Xi,xi)) + ((xi * xi) *
abs((Xi + xi))))) + ((abs(Xr) * (((Xr * Xr) * diffabs(Xr,xr)) + ((xr * xr) *
abs((Xr + xr))))) + ((abs((Xi + xi)) * (diffabs(Xi,xi) * ((Xi * Xi) + (xi *
3165 xi)))) + ((abs((Xr + xr)) * ((Xr * Xr) * diffabs(Xr,xr))) + (( - 1 * ((abs(Xi)
* (((Xr * Xr) * diffabs(Xi,xi)) + ((xr * xr) * abs((Xi + xi))))) + ((abs(Xr) *
(((Xi * Xi) * diffabs(Xr,xr)) + ((xi * xi) * abs((Xr + xr))))) + ((abs((Xi +
xi)) * (diffabs(Xi,xi) * ((Xr * Xr) + (xr * xr)))) + (abs((Xr + xr)) *
(diffabs(Xr,xr) * ((Xi * Xi) + (xi * xi)))))))) + ((2 * ((abs(Xr) * (Xr * (xr *
3170 abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xr * diffabs(Xr,xr)))))) + (( - 2
* ((abs(Xi) * (Xr * (xr * abs((Xi + xi))))) + ((abs(Xr) * (Xi * (xi * abs((Xr +
xr))))) + ((abs((Xi + xi)) * (Xr * (xr * diffabs(Xi,xi)))) + (abs((Xr + xr)) *
(Xi * (xi * diffabs(Xr,xr))))))))) + ((2 * ((abs(Xi) * (Xi * (xi * abs((Xi +
xi))))) + (abs((Xi + xi)) * (Xi * (xi * diffabs(Xi,xi)))))) + (( - 4 *
3175 ((abs(Xi) * ((Xi * ((Xr * diffabs(Xr,xr)) + (xr * abs((Xr + xr))))) + ((Xr *
(xi * abs((Xr + xr)))) + (xi * (xr * abs((Xr + xr))))))) + (abs((Xr + xr)) *
((Xi * (diffabs(Xi,xi) * (Xr + xr))) + ((Xr * (xi * diffabs(Xi,xi))) + (xi *
(xr * diffabs(Xi,xi))))))) + (abs((Xr + xr)) * ((xr * xr) *
diffabs(Xr,xr)))))))))))));
3180         xin = (ci + ((2 * ((abs(Xi) * ((xr * xr) * abs((Xr + xr)))) + ((abs(Xr)
```

```
* (xi * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (xi * (xr *
diffabs(Xr,xr)))))))) + (( - 2 * ((abs(Xi) * (((xi * xi) * abs((Xr + xr))) + (xi
* (xr * abs((Xi + xi)))))) + ((abs((Xi + xi)) * (xi * (xr * diffabs(Xi,xi)))) +
(abs((Xr + xr)) * ((xi * xi) * diffabs(Xi,xi)))))) + ((2 * ((abs(Xr) * (Xr *
(xi * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xi * diffabs(Xr,xr)))))) +
(( - 2 * ((abs(Xi) * (Xr * (xi * abs((Xi + xi))))) + (abs((Xi + xi)) * (Xr *
(xi * diffabs(Xi,xi)))))) + ((2 * ((abs(Xi) * ((Xr * Xr) * diffabs(Xr,xr))) +
((abs(Xr) * (Xi * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * ((Xi * (xr *
diffabs(Xr,xr))) + ((Xr * Xr) * diffabs(Xi,xi)))))) + (( - 2 * (abs((Xi + xi))
* (Xi * (xr * diffabs(Xi,xi))))) + ((2 * ((Xi * (Xr * diffabs(Xr,xr))) *
(abs(Xr) + abs((Xr + xr))))) + (( - 2 * ((abs(Xi) * (((Xi * Xi) *
diffabs(Xr,xr)) + (Xi * ((Xr * diffabs(Xi,xi)) + (xr * abs((Xi + xi))))))) +
((abs((Xi + xi)) * (Xi * (Xr * diffabs(Xi,xi)))) + (abs((Xr + xr)) * ((Xi * Xi)
* diffabs(Xi,xi)))))) + ((4 * ((abs(Xi) * (Xr * (xr * abs((Xr + xr))))) +
(abs((Xr + xr)) * (Xr * (xr * diffabs(Xi,xi)))))) + (( - 4 * ((abs(Xi) * (Xi *
(xi * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xi * (xi * diffabs(Xi,xi)))))) +
(2 * (abs((Xr + xr)) * ((xr * xr) * diffabs(Xi,xi)))))))))))))));
      </perturbation>
      <derivative t='M'>
        dxan = ((2 * (abs(Xxr) * (Xxr * (dxa * abs(Xxr))))) + (( - 2 *
((abs(Xxi) * (Xxr * (dxa * abs(Xxi)))) + (sgn(Xxi) * ((Xxr * Xxr) * (dya *
abs(Xxi)))))) + ((2 * (sgn(Xxr) * ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 2 *
((abs(Xxr) * (Xxi * (dya * abs(Xxr)))) + (sgn(Xxr) * ((Xxi * Xxi) * (dxa *
abs(Xxr)))))) + ((2 * ((abs(Xxi) * (Xxi * (dya * abs(Xxi)))) + (sgn(Xxi) *
((Xxi * Xxi) * (dya * abs(Xxi)))))) + (( - 4 * ((abs(Xxi) * ((Xxi * (dxa *
abs(Xxr))) + (Xxr * (dya * abs(Xxr))))) + ((sgn(Xxi) * (Xxi * (Xxr * (dya *
abs(Xxr))))) + (sgn(Xxr) * (Xxi * (Xxr * (dxa * abs(Xxi)))))))) + daa)))))));
        dxbn = ((2 * (abs(Xxr) * (Xxr * (dxb * abs(Xxr))))) + (( - 2 *
((abs(Xxi) * (Xxr * (dxb * abs(Xxi)))) + (sgn(Xxi) * ((Xxr * Xxr) * (dyb *
abs(Xxi)))))) + ((2 * (sgn(Xxr) * ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 2 *
((abs(Xxr) * (Xxi * (dyb * abs(Xxr)))) + (sgn(Xxr) * ((Xxi * Xxi) * (dxb *
abs(Xxr)))))) + ((2 * ((abs(Xxi) * (Xxi * (dyb * abs(Xxi)))) + (sgn(Xxi) *
((Xxi * Xxi) * (dyb * abs(Xxi)))))) + (( - 4 * ((abs(Xxi) * ((Xxi * (dxb *
abs(Xxr))) + (Xxr * (dyb * abs(Xxr))))) + ((sgn(Xxi) * (Xxi * (Xxr * (dyb *
abs(Xxr))))) + (sgn(Xxr) * (Xxi * (Xxr * (dxb * abs(Xxi)))))))) + dab)))))));
        dyan = ((2 * (abs(Xxr) * (Xxr * (dya * abs(Xxr))))) + (( - 2 *
(abs(Xxi) * (Xxr * (dya * abs(Xxi))))) + ((2 * ((abs(Xxr) * (Xxi * (dxa *
abs(Xxr)))) + (sgn(Xxi) * ((Xxr * Xxr) * (dya * abs(Xxr)))))) + (( - 2 *
(abs(Xxi) * (Xxi * (dxa * abs(Xxi))))) + ((2 * (sgn(Xxr) * ((Xxr * Xxr) * (dxa
* abs(Xxi))))) + (( - 2 * ((sgn(Xxi) * ((Xxi * Xxi) * (dya * abs(Xxr)))) +
(sgn(Xxr) * ((Xxi * Xxi) * (dxa * abs(Xxi)))))) + ((4 * (abs(Xxi) * (Xxr * (dxa
* abs(Xxr))))) + (( - 4 * ((abs(Xxi) * (Xxi * (dya * abs(Xxr)))) + (sgn(Xxi) *
(Xxi * (Xxr * (dya * abs(Xxi)))))))) + ((4 * (sgn(Xxr) * (Xxi * (Xxr * (dxa *
abs(Xxr)))))) + dba)))))))));
        dybn = ((2 * (abs(Xxr) * (Xxr * (dyb * abs(Xxr))))) + (( - 2 *
(abs(Xxi) * (Xxr * (dyb * abs(Xxi))))) + ((2 * ((abs(Xxr) * (Xxi * (dxb *
abs(Xxr)))) + (sgn(Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) + (( - 2 *
(abs(Xxi) * (Xxi * (dxb * abs(Xxi))))) + ((2 * (sgn(Xxr) * ((Xxr * Xxr) * (dxb
* abs(Xxi))))) + (( - 2 * ((sgn(Xxi) * ((Xxi * Xxi) * (dyb * abs(Xxr)))) +
(sgn(Xxr) * ((Xxi * Xxi) * (dxb * abs(Xxi)))))) + ((4 * (abs(Xxi) * (Xxr * (dxb
* abs(Xxr))))) + (( - 4 * ((abs(Xxi) * (Xxi * (dyb * abs(Xxr)))) + (sgn(Xxi) *
(Xxi * (Xxr * (dyb * abs(Xxi)))))))) + ((4 * (sgn(Xxr) * (Xxi * (Xxr * (dxb *
abs(Xxr)))))) + dbb)))))))));
      </derivative>
    </formula>
  </group>
```

```
  <group type="59" name="Mothbrot 4th 3x1">
    <formula power="4" glitch="1.0000000000000003e-5">
      <reference t='R'>
        Xrn = (Cr + ((Xi * ((Xi * Xi) * abs(Xi))) + (( - 3 * (((Xi * Xi) * (Xr
* abs(Xr))) + (Xi * ((Xr * Xr) * abs(Xi))))) + (Xr * ((Xr * Xr) * abs(Xr))))));
        Xin = (Ci + (( - 1 * (Xi * ((Xi * Xi) * abs(Xr)))) + ((3 * (Xi * ((Xr *
Xr) * abs(Xr)))) + (( - 3 * ((Xi * Xi) * (Xr * abs(Xi)))) + (Xr * ((Xr * Xr) *
abs(Xi)))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((Xi * ((Xi * Xi) * diffabs(Xi,xi))) + ((Xr * ((Xr * Xr) *
diffabs(Xr,xr))) + ((xi * ((xi * xi) * abs((Xi + xi)))) + (( - 3 * (((xi * xi)
* (xr * abs((Xr + xr)))) + (xi * ((xr * xr) * abs((Xi + xi)))))) + ((3 * (Xr *
((xr * xr) * abs((Xr + xr))))) + (( - 3 * (Xr * ((xi * xi) * abs((Xr + xr)))))
+ ((3 * ((Xr * Xr) * (xr * abs((Xr + xr))))) + (( - 3 * ((Xi * (((Xr * Xr) *
diffabs(Xi,xi)) + ((xr * xr) * abs((Xi + xi))))) + ((Xr * Xr) * (xi * abs((Xi +
xi)))))) + ((3 * (Xi * ((xi * xi) * abs((Xi + xi))))) + (( - 3 * ((Xi * Xi) *
(xr * abs((Xr + xr))))) + ((3 * ((Xi * Xi) * (xi * abs((Xi + xi))))) + (( - 3 *
((Xi * Xi) * (Xr * diffabs(Xr,xr)))) + (( - 6 * ((Xi * ((Xr * ((xi * abs((Xr +
xr))) + (xr * abs((Xi + xi))))) + (xi * (xr * abs((Xr + xr)))))) + (Xr * (xi *
(xr * abs((Xi + xi)))))))) + (xr * ((xr * xr) * abs((Xr + xr)))))))))))))))));
        xin = (ci + ((Xr * ((Xr * Xr) * diffabs(Xi,xi))) + (( - 1 * ((Xi * ((Xi
* Xi) * diffabs(Xr,xr))) + (xi * ((xi * xi) * abs((Xr + xr)))))) + ((3 * (xi *
((xr * xr) * abs((Xr + xr))))) + (( - 3 * ((xi * xi) * (xr * abs((Xi + xi)))))
+ ((3 * (Xr * ((xr * xr) * abs((Xi + xi))))) + (( - 3 * (Xr * ((xi * xi) *
abs((Xi + xi))))) + ((3 * ((Xi * (((Xr * Xr) * diffabs(Xr,xr)) + ((xr * xr) *
abs((Xr + xr))))) + ((Xr * Xr) * ((xi * abs((Xr + xr))) + (xr * abs((Xi +
xi)))))))) + (( - 3 * (((Xi * Xi) * ((Xr * diffabs(Xi,xi)) + ((xi * abs((Xr +
xr))) + (xr * abs((Xi + xi)))))) + (Xi * ((xi * xi) * abs((Xr + xr)))))) + ((6
* (Xr * (xi * (xr * abs((Xr + xr)))))) + (( - 6 * (Xi * (xi * (xr * abs((Xi +
xi)))))) + ((6 * (Xi * (Xr * (xr * abs((Xr + xr)))))) + (( - 6 * (Xi * (Xr *
(xi * abs((Xi + xi)))))) + (xr * ((xr * xr) * abs((Xi + xi)))))))))))))))));
      </perturbation>
      <derivative t='M'>
        dxan = ((Xxi * ((Xxi * Xxi) * (dya * sgn(Xxi)))) + ((Xxr * ((Xxr * Xxr)
* (dxa * sgn(Xxr)))) + (( - 3 * ((Xxr * Xxr) * (dya * abs(Xxi)))) + ((3 *
(((Xxi * Xxi) * (dya * abs(Xxi))) + ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 3
* (((Xxi * Xxi) * ((Xxr * (dxa * sgn(Xxr))) + (dxa * abs(Xxr)))) + (Xxi * ((Xxr
* Xxr) * (dya * sgn(Xxi)))))) + (( - 6 * (Xxi * (Xxr * ((dxa * abs(Xxi)) + (dya
* abs(Xxr)))))) + daa))))));
        dxbn = ((Xxi * ((Xxi * Xxi) * (dyb * sgn(Xxi)))) + ((Xxr * ((Xxr * Xxr)
* (dxb * sgn(Xxr)))) + (( - 3 * ((Xxr * Xxr) * (dyb * abs(Xxi)))) + ((3 *
(((Xxi * Xxi) * (dyb * abs(Xxi))) + ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 3
* (((Xxi * Xxi) * ((Xxr * (dxb * sgn(Xxr))) + (dxb * abs(Xxr)))) + (Xxi * ((Xxr
* Xxr) * (dyb * sgn(Xxi)))))) + (( - 6 * (Xxi * (Xxr * ((dxb * abs(Xxi)) + (dyb
* abs(Xxr)))))) + dab))))));
        dyan = ((Xxr * ((Xxr * Xxr) * (dya * sgn(Xxi)))) + (( - 1 * (Xxi *
((Xxi * Xxi) * (dxa * sgn(Xxr))))) + ((3 * ((Xxr * Xxr) * ((dxa * abs(Xxi)) +
(dya * abs(Xxr))))) + (( - 3 * ((Xxi * Xxi) * ((dxa * abs(Xxi)) + (dya *
abs(Xxr))))) + ((3 * (Xxi * ((Xxr * Xxr) * (dxa * sgn(Xxr))))) + (( - 3 * ((Xxi
* Xxi) * (Xxr * (dya * sgn(Xxi))))) + (( - 6 * (Xxi * (Xxr * (dya *
abs(Xxi))))) + ((6 * (Xxi * (Xxr * (dxa * abs(Xxr))))) + dba)))))));
        dybn = ((Xxr * ((Xxr * Xxr) * (dyb * sgn(Xxi)))) + (( - 1 * (Xxi *
((Xxi * Xxi) * (dxb * sgn(Xxr))))) + ((3 * ((Xxr * Xxr) * ((dxb * abs(Xxi)) +
(dyb * abs(Xxr))))) + (( - 3 * ((Xxi * Xxi) * ((dxb * abs(Xxi)) + (dyb *
abs(Xxr))))) + ((3 * (Xxi * ((Xxr * Xxr) * (dxb * sgn(Xxr))))) + (( - 3 * ((Xxi
* Xxi) * (Xxr * (dyb * sgn(Xxi))))) + (( - 6 * (Xxi * (Xxr * (dyb *
```

```
3295  abs(Xxi))))) + ((6 * (Xxi * (Xxr * (dxb * abs(Xxr))))) + dbb)))))))));
        </derivative>
      </formula>
    </group>

3300    <group type="60" name="Mothbrot 5th 1x4">
      <formula power="5" glitch="1.0000000000000002e-4">
        <reference t='R'>
          Xrn = (Cr + ((abs((Xi * (Xi * Xi))) * ((Xi * abs(Xr)) + (Xr *
    abs(Xi)))) + (( - 1 * (abs((Xr * (Xr * Xr))) * (Xi * abs(Xi)))) + (( - 3 *
3305  ((abs(((Xi * Xi) * Xr)) * (Xr * abs(Xr))) + (abs((Xi * (Xr * Xr))) * ((Xi *
    abs(Xr)) + (Xr * abs(Xi)))))) + ((3 * (abs(((Xi * Xi) * Xr)) * (Xi * abs(Xi))))
    + (abs((Xr * (Xr * Xr))) * (Xr * abs(Xr)))))))));
          Xin = (Ci + ((abs((Xi * (Xi * Xi))) * (Xi * abs(Xi))) + ((abs((Xr * (Xr
    * Xr))) * (Xi * abs(Xr))) + (( - 1 * (abs((Xi * (Xi * Xi))) * (Xr * abs(Xr))))
3310  + ((3 * (abs((Xi * (Xr * Xr))) * (Xr * abs(Xr)))) + (( - 3 * ((abs(((Xi * Xi) *
    Xr)) * ((Xi * abs(Xr)) + (Xr * abs(Xi)))) + (abs((Xi * (Xr * Xr))) * (Xi *
    abs(Xi))))) + (abs((Xr * (Xr * Xr))) * (Xr * abs(Xi)))))))));
        </reference>
        <perturbation t='R'>
3315      xrn = (cr + ((abs((Xi * (Xi * Xi))) * ((Xi * diffabs(Xr,xr)) + ((Xr *
    diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi))))))) +
    ((abs((Xr * (Xr * Xr))) * (Xr * diffabs(Xr,xr))) + ((abs((Xi + xi)) *
    (diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi *
    (xi * xi)))) * (Xr + xr))) + ((abs((Xr + xr)) * ((Xi * diffabs((Xi * (Xi *
3320  Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi))))) + ((Xr
    * diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr *
    (xr * xr))))) + ((xi * diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi *
    (xi * xi)))) + (xi * (xi * xi))))) + (xr * diffabs((Xr * (Xr * Xr)),((3 * (((Xr
    * Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr))))))))) + (( - 1 * ((abs((Xr
3325  * (Xr * Xr))) * ((Xi * diffabs(Xi,xi)) + (xi * abs((Xi + xi))))) + (abs((Xi +
    xi)) * (diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) +
    (xr * (xr * xr)))) * (Xi + xi))))) + (( - 3 * ((abs(((Xi * Xi) * Xr)) * (xr *
    abs((Xr + xr)))) + ((abs((Xi * (Xr * Xr))) * ((xi * abs((Xr + xr))) + (xr *
    abs((Xi + xi))))) + ((abs((Xi + xi)) * (xr * diffabs((Xi * (Xr * Xr)),((Xi *
3330  (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) +
    (xi * (xr * xr))))))))) + (abs((Xr + xr)) * ((xi * diffabs((Xi * (Xr * Xr)),((Xi
    * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr))))
    + (xi * (xr * xr)))))))) + (xr * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) +
    ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) *
3335  xr))))))))))))) + ((3 * ((abs(((Xi * Xi) * Xr)) * (xi * abs((Xi + xi)))) +
    (abs((Xi + xi)) * (xi * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi
    * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr))))))))) + (( - 3 *
    ((abs(((Xi * Xi) * Xr)) * (Xr * diffabs(Xr,xr))) + ((abs((Xi * (Xr * Xr))) *
    ((Xi * diffabs(Xr,xr)) + (Xr * diffabs(Xi,xi)))) + ((abs((Xi + xi)) * (Xr *
3340  diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
    (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))) + (abs((Xr + xr)) *
    ((Xi * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 *
    ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))) + (Xr *
    diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
3345  (xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))))) + ((3 * ((abs(((Xi * Xi) *
    Xr)) * (Xi * diffabs(Xi,xi))) + (abs((Xi + xi)) * (Xi * diffabs(((Xi * Xi) *
    Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) +
    ((xi * xi) * xr))))))))) + (abs((Xr * (Xr * Xr))) * (xr * abs((Xr +
    xr))))))))))))))));
3350      xin = (ci + ((abs((Xi * (Xi * Xi))) * ((Xi * diffabs(Xi,xi)) + (xi *
    abs((Xi + xi))))) + ((abs((Xr * (Xr * Xr))) * ((Xi * diffabs(Xr,xr)) + ((Xr *
```

```
diffabs(Xi,xi)) + (xi * abs((Xr + xr)))))) + ((abs((Xi + xi)) * ((Xi *
diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi *
(xi * xi))))) + ((Xr * diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr *
(xr * xr)))) + (xr * (xr * xr))))) + ((xi * diffabs((Xi * (Xi * Xi)),((3 *
(((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi))))) + (xr *
diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr *
(xr * xr)))))))))) + ((abs((Xr + xr)) * (diffabs((Xr * (Xr * Xr)),((3 * (((Xr *
Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr)))) * (Xi + xi))) + (( - 1 *
((abs((Xi * (Xi * Xi))) * ((Xr * diffabs(Xr,xr)) + (xr * abs((Xr + xr))))) +
(abs((Xr + xr)) * (diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi
* xi)))) + (xi * (xi * xi)))) * (Xr + xr))))) + ((3 * ((abs((Xi * (Xr * Xr))) *
(xr * abs((Xr + xr)))) + (abs((Xr + xr)) * (xr * diffabs((Xi * (Xr * Xr)),((Xi
* (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr))))
+ (xi * (xr * xr))))))))))) + (( - 3 * ((abs(((Xi * Xi) * Xr)) * ((xi * abs((Xr
+ xr))) + (xr * abs((Xi + xi))))) + ((abs((Xi * (Xr * Xr))) * (xi * abs((Xi +
xi)))) + ((abs((Xi + xi)) * ((xi * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) +
(((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr *
xr)))))))) + (xr * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi *
xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr)))))))) + (abs((Xr +
xr)) * (xi * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) +
((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))) + ((3 * ((abs((Xi *
(Xr * Xr))) * (Xr * diffabs(Xr,xr))) + (abs((Xr + xr)) * (Xr * diffabs((Xi *
(Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) +
(Xr * (xi * xr)))) + (xi * (xr * xr))))))))) + (( - 3 * ((abs(((Xi * Xi) *
Xr)) * ((Xi * diffabs(Xr,xr)) + (Xr * diffabs(Xi,xi)))) + ((abs((Xi * (Xr *
Xr))) * (Xi * diffabs(Xi,xi))) + ((abs((Xi + xi)) * ((Xi * diffabs((Xi * (Xr *
Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr *
(xi * xr)))) + (xi * (xr * xr))))))) + (Xr * diffabs(((Xi * Xi) * Xr),(((Xi *
Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) *
xr)))))))) + (abs((Xr + xr)) * (Xi * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr)
+ ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) *
xr))))))))))) + (abs((Xr * (Xr * Xr))) * (xr * abs((Xi + xi)))))))))))))));
      </perturbation>
      <derivative t="M">
        dxan = (daa + ((abs((Xxi * (Xxi * Xxi))) * ((dxa * abs(Xxi)) + (dya *
abs(Xxr)))) + ((abs((Xxr * (Xxr * Xxr))) * (dxa * abs(Xxr))) + ((sgn(Xxi) *
(Xxr * (dya * abs((Xxi * (Xxi * Xxi)))))) + ((sgn(Xxr) * ((Xxi * (dxa *
abs((Xxi * (Xxi * Xxi))))) + (Xxr * (dxa * abs((Xxr * (Xxr * Xxr)))))))) + (( -
1 * (sgn(Xxi) * (Xxi * (dya * abs((Xxr * (Xxr * Xxr))))))) + (( - 3 * (abs((Xxi
* (Xxr * Xxr))) * (dya * abs(Xxr)))) + ((3 * (abs(((Xxi * Xxi) * Xxr)) * (dya *
abs(Xxi)))) + (( - 3 * ((abs(((Xxi * Xxi) * Xxr)) * (dxa * abs(Xxr))) +
((abs((Xxi * (Xxr * Xxr))) * (dxa * abs(Xxi))) + ((sgn(Xxi) * ((Xxr * (dya *
abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) *
(dya * abs(Xxi))))))) + (sgn(Xxr) * (Xxr * (dxa * abs(((Xxi * Xxi) *
Xxr)))))))) + ((3 * ((sgn(Xxi) * (Xxi * (dya * abs(((Xxi * Xxi) * Xxr))))) +
(sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dxa *
abs(Xxr))))))) + (( - 3 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
(Xxr * (dxa * abs(Xxr)))))) + ((sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi *
((Xxr * Xxr) * (dya * abs(Xxr)))))) + (sgn(Xxr) * ((Xxi * (dxa * abs((Xxi *
(Xxr * Xxr))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxa *
abs(Xxi))))))))) + ((3 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * ((Xxi * ((Xxi *
Xxi) * (dya * abs(Xxr)))) + ((Xxi * Xxi) * (Xxr * (dya * abs(Xxi))))))) +
((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa *
abs(Xxi))))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxr *
Xxr) * (dya * abs(Xxr)))))) + (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr
* Xxr) * (dxa * abs(Xxi))))))) + ((6 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
((Xxi * Xxi) * (Xxr * (dya * abs(Xxi))))))) + (( - 6 * (sgn(Xxi) * ((sgn(Xxr) *
```

```
sgn(Xxr)) * ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxr))))))) + ( - 1 * (abs((Xxr *
(Xxr * Xxr))) * (dya * abs(Xxi))))))))))))))))));
        dxbn = (dab + ((abs((Xxi * (Xxi * Xxi))) * ((dxb * abs(Xxi)) + (dyb *
abs(Xxr)))) + ((abs((Xxr * (Xxr * Xxr))) * (dxb * abs(Xxr))) + ((sgn(Xxi) *
(Xxr * (dyb * abs((Xxi * (Xxi * Xxi)))))) + ((sgn(Xxr) * ((Xxi * (dxb *
abs((Xxi * (Xxi * Xxi))))) + (Xxr * (dxb * abs((Xxr * (Xxr * Xxr))))))) + (( -
1 * (sgn(Xxi) * (Xxi * (dyb * abs((Xxr * (Xxr * Xxr))))))) + (( - 3 * (abs((Xxi
* (Xxr * Xxr))) * (dyb * abs(Xxr)))) + ((3 * (abs(((Xxi * Xxi) * Xxr)) * (dyb *
abs(Xxi)))) + (( - 3 * ((abs(((Xxi * Xxi) * Xxr)) * (dxb * abs(Xxr))) +
((abs((Xxi * (Xxr * Xxr))) * (dxb * abs(Xxi))) + ((sgn(Xxi) * ((Xxr * (dyb *
abs((Xxi * (Xxr * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) *
(dyb * abs(Xxi))))))) + (sgn(Xxr) * (Xxr * (dxb * abs(((Xxi * Xxi) *
Xxr))))))))) + ((3 * ((sgn(Xxi) * (Xxi * (dyb * abs(((Xxi * Xxi) * Xxr))))) +
(sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dxb *
abs(Xxr))))))) + (( - 3 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
(Xxr * (dxb * abs(Xxr)))))) + ((sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi *
((Xxr * Xxr) * (dyb * abs(Xxr)))))) + (sgn(Xxr) * ((Xxi * (dxb * abs((Xxi *
(Xxr * Xxr))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxb *
abs(Xxi))))))))) + ((3 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * ((Xxi * ((Xxi *
Xxi) * (dyb * abs(Xxr)))) + ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi))))))) +
((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb *
abs(Xxi))))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxr *
Xxr) * (dyb * abs(Xxr)))))) + (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr
* Xxr) * (dxb * abs(Xxi))))))) + ((6 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi))))))) + (( - 6 * (sgn(Xxi) * ((sgn(Xxr) *
sgn(Xxr)) * ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxr))))))) + ( - 1 * (abs((Xxr *
(Xxr * Xxr))) * (dyb * abs(Xxi))))))))))))))))));
        dyan = (dba + ((abs((Xxi * (Xxi * Xxi))) * (dya * abs(Xxi))) +
((abs((Xxr * (Xxr * Xxr))) * (dxa * abs(Xxi))) + ((sgn(Xxi) * ((Xxi * (dya *
abs((Xxi * (Xxi * Xxi))))) + (Xxr * (dya * abs((Xxr * (Xxr * Xxr))))))) +
((sgn(Xxr) * (Xxi * (dxa * abs((Xxr * (Xxr * Xxr)))))) + (( - 1 * ((abs((Xxi *
(Xxi * Xxi))) * (dxa * abs(Xxr))) + (sgn(Xxr) * (Xxr * (dxa * abs((Xxi * (Xxi *
Xxi))))))))) + (( - 3 * ((abs(((Xxi * Xxi) * Xxr)) * (dya * abs(Xxr))) +
(abs((Xxi * (Xxr * Xxr))) * (dya * abs(Xxi))))) + ((3 * (abs((Xxi * (Xxr *
Xxr))) * (dxa * abs(Xxr)))) + (( - 3 * ((abs(((Xxi * Xxi) * Xxr)) * (dxa *
abs(Xxi))) + (sgn(Xxi) * (Xxr * (dya * abs(((Xxi * Xxi) * Xxr))))))) + ((3 *
((sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dya *
abs(Xxr)))))) + (sgn(Xxr) * ((Xxr * (dxa * abs((Xxi * (Xxr * Xxr))))) +
((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxi))))))) + (( - 3
* ((sgn(Xxi) * ((Xxi * (dya * abs((Xxi * (Xxr * Xxr)))))) + ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dya * abs(Xxi))))))) + (sgn(Xxr) * (Xxi *
(dxa * abs(((Xxi * Xxi) * Xxr))))))) + ((3 * ((sgn(Xxi) * ((sgn(Xxi) *
sgn(Xxi)) * (Xxi * ((Xxi * Xxi) * (dya * abs(Xxi)))))) + (sgn(Xxr) * ((sgn(Xxr)
* sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxa * abs(Xxr))))))) + (( - 3 *
((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * ((Xxi * Xxi) * (Xxr * (dya *
abs(Xxr)))))) + ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * ((Xxi * Xxi) *
(dxa * abs(Xxr)))) + ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi)))))))) + (( - 6 *
((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dya *
abs(Xxi))))))) + ((6 * (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr * Xxr)
* (dxa * abs(Xxr))))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi
* Xxi) * (Xxr * (dya * abs(Xxr)))))) + (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) *
((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi)))))))) + (abs((Xxr * (Xxr * Xxr))) * (dya
* abs(Xxr))))))))))))))))));
        dybn = (dbb + ((abs((Xxi * (Xxi * Xxi))) * (dyb * abs(Xxi))) +
((abs((Xxr * (Xxr * Xxr))) * (dxb * abs(Xxi))) + ((sgn(Xxi) * ((Xxi * (dyb *
abs((Xxi * (Xxi * Xxi))))) + (Xxr * (dyb * abs((Xxr * (Xxr * Xxr))))))) +
((sgn(Xxr) * (Xxi * (dxb * abs((Xxr * (Xxr * Xxr)))))) + (( - 1 * ((abs((Xxi *
```

```
(Xxi * Xxi))) * (dxb * abs(Xxr))) + (sgn(Xxr) * (Xxr * (dxb * abs((Xxi * (Xxi *
Xxi)))))))) + (( - 3 * ((abs(((Xxi * Xxi) * Xxr)) * (dyb * abs(Xxr))) +
(abs((Xxi * (Xxr * Xxr))) * (dyb * abs(Xxi))))) + ((3 * (abs((Xxi * (Xxr *
Xxr))) * (dxb * abs(Xxr)))) + (( - 3 * ((abs(((Xxi * Xxi) * Xxr)) * (dxb *
3470 abs(Xxi))) + (sgn(Xxi) * (Xxr * (dyb * abs(((Xxi * Xxi) * Xxr))))))) + ((3 *
((sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dyb *
abs(Xxr)))))) + (sgn(Xxr) * ((Xxr * (dxb * abs((Xxi * (Xxr * Xxr))))) +
((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxi)))))))) + (( - 3
* ((sgn(Xxi) * ((Xxi * (dyb * abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) *
3475 sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxi))))))) + (sgn(Xxr) * (Xxi *
(dxb * abs(((Xxi * Xxi) * Xxr))))))) + ((3 * ((sgn(Xxi) * ((sgn(Xxi) *
sgn(Xxi)) * (Xxi * ((Xxi * Xxi) * (dyb * abs(Xxi)))))) + (sgn(Xxr) * ((sgn(Xxr)
* sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxb * abs(Xxr)))))))) + (( - 3 *
((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * ((Xxi * Xxi) * (Xxr * (dyb *
3480 abs(Xxr)))))) + ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * ((Xxi * Xxi) *
(dxb * abs(Xxr)))) + ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi))))))))) + (( - 6 *
((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dyb *
abs(Xxi))))))) + ((6 * (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr * Xxr)
* (dxb * abs(Xxr))))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi
3485 * Xxi) * (Xxr * (dyb * abs(Xxr)))))) + (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) *
((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi))))))) + (abs((Xxr * (Xxr * Xxr))) * (dyb
* abs(Xxr))))))))))))))))));
      </derivative>
    </formula>
3490  </group>

  <group type="61" name="Mothbrot 5th 2x3">
    <formula power="5" glitch="1.0000000000000002e-4">
      <reference t='R'>
3495        Xrn = (Cr + ((abs((Xi * Xi)) * ((Xi * Xi) * abs(Xr))) + (( - 1 *
((abs((Xi * Xi)) * ((Xr * Xr) * abs(Xr))) + (abs((Xr * Xr)) * ((Xi * Xi) *
abs(Xr))))) + (( - 2 * ((abs((Xi * Xr)) * ((Xr * Xr) * abs(Xi))) + (abs((Xr *
Xr)) * (Xi * (Xr * abs(Xi)))))) + ((2 * ((abs((Xi * Xi)) * (Xi * (Xr *
abs(Xi))))) + (abs((Xi * Xr)) * ((Xi * Xi) * abs(Xi))))) + (( - 4 * (abs((Xi *
3500 Xr)) * (Xi * (Xr * abs(Xr))))) + (abs((Xr * Xr)) * ((Xr * Xr) * abs(Xr)))))))));
        Xin = (Ci + ((abs((Xi * Xi)) * ((Xi * Xi) * abs(Xi))) + (( - 1 *
((abs((Xi * Xi)) * ((Xr * Xr) * abs(Xi))) + (abs((Xr * Xr)) * ((Xi * Xi) *
abs(Xi))))) + ((2 * ((abs((Xi * Xr)) * ((Xr * Xr) * abs(Xr))) + (abs((Xr * Xr))
* (Xi * (Xr * abs(Xr)))))) + (( - 2 * ((abs((Xi * Xi)) * (Xi * (Xr * abs(Xr))))
3505 + (abs((Xi * Xr)) * ((Xi * Xi) * abs(Xr))))) + (( - 4 * (abs((Xi * Xr)) * (Xi *
(Xr * abs(Xi))))) + (abs((Xr * Xr)) * ((Xr * Xr) * abs(Xi)))))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs((Xi * Xi)) * (((Xi * Xi) * diffabs(Xr,xr)) + ((xi *
3510 xi) * abs((Xr + xr))))) + ((abs((Xr * Xr)) * ((Xr * Xr) * diffabs(Xr,xr))) +
((abs((Xr + xr)) * (((Xi * Xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
xi)))) + (((Xr * Xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))) +
(((xi * xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))) + ((xr * xr) *
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))))) + (( - 1 * ((abs((Xi *
3515 Xi)) * (((Xr * Xr) * diffabs(Xr,xr)) + ((xr * xr) * abs((Xr + xr))))) +
((abs((Xr * Xr)) * (((Xi * Xi) * diffabs(Xr,xr)) + ((xi * xi) * abs((Xr +
xr))))) + (abs((Xr + xr)) * (((Xi * Xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) +
(xr * xr)))) + (((Xr * Xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))
+ (((xi * xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))) + ((xr * xr)
3520 * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))))) + (( - 2 * ((abs((Xi
* Xr)) * ((xr * xr) * abs((Xi + xi)))) + ((abs((Xr * Xr)) * (xi * (xr * abs((Xi
+ xi))))) + (abs((Xi + xi)) * ((xr * xr) * diffabs((Xi * Xr),((Xi * xr) + (xi *
```

```
(Xr + xr))))))))) + ((2 * (abs((Xi * Xi)) * (xi * (xr * abs((Xi + xi)))))) + ((
- 2 * (abs((Xi + xi)) * (xi * (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr *
xr)))))))) + ((2 * ((abs((Xi * Xr)) * ((xi * xi) * abs((Xi + xi)))) + ((abs((Xr
* Xr)) * (Xr * (xr * abs((Xr + xr))))) + ((abs((Xi + xi)) * (((xi * xi) *
diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))) + (xi * (xr * diffabs((Xi *
Xi),((2 * (Xi * xi)) + (xi * xi))))))) + (abs((Xr + xr)) * (Xr * (xr *
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))))))) + (( - 2 * ((abs((Xi *
Xi)) * (Xr * (xr * abs((Xr + xr))))) + ((abs((Xr * Xr)) * (Xr * (xi * abs((Xi +
xi))))) + (abs((Xr + xr)) * (Xr * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) +
(xi * xi))))))))) + ((2 * (abs((Xi * Xi)) * (Xr * (xi * abs((Xi + xi)))))) + ((
- 2 * (abs((Xi + xi)) * (Xr * (xi * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr *
xr)))))))) + ((2 * (abs((Xi + xi)) * (Xr * (xi * diffabs((Xi * Xi),((2 * (Xi *
xi)) + (xi * xi))))))) + (( - 2 * ((abs((Xi * Xr)) * ((Xr * Xr) *
diffabs(Xi,xi))) + ((abs((Xr * Xr)) * (Xi * (xr * abs((Xi + xi))))) + (abs((Xi
+ xi)) * ((Xr * Xr) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) +
((2 * (abs((Xi * Xi)) * (Xi * (xr * abs((Xi + xi)))))) + (( - 2 * (abs((Xi +
xi)) * (Xi * (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + ((2 *
(abs((Xi + xi)) * (Xi * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
xi))))))) + (( - 2 * ((abs((Xr * Xr)) * (Xi * (xi * abs((Xr + xr))))) +
(abs((Xr + xr)) * (Xi * (xi * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr *
xr)))))))) + ((2 * ((abs((Xi * Xi)) * (Xi * (xi * abs((Xr + xr))))) + (abs((Xr
+ xr)) * (Xi * (xi * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))) + ((
- 2 * (abs((Xr * Xr)) * (Xi * (Xr * diffabs(Xi,xi))))) + ((2 * (abs((Xi * Xi))
* (Xi * (Xr * diffabs(Xi,xi))))) + (( - 2 * (abs((Xi + xi)) * (Xi * (Xr *
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + ((2 * ((abs((Xi * Xr)) *
((Xi * Xi) * diffabs(Xi,xi))) + (abs((Xi + xi)) * (((Xi * Xi) * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr))))) + (Xi * (Xr * diffabs((Xi * Xi),((2 * (Xi
* xi)) + (xi * xi))))))))) + (( - 4 * ((abs((Xi * Xr)) * ((Xi * (xr * abs((Xr +
xr)))) + ((Xr * ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi))))) + (xi * (xr *
abs((Xr + xr))))))) + ((abs((Xi + xi)) * (Xr * (xr * diffabs((Xi * Xr),((Xi *
xr) + (xi * (Xr + xr))))))) + (abs((Xr + xr)) * ((Xi * (xr * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr))))))) + ((Xr * (xi * diffabs((Xi * Xr),((Xi *
xr) + (xi * (Xr + xr))))))) + (xi * (xr * diffabs((Xi * Xr),((Xi * xr) + (xi *
(Xr + xr)))))))))))) + ((4 * ((abs((Xi * Xr)) * (Xi * (xi * abs((Xi + xi)))))) +
(abs((Xi + xi)) * (Xi * (xi * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr +
xr))))))))) + (( - 4 * ((abs((Xi * Xr)) * (Xi * (Xr * diffabs(Xr,xr)))) +
(abs((Xr + xr)) * (Xi * (Xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr +
xr))))))))) + (abs((Xr * Xr)) * ((xr * xr) * abs((Xr +
xr))))))))))))))))))))))))))));
        xin = (ci + ((abs((Xi * Xi)) * (((Xi * Xi) * diffabs(Xi,xi)) + ((xi *
xi) * abs((Xi + xi))))) + ((abs((Xr * Xr)) * ((Xr * Xr) * diffabs(Xi,xi))) +
((abs((Xi + xi)) * (((Xi * Xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
xi)))) + (((Xr * Xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))) +
(((xi * xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))) + ((xr * xr) *
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + (( - 1 * ((abs((Xi *
Xi)) * (((Xr * Xr) * diffabs(Xi,xi)) + ((xr * xr) * abs((Xi + xi))))) +
((abs((Xr * Xr)) * (((Xi * Xi) * diffabs(Xi,xi)) + ((xi * xi) * abs((Xi +
xi))))) + (abs((Xi + xi)) * (((Xi * Xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) +
(xr * xr)))) + (((Xr * Xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))
+ (((xi * xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))) + ((xr * xr)
* diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))))) + ((2 * ((abs((Xi *
Xr)) * ((xr * xr) * abs((Xr + xr)))) + ((abs((Xr * Xr)) * (xi * (xr * abs((Xr +
xr))))) + (abs((Xr + xr)) * ((xi * (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) +
(xr * xr))))) + ((xr * xr) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr +
xr))))))))) + (( - 2 * ((abs((Xi * Xi)) * (xi * (xr * abs((Xr + xr))))) +
((abs((Xi * Xr)) * ((xi * xi) * abs((Xr + xr)))) + (abs((Xr + xr)) * (((xi *
xi) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))) + (xi * (xr *
```

```
3580  diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))))) + ((2 * (abs((Xr * Xr))
      * (Xr * (xr * abs((Xi + xi)))))) + (( - 2 * (abs((Xi * Xi)) * (Xr * (xr *
      abs((Xi + xi)))))) + ((2 * (abs((Xi + xi)) * (Xr * (xr * diffabs((Xr * Xr),((2
      * (Xr * xr)) + (xr * xr)))))) + (( - 2 * (abs((Xi + xi)) * (Xr * (xr *
      diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))) + ((2 * ((abs((Xr * Xr)) *
3585  (Xr * (xi * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xi * diffabs((Xr *
      Xr),((2 * (Xr * xr)) + (xr * xr)))))))) + (( - 2 * ((abs((Xi * Xi)) * (Xr * (xi
      * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xi * diffabs((Xi * Xi),((2 * (Xi
      * xi)) + (xi * xi)))))))) + ((2 * ((abs((Xi * Xr)) * ((Xr * Xr) *
      diffabs(Xr,xr))) + ((abs((Xr * Xr)) * (Xi * (xr * abs((Xr + xr))))) + (abs((Xr
3590  + xr)) * ((Xi * (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))) + ((Xr
      * Xr) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))))) + (( - 2 *
      ((abs((Xi * Xi)) * (Xi * (xr * abs((Xr + xr))))) + ((abs((Xr * Xr)) * (Xi * (xi
      * abs((Xi + xi)))))) + (abs((Xr + xr)) * (Xi * (xr * diffabs((Xi * Xi),((2 * (Xi
      * xi)) + (xi * xi)))))))) + ((2 * (abs((Xi * Xi)) * (Xi * (xi * abs((Xi +
3595  xi)))))) + (( - 2 * (abs((Xi + xi)) * (Xi * (xi * diffabs((Xr * Xr),((2 * (Xr *
      xr)) + (xr * xr)))))))) + ((2 * ((abs((Xr * Xr)) * (Xi * (Xr * diffabs(Xr,xr))))
      + ((abs((Xi + xi)) * (Xi * (xi * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
      xi)))))) + (abs((Xr + xr)) * (Xi * (Xr * diffabs((Xr * Xr),((2 * (Xr * xr)) +
      (xr * xr)))))))) + (( - 2 * ((abs((Xi * Xi)) * (Xi * (Xr * diffabs(Xr,xr))))) +
3600  ((abs((Xi * Xr)) * ((Xi * Xi) * diffabs(Xr,xr))) + (abs((Xr + xr)) * (((Xi *
      Xi) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))) + (Xi * (Xr *
      diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))))) + (( - 4 * ((abs((Xi *
      Xr)) * (xi * (xr * abs((Xi + xi)))))) + (abs((Xi + xi)) * (xi * (xr *
      diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + ((4 * ((abs((Xi * Xr))
3605  * (Xr * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xr * diffabs((Xi *
      Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + (( - 4 * ((abs((Xi * Xr)) * ((Xi *
      ((Xr * diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi))))))) +
      (Xr * (xi * abs((Xi + xi))))))) + ((abs((Xi + xi)) * ((Xi * (diffabs((Xi *
      Xr),((Xi * xr) + (xi * (Xr + xr)))) * (Xr + xr))) + (Xr * (xi * diffabs((Xi *
3610  Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + (abs((Xr + xr)) * (Xi * (xi *
      diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + (abs((Xr * Xr)) *
      ((xr * xr) * abs((Xi + xi))))))))))))))))))))))))))));
            </perturbation>
            <derivative t="M">
3615            dxan = ((sgn(Xxr) * (((Xxi * Xxi) * (dxa * abs((Xxi * Xxi)))) + ((Xxr *
      Xxr) * (dxa * abs((Xxr * Xxr)))))) + (( - 1 * (sgn(Xxr) * (((Xxi * Xxi) * (dxa
      * abs((Xxr * Xxr)))) + ((Xxr * Xxr) * (dxa * abs((Xxi * Xxi))))))) + (( - 2 *
      (abs((Xxr * Xxr)) * (Xxr * (dya * abs(Xxi))))) + ((2 * ((abs((Xxi * Xxi)) *
      (Xxr * (dya * abs(Xxi)))) + (abs((Xxr * Xxr)) * (Xxr * (dxa * abs(Xxr))))))) +
3620  (( - 2 * ((abs((Xxi * Xxi)) * (Xxr * (dxa * abs(Xxr)))) + (sgn(Xxi) * (((Xxr *
      Xxr) * (dya * abs((Xxi * Xxr)))) + (sgn(Xxr) * (Xxr * ((Xxr * Xxr) * (dya *
      abs(Xxi)))))))) + ((2 * ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dxa *
      abs(Xxr)))))) + (( - 2 * (abs((Xxr * Xxr)) * (Xxi * (dya * abs(Xxr))))) + ((2 *
      (abs((Xxi * Xxi)) * (Xxi * (dya * abs(Xxr))))) + (( - 2 * (abs((Xxr * Xxr)) *
3625  (Xxi * (dxa * abs(Xxi))))) + ((2 * (abs((Xxi * Xxi)) * (Xxi * (dxa *
      abs(Xxi))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxr * (dya * abs((Xxr * Xxr))))))))
      + ((2 * (sgn(Xxi) * (Xxi * (Xxr * (dya * abs((Xxi * Xxi))))))) + (( - 2 *
      (((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxr * Xxr) * (dya * abs(Xxr)))))) + (sgn(Xxi)
      * (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxa * abs(Xxi)))))))) + ((2 * (sgn(Xxi) *
3630  (((Xxi * Xxi) * (dya * abs((Xxi * Xxr)))) + (sgn(Xxr) * ((Xxi * Xxi) * (Xxr *
      (dya * abs(Xxi))))))) + (( - 2 * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * (Xxr
      * (dxa * abs(Xxr)))))) + ((2 * (((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) *
      (dya * abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa *
      abs(Xxi))))))))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxr * ((dxa * abs(Xxi)) + (dya
3635  * abs(Xxr)))))) + ((4 * (abs((Xxi * Xxr)) * (Xxi * (dya * abs(Xxi))))) + (( - 4
      * ((abs((Xxi * Xxr)) * (Xxi * (dxa * abs(Xxr)))) + ((sgn(Xxi) * (sgn(Xxr) *
```

```
(Xxi * ((Xxr * Xxr) * (dya * abs(Xxr))))))) + ((sgn(Xxr) * (Xxi * (Xxr * (dxa *
abs((Xxi * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxa *
abs(Xxi)))))))) + ((4 * ((sgn(Xxi) * sgn(Xxi)) * ((Xxi * Xxi) * (Xxr * (dya *
abs(Xxi)))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dxa *
abs(Xxr))))))) + daa))))))))))))))))))));
        dxbn = ((sgn(Xxr) * (((Xxi * Xxi) * (dxb * abs((Xxi * Xxi)))) + ((Xxr *
Xxr) * (dxb * abs((Xxr * Xxr)))))) + (( - 1 * (sgn(Xxr) * (((Xxi * Xxi) * (dxb
* abs((Xxr * Xxr)))) + ((Xxr * Xxr) * (dxb * abs((Xxi * Xxi))))))) + (( - 2 *
(abs((Xxr * Xxr)) * (Xxr * (dyb * abs(Xxi))))) + ((2 * ((abs((Xxi * Xxi)) *
(Xxr * (dyb * abs(Xxi)))) + (abs((Xxr * Xxr)) * (Xxr * (dxb * abs(Xxr)))))) +
(( - 2 * ((abs((Xxi * Xxi)) * (Xxr * (dxb * abs(Xxr)))) + (sgn(Xxi) * (((Xxr *
Xxr) * (dyb * abs((Xxi * Xxr)))) + (sgn(Xxr) * (Xxr * ((Xxr * Xxr) * (dyb *
abs(Xxi)))))))) + ((2 * ((sgn(Xxr) * sgn(Xxr)) * (Xxr * ((Xxr * Xxr) * (dxb *
abs(Xxr)))))) + (( - 2 * (abs((Xxr * Xxr)) * (Xxi * (dyb * abs(Xxr))))) + ((2 *
(abs((Xxi * Xxi)) * (Xxi * (dyb * abs(Xxr))))) + (( - 2 * (abs((Xxr * Xxr)) *
(Xxi * (dxb * abs(Xxi))))) + ((2 * (abs((Xxi * Xxi)) * (Xxi * (dxb *
abs(Xxi))))) + (( - 2 * (sgn(Xxi) * (Xxi * (Xxr * (dyb * abs((Xxr * Xxr)))))))
+ ((2 * (sgn(Xxi) * (Xxi * (Xxr * (dyb * abs((Xxi * Xxi))))))) + (( - 2 *
(((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxr))))) + (sgn(Xxi)
* (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxb * abs(Xxi)))))))) + ((2 * (sgn(Xxi) *
(((Xxi * Xxi) * (dyb * abs((Xxi * Xxr)))) + (sgn(Xxr) * ((Xxi * Xxi) * (Xxr *
(dyb * abs(Xxi)))))))) + (( - 2 * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * (Xxr
* (dxb * abs(Xxr)))))) + ((2 * (((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) *
(dyb * abs(Xxr))))) + (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb *
abs(Xxi)))))))) + (( - 4 * (abs((Xxi * Xxr)) * (Xxr * ((dxb * abs(Xxi)) + (dyb
* abs(Xxr)))))) + ((4 * (abs((Xxi * Xxr)) * (Xxi * (dyb * abs(Xxi))))) + (( - 4
* ((abs((Xxi * Xxr)) * (Xxi * (dxb * abs(Xxr)))) + ((sgn(Xxi) * (sgn(Xxr) *
(Xxi * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) + ((sgn(Xxr) * (Xxi * (Xxr * (dxb *
abs((Xxi * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxb *
abs(Xxi)))))))) + ((4 * ((sgn(Xxi) * sgn(Xxi)) * ((Xxi * Xxi) * (Xxr * (dyb *
abs(Xxi)))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dxb *
abs(Xxr))))))) + dab))))))))))))))))))));
        dyan = ((sgn(Xxi) * (((Xxi * Xxi) * (dya * abs((Xxi * Xxi)))) + ((Xxr *
Xxr) * (dya * abs((Xxr * Xxr)))))) + (( - 1 * (sgn(Xxi) * (((Xxi * Xxi) * (dya
* abs((Xxr * Xxr)))) + ((Xxr * Xxr) * (dya * abs((Xxi * Xxi))))))) + ((2 *
(abs((Xxr * Xxr)) * (Xxr * (dya * abs(Xxr))))) + (( - 2 * (abs((Xxi * Xxi)) *
(Xxr * (dya * abs(Xxr))))) + ((2 * (abs((Xxr * Xxr)) * (Xxr * (dxa *
abs(Xxi))))) + (( - 2 * (abs((Xxi * Xxi)) * (Xxr * (dxa * abs(Xxi))))) + ((2 *
((sgn(Xxi) * (sgn(Xxr) * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))) +
((sgn(Xxr) * ((Xxr * Xxr) * (dxa * abs((Xxi * Xxr))))) + ((sgn(Xxr) * sgn(Xxr))
* (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxi)))))))) + (( - 2 * (abs((Xxr * Xxr)) *
(Xxi * (dya * abs(Xxi))))) + ((2 * ((abs((Xxi * Xxi)) * (Xxi * (dya *
abs(Xxi)))) + (abs((Xxr * Xxr)) * (Xxi * (dxa * abs(Xxr)))))) + (( - 2 *
(abs((Xxi * Xxi)) * (Xxi * (dxa * abs(Xxr))))) + ((2 * (sgn(Xxr) * (Xxi * (Xxr
* (dxa * abs((Xxr * Xxr)))))))) + (( - 2 * (((sgn(Xxi) * sgn(Xxi)) * (Xxi *
((Xxr * Xxr) * (dya * abs(Xxi))))) + (sgn(Xxr) * (Xxi * (Xxr * (dxa * abs((Xxi
* Xxi)))))))) + ((2 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxa *
abs(Xxr)))))))) + (( - 2 * ((sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dya *
abs(Xxr))))))) + ((sgn(Xxr) * ((Xxi * Xxi) * (dxa * abs((Xxi * Xxr))))) +
((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi))))))) + ((2 *
((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) * (dya * abs(Xxi)))))) + (( - 2 *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa * abs(Xxr))))))) + (( - 4 *
(abs((Xxi * Xxr)) * (Xxr * (dya * abs(Xxi))))) + ((4 * (abs((Xxi * Xxr)) * (Xxr
* (dxa * abs(Xxr))))) + (( - 4 * ((abs((Xxi * Xxr)) * (Xxi * ((dxa * abs(Xxi))
+ (dya * abs(Xxr))))) + (sgn(Xxi) * ((Xxi * (Xxr * (dya * abs((Xxi * Xxr)))))) +
(sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dya * abs(Xxi)))))))) + ((4 * ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + (( - 4 * (((sgn(Xxi) *
```

```
sgn(Xxi)) * ((Xxi * Xxi) * (Xxr * (dya * abs(Xxr))))) + (sgn(Xxi) * (sgn(Xxr) *
((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi)))))))) + dba))))))))))))))))))));
        dybn = ((sgn(Xxi) * (((Xxi * Xxi) * (dyb * abs((Xxi * Xxi)))) + ((Xxr *
Xxr) * (dyb * abs((Xxr * Xxr)))))) + (( - 1 * (sgn(Xxi) * (((Xxi * Xxi) * (dyb
* abs((Xxr * Xxr)))) + ((Xxr * Xxr) * (dyb * abs((Xxi * Xxi))))))) + ((2 *
(abs((Xxr * Xxr)) * (Xxr * (dyb * abs(Xxr))))) + (( - 2 * (abs((Xxi * Xxi)) *
(Xxr * (dyb * abs(Xxr))))) + ((2 * (abs((Xxr * Xxr)) * (Xxr * (dxb *
abs(Xxi))))) + (( - 2 * (abs((Xxi * Xxi)) * (Xxr * (dxb * abs(Xxi))))) + ((2 *
((sgn(Xxi) * (sgn(Xxr) * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) +
((sgn(Xxr) * ((Xxr * Xxr) * (dxb * abs((Xxi * Xxr))))) + ((sgn(Xxr) * sgn(Xxr))
* (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxi)))))))) + (( - 2 * (abs((Xxr * Xxr)) *
(Xxi * (dyb * abs(Xxi))))) + ((2 * ((abs((Xxi * Xxi)) * (Xxi * (dyb *
abs(Xxi)))) + (abs((Xxr * Xxr)) * (Xxi * (dxb * abs(Xxr)))))) + (( - 2 *
(abs((Xxi * Xxi)) * (Xxi * (dxb * abs(Xxr))))) + ((2 * (sgn(Xxr) * (Xxi * (Xxr
* (dxb * abs((Xxr * Xxr)))))) + (( - 2 * (((sgn(Xxi) * sgn(Xxi)) * (Xxi *
((Xxr * Xxr) * (dyb * abs(Xxi))))) + (sgn(Xxr) * (Xxi * (Xxr * (dxb * abs((Xxi
* Xxi)))))))) + ((2 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxb *
abs(Xxr)))))) + (( - 2 * ((sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dyb *
abs(Xxr)))))) + ((sgn(Xxr) * ((Xxi * Xxi) * (dxb * abs((Xxi * Xxr))))) +
((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi)))))))) + ((2 *
((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) * (dyb * abs(Xxi)))))) + (( - 2 *
(sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb * abs(Xxr)))))) + (( - 4 *
(abs((Xxi * Xxr)) * (Xxr * (dyb * abs(Xxi))))) + ((4 * (abs((Xxi * Xxr)) * (Xxr
* (dxb * abs(Xxr))))) + (( - 4 * ((abs((Xxi * Xxr)) * (Xxi * ((dxb * abs(Xxi))
+ (dyb * abs(Xxr))))) + (sgn(Xxi) * ((Xxi * (Xxr * (dyb * abs((Xxi * Xxr)))))) +
(sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxi)))))))) + ((4 * ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + (( - 4 * (((sgn(Xxi) *
sgn(Xxi)) * ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxr))))) + (sgn(Xxi) * (sgn(Xxr) *
((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi)))))))) + dbb))))))))))))))))))));
      </derivative>
    </formula>
  </group>

  <group type="62" name="Mothbrot 5th 3x2">
    <formula power="5" glitch="1.0000000000000002e-4">
      <reference t='R'>
        Xrn = (Cr + (( - 1 * (abs(Xi) * (Xr * ((Xr * Xr) * abs(Xi))))) + ((2 *
(abs(Xi) * (Xi * ((Xi * Xi) * abs(Xr))))) + ((3 * (abs(Xi) * ((Xi * Xi) * (Xr *
abs(Xi))))) + (( - 3 * (abs(Xr) * ((Xi * Xi) * (Xr * abs(Xr))))) + (( - 6 *
(abs(Xi) * (Xi * ((Xr * Xr) * abs(Xr))))) + (abs(Xr) * (Xr * ((Xr * Xr) *
abs(Xr)))))))));
        Xin = (Ci + ((abs(Xi) * (Xi * ((Xi * Xi) * abs(Xi)))) + (( - 1 *
(abs(Xr) * (Xi * ((Xi * Xi) * abs(Xr))))) + ((3 * (abs(Xr) * (Xi * ((Xr * Xr) *
abs(Xr))))) + (( - 3 * (abs(Xi) * (Xi * ((Xr * Xr) * abs(Xi))))) + (( - 6 *
(abs(Xi) * ((Xi * Xi) * (Xr * abs(Xr))))) + (2 * (abs(Xi) * (Xr * ((Xr * Xr) *
abs(Xr)))))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs(Xr) * ((Xr * ((Xr * Xr) * diffabs(Xr,xr))) + (xr *
((xr * xr) * abs((Xr + xr)))))) + ((abs((Xr + xr)) * (Xr * ((Xr * Xr) *
diffabs(Xr,xr)))) + (( - 1 * ((abs(Xi) * ((Xr * ((Xr * Xr) * diffabs(Xi,xi))) +
(xr * ((xr * xr) * abs((Xi + xi)))))) + (abs((Xi + xi)) * ((Xr * ((Xr * Xr) *
diffabs(Xi,xi))) + (xr * ((xr * xr) * diffabs(Xi,xi))))))) + ((2 * ((abs(Xi) *
((Xi * ((Xi * Xi) * diffabs(Xr,xr))) + (xi * ((xi * xi) * abs((Xr + xr)))))) +
(abs((Xr + xr)) * ((Xi * ((Xi * Xi) * diffabs(Xi,xi))) + (xi * ((xi * xi) *
diffabs(Xi,xi))))))) + (( - 3 * ((abs(Xr) * ((xi * xi) * (xr * abs((Xr +
xr))))) + (abs((Xr + xr)) * ((xi * xi) * (xr * diffabs(Xr,xr)))))) + ((3 *
```

```
((abs(Xi) * ((xi * xi) * (xr * abs((Xi + xi))))) + ((abs(Xr) * (Xr * ((xr * xr)
* abs((Xr + xr))))) + ((abs((Xi + xi)) * ((xi * xi) * (xr * diffabs(Xi,xi)))) +
(abs((Xr + xr)) * (Xr * ((xr * xr) * diffabs(Xr,xr)))))))) + (( - 3 * ((abs(Xi)
* (Xr * ((xr * xr) * abs((Xi + xi))))) + ((abs(Xr) * (Xr * ((xi * xi) * abs((Xr
+ xr))))) + ((abs((Xi + xi)) * (Xr * ((xr * xr) * diffabs(Xi,xi)))) + (abs((Xr
+ xr)) * (Xr * ((xi * xi) * diffabs(Xr,xr)))))))) + ((3 * ((abs(Xi) * (Xr *
((xi * xi) * abs((Xi + xi))))) + ((abs(Xr) * ((Xr * Xr) * (xr * abs((Xr +
xr))))) + ((abs((Xi + xi)) * (Xr * ((xi * xi) * diffabs(Xi,xi)))) + (abs((Xr +
xr)) * ((Xr * Xr) * (xr * diffabs(Xr,xr)))))))) + (( - 3 * ((abs(Xi) * ((Xr *
Xr) * (xr * abs((Xi + xi))))) + ((abs(Xr) * ((Xi * Xi) * (xr * abs((Xr +
xr))))) + ((abs((Xi + xi)) * ((Xr * Xr) * (xr * diffabs(Xi,xi)))) + (abs((Xr +
xr)) * ((Xi * Xi) * (xr * diffabs(Xr,xr)))))))) + ((3 * (abs((Xi + xi)) * ((Xi
* Xi) * (xr * diffabs(Xi,xi))))) + (( - 3 * (((Xi * Xi) * (Xr *
diffabs(Xr,xr))) * (abs(Xr) + abs((Xr + xr))))) + ((3 * ((abs(Xi) * ((Xi * Xi)
* ((Xr * diffabs(Xi,xi)) + (xr * abs((Xi + xi)))))) + (abs((Xi + xi)) * ((Xi *
Xi) * (Xr * diffabs(Xi,xi)))))) + (( - 6 * ((abs(Xi) * ((Xi * ((xr * xr) *
abs((Xr + xr)))) + (((Xr * Xr) * (xi * abs((Xr + xr)))) + (xi * ((xr * xr) *
abs((Xr + xr))))))) + ((abs(Xr) * (Xi * (xi * (xr * abs((Xr + xr)))))) +
(abs((Xr + xr)) * ((Xi * ((xi * (xr * diffabs(Xr,xr))) + ((xr * xr) *
diffabs(Xi,xi)))) + (((Xr * Xr) * (xi * diffabs(Xi,xi))) + (xi * ((xr * xr) *
diffabs(Xi,xi))))))))) + ((6 * ((abs(Xi) * (Xi * (((xi * xi) * abs((Xr + xr)))
+ (xi * (xr * abs((Xi + xi))))))) + ((abs((Xi + xi)) * (Xi * (xi * (xr *
diffabs(Xi,xi))))) + (abs((Xr + xr)) * (Xi * ((xi * xi) * diffabs(Xi,xi)))))))
+ (( - 6 * ((abs(Xr) * (Xi * (Xr * (xi * abs((Xr + xr)))))) + (abs((Xr + xr)) *
(Xi * (Xr * (xi * diffabs(Xr,xr))))))) + ((6 * ((abs(Xi) * (Xi * (Xr * (xi *
abs((Xi + xi)))))) + (abs((Xi + xi)) * (Xi * (Xr * (xi * diffabs(Xi,xi))))))) +
(( - 6 * ((abs(Xi) * (Xi * ((Xr * Xr) * diffabs(Xr,xr)))) + (abs((Xr + xr)) *
(Xi * ((Xr * Xr) * diffabs(Xi,xi)))))) + ((6 * ((abs(Xi) * ((Xi * Xi) * (xi *
abs((Xr + xr))))) + (abs((Xr + xr)) * ((Xi * Xi) * (xi * diffabs(Xi,xi)))))) +
(( - 12 * ((abs(Xi) * ((Xi * (Xr * (xr * abs((Xr + xr))))) + (Xr * (xi * (xr *
abs((Xr + xr))))))) + (abs((Xr + xr)) * ((Xi * (Xr * (xr * diffabs(Xi,xi)))) +
(Xr * (xi * (xr * diffabs(Xi,xi))))))))) + (abs((Xr + xr)) * (xr * ((xr * xr) *
diffabs(Xr,xr))))))))))))))))))))));
        xin = (ci + ((abs(Xi) * ((Xi * ((Xi * Xi) * diffabs(Xi,xi))) + (xi *
((xi * xi) * abs((Xi + xi)))))) + ((abs((Xi + xi)) * ((Xi * ((Xi * Xi) *
diffabs(Xi,xi))) + (xi * ((xi * xi) * diffabs(Xi,xi))))) + (( - 1 * ((abs(Xr) *
((Xi * ((Xi * Xi) * diffabs(Xr,xr))) + (xi * ((xi * xi) * abs((Xr + xr)))))) +
(abs((Xr + xr)) * ((Xi * ((Xi * Xi) * diffabs(Xr,xr))) + (xi * ((xi * xi) *
diffabs(Xr,xr))))))) + ((2 * ((abs(Xi) * ((Xr * ((Xr * Xr) * diffabs(Xr,xr))) +
(xr * ((xr * xr) * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xr * ((Xr * Xr) *
diffabs(Xi,xi)))))) + ((3 * ((abs(Xr) * (xi * ((xr * xr) * abs((Xr + xr))))) +
(abs((Xr + xr)) * (xi * ((xr * xr) * diffabs(Xr,xr)))))) + (( - 3 * ((abs(Xi) *
(xi * ((xr * xr) * abs((Xi + xi))))) + (abs((Xi + xi)) * (xi * ((xr * xr) *
diffabs(Xi,xi)))))) + ((3 * ((abs(Xr) * ((Xr * Xr) * (xi * abs((Xr + xr)))))) +
(abs((Xr + xr)) * ((Xr * Xr) * (xi * diffabs(Xr,xr)))))) + (( - 3 * ((abs(Xi) *
((Xr * Xr) * (xi * abs((Xi + xi))))) + (abs((Xi + xi)) * ((Xr * Xr) * (xi *
diffabs(Xi,xi)))))) + ((3 * ((abs(Xr) * (Xi * ((xr * xr) * abs((Xr + xr)))))) +
(abs((Xr + xr)) * (Xi * ((xr * xr) * diffabs(Xr,xr)))))) + (( - 3 * ((abs(Xi) *
(Xi * ((xr * xr) * abs((Xi + xi))))) + ((abs(Xr) * (Xi * ((xi * xi) * abs((Xr +
xr))))) + ((abs((Xi + xi)) * (Xi * ((xr * xr) * diffabs(Xi,xi)))) + (abs((Xr +
xr)) * (Xi * ((xi * xi) * diffabs(Xr,xr))))))))) + ((3 * ((abs(Xi) * (Xi * ((xi
* xi) * abs((Xi + xi)))))) + ((abs(Xr) * (Xi * ((Xr * Xr) * diffabs(Xr,xr))))) +
((abs((Xi + xi)) * (Xi * ((xi * xi) * diffabs(Xi,xi)))) + (abs((Xr + xr)) * (Xi
* ((Xr * Xr) * diffabs(Xr,xr))))))))) + (( - 3 * ((abs(Xi) * (Xi * ((Xr * Xr) *
diffabs(Xi,xi)))) + ((abs(Xr) * ((Xi * Xi) * (xi * abs((Xr + xr)))))) +
((abs((Xi + xi)) * (Xi * ((Xr * Xr) * diffabs(Xi,xi)))) + (abs((Xr + xr)) *
((Xi * Xi) * (xi * diffabs(Xr,xr))))))))) + ((3 * ((abs(Xi) * ((Xi * Xi) * (xi *
```

```
      abs((Xi + xi))))) + (abs((Xi + xi)) * ((Xi * Xi) * (xi * diffabs(Xi,xi)))))) +
      (( - 6 * ((abs(Xi) * ((xi * xi) * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) *
3810  ((xi * xi) * (xr * diffabs(Xi,xi)))))) + ((6 * ((abs(Xi) * (Xr * ((xr * xr) *
      abs((Xr + xr))))) + ((abs(Xr) * (Xr * (xi * (xr * abs((Xr + xr)))))) + (abs((Xr
      + xr)) * (Xr * ((xi * (xr * diffabs(Xr,xr))) + ((xr * xr) *
      diffabs(Xi,xi)))))))) + (( - 6 * ((abs(Xi) * (Xr * (((xi * xi) * abs((Xr +
      xr))) + (xi * (xr * abs((Xi + xi)))))))) + ((abs((Xi + xi)) * (Xr * (xi * (xr *
3815  diffabs(Xi,xi))))) + (abs((Xr + xr)) * (Xr * ((xi * xi) * diffabs(Xi,xi)))))))
      + ((6 * ((abs(Xi) * ((Xr * Xr) * (xr * abs((Xr + xr))))) + ((abs(Xr) * (Xi *
      (Xr * (xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * ((Xi * (Xr * (xr *
      diffabs(Xr,xr)))) + ((Xr * Xr) * (xr * diffabs(Xi,xi)))))))) + (( - 6 *
      ((abs(Xi) * (((Xi * Xi) * ((Xr * diffabs(Xr,xr)) + (xr * abs((Xr + xr))))) +
3820  (Xi * (Xr * (xr * abs((Xi + xi)))))))) + ((abs((Xi + xi)) * (Xi * (Xr * (xr *
      diffabs(Xi,xi))))) + (abs((Xr + xr)) * ((Xi * Xi) * (diffabs(Xi,xi) * (Xr +
      xr)))))))) + (( - 12 * ((abs(Xi) * (Xi * ((Xr * (xi * abs((Xr + xr)))) + (xi *
      (xr * abs((Xr + xr)))))))) + (abs((Xr + xr)) * (Xi * ((Xr * (xi *
      diffabs(Xi,xi))) + (xi * (xr * diffabs(Xi,xi)))))))) + (2 * (abs((Xr + xr)) *
3825  (xr * ((xr * xr) * diffabs(Xi,xi)))))))))))))))))))))))));
        </perturbation>
        <derivative t='M>
          dxan = (( - 2 * (sgn(Xxi) * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxi))))))
      + ((2 * ((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dya * abs(Xxr))))) + (sgn(Xxr) *
3830  ((Xxi * ((Xxi * Xxi) * (dxa * abs(Xxi)))) + (Xxr * ((Xxr * Xxr) * (dxa *
      abs(Xxr)))))))) + ((3 * (abs(Xxr) * ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 3
      * ((abs(Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxi)))) + (abs(Xxr) * ((Xxi * Xxi) *
      (dxa * abs(Xxr)))))) + ((3 * (abs(Xxi) * ((Xxi * Xxi) * (dxa * abs(Xxi))))) +
      (( - 6 * ((abs(Xxi) * ((Xxr * Xxr) * (dya * abs(Xxr)))) + (abs(Xxr) * (Xxi *
3835  (Xxr * (dya * abs(Xxr))))))) + ((6 * (abs(Xxi) * (Xxi * (Xxr * (dya *
      abs(Xxi)))))) + (( - 6 * (sgn(Xxi) * (Xxi * ((Xxr * Xxr) * (dya * abs(Xxr))))))
      + ((6 * ((abs(Xxi) * ((Xxi * Xxi) * (dya * abs(Xxr)))) + (sgn(Xxi) * ((Xxi *
      Xxi) * (Xxr * (dya * abs(Xxi))))))) + (( - 6 * (sgn(Xxr) * (((Xxi * Xxi) * (Xxr
      * (dxa * abs(Xxr)))) + (Xxi * ((Xxr * Xxr) * (dxa * abs(Xxi))))))) + (( - 12 *
3840  (abs(Xxi) * (Xxi * (Xxr * (dxa * abs(Xxr)))))) + daa)))))))))));
          dxbn = (( - 2 * (sgn(Xxi) * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxi))))))
      + ((2 * ((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dyb * abs(Xxr))))) + (sgn(Xxr) *
      ((Xxi * ((Xxi * Xxi) * (dxb * abs(Xxi)))) + (Xxr * ((Xxr * Xxr) * (dxb *
      abs(Xxr)))))))) + ((3 * (abs(Xxr) * ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 3
3845  * ((abs(Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxi)))) + (abs(Xxr) * ((Xxi * Xxi) *
      (dxb * abs(Xxr)))))) + ((3 * (abs(Xxi) * ((Xxi * Xxi) * (dxb * abs(Xxi))))) +
      (( - 6 * ((abs(Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxr)))) + (abs(Xxr) * (Xxi *
      (Xxr * (dyb * abs(Xxr))))))) + ((6 * (abs(Xxi) * (Xxi * (Xxr * (dyb *
      abs(Xxi)))))) + (( - 6 * (sgn(Xxi) * (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxr))))))
3850  + ((6 * ((abs(Xxi) * ((Xxi * Xxi) * (dyb * abs(Xxr)))) + (sgn(Xxi) * ((Xxi *
      Xxi) * (Xxr * (dyb * abs(Xxi))))))) + (( - 6 * (sgn(Xxr) * (((Xxi * Xxi) * (Xxr
      * (dxb * abs(Xxr)))) + (Xxi * ((Xxr * Xxr) * (dxb * abs(Xxi))))))) + (( - 12 *
      (abs(Xxi) * (Xxi * (Xxr * (dxb * abs(Xxr)))))) + dab)))))))))));
          dyan = ((2 * ((sgn(Xxi) * ((Xxi * ((Xxi * Xxi) * (dya * abs(Xxi)))) +
3855  (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))) + (sgn(Xxr) * (Xxr * ((Xxr * Xxr) *
      (dxa * abs(Xxi))))))) + (( - 2 * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa *
      abs(Xxr)))))) + ((3 * (abs(Xxr) * ((Xxr * Xxr) * (dya * abs(Xxr))))) + (( - 3 *
      ((abs(Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi)))) + (abs(Xxr) * ((Xxi * Xxi) *
      (dya * abs(Xxr)))))) + ((3 * (abs(Xxi) * ((Xxi * Xxi) * (dya * abs(Xxi))))) +
3860  ((6 * ((abs(Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr)))) + (abs(Xxr) * (Xxi * (Xxr
      * (dxa * abs(Xxr))))))) + (( - 6 * ((abs(Xxi) * (Xxi * (Xxr * (dxa *
      abs(Xxi))))) + (sgn(Xxi) * (Xxi * ((Xxr * Xxr) * (dya * abs(Xxi))))))) + ((6 *
      (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + (( - 6 * ((abs(Xxi) *
      ((Xxi * Xxi) * (dxa * abs(Xxr)))) + ((sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dya *
```

```
abs(Xxr))))) + (sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi)))))))) + (( -
12 * (abs(Xxi) * (Xxi * (Xxr * (dya * abs(Xxr)))))) + dba)))))))));
        dybn = ((2 * ((sgn(Xxi) * ((Xxi * ((Xxi * Xxi) * (dyb * abs(Xxi)))) +
(Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) + (sgn(Xxr) * (Xxr * ((Xxr * Xxr) *
(dxb * abs(Xxi))))))) + (( - 2 * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb *
abs(Xxr)))))) + ((3 * (abs(Xxr) * ((Xxr * Xxr) * (dyb * abs(Xxr))))) + (( - 3 *
((abs(Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi)))) + (abs(Xxr) * ((Xxi * Xxi) *
(dyb * abs(Xxr)))))) + ((3 * (abs(Xxi) * ((Xxi * Xxi) * (dyb * abs(Xxi))))) +
((6 * ((abs(Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr)))) + (abs(Xxr) * (Xxi * (Xxr
* (dxb * abs(Xxr))))))) + (( - 6 * ((abs(Xxi) * (Xxi * (Xxr * (dxb *
abs(Xxi))))) + (sgn(Xxi) * (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxi))))))) + ((6 *
(sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + (( - 6 * ((abs(Xxi) *
((Xxi * Xxi) * (dxb * abs(Xxr)))) + ((sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dyb *
abs(Xxr))))) + (sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi)))))))) + (( -
12 * (abs(Xxi) * (Xxi * (Xxr * (dyb * abs(Xxr)))))) + dbb)))))))));
      </derivative>
    </formula>
  </group>

  <group type="63" name="Mothbrot 5th 4x1">
    <formula power="5" glitch="1.0000000000000002e-4">
      <reference t='R'>
        Xrn = (Cr + ((((Xi * Xi) * (Xi * Xi)) * abs(Xr)) + (( - 4 * (Xi * (Xr *
((Xr * Xr) * abs(Xi))))) + ((4 * (Xi * ((Xi * Xi) * (Xr * abs(Xi))))) + (( - 6
* ((Xi * Xi) * ((Xr * Xr) * abs(Xr)))) + (((Xr * Xr) * (Xr * Xr)) *
abs(Xr)))))));
        Xin = (Ci + ((((Xi * Xi) * (Xi * Xi)) * abs(Xi)) + ((4 * (Xi * (Xr *
((Xr * Xr) * abs(Xr))))) + (( - 4 * (Xi * ((Xi * Xi) * (Xr * abs(Xr))))) + (( -
6 * ((Xi * Xi) * ((Xr * Xr) * abs(Xi)))) + (((Xr * Xr) * (Xr * Xr)) *
abs(Xi)))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((diffabs(Xr,xr) * (((Xi * Xi) * (Xi * Xi)) + ((Xr * Xr) *
(Xr * Xr)))) + ((((xi * xi) * (xi * xi)) * abs((Xr + xr))) + (( - 4 * (xi * (xr
* ((xr * xr) * abs((Xi + xi)))))) + ((4 * ((Xr * (((Xr * Xr) * (xr * abs((Xr +
xr)))) + ((xi * ((xi * xi) * abs((Xi + xi)))) + (xr * ((xr * xr) * abs((Xr +
xr))))))) + (xi * ((xi * xi) * (xr * abs((Xi + xi)))))))) + (( - 4 * ((Xi * (xr
* ((xr * xr) * abs((Xi + xi))))) + (Xr * ((Xr * Xr) * (xi * abs((Xi + xi))))))))
+ ((4 * (Xi * (xi * ((xi * xi) * abs((Xr + xr)))))) + (( - 4 * (Xi * (Xr * ((Xr
* Xr) * diffabs(Xi,xi))))) + ((4 * (Xi * ((Xi * Xi) * ((Xr * diffabs(Xi,xi)) +
((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))))) + (( - 6 * ((xi * xi) *
((xr * xr) * abs((Xr + xr))))) + ((6 * ((Xr * Xr) * ((xr * xr) * abs((Xr +
xr))))) + (( - 6 * (((Xi * Xi) * (((Xr * Xr) * diffabs(Xr,xr)) + ((xr * xr) *
abs((Xr + xr))))) + ((Xr * Xr) * ((xi * xi) * abs((Xr + xr)))))) + ((6 * ((Xi *
Xi) * ((xi * xi) * abs((Xr + xr))))) + (( - 12 * ((Xi * (xi * ((xr * xr) *
abs((Xr + xr))))) + (((Xr * Xr) * (xi * (xr * abs((Xi + xi))))) + (Xr * (((xi *
xi) * (xr * abs((Xr + xr)))) + (xi * ((xr * xr) * abs((Xi + xi))))))))) + ((12
* (Xi * ((xi * xi) * (xr * abs((Xi + xi)))))) + (( - 12 * (Xi * (Xr * ((xr *
xr) * abs((Xi + xi)))))) + ((12 * (Xi * (Xr * ((xi * xi) * abs((Xi + xi)))))) +
(( - 12 * (Xi * ((Xr * Xr) * ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))))
+ ((12 * ((Xi * Xi) * (xi * (xr * abs((Xi + xi)))))) + (( - 12 * ((Xi * Xi) *
(Xr * (xr * abs((Xr + xr)))))) + ((12 * ((Xi * Xi) * (Xr * (xi * abs((Xi +
xi)))))) + (( - 24 * (Xi * (Xr * (xi * (xr * abs((Xr + xr))))))) + (((xr * xr)
* (xr * xr)) * abs((Xr + xr))))))))))))))))))))));
        xin = (ci + ((diffabs(Xi,xi) * (((Xi * Xi) * (Xi * Xi)) + ((Xr * Xr) *
(Xr * Xr)))) + ((((xi * xi) * (xi * xi)) * abs((Xi + xi))) + ((4 * (xi * (xr *
((xr * xr) * abs((Xr + xr)))))) + (( - 4 * (xi * ((xi * xi) * (xr * abs((Xr +
```

```
xr))))))) + ((4 * (Xr * (xr * ((xr * xr) * abs((Xi + xi)))))) + (( - 4 * (Xr *
(xi * ((xi * xi) * abs((Xr + xr)))))) + ((4 * ((Xi * ((Xr * ((Xr * Xr) *
diffabs(Xr,xr))) + ((xi * ((xi * xi) * abs((Xi + xi)))) + (xr * ((xr * xr) *
abs((Xr + xr))))))) + (Xr * ((Xr * Xr) * ((xi * abs((Xr + xr))) + (xr * abs((Xi
+ xi)))))))) + (( - 4 * (Xi * ((Xi * Xi) * ((Xr * diffabs(Xr,xr)) + (xr *
abs((Xr + xr))))))) + ((4 * (Xi * ((Xi * Xi) * (xi * abs((Xi + xi)))))) + (( -
6 * ((xi * xi) * ((xr * xr) * abs((Xi + xi))))) + ((6 * ((Xr * Xr) * ((xr * xr)
* abs((Xi + xi))))) + (( - 6 * (((Xi * Xi) * (((Xr * Xr) * diffabs(Xi,xi)) +
((xr * xr) * abs((Xi + xi))))) + ((Xr * Xr) * ((xi * xi) * abs((Xi + xi)))))) +
((6 * ((Xi * Xi) * ((xi * xi) * abs((Xi + xi))))) + ((12 * (Xr * (xi * ((xr *
xr) * abs((Xr + xr)))))) + (( - 12 * (Xr * ((xi * xi) * (xr * abs((Xi +
xi)))))) + ((12 * ((Xr * Xr) * (xi * (xr * abs((Xr + xr)))))) + (( - 12 * (Xi *
(((xi * xi) * (xr * abs((Xr + xr)))) + (xi * ((xr * xr) * abs((Xi + xi))))))) +
((12 * (Xi * (Xr * ((xr * xr) * abs((Xr + xr)))))) + (( - 12 * (Xi * (Xr * ((xi
* xi) * abs((Xr + xr)))))) + ((12 * (Xi * ((Xr * Xr) * (xr * abs((Xr + xr))))))
+ (( - 12 * (((Xi * Xi) * ((Xr * ((xi * abs((Xr + xr))) + (xr * abs((Xi +
xi))))) + (xi * (xr * abs((Xr + xr)))))) + (Xi * ((Xr * Xr) * (xi * abs((Xi +
xi))))))) + (( - 24 * (Xi * (Xr * (xi * (xr * abs((Xi + xi))))))) + (((xr * xr)
* (xr * xr)) * abs((Xi + xi))))))))))))))))))))))));
      </perturbation>
      <derivative t="M">
        dxan = (((dxa * sgn(Xxr)) * (((Xxi * Xxi) * (Xxi * Xxi)) + ((Xxr * Xxr)
* (Xxr * Xxr)))) + (( - 4 * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxi))))) + ((4 *
(Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 4 * (Xxi * (Xxr * ((Xxr * Xxr)
* (dya * sgn(Xxi)))))) + ((4 * (Xxi * ((Xxi * Xxi) * ((Xxr * (dya * sgn(Xxi)))
+ ((dxa * abs(Xxi)) + (dya * abs(Xxr))))))) + (( - 6 * ((Xxi * Xxi) * ((Xxr *
Xxr) * (dxa * sgn(Xxr))))) + (( - 12 * (Xxi * ((Xxr * Xxr) * ((dxa * abs(Xxi))
+ (dya * abs(Xxr)))))) + ((12 * ((Xxi * Xxi) * (Xxr * (dya * abs(Xxi))))) + ((
- 12 * ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxr))))) + daa)))))))));
        dxbn = (((dxb * sgn(Xxr)) * (((Xxi * Xxi) * (Xxi * Xxi)) + ((Xxr * Xxr)
* (Xxr * Xxr)))) + (( - 4 * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxi))))) + ((4 *
(Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 4 * (Xxi * (Xxr * ((Xxr * Xxr)
* (dyb * sgn(Xxi)))))) + ((4 * (Xxi * ((Xxi * Xxi) * ((Xxr * (dyb * sgn(Xxi)))
+ ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))))) + (( - 6 * ((Xxi * Xxi) * ((Xxr *
Xxr) * (dxb * sgn(Xxr))))) + (( - 12 * (Xxi * ((Xxr * Xxr) * ((dxb * abs(Xxi))
+ (dyb * abs(Xxr)))))) + ((12 * ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi))))) + ((
- 12 * ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxr))))) + dab)))))))));
        dyan = (((dya * sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) + ((Xxr * Xxr)
* (Xxr * Xxr)))) + ((4 * ((Xxi * (((Xxi * Xxi) * (dya * abs(Xxi))) + (Xxr *
((Xxr * Xxr) * (dxa * sgn(Xxr)))))) + (Xxr * ((Xxr * Xxr) * ((dxa * abs(Xxi)) +
(dya * abs(Xxr))))))) + (( - 4 * (Xxi * ((Xxi * Xxi) * ((Xxr * (dxa *
sgn(Xxr))) + (dxa * abs(Xxr)))))) + (( - 6 * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya
* sgn(Xxi))))) + (( - 12 * (Xxi * ((Xxr * Xxr) * (dya * abs(Xxi))))) + ((12 *
(Xxi * ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 12 * ((Xxi * Xxi) * (Xxr *
((dxa * abs(Xxi)) + (dya * abs(Xxr)))))) + dba)))))));
        dybn = (((dyb * sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) + ((Xxr * Xxr)
* (Xxr * Xxr)))) + ((4 * ((Xxi * (((Xxi * Xxi) * (dyb * abs(Xxi))) + (Xxr *
((Xxr * Xxr) * (dxb * sgn(Xxr)))))) + (Xxr * ((Xxr * Xxr) * ((dxb * abs(Xxi)) +
(dyb * abs(Xxr))))))) + (( - 4 * (Xxi * ((Xxi * Xxi) * ((Xxr * (dxb *
sgn(Xxr))) + (dxb * abs(Xxr)))))) + (( - 6 * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb
* sgn(Xxi))))) + (( - 12 * (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxi))))) + ((12 *
(Xxi * ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 12 * ((Xxi * Xxi) * (Xxr *
((dxb * abs(Xxi)) + (dyb * abs(Xxr)))))) + dbb)))))));
      </derivative>
    </formula>
  </group>
```

```
  <group type="64" name="Mothbrot 6th 1x5">
    <formula power="6" glitch="1.0000000000000002e-4">
      <reference t='R'>
        Xrn = (Cr + ((abs(((Xi * Xi) * (Xi * Xi))) * (Xr * abs(Xr))) + (( - 1 *
((Xi * abs(Xi)) * (abs(((Xi * Xi) * (Xi * Xi))) + abs(((Xr * Xr) * (Xr *
Xr)))))) + (( - 4 * (abs((Xi * (Xr * (Xr * Xr)))) * (Xr * abs(Xi)))) + ((4 *
(abs((Xi * ((Xi * Xi) * Xr))) * (Xr * abs(Xi)))) + (( - 4 * (abs((Xi * (Xr *
(Xr * Xr)))) * (Xi * abs(Xr)))) + ((4 * (abs((Xi * ((Xi * Xi) * Xr))) * (Xi *
abs(Xr)))) + (( - 6 * (abs(((Xi * Xi) * (Xr * Xr))) * (Xr * abs(Xr)))) + ((6 *
(abs(((Xi * Xi) * (Xr * Xr))) * (Xi * abs(Xi)))) + (abs(((Xr * Xr) * (Xr *
Xr))) * (Xr * abs(Xr))))))))))));
        Xin = (Ci + ((abs(((Xi * Xi) * (Xi * Xi))) * ((Xi * abs(Xr)) + (Xr *
abs(Xi)))) + ((abs(((Xr * Xr) * (Xr * Xr))) * (Xi * abs(Xr))) + ((4 * (abs((Xi
* (Xr * (Xr * Xr)))) * (Xr * abs(Xr)))) + (( - 4 * ((abs((Xi * ((Xi * Xi) *
Xr))) * (Xr * abs(Xr))) + (abs((Xi * (Xr * (Xr * Xr)))) * (Xi * abs(Xi))))) +
((4 * (abs((Xi * ((Xi * Xi) * Xr))) * (Xi * abs(Xi)))) + (( - 6 * (abs(((Xi *
Xi) * (Xr * Xr))) * ((Xi * abs(Xr)) + (Xr * abs(Xi))))) + (abs(((Xr * Xr) * (Xr
* Xr))) * (Xr * abs(Xi))))))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs(((Xi * Xi) * (Xi * Xi))) * ((Xr * diffabs(Xr,xr)) +
(xr * abs((Xr + xr))))) + ((abs(((Xr * Xr) * (Xr * Xr))) * (Xr *
diffabs(Xr,xr))) + ((abs((Xr + xr)) * ((diffabs(((Xi * Xi) * (Xi * Xi)),((4 *
(Xi * (xi * ((Xi * Xi) + (xi * xi))))) + ((6 * ((Xi * Xi) * (xi * xi))) + ((xi
* xi) * (xi * xi))))) + diffabs(((Xr * Xr) * (Xr * Xr)),((4 * (Xr * (xr * ((Xr
* Xr) + (xr * xr))))) + ((6 * ((Xr * Xr) * (xr * xr))) + ((xr * xr) * (xr *
xr))))))) * (Xr + xr))) + (( - 1 * ((abs(Xi) * (Xi * (diffabs(((Xi * Xi) * (Xi *
Xi)),((4 * (Xi * (xi * ((Xi * Xi) + (xi * xi))))) + ((6 * ((Xi * Xi) * (xi *
xi))) + ((xi * xi) * (xi * xi))))) + diffabs(((Xr * Xr) * (Xr * Xr)),((4 * (Xr
* (xr * ((Xr * Xr) + (xr * xr))))) + ((6 * ((Xr * Xr) * (xr * xr))) + ((xr *
xr) * (xr * xr))))))))) + (((Xi * diffabs(Xi,xi)) + (xi * abs((Xi + xi)))) *
(abs((((Xi * Xi) * (Xi * Xi)) + ((4 * (Xi * (xi * ((Xi * Xi) + (xi * xi))))) +
((6 * ((Xi * Xi) * (xi * xi))) + ((xi * xi) * (xi * xi)))))) + abs((((Xr * Xr)
* (Xr * Xr)) + ((4 * (Xr * (xr * ((Xr * Xr) + (xr * xr))))) + ((6 * ((Xr * Xr)
* (xr * xr))) + ((xr * xr) * (xr * xr))))))))))) + (( - 4 * (abs((Xi * (Xr * (Xr
* Xr)))) * (xr * abs((Xi + xi))))) + ((4 * (abs((Xi * ((Xi * Xi) * Xr))) * (xr
* abs((Xi + xi))))) + (( - 4 * (abs((Xi + xi)) * (xr * diffabs((Xi * (Xr * (Xr
* Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) * xi)) + ((3 * ((Xi *
(((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi * xr)) + (Xr * (xi *
(xr * xr))))))) + (xi * (xr * (xr * xr))))))))))) + ((4 * (abs((Xi + xi)) * (xr *
diffabs((Xi * ((Xi * Xi) * Xr)),((Xi * ((Xi * Xi) * xr)) + ((Xr * (xi * (xi *
xi))) + ((3 * (((Xi * Xi) * (xi * (Xr + xr))) + (Xi * ((xi * xi) * (Xr +
xr))))) + (xi * ((xi * xi) * xr))))))))) + (( - 4 * (abs((Xr + xr)) * (xi *
diffabs((Xi * (Xr * (Xr * Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) *
xi)) + ((3 * ((Xi * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi
* xr)) + (Xr * (xi * (xr * xr))))))) + (xi * (xr * (xr * xr))))))))))) + ((4 *
(abs((Xr + xr)) * (xi * diffabs((Xi * ((Xi * Xi) * Xr)),((Xi * ((Xi * Xi) *
xr)) + ((Xr * (xi * (xi * xi))) + ((3 * (((Xi * Xi) * (xi * (Xr + xr))) + (Xi *
((xi * xi) * (Xr + xr))))) + (xi * ((xi * xi) * xr))))))))) + (( - 4 * (abs((Xi
* (Xr * (Xr * Xr)))) * (xi * abs((Xr + xr))))) + ((4 * (abs((Xi * ((Xi * Xi) *
Xr))) * (xi * abs((Xr + xr))))) + (( - 4 * (abs((Xi * (Xr * (Xr * Xr)))) * (Xr
* diffabs(Xi,xi)))) + ((4 * (abs((Xi * ((Xi * Xi) * Xr))) * (Xr *
diffabs(Xi,xi)))) + (( - 4 * (abs((Xi + xi)) * (Xr * diffabs((Xi * (Xr * (Xr *
Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) * xi)) + ((3 * ((Xi * (((Xr
* Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi * xr)) + (Xr * (xi * (xr *
xr))))))) + (xi * (xr * (xr * xr))))))))))) + ((4 * (abs((Xi + xi)) * (Xr *
diffabs((Xi * ((Xi * Xi) * Xr)),((Xi * ((Xi * Xi) * xr)) + ((Xr * (xi * (xi *
```

```
xi))) + ((3 * (((Xi * Xi) * (xi * (Xr + xr))) + (Xi * ((xi * xi) * (Xr +
xr))))) + (xi * ((xi * xi) * xr)))))))))) + (( - 4 * (abs((Xr + xr)) * (Xi *
diffabs((Xi * (Xr * (Xr * Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) *
xi)) + ((3 * ((Xi * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi
* xr)) + (Xr * (xi * (xr * xr)))))) + (xi * (xr * (xr * xr))))))))))) + ((4 *
(abs((Xr + xr)) * (Xi * diffabs((Xi * ((Xi * Xi) * Xr)),((Xi * ((Xi * Xi) *
xr)) + ((Xr * (xi * (xi * xi))) + ((3 * (((Xi * Xi) * (xi * (Xr + xr))) + (Xi *
((xi * xi) * (Xr + xr))))) + (xi * ((xi * xi) * xr)))))))))) + (( - 4 * (abs((Xi
* (Xr * (Xr * Xr)))) * (Xi * diffabs(Xr,xr)))) + ((4 * (abs((Xi * ((Xi * Xi) *
Xr))) * (Xi * diffabs(Xr,xr)))) + (( - 6 * ((abs(((Xi * Xi) * (Xr * Xr))) * (xr
* abs((Xr + xr)))) + (abs((Xr + xr)) * (xr * diffabs(((Xi * Xi) * (Xr *
Xr)),(((Xi * Xi) * (xr * xr)) + (((Xr * Xr) * (xi * xi)) + ((2 * (((Xi * Xi) *
(Xr * xr)) + ((Xi * (xi * ((Xr * Xr) + (xr * xr)))) + (Xr * ((xi * xi) *
xr))))) + ((4 * (Xi * (Xr * (xi * xr)))) + ((xi * xi) * (xr * xr))))))))))) +
((6 * ((abs(((Xi * Xi) * (Xr * Xr))) * (xi * abs((Xi + xi)))) + (abs((Xi + xi))
* (xi * diffabs(((Xi * Xi) * (Xr * Xr)),(((Xi * Xi) * (xr * xr)) + (((Xr * Xr)
* (xi * xi)) + ((2 * (((Xi * Xi) * (Xr * xr)) + ((Xi * (xi * ((Xr * Xr) + (xr *
xr)))) + (Xr * ((xi * xi) * xr))))) + ((4 * (Xi * (Xr * (xi * xr)))) + ((xi *
xi) * (xr * xr))))))))))) + (( - 6 * ((abs(((Xi * Xi) * (Xr * Xr))) * (Xr *
diffabs(Xr,xr))) + (abs((Xr + xr)) * (Xr * diffabs(((Xi * Xi) * (Xr *
Xr)),(((Xi * Xi) * (xr * xr)) + (((Xr * Xr) * (xi * xi)) + ((2 * (((Xi * Xi) *
(Xr * xr)) + ((Xi * (xi * ((Xr * Xr) + (xr * xr)))) + (Xr * ((xi * xi) *
xr))))) + ((4 * (Xi * (Xr * (xi * xr)))) + ((xi * xi) * (xr * xr))))))))))) +
((6 * ((abs(((Xi * Xi) * (Xr * Xr))) * (Xi * diffabs(Xi,xi))) + (abs((Xi + xi))
* (Xi * diffabs(((Xi * Xi) * (Xr * Xr)),(((Xi * Xi) * (xr * xr)) + (((Xr * Xr)
* (xi * xi)) + ((2 * (((Xi * Xi) * (Xr * xr)) + ((Xi * (xi * ((Xr * Xr) + (xr *
xr)))) + (Xr * ((xi * xi) * xr))))) + ((4 * (Xi * (Xr * (xi * xr)))) + ((xi *
xi) * (xr * xr))))))))))) + (abs(((Xr * Xr) * (Xr * Xr))) * (xr * abs((Xr +
xr)))))))))))))))))))))))))));
        xin = (ci + ((abs(((Xi * Xi) * (Xi * Xi))) * ((Xi * diffabs(Xr,xr)) +
((Xr * diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi))))))) +
((abs(((Xr * Xr) * (Xr * Xr))) * ((Xi * diffabs(Xr,xr)) + ((Xr *
diffabs(Xi,xi)) + (xi * abs((Xr + xr)))))) + ((abs((Xi + xi)) * ((diffabs(((Xi
* Xi) * (Xi * Xi)),((4 * (Xi * (xi * ((Xi * Xi) + (xi * xi))))) + ((6 * ((Xi *
Xi) * (xi * xi))) + ((xi * xi) * (xi * xi))))) + diffabs(((Xr * Xr) * (Xr *
Xr)),((4 * (Xr * (xr * ((Xr * Xr) + (xr * xr))))) + ((6 * ((Xr * Xr) * (xr *
xr))) + ((xr * xr) * (xr * xr)))))) * (Xr + xr))) + ((abs((Xr + xr)) *
((diffabs(((Xi * Xi) * (Xi * Xi)),((4 * (Xi * (xi * ((Xi * Xi) + (xi * xi)))))
+ ((6 * ((Xi * Xi) * (xi * xi))) + ((xi * xi) * (xi * xi))))) + diffabs(((Xr *
Xr) * (Xr * Xr)),((4 * (Xr * (xr * ((Xr * Xr) + (xr * xr))))) + ((6 * ((Xr *
Xr) * (xr * xr))) + ((xr * xr) * (xr * xr)))))) * (Xi + xi))) + ((4 * (abs((Xr
+ xr)) * (xr * diffabs((Xi * (Xr * (Xr * Xr))),((Xi * (xr * (xr * xr))) + ((Xr
* ((Xr * Xr) * xi)) + ((3 * ((Xi * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) +
(((Xr * Xr) * (xi * xr)) + (Xr * (xi * (xr * xr)))))) + (xi * (xr * (xr *
xr))))))))) + (( - 4 * (abs((Xr + xr)) * (xr * diffabs((Xi * ((Xi * Xi) *
Xr)),((Xi * ((Xi * Xi) * xr)) + ((Xr * (xi * (xi * xi))) + ((3 * (((Xi * Xi) *
(xi * (Xr + xr))) + (Xi * ((xi * xi) * (Xr + xr))))) + (xi * ((xi * xi) *
xr)))))))))) + ((4 * (abs((Xi * (Xr * (Xr * Xr)))) * (xr * abs((Xr + xr))))) +
(( - 4 * ((abs((Xi * ((Xi * Xi) * Xr))) * (xr * abs((Xr + xr)))) + (abs((Xi *
(Xr * (Xr * Xr)))) * (xi * abs((Xi + xi)))))) + ((4 * (abs((Xi * ((Xi * Xi) *
Xr))) * (xi * abs((Xi + xi))))) + (( - 4 * (abs((Xi + xi)) * (xi * diffabs((Xi
* (Xr * (Xr * Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) * xi)) + ((3 *
((Xi * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi * xr)) + (Xr
* (xi * (xr * xr))))))) + (xi * (xr * (xr * xr))))))))))) + ((4 * ((abs((Xi +
xi)) * (xi * diffabs((Xi * ((Xi * Xi) * Xr)),((Xi * ((Xi * Xi) * xr)) + ((Xr *
(xi * (xi * xi))) + ((3 * (((Xi * Xi) * (xi * (Xr + xr))) + (Xi * ((xi * xi) *
(Xr + xr))))) + (xi * ((xi * xi) * xr))))))))) + (abs((Xr + xr)) * (Xr *
```

```
diffabs((Xi * (Xr * (Xr * Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) *
xi)) + ((3 * ((Xi * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi
* xr)) + (Xr * (xi * (xr * xr)))))) + (xi * (xr * (xr * xr))))))))))) + (( - 4
* (abs((Xr + xr)) * (Xr * diffabs((Xi * ((Xi * Xi) * Xr)),((Xi * ((Xi * Xi) *
xr)) + ((Xr * (xi * (xi * xi))) + ((3 * (((Xi * Xi) * (xi * (Xr + xr))) + (Xi *
((xi * xi) * (Xr + xr))))) + (xi * ((xi * xi) * xr)))))))))) + ((4 * (abs((Xi *
(Xr * (Xr * Xr)))) * (Xr * diffabs(Xr,xr)))) + (( - 4 * ((abs((Xi * ((Xi * Xi)
* Xr))) * (Xr * diffabs(Xr,xr))) + (abs((Xi * (Xr * (Xr * Xr)))) * (Xi *
diffabs(Xi,xi))))) + ((4 * (abs((Xi + xi)) * (Xi * diffabs((Xi * ((Xi * Xi) *
Xr)),((Xi * ((Xi * Xi) * xr)) + ((Xr * (xi * (xi * xi))) + ((3 * (((Xi * Xi) *
(xi * (Xr + xr))) + (Xi * ((xi * xi) * (Xr + xr))))) + (xi * ((xi * xi) *
xr)))))))))) + (( - 4 * (abs((Xi + xi)) * (Xi * diffabs((Xi * (Xr * (Xr *
Xr))),((Xi * (xr * (xr * xr))) + ((Xr * ((Xr * Xr) * xi)) + ((3 * ((Xi * (((Xr
* Xr) * xr) + (Xr * (xr * xr)))) + (((Xr * Xr) * (xi * xr)) + (Xr * (xi * (xr *
xr)))))) + (xi * (xr * (xr * xr))))))))))) + ((4 * (abs((Xi * ((Xi * Xi) * Xr)))
* (Xi * diffabs(Xi,xi)))) + (( - 6 * ((abs(((Xi * Xi) * (Xr * Xr))) * ((Xi *
diffabs(Xr,xr)) + ((Xr * diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr *
abs((Xi + xi)))))))) + ((abs((Xi + xi)) * (diffabs(((Xi * Xi) * (Xr * Xr)),(((Xi
* Xi) * (xr * xr)) + (((Xr * Xr) * (xi * xi)) + ((2 * (((Xi * Xi) * (Xr * xr))
+ ((Xi * (xi * ((Xr * Xr) + (xr * xr)))) + (Xr * ((xi * xi) * xr))))) + ((4 *
(Xi * (Xr * (xi * xr)))) + ((xi * xi) * (xr * xr)))))))) * (Xr + xr))) +
(abs((Xr + xr)) * (diffabs(((Xi * Xi) * (Xr * Xr)),(((Xi * Xi) * (xr * xr)) +
(((Xr * Xr) * (xi * xi)) + ((2 * (((Xi * Xi) * (Xr * xr)) + ((Xi * (xi * ((Xr *
Xr) + (xr * xr)))) + (Xr * ((xi * xi) * xr))))) + ((4 * (Xi * (Xr * (xi *
xr)))) + ((xi * xi) * (xr * xr)))))))) * (Xi + xi)))))) + (abs(((Xr * Xr) * (Xr
* Xr))) * (xr * abs((Xi + xi))))))))))))))))))))))));
      </perturbation>
      <derivative t='M'>
        dxan = (daa + (((dxa * abs(Xxr)) * (abs(((Xxi * Xxi) * (Xxi * Xxi))) +
abs(((Xxr * Xxr) * (Xxr * Xxr))))) + ((sgn(Xxr) * (Xxr * (dxa * (abs(((Xxi *
Xxi) * (Xxi * Xxi))) + abs(((Xxr * Xxr) * (Xxr * Xxr))))))) + (( - 1 *
((abs(((Xxi * Xxi) * (Xxi * Xxi))) * (dya * abs(Xxi))) + (sgn(Xxi) * (Xxi *
(dya * (abs(((Xxi * Xxi) * (Xxi * Xxi))) + abs(((Xxr * Xxr) * (Xxr *
Xxr)))))))) + (( - 4 * (abs((Xxi * (Xxr * (Xxr * Xxr)))) * (dya * abs(Xxr))))
+ ((4 * (abs((Xxi * ((Xxi * Xxi) * Xxr))) * (dya * abs(Xxr)))) + (( - 4 *
(abs((Xxi * (Xxr * (Xxr * Xxr)))) * (dxa * abs(Xxi)))) + ((4 * (abs((Xxi *
((Xxi * Xxi) * Xxr))) * (dxa * abs(Xxi)))) + (( - 4 * (sgn(Xxi) * (Xxr * (dya *
abs((Xxi * (Xxr * (Xxr * Xxr)))))))) + ((4 * (sgn(Xxi) * (Xxr * (dya * abs((Xxi
* ((Xxi * Xxi) * Xxr)))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) *
sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dya * abs(Xxi))))))) + ((4 *
(((sgn(Xxr) * sgn(Xxr)) * (sgn(Xxr) * sgn(Xxr))) * (((Xxr * Xxr) * (Xxr * Xxr))
* (dxa * abs(Xxr))))) + (( - 4 * (sgn(Xxr) * (Xxi * (dxa * abs((Xxi * (Xxr *
(Xxr * Xxr))))))) + ((4 * (sgn(Xxr) * (Xxi * (dxa * abs((Xxi * ((Xxi * Xxi) *
Xxr)))))) + (( - 4 * ((sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi *
(Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))))) + (((sgn(Xxr) * sgn(Xxr)) *
(sgn(Xxr) * sgn(Xxr))) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxi))))))) +
((4 * ((((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxi) * sgn(Xxi))) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dya * abs(Xxr)))))) + (sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
((((Xxi * Xxi) * (Xxi * Xxi)) * (dxa * abs(Xxr))) + (Xxi * ((Xxi * Xxi) * (Xxr
* (dxa * abs(Xxi))))))))))) + (( - 4 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxi) *
sgn(Xxi))) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dya * abs(Xxi))))) + ((6 *
(abs(((Xxi * Xxi) * (Xxr * Xxr))) * (dya * abs(Xxi)))) + (( - 6 * ((abs(((Xxi *
Xxi) * (Xxr * Xxr))) * (dxa * abs(Xxr))) + (sgn(Xxr) * (Xxr * (dxa * abs(((Xxi
* Xxi) * (Xxr * Xxr))))))))) + ((6 * (sgn(Xxi) * (Xxi * (dya * abs(((Xxi * Xxi)
* (Xxr * Xxr))))))) + (( - 12 * (((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))))) + (sgn(Xxi) *
(sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa *
```

```
4150  abs(Xxi))))))))))) + ((12 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
      ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi)))))))) + ((sgn(Xxi) * sgn(Xxi)) *
      ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi))))))))) +
      (( - 12 * (((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) *
      ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) *
4155  sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr))))))))) + ((12 *
      ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr *
      (dya * abs(Xxr))))))))) + ((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi
      * ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi))))))))) + ( - 1 * (abs(((Xxr * Xxr) *
      (Xxr * Xxr))) * (dya * abs(Xxi))))))))))))))))))))))))))));
4160          dxbn = (dab + (((dxb * abs(Xxr)) * (abs(((Xxi * Xxi) * (Xxi * Xxi))) +
      abs(((Xxr * Xxr) * (Xxr * Xxr))))) + ((sgn(Xxr) * (Xxr * (dxb * (abs(((Xxi *
      Xxi) * (Xxi * Xxi))) + abs(((Xxr * Xxr) * (Xxr * Xxr))))))) + (( - 1 *
      ((abs(((Xxi * Xxi) * (Xxi * Xxi))) * (dyb * abs(Xxi))) + (sgn(Xxi) * (Xxi *
      (dyb * (abs(((Xxi * Xxi) * (Xxi * Xxi))) + abs(((Xxr * Xxr) * (Xxr *
4165  Xxr))))))))) + (( - 4 * (abs((Xxi * (Xxr * (Xxr * Xxr)))) * (dyb * abs(Xxr))))
      + ((4 * (abs((Xxi * ((Xxi * Xxi) * Xxr))) * (dyb * abs(Xxr)))) + (( - 4 *
      (abs((Xxi * (Xxr * (Xxr * Xxr)))) * (dxb * abs(Xxi)))) + ((4 * (abs((Xxi *
      ((Xxi * Xxi) * Xxr))) * (dxb * abs(Xxi)))) + (( - 4 * (sgn(Xxi) * (Xxr * (dyb *
      abs((Xxi * (Xxr * (Xxr * Xxr))))))))) + ((4 * (sgn(Xxi) * (Xxr * (dyb * abs((Xxi
4170  * ((Xxi * Xxi) * Xxr)))))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) *
      sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dyb * abs(Xxi))))))) + ((4 *
      (((sgn(Xxr) * sgn(Xxr)) * (sgn(Xxr) * sgn(Xxr))) * (((Xxr * Xxr) * (Xxr * Xxr))
      * (dxb * abs(Xxr))))) + (( - 4 * (sgn(Xxr) * (Xxi * (dxb * abs((Xxi * (Xxr *
      (Xxr * Xxr))))))))) + ((4 * (sgn(Xxr) * (Xxi * (dxb * abs((Xxi * ((Xxi * Xxi) *
4175  Xxr)))))))) + (( - 4 * ((sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi *
      (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr))))))))) + (((sgn(Xxr) * sgn(Xxr)) *
      (sgn(Xxr) * sgn(Xxr))) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxi))))))))) +
      ((4 * ((((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxi) * sgn(Xxi))) * (Xxi * ((Xxi * Xxi) *
      (Xxr * (dyb * abs(Xxr)))))) + (sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
4180  ((((Xxi * Xxi) * (Xxi * Xxi)) * (dxb * abs(Xxr))) + (Xxi * ((Xxi * Xxi) * (Xxr
      * (dxb * abs(Xxi))))))))))) + (( - 4 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxi) *
      sgn(Xxi))) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dyb * abs(Xxi))))) + ((6 *
      (abs(((Xxi * Xxi) * (Xxr * Xxr))) * (dyb * abs(Xxi)))) + (( - 6 * ((abs(((Xxi *
      Xxi) * (Xxr * Xxr))) * (dxb * abs(Xxr))) + (sgn(Xxr) * (Xxr * (dxb * abs(((Xxi
4185  * Xxi) * (Xxr * Xxr)))))))) + ((6 * (sgn(Xxi) * (Xxi * (dyb * abs(((Xxi * Xxi)
      * (Xxr * Xxr))))))) + (( - 12 * (((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) *
      sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr))))))) + (sgn(Xxi) *
      (sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb *
      abs(Xxi))))))))) + ((12 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
4190  ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi))))))) + ((sgn(Xxi) * sgn(Xxi)) *
      ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi)))))))) +
      (( - 12 * (((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) *
      ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) *
      sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr))))))))) + ((12 *
4195  ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr *
      (dyb * abs(Xxr))))))))) + ((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi
      * ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi))))))))) + ( - 1 * (abs(((Xxr * Xxr) *
      (Xxr * Xxr))) * (dyb * abs(Xxi))))))))))))))))))))))))))));
            dyan = (dba + ((abs(((Xxi * Xxi) * (Xxi * Xxi))) * ((dxa * abs(Xxi)) +
4200  (dya * abs(Xxr)))) + ((abs(((Xxr * Xxr) * (Xxr * Xxr))) * (dxa * abs(Xxi))) +
      ((sgn(Xxi) * (Xxr * (dya * (abs(((Xxi * Xxi) * (Xxi * Xxi))) + abs(((Xxr * Xxr)
      * (Xxr * Xxr))))))) + ((sgn(Xxr) * (Xxi * (dxa * (abs(((Xxi * Xxi) * (Xxi *
      Xxi))) + abs(((Xxr * Xxr) * (Xxr * Xxr))))))) + (( - 4 * (abs((Xxi * (Xxr *
      (Xxr * Xxr)))) * (dya * abs(Xxi)))) + ((4 * ((abs((Xxi * ((Xxi * Xxi) * Xxr)))
4205  * (dya * abs(Xxi))) + (abs((Xxi * (Xxr * (Xxr * Xxr)))) * (dxa * abs(Xxr))))) +
      (( - 4 * (abs((Xxi * ((Xxi * Xxi) * Xxr))) * (dxa * abs(Xxr)))) + ((4 *
```

```
(sgn(Xxr) * (Xxr * (dxa * abs((Xxi * (Xxr * (Xxr * Xxr)))))))) + (( - 4 *
(sgn(Xxr) * (Xxr * (dxa * abs((Xxi * ((Xxi * Xxi) * Xxr)))))))) + ((4 *
((sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr))
* (dya * abs(Xxr)))))) + (((sgn(Xxr) * sgn(Xxr)) * (sgn(Xxr) * sgn(Xxr))) *
(((Xxr * Xxr) * (Xxr * Xxr)) * (dxa * abs(Xxi)))))) + (( - 4 * (sgn(Xxi) * (Xxi
* (dya * abs((Xxi * (Xxr * (Xxr * Xxr)))))))) + ((4 * (sgn(Xxi) * (Xxi * (dya *
abs((Xxi * ((Xxi * Xxi) * Xxr)))))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxr) *
((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxi)))))))))) +
((4 * ((((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxi) * sgn(Xxi))) * ((((Xxi * Xxi) * (Xxi
* Xxi)) * (dya * abs(Xxr))) + (Xxi * ((Xxi * Xxi) * (Xxr * (dya *
abs(Xxi))))))) + ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (((Xxi *
Xxi) * (Xxi * Xxi)) * (dxa * abs(Xxi)))))) + (((sgn(Xxr) * sgn(Xxr)) *
(sgn(Xxr) * sgn(Xxr))) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr)))))))) +
(( - 4 * (sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dxa * abs(Xxr)))))))) + (( - 6 * ((abs(((Xxi * Xxi) * (Xxr * Xxr))) *
((dxa * abs(Xxi)) + (dya * abs(Xxr)))) + ((sgn(Xxi) * (Xxr * (dya * abs(((Xxi *
Xxi) * (Xxr * Xxr)))))) + (sgn(Xxr) * (Xxi * (dxa * abs(((Xxi * Xxi) * (Xxr *
Xxr))))))))) + (( - 12 * ((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi
* (Xxr * ((Xxr * Xxr) * (dya * abs(Xxi)))))))) + ((12 * (sgn(Xxi) * (sgn(Xxr) *
((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr)))))))) +
(( - 12 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
((Xxr * Xxr) * (dya * abs(Xxr))))))) + (((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) *
sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * ((dxa * abs(Xxi)) + (dya *
abs(Xxr))))))) + (sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi)
* ((Xxr * Xxr) * (dxa * abs(Xxi))))))))) + ((12 * (sgn(Xxi) * ((sgn(Xxi) *
sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr * (dya * abs(Xxi)))))))) +
(( - 12 * ((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi * Xxi)
* (Xxr * (dxa * abs(Xxr)))))))) + (abs(((Xxr * Xxr) * (Xxr * Xxr))) * (dya *
abs(Xxr)))))))))))))))))))))));
        dybn = (dbb + ((abs(((Xxi * Xxi) * (Xxi * Xxi))) * ((dxb * abs(Xxi)) +
(dyb * abs(Xxr)))) + ((abs(((Xxr * Xxr) * (Xxr * Xxr))) * (dxb * abs(Xxi))) +
((sgn(Xxi) * (Xxr * (dyb * (abs(((Xxi * Xxi) * (Xxi * Xxi))) + abs(((Xxr * Xxr)
* (Xxr * Xxr))))))) + ((sgn(Xxr) * (Xxi * (dxb * (abs(((Xxi * Xxi) * (Xxi *
Xxi))) + abs(((Xxr * Xxr) * (Xxr * Xxr))))))) + (( - 4 * (abs((Xxi * (Xxr *
(Xxr * Xxr)))) * (dyb * abs(Xxi)))) + ((4 * ((abs((Xxi * ((Xxi * Xxi) * Xxr)))
* (dyb * abs(Xxi))) + (abs((Xxi * (Xxr * (Xxr * Xxr)))) * (dxb * abs(Xxr))))) +
(( - 4 * (abs((Xxi * ((Xxi * Xxi) * Xxr))) * (dxb * abs(Xxr)))) + ((4 *
(sgn(Xxr) * (Xxr * (dxb * abs((Xxi * (Xxr * (Xxr * Xxr)))))))) + (( - 4 *
(sgn(Xxr) * (Xxr * (dxb * abs((Xxi * ((Xxi * Xxi) * Xxr))))))) + ((4 *
((sgn(Xxi) * (sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr))
* (dyb * abs(Xxr)))))) + (((sgn(Xxr) * sgn(Xxr)) * (sgn(Xxr) * sgn(Xxr))) *
(((Xxr * Xxr) * (Xxr * Xxr)) * (dxb * abs(Xxi)))))) + (( - 4 * (sgn(Xxi) * (Xxi
* (dyb * abs((Xxi * (Xxr * (Xxr * Xxr)))))))) + ((4 * (sgn(Xxi) * (Xxi * (dyb *
abs((Xxi * ((Xxi * Xxi) * Xxr)))))))) + (( - 4 * (sgn(Xxi) * (sgn(Xxr) *
((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxi)))))))))) +
((4 * ((((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxi) * sgn(Xxi))) * ((((Xxi * Xxi) * (Xxi
* Xxi)) * (dyb * abs(Xxr))) + (Xxi * ((Xxi * Xxi) * (Xxr * (dyb *
abs(Xxi))))))) + ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (((Xxi *
Xxi) * (Xxi * Xxi)) * (dxb * abs(Xxi)))))) + (((sgn(Xxr) * sgn(Xxr)) *
(sgn(Xxr) * sgn(Xxr))) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr)))))))) +
(( - 4 * (sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dxb * abs(Xxr)))))))) + (( - 6 * ((abs(((Xxi * Xxi) * (Xxr * Xxr))) *
((dxb * abs(Xxi)) + (dyb * abs(Xxr)))) + ((sgn(Xxi) * (Xxr * (dyb * abs(((Xxi *
Xxi) * (Xxr * Xxr)))))) + (sgn(Xxr) * (Xxi * (dxb * abs(((Xxi * Xxi) * (Xxr *
Xxr))))))))) + (( - 12 * ((sgn(Xxi) * sgn(Xxi)) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi
* (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxi)))))))) + ((12 * (sgn(Xxi) * (sgn(Xxr) *
((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr)))))))) +
```

```
((-12*((sgn(Xxi)*((sgn(Xxi)*sgn(Xxi))*(sgn(Xxr)*((Xxi*Xxi)*
((Xxr*Xxr)*(dyb*abs(Xxr)))))))+(((sgn(Xxi)*sgn(Xxi))*((sgn(Xxr)*
sgn(Xxr))*((Xxi*Xxi)*((Xxr*Xxr)*((dxb*abs(Xxi))+(dyb*
abs(Xxr)))))))+(sgn(Xxi)*(sgn(Xxr)*((sgn(Xxr)*sgn(Xxr))*((Xxi*Xxi)
*((Xxr*Xxr)*(dxb*abs(Xxi))))))))))+((12*(sgn(Xxi)*((sgn(Xxi)*
sgn(Xxi))*(sgn(Xxr)*(Xxi*((Xxi*Xxi)*(Xxr*(dyb*abs(Xxi)))))))))+
((-12*((sgn(Xxi)*sgn(Xxi))*((sgn(Xxr)*sgn(Xxr))*(Xxi*((Xxi*Xxi)
*(Xxr*(dxb*abs(Xxr)))))))+(abs(((Xxr*Xxr)*(Xxr*Xxr)))*(dyb*
abs(Xxr))))))))))))))))))))))));
      </derivative>
    </formula>
  </group>

  <group type="65" name="Mothbrot 6th 2x4 (Simon's Mothbrot)">
    <formula power="6" glitch="1.0000000000000002e-4">
      <reference t='R'>
        Xrn = (Cr + ((abs((Xi*(Xi*Xi)))*((Xr*Xr)*abs(Xi)))+((-1*
((abs((Xi*(Xi*Xi)))*((Xi*Xi)*abs(Xi)))+(abs((Xr*(Xr*Xr)))*
((Xi*Xi)*abs(Xr)))))+((-2*(abs((Xr*(Xr*Xr)))*(Xi*(Xr*
abs(Xi)))))+((2*(abs((Xi*(Xi*Xi)))*(Xi*(Xr*abs(Xr)))))+((-3*
((abs(((Xi*Xi)*Xr))*((Xr*Xr)*abs(Xr)))+(abs((Xi*(Xr*Xr)))*
((Xr*Xr)*abs(Xi)))))+((3*((abs(((Xi*Xi)*Xr))*((Xi*Xi)*
abs(Xr)))+(abs((Xi*(Xr*Xr)))*((Xi*Xi)*abs(Xi)))))+((-6*
(abs((Xi*(Xr*Xr)))*(Xi*(Xr*abs(Xr)))))+((6*(abs(((Xi*Xi)*Xr))
*(Xi*(Xr*abs(Xi)))))+(abs((Xr*(Xr*Xr)))*((Xr*Xr)*
abs(Xr))))))))))));
        Xin = (Ci + ((abs((Xi*(Xi*Xi)))*((Xi*Xi)*abs(Xr)))+((-1*
((abs((Xi*(Xi*Xi)))*((Xr*Xr)*abs(Xr)))+(abs((Xr*(Xr*Xr)))*
((Xi*Xi)*abs(Xi)))))+((2*((abs((Xi*(Xi*Xi)))*(Xi*(Xr*
abs(Xi))))+(abs((Xr*(Xr*Xr)))*(Xi*(Xr*abs(Xr))))))+((3*(abs((Xi
*(Xr*Xr)))*((Xr*Xr)*abs(Xr))))+((-3*((abs(((Xi*Xi)*Xr))*
((Xr*Xr)*abs(Xi)))+(abs((Xi*(Xr*Xr)))*((Xi*Xi)*abs(Xr)))))+
((3*(abs(((Xi*Xi)*Xr))*((Xi*Xi)*abs(Xi))))+((-6*((abs(((Xi*
Xi)*Xr))*(Xi*(Xr*abs(Xr))))+(abs((Xi*(Xr*Xr)))*(Xi*(Xr*
abs(Xi))))))+(abs((Xr*(Xr*Xr)))*((Xr*Xr)*abs(Xi))))))))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs((Xi*(Xi*Xi)))*(((Xr*Xr)*diffabs(Xi,xi))+
((xr*xr)*abs((Xi+xi)))))+((abs((Xr*(Xr*Xr)))*((Xr*Xr)*
diffabs(Xr,xr)))+((abs((Xi+xi))*(diffabs((Xi*(Xi*Xi)),((3*(((Xi*
Xi)*xi)+(Xi*(xi*xi))))+(xi*(xi*xi))))*((Xr*Xr)+(xr*xr))))
+((abs((Xr+xr))*(diffabs((Xr*(Xr*Xr)),((3*(((Xr*Xr)*xr)+(Xr*
(xr*xr))))+(xr*(xr*xr))))*((Xr*Xr)+(xr*xr))))+((-1*
((abs((Xi*(Xi*Xi)))*(((Xi*Xi)*diffabs(Xi,xi))+((xi*xi)*abs((Xi
+xi)))))+((abs((Xr*(Xr*Xr)))*(((Xi*Xi)*diffabs(Xr,xr))+((xi*
xi)*abs((Xr+xr)))))+((abs((Xi+xi))*(diffabs((Xi*(Xi*Xi)),((3*
(((Xi*Xi)*xi)+(Xi*(xi*xi))))+(xi*(xi*xi))))*((Xi*Xi)+(xi
*xi))))+(abs((Xr+xr))*(diffabs((Xr*(Xr*Xr)),((3*(((Xr*Xr)*xr)
+(Xr*(xr*xr))))+(xr*(xr*xr))))*((Xi*Xi)+(xi*xi)))))))))+((
-2*(abs((Xr*(Xr*Xr)))*(xi*(xr*abs((Xi+xi)))))))+((2*((abs((Xi
*(Xi*Xi)))*(xi*(xr*abs((Xr+xr))))))+(abs((Xr+xr))*(xi*(xr*
diffabs((Xi*(Xi*Xi)),((3*(((Xi*Xi)*xi)+(Xi*(xi*xi))))+(xi*
(xi*xi)))))))))+((-2*(abs((Xi+xi))*(xi*(xr*diffabs((Xr*(Xr*
Xr)),((3*(((Xr*Xr)*xr)+(Xr*(xr*xr))))+(xr*(xr*xr)))))))))+
((2*((abs((Xi*(Xi*Xi)))*(Xr*(xr*abs((Xi+xi))))))+((abs((Xr*(Xr
*Xr)))*(Xr*(xr*abs((Xr+xr)))))+((abs((Xi+xi))*(Xr*(xr*
diffabs((Xi*(Xi*Xi)),((3*(((Xi*Xi)*xi)+(Xi*(xi*xi))))+(xi*
```

```
(xi * xi))))))) + (abs((Xr + xr)) * (Xr * (xr * diffabs((Xr * (Xr * Xr)),((3 *
(((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr))))))))))) + (( - 2 *
(abs((Xr * (Xr * Xr))) * (Xr * (xi * abs((Xi + xi)))))) + ((2 * ((abs((Xi * (Xi
* Xi))) * (Xr * (xi * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xi *
diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi *
(xi * xi)))))))))) + (( - 2 * ((abs((Xr * (Xr * Xr))) * (Xi * (xr * abs((Xi +
xi))))) + (abs((Xi + xi)) * (Xr * (xi * diffabs((Xr * (Xr * Xr)),((3 * (((Xr *
Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr)))))))))) + ((2 * ((abs((Xi *
(Xi * Xi))) * (Xi * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xi * (xr *
diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi *
(xi * xi)))))))))) + (( - 2 * ((abs((Xi * (Xi * Xi))) * (Xi * (xi * abs((Xi +
xi))))) + ((abs((Xr * (Xr * Xr))) * (Xi * ((Xr * diffabs(Xi,xi)) + (xi *
abs((Xr + xr)))))) + ((abs((Xi + xi)) * (Xi * ((xi * diffabs((Xi * (Xi *
Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi))))) + (xr *
diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr *
(xr * xr)))))))) + (abs((Xr + xr)) * (Xi * (xi * diffabs((Xr * (Xr * Xr)),((3 *
(((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr))))))))))) + ((2 *
((abs((Xi * (Xi * Xi))) * (Xi * (Xr * diffabs(Xr,xr)))) + (abs((Xr + xr)) * (Xi
* (Xr * diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) +
(xi * (xi * xi)))))))))) + (( - 2 * (abs((Xi + xi)) * (Xi * (Xr * diffabs((Xr *
(Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr *
xr)))))))) + (( - 3 * ((abs(((Xi * Xi) * Xr)) * ((xr * xr) * abs((Xr + xr)))) +
((abs((Xi * (Xr * Xr))) * ((xr * xr) * abs((Xi + xi)))) + ((abs((Xi + xi)) *
((xr * xr) * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) +
((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))) +
(abs((Xr + xr)) * ((xr * xr) * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) +
((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))
+ ((3 * ((abs(((Xi * Xi) * Xr)) * ((xi * xi) * abs((Xr + xr)))) + ((abs((Xi *
(Xr * Xr))) * ((xi * xi) * abs((Xi + xi)))) + ((abs((Xi + xi)) * ((xi * xi) *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))) + (abs((Xr + xr)) *
((xi * xi) * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) +
((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))) + (( - 3 *
((abs(((Xi * Xi) * Xr)) * ((Xr * Xr) * diffabs(Xr,xr))) + ((abs((Xi * (Xr *
Xr))) * ((Xr * Xr) * diffabs(Xi,xi))) + ((abs((Xi + xi)) * ((Xr * Xr) *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))) + (abs((Xr + xr)) *
((Xr * Xr) * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) +
((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))) + ((3 * ((abs(((Xi
* Xi) * Xr)) * ((Xi * Xi) * diffabs(Xr,xr))) + ((abs((Xi * (Xr * Xr))) * ((Xi *
Xi) * diffabs(Xi,xi))) + ((abs((Xi + xi)) * ((Xi * Xi) * diffabs((Xi * (Xr *
Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr *
(xi * xr)))) + (xi * (xr * xr))))))))) + (abs((Xr + xr)) * ((Xi * Xi) *
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))) + (( - 6 * ((abs((Xi * (Xr *
Xr))) * (xi * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (xi * (xr *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))))) + ((6 * ((abs(((Xi
* Xi) * Xr)) * (xi * (xr * abs((Xi + xi))))) + (abs((Xi + xi)) * (xi * (xr *
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr))))))))))) + (( - 6 * ((abs(((Xi * Xi) *
Xr)) * (Xr * (xr * abs((Xr + xr))))) + ((abs((Xi * (Xr * Xr))) * (Xr * ((xi *
abs((Xr + xr))) + (xr * abs((Xi + xi))))))) + ((abs((Xi + xi)) * (Xr * (xr *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))) + (abs((Xr + xr)) *
(Xr * ((xi * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) +
((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))) + (xr *
```

```
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr)))))))))))) + ((6 * ((abs(((Xi * Xi) *
4380 Xr)) * (Xr * (xi * abs((Xi + xi))))) + (abs((Xi + xi)) * (Xr * (xi *
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr)))))))))) + (( - 6 * ((abs((Xi * (Xr *
Xr))) * (Xi * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xi * (xr *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
4385 (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr)))))))))))) + ((6 * ((abs(((Xi
* Xi) * Xr)) * (Xi * ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))) +
((abs((Xi * (Xr * Xr))) * (Xi * (xi * abs((Xi + xi))))) + ((abs((Xi + xi)) *
(Xi * ((xi * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) +
((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr)))))))) + (xr *
4390 diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr)))))))))) + (abs((Xr + xr)) * (Xi * (xi *
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr)))))))))))) + (( - 6 * ((abs((Xi * (Xr *
Xr))) * (Xi * (Xr * diffabs(Xr,xr)))) + (abs((Xr + xr)) * (Xi * (Xr *
4395 diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr)))))))))))) + ((6 * ((abs(((Xi
* Xi) * Xr)) * (Xi * (Xr * diffabs(Xi,xi)))) + (abs((Xi + xi)) * (Xi * (Xr *
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr)))))))))) + (abs((Xr * (Xr * Xr))) * ((xr
4400 * xr) * abs((Xr + xr))))))))))))))))))))))))))))))))));
        xin = (ci + ((abs((Xi * (Xi * Xi))) * (((Xi * Xi) * diffabs(Xr,xr)) +
((xi * xi) * abs((Xr + xr))))) + ((abs((Xr * (Xr * Xr))) * ((Xr * Xr) *
diffabs(Xi,xi))) + ((abs((Xi + xi)) * (diffabs((Xr * (Xr * Xr)),((3 * (((Xr *
Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr)))) * ((Xr * Xr) + (xr * xr))))
4405 + ((abs((Xr + xr)) * (diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi *
(xi * xi)))) + (xi * (xi * xi)))) * ((Xi * Xi) + (xi * xi)))) + (( - 1 *
((abs((Xi * (Xi * Xi))) * (((Xr * Xr) * diffabs(Xr,xr)) + ((xr * xr) * abs((Xr
+ xr))))) + ((abs((Xr * (Xr * Xr))) * (((Xi * Xi) * diffabs(Xi,xi)) + ((xi *
xi) * abs((Xi + xi))))) + ((abs((Xi + xi)) * (diffabs((Xr * (Xr * Xr)),((3 *
4410 (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr)))) * ((Xi * Xi) + (xi
* xi)))) + (abs((Xr + xr)) * (diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi)
+ (Xi * (xi * xi)))) + (xi * (xi * xi)))) * ((Xr * Xr) + (xr * xr)))))))) + ((2
* ((abs((Xi * (Xi * Xi))) * (xi * (xr * abs((Xi + xi))))) + ((abs((Xr * (Xr *
Xr))) * ((Xr * (xr * abs((Xi + xi)))) + (xi * (xr * abs((Xr + xr)))))) +
4415 ((abs((Xi + xi)) * (xi * (xr * diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi)
+ (Xi * (xi * xi)))) + (xi * (xi * xi)))))) + (abs((Xr + xr)) * (xi * (xr *
diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr *
(xr * xr)))))))))) + (( - 2 * ((abs((Xi * (Xi * Xi))) * (Xr * (xr * abs((Xr +
xr))))) + (abs((Xr + xr)) * (Xr * (xr * diffabs((Xi * (Xi * Xi)),((3 * (((Xi *
4420 Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi))))))))) + ((2 * ((abs((Xi *
(Xi * Xi))) * ((Xi * (xr * abs((Xi + xi)))) + (Xr * (xi * abs((Xi + xi)))))) +
((abs((Xr * (Xr * Xr))) * ((Xi * (xr * abs((Xr + xr)))) + (Xr * (xi * abs((Xr +
xr)))))) + ((abs((Xi + xi)) * ((Xi * (xr * diffabs((Xi * (Xi * Xi)),((3 * (((Xi
* Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi)))))) + (Xr * ((xi *
4425 diffabs((Xi * (Xi * Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi *
(xi * xi))))) + (xr * diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr *
(xr * xr)))) + (xr * (xr * xr))))))))) + (abs((Xr + xr)) * ((Xi * (xr *
diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr * xr)))) + (xr *
(xr * xr)))))) + (Xr * (xi * diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) +
4430 (Xr * (xr * xr)))) + (xr * (xr * xr))))))))))) + (( - 2 * (abs((Xr * (Xr *
Xr))) * (Xi * (xi * abs((Xi + xi)))))) + ((2 * ((abs((Xi * (Xi * Xi))) * (Xi *
(xi * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xi * (xi * diffabs((Xi * (Xi *
Xi)),((3 * (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi))))))))) +
(( - 2 * (abs((Xi + xi)) * (Xi * (xi * diffabs((Xr * (Xr * Xr)),((3 * (((Xr *
```

```
Xr) * xr) + (Xr * (xr * xr)))) + (xr * (xr * xr)))))))) + ((2 * ((abs((Xi * (Xi
* Xi))) * (Xi * (Xr * diffabs(Xi,xi)))) + ((abs((Xr * (Xr * Xr))) * (Xi * (Xr *
diffabs(Xr,xr)))) + ((abs((Xi + xi)) * (Xi * (Xr * diffabs((Xi * (Xi * Xi)),((3
* (((Xi * Xi) * xi) + (Xi * (xi * xi)))) + (xi * (xi * xi))))))) + (abs((Xr +
xr)) * (Xi * (Xr * diffabs((Xr * (Xr * Xr)),((3 * (((Xr * Xr) * xr) + (Xr * (xr
* xr)))) + (xr * (xr * xr)))))))))) + ((3 * ((abs((Xi * (Xr * Xr))) * ((xr *
xr) * abs((Xr + xr)))) + (abs((Xr + xr)) * ((xr * xr) * diffabs((Xi * (Xr *
Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr *
(xi * xr)))) + (xi * (xr * xr))))))))) + (( - 3 * ((abs(((Xi * Xi) * Xr)) *
((xr * xr) * abs((Xi + xi)))) + ((abs((Xi * (Xr * Xr))) * ((xi * xi) * abs((Xr
+ xr)))) + ((abs((Xi + xi)) * ((xr * xr) * diffabs(((Xi * Xi) * Xr),(((Xi * Xi)
* xr) + ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) *
xr)))))))) + (abs((Xr + xr)) * ((xi * xi) * diffabs((Xi * (Xr * Xr)),((Xi * (xr
* xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi
* (xr * xr))))))))))) + ((3 * ((abs(((Xi * Xi) * Xr)) * ((xi * xi) * abs((Xi +
xi)))) + ((abs((Xi * (Xr * Xr))) * ((Xr * Xr) * diffabs(Xr,xr))) + ((abs((Xi +
xi)) * ((xi * xi) * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi *
xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr)))))))) + (abs((Xr +
xr)) * ((Xr * Xr) * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) *
xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))))))
+ (( - 3 * ((abs(((Xi * Xi) * Xr)) * ((Xr * Xr) * diffabs(Xi,xi))) + ((abs((Xi
* (Xr * Xr))) * ((Xi * Xi) * diffabs(Xr,xr))) + ((abs((Xi + xi)) * ((Xr * Xr) *
diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi *
(xi * (Xr + xr)))) + ((xi * xi) * xr)))))))) + (abs((Xr + xr)) * ((Xi * Xi) *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))))))) + ((6 * ((abs((Xi *
Xr))) * (Xr * (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xr * (xr *
diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) * xi) + ((2 * ((Xi *
(Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))))))) + (( - 6 *
((abs(((Xi * Xi) * Xr)) * ((Xi * (xr * abs((Xr + xr)))) + (Xr * ((xi * abs((Xr
+ xr))) + (xr * abs((Xi + xi))))))) + ((abs((Xi * (Xr * Xr))) * ((Xi * ((xi *
abs((Xr + xr))) + (xr * abs((Xi + xi)))))) + (Xr * (xi * abs((Xi + xi))))))) +
((abs((Xi + xi)) * ((Xi * (xr * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) +
(((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr *
xr))))))))) + (Xr * ((xi * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr *
Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr *
xr))))))) + (xr * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi *
xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr)))))))))) + (abs((Xr +
xr)) * ((Xi * ((xi * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) + (((Xr * Xr) *
xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr * xr))))))) +
(xr * diffabs(((Xi * Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 *
(Xi * (xi * (Xr + xr)))) + ((xi * xi) * xr)))))))) + (Xr * (xi * diffabs(((Xi *
Xi) * Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr +
xr)))) + ((xi * xi) * xr))))))))))))) + ((6 * ((abs(((Xi * Xi) * Xr)) * (Xi *
(xi * abs((Xi + xi))))) + (abs((Xi + xi)) * (Xi * (xi * diffabs(((Xi * Xi) *
Xr),(((Xi * Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) +
((xi * xi) * xr))))))))))) + (( - 6 * ((abs(((Xi * Xi) * Xr)) * (Xi * (Xr *
diffabs(Xr,xr)))) + ((abs((Xi * (Xr * Xr))) * (Xi * (Xr * diffabs(Xi,xi)))) +
```

```
((abs((Xi + xi)) * (Xi * (Xr * diffabs((Xi * (Xr * Xr)),((Xi * (xr * xr)) +
(((Xr * Xr) * xi) + ((2 * ((Xi * (Xr * xr)) + (Xr * (xi * xr)))) + (xi * (xr *
xr)))))))) + (abs((Xr + xr)) * (Xi * (Xr * diffabs(((Xi * Xi) * Xr),(((Xi *
Xi) * xr) + ((Xr * (xi * xi)) + ((2 * (Xi * (xi * (Xr + xr)))) + ((xi * xi) *
xr)))))))))) + (abs((Xr * (Xr * Xr))) * ((xr * xr) * abs((Xi +
xi))))))))))))))))))))))));
      </perturbation>
      <derivative t='M'>
        dxan = ((sgn(Xxi) * ((Xxr * Xxr) * (dya * abs((Xxi * (Xxi * Xxi)))))) +
((sgn(Xxr) * ((Xxr * Xxr) * (dxa * abs((Xxr * (Xxr * Xxr)))))) + (( - 1 *
((sgn(Xxi) * ((Xxi * Xxi) * (dya * abs((Xxi * (Xxi * Xxi)))))) + (sgn(Xxr) *
((Xxi * Xxi) * (dxa * abs((Xxr * (Xxr * Xxr)))))))) + (( - 2 * (abs((Xxr * (Xxr
* Xxr))) * (Xxr * (dya * abs(Xxi))))) + ((2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxr
* ((dxa * abs(Xxi)) + (dya * abs(Xxr))))) + (abs((Xxr * (Xxr * Xxr))) * (Xxr *
(dxa * abs(Xxr)))))) + (( - 2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxi * (dya *
abs(Xxi))))) + (abs((Xxr * (Xxr * Xxr))) * (Xxi * ((dxa * abs(Xxi)) + (dya *
abs(Xxr)))))))) + ((2 * (abs((Xxi * (Xxi * Xxi))) * (Xxi * (dxa * abs(Xxr))))) +
(( - 2 * (sgn(Xxi) * (Xxi * (Xxr * (dya * abs((Xxr * (Xxr * Xxr)))))))) + ((2 *
(sgn(Xxr) * (Xxi * (Xxr * (dxa * abs((Xxi * (Xxi * Xxi)))))))) + (( - 3 *
((sgn(Xxi) * (((Xxr * Xxr) * (dya * abs((Xxi * (Xxr * Xxr)))))) + ((sgn(Xxr) *
sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dya * abs(Xxi)))))) + (sgn(Xxr) *
((Xxr * Xxr) * (dxa * abs(((Xxi * Xxi) * Xxr)))))))) + ((3 * ((sgn(Xxi) * (((Xxi
* Xxi) * (dya * abs((Xxi * (Xxr * Xxr)))))) + (((Xxi * Xxi) * ((Xxr * Xxr) *
(dya * abs(Xxi)))) * ((sgn(Xxi) * sgn(Xxi)) + (sgn(Xxr) * sgn(Xxr)))))) +
(sgn(Xxr) * (((Xxi * Xxi) * (dxa * abs(((Xxi * Xxi) * Xxr)))) + ((sgn(Xxr) *
sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxa * abs(Xxr)))))))) + (( - 3 *
(sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa *
abs(Xxr))))))) + ((3 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (((Xxi * Xxi) *
(Xxi * Xxi)) * (dxa * abs(Xxr)))))) + (( - 3 * ((sgn(Xxi) * ((sgn(Xxi) *
sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dya * abs(Xxi))))) + ((sgn(Xxi) *
sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr)))))))) +
(( - 6 * (abs((Xxi * (Xxr * Xxr))) * (Xxr * (dya * abs(Xxr))))) + ((6 *
(abs(((Xxi * Xxi) * Xxr)) * (Xxr * (dya * abs(Xxi))))) + (( - 6 * ((abs(((Xxi *
Xxi) * Xxr)) * (Xxr * (dxa * abs(Xxr)))) + (abs((Xxi * (Xxr * Xxr))) * (Xxr *
(dxa * abs(Xxi)))))) + ((6 * ((abs(((Xxi * Xxi) * Xxr)) * (Xxi * (dya *
abs(Xxr)))) + (abs((Xxi * (Xxr * Xxr))) * (Xxi * (dya * abs(Xxi)))))) + (( - 6
* (abs((Xxi * (Xxr * Xxr))) * (Xxi * (dxa * abs(Xxr))))) + ((6 * ((abs(((Xxi *
Xxi) * Xxr)) * (Xxi * (dxa * abs(Xxi)))) + (sgn(Xxi) * (Xxi * (Xxr * (dya *
abs(((Xxi * Xxi) * Xxr)))))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
(Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr))))))) + ((sgn(Xxi) * ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * ((dxa * abs(Xxi)) + (dya *
abs(Xxr)))))))) + (sgn(Xxr) * ((Xxi * (Xxr * (dxa * abs((Xxi * (Xxr * Xxr))))))
+ ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa *
abs(Xxi)))))))))) + ((6 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dya * abs(Xxr))))))) + (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
(Xxi * ((Xxi * Xxi) * (Xxr * ((dxa * abs(Xxi)) + (dya * abs(Xxr)))))))) +
(sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi * Xxi) * (Xxr * (dxa *
abs(Xxi)))))))))) + ((12 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
((Xxr * Xxr) * (dya * abs(Xxi))))))) + (( - 12 * (sgn(Xxi) * ((sgn(Xxr) *
sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr))))))) +
daa))))))))))))))))))))));
        dxbn = ((sgn(Xxi) * ((Xxr * Xxr) * (dyb * abs((Xxi * (Xxi * Xxi)))))) +
((sgn(Xxr) * ((Xxr * Xxr) * (dxb * abs((Xxr * (Xxr * Xxr)))))) + (( - 1 *
((sgn(Xxi) * ((Xxi * Xxi) * (dyb * abs((Xxi * (Xxi * Xxi)))))) + (sgn(Xxr) *
((Xxi * Xxi) * (dxb * abs((Xxr * (Xxr * Xxr)))))))) + (( - 2 * (abs((Xxr * (Xxr
* Xxr))) * (Xxr * (dyb * abs(Xxi))))) + ((2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxr
* ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))) + (abs((Xxr * (Xxr * Xxr))) * (Xxr *
```

```
(dxb * abs(Xxr)))))) + (( - 2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxi * (dyb *
4550 abs(Xxi)))) + (abs((Xxr * (Xxr * Xxr))) * (Xxi * ((dxb * abs(Xxi)) + (dyb *
abs(Xxr)))))))) + ((2 * (abs((Xxi * (Xxi * Xxi))) * (Xxi * (dxb * abs(Xxr))))) +
(( - 2 * (sgn(Xxi) * (Xxi * (Xxr * (dyb * abs((Xxr * (Xxr * Xxr)))))))) + ((2 *
(sgn(Xxr) * (Xxi * (Xxr * (dxb * abs((Xxi * (Xxi * Xxi)))))))) + (( - 3 *
((sgn(Xxi) * (((Xxr * Xxr) * (dyb * abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) *
4555 sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dyb * abs(Xxi)))))) + (sgn(Xxr) *
((Xxr * Xxr) * (dxb * abs(((Xxi * Xxi) * Xxr)))))) + ((3 * ((sgn(Xxi) * (((Xxi
* Xxi) * (dyb * abs((Xxi * (Xxr * Xxr))))) + (((Xxi * Xxi) * ((Xxr * Xxr) *
(dyb * abs(Xxi)))) * ((sgn(Xxi) * sgn(Xxi)) + (sgn(Xxr) * sgn(Xxr)))))) +
(sgn(Xxr) * (((Xxi * Xxi) * (dxb * abs(((Xxi * Xxi) * Xxr)))) + ((sgn(Xxr) *
4560 sgn(Xxr)) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxb * abs(Xxr)))))))) + (( - 3 *
(sgn(Xxr) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb *
abs(Xxr)))))) + ((3 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (((Xxi * Xxi) *
(Xxi * Xxi)) * (dxb * abs(Xxr)))))) + (( - 3 * ((sgn(Xxi) * ((sgn(Xxi) *
sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dyb * abs(Xxi))))) + ((sgn(Xxi) *
4565 sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr)))))))) +
(( - 6 * (abs((Xxi * (Xxr * Xxr))) * (Xxr * (dyb * abs(Xxr))))) + ((6 *
(abs(((Xxi * Xxi) * Xxr)) * (Xxr * (dyb * abs(Xxi))))) + (( - 6 * ((abs(((Xxi *
Xxi) * Xxr)) * (Xxr * (dxb * abs(Xxr)))) + (abs((Xxi * (Xxr * Xxr))) * (Xxr *
(dxb * abs(Xxi)))))) + ((6 * ((abs(((Xxi * Xxi) * Xxr)) * (Xxi * (dyb *
4570 abs(Xxr)))) + (abs((Xxi * (Xxr * Xxr))) * (Xxi * (dyb * abs(Xxi)))))) + (( - 6
* (abs((Xxi * (Xxr * Xxr))) * (Xxi * (dxb * abs(Xxr))))) + ((6 * ((abs(((Xxi *
Xxi) * Xxr)) * (Xxi * (dxb * abs(Xxi)))) + (sgn(Xxi) * (Xxi * (Xxr * (dyb *
abs(((Xxi * Xxi) * Xxr)))))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
(Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr)))))))) + ((sgn(Xxi) * ((sgn(Xxr) *
4575 sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * ((dxb * abs(Xxi)) + (dyb *
abs(Xxr)))))))) + (sgn(Xxr) * ((Xxi * (Xxr * (dxb * abs((Xxi * (Xxr * Xxr)))))))
+ ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb *
abs(Xxi)))))))))) + ((6 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dyb * abs(Xxr)))))))) + (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) *
4580 (Xxi * ((Xxi * Xxi) * (Xxr * ((dxb * abs(Xxi)) + (dyb * abs(Xxr)))))))) +
(sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi * Xxi) * (Xxr * (dxb *
abs(Xxi)))))))) + ((12 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
((Xxr * Xxr) * (dyb * abs(Xxi)))))))) + (( - 12 * (sgn(Xxi) * ((sgn(Xxr) *
sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr))))))) +
4585 dab))))))))))))))))))))));
        dyan = ((sgn(Xxi) * ((Xxr * Xxr) * (dya * abs((Xxr * (Xxr * Xxr)))))) +
((sgn(Xxr) * ((Xxi * Xxi) * (dxa * abs((Xxi * (Xxi * Xxi)))))) + (( - 1 *
((sgn(Xxi) * ((Xxi * Xxi) * (dya * abs((Xxr * (Xxr * Xxr)))))) + (sgn(Xxr) *
((Xxr * Xxr) * (dxa * abs((Xxi * (Xxi * Xxi)))))))) + ((2 * ((abs((Xxi * (Xxi *
4590 Xxi))) * (Xxr * (dya * abs(Xxi)))) + (abs((Xxr * (Xxr * Xxr))) * (Xxr * ((dxa *
abs(Xxi)) + (dya * abs(Xxr)))))) + (( - 2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxr
* (dxa * abs(Xxr)))) + (abs((Xxr * (Xxr * Xxr))) * (Xxi * (dya * abs(Xxi))))))
+ ((2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxi * ((dxa * abs(Xxi)) + (dya *
abs(Xxr))))) + ((abs((Xxr * (Xxr * Xxr))) * (Xxi * (dxa * abs(Xxr)))) +
4595 ((sgn(Xxi) * (Xxi * (Xxr * (dya * abs((Xxi * (Xxi * Xxi))))))) + (sgn(Xxr) *
(Xxi * (Xxr * (dxa * abs((Xxr * (Xxr * Xxr)))))))))) + (( - 3 * (sgn(Xxi) *
((Xxr * Xxr) * (dya * abs(((Xxi * Xxi) * Xxr)))))) + ((3 * ((sgn(Xxi) * (((Xxi
* Xxi) * (dya * abs(((Xxi * Xxi) * Xxr)))) + ((sgn(Xxr) * sgn(Xxr)) * (((Xxr *
Xxr) * (Xxr * Xxr)) * (dya * abs(Xxr)))))) + (sgn(Xxr) * (((Xxr * Xxr) * (dxa *
4600 abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) * sgn(Xxr)) * (((Xxr * Xxr) * (Xxr *
Xxr)) * (dxa * abs(Xxi)))))))) + (( - 3 * ((sgn(Xxi) * (((Xxi * Xxi) * ((Xxr *
Xxr) * (dya * abs(Xxr)))) * ((sgn(Xxi) * sgn(Xxi)) + (sgn(Xxr) * sgn(Xxr)))))) +
(sgn(Xxr) * (((Xxi * Xxi) * (dxa * abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) *
sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxi)))))))) + ((3 *
4605 (sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dya *
```

```
abs(Xxr))))))) + (( - 3 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
((Xxr * Xxr) * (dxa * abs(Xxi)))))))) + ((3 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr)
* (((Xxi * Xxi) * (Xxi * Xxi)) * (dxa * abs(Xxi)))))) + (( - 6 * ((abs(((Xxi *
Xxi) * Xxr)) * (Xxr * (dya * abs(Xxr)))) + (abs((Xxi * (Xxr * Xxr))) * (Xxr *
(dya * abs(Xxi)))))) + ((6 * (abs((Xxi * (Xxr * Xxr))) * (Xxr * (dxa *
abs(Xxr))))) + (( - 6 * ((abs(((Xxi * Xxi) * Xxr)) * (Xxr * (dxa * abs(Xxi))))
+ (abs((Xxi * (Xxr * Xxr))) * (Xxi * (dya * abs(Xxr)))))) + ((6 * (abs(((Xxi *
Xxi) * Xxr)) * (Xxi * (dya * abs(Xxi))))) + (( - 6 * ((abs(((Xxi * Xxi) * Xxr))
* (Xxi * (dxa * abs(Xxr)))) + ((abs((Xxi * (Xxr * Xxr))) * (Xxi * (dxa *
abs(Xxi)))) + ((sgn(Xxi) * (Xxi * (Xxr * (dya * abs((Xxi * (Xxr * Xxr))))))) +
(((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya *
abs(Xxi)))))) + ((sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr *
Xxr) * (dya * abs(Xxi)))))) + (sgn(Xxr) * (Xxi * (Xxr * (dxa * abs(((Xxi *
Xxi) * Xxr)))))))))))) + ((6 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (Xxi *
((Xxi * Xxi) * (Xxr * (dya * abs(Xxi)))))))) + (((sgn(Xxi) * sgn(Xxi)) *
(sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr * (dya * abs(Xxi)))))))) + (((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) * (sgn(Xxi) +
sgn(Xxr)))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dxa * abs(Xxr))))))) + (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi
* ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxr)))))))) + (( - 12 * (((sgn(Xxi) *
sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxr)))))) +
(sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa *
abs(Xxi)))))))) + dba))))))))))))))))))));
        dybn = ((sgn(Xxi) * ((Xxr * Xxr) * (dyb * abs((Xxr * (Xxr * Xxr)))))) +
((sgn(Xxr) * ((Xxi * Xxi) * (dxb * abs((Xxi * (Xxi * Xxi)))))) + (( - 1 *
((sgn(Xxi) * ((Xxi * Xxi) * (dyb * abs((Xxr * (Xxr * Xxr)))))) + (sgn(Xxr) *
((Xxr * Xxr) * (dxb * abs((Xxi * (Xxi * Xxi))))))))) + ((2 * ((abs((Xxi * (Xxi *
Xxi))) * (Xxr * (dyb * abs(Xxi)))) + (abs((Xxr * (Xxr * Xxr))) * (Xxr * ((dxb *
abs(Xxi)) + (dyb * abs(Xxr))))))) + (( - 2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxr
* (dxb * abs(Xxr)))) + (abs((Xxr * (Xxr * Xxr))) * (Xxi * (dyb * abs(Xxi))))))
+ ((2 * ((abs((Xxi * (Xxi * Xxi))) * (Xxi * ((dxb * abs(Xxi)) + (dyb *
abs(Xxr)))))) + ((abs((Xxr * (Xxr * Xxr))) * (Xxi * (dxb * abs(Xxr)))) +
((sgn(Xxi) * (Xxi * (Xxr * (dyb * abs((Xxi * (Xxi * Xxi))))))) + (sgn(Xxr) *
(Xxi * (Xxr * (dxb * abs((Xxr * (Xxr * Xxr)))))))))) + (( - 3 * (sgn(Xxi) *
((Xxr * Xxr) * (dyb * abs(((Xxi * Xxi) * Xxr)))))) + ((3 * ((sgn(Xxi) * (((Xxi
* Xxi) * (dyb * abs(((Xxi * Xxi) * Xxr)))) + ((sgn(Xxr) * sgn(Xxr)) * (((Xxr *
Xxr) * (Xxr * Xxr)) * (dyb * abs(Xxr)))))) + (sgn(Xxr) * (((Xxr * Xxr) * (dxb *
abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) * sgn(Xxr)) * (((Xxr * Xxr) * (Xxr *
Xxr)) * (dxb * abs(Xxi))))))))) + (( - 3 * ((sgn(Xxi) * (((Xxi * Xxi) * ((Xxr *
Xxr) * (dyb * abs(Xxr)))) * ((sgn(Xxi) * sgn(Xxi)) + (sgn(Xxr) * sgn(Xxr))))) +
(sgn(Xxr) * (((Xxi * Xxi) * (dxb * abs((Xxi * (Xxr * Xxr))))) + ((sgn(Xxr) *
sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxi))))))))) + ((3 *
(sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dyb *
abs(Xxr)))))) + (( - 3 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) *
((Xxr * Xxr) * (dxb * abs(Xxi))))))) + ((3 * ((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr)
* (((Xxi * Xxi) * (Xxi * Xxi)) * (dxb * abs(Xxi)))))) + (( - 6 * ((abs(((Xxi *
Xxi) * Xxr)) * (Xxr * (dyb * abs(Xxr)))) + (abs((Xxi * (Xxr * Xxr))) * (Xxr *
(dyb * abs(Xxi)))))) + ((6 * (abs((Xxi * (Xxr * Xxr))) * (Xxr * (dxb *
abs(Xxr))))) + (( - 6 * ((abs(((Xxi * Xxi) * Xxr)) * (Xxr * (dxb * abs(Xxi))))
+ (abs((Xxi * (Xxr * Xxr))) * (Xxi * (dyb * abs(Xxr)))))) + ((6 * (abs(((Xxi *
Xxi) * Xxr)) * (Xxi * (dyb * abs(Xxi))))) + (( - 6 * ((abs(((Xxi * Xxi) * Xxr))
* (Xxi * (dxb * abs(Xxr)))) + ((abs((Xxi * (Xxr * Xxr))) * (Xxi * (dxb *
abs(Xxi)))) + ((sgn(Xxi) * (Xxi * (Xxr * (dyb * abs((Xxi * (Xxr * Xxr))))))) +
(((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb *
abs(Xxi)))))) + ((sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr *
Xxr) * (dyb * abs(Xxi)))))) + (sgn(Xxr) * (Xxi * (Xxr * (dxb * abs(((Xxi *
Xxi) * Xxr)))))))))))) + ((6 * ((sgn(Xxi) * ((sgn(Xxi) * sgn(Xxi)) * (Xxi *
```

```
((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi))))))) + (((sgn(Xxi) * sgn(Xxi)) *
(sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi))))))) + (((sgn(Xxr) *
4665 sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) * (sgn(Xxi) +
sgn(Xxr)))))) + (( - 6 * (((sgn(Xxi) * sgn(Xxi)) * (sgn(Xxr) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dxb * abs(Xxr))))))) + (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * (Xxi
* ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxr)))))))) + (( - 12 * (((sgn(Xxi) *
sgn(Xxi)) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) +
4670 (sgn(Xxi) * ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb *
abs(Xxi))))))) + dbb))))))))))))))))))));
      </derivative>
    </formula>
  </group>
4675

  <group type="66" name="Mothbrot 6th 3x3 (aka SimonBrot 6th)">
    <formula power="6" glitch="1.0000000000000002e-4">
      <reference t='R'>
        Xrn = (Cr + ((abs((Xr * Xr)) * (Xi * ((Xi * Xi) * abs(Xi)))) + (( - 1 *
4680 (abs((Xi * Xi)) * ((Xi * ((Xi * Xi) * abs(Xi))) + (Xr * ((Xr * Xr) *
abs(Xr)))))) + (( - 2 * (abs((Xi * Xr)) * (Xr * ((Xr * Xr) * abs(Xi))))) + ((2
* (abs((Xi * Xr)) * (Xi * ((Xi * Xi) * abs(Xr))))) + (( - 3 * (abs((Xr * Xr)) *
(Xi * ((Xr * Xr) * abs(Xi))))) + ((3 * (abs((Xi * Xi)) * (Xi * ((Xr * Xr) *
abs(Xi))))) + (( - 3 * (abs((Xr * Xr)) * ((Xi * Xi) * (Xr * abs(Xr))))) + ((3 *
4685 (abs((Xi * Xi)) * ((Xi * Xi) * (Xr * abs(Xr))))) + (( - 6 * (abs((Xi * Xr)) *
(Xi * ((Xr * Xr) * abs(Xr))))) + ((6 * (abs((Xi * Xr)) * ((Xi * Xi) * (Xr *
abs(Xi))))) + (abs((Xr * Xr)) * (Xr * ((Xr * Xr) * abs(Xr))))))))))))));
        Xin = (Ci + ((abs((Xi * Xi)) * (Xi * ((Xi * Xi) * abs(Xr)))) + (( - 1 *
((abs((Xi * Xi)) * (Xr * ((Xr * Xr) * abs(Xi)))) + (abs((Xr * Xr)) * (Xi * ((Xi
4690 * Xi) * abs(Xr)))))) + ((2 * (abs((Xi * Xr)) * ((Xi * ((Xi * Xi) * abs(Xi))) +
(Xr * ((Xr * Xr) * abs(Xr)))))) + ((3 * (abs((Xr * Xr)) * (Xi * ((Xr * Xr) *
abs(Xr))))) + (( - 3 * ((abs((Xi * Xi)) * (Xi * ((Xr * Xr) * abs(Xr)))) +
(abs((Xr * Xr)) * ((Xi * Xi) * (Xr * abs(Xi)))))) + ((3 * (abs((Xi * Xi)) *
((Xi * Xi) * (Xr * abs(Xi))))) + (( - 6 * (abs((Xi * Xr)) * (((Xi * Xi) * (Xr *
4695 abs(Xr))) + (Xi * ((Xr * Xr) * abs(Xi)))))) + (abs((Xr * Xr)) * (Xr * ((Xr *
Xr) * abs(Xi))))))))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((abs((Xr * Xr)) * ((Xi * ((Xi * Xi) * diffabs(Xi,xi))) +
4700 ((Xr * ((Xr * Xr) * diffabs(Xr,xr))) + (xi * ((xi * xi) * abs((Xi + xi))))))) +
((abs((Xi + xi)) * ((Xi * ((Xi * Xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr
* xr))))) + (xi * ((xi * xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr *
xr)))))))) + ((abs((Xr + xr)) * ((Xr * ((Xr * Xr) * diffabs((Xr * Xr),((2 * (Xr
* xr)) + (xr * xr))))) + (xr * ((xr * xr) * diffabs((Xr * Xr),((2 * (Xr * xr))
4705 + (xr * xr)))))))) + (( - 1 * ((abs((Xi * Xi)) * ((Xi * ((Xi * Xi) *
diffabs(Xi,xi))) + ((Xr * ((Xr * Xr) * diffabs(Xr,xr))) + ((xi * ((xi * xi) *
abs((Xi + xi)))) + (xr * ((xr * xr) * abs((Xr + xr))))))))) + ((abs((Xi + xi)) *
((Xi * ((Xi * Xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))) + (xi *
((xi * xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + (abs((Xr +
4710 xr)) * ((Xr * ((Xr * Xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))) +
(xr * ((xr * xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))))) + ((
- 2 * ((abs((Xi * Xr)) * (xr * ((xr * xr) * abs((Xi + xi))))) + (abs((Xi + xi))
* (xr * ((xr * xr) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))))) +
((2 * ((abs((Xi * Xr)) * (xi * ((xi * xi) * abs((Xr + xr))))) + (abs((Xr + xr))
4715 * (xi * ((xi * xi) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))))) +
(( - 2 * ((abs((Xi * Xr)) * (Xr * ((Xr * Xr) * diffabs(Xi,xi))))) + (abs((Xi +
xi)) * (Xr * ((Xr * Xr) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr +
xr)))))))))) + ((2 * ((abs((Xi * Xr)) * (Xi * ((Xi * Xi) * diffabs(Xr,xr))))) +
(abs((Xr + xr)) * (Xi * ((Xi * Xi) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr +
```

```
4720  xr))))))))) + (( - 3 * (abs((Xr * Xr)) * (xi * ((xr * xr) * abs((Xi + xi))))))
      + ((3 * (abs((Xi * Xi)) * (xi * ((xr * xr) * abs((Xi + xi)))))) + (( - 3 *
      (abs((Xi + xi)) * (xi * ((xr * xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr *
      xr))))))) + ((3 * (abs((Xi + xi)) * (xi * ((xr * xr) * diffabs((Xi * Xi),((2 *
      (Xi * xi)) + (xi * xi))))))) + (( - 3 * ((abs((Xr * Xr)) * ((xi * xi) * (xr *
4725  abs((Xr + xr))))) + (abs((Xr + xr)) * ((xi * xi) * (xr * diffabs((Xr * Xr),((2
      * (Xr * xr)) + (xr * xr)))))))) + ((3 * ((abs((Xi * Xi)) * ((xi * xi) * (xr *
      abs((Xr + xr))))) + ((abs((Xr * Xr)) * (Xr * ((xr * xr) * abs((Xr + xr))))) +
      (abs((Xr + xr)) * ((Xr * ((xr * xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr
      * xr))))) + ((xi * xi) * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
4730  xi)))))))))) + (( - 3 * ((abs((Xi * Xi)) * (Xr * ((xr * xr) * abs((Xr + xr)))))
      + ((abs((Xr * Xr)) * (Xr * ((xi * xi) * abs((Xr + xr))))) + (abs((Xr + xr)) *
      (Xr * (((xi * xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))) + ((xr *
      xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))))) + ((3 * ((abs((Xi
      * Xi)) * (Xr * ((xi * xi) * abs((Xr + xr))))) + ((abs((Xr * Xr)) * ((Xr * Xr) *
4735  (xr * abs((Xr + xr))))) + (abs((Xr + xr)) * (((Xr * Xr) * (xr * diffabs((Xr *
      Xr),((2 * (Xr * xr)) + (xr * xr))))) + (Xr * ((xi * xi) * diffabs((Xi * Xi),((2
      * (Xi * xi)) + (xi * xi))))))))) + (( - 3 * ((abs((Xi * Xi)) * ((Xr * Xr) *
      (xr * abs((Xr + xr))))) + ((abs((Xr * Xr)) * ((Xr * Xr) * (xi * abs((Xi +
      xi))))) + (abs((Xr + xr)) * ((Xr * Xr) * (xr * diffabs((Xi * Xi),((2 * (Xi *
4740  xi)) + (xi * xi)))))))) + ((3 * (abs((Xi * Xi)) * ((Xr * Xr) * (xi * abs((Xi +
      xi)))))) + (( - 3 * (abs((Xi + xi)) * ((Xr * Xr) * (xi * diffabs((Xr * Xr),((2
      * (Xr * xr)) + (xr * xr))))))) + ((3 * (abs((Xi + xi)) * ((Xr * Xr) * (xi *
      diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))) + (( - 3 * (abs((Xr * Xr))
      * (Xi * ((xr * xr) * abs((Xi + xi))))))) + ((3 * (abs((Xi * Xi)) * (Xi * ((xr *
4745  xr) * abs((Xi + xi)))))) + (( - 3 * (abs((Xi + xi)) * (Xi * ((xr * xr) *
      diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + ((3 * ((abs((Xr * Xr)) *
      (Xi * ((xi * xi) * abs((Xi + xi))))) + (abs((Xi + xi)) * (Xi * ((xr * xr) *
      diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + (( - 3 * (abs((Xi *
      Xi)) * (Xi * ((xi * xi) * abs((Xi + xi)))))) + ((3 * (abs((Xi + xi)) * (Xi *
4750  ((xi * xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + (( - 3 *
      ((abs((Xr * Xr)) * (Xi * ((Xr * Xr) * diffabs(Xi,xi)))) + (abs((Xi + xi)) * (Xi
      * ((xi * xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + ((3 *
      (abs((Xi * Xi)) * (Xi * ((Xr * Xr) * diffabs(Xi,xi))))) + (( - 3 * (abs((Xi +
      xi)) * (Xi * ((Xr * Xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))))))
4755  + ((3 * (abs((Xi + xi)) * (Xi * ((Xr * Xr) * diffabs((Xi * Xi),((2 * (Xi * xi))
      + (xi * xi))))))) + (( - 3 * ((abs((Xr * Xr)) * ((Xi * Xi) * (xr * abs((Xr +
      xr))))) + (abs((Xr + xr)) * ((Xi * Xi) * (xr * diffabs((Xr * Xr),((2 * (Xr *
      xr)) + (xr * xr)))))))) + ((3 * ((abs((Xi * Xi)) * ((Xi * Xi) * (xr * abs((Xr +
      xr))))) + ((abs((Xr * Xr)) * ((Xi * Xi) * (xi * abs((Xi + xi)))))) + (abs((Xr +
4760  xr)) * ((Xi * Xi) * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
      xi)))))))) + (( - 3 * (abs((Xi * Xi)) * ((Xi * Xi) * (xi * abs((Xi + xi))))))
      + ((3 * (abs((Xi + xi)) * ((Xi * Xi) * (xi * diffabs((Xr * Xr),((2 * (Xr * xr))
      + (xr * xr))))))) + (( - 3 * ((abs((Xr * Xr)) * ((Xi * Xi) * (Xr *
      diffabs(Xr,xr)))) + ((abs((Xi + xi)) * ((Xi * Xi) * (xi * diffabs((Xi * Xi),((2
4765  * (Xi * xi)) + (xi * xi)))))) + (abs((Xr + xr)) * ((Xi * Xi) * (Xr *
      diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))))))) + ((3 * ((abs((Xi * Xi))
      * ((Xi * Xi) * (Xr * diffabs(Xr,xr)))) + (abs((Xr + xr)) * ((Xi * Xi) * (Xr *
      diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))) + (( - 6 * ((abs((Xi *
      Xr)) * (xi * ((xr * xr) * abs((Xr + xr))))) + (abs((Xr + xr)) * (xi * ((xr *
4770  xr) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + ((6 * ((abs((Xi
      * Xr)) * ((xi * xi) * (xr * abs((Xi + xi))))) + (abs((Xr + xi)) * ((xi * xi) *
      (xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + (( - 6 *
      ((abs((Xi * Xr)) * (Xr * ((xr * xr) * abs((Xi + xi))))) + ((abs((Xr * Xr)) *
      (Xr * (xi * (xr * abs((Xi + xi)))))) + (abs((Xr + xi)) * (Xr * ((xr * xr) *
4775  diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + ((6 * (abs((Xi * Xi))
      * (Xr * (xi * (xr * abs((Xi + xi))))))) + (( - 6 * (abs((Xr + xi)) * (Xr * (xi
```

```
* (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))))) + ((6 * ((abs((Xi
* Xr)) * (Xr * ((xi * xi) * abs((Xi + xi))))) + (abs((Xi + xi)) * (Xr * (((xi *
xi) * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))) + (xi * (xr *
diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))))) + (( - 6 * ((abs((Xi *
Xr)) * ((Xi * ((xr * xr) * abs((Xr + xr)))) + ((Xr * Xr) * ((xi * abs((Xr +
xr))) + (xr * abs((Xi + xi))))))) + ((abs((Xr * Xr)) * (Xi * (xi * (xr *
abs((Xr + xr)))))) + ((abs((Xi + xi)) * ((Xr * Xr) * (xr * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr))))))) + (abs((Xr + xr)) * ((Xi * ((xi * (xr *
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))) + ((xr * xr) * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr))))))) + ((Xr * Xr) * (xi * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))))))))) + ((6 * ((abs((Xi * Xi)) * (Xi *
(xi * (xr * abs((Xr + xr)))))) + ((abs((Xi * Xr)) * (Xi * ((xi * xi) * abs((Xr
+ xr))))) + (abs((Xr + xr)) * (Xi * (((xi * xi) * diffabs((Xi * Xr),((Xi * xr)
+ (xi * (Xr + xr))))) + (xi * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
xi))))))))))) + (( - 6 * (abs((Xr * Xr)) * (Xi * (Xr * (xr * abs((Xi +
xi))))))) + ((6 * (abs((Xi * Xi)) * (Xi * (Xr * (xr * abs((Xi + xi))))))) + ((
- 6 * (abs((Xi + xi)) * (Xi * (Xr * (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) +
(xr * xr)))))))) + ((6 * (abs((Xi + xi)) * (Xi * (Xr * (xr * diffabs((Xi *
Xi),((2 * (Xi * xi)) + (xi * xi))))))))) + (( - 6 * ((abs((Xr * Xr)) * (Xi * (Xr
* (xi * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * (Xr * (xi * diffabs((Xr *
Xr),((2 * (Xr * xr)) + (xr * xr))))))))) + ((6 * ((abs((Xi * Xi)) * (Xi * (Xr *
(xi * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * (Xr * (xi * diffabs((Xi *
Xi),((2 * (Xi * xi)) + (xi * xi))))))))) + (( - 6 * ((abs((Xi * Xr)) * (Xi *
((Xr * Xr) * diffabs(Xr,xr)))) + (abs((Xr + xr)) * (Xi * ((Xr * Xr) *
diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + ((6 * ((abs((Xi * Xr))
* ((Xi * Xi) * ((Xr * diffabs(Xi,xi)) + ((xi * abs((Xr + xr))) + (xr * abs((Xi
+ xi))))))) + ((abs((Xi + xi)) * ((Xi * Xi) * (diffabs((Xi * Xr),((Xi * xr) +
(xi * (Xr + xr)))) * (Xr + xr)))) + (abs((Xr + xr)) * ((Xi * Xi) * (xi *
diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))))) + (( - 12 * ((abs((Xi *
Xr)) * (Xr * (xi * (xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xr * (xi * (xr
* diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))))) + ((12 * ((abs((Xi *
Xr)) * (Xi * (xi * (xr * abs((Xi + xi)))))) + (abs((Xi + xi)) * (Xi * (xi * (xr
* diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))))) + (( - 12 * ((abs((Xi
* Xr)) * (Xi * (Xr * (xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * (Xr *
(xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))))) + ((12 *
((abs((Xi * Xr)) * (Xi * (Xr * (xi * abs((Xi + xi)))))) + (abs((Xi + xi)) * (Xi
* (Xr * (xi * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))))) +
(abs((Xr * Xr)) * (xr * ((xr * xr) * abs((Xr +
xr))))))))))
```

```
Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + (abs((Xr + xr)) * ((Xr * ((Xr * Xr) *
4835 diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))) + (xr * ((xr * xr) *
diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))))) + ((3 * ((abs((Xr *
Xr)) * (xi * ((xr * xr) * abs((Xr + xr))))) + (abs((Xr + xr)) * (xi * ((xr *
xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))))))) + (( - 3 *
((abs((Xi * Xi)) * (xi * ((xr * xr) * abs((Xr + xr))))) + ((abs((Xr * Xr)) *
4840 ((xi * xi) * (xr * abs((Xi + xi))))) + (abs((Xr + xr)) * (xi * ((xr * xr) *
diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + ((3 * (abs((Xi * Xi))
* ((xi * xi) * (xr * abs((Xi + xi)))))) + (( - 3 * (abs((Xi + xi)) * ((xi * xi)
* (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + ((3 * ((abs((Xr
* Xr)) * (Xr * ((xr * xr) * abs((Xi + xi))))) + (abs((Xi + xi)) * ((xi * xi) *
4845 (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + (( - 3 * (abs((Xi
* Xi)) * (Xr * ((xr * xr) * abs((Xi + xi)))))) + ((3 * (abs((Xi + xi)) * (Xr *
((xr * xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + (( - 3 *
((abs((Xr * Xr)) * (Xr * ((xi * xi) * abs((Xi + xi))))) + (abs((Xi + xi)) * (Xr
* ((xr * xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + ((3 *
4850 (abs((Xi * Xi)) * (Xr * ((xi * xi) * abs((Xi + xi)))))) + (( - 3 * (abs((Xi +
xi)) * (Xr * ((xi * xi) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))
+ ((3 * ((abs((Xr * Xr)) * ((Xr * Xr) * (xr * abs((Xi + xi))))) + (abs((Xi +
xi)) * (Xr * ((xi * xi) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi)))))))
+ (( - 3 * (abs((Xi * Xi)) * ((Xr * Xr) * (xr * abs((Xi + xi)))))) + ((3 *
4855 (abs((Xi + xi)) * ((Xr * Xr) * (xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr *
xr))))))) + (( - 3 * (abs((Xi + xi)) * ((Xr * Xr) * (xr * diffabs((Xi * Xi),((2
* (Xi * xi)) + (xi * xi))))))) + ((3 * ((abs((Xr * Xr)) * ((Xr * Xr) * (xi *
abs((Xr + xr))))) + (abs((Xr + xr)) * ((Xr * Xr) * (xi * diffabs((Xr * Xr),((2
* (Xr * xr)) + (xr * xr)))))))) + (( - 3 * ((abs((Xi * Xi)) * ((Xr * Xr) * (xi
4860 * abs((Xr + xr))))) + (abs((Xr + xr)) * ((Xr * Xr) * (xi * diffabs((Xi *
Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + ((3 * ((abs((Xr * Xr)) * (Xi * ((xr *
xr) * abs((Xr + xr))))) + (abs((Xr + xr)) * (Xi * ((xr * xr) * diffabs((Xr *
Xr),((2 * (Xr * xr)) + (xr * xr)))))))) + (( - 3 * ((abs((Xi * Xi)) * (Xi *
((xr * xr) * abs((Xr + xr))))) + ((abs((Xr * Xr)) * (Xi * ((xi * xi) * abs((Xr
4865 + xr))))) + (abs((Xr + xr)) * (Xi * (((xi * xi) * diffabs((Xr * Xr),((2 * (Xr *
xr)) + (xr * xr)))) + ((xr * xr) * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
xi))))))))))) + ((3 * ((abs((Xi * Xi)) * (Xi * ((xi * xi) * abs((Xr + xr)))))) +
((abs((Xr * Xr)) * (Xi * ((Xr * Xr) * diffabs(Xr,xr)))) + (abs((Xr + xr)) * (Xi
* (((Xr * Xr) * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))) + ((xi * xi)
4870 * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))) + (( - 3 * ((abs((Xi
* Xi)) * (Xi * ((Xr * Xr) * diffabs(Xr,xr)))) + ((abs((Xr * Xr)) * ((Xi * Xi) *
(xr * abs((Xi + xi))))) + (abs((Xr + xr)) * (Xi * ((Xr * Xr) * diffabs((Xi *
Xi),((2 * (Xi * xi)) + (xi * xi)))))))) + ((3 * (abs((Xi * Xi)) * ((Xi * Xi) *
(xr * abs((Xi + xi)))))) + (( - 3 * (abs((Xi + xi)) * ((Xi * Xi) * (xr *
4875 diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + ((3 * (abs((Xi + xi)) *
((Xi * Xi) * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))) + (( -
3 * ((abs((Xr * Xr)) * ((Xi * Xi) * (xi * abs((Xr + xr))))) + (abs((Xr + xr)) *
((Xi * Xi) * (xi * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))))))) + ((3
* ((abs((Xi * Xi)) * ((Xi * Xi) * (xi * abs((Xr + xr))))) + (abs((Xr + xr)) *
4880 ((Xi * Xi) * (xi * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))) + (( -
3 * (abs((Xr * Xr)) * ((Xi * Xi) * (Xr * diffabs(Xi,xi))))) + ((3 * (abs((Xi *
Xi)) * ((Xi * Xi) * (Xr * diffabs(Xi,xi))))) + (( - 3 * (abs((Xi + xi)) * ((Xi
* Xi) * (Xr * diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))))) + ((3 *
(abs((Xi + xi)) * ((Xi * Xi) * (Xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
4885 xi))))))) + (( - 6 * ((abs((Xi * Xr)) * (((xi * xi) * (xr * abs((Xr + xr)))) +
(xi * ((xr * xr) * abs((Xi + xi)))))) + ((abs((Xi + xi)) * (xi * ((xr * xr) *
diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))) + (abs((Xr + xr)) * ((xi *
xi) * (xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))))) + ((6 *
((abs((Xi * Xr)) * (Xr * ((xr * xr) * abs((Xr + xr))))) + ((abs((Xr * Xr)) *
4890 (Xr * (xi * (xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xr * ((xi * (xr *
```

```
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr))))) + ((xr * xr) * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))))))))) + (( - 6 * ((abs((Xi * Xi)) * (Xr *
(xi * (xr * abs((Xr + xr)))))) + ((abs((Xi * Xr)) * (Xr * ((xi * xi) * abs((Xr
+ xr))))) + (abs((Xr + xr)) * (Xr * (((xi * xi) * diffabs((Xi * Xr),((Xi * xr)
+ (xi * (Xr + xr))))) + (xi * (xr * diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi *
xi)))))))))))) + ((6 * ((abs((Xi * Xr)) * ((Xr * Xr) * (xr * abs((Xr + xr)))))) +
(abs((Xr + xr)) * ((Xr * Xr) * (xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr +
xr)))))))))) + (( - 6 * ((abs((Xi * Xr)) * ((Xi * ((xr * xr) * abs((Xi + xi))))
+ ((Xr * Xr) * (xi * abs((Xi + xi)))))))) + ((abs((Xr * Xr)) * (Xi * (xi * (xr *
abs((Xi + xi)))))))) + ((abs((Xi + xi)) * ((Xi * ((xr * xr) * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + ((Xr * Xr) * (xi * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))))))))) + ((6 * (abs((Xi * Xi)) * (Xi * (xi
* (xr * abs((Xi + xi))))))) + (( - 6 * (abs((Xi + xi)) * (Xi * (xi * (xr *
diffabs((Xr * Xr),((2 * (Xr * xr)) + (xr * xr)))))))) + ((6 * ((abs((Xi * Xr))
* (Xi * ((xi * xi) * abs((Xi + xi)))))) + ((abs((Xr * Xr)) * (Xi * (Xr * (xr *
abs((Xr + xr)))))) + ((abs((Xi + xi)) * (Xi * (((xi * xi) * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))) + (xi * (xr * diffabs((Xi * Xi),((2 * (Xi
* xi)) + (xi * xi))))))))) + (abs((Xr + xr)) * (Xi * (Xr * (xr * diffabs((Xr *
Xr),((2 * (Xr * xr)) + (xr * xr)))))))))))) + (( - 6 * ((abs((Xi * Xi)) * (Xi *
(Xr * (xr * abs((Xr + xr)))))) + ((abs((Xr * Xr)) * (Xi * (Xr * (xi * abs((Xi +
xi)))))) + (abs((Xr + xr)) * (Xi * (Xr * (xr * diffabs((Xi * Xi),((2 * (Xi *
xi)) + (xi * xi)))))))))) + ((6 * (abs((Xi * Xi)) * (Xi * (Xr * (xi * abs((Xi +
xi))))))) + (( - 6 * (abs((Xi + xi)) * (Xi * (Xr * (xi * diffabs((Xr * Xr),((2
* (Xr * xr)) + (xr * xr))))))))) + ((6 * (abs((Xi + xi)) * (Xi * (Xr * (xi *
diffabs((Xi * Xi),((2 * (Xi * xi)) + (xi * xi))))))))) + (( - 6 * ((abs((Xi *
Xr)) * (((Xi * Xi) * (xr * abs((Xr + xr)))) + (Xi * ((Xr * Xr) *
diffabs(Xi,xi))))) + ((abs((Xi + xi)) * (Xi * ((Xr * Xr) * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr))))))) + (abs((Xr + xr)) * ((Xi * Xi) * (xr *
diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))))) + ((6 * ((abs((Xi *
Xr)) * ((Xi * Xi) * (xi * abs((Xi + xi))))) + (abs((Xi + xi)) * ((Xi * Xi) *
(xi * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr)))))))))) + (( - 6 *
((abs((Xi * Xr)) * ((Xi * Xi) * (Xr * diffabs(Xr,xr)))) + (abs((Xr + xr)) *
((Xi * Xi) * (Xr * diffabs((Xi * Xr),((Xi * xr) + (xi * (Xr + xr))))))))) + ((
- 12 * ((abs((Xi * Xr)) * ((Xi * ((Xr * ((xi * abs((Xr + xr))) + (xr * abs((Xi
+ xi))))) + (xi * (xr * abs((Xr + xr)))))) + (Xr * (xi * (xr * abs((Xi +
xi))))))) + ((abs((Xi + xi)) * ((Xi * (Xr * (xr * diffabs((Xi * Xr),((Xi * xr)
+ (xi * (Xr + xr))))))) + (Xr * (xi * (xr * diffabs((Xi * Xr),((Xi * xr) + (xi
* (Xr + xr)))))))) + (abs((Xr + xr)) * (Xi * ((Xr * (xi * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))) + (xi * (xr * diffabs((Xi * Xr),((Xi *
xr) + (xi * (Xr + xr)))))))))) + (abs((Xr * Xr)) * (xr * ((xr * xr) * abs((Xi
+ xi))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))));
      </perturbation>
      <derivative t='M>
        dxan = ((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dya * abs((Xxr * Xxr)))))) +
((sgn(Xxr) * (Xxr * ((Xxr * Xxr) * (dxa * abs((Xxr * Xxr)))))) + (( - 1 *
((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dya * abs((Xxi * Xxi)))))) + (sgn(Xxr) *
(Xxr * ((Xxr * Xxr) * (dxa * abs((Xxi * Xxi))))))))) + (( - 2 * (sgn(Xxi) *
((Xxr * ((Xxr * Xxr) * (dya * abs((Xxi * Xxr))))) + (sgn(Xxr) * (((Xxr * Xxr) *
(Xxr * Xxr)) * (dya * abs(Xxi)))))))) + ((2 * ((sgn(Xxr) * sgn(Xxr)) * (((Xxr *
Xxr) * (Xxr * Xxr)) * (dxa * abs(Xxr))))) + (( - 2 * (((sgn(Xxi) * sgn(Xxi)) *
(Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) *
(Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxi))))))))) + ((2 * ((sgn(Xxi) *
(sgn(Xxr) * ((((Xxi * Xxi) * (Xxi * Xxi)) * (dxa * abs(Xxr))) + (Xxi * ((Xxi *
Xxi) * (Xxr * (dya * abs(Xxr)))))))) + (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa *
abs((Xxi * Xxr))))))))) + (( - 2 * ((sgn(Xxi) * sgn(Xxi)) * (((Xxi * Xxi) * (Xxi
* Xxi)) * (dya * abs(Xxi))))) + ((2 * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dxa * abs(Xxi))))))) + (( - 3 * (abs((Xxr * Xxr)) * ((Xxr * Xxr)
```

```
* (dya * abs(Xxi))))) + ((3 * ((abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dya *
abs(Xxi)))) + (abs((Xxr * Xxr)) * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + (( - 3
* ((abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dxa * abs(Xxr)))) + (sgn(Xxi) * (Xxi *
((Xxr * Xxr) * (dya * abs((Xxr * Xxr)))))))) + ((3 * ((abs((Xxr * Xxr)) * ((Xxi
* Xxi) * (dya * abs(Xxi)))) + (sgn(Xxi) * (Xxi * ((Xxr * Xxr) * (dya * abs((Xxi
* Xxi))))))) + (( - 3 * ((abs((Xxi * Xxi)) * ((Xxi * Xxi) * (dya * abs(Xxi))))
+ (abs((Xxr * Xxr)) * ((Xxi * Xxi) * (dxa * abs(Xxr)))))) + ((3 * (abs((Xxi *
Xxi)) * ((Xxi * Xxi) * (dxa * abs(Xxr))))) + (( - 3 * (sgn(Xxr) * ((Xxi * Xxi)
* (Xxr * (dxa * abs((Xxr * Xxr))))))) + ((3 * (sgn(Xxr) * ((Xxi * Xxi) * (Xxr *
(dxa * abs((Xxi * Xxi))))))) + (( - 6 * ((abs((Xxi * Xxr)) * ((Xxr * Xxr) *
((dxa * abs(Xxi)) + (dya * abs(Xxr))))) + (abs((Xxr * Xxr)) * (Xxi * (Xxr *
(dya * abs(Xxr))))))) + ((6 * (abs((Xxi * Xxi)) * (Xxi * (Xxr * (dya *
abs(Xxr)))))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi * (Xxr * (dxa * abs(Xxi))))))
+ ((6 * (abs((Xxi * Xxi)) * (Xxi * (Xxr * (dxa * abs(Xxi)))))) + (( - 6 *
((sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr))))))) +
((sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxa * abs((Xxi * Xxr))))))) + ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxi)))))))) + ((6 *
((abs((Xxi * Xxr)) * ((Xxi * Xxi) * ((dxa * abs(Xxi)) + (dya * abs(Xxr))))) +
((sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dya * abs((Xxi * Xxr)))))) + (((sgn(Xxi) *
sgn(Xxi)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi))))) + (sgn(Xxi) *
(sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi)))))))))) + (( - 6 *
((sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) +
((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr))))))) +
((6 * (((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) * (Xxr * (dya *
abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr * (dxa *
abs(Xxi)))))))) + ((12 * (abs((Xxi * Xxr)) * (Xxi * (Xxr * (dya *
abs(Xxi)))))) + (( - 12 * (abs((Xxi * Xxr)) * (Xxi * (Xxr * (dxa *
abs(Xxr)))))) + daa))))))))))))))))))))))))));
        dxbn = ((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dyb * abs((Xxr * Xxr)))))) +
((sgn(Xxr) * (Xxr * ((Xxr * Xxr) * (dxb * abs((Xxr * Xxr)))))) + (( - 1 *
((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dyb * abs((Xxi * Xxi)))))) + (sgn(Xxr) *
(Xxr * ((Xxr * Xxr) * (dxb * abs((Xxi * Xxi)))))))) + (( - 2 * (sgn(Xxi) *
((Xxr * ((Xxr * Xxr) * (dyb * abs((Xxi * Xxr))))) + (sgn(Xxr) * (((Xxr * Xxr) *
(Xxr * Xxr)) * (dyb * abs(Xxi)))))) + ((2 * ((sgn(Xxr) * sgn(Xxr)) * (((Xxr *
Xxr) * (Xxr * Xxr)) * (dxb * abs(Xxr))))) + (( - 2 * (((sgn(Xxi) * sgn(Xxi)) *
(Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) *
(Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxi)))))))) + ((2 * ((sgn(Xxi) *
(sgn(Xxr) * ((((Xxi * Xxi) * (Xxi * Xxi)) * (dxb * abs(Xxr))) + (Xxi * ((Xxi *
Xxi) * (Xxr * (dyb * abs(Xxr))))))) + (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb *
abs((Xxi * Xxr)))))))) + (( - 2 * ((sgn(Xxi) * sgn(Xxi)) * (((Xxi * Xxi) * (Xxi
* Xxi)) * (dyb * abs(Xxi))))) + ((2 * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dxb * abs(Xxi))))))) + (( - 3 * (abs((Xxr * Xxr)) * ((Xxr * Xxr)
* (dyb * abs(Xxi))))) + ((3 * ((abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dyb *
abs(Xxi)))) + (abs((Xxr * Xxr)) * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + (( - 3
* ((abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dxb * abs(Xxr)))) + (sgn(Xxi) * (Xxi *
((Xxr * Xxr) * (dyb * abs((Xxr * Xxr)))))))) + ((3 * ((abs((Xxr * Xxr)) * ((Xxi
* Xxi) * (dyb * abs(Xxi)))) + (sgn(Xxi) * (Xxi * ((Xxr * Xxr) * (dyb * abs((Xxi
* Xxi))))))) + (( - 3 * ((abs((Xxi * Xxi)) * ((Xxi * Xxi) * (dyb * abs(Xxi))))
+ (abs((Xxr * Xxr)) * ((Xxi * Xxi) * (dxb * abs(Xxr)))))) + ((3 * (abs((Xxi *
Xxi)) * ((Xxi * Xxi) * (dxb * abs(Xxr))))) + (( - 3 * (sgn(Xxr) * ((Xxi * Xxi)
* (Xxr * (dxb * abs((Xxr * Xxr))))))) + ((3 * (sgn(Xxr) * ((Xxi * Xxi) * (Xxr *
(dxb * abs((Xxi * Xxi))))))) + (( - 6 * ((abs((Xxi * Xxr)) * ((Xxr * Xxr) *
((dxb * abs(Xxi)) + (dyb * abs(Xxr))))) + (abs((Xxr * Xxr)) * (Xxi * (Xxr *
(dyb * abs(Xxr))))))) + ((6 * (abs((Xxi * Xxi)) * (Xxi * (Xxr * (dyb *
abs(Xxr)))))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi * (Xxr * (dxb * abs(Xxi))))))
+ ((6 * (abs((Xxi * Xxi)) * (Xxi * (Xxr * (dxb * abs(Xxi)))))) + (( - 6 *
((sgn(Xxi) * (sgn(Xxr) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr))))))) +
```

```
((sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxb * abs((Xxi * Xxr)))))) + ((sgn(Xxr) *
sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxi)))))))) + ((6 *
((abs((Xxi * Xxr)) * ((Xxi * Xxi) * ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))) +
((sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dyb * abs((Xxi * Xxr)))))) + (((sgn(Xxi) *
sgn(Xxi)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi))))) + (sgn(Xxi) *
(sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi))))))))) + (( - 6 *
((sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) +
((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) +
((6 * (((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) * (Xxr * (dyb *
abs(Xxr)))))) + (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (Xxr * (dxb *
abs(Xxi))))))))) + ((12 * (abs((Xxi * Xxr)) * (Xxi * (Xxr * (dyb *
abs(Xxi)))))) + (( - 12 * (abs((Xxi * Xxr)) * (Xxi * (Xxr * (dxb *
abs(Xxr)))))) + dab)))))))))))))))))))))))));
        dyan = ((sgn(Xxi) * (Xxr * ((Xxr * Xxr) * (dya * abs((Xxr * Xxr)))))) +
((sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa * abs((Xxi * Xxi)))))) + (( - 1 *
((sgn(Xxi) * (Xxr * ((Xxr * Xxr) * (dya * abs((Xxi * Xxi)))))) + (sgn(Xxr) *
(Xxi * ((Xxi * Xxi) * (dxa * abs((Xxr * Xxr))))))))) + ((2 * ((sgn(Xxi) *
(sgn(Xxr) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dya * abs(Xxr))))) + ((sgn(Xxr) *
(Xxr * ((Xxr * Xxr) * (dxa * abs((Xxi * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) *
(((Xxr * Xxr) * (Xxr * Xxr)) * (dxa * abs(Xxi))))))) + (( - 2 * ((sgn(Xxi) *
sgn(Xxi)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxi))))))) + ((2 *
((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dya * abs((Xxi * Xxr)))))) + (((sgn(Xxi) *
sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dya * abs(Xxr)))) + (sgn(Xxi) *
(sgn(Xxr) * ((((Xxi * Xxi) * (Xxi * Xxi)) * (dxa * abs(Xxi))) + (Xxi * (((Xxi *
Xxi) * (Xxr * (dya * abs(Xxi)))) + (Xxr * ((Xxr * Xxr) * (dxa *
abs(Xxr))))))))))) + (( - 2 * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dxa * abs(Xxr))))))) + ((3 * (abs((Xxr * Xxr)) * ((Xxr * Xxr) * (dya *
abs(Xxr))))) + (( - 3 * (abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dya * abs(Xxr)))))
+ ((3 * (abs((Xxr * Xxr)) * ((Xxr * Xxr) * (dxa * abs(Xxi))))) + (( - 3 *
(abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dxa * abs(Xxi))))) + ((3 * (sgn(Xxr) * (Xxi
* ((Xxr * Xxr) * (dxa * abs((Xxr * Xxr))))))) + (( - 3 * ((abs((Xxr * Xxr)) *
((Xxi * Xxi) * (dya * abs(Xxr)))) + (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxa *
abs((Xxi * Xxi)))))))) + ((3 * (abs((Xxi * Xxi)) * ((Xxi * Xxi) * (dya *
abs(Xxr))))) + (( - 3 * (abs((Xxr * Xxr)) * ((Xxi * Xxi) * (dxa * abs(Xxi)))))
+ ((3 * (abs((Xxi * Xxi)) * ((Xxi * Xxi) * (dxa * abs(Xxi))))) + (( - 3 *
(sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dya * abs((Xxr * Xxr))))))) + ((3 *
(sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dya * abs((Xxi * Xxi))))))) + (( - 6 *
(abs((Xxi * Xxr)) * ((Xxr * Xxr) * (dya * abs(Xxi))))) + ((6 * (abs((Xxi *
Xxr)) * ((Xxr * Xxr) * (dxa * abs(Xxr))))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi
* (Xxr * (dya * abs(Xxi)))))) + ((6 * ((abs((Xxi * Xxi)) * (Xxi * (Xxr * (dya *
abs(Xxi))))) + (abs((Xxr * Xxr)) * (Xxi * (Xxr * (dxa * abs(Xxr))))))) + (( - 6
* ((abs((Xxi * Xxi)) * (Xxi * (Xxr * (dxa * abs(Xxr)))))) + (sgn(Xxi) * ((Xxi *
((Xxr * Xxr) * (dya * abs((Xxi * Xxr)))))) + (sgn(Xxr) * (Xxi * (Xxr * ((Xxr *
Xxr) * (dya * abs(Xxi))))))))) + ((6 * ((abs((Xxi * Xxr)) * ((Xxi * Xxi) *
(dya * abs(Xxi))))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa
* abs(Xxr))))))) + (( - 6 * ((abs((Xxi * Xxr)) * ((Xxi * Xxi) * (dxa *
abs(Xxr))))) + (((sgn(Xxi) * sgn(Xxi)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya *
abs(Xxr))))) + ((sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * ((dxa *
abs(Xxi)) + (dya * abs(Xxr))))))) + ((sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dxa *
abs((Xxi * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) *
(dxa * abs(Xxi))))))))))) + ((6 * ((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dya * abs(Xxi))))))) + (( - 6 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dxa * abs(Xxr))))))) + (( - 12 * (abs((Xxi * Xxr)) * (Xxi *
(Xxr * ((dxa * abs(Xxi)) + (dya * abs(Xxr)))))) +
dba)))))))))))))))))))))))));
        dybn = ((sgn(Xxi) * (Xxr * ((Xxr * Xxr) * (dyb * abs((Xxr * Xxr)))))) +
((sgn(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb * abs((Xxi * Xxi)))))) + (( - 1 *
```

```
((sgn(Xxi) * (Xxr * ((Xxr * Xxr) * (dyb * abs((Xxi * Xxi)))))) + (sgn(Xxr) *
(Xxi * ((Xxi * Xxi) * (dxb * abs((Xxr * Xxr)))))))) + ((2 * ((sgn(Xxi) *
(sgn(Xxr) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dyb * abs(Xxr))))) + ((sgn(Xxr) *
(Xxr * ((Xxr * Xxr) * (dxb * abs((Xxi * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) *
(((Xxr * Xxr) * (Xxr * Xxr)) * (dxb * abs(Xxi)))))))) + (( - 2 * ((sgn(Xxi) *
sgn(Xxi)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxi))))))) + ((2 *
((sgn(Xxi) * (Xxi * ((Xxi * Xxi) * (dyb * abs((Xxi * Xxr)))))) + (((sgn(Xxi) *
sgn(Xxi)) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dyb * abs(Xxr)))) + (sgn(Xxi) *
(sgn(Xxr) * ((((Xxi * Xxi) * (Xxi * Xxi)) * (dxb * abs(Xxi))) + (Xxi * (((Xxi *
Xxi) * (Xxr * (dyb * abs(Xxi)))) + (Xxr * ((Xxr * Xxr) * (dxb *
abs(Xxr))))))))))) + (( - 2 * ((sgn(Xxr) * sgn(Xxr)) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dxb * abs(Xxr))))))) + ((3 * (abs((Xxr * Xxr)) * ((Xxr * Xxr) * (dyb *
abs(Xxr))))) + (( - 3 * (abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dyb * abs(Xxr)))))
+ ((3 * (abs((Xxr * Xxr)) * ((Xxr * Xxr) * (dxb * abs(Xxi))))) + (( - 3 *
(abs((Xxi * Xxi)) * ((Xxr * Xxr) * (dxb * abs(Xxi))))) + ((3 * (sgn(Xxr) * (Xxi
* ((Xxr * Xxr) * (dxb * abs((Xxr * Xxr))))))) + (( - 3 * ((abs((Xxr * Xxr)) *
((Xxi * Xxi) * (dyb * abs(Xxr)))) + (sgn(Xxr) * (Xxi * ((Xxr * Xxr) * (dxb *
abs((Xxi * Xxi))))))))) + ((3 * (abs((Xxi * Xxi)) * ((Xxi * Xxi) * (dyb *
abs(Xxr))))) + (( - 3 * (abs((Xxr * Xxr)) * ((Xxi * Xxi) * (dxb * abs(Xxi)))))
+ ((3 * (abs((Xxi * Xxi)) * ((Xxi * Xxi) * (dxb * abs(Xxi))))) + (( - 3 *
(sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dyb * abs((Xxr * Xxr))))))) + ((3 *
(sgn(Xxi) * ((Xxi * Xxi) * (Xxr * (dyb * abs((Xxi * Xxi))))))) + (( - 6 *
(abs((Xxi * Xxr)) * ((Xxr * Xxr) * (dyb * abs(Xxi))))) + ((6 * (abs((Xxi *
Xxr)) * ((Xxr * Xxr) * (dxb * abs(Xxr))))) + (( - 6 * (abs((Xxr * Xxr)) * (Xxi
* (Xxr * (dyb * abs(Xxi)))))) + ((6 * ((abs((Xxi * Xxi)) * (Xxi * (Xxr * (dyb *
abs(Xxi))))) + (abs((Xxr * Xxr)) * (Xxi * (Xxr * (dxb * abs(Xxr))))))) + (( - 6
* ((abs((Xxi * Xxi)) * (Xxi * (Xxr * (dxb * abs(Xxr)))))) + (sgn(Xxi) * ((Xxi *
((Xxr * Xxr) * (dyb * abs((Xxi * Xxr))))) + (sgn(Xxr) * (Xxi * (Xxr * ((Xxr *
Xxr) * (dyb * abs(Xxi))))))))))) + ((6 * ((abs((Xxi * Xxr)) * ((Xxi * Xxi) *
(dyb * abs(Xxi)))) + ((sgn(Xxr) * sgn(Xxr)) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb
* abs(Xxr))))))) + (( - 6 * ((abs((Xxi * Xxr)) * ((Xxi * Xxi) * (dxb *
abs(Xxr)))) + (((sgn(Xxi) * sgn(Xxi)) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb *
abs(Xxr))))) + ((sgn(Xxi) * (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * ((dxb *
abs(Xxi)) + (dyb * abs(Xxr))))))) + ((sgn(Xxr) * ((Xxi * Xxi) * (Xxr * (dxb *
abs((Xxi * Xxr)))))) + ((sgn(Xxr) * sgn(Xxr)) * ((Xxi * Xxi) * ((Xxr * Xxr) *
(dxb * abs(Xxi))))))))))) + ((6 * ((sgn(Xxi) * sgn(Xxi)) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dyb * abs(Xxi))))))) + (( - 6 * (sgn(Xxi) * (sgn(Xxr) * (Xxi * ((Xxi *
Xxi) * (Xxr * (dxb * abs(Xxr))))))))) + (( - 12 * (abs((Xxi * Xxr)) * (Xxi *
(Xxr * ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))))) +
dbb))))))))))))))))))))))))))));
        </derivative>
      </formula>
    </group>

    <group type="67" name="Mothbrot 6th 4x2">
      <formula power="6" glitch="1.0000000000000002e-4">
        <reference t='R'>
          Xrn = (Cr + ((abs(Xr) * (((Xi * Xi) * (Xi * Xi)) * abs(Xr))) + (( - 1 *
((abs(Xi) * abs(Xi)) * (((Xi * Xi) * (Xi * Xi)) + ((Xr * Xr) * (Xr * Xr))))) +
(( - 6 * (abs(Xr) * ((Xi * Xi) * ((Xr * Xr) * abs(Xr))))) + ((6 * (abs(Xi) *
((Xi * Xi) * ((Xr * Xr) * abs(Xi))))) + (( - 8 * (abs(Xi) * (Xi * (Xr * ((Xr *
Xr) * abs(Xr)))))) + ((8 * (abs(Xi) * (Xi * ((Xi * Xi) * (Xr * abs(Xr)))))) +
(abs(Xr) * (((Xr * Xr) * (Xr * Xr)) * abs(Xr))))))))));
          Xin = (Ci + ((2 * (abs(Xi) * (((Xi * Xi) * (Xi * Xi)) * abs(Xr)))) +
((4 * (abs(Xr) * (Xi * (Xr * ((Xr * Xr) * abs(Xr)))))) + (( - 4 * (abs(Xi) *
(Xi * (Xr * ((Xr * Xr) * abs(Xi)))))) + ((4 * (abs(Xi) * (Xi * ((Xi * Xi) * (Xr
* abs(Xi)))))) + (( - 4 * (abs(Xr) * (Xi * ((Xi * Xi) * (Xr * abs(Xr)))))) + ((
```

```
    - 12 * (abs(Xi) * ((Xi * Xi) * ((Xr * Xr) * abs(Xr))))) + (2 * (abs(Xi) * (((Xr
5120 * Xr) * (Xr * Xr)) * abs(Xr)))))))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((((xi * xi) * (xi * xi)) * abs((Xr * (Xr + xr)))) +
    ((abs(Xr) * (diffabs(Xr,xr) * (((Xi * Xi) * (Xi * Xi)) + ((Xr * Xr) * (Xr *
5125 Xr))))) + ((abs((Xr + xr)) * (diffabs(Xr,xr) * (((Xi * Xi) * (Xi * Xi)) + (((Xr
    * Xr) * (Xr * Xr)) + (((xi * xi) * (xi * xi)) + ((xr * xr) * (xr * xr)))))))) +
    (( - 1 * (((Xi * Xi) * ((xr * xr) * (xr * xr))) + ((xi * xi) * (((Xr * Xr) *
    (Xr * Xr)) + (((xi * xi) * (xi * xi)) + ((xr * xr) * (xr * xr)))))))) + (( - 2 *
    (Xi * (xi * (((Xr * Xr) * (Xr * Xr)) + ((xr * xr) * (xr * xr)))))) + ((4 *
5130 ((abs(Xr) * (Xr * (xr * ((xr * xr) * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xr
    * (xr * ((xr * xr) * diffabs(Xr,xr)))))))) + (( - 4 * (Xr * ((xi * xi) * (xr *
    (xr * xr))))) + ((4 * ((abs(Xr) * (Xr * ((Xr * Xr) * (xr * abs((Xr + xr)))))) +
    (abs((Xr + xr)) * (Xr * ((Xr * Xr) * (xr * diffabs(Xr,xr)))))))) + (( - 4 * (Xr
    * ((Xr * Xr) * ((xi * xi) * xr)))) + ((4 * ((abs(Xr) * (Xi * (xi * ((xi * xi) *
5135 abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * (xi * ((xi * xi) *
    diffabs(Xr,xr)))))))) + (( - 4 * ((Xi * Xi) * (Xr * (xr * ((Xr * Xr) + (xr *
    xr)))))) + ((4 * ((abs(Xr) * (Xi * ((Xi * Xi) * (xi * abs((Xr + xr)))))) +
    (abs((Xr + xr)) * (Xi * ((Xi * Xi) * (xi * diffabs(Xr,xr)))))))) + (( - 6 *
    ((abs(Xr) * ((xi * xi) * ((xr * xr) * abs((Xr + xr))))) + (abs((Xr + xr)) *
5140 ((xi * xi) * ((xr * xr) * diffabs(Xr,xr)))))) + ((6 * ((abs(Xi) * ((xi * xi) *
    ((xr * xr) * abs((Xi + xi))))) + ((abs(Xr) * ((Xr * Xr) * ((xr * xr) * abs((Xr
    + xr))))) + ((abs((Xi + xi)) * ((xi * xi) * ((xr * xr) * diffabs(Xi,xi)))) +
    (abs((Xr + xr)) * ((Xr * Xr) * ((xr * xr) * diffabs(Xr,xr)))))))) + (( - 6 *
    ((abs(Xr) * ((Xr * Xr) * ((xi * xi) * abs((Xr + xr))))) + (abs((Xr + xr)) *
5145 ((Xr * Xr) * ((xi * xi) * diffabs(Xr,xr)))))) + ((6 * ((abs(Xi) * ((Xr * Xr) *
    ((xi * xi) * abs((Xi + xi))))) + (abs((Xi + xi)) * ((Xr * Xr) * ((xi * xi) *
    diffabs(Xi,xi)))))) + (( - 6 * ((Xi * (xi * ((xi * xi) * (xi * xi)))) + (((Xr *
    Xr) * ((xi * xi) * (xr * xr))) + ((abs(Xr) * ((Xi * Xi) * ((xr * xr) * abs((Xr
    + xr))))) + (abs((Xr + xr)) * ((Xi * Xi) * ((xr * xr) * diffabs(Xr,xr)))))))) +
5150 ((6 * ((abs(Xi) * ((Xi * Xi) * ((xr * xr) * abs((Xi + xi))))) + ((abs(Xr) *
    ((Xi * Xi) * ((xi * xi) * abs((Xr + xr))))) + ((abs((Xi + xi)) * ((Xi * Xi) *
    ((xr * xr) * diffabs(Xi,xi)))) + (abs((Xr + xr)) * ((Xi * Xi) * ((xi * xi) *
    diffabs(Xr,xr)))))))) + (( - 6 * (((Xi * Xi) * ((Xr * Xr) * diffabs(Xr,xr))) *
    (abs(Xr) + abs((Xr + xr))))) + ((6 * (((Xi * Xi) * ((Xr * Xr) *
5155 diffabs(Xi,xi))) * (abs(Xi) + abs((Xi + xi))))) + (( - 6 * ((Xi * (((Xi * Xi) *
    (Xi * Xi)) * xi)) + ((Xi * Xi) * ((Xr * Xr) * (xr * xr))))) + (( - 8 *
    ((abs(Xi) * (xi * (xr * ((xr * xr) * abs((Xr + xr)))))) + (abs((Xr + xr)) * (xi
    * (xr * ((xr * xr) * diffabs(Xi,xi))))))) + ((8 * ((abs(Xi) * ((Xr * (xi * ((xi
    * xi) * abs((Xr + xr))))) + (xi * ((xi * xi) * (xr * abs((Xr + xr))))))) +
5160 (abs((Xr + xr)) * ((Xr * (xi * ((xi * xi) * diffabs(Xi,xi)))) + (xi * ((xi *
    xi) * (xr * diffabs(Xi,xi))))))))) + (( - 8 * ((Xi * (Xr * (((Xr * Xr) * (xi *
    xr)) + (xi * (xr * (xr * xr)))))) + ((abs(Xi) * ((Xi * ((Xr * ((Xr * Xr) *
    diffabs(Xr,xr))) + (xr * ((xr * xr) * abs((Xr + xr)))))) + (Xr * ((Xr * Xr) *
    (xi * abs((Xr + xr))))))) + (abs((Xr + xr)) * ((Xi * ((Xr * ((Xr * Xr) *
5165 diffabs(Xi,xi))) + (xr * ((xr * xr) * diffabs(Xi,xi))))) + (Xr * ((Xr * Xr) *
    (xi * diffabs(Xi,xi))))))))) + ((8 * ((abs(Xi) * (Xi * ((Xi * Xi) * ((Xr *
    diffabs(Xr,xr)) + (xr * abs((Xr + xr))))))) + (abs((Xr + xr)) * (Xi * ((Xi *
    Xi) * (diffabs(Xi,xi) * (Xr + xr))))))) + (( - 12 * ((abs(Xr) * (Xr * ((xi *
    xi) * (xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xr * ((xi * xi) * (xr *
5170 diffabs(Xr,xr))))))) + ((12 * ((abs(Xi) * (Xr * ((xi * xi) * (xr * abs((Xi +
    xi)))))) + (abs((Xi + xi)) * (Xr * ((xi * xi) * (xr * diffabs(Xi,xi))))))) + ((
    - 12 * ((abs(Xr) * (Xi * (xi * ((xr * xr) * abs((Xr + xr)))))) + (abs((Xr +
    xr)) * (Xi * (xi * ((xr * xr) * diffabs(Xr,xr))))))) + ((12 * ((abs(Xi) * (Xi *
    (xi * ((xr * xr) * abs((Xi + xi)))))) + (abs((Xi + xi)) * (Xi * (xi * ((xr *
5175 xr) * diffabs(Xi,xi))))))) + (( - 12 * ((abs(Xr) * (Xi * ((Xr * Xr) * (xi *
```

```
abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * ((Xr * Xr) * (xi *
diffabs(Xr,xr)))))))) + ((12 * ((abs(Xi) * (Xi * ((Xr * Xr) * (xi * abs((Xi +
xi)))))) + (abs((Xi + xi)) * (Xi * ((Xr * Xr) * (xi * diffabs(Xi,xi))))))) + ((
- 12 * ((Xi * ((Xr * Xr) * (xi * (xr * xr)))) + ((abs(Xr) * ((Xi * Xi) * (Xr *
(xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * ((Xi * Xi) * (Xr * (xr *
diffabs(Xr,xr)))))))) + ((12 * ((abs(Xi) * ((Xi * Xi) * (Xr * (xr * abs((Xi +
xi)))))) + (abs((Xi + xi)) * ((Xi * Xi) * (Xr * (xr * diffabs(Xi,xi))))))) + ((
- 15 * ((((Xi * Xi) * (Xi * Xi)) * (xi * xi)) + ((Xi * Xi) * ((xi * xi) * (xi *
xi)))))) + (( - 20 * (Xi * ((Xi * Xi) * (xi * (xi * xi))))) + (( - 24 *
((abs(Xi) * (((Xr * Xr) * (xi * (xr * abs((Xr + xr))))) + (Xr * (xi * ((xr *
xr) * abs((Xr + xr))))))) + (abs((Xr + xr)) * (((Xr * Xr) * (xi * (xr *
diffabs(Xi,xi)))) + (Xr * (xi * ((xr * xr) * diffabs(Xi,xi))))))))) + ((24 *
((abs(Xi) * (Xi * ((xi * xi) * (xr * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi
* ((xi * xi) * (xr * diffabs(Xi,xi))))))) + (( - 24 * ((abs(Xi) * (Xi * (Xr *
((xr * xr) * abs((Xr + xr)))))) + ((abs(Xr) * (Xi * (Xr * (xi * (xr * abs((Xr +
xr))))))) + (abs((Xr + xr)) * (Xi * (Xr * ((xi * (xr * diffabs(Xr,xr))) + ((xr
* xr) * diffabs(Xi,xi))))))))) + ((24 * ((abs(Xi) * (Xi * (Xr * (((xi * xi) *
abs((Xr + xr))) + (xi * (xr * abs((Xi + xi))))))))) + ((abs((Xi + xi)) * (Xi *
(Xr * (xi * (xr * diffabs(Xi,xi)))))) + (abs((Xr + xr)) * (Xi * (Xr * ((xi *
xi) * diffabs(Xi,xi))))))))) + (( - 24 * ((abs(Xi) * (Xi * ((Xr * Xr) * (xr *
abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * ((Xr * Xr) * (xr *
diffabs(Xi,xi))))))) + ((24 * ((abs(Xi) * ((Xi * Xi) * ((Xr * (xi * abs((Xr +
xr)))) + (xi * (xr * abs((Xr + xr))))))) + (abs((Xr + xr)) * ((Xi * Xi) * ((Xr
* (xi * diffabs(Xi,xi))) + (xi * (xr * diffabs(Xi,xi))))))) + (((xr * xr) *
(xr * xr)) * abs((Xr * (Xr + xr))))))))))))))))))))))))))))))))))))))))))))));
        xin = (ci + ((2 * ((abs((Xi * (Xr + xr))) * (((xi * xi) * (xi * xi)) +
((xr * xr) * (xr * xr)))) + ((abs(Xi) * (diffabs(Xr,xr) * (((Xi * Xi) * (Xi *
Xi)) + ((Xr * Xr) * (Xr * Xr))))) + (abs((Xr + xr)) * (diffabs(Xi,xi) * (((Xi *
Xi) * (Xi * Xi)) + (((Xr * Xr) * (Xr * Xr)) + ((xi * xi) * (xi * xi))))))))) +
((4 * ((abs(Xr) * (xi * (xr * ((xr * xr) * abs((Xr + xr)))))) + (abs((Xr + xr))
* (xi * (xr * ((xr * xr) * diffabs(Xr,xr))))))) + (( - 4 * ((abs(Xi) * (xi *
(xr * ((xr * xr) * abs((Xi + xi)))))) + ((abs(Xr) * (xi * ((xi * xi) * (xr *
abs((Xr + xr)))))) + ((abs((Xi + xi)) * (xi * (xr * ((xr * xr) *
diffabs(Xi,xi))))) + (abs((Xr + xr)) * (xi * ((xi * xi) * (xr *
diffabs(Xr,xr))))))))) + ((4 * ((abs(Xi) * (xi * ((xi * xi) * (xr * abs((Xi +
xi)))))) + (abs((Xi + xi)) * (xi * ((xi * xi) * (xr * diffabs(Xi,xi))))))) + ((
- 4 * ((abs(Xr) * (Xr * (xi * ((xi * xi) * abs((Xr + xr)))))) + (abs((Xr + xr))
* (Xr * (xi * ((xi * xi) * diffabs(Xr,xr))))))) + ((4 * ((abs(Xi) * (Xr * (xi *
((xi * xi) * abs((Xi + xi)))))) + ((abs(Xr) * (Xr * ((Xr * Xr) * (xi * abs((Xr
+ xr)))))) + ((abs((Xi + xi)) * (Xr * (xi * ((xi * xi) * diffabs(Xi,xi))))) +
(abs((Xr + xr)) * (Xr * ((Xr * Xr) * (xi * diffabs(Xr,xr))))))))) + (( - 4 *
((abs(Xi) * (Xr * ((Xr * Xr) * (xi * abs((Xi + xi)))))) + (abs((Xi + xi)) * (Xr
* ((Xr * Xr) * (xi * diffabs(Xi,xi))))))) + ((4 * ((abs(Xr) * (Xi * (xr * ((xr
* xr) * abs((Xr + xr)))))) + (abs((Xr + xr)) * (Xi * (xr * ((xr * xr) *
diffabs(Xr,xr))))))) + (( - 4 * ((abs(Xi) * (Xi * (xr * ((xr * xr) * abs((Xi +
xi)))))) + (abs((Xi + xi)) * (Xi * (xr * ((xr * xr) * diffabs(Xi,xi))))))) +
((4 * ((Xi * (Xr * ((Xr * Xr) * diffabs(Xr,xr)))) * (abs(Xr) + abs((Xr +
xr))))) + (( - 4 * ((abs(Xi) * (Xi * (Xr * ((Xr * Xr) * diffabs(Xi,xi))))) +
((abs(Xr) * (Xi * ((Xi * Xi) * (xr * abs((Xr + xr)))))) + ((abs((Xi + xi)) *
(Xi * (Xr * ((Xr * Xr) * diffabs(Xi,xi))))) + ((4 * (abs((Xi + xi)) * (Xi * ((Xi * Xi) *
(xr * diffabs(Xi,xi)))))) + (( - 4 * ((Xi * ((Xi * Xi) * (Xr *
diffabs(Xr,xr)))) * (abs(Xr) + abs((Xr + xr))))) + ((4 * ((abs(Xi) * (Xi * ((Xi
* Xi) * ((Xr * diffabs(Xi,xi)) + (xr * abs((Xi + xi))))))) + (abs((Xi + xi)) *
(Xi * ((Xi * Xi) * (Xr * diffabs(Xi,xi))))))) + ((8 * ((abs(Xi) * ((Xi * (((Xi
* Xi) * (xi * abs((Xr + xr)))) + (xi * ((xi * xi) * abs((Xr + xr)))))) + (Xr *
(((Xr * Xr) * (xr * abs((Xr + xr)))) + (xr * ((xr * xr) * abs((Xr + xr))))))))
```

```
+ (abs((Xr + xr)) * ((Xi * (((Xi * Xi) * (xi * diffabs(Xi,xi))) + (xi * ((xi *
xi) * diffabs(Xi,xi))))) + (Xr * (((Xr * Xr) * (xr * diffabs(Xi,xi))) + (xr *
((xr * xr) * diffabs(Xi,xi)))))))))) + (( - 12 * ((abs(Xi) * ((xi * xi) * ((xr *
xr) * abs((Xr + xr))))) + (abs((Xr + xr)) * ((xi * xi) * ((xr * xr) *
diffabs(Xi,xi)))))) + ((12 * ((abs(Xr) * (Xr * (xi * ((xr * xr) * abs((Xr +
xr)))))) + (abs((Xr + xr)) * (Xr * (xi * ((xr * xr) * diffabs(Xr,xr))))))) + ((
- 12 * ((abs(Xi) * (Xr * (xi * ((xr * xr) * abs((Xi + xi)))))) + (abs((Xi +
xi)) * (Xr * (xi * ((xr * xr) * diffabs(Xi,xi))))))) + ((12 * ((abs(Xi) * ((Xr
* Xr) * ((xr * xr) * abs((Xr + xr))))) + ((abs(Xr) * ((Xr * Xr) * (xi * (xr *
abs((Xr + xr)))))) + (abs((Xr + xr)) * ((Xr * Xr) * ((xi * (xr *
diffabs(Xr,xr))) + ((xr * xr) * diffabs(Xi,xi)))))))) + (( - 12 * ((abs(Xi) *
((Xr * Xr) * (((xi * xi) * abs((Xr + xr))) + (xi * (xr * abs((Xi + xi))))))) +
((abs(Xr) * (Xi * ((xi * xi) * (xr * abs((Xr + xr)))))) + ((abs((Xi + xi)) *
((Xr * Xr) * (xi * (xr * diffabs(Xi,xi))))) + (abs((Xr + xr)) * ((Xi * ((xi *
xi) * (xr * diffabs(Xr,xr)))) + ((Xr * Xr) * ((xi * xi) *
diffabs(Xi,xi))))))))) + ((12 * ((abs(Xi) * (Xi * ((xi * xi) * (xr * abs((Xi +
xi)))))) + ((abs(Xr) * (Xi * (Xr * ((xr * xr) * abs((Xr + xr)))))) + ((abs((Xi
+ xi)) * (Xi * ((xi * xi) * (xr * diffabs(Xi,xi))))) + (abs((Xr + xr)) * (Xi *
(Xr * ((xr * xr) * diffabs(Xr,xr))))))))) + (( - 12 * ((abs(Xi) * (Xi * (Xr *
((xr * xr) * abs((Xi + xi)))))) + ((abs(Xr) * (Xi * (Xr * ((xi * xi) * abs((Xr
+ xr)))))) + ((abs((Xi + xi)) * (Xi * (Xr * ((xr * xr) * diffabs(Xi,xi))))) +
(abs((Xr + xr)) * (Xi * (Xr * ((xi * xi) * diffabs(Xr,xr))))))))) + ((12 *
((abs(Xi) * (Xi * (Xr * ((xi * xi) * abs((Xi + xi)))))) + ((abs(Xr) * (Xi *
((Xr * Xr) * (xr * abs((Xr + xr)))))) + ((abs((Xi + xi)) * (Xi * (Xr * ((xi *
xi) * diffabs(Xi,xi))))) + (abs((Xr + xr)) * (Xi * ((Xr * Xr) * (xr *
diffabs(Xr,xr))))))))) + (( - 12 * ((abs(Xi) * (((Xi * Xi) * ((xr * xr) *
abs((Xr + xr)))) + (Xi * ((Xr * Xr) * (xr * abs((Xi + xi))))))) + ((abs(Xr) *
((Xi * Xi) * (xi * (xr * abs((Xr + xr)))))) + ((abs((Xi + xi)) * (Xi * ((Xr *
Xr) * (xr * diffabs(Xi,xi))))) + (abs((Xr + xr)) * ((Xi * Xi) * ((xi * (xr *
diffabs(Xr,xr))) + ((xr * xr) * diffabs(Xi,xi))))))))) + ((12 * ((abs(Xi) *
((Xi * Xi) * (((xi * xi) * abs((Xr + xr))) + (xi * (xr * abs((Xi + xi))))))) +
((abs((Xi + xi)) * ((Xi * Xi) * (xi * (xr * diffabs(Xi,xi))))) + (abs((Xr +
xr)) * ((Xi * Xi) * ((xi * xi) * diffabs(Xi,xi))))))) + (( - 12 * ((abs(Xr) *
((Xi * Xi) * (Xr * (xi * abs((Xr + xr)))))) + (abs((Xr + xr)) * ((Xi * Xi) *
(Xr * (xi * diffabs(Xr,xr))))))) + ((12 * ((abs(Xi) * ((Xi * Xi) * (Xr * (xi *
abs((Xi + xi)))))) + (abs((Xi + xi)) * ((Xi * Xi) * (Xr * (xi *
diffabs(Xi,xi))))))) + (( - 12 * ((abs(Xi) * ((Xi * Xi) * ((Xr * Xr) *
diffabs(Xr,xr)))) + (abs((Xr + xr)) * ((Xi * Xi) * ((Xr * Xr) *
diffabs(Xi,xi)))))) + (( - 24 * ((abs(Xi) * (((Xi * Xi) * (Xr * (xr * abs((Xr +
xr))))) + ((Xi * (((Xr * Xr) * (xi * abs((Xr + xr)))) + (xi * ((xr * xr) *
abs((Xr + xr)))))) + (Xr * ((xi * xi) * (xr * abs((Xr + xr))))))))) + (abs((Xr +
xr)) * (((Xi * Xi) * (Xr * (xr * diffabs(Xi,xi)))) + ((Xi * (((Xr * Xr) * (xi *
diffabs(Xi,xi)))) + (xi * ((xr * xr) * diffabs(Xi,xi)))))) + (Xr * ((xi * xi) *
(xr * diffabs(Xi,xi))))))))) + (( - 48 * ((abs(Xi) * (Xi * (Xr * (xi * (xr *
abs((Xr + xr))))))) + (abs((Xr + xr)) * (Xi * (Xr * (xi * (xr *
diffabs(Xi,xi))))))))) + (2 * (abs((Xr + xr)) * (((xr * xr) * (xr * xr)) *
diffabs(Xi,xi))))))))))))))))))))))))))))))))));
      </perturbation>
      <derivative t='M'>
        dxan = (( - 2 * (sgn(Xxi) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dya *
abs(Xxi))))) + ((2 * (sgn(Xxr) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxa *
abs(Xxr))))) + (( - 2 * (sgn(Xxi) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dya *
abs(Xxi))))) + ((2 * (sgn(Xxr) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dxa *
abs(Xxr))))) + ((4 * (abs(Xxr) * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) +
(( - 4 * ((Xxi * Xxi) * (Xxr * ((Xxr * Xxr) * dxa)))) + ((4 * (abs(Xxr) * (Xxi
* ((Xxi * Xxi) * (dya * abs(Xxr)))))) + (( - 4 * (Xxi * (((Xxi * Xxi) * (Xxi *
Xxi)) * dya))) + (( - 8 * ((abs(Xxi) * (Xxr * ((Xxr * Xxr) * (dya *
```

```
5290    abs(Xxr))))) + ((sgn(Xxi) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))) +
        (sgn(Xxr) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxi))))))))) + ((8 *
        ((abs(Xxi) * (Xxi * ((Xxi * Xxi) * (dxa * abs(Xxr))))) + ((sgn(Xxi) * (Xxi *
        ((Xxi * Xxi) * (Xxr * (dya * abs(Xxr)))))) + (sgn(Xxr) * (Xxi * ((Xxi * Xxi) *
        (Xxr * (dxa * abs(Xxi))))))))) + (( - 12 * (abs(Xxr) * (Xxi * ((Xxr * Xxr) *
5295    (dya * abs(Xxr)))))) + ((12 * (abs(Xxi) * (Xxi * ((Xxr * Xxr) * (dya *
        abs(Xxi)))))) + (( - 12 * (abs(Xxr) * ((Xxi * Xxi) * (Xxr * (dxa *
        abs(Xxr)))))) + ((12 * ((abs(Xxi) * ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxi))))) +
        (sgn(Xxi) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi)))))) + (( - 12 *
        (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + (( - 24 *
5300    (abs(Xxi) * (Xxi * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + ((24 * (abs(Xxi) *
        ((Xxi * Xxi) * (Xxr * (dya * abs(Xxr)))))) + daa))))))))))))))));
                dxbn = (( - 2 * (sgn(Xxi) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dyb *
        abs(Xxi))))) + ((2 * (sgn(Xxr) * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxb *
        abs(Xxr))))) + (( - 2 * (sgn(Xxi) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dyb *
5305    abs(Xxi))))) + ((2 * (sgn(Xxr) * (((Xxi * Xxi) * (Xxi * Xxi)) * (dxb *
        abs(Xxr))))) + ((4 * (abs(Xxr) * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) +
        (( - 4 * ((Xxi * Xxi) * (Xxr * ((Xxr * Xxr) * dxb)))) + ((4 * (abs(Xxr) * (Xxi
        * ((Xxi * Xxi) * (dyb * abs(Xxr)))))) + (( - 4 * (Xxi * (((Xxi * Xxi) * (Xxi *
        Xxi)) * dyb))) + (( - 8 * ((abs(Xxi) * (Xxr * ((Xxr * Xxr) * (dyb *
5310    abs(Xxr))))) + ((sgn(Xxi) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) +
        (sgn(Xxr) * (Xxi * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxi))))))))) + ((8 *
        ((abs(Xxi) * (Xxi * ((Xxi * Xxi) * (dxb * abs(Xxr))))) + ((sgn(Xxi) * (Xxi *
        ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxr)))))) + (sgn(Xxr) * (Xxi * ((Xxi * Xxi) *
        (Xxr * (dxb * abs(Xxi))))))))) + (( - 12 * (abs(Xxr) * (Xxi * ((Xxr * Xxr) *
5315    (dyb * abs(Xxr)))))) + ((12 * (abs(Xxi) * (Xxi * ((Xxr * Xxr) * (dyb *
        abs(Xxi)))))) + (( - 12 * (abs(Xxr) * ((Xxi * Xxi) * (Xxr * (dxb *
        abs(Xxr)))))) + ((12 * ((abs(Xxi) * ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxi))))) +
        (sgn(Xxi) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi)))))) + (( - 12 *
        (sgn(Xxr) * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + (( - 24 *
5320    (abs(Xxi) * (Xxi * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + ((24 * (abs(Xxi) *
        ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxr)))))) + dab))))))))))))))));
                dyan = ((2 * ((sgn(Xxi) * ((dya * abs(Xxr)) * (((Xxi * Xxi) * (Xxi *
        Xxi)) + ((Xxr * Xxr) * (Xxr * Xxr))))) + (sgn(Xxr) * ((dxa * abs(Xxi)) * (((Xxi
        * Xxi) * (Xxi * Xxi)) + ((Xxr * Xxr) * (Xxr * Xxr))))))) + ((4 * (abs(Xxr) *
5325    (Xxr * ((Xxr * Xxr) * (dya * abs(Xxr)))))) + (( - 4 * ((abs(Xxi) * (Xxr * ((Xxr
        * Xxr) * (dya * abs(Xxi))))) + (abs(Xxr) * (Xxi * ((Xxi * Xxi) * (dxa *
        abs(Xxr))))))) + ((4 * (abs(Xxi) * (Xxi * ((Xxi * Xxi) * (dxa * abs(Xxi)))))) +
        ((8 * (abs(Xxi) * (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr)))))) + (( - 8 *
        (sgn(Xxi) * (Xxi * (Xxr * ((Xxr * Xxr) * (dya * abs(Xxi))))))) + ((8 *
5330    ((abs(Xxi) * (Xxi * ((Xxi * Xxi) * (dya * abs(Xxr))))) + ((sgn(Xxi) * (Xxi *
        ((Xxi * Xxi) * (Xxr * (dya * abs(Xxi)))))) + (sgn(Xxr) * (Xxi * (Xxr * ((Xxr *
        Xxr) * (dxa * abs(Xxr))))))))) + (( - 8 * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) *
        (Xxr * (dxa * abs(Xxr))))))) + ((12 * (abs(Xxr) * (Xxi * ((Xxr * Xxr) * (dxa *
        abs(Xxr)))))) + (( - 12 * ((abs(Xxi) * (Xxi * ((Xxr * Xxr) * (dxa *
5335    abs(Xxi))))) + (abs(Xxr) * ((Xxi * Xxi) * (Xxr * (dya * abs(Xxr))))))) + ((12 *
        (abs(Xxi) * ((Xxi * Xxi) * (Xxr * (dya * abs(Xxi)))))) + (( - 12 * ((sgn(Xxi) *
        ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxr))))) + (sgn(Xxr) * ((Xxi * Xxi) *
        ((Xxr * Xxr) * (dxa * abs(Xxi))))))) + (( - 24 * (abs(Xxi) * (((Xxi * Xxi) *
        (Xxr * (dxa * abs(Xxr)))) + (Xxi * ((Xxr * Xxr) * (dya * abs(Xxr))))))) +
5340    dba))))))))))));
                dybn = ((2 * ((sgn(Xxi) * ((dyb * abs(Xxr)) * (((Xxi * Xxi) * (Xxi *
        Xxi)) + ((Xxr * Xxr) * (Xxr * Xxr))))) + (sgn(Xxr) * ((dxb * abs(Xxi)) * (((Xxi
        * Xxi) * (Xxi * Xxi)) + ((Xxr * Xxr) * (Xxr * Xxr))))))) + ((4 * (abs(Xxr) *
        (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxr)))))) + (( - 4 * ((abs(Xxi) * (Xxr * ((Xxr
5345    * Xxr) * (dyb * abs(Xxi))))) + (abs(Xxr) * (Xxi * ((Xxi * Xxi) * (dxb *
        abs(Xxr))))))) + ((4 * (abs(Xxi) * (Xxi * ((Xxi * Xxi) * (dxb * abs(Xxi)))))) +
```

```
((8 * (abs(Xxi) * (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr)))))) + (( - 8 *
(sgn(Xxi) * (Xxi * (Xxr * ((Xxr * Xxr) * (dyb * abs(Xxi))))))) + ((8 *
((abs(Xxi) * (Xxi * ((Xxi * Xxi) * (dyb * abs(Xxr))))) + ((sgn(Xxi) * (Xxi *
((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi)))))) + (sgn(Xxr) * (Xxi * (Xxr * ((Xxr *
Xxr) * (dxb * abs(Xxr)))))))) + (( - 8 * (sgn(Xxr) * (Xxi * ((Xxi * Xxi) *
(Xxr * (dxb * abs(Xxr))))))) + ((12 * (abs(Xxr) * (Xxi * ((Xxr * Xxr) * (dxb *
abs(Xxr)))))) + (( - 12 * ((abs(Xxi) * (Xxi * ((Xxr * Xxr) * (dxb *
abs(Xxi))))) + (abs(Xxr) * ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxr))))))) + ((12 *
(abs(Xxi) * ((Xxi * Xxi) * (Xxr * (dyb * abs(Xxi)))))) + (( - 12 * ((sgn(Xxi) *
((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxr))))) + (sgn(Xxr) * ((Xxi * Xxi) *
((Xxr * Xxr) * (dxb * abs(Xxi))))))) + (( - 24 * (abs(Xxi) * (((Xxi * Xxi) *
(Xxr * (dxb * abs(Xxr)))) + (Xxi * ((Xxr * Xxr) * (dyb * abs(Xxr))))))) +
dbb))))))))))));
      </derivative>
    </formula>
  </group>

  <group type="68" name="Mothbrot 6th 5x1">
    <formula power="6" glitch="1.0000000000000002e-4">
      <reference t='R'>
        Xrn = (Cr + (( - 1 * (Xi * (((Xi * Xi) * (Xi * Xi)) * abs(Xi)))) + (( -
5 * (Xi * (((Xr * Xr) * (Xr * Xr)) * abs(Xi)))) + ((5 * (((Xi * Xi) * (Xi *
Xi)) * (Xr * abs(Xr)))) + ((10 * (Xi * ((Xi * Xi) * ((Xr * Xr) * abs(Xi))))) +
(( - 10 * ((Xi * Xi) * (Xr * ((Xr * Xr) * abs(Xr))))) + (Xr * (((Xr * Xr) * (Xr
* Xr)) * abs(Xr))))))))));
        Xin = (Ci + ((Xi * (((Xi * Xi) * (Xi * Xi)) * abs(Xr))) + ((5 * ((((Xi
* Xi) * (Xi * Xi)) * (Xr * abs(Xi))) + (Xi * (((Xr * Xr) * (Xr * Xr)) *
abs(Xr))))) + (( - 10 * ((Xi * ((Xi * Xi) * ((Xr * Xr) * abs(Xr)))) + ((Xi *
Xi) * (Xr * ((Xr * Xr) * abs(Xi)))))) + (Xr * (((Xr * Xr) * (Xr * Xr)) *
abs(Xi)))))));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((Xr * (((Xr * Xr) * (Xr * Xr)) * diffabs(Xr,xr))) + (( - 1
* ((Xi * (((Xi * Xi) * (Xi * Xi)) * diffabs(Xi,xi))) + (xi * (((xi * xi) * (xi
* xi)) * abs((Xi + xi)))))) + (( - 5 * (xi * (((xr * xr) * (xr * xr)) * abs((Xi
+ xi))))) + ((5 * ((((Xr * Xr) * (Xr * Xr)) * (xr * abs((Xr + xr)))) + ((Xr *
(abs((Xr + xr)) * (((xi * xi) * (xi * xi)) + ((xr * xr) * (xr * xr))))) + (((xi
* xi) * (xi * xi)) * (xr * abs((Xr + xr)))))))) + (( - 5 * ((Xi * ((((Xr * Xr) *
(Xr * Xr)) * diffabs(Xi,xi)) + (abs((Xi + xi)) * (((xi * xi) * (xi * xi)) +
((xr * xr) * (xr * xr)))))) + (((Xr * Xr) * (Xr * Xr)) * (xi * abs((Xi +
xi)))))) + ((5 * (((Xi * Xi) * (Xi * Xi)) * ((Xr * diffabs(Xr,xr)) + (xr *
abs((Xr + xr)))))) + (( - 5 * (((Xi * Xi) * (Xi * Xi)) * (xi * abs((Xi +
xi))))) + (( - 10 * ((xi * xi) * (xr * ((xr * xr) * abs((Xr + xr)))))) + ((10 *
((Xr * ((Xr * Xr) * ((xr * xr) * abs((Xr + xr))))) + (((Xr * Xr) * ((xi * ((xi
* xi) * abs((Xi + xi)))) + (xr * ((xr * xr) * abs((Xr + xr)))))) + (xi * ((xi *
xi) * ((xr * xr) * abs((Xi + xi))))))))) + (( - 10 * (((Xi * Xi) * ((Xr * ((Xr *
Xr) * diffabs(Xr,xr))) + ((xi * ((xi * xi) * abs((Xi + xi)))) + (xr * ((xr *
xr) * abs((Xr + xr))))))) + (Xr * ((Xr * Xr) * ((xi * xi) * abs((Xr + xr)))))))
+ ((10 * (Xi * ((Xi * Xi) * (((Xr * Xr) * diffabs(Xi,xi)) + ((xr * xr) *
abs((Xi + xi))))))) + (( - 10 * (Xi * ((Xi * Xi) * ((xi * xi) * abs((Xi +
xi)))))) + (( - 20 * (Xr * (xi * (xr * ((xr * xr) * abs((Xi + xi))))))) + ((20
* (Xr * (xi * ((xi * xi) * (xr * abs((Xi + xi))))))) + (( - 20 * ((Xi * (xi *
(xr * ((xr * xr) * abs((Xr + xr)))))) + (Xr * ((Xr * Xr) * (xi * (xr * abs((Xi
+ xi))))))))) + ((20 * (Xi * (xi * ((xi * xi) * (xr * abs((Xr + xr))))))) + (( -
20 * (Xi * (Xr * (xr * ((xr * xr) * abs((Xi + xi))))))) + ((20 * (Xi * (Xr *
(xi * ((xi * xi) * abs((Xr + xr))))))) + (( - 20 * (Xi * (Xr * ((Xr * Xr) *
((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))))) + ((20 * (Xi * ((Xi * Xi) *
```

```
((Xr * ((xi * abs((Xr + xr))) + (xr * abs((Xi + xi))))) + (xi * (xr * abs((Xr +
5405 xr)))))))) + (( - 30 * (((Xr * Xr) * (((xi * xi) * (xr * abs((Xr + xr)))) + (xi
* ((xr * xr) * abs((Xi + xi)))))) + (Xr * ((xi * xi) * ((xr * xr) * abs((Xr +
xr))))))) + ((30 * (Xi * ((xi * xi) * ((xr * xr) * abs((Xi + xi)))))) + (( - 30
* (Xi * ((Xr * Xr) * ((xr * xr) * abs((Xi + xi)))))) + ((30 * (((Xi * Xi) *
(((xi * xi) * (xr * abs((Xr + xr)))) + (xi * ((xr * xr) * abs((Xi + xi)))))) +
5410 (Xi * ((Xr * Xr) * ((xi * xi) * abs((Xi + xi))))))) + (( - 30 * ((Xi * Xi) *
(Xr * ((xr * xr) * abs((Xr + xr)))))) + ((30 * ((Xi * Xi) * (Xr * ((xi * xi) *
abs((Xr + xr)))))) + (( - 30 * ((Xi * Xi) * ((Xr * Xr) * (xr * abs((Xr +
xr)))))) + ((30 * ((Xi * Xi) * ((Xr * Xr) * (xi * abs((Xi + xi)))))) + (( - 60
* (Xi * (Xr * (xi * ((xr * xr) * abs((Xr + xr))))))) + ((60 * (Xi * (Xr * ((xi
5415 * xi) * (xr * abs((Xi + xi))))))) + (( - 60 * (Xi * ((Xr * Xr) * (xi * (xr *
abs((Xr + xr))))))) + ((60 * ((Xi * Xi) * (Xr * (xi * (xr * abs((Xi + xi)))))))
+ (xr * (((xr * xr) * (xr * xr)) * abs((Xr +
xr))))))))))))))))))))))))))))))))));
        xin = (ci + ((Xi * (((Xi * Xi) * (Xi * Xi)) * diffabs(Xr,xr))) + ((Xr *
5420 (((Xr * Xr) * (Xr * Xr)) * diffabs(Xi,xi))) + ((xi * (((xi * xi) * (xi * xi)) *
abs((Xr + xr)))) + ((5 * ((((Xi * Xi) * (Xi * Xi)) * ((Xr * diffabs(Xi,xi)) +
((xi * abs((Xr + xr))) + (xr * abs((Xi + xi)))))) + ((Xi * ((((Xr * Xr) * (Xr *
Xr)) * diffabs(Xr,xr)) + (abs((Xr + xr)) * (((xi * xi) * (xi * xi)) + ((xr *
xr) * (xr * xr)))))) + ((((Xr * Xr) * (Xr * Xr)) * ((xi * abs((Xr + xr))) + (xr
5425 * abs((Xi + xi))))) + ((Xr * (abs((Xi + xi)) * (((xi * xi) * (xi * xi)) + ((xr
* xr) * (xr * xr))))) + ((((xi * xi) * (xi * xi)) * (xr * abs((Xi + xi)))) +
(xi * (((xr * xr) * (xr * xr)) * abs((Xr + xr))))))))))) + (( - 10 * ((xi * ((xi
* xi) * ((xr * xr) * abs((Xr + xr))))) + ((xi * xi) * (xr * ((xr * xr) *
abs((Xi + xi))))))) + ((10 * ((Xr * Xr) * (xr * ((xr * xr) * abs((Xi + xi)))))))
5430 + (( - 10 * ((Xr * Xr) * (xi * ((xi * xi) * abs((Xr + xr)))))) + ((10 * (Xr *
((Xr * Xr) * ((xr * xr) * abs((Xi + xi)))))) + (( - 10 * (((Xi * Xi) * (xr *
((xr * xr) * abs((Xi + xi))))) + (Xr * ((Xr * Xr) * ((xi * xi) * abs((Xi +
xi))))))) + ((10 * ((Xi * Xi) * (xi * ((xi * xi) * abs((Xr + xr)))))) + (( - 10
* ((Xi * ((Xi * Xi) * (((Xr * Xr) * diffabs(Xr,xr)) + ((xr * xr) * abs((Xr +
5435 xr)))))) + ((Xi * Xi) * (Xr * ((Xr * Xr) * diffabs(Xi,xi)))))) + ((10 * (Xi *
((Xi * Xi) * ((xi * xi) * abs((Xr + xr)))))) + ((20 * (Xr * (xi * (xr * ((xr *
xr) * abs((Xr + xr))))))) + (( - 20 * (Xr * (xi * ((xi * xi) * (xr * abs((Xr +
xr))))))) + ((20 * (Xr * ((Xr * Xr) * (xi * (xr * abs((Xr + xr))))))) + (( - 20
* (Xi * (xi * (xr * ((xr * xr) * abs((Xi + xi))))))) + ((20 * (Xi * ((Xr *
5440 (((Xr * Xr) * (xr * abs((Xr + xr)))) + ((xi * ((xi * xi) * abs((Xi + xi)))) +
(xr * ((xr * xr) * abs((Xr + xr))))))) + (xi * ((xi * xi) * (xr * abs((Xi +
xi)))))))) + (( - 20 * (Xi * (Xr * ((Xr * Xr) * (xi * abs((Xi + xi))))))) +
((20 * (Xi * ((Xi * Xi) * (xi * (xr * abs((Xi + xi))))))) + (( - 20 * (Xi *
((Xi * Xi) * (Xr * (xr * abs((Xr + xr))))))) + ((20 * (Xi * ((Xi * Xi) * (Xr *
5445 (xi * abs((Xi + xi))))))) + (( - 30 * (Xr * ((xi * xi) * ((xr * xr) * abs((Xi +
xi)))))) + ((30 * ((Xr * Xr) * (xi * ((xr * xr) * abs((Xr + xr)))))) + (( - 30
* ((Xi * ((xi * xi) * ((xr * xr) * abs((Xr + xr)))))) + ((Xr * Xr) * ((xi * xi)
* (xr * abs((Xi + xi))))))) + ((30 * (Xi * ((Xr * Xr) * ((xr * xr) * abs((Xr +
xr)))))) + (( - 30 * (((Xi * Xi) * (xi * ((xr * xr) * abs((Xr + xr)))))) + (Xi *
5450 ((Xr * Xr) * ((xi * xi) * abs((Xr + xr))))))) + ((30 * ((Xi * Xi) * ((xi * xi)
* (xr * abs((Xi + xi)))))) + (( - 30 * ((Xi * Xi) * (Xr * ((xr * xr) * abs((Xi
+ xi)))))) + ((30 * ((Xi * Xi) * (Xr * ((xi * xi) * abs((Xi + xi)))))) + (( -
30 * ((Xi * Xi) * ((Xr * Xr) * ((xi * abs((Xr + xr))) + (xr * abs((Xi +
xi))))))) + (( - 60 * (((Xi * Xi) * (Xr * (xi * (xr * abs((Xr + xr)))))) + (Xi
5455 * (((Xr * Xr) * (xi * (xr * abs((Xi + xi))))) + (Xr * (((xi * xi) * (xr *
abs((Xr + xr)))) + (xi * ((xr * xr) * abs((Xi + xi))))))))) + (xr * (((xr *
xr) * (xr * xr)) * abs((Xi + xi))))))))))))))))))))))))))))))))));
      </perturbation>
      <derivative t="M">
5460    dxan = ((Xxr * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxa * sgn(Xxr)))) + (( -
```

```
      1 * (Xxi * (((Xxi * Xxi) * (Xxi * Xxi)) * (dya * sgn(Xxi))))) + (( - 5 * (((Xxr
      * Xxr) * (Xxr * Xxr)) * (dya * abs(Xxi)))) + ((5 * (((Xxr * Xxr) * (Xxr * Xxr))
      * (dxa * abs(Xxr)))) + (( - 5 * ((((Xxi * Xxi) * (Xxi * Xxi)) * (dya *
      abs(Xxi))) + (Xxi * (((Xxr * Xxr) * (Xxr * Xxr)) * (dya * sgn(Xxi)))))) + ((5 *
5465  (((Xxi * Xxi) * (Xxi * Xxi)) * ((Xxr * (dxa * sgn(Xxr))) + (dxa * abs(Xxr))))
      + (( - 10 * ((Xxi * Xxi) * (Xxr * ((Xxr * Xxr) * (dxa * sgn(Xxr)))))) + ((10 *
      (Xxi * ((Xxi * Xxi) * ((Xxr * Xxr) * (dya * sgn(Xxi)))))) + (( - 20 * (Xxi *
      (Xxr * ((Xxr * Xxr) * ((dxa * abs(Xxi)) + (dya * abs(Xxr))))))) + ((20 * (Xxi *
      ((Xxi * Xxi) * (Xxr * ((dxa * abs(Xxi)) + (dya * abs(Xxr))))))) + ((30 * ((Xxi
5470  * Xxi) * ((Xxr * Xxr) * (dya * abs(Xxi))))) + (( - 30 * ((Xxi * Xxi) * ((Xxr *
      Xxr) * (dxa * abs(Xxr))))) + daa))))))))))));
            dxbn = ((Xxr * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxb * sgn(Xxr)))) + (( -
      1 * (Xxi * (((Xxi * Xxi) * (Xxi * Xxi)) * (dyb * sgn(Xxi))))) + (( - 5 * (((Xxr
      * Xxr) * (Xxr * Xxr)) * (dyb * abs(Xxi)))) + ((5 * (((Xxr * Xxr) * (Xxr * Xxr))
5475  * (dxb * abs(Xxr)))) + (( - 5 * ((((Xxi * Xxi) * (Xxi * Xxi)) * (dyb *
      abs(Xxi))) + (Xxi * (((Xxr * Xxr) * (Xxr * Xxr)) * (dyb * sgn(Xxi)))))) + ((5 *
      (((Xxi * Xxi) * (Xxi * Xxi)) * ((Xxr * (dxb * sgn(Xxr))) + (dxb * abs(Xxr))))
      + (( - 10 * ((Xxi * Xxi) * (Xxr * ((Xxr * Xxr) * (dxb * sgn(Xxr)))))) + ((10 *
      (Xxi * ((Xxi * Xxi) * ((Xxr * Xxr) * (dyb * sgn(Xxi)))))) + (( - 20 * (Xxi *
5480  (Xxr * ((Xxr * Xxr) * ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))))) + ((20 * (Xxi *
      ((Xxi * Xxi) * (Xxr * ((dxb * abs(Xxi)) + (dyb * abs(Xxr))))))) + ((30 * ((Xxi
      * Xxi) * ((Xxr * Xxr) * (dyb * abs(Xxi))))) + (( - 30 * ((Xxi * Xxi) * ((Xxr *
      Xxr) * (dxb * abs(Xxr))))) + dab))))))))))));
            dyan = ((Xxi * (((Xxi * Xxi) * (Xxi * Xxi)) * (dxa * sgn(Xxr)))) +
5485  ((Xxr * (((Xxr * Xxr) * (Xxr * Xxr)) * (dya * sgn(Xxi)))) + ((5 * ((((Xxi *
      Xxi) * (Xxi * Xxi)) * ((Xxr * (dya * sgn(Xxi))) + ((dxa * abs(Xxi)) + (dya *
      abs(Xxr))))) + ((Xxi * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxa * sgn(Xxr)))) +
      (((Xxr * Xxr) * (Xxr * Xxr)) * ((dxa * abs(Xxi)) + (dya * abs(Xxr)))))) + (( -
      10 * ((Xxi * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxa * sgn(Xxr))))) + ((Xxi * Xxi) *
5490  (Xxr * ((Xxr * Xxr) * (dya * sgn(Xxi))))))) + (( - 20 * (Xxi * (Xxr * ((Xxr *
      Xxr) * (dya * abs(Xxi)))))) + ((20 * (Xxi * (((Xxi * Xxi) * (Xxr * (dya *
      abs(Xxi)))) + (Xxr * ((Xxr * Xxr) * (dxa * abs(Xxr))))))) + (( - 20 * (Xxi *
      ((Xxi * Xxi) * (Xxr * (dxa * abs(Xxr)))))) + (( - 30 * ((Xxi * Xxi) * ((Xxr *
      Xxr) * ((dxa * abs(Xxi)) + (dya * abs(Xxr)))))) + dba))))))));
5495          dybn = ((Xxi * (((Xxi * Xxi) * (Xxi * Xxi)) * (dxb * sgn(Xxr)))) +
      ((Xxr * (((Xxr * Xxr) * (Xxr * Xxr)) * (dyb * sgn(Xxi)))) + ((5 * ((((Xxi *
      Xxi) * (Xxi * Xxi)) * ((Xxr * (dyb * sgn(Xxi))) + ((dxb * abs(Xxi)) + (dyb *
      abs(Xxr))))) + ((Xxi * (((Xxr * Xxr) * (Xxr * Xxr)) * (dxb * sgn(Xxr)))) +
      (((Xxr * Xxr) * (Xxr * Xxr)) * ((dxb * abs(Xxi)) + (dyb * abs(Xxr)))))) + (( -
5500  10 * ((Xxi * ((Xxi * Xxi) * ((Xxr * Xxr) * (dxb * sgn(Xxr))))) + ((Xxi * Xxi) *
      (Xxr * ((Xxr * Xxr) * (dyb * sgn(Xxi))))))) + (( - 20 * (Xxi * (Xxr * ((Xxr *
      Xxr) * (dyb * abs(Xxi)))))) + ((20 * (Xxi * (((Xxi * Xxi) * (Xxr * (dyb *
      abs(Xxi)))) + (Xxr * ((Xxr * Xxr) * (dxb * abs(Xxr))))))) + (( - 20 * (Xxi *
      ((Xxi * Xxi) * (Xxr * (dxb * abs(Xxr)))))) + (( - 30 * ((Xxi * Xxi) * ((Xxr *
5505  Xxr) * ((dxb * abs(Xxi)) + (dyb * abs(Xxr)))))) + dbb))))))));
        </derivative>
      </formula>
    </group>

5510  <group type="69" name="General Abs Quadratic Minus">
      <formula power="2" glitch="1.0000000000000004e-7">
        <reference t='R'>
          Xrn = (Cr + (( - 1 * (Xi * Xi)) + (Xr * Xr)));
          Xin = ((Ai * (Xr * Xr)) + ((2 * (Ar * abs((Xi * Xr)))) + Ci));
5515      </reference>
        <perturbation t='R'>
          xrn = (cr + (( - 1 * (xi * xi)) + ((2 * (Xr * xr)) + (( - 2 * (Xi *
```

```
xi)) + (xr * xr))));
        xin = ((Ai * (xr * xr)) + ((2 * ((Ai * (Xr * xr)) + (Ar * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + ci));
      </perturbation>
      <derivative t='M'>
        dxan = (( - 2 * (Xxi * dya)) + ((2 * (Xxr * dxa)) + daa));
        dxbn = (( - 2 * (Xxi * dyb)) + ((2 * (Xxr * dxb)) + dab));
        dyan = ((2 * ((g_FactorAI * (Xxr * dxa)) + (sgn(Xxi) * (g_FactorAR *
((Xxi * (dxa * sgn(Xxr))) + (Xxr * (dya * sgn(Xxr)))))))) + dba);
        dybn = ((2 * ((g_FactorAI * (Xxr * dxb)) + (sgn(Xxi) * (g_FactorAR *
((Xxi * (dxb * sgn(Xxr))) + (Xxr * (dyb * sgn(Xxr)))))))) + dbb);
      </derivative>
    </formula>
  </group>

  <group type="70" name="General Abs Quadratic Plus">
    <formula power="2" glitch="1.0000000000000004e-7">
      <reference t='R'>
        Xrn = (Cr + ((Xi2) + (Xr2)));
        Xin = ((Ai * (Xr2)) + ((2 * (Ar * abs((Xi * Xr)))) + Ci));
      </reference>
      <perturbation t='R'>
        xrn = (cr + ((xi * xi) + ((2 * ((Xi * xi) + (Xr * xr))) + (xr * xr))));
        xin = ((Ai * (xr * xr)) + ((2 * ((Ai * (Xr * xr)) + (Ar * diffabs((Xi *
Xr),((Xi * xr) + (xi * (Xr + xr)))))))) + ci));
      </perturbation>
      <derivative t='M'>
        dxan = ((2 * ((Xxi * dya) + (Xxr * dxa))) + daa);
        dxbn = ((2 * ((Xxi * dyb) + (Xxr * dxb))) + dab);
        dyan = ((2 * ((g_FactorAI * (Xxr * dxa)) + (sgn(Xxi) * (g_FactorAR *
((Xxi * (dxa * sgn(Xxr))) + (Xxr * (dya * sgn(Xxr)))))))) + dba);
        dybn = ((2 * ((g_FactorAI * (Xxr * dxb)) + (sgn(Xxi) * (g_FactorAR *
((Xxi * (dxb * sgn(Xxr))) + (Xxr * (dyb * sgn(Xxr)))))))) + dbb);
      </derivative>
    </formula>
  </group>

</formulas>
```

43 formula/formula.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
```

```
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" />
<xsl:template match="/">

#include "../fraktal_sft/CFixedFloat.h"
#include "../fraktal_sft/floatexp.h"
#include "../fraktal_sft/complex.h"

#define FORMULA2(function,type,power) function ## _ ## type ## _ ## power
#define FORMULA(a,b,c) FORMULA2(a,b,c)

#ifdef PASS1 // reference

<xsl:for-each select="formulas/group/formula">

template &lt;typename T&gt;
bool FORMULA(reference,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
  ( const int m_nFractalType, const int m_nPower
  , T *m_db_dxr, T *m_db_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
  , const double terminate, const double g_real, const double g_imag
  , const bool m_bGlitchLowTolerance
  , int64_t &amp;antal, double &amp;test1, double &amp;test2
  , double &amp;Xxr0, double &amp;Xxi0
  )
{
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower == <xsl:value-of select="@power" />)
  {
    bool stored = false;
    double old_absval = 0;
    double abs_val = 0;
    m_nGlitchIter = m_nMaxIter + 1;
    int64_t nMaxIter = m_nMaxIter;
    int64_t i;
    double glitch = <xsl:value-of select="@glitch" />;
    if (m_bGlitchLowTolerance) {
      glitch = sqrt(glitch);
    }
    mp_bitcnt_t bits = mpfr_get_prec(Cr0.m_f.backend().data());
    mpfr_t Cr; mpfr_init2(Cr, bits); mpfr_set(Cr, Cr0.m_f.backend().data(), MPFR_RNDN);
    mpfr_t Ci; mpfr_init2(Ci, bits); mpfr_set(Ci, Ci0.m_f.backend().data(), MPFR_RNDN);
    mpfr_t Xr; mpfr_init2(Xr, bits); mpfr_set_d(Xr, g_SeedR, MPFR_RNDN);
    mpfr_t Xi; mpfr_init2(Xi, bits); mpfr_set_d(Xi, g_SeedI, MPFR_RNDN);
    T Xrd = mpfr_get(Xr, T(0.0), MPFR_RNDN);
    T Xid = mpfr_get(Xi, T(0.0), MPFR_RNDN);
    mpfr_t Xr2; mpfr_init2(Xr2, bits); mpfr_sqr(Xr2, Xr, MPFR_RNDN);
```

```
    mpfr_t Xi2; mpfr_init2(Xi2, bits); mpfr_sqr(Xi2, Xi, MPFR_RNDN);
    mpfr_t Xrn; mpfr_init2(Xrn, bits);
    mpfr_t Xin; mpfr_init2(Xin, bits);
    mpfr_t Ar; mpfr_init2(Ar, bits); mpfr_set_d(Ar, g_FactorAR, MPFR_RNDN);
    mpfr_t Ai; mpfr_init2(Ai, bits); mpfr_set_d(Ai, g_FactorAI, MPFR_RNDN);
<xsl:choose>
<xsl:when test="reference/@t='C'">
    complex&lt;CFixedFloat&gt; C, A, X, Xn;
    mpfr_set(C.m_r.m_f.backend().data(), Cr, MPFR_RNDN);
    mpfr_set(C.m_i.m_f.backend().data(), Ci, MPFR_RNDN);
    mpfr_set(A.m_r.m_f.backend().data(), Ar, MPFR_RNDN);
    mpfr_set(A.m_i.m_f.backend().data(), Ai, MPFR_RNDN);
</xsl:when>
</xsl:choose>

#define LOOP \
      mpfr_set(Xr, Xrn, MPFR_RNDN); \
      mpfr_set(Xi, Xin, MPFR_RNDN); \
      mpfr_sqr(Xr2, Xr, MPFR_RNDN); \
      mpfr_sqr(Xi2, Xi, MPFR_RNDN); \
      m_nRDone++; \
      Xrd = mpfr_get(Xr, T(0.0), MPFR_RNDN); \
      Xid = mpfr_get(Xi, T(0.0), MPFR_RNDN); \
      old_absval = abs_val; \
      abs_val = double(g_real * Xrd * Xrd + g_imag * Xid * Xid); \
      const double Xz = abs_val * glitch; \
      m_db_dxr[i] = Xrd; \
      m_db_dxi[i] = Xid; \
      m_db_z[i] = Xz; \
      if (abs_val &gt;= 4) \
      { \
        if (terminate == 4 &amp;&amp; !stored) \
        { \
          stored = true; \
          antal = i; \
          test1 = abs_val; \
          test2 = old_absval; \
        } \
      } \
      if (abs_val &gt;= terminate){ \
        if (terminate &gt; 4 &amp;&amp; !stored) \
        { \
          stored = true; \
          antal = i; \
          test1 = abs_val; \
          test2 = old_absval; \
        } \
        if (nMaxIter == m_nMaxIter){ \
          nMaxIter = i + 3; \
          if (nMaxIter &gt; m_nMaxIter) \
            nMaxIter = m_nMaxIter; \
          m_nGlitchIter = nMaxIter; \
        } \
      }

<xsl:choose>
<xsl:when test="reference/@t='C'">
```

```
for (i = 0; i &lt; nMaxIter &amp;&amp; !m_bStop; i++)
      {
        mpfr_set(X.m_r.m_f.backend().data(), Xr, MPFR_RNDN);
        mpfr_set(X.m_i.m_f.backend().data(), Xi, MPFR_RNDN);
        {
          <xsl:value-of select="reference" />
        }
        mpfr_set(Xrn, Xn.m_r.m_f.backend().data(), MPFR_RNDN);
        mpfr_set(Xin, Xn.m_i.m_f.backend().data(), MPFR_RNDN);
LOOP  }
</xsl:when>
<xsl:when test="reference/@t='R'">
#define DLOOP
@rr   {
        <xsl:value-of select="reference" />
      }
#undef DLOOP
</xsl:when>
</xsl:choose>

#undef LOOP
    mpfr_clear(Cr);
    mpfr_clear(Ci);
    mpfr_clear(Xr);
    mpfr_clear(Xi);
    mpfr_clear(Xr2);
    mpfr_clear(Xi2);
    mpfr_clear(Xrn);
    mpfr_clear(Xin);
    mpfr_clear(Ar);
    mpfr_clear(Ai);
    Xxr0 = double(Xrd);
    Xxi0 = double(Xid);
    return true;
  }
  return false;
}

</xsl:for-each>

template &lt;typename T&gt;
bool reference
  ( const int m_nFractalType, const int m_nPower
  , T *m_dxr, T *m_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
    ↳ int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr, const CFixedFloat &amp;Ci
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
  , const double terminate, const double g_real, const double g_imag
  , const bool m_bGlitchLowTolerance
  , int64_t &amp;antal, double &amp;test1, double &amp;test2
  , double &amp;Xxr0, double &amp;Xxi0
  )
{
  switch (m_nFractalType)
  {
```

```
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
      <xsl:for-each select="formula">
        case <xsl:value-of select="@power" />:
          return FORMULA(reference,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
            ( m_nFractalType, m_nPower
            , m_dxr, m_dxi, m_db_z
            , m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter
            , Cr, Ci
            , g_SeedR, g_SeedI
            , g_FactorAR, g_FactorAI
            , terminate, g_real, g_imag
            , m_bGlitchLowTolerance
            , antal, test1, test2
            , Xxr0, Xxi0
            );
      </xsl:for-each>
      }
      break;
  </xsl:for-each>
  }
  return false;
}

template bool reference&lt;double&gt;
  ( const int m_nFractalType, const int m_nPower
  , double *m_db_dxr, double *m_db_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
  , const double terminate, const double g_real, const double g_imag
  , const bool m_bGlitchLowTolerance
  , int64_t &amp;antal, double &amp;test1, double &amp;test2
  , double &amp;Xxr0, double &amp;Xxi0
  );
template bool reference&lt;long double&gt;
  ( const int m_nFractalType, const int m_nPower
  , long double *m_db_dxr, long double *m_db_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
  , const double terminate, const double g_real, const double g_imag
  , const bool m_bGlitchLowTolerance
  , int64_t &amp;antal, double &amp;test1, double &amp;test2
  , double &amp;Xxr0, double &amp;Xxi0
  );
template bool reference&lt;floatexp&gt;
  ( const int m_nFractalType, const int m_nPower
  , floatexp *m_db_dxr, floatexp *m_db_dxi, double *m_db_z
```

```
    , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
      ↳ int64_t &amp;m_nMaxIter
    , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
    , const double g_SeedR, const double g_SeedI
    , const double g_FactorAR, const double g_FactorAI
    , const double terminate, const double g_real, const double g_imag
    , const bool m_bGlitchLowTolerance
    , int64_t &amp;antal, double &amp;test1, double &amp;test2
    , double &amp;Xxr0, double &amp;Xxi0
    );

#endif

#ifdef PASS2 // reference with derivatives

<xsl:for-each select="formulas/group/formula">

template &lt;typename S, typename T&gt;
bool FORMULA(reference,<xsl:value-of select="../@type" />,<xsl:value-of select="↙
    ↳ @power" />)
  ( const int m_nFractalType, const int m_nPower
  , T *m_db_dxr, T *m_db_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
      ↳ int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
  , const double terminate, const double g_real, const double g_imag
  , const bool m_bGlitchLowTolerance
  , int64_t &amp;antal, double &amp;test1, double &amp;test2
  , double &amp;Xxr0, double &amp;Xxi0
  , S &amp;dr0, S &amp;di0
  , const S &amp;daa, const S &amp;dab, const S &amp;dba, const S &amp;dbb
  )
{
  (void) dr0; // -Wunused-parameter
  (void) di0; // -Wunused-parameter
  (void) daa; // -Wunused-parameter
  (void) dab; // -Wunused-parameter
  (void) dba; // -Wunused-parameter
  (void) dbb; // -Wunused-parameter
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
      ↳ == <xsl:value-of select="@power" />)
  {
    bool stored = false;
    double old_absval = 0;
    double abs_val = 0;
    m_nGlitchIter = m_nMaxIter + 1;
    int64_t nMaxIter = m_nMaxIter;
    int64_t i;
    double glitch = <xsl:value-of select="@glitch" />;
    if (m_bGlitchLowTolerance) {
      glitch = sqrt(glitch);
    }
    mp_bitcnt_t bits = mpfr_get_prec(Cr0.m_f.backend().data());
    mpfr_t Cr; mpfr_init2(Cr, bits); mpfr_set(Cr, Cr0.m_f.backend().data(), ↙
        ↳ MPFR_RNDN);
```

```
        mpfr_t Ci; mpfr_init2(Ci, bits); mpfr_set(Ci, Ci0.m_f.backend().data(), ↙
            ↳ MPFR_RNDN);
        mpfr_t Xr; mpfr_init2(Xr, bits); mpfr_set_d(Xr, g_SeedR, MPFR_RNDN);
290     mpfr_t Xi; mpfr_init2(Xi, bits); mpfr_set_d(Xi, g_SeedI, MPFR_RNDN);
        T Xrd = mpfr_get(Xr, T(0.0), MPFR_RNDN);
        T Xid = mpfr_get(Xi, T(0.0), MPFR_RNDN);
        mpfr_t Xr2; mpfr_init2(Xr2, bits); mpfr_sqr(Xr2, Xr, MPFR_RNDN);
        mpfr_t Xi2; mpfr_init2(Xi2, bits); mpfr_sqr(Xi2, Xi, MPFR_RNDN);
295     mpfr_t Xrn; mpfr_init2(Xrn, bits);
        mpfr_t Xin; mpfr_init2(Xin, bits);
        mpfr_t Ar; mpfr_init2(Ar, bits); mpfr_set_d(Ar, g_FactorAR, MPFR_RNDN);
        mpfr_t Ai; mpfr_init2(Ai, bits); mpfr_set_d(Ai, g_FactorAI, MPFR_RNDN);
    <xsl:choose>
300 <xsl:when test="derivative/@t='R' or derivative/@t='C'">
        S dr = dr0, di = di0;
        S drn = 0, din = 0;
    </xsl:when>
    <xsl:when test="derivative/@t='M'">
305     S dxa = daa, dxb = dab, dya = dba, dyb = dbb;
        S dxan = 0, dxbn = 0, dyan = 0, dybn = 0;
    </xsl:when>
    </xsl:choose>
    <xsl:choose>
310 <xsl:when test="reference/@t='C'">
        complex&lt;CFixedFloat&gt; C, A, X, Xn;
        mpfr_set(C.m_r.m_f.backend().data(), Cr, MPFR_RNDN);
        mpfr_set(C.m_i.m_f.backend().data(), Ci, MPFR_RNDN);
        mpfr_set(A.m_r.m_f.backend().data(), Ar, MPFR_RNDN);
315     mpfr_set(A.m_i.m_f.backend().data(), Ai, MPFR_RNDN);
    </xsl:when>
    </xsl:choose>

    #define LOOP \
320 <xsl:choose><xsl:when test="derivative/@t='R' or derivative/@t='C'"> \
        dr = drn; di = din; \
    </xsl:when><xsl:when test="derivative/@t='M'"> \
        dxa = dxan; dxb = dxbn; dya = dyan, dyb = dybn; \
    </xsl:when></xsl:choose> \
325       mpfr_set(Xr, Xrn, MPFR_RNDN); \
          mpfr_set(Xi, Xin, MPFR_RNDN); \
          mpfr_sqr(Xr2, Xr, MPFR_RNDN); \
          mpfr_sqr(Xi2, Xi, MPFR_RNDN); \
          m_nRDone++; \
330       Xrd = mpfr_get(Xr, T(0.0), MPFR_RNDN); \
          Xid = mpfr_get(Xi, T(0.0), MPFR_RNDN); \
          old_absval = abs_val; \
          abs_val = double(g_real * Xrd * Xrd + g_imag * Xid * Xid); \
          const double Xz = abs_val * glitch; \
335       m_db_dxr[i] = Xrd; \
          m_db_dxi[i] = Xid; \
          m_db_z[i] = Xz; \
          if (abs_val &gt;= 4) \
          { \
340         if (terminate == 4 &amp;&amp; !stored) \
            { \
              stored = true; \
              antal = i; \
```

```
            test1 = abs_val; \
            test2 = old_absval; \
          } \
        } \
        if (abs_val &gt;= terminate){ \
          if (terminate &gt; 4 &amp;&amp; !stored) \
          { \
            stored = true; \
            antal = i; \
            test1 = abs_val; \
            test2 = old_absval; \
          } \
          if (nMaxIter == m_nMaxIter){ \
            nMaxIter = i + 3; \
            if (nMaxIter &gt; m_nMaxIter) \
              nMaxIter = m_nMaxIter; \
            m_nGlitchIter = nMaxIter; \
          } \
        }

    <xsl:choose>
    <xsl:when test="reference/@t='C'">
    @rd    {
            const T Xxr = Xrd; (void) Xxr;
            const T Xxi = Xid; (void) Xxi;
            const complex&lt;T&gt; Xx(Xxr, Xxi), A(g_FactorAR, g_FactorAI);
    <xsl:choose>
    <xsl:when test="derivative/@t='C'">
            const complex&lt;S&gt; d(dr, di), d0(daa, dba); <!-- FIXME matrix derivatives -->
            complex&lt;S&gt; dn(0.0, 0.0);
            <xsl:value-of select="derivative" />
            drn = dn.m_r; din = dn.m_i;
    </xsl:when>
    <xsl:when test="derivative/@t='R' or derivative/@t='M'">
           <xsl:value-of select="derivative" />
    </xsl:when>
    </xsl:choose>
        }
    for (i = 0; i &lt; nMaxIter &amp;&amp; !m_bStop; i++)
        {
          mpfr_set(X.m_r.m_f.backend().data(), Xr, MPFR_RNDN);
          mpfr_set(X.m_i.m_f.backend().data(), Xi, MPFR_RNDN);
          {
            <xsl:value-of select="reference" />
            DLOOP
          }
          mpfr_set(Xrn, Xn.m_r.m_f.backend().data(), MPFR_RNDN);
          mpfr_set(Xin, Xn.m_i.m_f.backend().data(), MPFR_RNDN);
    LOOP  }
    #undef DLOOP
    </xsl:when>
    <xsl:when test="reference/@t='R'">
    @rd    {
            const T Xxr = Xrd; (void) Xxr;
            const T Xxi = Xid; (void) Xxi;
    <xsl:choose>
```

```
<xsl:when test="derivative/@t='C'">
        const complex&lt;S&gt; Xx(Xxr, Xxi), A(g_FactorAR, g_FactorAI);
        const complex&lt;T&gt; d(dr, di);
        const complex&lt;S&gt; d0(daa, dba); <!-- FIXME matrix derivatives -->
        complex&lt;S&gt; dn(0.0, 0.0);
        <xsl:value-of select="derivative" />
        drn = dn.m_r; din = dn.m_i;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
        <xsl:value-of select="derivative" />
</xsl:when>
</xsl:choose>
      }
@rr   {
        <xsl:value-of select="reference" />
      }
</xsl:when>
</xsl:choose>

#undef LOOP
    mpfr_clear(Cr);
    mpfr_clear(Ci);
    mpfr_clear(Xr);
    mpfr_clear(Xi);
    mpfr_clear(Xr2);
    mpfr_clear(Xi2);
    mpfr_clear(Xrn);
    mpfr_clear(Xin);
    mpfr_clear(Ar);
    mpfr_clear(Ai);
    Xxr0 = double(Xrd);
    Xxi0 = double(Xid);
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
    dr0 = dr; di0 = di;
</xsl:when>
<xsl:when test="derivative/@t='M'">
    dr0 = (Xrd * dxa + Xid * dya) / sqrt(test1);
    di0 = (Xrd * dxb + Xid * dyb) / sqrt(test1);
</xsl:when>
</xsl:choose>
    return true;
  }
  return false;
}

</xsl:for-each>

template &lt;typename S, typename T&gt;
bool reference
  ( const int m_nFractalType, const int m_nPower
  , T *m_db_dxr, T *m_db_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
    ↳ int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr, const CFixedFloat &amp;Ci
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
```

```
    , const double terminate, const double g_real, const double g_imag
    , const bool m_bGlitchLowTolerance
    , int64_t &amp;antal, double &amp;test1, double &amp;test2
    , double &amp;Xxr0, double &amp;Xxi0
    , S &amp;dr, S &amp;di
    , const S &amp;daa, const S &amp;dab, const S &amp;dba, const S &amp;dbb
    )
{
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
      <xsl:for-each select="formula">
        case <xsl:value-of select="@power" />:
          return FORMULA(reference,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
            ( m_nFractalType, m_nPower
            , m_db_dxr, m_db_dxi, m_db_z
            , m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter
            , Cr, Ci
            , g_SeedR, g_SeedI
            , g_FactorAR, g_FactorAI
            , terminate, g_real, g_imag
            , m_bGlitchLowTolerance
            , antal, test1, test2
            , Xxr0, Xxi0
            , dr, di
            , daa, dab, dba, dbb
            );
      </xsl:for-each>
      }
      break;
  </xsl:for-each>
  }
  return false;
}

template bool reference&lt;double, double&gt;
  ( const int m_nFractalType, const int m_nPower
  , double *m_db_dxr, double *m_db_dxi, double *m_db_z
  , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, int64_t &amp;m_nMaxIter
  , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
  , const double g_SeedR, const double g_SeedI
  , const double g_FactorAR, const double g_FactorAI
  , const double terminate, const double g_real, const double g_imag
  , const bool m_bGlitchLowTolerance
  , int64_t &amp;antal, double &amp;test1, double &amp;test2
  , double &amp;Xxr0, double &amp;Xxi0
  , double &amp;dr0, double &amp;di0
  , const double &amp;daa, const double &amp;dab, const double &amp;dba, const double &amp;dbb
  );
template bool reference&lt;long double, double&gt;
```

```
    ( const int m_nFractalType, const int m_nPower
    , double *m_db_dxr, double *m_db_dxi, double *m_db_z
    , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
      ↳ int64_t &amp;m_nMaxIter
    , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
    , const double g_SeedR, const double g_SeedI
    , const double g_FactorAR, const double g_FactorAI
    , const double terminate, const double g_real, const double g_imag
    , const bool m_bGlitchLowTolerance
    , int64_t &amp;antal, double &amp;test1, double &amp;test2
    , double &amp;Xxr0, double &amp;Xxi0
    , long double &amp;dr0, long double &amp;di0
    , const long double &amp;daa, const long double &amp;dab, const long double &↙
      ↳ amp;dba, const long double &amp;dbb
    );
  template bool reference&lt;long double, long double&gt;
    ( const int m_nFractalType, const int m_nPower
    , long double *m_db_dxr, long double *m_db_dxi, double *m_db_z
    , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
      ↳ int64_t &amp;m_nMaxIter
    , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
    , const double g_SeedR, const double g_SeedI
    , const double g_FactorAR, const double g_FactorAI
    , const double terminate, const double g_real, const double g_imag
    , const bool m_bGlitchLowTolerance
    , int64_t &amp;antal, double &amp;test1, double &amp;test2
    , double &amp;Xxr0, double &amp;Xxi0
    , long double &amp;dr0, long double &amp;di0
    , const long double &amp;daa, const long double &amp;dab, const long double &↙
      ↳ amp;dba, const long double &amp;dbb
    );
  template bool reference&lt;floatexp, long double&gt;
    ( const int m_nFractalType, const int m_nPower
    , long double *m_db_dxr, long double *m_db_dxi, double *m_db_z
    , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
      ↳ int64_t &amp;m_nMaxIter
    , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
    , const double g_SeedR, const double g_SeedI
    , const double g_FactorAR, const double g_FactorAI
    , const double terminate, const double g_real, const double g_imag
    , const bool m_bGlitchLowTolerance
    , int64_t &amp;antal, double &amp;test1, double &amp;test2
    , double &amp;Xxr0, double &amp;Xxi0
    , floatexp &amp;dr0, floatexp &amp;di0
    , const floatexp &amp;daa, const floatexp &amp;dab, const floatexp &amp;dba, ↙
      ↳ const floatexp &amp;dbb
    );
  template bool reference&lt;floatexp, floatexp&gt;
    ( const int m_nFractalType, const int m_nPower
    , floatexp *m_db_dxr, floatexp *m_db_dxi, double *m_db_z
    , bool &amp;m_bStop, int64_t &amp;m_nRDone, int64_t &amp;m_nGlitchIter, ↙
      ↳ int64_t &amp;m_nMaxIter
    , const CFixedFloat &amp;Cr0, const CFixedFloat &amp;Ci0
    , const double g_SeedR, const double g_SeedI
    , const double g_FactorAR, const double g_FactorAI
    , const double terminate, const double g_real, const double g_imag
    , const bool m_bGlitchLowTolerance
```

```
560     , int64_t &amp;antal, double &amp;test1, double &amp;test2
        , double &amp;Xxr0, double &amp;Xxi0
        , floatexp &amp;dr0, floatexp &amp;di0
        , const floatexp &amp;daa, const floatexp &amp;dab, const floatexp &amp;dba, ↙
          ↳ const floatexp &amp;dbb
        );
565
    #endif

    #ifdef PASS3 // perturbation

570 <xsl:for-each select="formulas/group/formula">

    template &lt;typename T&gt;
    bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of ↙
        ↳ select="@power" />)
        ( int m_nFractalType, int m_nPower
575     , const T *m_db_dxr, const T *m_db_dxi, const double *m_db_z
        , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
          ↳ bGlitch
        , double m_nBailout2, const int64_t nMaxIter
        , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
        , const double g_FactorAR, const double g_FactorAI
580     , T &amp;xr0, T &amp;xi0
        , const T &amp;cr, const T &amp;ci
        )
    {
      if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;amp;&amp;amp; m_nPower ↙
          ↳ == <xsl:value-of select="@power" />)
585   {
        const T Ar = g_FactorAR;
        const T Ai = g_FactorAI;
        const complex&lt;T&gt; A = { Ar, Ai };
        const complex&lt;T&gt; c = { cr, ci };
590     (void) Ar; // -Wunused-variable
        (void) Ai; // -Wunused-variable
        (void) A; // -Wunused-variable
        (void) c; // -Wunused-variable
        bool no_g = g_real == 1.0 &amp;amp;&amp;amp; g_imag == 1.0;
595     int64_t antal = antal0;
        double test1 = test10;
        double test2 = test20;
        T xr = xr0;
        T xi = xi0;
600     T Xxr = 0;
        T Xxi = 0;
        for (; antal &lt; nMaxIter; antal++)
        {
          const T Xr = m_db_dxr[antal];
605       const T Xi = m_db_dxi[antal];
          const double Xz = m_db_z[antal];
          Xxr = Xr + xr;
          Xxi = Xi + xi;
          test2 = test1;
610       if (no_g)
          {
            test1 = double(Xxr * Xxr + Xxi * Xxi);
```

```
        }
        else
        {
          test1 = double(g_real * Xxr * Xxr + g_imag * Xxi * Xxi);
        }
        if (test1 &lt; Xz)
        {
          bGlitch = true;
          if (! m_bNoGlitchDetection)
            break;
        }
        if (test1 &gt; m_nBailout2)
        {
          break;
        }
        T xrn, xin;

<xsl:choose>
<xsl:when test="perturbation/@t='C'">
        const complex&lt;T&gt; X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi};
        complex&lt;T&gt; xn;
        (void) X; (void) x; (void) Xx;
        using V = T;
        V dummy;
        (void) dummy;
@dc     {
          <xsl:value-of select="perturbation" />
        }
        xrn = xn.m_r; xin = xn.m_i;
</xsl:when>
<xsl:when test="perturbation/@t='R'">
        using V = T;
        V dummy;
        (void) dummy;
@d      {
          <xsl:value-of select="perturbation" />
        }
</xsl:when>
</xsl:choose>

        xr = xrn;
        xi = xin;
      }
      antal0 = antal;
      test10 = test1;
      test20 = test2;
      xr0 = Xxr;
      xi0 = Xxi;
      return true;
    }
    return false;
}

</xsl:for-each>

template &lt;typename T&gt;
bool perturbation
```

```
670     ( const int m_nFractalType, const int m_nPower
        , const T *m_dxr, const T *m_dxi, const double *m_db_z
        , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
        , const double m_nBailout2, const int64_t nMaxIter
        , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
675     , const double g_FactorAR, const double g_FactorAI
        , T &amp;xr, T &amp;xi
        , const T &amp;cr, const T &amp;ci
        )
    {
680   switch (m_nFractalType)
      {
      <xsl:for-each select="formulas/group">
        // <xsl:value-of select="@name" />
        case <xsl:value-of select="@type" />:
685       switch (m_nPower)
          {
          <xsl:for-each select="formula">
            case <xsl:value-of select="@power" />:
              return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:↙
                  ↳ value-of select="@power" />)
690             ( m_nFractalType, m_nPower
                , m_dxr, m_dxi, m_db_z
                , antal, test1, test2, bGlitch
                , m_nBailout2, nMaxIter
                , m_bNoGlitchDetection, g_real, g_imag
695             , g_FactorAR, g_FactorAI
                , xr, xi
                , cr, ci
                );
          </xsl:for-each>
700       }
          break;
      </xsl:for-each>
      }
      return false;
705 }

    template bool perturbation&lt;double&gt;
      ( const int m_nFractalType, const int m_nPower
      , const double *m_dxr, const double *m_dxi, const double *m_db_z
710   , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
      , const double m_nBailout2, const int64_t nMaxIter
      , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
      , const double g_FactorAR, const double g_FactorAI
      , double &amp;xr, double &amp;xi
715   , const double &amp;cr, const double &amp;ci
      );
    template bool perturbation&lt;long double&gt;
      ( const int m_nFractalType, const int m_nPower
      , const long double *m_dxr, const long double *m_dxi, const double *m_db_z
720   , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
      , const double m_nBailout2, const int64_t nMaxIter
      , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
      , const double g_FactorAR, const double g_FactorAI
      , long double &amp;xr, long double &amp;xi
725   , const long double &amp;cr, const long double &amp;ci
```

```
  );
template bool perturbation&lt;floatexp&gt;
  ( const int m_nFractalType, const int m_nPower
  , const floatexp *m_dxr, const floatexp *m_dxi, const double *m_db_z
  , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , floatexp &amp;xr, floatexp &amp;xi
  , const floatexp &amp;cr, const floatexp &amp;ci
  );

#endif

#ifdef PASS4 // perturbation with derivatives

<xsl:for-each select="formulas/group/formula">

template &lt;typename T&gt;
bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of ↙
    ↳ select="@power" />)
  ( int m_nFractalType, int m_nPower
  , const T *m_db_dxr, const T *m_db_dxi, const double *m_db_z
  , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
      ↳ bGlitch
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , T &amp;xr0, T &amp;xi0
  , const T &amp;cr, const T &amp;ci
  , T &amp;dr0, T &amp;di0
  , const T &amp;e, const T &amp;h
  , const T &amp;daa, const T &amp;dab, const T &amp;dba, const T &amp;dbb
  )
{
  (void) dr0; // -Wunused-parameter
  (void) di0; // -Wunused-parameter
  (void) h; // -Wunused-parameter
  (void) e; // -Wunused-parameter
  (void) daa; // -Wunused-parameter
  (void) dab; // -Wunused-parameter
  (void) dba; // -Wunused-parameter
  (void) dbb; // -Wunused-parameter
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
      ↳ == <xsl:value-of select="@power" />)
  {
    bool no_g = g_real == 1.0 &amp;&amp; g_imag == 1.0;
    const T Ar = g_FactorAR;
    const T Ai = g_FactorAI;
    const complex&lt;T&gt; A = { Ar, Ai };
    const complex&lt;T&gt; c = { cr, ci };
    (void) Ar; // -Wunused-variable
    (void) Ai; // -Wunused-variable
    (void) A; // -Wunused-variable
    (void) c; // -Wunused-variable
    int64_t antal = antal0;
    double test1 = test10;
```

```
780       double test2 = test20;
          T xr = xr0;
          T xi = xi0;
          (void) dr0; // -Wunused-parameter
          (void) di0; // -Wunused-parameter
785   <xsl:choose>
      <xsl:when test="derivative/@t='C' or derivative/@t='R'">
          T dr = dr0, di = di0;
      </xsl:when>
      <xsl:when test="derivative/@t='M'">
790       T dxa = dr0, dxb = -di0, dya = di0, dyb = dr0;
      </xsl:when>
      </xsl:choose>
          T Xxr = 0;
          T Xxi = 0;
795       for (; antal &lt; nMaxIter; antal++)
          {
            const T Xr = m_db_dxr[antal];
            const T Xi = m_db_dxi[antal];
            const double Xz = m_db_z[antal];
800         Xxr = Xr + xr;
            Xxi = Xi + xi;
            test2 = test1;
            if (no_g)
              test1 = double(Xxr * Xxr + Xxi * Xxi);
805         else
              test1 = double(g_real * Xxr * Xxr + g_imag * Xxi * Xxi);
            if (test1 &lt; Xz)
            {
      <xsl:choose>
810   <xsl:when test="../@type='0' and @power='2'">
              if (type_0_power_2_pixel_has_glitched(cr, ci, xr, xi, Xr, Xi, dxa / h, ↙
                  ↳ dya / h, e, h)) // FIXME matrix derivatives
              {
      </xsl:when>
      </xsl:choose>
815           bGlitch = true;
              if (! m_bNoGlitchDetection)
                break;
      <xsl:choose>
      <xsl:when test="../@type='0' and @power='2'">
820           }
      </xsl:when>
      </xsl:choose>
          }
          if (test1 &gt; m_nBailout2)
825       {
            break;
          }
          T xrn, xin;

830   <xsl:choose>
      <xsl:when test="derivative/@t='C' or derivative/@t='R'">
          T drn = 0, din = 0;
      </xsl:when>
      <xsl:when test="derivative/@t='M'">
835       T dxan = 0, dxbn = 0, dyan = 0, dybn = 0;
```

```
</xsl:when>
</xsl:choose>

        using V = T;
        V dummy;
        (void) dummy;
<xsl:choose>
<xsl:when test="perturbation/@t='C'">
        const complex&lt;T&gt; X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi};
        complex&lt;T&gt; xn;
        (void) X; (void) x; (void) Xx;
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@d      {
          <xsl:value-of select="derivative" />
        }
</xsl:when>
<xsl:when test="derivative/@t='C'">
        const complex&lt;T&gt; d = {dr, di}, d0 = {daa, dba}; <!-- FIXME matrix derivatives -->
        complex&lt;T&gt; dn;
@dc     {
          <xsl:value-of select="derivative" />
        }
        drn = dn.m_r; din = dn.m_i;
</xsl:when>
</xsl:choose>
@dc     {
          <xsl:value-of select="perturbation" />
        }
        xrn = xn.m_r; xin = xn.m_i;
</xsl:when>
<xsl:when test="perturbation/@t='R'">
<xsl:choose>
<xsl:when test="derivative/@t='C'">
        const complex&lt;T&gt; X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi}, d = {dr, di}, d0 = {daa, dba}; <!-- FIXME matrix derivatives -->
        complex&lt;T&gt; dn;
        (void) X; (void) x; (void) Xx; (void) d;
@dc     {
          <xsl:value-of select="derivative" />
        }
        drn = dn.m_r; din = dn.m_i;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@d      {
          <xsl:value-of select="derivative" />
        }
</xsl:when>
</xsl:choose>
@d      {
          <xsl:value-of select="perturbation" />
        }
</xsl:when>
</xsl:choose>

        xr = xrn;
```

```
      xi = xin;
<xsl:choose>
<xsl:when test="derivative/@t='M'">
      dxa = dxan; dxb = dxbn; dya = dyan; dyb = dybn;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
      dr = drn; di = din;
</xsl:when>
</xsl:choose>
    }
    antal0 = antal;
    test10 = test1;
    test20 = test2;
    xr0 = Xxr;
    xi0 = Xxi;
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
    dr0 = dr; di0 = di;
</xsl:when>
<xsl:when test="derivative/@t='M'">
    dr0 = (Xxr * dxa + Xxi * dya) / sqrt(test1);
    di0 = (Xxr * dxb + Xxi * dyb) / sqrt(test1);
</xsl:when>
</xsl:choose>
    return true;
  }
  return false;
}

</xsl:for-each>

template &lt;typename T&gt;
bool perturbation
  ( int m_nFractalType, int m_nPower
  , const T *m_db_dxr, const T *m_db_dxi, const double *m_db_z
  , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , T &amp;xr, T &amp;xi
  , const T &amp;cr, const T &amp;ci
  , T &amp;dr, T &amp;di
  , const T &amp;e, const T &amp;h
  , const T &amp;daa, const T &amp;dab, const T &amp;dba, const T &amp;dbb
  )
{
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
      <xsl:for-each select="formula">
        case <xsl:value-of select="@power" />:
          return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
```

```
                ( m_nFractalType, m_nPower
                , m_db_dxr, m_db_dxi, m_db_z
                , antal, test1, test2, bGlitch
                , m_nBailout2, nMaxIter
                , m_bNoGlitchDetection, g_real, g_imag
                , g_FactorAR, g_FactorAI
                , xr, xi
                , cr, ci
                , dr, di
                , e, h
                , daa, dab, dba, dbb
                );
        </xsl:for-each>
        }
        break;
    </xsl:for-each>
    }
    return false;
}

template bool perturbation&lt;double&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
      ↳ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double &amp;xr0, double &amp;xi0
    , const double &amp;cr, const double &amp;ci
    , double &amp;dr0, double &amp;di0
    , const double &amp;e, const double &amp;h
    , const double &amp;daa, const double &amp;dab, const double &amp;dba, const ↙
      ↳ double &amp;dbb
    );
template bool perturbation&lt;long double&gt;
    ( int m_nFractalType, int m_nPower
    , const long double *m_db_dxr, const long double *m_db_dxi, const double *↙
      ↳ m_db_z
    , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
      ↳ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , long double &amp;xr0, long double &amp;xi0
    , const long double &amp;cr, const long double &amp;ci
    , long double &amp;dr0, long double &amp;di0
    , const long double &amp;e, const long double &amp;h
    , const long double &amp;daa, const long double &amp;dab, const long double &↙
      ↳ amp;dba, const long double &amp;dbb
    );
template bool perturbation&lt;floatexp&gt;
    ( int m_nFractalType, int m_nPower
    , const floatexp *m_db_dxr, const floatexp *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
      ↳ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
```

```
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , floatexp &amp;xr0, floatexp &amp;xi0
    , const floatexp &amp;cr, const floatexp &amp;ci
    , floatexp &amp;dr0, floatexp &amp;di0
    , const floatexp &amp;e, const floatexp &amp;h
    , const floatexp &amp;daa, const floatexp &amp;dab, const floatexp &amp;dba, const floatexp &amp;dbb
  );

#endif

#ifdef PASS5 // perturbation with SIMD

<xsl:for-each select="formulas/group/formula">

template &lt;typename intN, typename doubleN&gt;
bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
  ( int m_nFractalType, int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , intN &amp;antal0, doubleN &amp;test10, doubleN &amp;test20, intN &amp;bGlitch0
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , doubleN &amp;xr00, doubleN &amp;xi00
  , const doubleN &amp;cr0, const doubleN &amp;ci0
  , const int64_t chunksize
  )
{
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower == <xsl:value-of select="@power" />)
  {
    const double Ar = g_FactorAR;
    const double Ai = g_FactorAI;
    const complex&lt;double&gt; A = { Ar, Ai };
    const complex&lt;doubleN&gt; c = { cr0, ci0 };
    (void) Ar; // -Wunused-variable
    (void) Ai; // -Wunused-variable
    (void) A; // -Wunused-variable
    (void) c; // -Wunused-variable
    bool no_g = g_real == 1.0 &amp;&amp; g_imag == 1.0;
    doubleN test1 = test10;
    doubleN test2 = test20;
    doubleN xr0 = xr00;
    doubleN xi0 = xi00;
    intN antal = antal0;
    intN bGlitch = bGlitch0;
    intN bBailed = test1 &gt; m_nBailout2;
    // vectorized loop
    if (all(antal == antal[0]))
    {
      for (; antal[0] + chunksize - 1 &lt; nMaxIter; antal = antal + chunksize)
      {
        doubleN xr_saved = xr0;
        doubleN xi_saved = xi0;
```

```
          doubleN test1_saved = test1;
          for (int64_t q = 0; q &lt; chunksize; ++q)
          {
            const double Xr = m_db_dxr[antal[0] + q];
1055        const double Xi = m_db_dxi[antal[0] + q];
            const double Xz = m_db_z[antal[0] + q];
            doubleN Xxr = Xr + xr0;
            doubleN Xxi = Xi + xi0;
            test2 = test1;
1060        if (no_g)
            {
              test1 = Xxr * Xxr + Xxi * Xxi;
            }
            else
1065        {
              test1 = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
            }
            bGlitch |= test1 &lt; Xz;
            bBailed |= test1 &gt; m_nBailout2;
1070        doubleN xrn, xin;

        using T = double;
        T dummyT;
        (void) dummyT;
1075    using V = doubleN;
        V dummyV;
        (void) dummyV;
    <xsl:choose>
    <xsl:when test="perturbation/@t='C'">
1080    const complex&lt;double&gt; X = {Xr, Xi};
        const complex&lt;doubleN&gt; x = {xr0, xi0}, Xx = {Xxr, Xxi};
        complex&lt;doubleN&gt; xn;
        (void) X; (void) x; (void) Xx;
    @dc {
1085      <xsl:value-of select="perturbation" />
        }
        xrn = xn.m_r; xin = xn.m_i;
    </xsl:when>
    <xsl:when test="perturbation/@t='R'">
1090    doubleN cr = cr0, ci = ci0, xr = xr0, xi = xi0;
    @d  {
          <xsl:value-of select="perturbation" />
        }
    </xsl:when>
1095 </xsl:choose>

            xr0 = xrn;
            xi0 = xin;
          }
1100      if (any(bGlitch) || any(bBailed))
          {
            // rollback last chunk
            xr0 = xr_saved;
            xi0 = xi_saved;
1105        test1 = test1_saved;
            bGlitch = bGlitch0;
            break;
```

```
        }
      }
    }

    // finish up unvectorized
    const int64_t N = sizeof(doubleN) / sizeof(double);
    for (int64_t k = 0; k &lt; N; ++k)
    {
      double Xxr = 0, Xxi = 0;
      for (; antal[k] &lt; nMaxIter; antal[k] = antal[k] + 1)
      {
        const double Xr = m_db_dxr[antal[k]];
        const double Xi = m_db_dxi[antal[k]];
        const double Xz = m_db_z[antal[k]];
        Xxr = Xr + xr0[k];
        Xxi = Xi + xi0[k];
        test2[k] = test1[k];
        if (no_g)
        {
          test1[k] = Xxr * Xxr + Xxi * Xxi;
        }
        else
        {
          test1[k] = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
        }
        if (test1[k] &lt; Xz)
        {
          bGlitch[k] = true;
          if (! m_bNoGlitchDetection)
            break;
        }
        if (test1[k] &gt; m_nBailout2)
        {
          break;
        }
        double xrn, xin;

      using T = double;
      T dummyT;
      (void) dummyT;
      using V = double;
      V dummyV;
      (void) dummyV;
<xsl:choose>
<xsl:when test="perturbation/@t='C'">
      const complex&lt;double&gt; X = {Xr, Xi}, x = {xr0[k], xi0[k]}, Xx = {Xxr,↙
          ↪ Xxi}, c = { cr0[k], ci0[k] };
      complex&lt;double&gt; xn;
      (void) X; (void) x; (void) Xx;
@dc   {
        <xsl:value-of select="perturbation" />
      }
      xrn = xn.m_r; xin = xn.m_i;
</xsl:when>
<xsl:when test="perturbation/@t='R'">
      double cr = cr0[k], ci = ci0[k], xr = xr0[k], xi = xi0[k];
@d    {
```

```
        <xsl:value-of select="perturbation" />
      }
</xsl:when>
</xsl:choose>

        xr0[k] = xrn;
        xi0[k] = xin;
      }
      xr00[k] = Xxr;
      xi00[k] = Xxi;
    }

    antal0 = antal;
    bGlitch0 = bGlitch;
    test10 = test1;
    test20 = test2;
    return true;
  }
  return false;
}

</xsl:for-each>

template &lt;typename intN, typename doubleN&gt;
bool perturbation
  ( const int m_nFractalType, const int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , intN &amp;antal, doubleN &amp;test1, doubleN &amp;test2, intN &amp;bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , doubleN &amp;xr, doubleN &amp;xi
  , const doubleN &amp;cr, const doubleN &amp;ci
  , const int64_t chunksize
  )
{
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
      <xsl:for-each select="formula">
        case <xsl:value-of select="@power" />:
          return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
            ( m_nFractalType, m_nPower
            , m_db_dxr, m_db_dxi, m_db_z
            , antal, test1, test2, bGlitch
            , m_nBailout2, nMaxIter
            , m_bNoGlitchDetection, g_real, g_imag
            , g_FactorAR, g_FactorAI
            , xr, xi
            , cr, ci
            , chunksize
            );
```

```
1220          </xsl:for-each>
              }
              break;
        </xsl:for-each>
        }
1225    return false;
}

// explicit template instantiation

1230 #if KF_SIMD >= 1
template bool perturbation&lt;int2, double2&gt;
  ( const int m_nFractalType, const int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , int2 &amp;antal, double2 &amp;test1, double2 &amp;test2, int2 &amp;bGlitch
1235   , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_b2oGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , double2 &amp;xr, double2 &amp;xi
  , const double2 &amp;cr, const double2 &amp;ci
1240   , const int64_t chunksize
  );
#endif

#if KF_SIMD >= 2
1245 template bool perturbation&lt;int4, double4&gt;
  ( const int m_nFractalType, const int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , int4 &amp;antal, double4 &amp;test1, double4 &amp;test2, int4 &amp;bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
1250   , const bool m_b4oGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , double4 &amp;xr, double4 &amp;xi
  , const double4 &amp;cr, const double4 &amp;ci
  , const int64_t chunksize
1255   );
#endif

#if KF_SIMD >= 3
template bool perturbation&lt;int8, double8&gt;
1260   ( const int m_nFractalType, const int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , int8 &amp;antal, double8 &amp;test1, double8 &amp;test2, int8 &amp;bGlitch
  , const double m_nBailout2, const int64_t nMaxIter
  , const bool m_b8oGlitchDetection, const double g_real, const double g_imag
1265   , const double g_FactorAR, const double g_FactorAI
  , double8 &amp;xr, double8 &amp;xi
  , const double8 &amp;cr, const double8 &amp;ci
  , const int64_t chunksize
  );
1270 #endif

#if KF_SIMD >= 4
template bool perturbation&lt;int16, double16&gt;
  ( const int m_nFractalType, const int m_nPower
1275   , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , int16 &amp;antal, double16 &amp;test1, double16 &amp;test2, int16 &amp;↙
```

```
      ↪ bGlitch
    , const double m_nBailout2, const int64_t nMaxIter
    , const bool m_b16oGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
1280  , double16 &amp;xr, double16 &amp;xi
    , const double16 &amp;cr, const double16 &amp;ci
    , const int64_t chunksize
    );
#endif
1285
#endif

#ifdef PASSA // perturbation with SIMD and derivatives

1290 <xsl:for-each select="formulas/group/formula">

template &lt;typename intN, typename doubleN&gt;
bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of ↙
    ↪ select="@power" />)
  ( int m_nFractalType, int m_nPower
1295  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , intN &amp;antal0, doubleN &amp;test10, doubleN &amp;test20, intN &amp;↙
      ↪ bGlitch0
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
1300  , doubleN &amp;xr00, doubleN &amp;xi00
  , const doubleN &amp;cr0, const doubleN &amp;ci0
  , doubleN &amp;dr00, doubleN &amp;di00
  , const double e, const double h
  , const doubleN &amp;daa0, const doubleN &amp;dab0, const doubleN &amp;dba0, ↙
      ↪ const doubleN &amp;dbb0
1305  , const int64_t chunksize
  )
{
  (void) dr00; // -Wunused-parameter
  (void) di00; // -Wunused-parameter
1310  (void) h; // -Wunused-parameter
  (void) e; // -Wunused-parameter
  (void) daa0; // -Wunused-parameter
  (void) dab0; // -Wunused-parameter
  (void) dba0; // -Wunused-parameter
1315  (void) dbb0; // -Wunused-parameter
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
      ↪ == <xsl:value-of select="@power" />)
  {
    bool no_g = g_real == 1.0 &amp;&amp; g_imag == 1.0;
    const double Ar = g_FactorAR;
1320    const double Ai = g_FactorAI;
    const complex&lt;double&gt; A = { Ar, Ai };
    (void) Ar; // -Wunused-variable
    (void) Ai; // -Wunused-variable
    (void) A; // -Wunused-variable
1325    (void) dr00; // -Wunused-parameter
    (void) di00; // -Wunused-parameter
    doubleN test1 = test10;
    doubleN test2 = test20;
```

```
    doubleN xr0 = xr00;
    doubleN xi0 = xi00;
    intN antal = antal0;
    intN bGlitch = bGlitch0;
    intN bBailed = test1 &gt; m_nBailout2;
<xsl:choose>
<xsl:when test="derivative/@t='C' or derivative/@t='R'">
    doubleN dr0 = dr00, di0 = di00;
</xsl:when>
<xsl:when test="derivative/@t='M'">
    doubleN dxa0 = dr00, dxb0 = -di00, dya0 = di00, dyb0 = dr00;
</xsl:when>
</xsl:choose>
    // vectorized loop
    if (all(antal == antal[0]))
    {
      for (; antal[0] + chunksize - 1 &lt; nMaxIter; antal = antal + chunksize)
      {
        doubleN xr_saved = xr0;
        doubleN xi_saved = xi0;
<xsl:choose>
<xsl:when test="derivative/@t='M'">
        doubleN dxa_saved = dxa0, dxb_saved = dxb0, dya_saved = dya0, dyb_saved ↙
            ↳ = dyb0;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
        doubleN dr_saved = dr0, di_saved = di0;
</xsl:when>
</xsl:choose>
        doubleN test1_saved = test1;
        for (int64_t q = 0; q &lt; chunksize; ++q)
        {
          const double Xr = m_db_dxr[antal[0] + q];
          const double Xi = m_db_dxi[antal[0] + q];
          const double Xz = m_db_z[antal[0] + q];
          doubleN Xxr = Xr + xr0;
          doubleN Xxi = Xi + xi0;
          test2 = test1;
          if (no_g)
            test1 = Xxr * Xxr + Xxi * Xxi;
          else
            test1 = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
          bGlitch |= test1 &lt; Xz;
          bBailed |= test1 &gt; m_nBailout2;
      doubleN xr = xr0, xi = xi0, cr = cr0, ci = ci0;
      doubleN xrn, xin;
<xsl:choose>
<xsl:when test="derivative/@t='C' or derivative/@t='R'">
      doubleN dr = dr0, di = di0;
      doubleN drn, din;
</xsl:when>
<xsl:when test="derivative/@t='M'">
      doubleN daa = daa0, dab = dab0, dba = dba0, dbb = dbb0;
      doubleN dxa = dxa0, dxb = dxb0, dya = dya0, dyb = dyb0;
      doubleN dxan, dxbn, dyan, dybn;
</xsl:when>
</xsl:choose>
```

```
1385
        using T = double;
        T dummyT;
        (void) dummyT;
        using V = doubleN;
1390    V dummyV;
        (void) dummyV;
    <xsl:choose>
    <xsl:when test="perturbation/@t='C'">
        const complex&lt;double&gt; X = {Xr, Xi};
1395    const complex&lt;doubleN&gt; x = {xr, xi}, Xx = {Xxr, Xxi}, c = {cr, ci};
        complex&lt;doubleN&gt; xn;
        (void) X; (void) x; (void) Xx;
    <xsl:choose>
    <xsl:when test="derivative/@t='R' or derivative/@t='M'">
1400 @d     {
          <xsl:value-of select="derivative" />
        }
    </xsl:when>
    <xsl:when test="derivative/@t='C'">
1405    const complex&lt;doubleN&gt; d = {dr, di}, d0 = {daa0, dba0}; <!-- FIXME ↙
            ↳ matrix derivatives -->
        complex&lt;doubleN&gt; dn;
    @dc     {
          <xsl:value-of select="derivative" />
        }
1410    drn = dn.m_r; din = dn.m_i;
    </xsl:when>
    </xsl:choose>
    @dc     {
          <xsl:value-of select="perturbation" />
1415    }
        xrn = xn.m_r; xin = xn.m_i;
    </xsl:when>
    <xsl:when test="perturbation/@t='R'">
    <xsl:choose>
1420 <xsl:when test="derivative/@t='C'">
        const complex&lt;double&gt; X = {Xr, Xi};
        const complex&lt;doubleN&gt; x = {xr, xi}, Xx = {Xxr, Xxi}, c = {cr, ci}, ↙
            ↳ d = {dr, di}, d0 = {daa, dba}; <!-- FIXME matrix derivatives -->
        complex&lt;doubleN&gt; dn;
        (void) X; (void) x; (void) Xx; (void) d;
1425 @dc     {
          <xsl:value-of select="derivative" />
        }
        drn = dn.m_r; din = dn.m_i;
    </xsl:when>
1430 <xsl:when test="derivative/@t='R' or derivative/@t='M'">
    @d      {
          <xsl:value-of select="derivative" />
        }
    </xsl:when>
1435 </xsl:choose>
    @d      {
          <xsl:value-of select="perturbation" />
        }
    </xsl:when>
```

```
</xsl:choose>

          xr0 = xrn;
          xi0 = xin;
<xsl:choose>
<xsl:when test="derivative/@t='M'">
          dxa0 = dxan; dxb0 = dxbn; dya0 = dyan; dyb0 = dybn;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
          dr0 = drn; di0 = din;
</xsl:when>
</xsl:choose>
        }

        if (any(bGlitch) || any(bBailed))
        {
          // rollback last chunk
          xr0 = xr_saved;
          xi0 = xi_saved;
<xsl:choose>
<xsl:when test="derivative/@t='M'">
          dxa0 = dxa_saved; dxb0 = dxb_saved; dya0 = dya_saved; dyb0 = dyb_saved;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
          dr0 = dr_saved; di0 = di_saved;
</xsl:when>
</xsl:choose>
          test1 = test1_saved;
          bGlitch = bGlitch0;
          break;
        }
      }
    }

    // finish up unvectorized
    const int64_t N = sizeof(doubleN) / sizeof(double);
    for (int64_t k = 0; k &lt; N; ++k)
    {
      double Xxr = 0, Xxi = 0;
      for (; antal[k] &lt; nMaxIter; antal[k] = antal[k] + 1)
      {
        const double Xr = m_db_dxr[antal[k]];
        const double Xi = m_db_dxi[antal[k]];
        const double Xz = m_db_z[antal[k]];
        const double cr = cr0[k], ci = ci0[k];
        double xr = xr0[k], xi = xi0[k];
        Xxr = Xr + xr;
        Xxi = Xi + xi;
        test2[k] = test1[k];
        if (no_g)
        {
          test1[k] = Xxr * Xxr + Xxi * Xxi;
        }
        else
        {
          test1[k] = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
```

```
            }
            if (test1[k] &lt; Xz)
            {
    <xsl:choose>
1500    <xsl:when test="../@type='0' and @power='2'">
    #ifdef KF_USE_TYPE_0_POWER_2_HAS_GLITCHED
            if (type_0_power_2_pixel_has_glitched(cr, ci, xr, xi, Xr, Xi, dxa0[k] / ↙
                ↳ h, dya0[k] / h, e, h)) // FIXME matrix derivatives
            {
    #endif
1505    </xsl:when>
    </xsl:choose>
            bGlitch[k] = true;
            if (! m_bNoGlitchDetection)
              break;
1510    <xsl:choose>
    <xsl:when test="../@type='0' and @power='2'">
    #ifdef KF_USE_TYPE_0_POWER_2_HAS_GLITCHED
            }
    #endif
1515    </xsl:when>
    </xsl:choose>
            }
            if (test1[k] &gt; m_nBailout2)
            {
1520          break;
            }
            double xrn = 0, xin = 0;

          using T = double;
1525      T dummyT;
          (void) dummyT;
          using V = double;
          V dummyV;
          (void) dummyV;
1530    <xsl:choose>
    <xsl:when test="derivative/@t='C' or derivative/@t='R'">
          double dr = dr0[k], di = di0[k];
          double drn = 0, din = 0;
    </xsl:when>
1535    <xsl:when test="derivative/@t='M'">
          double daa = daa0[k], dab = dab0[k], dba = dba0[k], dbb = dbb0[k];
          double dxa = dxa0[k], dxb = dxb0[k], dya = dya0[k], dyb = dyb0[k];
          double dxan = 0, dxbn = 0, dyan = 0, dybn = 0;
    </xsl:when>
1540    </xsl:choose>
    <xsl:choose>
    <xsl:when test="perturbation/@t='C'">
          const complex&lt;double&gt; X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi}, c↙
              ↳  = {cr, ci};
          complex&lt;double&gt; xn;
1545      (void) X; (void) x; (void) Xx;
    <xsl:choose>
    <xsl:when test="derivative/@t='R' or derivative/@t='M'">
    @d    {
            <xsl:value-of select="derivative" />
1550      }
```

```
</xsl:when>
<xsl:when test="derivative/@t='C'">
      const complex&lt;double&gt; d = {dr, di}, d0 = {daa0[k], dba0[k]}; <!-- ↙
          ↳ FIXME matrix derivatives -->
      complex&lt;double&gt; dn;
@dc   {
        <xsl:value-of select="derivative" />
      }
      drn = dn.m_r; din = dn.m_i;
</xsl:when>
</xsl:choose>
@dc   {
        <xsl:value-of select="perturbation" />
      }
      xrn = xn.m_r; xin = xn.m_i;
</xsl:when>
<xsl:when test="perturbation/@t='R'">
<xsl:choose>
<xsl:when test="derivative/@t='C'">
      const complex&lt;double&gt; X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi}, c↙
          ↳  = {cr, ci}, d = {dr, di}, d0 = {daa[k], dba[k]}; <!-- FIXME matrix ↙
          ↳ derivatives -->
      complex&lt;double&gt; dn;
      (void) X; (void) x; (void) Xx; (void) d;
@dc   {
        <xsl:value-of select="derivative" />
      }
      drn = dn.m_r; din = dn.m_i;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@d    {
        <xsl:value-of select="derivative" />
      }
</xsl:when>
</xsl:choose>
@d    {
        <xsl:value-of select="perturbation" />
      }
</xsl:when>
</xsl:choose>

<xsl:choose>
<xsl:when test="derivative/@t='M'">
      dxa0[k] = dxan; dxb0[k] = dxbn; dya0[k] = dyan; dyb0[k] = dybn;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
      dr0[k] = drn; di0[k] = din;
</xsl:when>
</xsl:choose>

        xr0[k] = xrn;
        xi0[k] = xin;

      }
      xr00[k] = Xxr;
      xi00[k] = Xxi;
<xsl:choose>
```

```
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
    dr00[k] = dr0[k]; di00[k] = di0[k];
</xsl:when>
<xsl:when test="derivative/@t='M'">
    dr00[k] = (Xxr * dxa0[k] + Xxi * dya0[k]) / sqrt(test1[k]);
    di00[k] = (Xxr * dxb0[k] + Xxi * dyb0[k]) / sqrt(test1[k]);
</xsl:when>
</xsl:choose>
    }

    antal0 = antal;
    bGlitch0 = bGlitch;
    test10 = test1;
    test20 = test2;
    return true;
  }
  return false;
}

</xsl:for-each>

template &lt;typename intN, typename doubleN&gt;
bool perturbation
  ( int m_nFractalType, int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , intN &amp;antal0, doubleN &amp;test10, doubleN &amp;test20, intN &amp;bGlitch0
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , doubleN &amp;xr0, doubleN &amp;xi0
  , const doubleN &amp;cr0, const doubleN &amp;ci0
  , doubleN &amp;dr00, doubleN &amp;di00
  , const double e, const double h
  , const doubleN &amp;daa0, const doubleN &amp;dab0, const doubleN &amp;dba0, const doubleN &amp;dbb0
  , const int64_t chunk_size
  )
{
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
      <xsl:for-each select="formula">
        case <xsl:value-of select="@power" />:
          return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
            ( m_nFractalType, m_nPower
            , m_db_dxr, m_db_dxi, m_db_z
            , antal0, test10, test20, bGlitch0
            , m_nBailout2, nMaxIter
            , m_bNoGlitchDetection, g_real, g_imag
            , g_FactorAR, g_FactorAI
            , xr0, xi0
```

```
                , cr0, ci0
                , dr00, di00
                , e, h
                , daa0, dab0, dba0, dbb0
                , chunk_size
                );
        </xsl:for-each>
        }
        break;
    </xsl:for-each>
    }
    return false;
}

#if KF_SIMD >= 1
template bool perturbation&lt;int2, double2&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int2 &amp;antal0, double2 &amp;test10, double2 &amp;test20, int2 &amp;
      ↳ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double2 &amp;xr0, double2 &amp;xi0
    , const double2 &amp;cr0, const double2 &amp;ci0
    , double2 &amp;dr00, double2 &amp;di00
    , const double e, const double h
    , const double2 &amp;daa0, const double2 &amp;dab0, const double2 &amp;dba0,
      ↳ const double2 &amp;dbb0
    , const int64_t chunk_size
    );
#endif

#if KF_SIMD >= 2
template bool perturbation&lt;int4, double4&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int4 &amp;antal0, double4 &amp;test10, double4 &amp;test20, int4 &amp;
      ↳ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double4 &amp;xr0, double4 &amp;xi0
    , const double4 &amp;cr0, const double4 &amp;ci0
    , double4 &amp;dr00, double4 &amp;di00
    , const double e, const double h
    , const double4 &amp;daa0, const double4 &amp;dab0, const double4 &amp;dba0,
      ↳ const double4 &amp;dbb0
    , const int64_t chunk_size
    );
#endif

#if KF_SIMD >= 3
template bool perturbation&lt;int8, double8&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int8 &amp;antal0, double8 &amp;test10, double8 &amp;test20, int8 &amp;
```

```
      ↪ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double8 &amp;xr0, double8 &amp;xi0
    , const double8 &amp;cr0, const double8 &amp;ci0
    , double8 &amp;dr00, double8 &amp;di00
    , const double e, const double h
    , const double8 &amp;daa0, const double8 &amp;dab0, const double8 &amp;dba0, ↙
      ↪ const double8 &amp;dbb0
    , const int64_t chunk_sizes
    );
#endif

#if KF_SIMD >= 4
template bool perturbation&lt;int16, double16&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int16 &amp;antal0, double16 &amp;test10, double16 &amp;test20, int16 &amp;↙
      ↪ bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double16 &amp;xr0, double16 &amp;xi0
    , const double16 &amp;cr0, const double16 &amp;ci0
    , double16 &amp;dr00, double16 &amp;di00
    , const double e, const double h
    , const double16 &amp;daa0, const double16 &amp;dab0, const double16 &amp;dba0↙
      ↪ , const double16 &amp;dbb0
    , const int64_t chunk_size
    );
#endif

#endif

#ifdef PASS6 // perturbation with scaling

<xsl:for-each select="//scaled/..">

template &lt;typename T&gt;
bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of ↙
  ↪ select="@power" />)
    ( const int m_nFractalType, const int m_nPower
    , const T *m_db_dxr, const T *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
      ↪ bGlitch0
    , const double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , T &amp;xr00, T &amp;xi00
    , const T &amp;cr0, const T &amp;ci0
    , const T &amp;s, const T &amp;S
    )
{
    if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
      ↪ == <xsl:value-of select="@power" />)
    {
```

```
      const double Ar = g_FactorAR;
      const double Ai = g_FactorAI;
      const complex&lt;double&gt; A = { Ar, Ai };
1765  const complex&lt;T&gt; c = { cr0, ci0 };
      (void) Ar; // -Wunused-variable
      (void) Ai; // -Wunused-variable
      (void) A; // -Wunused-variable
      (void) c; // -Wunused-variable
1770  bool no_g = g_real == 1.0 &amp;amp;&amp;amp; g_imag == 1.0;
      double test1 = test10;
      double test2 = test20;
      T xr0 = xr00;
      T xi0 = xi00;
1775  int64_t antal = antal0;
      bool bGlitch = bGlitch0;
      T Xxr = 0;
      T Xxi = 0;
        for (; antal &lt; nMaxIter; antal = antal + 1)
1780    {
          const T Xr = m_db_dxr[antal];
          const T Xi = m_db_dxi[antal];
          const double Xz = m_db_z[antal];
          Xxr = Xr + xr0 * s;
1785      Xxi = Xi + xi0 * s;
          test2 = test1;
          if (no_g)
          {
            test1 = Xxr * Xxr + Xxi * Xxi;
1790      }
          else
          {
            test1 = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
          }
1795      if (test1 &lt; Xz)
          {
            bGlitch = true;
            if (! m_bNoGlitchDetection)
              break;
1800      }
          if (test1 &gt; m_nBailout2)
          {
            break;
          }
1805      double xrn, xin;

        using V = T;
        V dummyV;
        (void) dummyV;
1810  <xsl:choose>
      <xsl:when test="scaled/@t='R'">
        T cr = cr0, ci = ci0, xr = xr0, xi = xi0;
      @d  {
            <xsl:value-of select="scaled" />
1815    }
      </xsl:when>
      </xsl:choose>
```

```
          xr0 = xrn;
          xi0 = xin;
        }

      antal0 = antal;
      bGlitch0 = bGlitch;
      test10 = test1;
      test20 = test2;
      xr00 = Xxr;
      xi00 = Xxi;
      return true;
    }
    return false;
  }

  </xsl:for-each>

  template &lt;typename T&gt;
  bool perturbation
    ( const int m_nFractalType, const int m_nPower
    , const T *m_db_dxr, const T *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
    , const double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , T &amp;xr00, T &amp;xi00
    , const T &amp;cr0, const T &amp;ci0
    , const T &amp;s, const T &amp;S
    )
  {
  <xsl:for-each select="//scaled/..">
    if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
        ↳ == <xsl:value-of select="@power" />)
      return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of↙
          ↳  select="@power" />)
        ( m_nFractalType, m_nPower
        , m_db_dxr, m_db_dxi, m_db_z
        , antal, test1, test2, bGlitch
        , m_nBailout2, nMaxIter
        , m_bNoGlitchDetection, g_real, g_imag
        , g_FactorAR, g_FactorAI
        , xr00, xi00
        , cr0, ci0
        , s, S
        );
  </xsl:for-each>
    return false;
  }

  template bool perturbation&lt;double&gt;
    ( const int m_nFractalType, const int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
    , const double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double &amp;xr00, double &amp;xi00
```

```
    , const double &amp;cr0, const double &amp;ci0
    , const double &amp;s, const double &amp;S
    );
  template bool perturbation&lt;long double&gt;
    ( const int m_nFractalType, const int m_nPower
    , const long double *m_db_dxr, const long double *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
    , const double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , long double &amp;xr00, long double &amp;xi00
    , const long double &amp;cr0, const long double &amp;ci0
    , const long double &amp;s, const long double &amp;S
    );

  #endif

  #ifdef PASS7 // perturbation with SIMD and scaling

  <xsl:for-each select="//scaled/..">

  template &lt;typename intN, typename doubleN&gt;
  bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , intN &amp;antal0, doubleN &amp;test10, doubleN &amp;test20, intN &amp;bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , doubleN &amp;xr00, doubleN &amp;xi00
    , const doubleN &amp;cr0, const doubleN &amp;ci0
    , const int64_t chunksize
    , const double s, const double S
    )
  {
    if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower == <xsl:value-of select="@power" />)
    {
      const double Ar = g_FactorAR;
      const double Ai = g_FactorAI;
      const complex&lt;double&gt; A = { Ar, Ai };
      const complex&lt;doubleN&gt; c = { cr0, ci0 };
      (void) Ar; // -Wunused-variable
      (void) Ai; // -Wunused-variable
      (void) A; // -Wunused-variable
      (void) c; // -Wunused-variable
      bool no_g = g_real == 1.0 &amp;&amp; g_imag == 1.0;
      doubleN test1 = test10;
      doubleN test2 = test20;
      doubleN xr0 = xr00;
      doubleN xi0 = xi00;
      intN antal = antal0;
      intN bGlitch = bGlitch0;
      intN bBailed = test1 &gt; m_nBailout2;
```

```
      // vectorized loop
      if (all(antal == antal[0]))
      {
1930    for (; antal[0] + chunksize - 1 &lt; nMaxIter; antal = antal + chunksize)
        {
          doubleN xr_saved = xr0;
          doubleN xi_saved = xi0;
          doubleN test1_saved = test1;
1935      for (int64_t q = 0; q &lt; chunksize; ++q)
          {
            const double Xr = m_db_dxr[antal[0] + q];
            const double Xi = m_db_dxi[antal[0] + q];
            const double Xz = m_db_z[antal[0] + q];
1940        doubleN Xxr = Xr + xr0 * s;
            doubleN Xxi = Xi + xi0 * s;
            test2 = test1;
            if (no_g)
            {
1945          test1 = Xxr * Xxr + Xxi * Xxi;
            }
            else
            {
              test1 = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
1950        }
            bGlitch |= test1 &lt; Xz;
            bBailed |= test1 &gt; m_nBailout2;
            doubleN xrn, xin;

1955        using T = double;
            T dummyT;
            (void) dummyT;
            using V = doubleN;
            V dummyV;
1960        (void) dummyV;
      <xsl:choose>
      <xsl:when test="scaled/@t='R'">
            doubleN cr = cr0, ci = ci0, xr = xr0, xi = xi0;
      @d    {
1965          <xsl:value-of select="scaled" />
            }
      </xsl:when>
      </xsl:choose>

1970          xr0 = xrn;
              xi0 = xin;
            }
            if (any(bGlitch) || any(bBailed))
            {
1975          // rollback last chunk
              xr0 = xr_saved;
              xi0 = xi_saved;
              test1 = test1_saved;
              bGlitch = bGlitch0;
1980          break;
            }
          }
        }
```

```
1985        // finish up unvectorized
            int64_t N = sizeof(doubleN) / sizeof(double);
            for (int64_t k = 0; k &lt; N; ++k)
            {
              double Xxr = 0, Xxi = 0;
1990          for (; antal[k] &lt; nMaxIter; antal[k] = antal[k] + 1)
              {
                const double Xr = m_db_dxr[antal[k]];
                const double Xi = m_db_dxi[antal[k]];
                const double Xz = m_db_z[antal[k]];
1995            Xxr = Xr + xr0[k] * s;
                Xxi = Xi + xi0[k] * s;
                test2[k] = test1[k];
                if (no_g)
                {
2000              test1[k] = Xxr * Xxr + Xxi * Xxi;
                }
                else
                {
                  test1[k] = g_real * Xxr * Xxr + g_imag * Xxi * Xxi;
2005            }
                if (test1[k] &lt; Xz)
                {
                  bGlitch[k] = true;
                  if (! m_bNoGlitchDetection)
2010                break;
                }
                if (test1[k] &gt; m_nBailout2)
                {
                  break;
2015            }
                using T = double;
                T dummy;
                (void) dummy;
                using V = double;
2020            V dummyV;
                (void) dummyV;
                double xrn, xin;

            <xsl:choose>
2025        <xsl:when test="scaled/@t='R'">
              double cr = cr0[k], ci = ci0[k], xr = xr0[k], xi = xi0[k];
            @d  {
                  <xsl:value-of select="scaled" />
                }
2030        </xsl:when>
            </xsl:choose>

                  xr0[k] = xrn;
                  xi0[k] = xin;
2035            }
                xr00[k] = Xxr;
                xi00[k] = Xxi;
              }

2040          antal0 = antal;
```

```
      bGlitch0 = bGlitch;
      test10 = test1;
      test20 = test2;
      return true;
    }
    return false;
  }

  </xsl:for-each>

  template &lt;typename intN, typename doubleN&gt;
  bool perturbation
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , intN &amp;antal0, doubleN &amp;test10, doubleN &amp;test20, intN &amp;↙
      ↳ bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , doubleN &amp;xr00, doubleN &amp;xi00
    , const doubleN &amp;cr0, const doubleN &amp;ci0
    , const int64_t chunksize
    , const double s, const double S
    )
  {
  <xsl:for-each select="//scaled/..">
    if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
      ↳ == <xsl:value-of select="@power" />)
      return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of↙
        ↳ select="@power" />)
        ( m_nFractalType, m_nPower
        , m_db_dxr, m_db_dxi, m_db_z
        , antal0, test10, test20, bGlitch0
        , m_nBailout2, nMaxIter
        , m_bNoGlitchDetection, g_real, g_imag
        , g_FactorAR, g_FactorAI
        , xr00, xi00
        , cr0, ci0
        , chunksize
        , s, S
        );
  </xsl:for-each>
    return false;
  }

  #if KF_SIMD >= 1
  template bool perturbation&lt;int2, double2&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int2 &amp;antal0, double2 &amp;test10, double2 &amp;test20, int2 &amp;↙
      ↳ bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_b2oGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double2 &amp;xr00, double2 &amp;xi00
    , const double2 &amp;cr0, const double2 &amp;ci0
    , const int64_t chunksize
```

```
    , const double s, const double S
    );
#endif

#if KF_SIMD >= 2
template bool perturbation&lt;int4, double4&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int4 &amp;antal0, double4 &amp;test10, double4 &amp;test20, int4 &amp;↙
      ↳ bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_b4oGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double4 &amp;xr00, double4 &amp;xi00
    , const double4 &amp;cr0, const double4 &amp;ci0
    , const int64_t chunksize
    , const double s, const double S
    );
#endif

#if KF_SIMD >= 3
template bool perturbation&lt;int8, double8&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int8 &amp;antal0, double8 &amp;test10, double8 &amp;test20, int8 &amp;↙
      ↳ bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_b8oGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double8 &amp;xr00, double8 &amp;xi00
    , const double8 &amp;cr0, const double8 &amp;ci0
    , const int64_t chunksize
    , const double s, const double S
    );
#endif

#if KF_SIMD >= 4
template bool perturbation&lt;int16, double16&gt;
    ( int m_nFractalType, int m_nPower
    , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
    , int16 &amp;antal0, double16 &amp;test10, double16 &amp;test20, int16 &amp;↙
      ↳ bGlitch0
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_b16oGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , double16 &amp;xr00, double16 &amp;xi00
    , const double16 &amp;cr0, const double16 &amp;ci0
    , const int64_t chunksize
    , const double s, const double S
    );
#endif

#endif

#ifdef PASS8 // perturbation with derivatives and scaling

<xsl:for-each select="//scaled/..">
```

```
template &lt;typename D, typename Z&gt;
bool FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of select="@power" />)
  ( int m_nFractalType, int m_nPower
  , const Z *m_db_dxr, const Z *m_db_dxi, const double *m_db_z
  , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;bGlitch
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , Z &amp;xr0, Z &amp;xi0
  , const Z &amp;cr, const Z &amp;ci
  , D &amp;dr0, D &amp;di0
  , const D &amp;e, const D &amp;h
  , const D &amp;daa, const D &amp;dab, const D &amp;dba, const D &amp;dbb
  , const Z &amp;s, const Z &amp;S
  )
{
  (void) dr0; // -Wunused-parameter
  (void) di0; // -Wunused-parameter
  (void) h; // -Wunused-parameter
  (void) e; // -Wunused-parameter
  (void) daa; // -Wunused-parameter
  (void) dab; // -Wunused-parameter
  (void) dba; // -Wunused-parameter
  (void) dbb; // -Wunused-parameter
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower == <xsl:value-of select="@power" />)
  {
    bool no_g = g_real == 1.0 &amp;&amp; g_imag == 1.0;
    const Z Ar = g_FactorAR;
    const Z Ai = g_FactorAI;
    const complex&lt;Z&gt; A = { Ar, Ai };
    const complex&lt;Z&gt; c = { cr, ci };
    (void) Ar; // -Wunused-variable
    (void) Ai; // -Wunused-variable
    (void) A; // -Wunused-variable
    (void) c; // -Wunused-variable
    int64_t antal = antal0;
    double test1 = test10;
    double test2 = test20;
    Z xr = xr0;
    Z xi = xi0;
    (void) dr0; // -Wunused-parameter
    (void) di0; // -Wunused-parameter
<xsl:choose>
<xsl:when test="derivative/@t='C' or derivative/@t='R'">
    D dr = dr0, di = di0;
</xsl:when>
<xsl:when test="derivative/@t='M'">
    D dxa = dr0, dxb = -di0, dya = di0, dyb = dr0;
</xsl:when>
</xsl:choose>
    Z Xxr = 0;
    Z Xxi = 0;
    for (; antal &lt; nMaxIter; antal++)
```

```
      {
        const Z Xr = m_db_dxr[antal];
        const Z Xi = m_db_dxi[antal];
2205    const double Xz = m_db_z[antal];
        Xxr = Xr + xr * s;
        Xxi = Xi + xi * s;
        test2 = test1;
        if (no_g)
2210      test1 = double(Xxr * Xxr + Xxi * Xxi);
        else
          test1 = double(g_real * Xxr * Xxr + g_imag * Xxi * Xxi);
        if (test1 &lt; Xz)
        {
2215  <xsl:choose>
      <xsl:when test="../@type='0' and @power='2'">
      #ifdef KF_USE_TYPE_0_POWER_2_HAS_GLITCHED
          if (type_0_power_2_pixel_has_glitched(cr, ci, xr, xi, Xr, Xi, dxa / h, ↙
              ↪ dya / h, e, h)) // FIXME matrix derivatives
          {
2220  #endif
      </xsl:when>
      </xsl:choose>
          bGlitch = true;
          if (! m_bNoGlitchDetection)
2225        break;
      <xsl:choose>
      <xsl:when test="../@type='0' and @power='2'">
      #ifdef KF_USE_TYPE_0_POWER_2_HAS_GLITCHED
          }
2230  #endif
      </xsl:when>
      </xsl:choose>
        }
        if (test1 &gt; m_nBailout2)
2235    {
          break;
        }
        Z xrn, xin;

2240  <xsl:choose>
      <xsl:when test="derivative/@t='C' or derivative/@t='R'">
        D drn = 0, din = 0;
      </xsl:when>
      <xsl:when test="derivative/@t='M'">
2245    D dxan = 0, dxbn = 0, dyan = 0, dybn = 0;
      </xsl:when>
      </xsl:choose>
      <xsl:choose>
      <xsl:when test="scaled/@t='R'">
2250  {
        using T = D;
        T dummyT;
        (void) dummyT;
        using V = D;
2255    V dummyV;
        (void) dummyV;
      <xsl:choose>
```

```
<xsl:when test="derivative/@t='C'">
      const complex&lt;Z&gt; X = {Xr, Xi}, x = {xr, xi}, Xx = {Xxr, Xxi};
      const complex&lt;D&gt; d = {dr, di}, d0 = {daa, dba}; <!-- FIXME matrix derivatives -->
      complex&lt;D&gt; dn;
      (void) X; (void) x; (void) Xx; (void) d;
@dc   {
        <xsl:value-of select="derivative" />
      }
      drn = dn.m_r; din = dn.m_i;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='M'">
@d    {
        <xsl:value-of select="derivative" />
      }
</xsl:when>
</xsl:choose>
}
  using T = Z;
  T dummyT;
  (void) dummyT;
  using V = Z;
  V dummyV;
  (void) dummyV;

@d    {
        <xsl:value-of select="scaled" />
      }
</xsl:when>
</xsl:choose>

      xr = xrn;
      xi = xin;
<xsl:choose>
<xsl:when test="derivative/@t='M'">
      dxa = dxan; dxb = dxbn; dya = dyan; dyb = dybn;
</xsl:when>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
      dr = drn; di = din;
</xsl:when>
</xsl:choose>
    }
    antal0 = antal;
    test10 = test1;
    test20 = test2;
    xr0 = Xxr;
    xi0 = Xxi;
<xsl:choose>
<xsl:when test="derivative/@t='R' or derivative/@t='C'">
    dr0 = dr; di0 = di;
</xsl:when>
<xsl:when test="derivative/@t='M'">
    Z t = 1 / sqrt(test1);
    dr0 = Xxr * t * dxa + Xxi * t * dya;
    di0 = Xxr * t * dxb + Xxi * t * dyb;
</xsl:when>
</xsl:choose>
```

```
    return true;
  }
  return false;
}

</xsl:for-each>

template &lt;typename D, typename Z&gt;
bool perturbation
  ( int m_nFractalType, int m_nPower
  , const Z *m_db_dxr, const Z *m_db_dxi, const double *m_db_z
  , int64_t &amp;antal, double &amp;test1, double &amp;test2, bool &amp;bGlitch
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , Z &amp;xr, Z &amp;xi
  , const Z &amp;cr, const Z &amp;ci
  , D &amp;dr, D &amp;di
  , const D &amp;e, const D &amp;h
  , const D &amp;daa, const D &amp;dab, const D &amp;dba, const D &amp;dbb
  , const Z &amp;s, const Z &amp;S
  )
{
<xsl:for-each select="//scaled/..">
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower ↙
      ↳ == <xsl:value-of select="@power" />)
    return FORMULA(perturbation,<xsl:value-of select="../@type" />,<xsl:value-of↙
        ↳ select="@power" />)
      ( m_nFractalType, m_nPower
      , m_db_dxr, m_db_dxi, m_db_z
      , antal, test1, test2, bGlitch
      , m_nBailout2, nMaxIter
      , m_bNoGlitchDetection, g_real, g_imag
      , g_FactorAR, g_FactorAI
      , xr, xi
      , cr, ci
      , dr, di
      , e, h
      , daa, dab, dba, dbb
      , s, S
      );
</xsl:for-each>
  return false;
}

template bool perturbation&lt;long double, double&gt;
  ( int m_nFractalType, int m_nPower
  , const double *m_db_dxr, const double *m_db_dxi, const double *m_db_z
  , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;↙
      ↳ bGlitch
  , double m_nBailout2, const int64_t nMaxIter
  , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
  , const double g_FactorAR, const double g_FactorAI
  , double &amp;xr0, double &amp;xi0
  , const double &amp;cr, const double &amp;ci
  , long double &amp;dr0, long double &amp;di0
  , const long double &amp;e, const long double &amp;h
```

```
    , const long double &amp;daa, const long double &amp;dab, const long double &amp;dba, const long double &amp;dbb
    , const double &amp;s, const double &amp;S
    );
template bool perturbation&lt;floatexp, long double&gt;
    ( int m_nFractalType, int m_nPower
    , const long double *m_db_dxr, const long double *m_db_dxi, const double *m_db_z
    , int64_t &amp;antal0, double &amp;test10, double &amp;test20, bool &amp;bGlitch
    , double m_nBailout2, const int64_t nMaxIter
    , const bool m_bNoGlitchDetection, const double g_real, const double g_imag
    , const double g_FactorAR, const double g_FactorAI
    , long double &amp;xr0, long double &amp;xi0
    , const long double &amp;cr, const long double &amp;ci
    , floatexp &amp;dr0, floatexp &amp;di0
    , const floatexp &amp;e, const floatexp &amp;h
    , const floatexp &amp;daa, const floatexp &amp;dab, const floatexp &amp;dba, const floatexp &amp;dbb
    , const long double &amp;s, const long double &amp;S
    );

#endif

#ifdef PASS9 // miscellaneous

bool scaling_supported(const int m_nFractalType, const int m_nPower, const bool derivatives)
{
  <xsl:for-each select="//scaled/..">
  if (m_nFractalType == <xsl:value-of select="../@type" /> &amp;&amp; m_nPower == <xsl:value-of select="@power" />)
    return true;
  </xsl:for-each>
  return false;
}

void combo5_addstrings(HWND hWnd, const int IDC_COMBO5)
{
  <xsl:for-each select="formulas/group">
    SendDlgItemMessage(hWnd,IDC_COMBO5,CB_ADDSTRING,0,(LPARAM)"<xsl:value-of select="@name" />"); // <xsl:value-of select="@type" />
  </xsl:for-each>
}

int validate_power_for_fractal_type(const int m_nFractalType, const int m_nPower)
{
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
      switch (m_nPower)
      {
        default:
      <xsl:for-each select="formula">
```

```
          case <xsl:value-of select="@power" />: return <xsl:value-of select="
              ↪ @power" />;
        </xsl:for-each>
        }
        break;
    </xsl:for-each>
      default:
        return 2;
    }
}

void update_power_dropdown_for_fractal_type(HWND hWnd, const int IDC_COMBO3, 
    ↪ const int m_nFractalType, const int m_nPower)
{
  SendDlgItemMessage(hWnd,IDC_COMBO3,CB_RESETCONTENT,0,0);
  int selected = 0;
  int ix = 0;
  switch (m_nFractalType)
  {
  <xsl:for-each select="formulas/group">
    // <xsl:value-of select="@name" />
    case <xsl:value-of select="@type" />:
    {
      <xsl:for-each select="formula">
        if (<xsl:value-of select="@power" /> == m_nPower) selected = ix;
        SendDlgItemMessage(hWnd,IDC_COMBO3,CB_ADDSTRING,0,(LPARAM)"<xsl:value-of
            ↪  select="@power" />");
        ix++;
      </xsl:for-each>
      break;
    }
  </xsl:for-each>
  }
  SendDlgItemMessage(hWnd,IDC_COMBO3,CB_SETCURSEL,selected,0);
  EnableWindow(GetDlgItem(hWnd,IDC_COMBO3), ix > 1);
}

#endif

// EOF
</xsl:template>
</xsl:stylesheet>
```

44 formula/generated/formula.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

incorporating components derived from:

et -- escape time fractals
Copyright (C) 2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
```

```
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
15
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
20
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

25  #ifndef KF_GENERATED_FORMULA_H
    #define KF_GENERATED_FORMULA_H 1

    #include <assert.h>
    #include <math.h>
30  #include <mpfr.h>

    #define abs(x) ((x)<0?-(x):(x))

    #define sgn(x) (((x)>=0)-((x)<0))
35
    static inline float diffabsf(float c, float d) {
      if (c >= 0.0f) {
        if (c + d >= 0.0f) { return d; }
        else { return -(2.0f * c + d); }
40    } else {
        if (c + d > 0.0f) { return 2.0f * c + d; }
        else { return -d; }
      }
    }
45
    static inline double diffabs(double c, double d) {
      if (c >= 0.0) {
        if (c + d >= 0.0) { return d; }
        else { return -(2.0 * c + d); }
50    } else {
        if (c + d > 0.0) { return 2.0 * c + d; }
        else { return -d; }
      }
    }
55
    static inline long double diffabsl(long double c, long double d) {
      if (c >= 0.0L) {
        if (c + d >= 0.0L) { return d; }
        else { return -(2.0L * c + d); }
60    } else {
        if (c + d > 0.0L) { return 2.0L * c + d; }
        else { return -d; }
      }
    }
65
    typedef int f_plainf(int,float,float*,float,float,float,float,float*,volatile ↙
        ↳ int*);
    typedef int f_plain(int,double,double*,double,double,double,double,double*,↙
        ↳ volatile int*);
```

```
typedef int f_plainl(int,long double,long double*,long double,long double,long ↙
    ↳ double,long double,long double*,volatile int*);
typedef int f_referencef(int,float,float,float,mpfr_t,mpfr_t,float*,volatile int↙
    ↳ *);
typedef int f_reference(int,double,double,double,mpfr_t,mpfr_t,double*,volatile ↙
    ↳ int*);
typedef int f_referencel(int,long double,long double,long double,mpfr_t,mpfr_t,↙
    ↳ long double*,volatile int*);
typedef int f_perturbationf(int,int,float,float*,float,float,float,float,float*,↙
    ↳ float*,volatile int*);
typedef int f_perturbation(int,int,double,double*,double,double,double,double,↙
    ↳ double*,double*,volatile int*);
typedef int f_perturbationl(int,int,long double,long double*,long double,long ↙
    ↳ double,long double,long double,long double*,long double*,volatile int*);
typedef int f_period_tri(int,double,double,double,mpfr_t,mpfr_t,mpfr_t,volatile ↙
    ↳ int*);
typedef int f_period_jsk(int,double,double,double,mpfr_t,mpfr_t,mpfr_t,double*,↙
    ↳ volatile int*);
typedef int f_newton(int,int,double,double,mpfr_t,mpfr_t,volatile int*);
typedef int f_size(int,double,double,mpfr_t,mpfr_t,mpfr_t,double*,volatile int*)↙
    ↳ ;
typedef int f_skew(int,double,double,mpfr_t,mpfr_t,int,double*,volatile int*);
typedef int f_domain_size(int,double,double,mpfr_t,mpfr_t,mpfr_t,volatile int*);

#ifndef KF_MAIN
static f_plainf plainf;
static f_plain plain;
static f_plainl plainl;
static f_referencef referencef;
static f_reference reference;
static f_referencel referencel;
static f_perturbationf perturbationf;
static f_perturbation perturbation;
static f_perturbationl perturbationl;
static f_period_tri period_tri;
static f_period_jsk period_jsk;
static f_newton newton;
static f_size size;
static f_skew skew;
static f_domain_size domain_size;
//static const char name[];
//static const char source[];
#endif

#define MAGIC ((int)(0xC01dCaf3))
#define SIZE ((int)(sizeof(struct formula)))
#define VERSION 6

struct formula
{
  int magic;
  int ssize;
  int version;
#if 0
  const char *name;
  const char *source;
  double degree;
```

```
115     f_plainf *plainf;
        f_plain *plain;
        f_plainl *plainl;
        f_referencef *referencef;
        f_reference *reference;
120     f_referencel *referencel;
        f_perturbationf *perturbationf;
        f_perturbation *perturbation;
        f_perturbationl *perturbationl;
    #endif
125     f_period_tri *period_tri;
        f_period_jsk *period_jsk;
        f_newton *newton;
        f_size *size;
        f_skew *skew;
130 #if 0
        f_domain_size *domain_size;
    #endif
    };

135 #if 1
    #define FORMULA(name,source,degree) \
    extern "C" { __declspec(dllexport) struct formula et = \
    { MAGIC, SIZE, VERSION \
    , &period_tri, &period_jsk, &newton, &size, &skew \
140 }; }
    #else
    #define FORMULA(name,source,degree) \
    struct formula et = \
    { MAGIC, SIZE, VERSION \
145 , name, source, degree \
    , &plainf, &plain, &plainl \
    , &referencef, &reference, &referencel \
    , &perturbationf, &perturbation, &perturbationl \
    , &period_tri, &period_jsk, &newton, &size, &skew, &domain_size \
150 };
    #endif

    #endif
```

45 formula/generated/.gitignore

```
*.c
```

46 fraktal_sft/CDecNumber.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
```

```
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

    #include "CDecNumber.h"

    #include <iomanip>
    #include <sstream>
    #include <string>

    std::string CDecNumber::ToText() const
    {
            std::ostringstream os;
            os << std::fixed << std::setprecision(m_dec.precision() + 3) << m_dec;
            std::string s = os.str();
            std::size_t e = s.find('e');
            if (e != std::string::npos)
                    s[e] = 'E';
            if (s.find('.') != std::string::npos && e == std::string::npos)
            {
                    std::size_t n = s.length() - 1;
                    while (s[n] == '0')
                            s[n--] = 0;
                    if(s[n] == '.')
                            s[n] = 0;
            }
            else
            if (s[0] == '0' && s[1] == 'E')
            {
                    s[1] = 0;
            }
            else if (e != std::string::npos)
            {
                    int nExp = atoi(s.c_str() + e + 1);
                    if (nExp > 0)
                    {
                            int i;
                            for (i = 0; s[i] && s[i] != '.' && s[i+1] != 'E'; i++)
                                    ;
                            while (nExp){
                                    if (s[i+1] == 'E')
                                    {
                                            if (i == 0)
                                                    i++;
                                            break;
                                    }
                                    s[i] = s[i+1];
                                    nExp--;
                                    i++;
                            }
                            s.replace(s.begin() + i, s.end(), nExp, '0');
                    }
```

```
                e = s.find('E');
                while (e != std::string::npos && e != 0 && (s[e-1] == '0' || s[e
                    -1] == '.'))
                {
                        s.replace(s.begin() + e-1, s.end(), s.begin() + e, s.end
                            ());
                        e--;
                }
        }
        return s;
}
```

47 fraktal_sft/CDecNumber.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_CDECNUMBER_H
#define KF_CDECNUMBER_H

#include <boost/multiprecision/mpfr.hpp>
typedef boost::multiprecision::number<boost::multiprecision::mpfr_float_backend
    <0>> decNumber;

#define LOW_PRECISION 20u

#include <cassert>
#include <string>

class Precision
{
        unsigned digits10;
public:
        inline Precision(unsigned digits10)
        : digits10(digits10)
        {
                decNumber::default_precision(digits10);
        };
        inline ~Precision()
        {
                decNumber::default_precision(digits10);
```

```
        };
};

class CDecNumber
{
public:
        decNumber m_dec;

  inline CDecNumber()
  {
                m_dec.precision(std::max(decNumber::default_precision(), LOW_PRECISION));
                m_dec = 0;
        };
  inline CDecNumber(const CDecNumber &a)
  {
                m_dec.precision(std::max(decNumber::default_precision(), a.m_dec.precision()));
                m_dec = a.m_dec;
        };
  inline CDecNumber(const decNumber &a)
  {
                m_dec.precision(std::max(decNumber::default_precision(), a.precision()));
                m_dec = a;
        };
  inline CDecNumber(const char *a)
  {
                m_dec.precision(std::max(decNumber::default_precision(), unsigned(strlen(a))));
                Precision p(m_dec.precision());
                m_dec = decNumber(a);
        };
  inline CDecNumber(const std::string &a)
  : CDecNumber(a.c_str())
  {
        };
  inline CDecNumber(double a)
  {
                m_dec.precision(std::max(decNumber::default_precision(), LOW_PRECISION));
                m_dec = a;
        };
  inline CDecNumber(int a)
  {
                m_dec.precision(std::max(decNumber::default_precision(), LOW_PRECISION));
                m_dec = a;
        };
  inline ~CDecNumber()
  {
        };

  inline CDecNumber &operator*=(double b)
  {
                m_dec *= b;
                return *this;
```

```
        }

  inline friend CDecNumber operator-(const CDecNumber &a);
  inline friend CDecNumber operator+(const CDecNumber &a, const CDecNumber &b);
  inline friend CDecNumber operator-(const CDecNumber &a, const CDecNumber &b);
  inline friend CDecNumber operator*(const CDecNumber &a, const CDecNumber &b);
  inline friend CDecNumber operator/(const CDecNumber &a, const CDecNumber &b);
  inline friend CDecNumber operator^(const CDecNumber &a, int b);

  inline friend bool operator>(const CDecNumber &a, int b);
  inline friend bool operator<(const CDecNumber &a, int b);
  inline friend bool operator==(const CDecNumber &a, int b);
  inline friend bool operator<(const CDecNumber &a, const CDecNumber &b);

  inline friend CDecNumber sqrt(const CDecNumber &a);

  std::string ToText() const;

  inline int ToInt() const
  {
                return int(m_dec);
        };
};

inline CDecNumber operator-(const CDecNumber &a)
{
        Precision p(std::max(decNumber::default_precision(), a.m_dec.precision()↙
            ↳ ));
        return CDecNumber(-a.m_dec);
}

inline CDecNumber operator+(const CDecNumber &a, const CDecNumber &b)
{
        Precision p(std::max(decNumber::default_precision(), std::max(a.m_dec.↙
            ↳ precision(), b.m_dec.precision())));
        return CDecNumber(a.m_dec + b.m_dec);
}

inline CDecNumber operator-(const CDecNumber &a, const CDecNumber &b)
{
        Precision p(std::max(decNumber::default_precision(), std::max(a.m_dec.↙
            ↳ precision(), b.m_dec.precision())));
        return CDecNumber(a.m_dec - b.m_dec);
}

inline CDecNumber operator*(const CDecNumber &a, const CDecNumber &b)
{
        Precision p(std::max(decNumber::default_precision(), std::max(a.m_dec.↙
            ↳ precision(), b.m_dec.precision())));
        return CDecNumber(a.m_dec * b.m_dec);
}

inline CDecNumber operator*(const double &a, const CDecNumber &b)
{
        Precision p(std::max(decNumber::default_precision(), std::max(53u, b.↙
            ↳ m_dec.precision())));
        return CDecNumber(a * b.m_dec);
```

```
}

inline CDecNumber operator*(const CDecNumber &b, const double &a)
{
        Precision p(std::max(decNumber::default_precision(), std::max(53u, b.↙
            ↳ m_dec.precision())));
        return CDecNumber(b.m_dec * a);
}

inline CDecNumber operator/(const CDecNumber &a, const CDecNumber &b)
{
        Precision p(std::max(decNumber::default_precision(), std::max(a.m_dec.↙
            ↳ precision(), b.m_dec.precision())));
        if (b.m_dec == 0)
                return CDecNumber(0);
        return CDecNumber(a.m_dec / b.m_dec);
}

inline CDecNumber operator^(const CDecNumber &a, int b)
{
        Precision p(std::max(decNumber::default_precision(), a.m_dec.precision()↙
            ↳ ));
        using std::pow;
        return CDecNumber(pow(a.m_dec, b));
}

inline bool operator>(const CDecNumber &a, int b)
{
        return a.m_dec > b;
}

inline bool operator<(const CDecNumber &a, int b)
{
        return a.m_dec < b;
}

inline bool operator==(const CDecNumber &a, int b)
{
        return a.m_dec == b;
}

inline bool operator<(const CDecNumber &a, const CDecNumber &b)
{
        return a.m_dec < b.m_dec;
}

inline CDecNumber sqrt(const CDecNumber &a)
{
        Precision p(std::max(decNumber::default_precision(), a.m_dec.precision()↙
            ↳ ));
        using std::sqrt;
        return CDecNumber(sqrt(a.m_dec));
}

#endif
```

48 fraktal_sft/CFixedFloat.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "CFixedFloat.h"

#include <iomanip>
#include <sstream>
#include <string>

std::string CFixedFloat::ToText() const
{
  std::ostringstream os;
  os << std::fixed << std::setprecision(m_f.precision() + 3) << m_f;
        std::string s = os.str();
        std::size_t e = s.find('e');
        if (e != std::string::npos)
                s[e] = 'E';
        return s;
}
```

49 fraktal_sft/CFixedFloat.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
```

```
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_CFIXEDFLOAT_H
#define KF_CFIXEDFLOAT_H

#include "CDecNumber.h"

typedef decNumber FixedFloat;

class CFixedFloat
{
public:
	FixedFloat m_f;

	inline CFixedFloat()
	{
		unsigned p = FixedFloat::default_precision();
		Precision q(p);
		m_f.precision(p);
		m_f = 0;
	};

	inline CFixedFloat(const CFixedFloat &a)
	{
		unsigned p = a.m_f.precision();
		Precision q(p);
		m_f.precision(p);
		m_f = a.m_f;
	};

	inline CFixedFloat(const FixedFloat &a)
	{
		unsigned p = a.precision();
		Precision q(p);
		m_f.precision(p);
		m_f = a;
	};

	inline CFixedFloat(const char *sz)
	{
		m_f.precision(std::max(FixedFloat::default_precision(), unsigned(strlen(sz))));
		Precision p(m_f.precision());
		m_f = FixedFloat(sz);
	};

	inline CFixedFloat(const std::string &sz)
	{
		m_f.precision(std::max(FixedFloat::default_precision(), unsigned(sz.length())));
		Precision p(m_f.precision());
		m_f = FixedFloat(sz);
	};

	inline CFixedFloat(int a)
	{
```

```
                unsigned p = FixedFloat::default_precision();
                Precision q(p);
                m_f.precision(p);
                m_f = a;
        };

        inline CFixedFloat(double a)
        {
                unsigned p = FixedFloat::default_precision();
                Precision q(p);
                m_f.precision(p);
                m_f = a;
        };

        inline ~CFixedFloat()
        {
        };

        std::string ToText() const;

        inline int ToInt()
        {
                return int(m_f);
        };

        inline double ToDouble(int nScaling = 0)
        {
                using std::pow;
                unsigned p = LOW_PRECISION;
                Precision q(p);
                FixedFloat f;
                f.precision(p);
                f = m_f;
                return double(FixedFloat(f * pow(FixedFloat(10), nScaling)));
        };
        inline long double ToLongDouble(int nScaling = 0)
        {
                using std::pow;
                unsigned p = LOW_PRECISION;
                Precision q(p);
                FixedFloat f;
                f.precision(p);
                f = m_f;
                return (long double)(FixedFloat(f * pow(FixedFloat(10), nScaling↙
                    ↳ )));
        };

        inline CFixedFloat Add(const CFixedFloat &A) const
        {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f + A.m_f));
        };

        inline CFixedFloat Subtract(const CFixedFloat &A) const
        {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f - A.m_f));
```

```
        };

130     inline CFixedFloat Multiply(const CFixedFloat &A) const
        {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f * A.m_f));
        };
135
        inline CFixedFloat Square() const
        {
                Precision p(m_f.precision());
                FixedFloat r(m_f);
140             mpfr_sqr(r.backend().data(), r.backend().data(), MPFR_RNDN);
                return CFixedFloat(r);
        };

        inline CFixedFloat Divide(const CFixedFloat &A) const
145     {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f / A.m_f));
        };

150     inline CFixedFloat &Double()
        {
                mpfr_mul_2ui(m_f.backend().data(), m_f.backend().data(), 1, ↙
                    ↳ MPFR_RNDN);
                return *this;
        };
155
        inline CFixedFloat &AbsAdd(CFixedFloat &a, CFixedFloat &b) // TODO FIXME↙
            ↳ avoid copies
        {
                Precision p(std::max(a.m_f.precision(), b.m_f.precision()));
                using std::abs;
160             m_f = abs(a.m_f) + abs(b.m_f);
                return *this;
        };

        inline CFixedFloat &Abs()
165     {
                mpfr_abs(m_f.backend().data(), m_f.backend().data(), MPFR_RNDN);
                return *this;
        };

170     inline bool operator>(const CFixedFloat &A) const
        {
                return m_f > A.m_f;
        };

175     inline bool operator<(const CFixedFloat &A) const
        {
                return m_f < A.m_f;
        };

180     inline bool operator==(const CFixedFloat &A) const
        {
                return m_f == A.m_f;
```

```
        };

185     inline CFixedFloat &operator=(const CFixedFloat &A)
        {
                m_f.precision(A.m_f.precision());
                m_f = A.m_f;
                return *this;
190     };

        inline CFixedFloat &operator=(const std::string &sz)
        {
                *this = CFixedFloat(sz);
195             return *this;
        };

        inline CFixedFloat &operator=(const char *sz)
        {
200             *this = CFixedFloat(sz);
                return *this;
        };

        inline CFixedFloat &operator=(int a)
205     {
                *this = CFixedFloat(a);
                return *this;
        };

210     inline CFixedFloat &operator=(double a)
        {
                *this = CFixedFloat(a);
                return *this;
        };
215
        inline CFixedFloat operator*(const CFixedFloat &A) const
        {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f * A.m_f));
220     };

        inline CFixedFloat operator/(const CFixedFloat &A) const
        {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
225             return CFixedFloat(FixedFloat(m_f / A.m_f));
        };

        inline CFixedFloat operator+(const CFixedFloat &A) const
        {
230             Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f + A.m_f));
        };

        inline CFixedFloat operator-(const CFixedFloat &A) const
235     {
                Precision p(std::max(m_f.precision(), A.m_f.precision()));
                return CFixedFloat(FixedFloat(m_f - A.m_f));
        };
```

```
240         inline CFixedFloat operator-() const
            {
                    Precision p(m_f.precision());
                    return CFixedFloat(FixedFloat(-m_f));
            };
245
            inline CFixedFloat &operator*=(const CFixedFloat &A)
            {
                    *this = *this * A;
                    return *this;
250         };

            inline CFixedFloat &operator/=(const CFixedFloat &A)
            {
                    *this = *this / A;
255                 return *this;
            };

            inline CFixedFloat &operator+=(const CFixedFloat &A)
            {
260                 *this = *this + A;
                    return *this;
            };

            inline CFixedFloat &operator-=(const CFixedFloat &A)
265         {
                    *this = *this - A;
                    return *this;
            };

270         inline CFixedFloat &operator*=(int A)
            {
                    m_f *= A;
                    return *this;
            };
275
            inline CFixedFloat &operator/=(int A)
            {
                    m_f /= A;
                    return *this;
280         };

            inline CFixedFloat &operator+=(int A)
            {
                    m_f += A;
285                 return *this;
            };

            inline CFixedFloat &operator-=(int A)
            {
290                 m_f -= A;
                    return *this;
            };

            friend class floatexp;
295         friend class floatexp2;
};
```

```
inline CFixedFloat abs(const CFixedFloat &A)
{
        using std::abs;
        Precision p(A.m_f.precision());
        return CFixedFloat(abs(A.m_f));
}
inline CFixedFloat min(const CFixedFloat &A, const CFixedFloat &B)
{
        using std::min;
        Precision p(std::max(A.m_f.precision(), B.m_f.precision()));
        return CFixedFloat(min(A.m_f, B.m_f));
}
inline CFixedFloat max(const CFixedFloat &A, const CFixedFloat &B)
{
        using std::max;
        Precision p(std::max(A.m_f.precision(), B.m_f.precision()));
        return CFixedFloat(max(A.m_f, B.m_f));
}

inline bool operator==(const CFixedFloat &A,int nB)
{
        return A.m_f == nB;
}

inline bool operator==(int nB,const CFixedFloat &A)
{
        return nB == A.m_f;
}

inline bool operator>(const CFixedFloat &A,int nB)
{
        return A.m_f > nB;
}

inline bool operator>(int nB,const CFixedFloat &A)
{
        return nB > A.m_f;
}

inline bool operator>(const CFixedFloat &A,double nB)
{
        return A.m_f > nB;
}

inline bool operator>(double nB,const CFixedFloat &A)
{
        return nB > A.m_f;
}

inline bool operator<(const CFixedFloat &A,int nB)
{
        return A.m_f < nB;
}

inline bool operator<(int nB,const CFixedFloat &A)
{
```

```
        return nB < A.m_f;
}

inline bool operator<(const CFixedFloat &A, double nB)
{
        return A.m_f < nB;
}

inline bool operator<(double nB, const CFixedFloat &A)
{
        return nB < A.m_f;
}

inline CFixedFloat operator*(const CFixedFloat &A, long nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f * nB));
}

inline CFixedFloat operator*(long nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB * A.m_f));
}

inline CFixedFloat operator*(const CFixedFloat &A, int nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f * nB));
}

inline CFixedFloat operator*(int nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB * A.m_f));
}

inline CFixedFloat operator*(const CFixedFloat &A, double nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f * nB));
}

inline CFixedFloat operator*(double nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB * A.m_f));
}

inline CFixedFloat operator/(const CFixedFloat &A, int nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f / nB));
}

inline CFixedFloat operator/(int nB, const CFixedFloat &A)
{
```

```
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB / A.m_f));
}

inline CFixedFloat operator/(const CFixedFloat &A, double nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f / nB));
}

inline CFixedFloat operator/(double nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB / A.m_f));
}

inline CFixedFloat operator+(const CFixedFloat &A, int nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f + nB));
}

inline CFixedFloat operator+(int nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB + A.m_f));
}

inline CFixedFloat operator+(const CFixedFloat &A, double nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f * nB));
}

inline CFixedFloat operator+(double nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB * A.m_f));
}

inline CFixedFloat operator-(const CFixedFloat &A, int nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f * nB));
}

inline CFixedFloat operator-(int nB, const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB * A.m_f));
}

inline CFixedFloat operator-(const CFixedFloat &A, double nB)
{
        Precision p(std::max(A.m_f.precision(), LOW_PRECISION));
        return CFixedFloat(FixedFloat(A.m_f - nB));
}
```

```
inline CFixedFloat operator-(double nB,const CFixedFloat &A)
{
        Precision p(std::max(LOW_PRECISION, A.m_f.precision()));
        return CFixedFloat(FixedFloat(nB - A.m_f));
}

inline CFixedFloat operator^(const CFixedFloat &A,long nB)
{
        using std::pow;
        Precision p(A.m_f.precision());
        return CFixedFloat(FixedFloat(pow(A.m_f, nB)));
}

inline CFixedFloat sqr(const CFixedFloat &A)
{
        return A.Square();
}

#endif
```

50 fraktal_sft/check_for_update.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <windows.h>
#include <tchar.h>
#include <urlmon.h>
#include <wininet.h>

#include <sstream>

#include "check_for_update.h"

std::string CheckForUpdate()
{
  std::string sret = "unknown";
  const auto url = TEXT("https://mathr.co.uk/kf/VERSION.txt");
  const auto prefix = TEXT("KF-VERSION");
  TCHAR dir[MAX_PATH];
```

```
35    DWORD dret = GetTempPath(MAX_PATH, dir);
      if (0 < dret && dret < MAX_PATH)
      {
        TCHAR path[MAX_PATH];
        UINT uret = GetTempFileName(dir, prefix, 0, path);
40      if (uret)
        {
          DeleteUrlCacheEntry(url);
          HRESULT hr = URLDownloadToFile(nullptr, url, path, 0, nullptr);
          if (SUCCEEDED(hr))
45        {
            HANDLE f = CreateFile(path, GENERIC_READ, 0, nullptr, OPEN_EXISTING, ↙
                ↳ FILE_ATTRIBUTE_NORMAL, nullptr);
            if (f != INVALID_HANDLE_VALUE)
            {
#define BUFFERSIZE 64
50            char buffer[BUFFERSIZE];
              DWORD bytes = 0;
              if (ReadFile(f, buffer, BUFFERSIZE, &bytes, nullptr))
              {
                if (bytes < BUFFERSIZE - 1)
55              {
                  buffer[bytes] = 0;
                  int v[5] = { 0, 0, 0, 0, 0 };
                  if (4 == sscanf(buffer, "%d.%d.%d.%d\n", v+0, v+1, v+2, v+3)
                   || 3 == sscanf(buffer, "%d.%d.%d\n", v+0, v+1, v+2))
60                {
                    std::ostringstream os;
                    os << v[0] << "." << v[1] << "." << v[2];
                    if (v[3]) os << "." << v[3];
                    sret = os.str();
65                }
                }
              }
              CloseHandle(f);
            }
70          DeleteFile(path);
            DeleteUrlCacheEntry(url);
          }
        }
      }
75    return sret;
    }
```

51 fraktal_sft/check_for_update.h

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2018 Claude Heiland-Allen
 5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
```

```
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_CHECK_FOR_UPDATE_H
#define KF_CHECK_FOR_UPDATE_H 1

#include <string>

std::string CheckForUpdate();

#endif
```

52 fraktal_sft/cmdline.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "cmdline.h"
#include <vector>
#include <cctype>

LogLevel g_log_level = LogLevel_Status;

static std::vector<std::string> SplitCommandLine(const std::string&commandline)
{
        std::vector<std::string> args;
        bool backslash = false;
        char quote = 0;
        int n = commandline.length();
        std::string arg = "";
        for (int i = 0; i < n; ++i)
        {
                if (!backslash && quote == '\'' && commandline[i] == '\'')
                {
                        quote = 0;
                }
```

```
                else if (!backslash && quote == '"' && commandline[i] == '"')
                {
                        quote = 0;
                }
                else if (!backslash && (commandline[i] == '\'' || commandline[i]
                    ↳ == '"'))
                {
                        quote = commandline[i];
                }
                else if (!backslash && !quote && isspace(commandline[i]))
                {
                        if (i > 0 && !isspace(commandline[i - 1]))
                                args.push_back(arg);
                        arg = "";
                }
                else if (backslash)
                {
                        arg += commandline[i];
                        backslash = false;
                }
                else if (commandline[i] == '\\')
                {
                        backslash = true;
                }
                else
                {
                        arg += commandline[i];
                }
        }
        if (n > 0 && !isspace(commandline[n - 1]))
                args.push_back(arg);
        return args;
}

CommandLineArguments::CommandLineArguments(const std::string &commandline)
: bLoadMap(false)
, sLoadMap("")
, bLoadPalette(false)
, sLoadPalette("")
, bLoadLocation(false)
, sLoadLocation("")
, bLoadSettings(false)
, sLoadSettings("")
, bSaveEXR(false)
, sSaveEXR("")
, bSaveTIF(false)
, sSaveTIF("")
, bSavePNG(false)
, sSavePNG("")
, bSaveJPG(false)
, sSaveJPG("")
, bSaveMap(false)
, sSaveMap("")
, bZoomOut(false)
, nZoomOut(0)
, bVersion(false)
, bHelp(false)
```

```
95  , bError(false)
    {
        std::vector<std::string> args = SplitCommandLine(commandline);
      for (size_t i = 0; i < args.size(); ++i)
      {
100                 if ("-o" == args[i] || "--load-map" == args[i])
                    {
                        ++i;
                        if (i < args.size())
                        {
105                         bLoadMap = true;
                            sLoadMap = args[i];
                        }
                        else
                        {
110                         bError = true;
                        }
                    }
                    else if ("-c" == args[i] || "--load-palette" == args[i])
                    {
115                     ++i;
                        if (i < args.size())
                        {
                            bLoadPalette = true;
                            sLoadPalette = args[i];
120                     }
                        else
                        {
                            bError = true;
                        }
125                 }
                    else if ("-l" == args[i] || "--load-location" == args[i])
                    {
                        ++i;
                        if (i < args.size())
130                     {
                            bLoadLocation = true;
                            sLoadLocation = args[i];
                        }
                        else
135                     {
                            bError = true;
                        }
                    }
                    else if ("-s" == args[i] || "--load-settings" == args[i])
140                 {
                        ++i;
                        if (i < args.size())
                        {
                            bLoadSettings = true;
145                         sLoadSettings = args[i];
                        }
                        else
                        {
                            bError = true;
150                     }
                    }
```

```
		else if ("-x" == args[i] || "--save-exr" == args[i])
		{
			++i;
			if (i < args.size())
			{
				bSaveEXR = true;
				sSaveEXR = args[i];
			}
			else
			{
				bError = true;
			}
		}
		else if ("-t" == args[i] || "--save-tif" == args[i])
		{
			++i;
			if (i < args.size())
			{
				bSaveTIF = true;
				sSaveTIF = args[i];
			}
			else
			{
				bError = true;
			}
		}
		else if ("-p" == args[i] || "--save-png" == args[i])
		{
			++i;
			if (i < args.size())
			{
				bSavePNG = true;
				sSavePNG = args[i];
			}
			else
			{
				bError = true;
			}
		}
		else if ("-j" == args[i] || "--save-jpg" == args[i])
		{
			++i;
			if (i < args.size())
			{
				bSaveJPG = true;
				sSaveJPG = args[i];
			}
			else
			{
				bError = true;
			}
		}
		else if ("-m" == args[i] || "--save-map" == args[i])
		{
			++i;
			if (i < args.size())
			{
```

```
                                bSaveMap = true;
                                sSaveMap = args[i];
                        }
                        else
                        {
                                bError = true;
                        }
                }
                else if ("-z" == args[i] || "--zoom-out" == args[i])
                {
                        ++i;
                        if (i < args.size())
                        {
                                bZoomOut = true;
                                nZoomOut = atoi(args[i].c_str());
                        }
                        else
                        {
                                bError = true;
                        }
                }
                else if ("--log" == args[i])
                {
                        ++i;
                        if (i < args.size())
                        {
                                if ("debug" == args[i]) g_log_level = LogLevel_Debug;
                                else if ("status" == args[i]) g_log_level = LogLevel_Status;
                                else if ("info" == args[i]) g_log_level = LogLevel_Info;
                                else if ("warn" == args[i]) g_log_level = LogLevel_Warn;
                                else if ("error" == args[i]) g_log_level = LogLevel_Error;
                                else bError = true;
                        }
                        else
                        {
                                bError = true;
                        }
                }
                else if ("-v" == args[i] || "-V" == args[i] || "--version" == args[i])
                {
                        bVersion = true;
                }
                else if ("-h" == args[i] || "-H" == args[i] || "--help" == args[i] || "-?" == args[i])
                {
                        bHelp = true;
                }
                else
                {
                        bError = true;
                }
```

```
        }
}

const std::string usage =
"kf.exe [options]\n"
"    -o, --load-map       [FILE.kfb]  load map file\n"
"    -c, --load-palette   [FILE.kfp]  load palette file\n"
"    -l, --load-location  [FILE.kfr]  load location file\n"
"    -s, --load-settings  [FILE.kfs]  load settings file\n"
"    -x, --save-exr       [FILE.exr]  save EXR\n"
"    -t, --save-tif       [FILE.tif]  save TIFF\n"
"    -p, --save-png       [FILE.png]  save PNG\n"
"    -j, --save-jpg       [FILE.jpg]  save JPEG\n"
"    -m, --save-map       [FILE.kfb]  save KFB\n"
"    -z, --zoom-out       [NFRAMES]   zoom sequence\n"
"    --log (debug|status|info|warn|error)\n"
"                                     logging verbosity\n"
"    -v, -V, --version                show version\n"
"    -h, -H, -?, --help               show this help\n"
;

const std::string version = "2.14.8";

// bump these when changing what is saved in KFR/KFS
const int kfr_version_number = 1;
const int kfs_version_number = 1;
```

53 fraktal_sft/cmdline.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_CMDLINE_H
#define KF_CMDLINE_H 1

#include <iostream>
#include <string>

enum LogLevel
{
  LogLevel_Debug = 0,
```

```
  LogLevel_Status = 1,
  LogLevel_Info = 2,
  LogLevel_Warn = 3,
  LogLevel_Error = 4
};

extern LogLevel g_log_level;
#define output_log_message(level, message) \
  do { \
    if((LogLevel_ ## level) >= g_log_level) \
      ((LogLevel_ ## level) >= LogLevel_Status ? std::cerr : std::cout) \
        << message << std::endl; \
  } while(0)

class CommandLineArguments
{
public:
  bool        bLoadMap;
  std::string sLoadMap;
  bool        bLoadPalette;
  std::string sLoadPalette;
  bool        bLoadLocation;
  std::string sLoadLocation;
  bool        bLoadSettings;
  std::string sLoadSettings;
  bool        bSaveEXR;
  std::string sSaveEXR;
  bool        bSaveTIF;
  std::string sSaveTIF;
  bool        bSavePNG;
  std::string sSavePNG;
  bool        bSaveJPG;
  std::string sSaveJPG;
  bool        bSaveMap;
  std::string sSaveMap;
  bool        bZoomOut;
  int         nZoomOut;
  bool        bVersion;
  bool        bHelp;
  bool        bError;
  CommandLineArguments(const std::string &commandline);
};

extern const std::string usage;
extern const std::string version;
extern const int kfr_version_number;
extern const int kfs_version_number;

#endif
```

54 fraktal_sft/colour.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

```

```
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #ifndef KF_COLOUR_H
    #define KF_COLOUR_H 1

    // linear rgb in 0..1

25  struct lrgb
    {
      float r, g, b;
    };

30  inline lrgb blend(const lrgb &a, const lrgb &b, float t)
    {
      float t1 = 1.0f - t;
      lrgb o;
      o.r = a.r * t1 + t * b.r;
35    o.r = a.g * t1 + t * b.g;
      o.r = a.b * t1 + t * b.b;
      return o;
    }

40  // srgb in 0..1

    struct srgb
    {
      float r, g, b;
45  };

    inline float srgb2lrgb1(float s)
    {
      if (s <= 0.04045f)
50      return s / 12.92f;
      return pow((s + 0.055f) / 1.055f, 2.4f);
    }

    inline lrgb srgb2lrgb(const srgb &a)
55  {
      lrgb o;
      o.r = srgb2lrgb1(a.r);
      o.g = srgb2lrgb1(a.g);
      o.b = srgb2lrgb1(a.b);
60    return o;
    }
```

```
inline float lrgb2srgb1(float l)
{
  if (l <= 0.0031308f)
    return l * 12.92f;
  return 1.055f  * pow(l, 1.0f / 2.4f) - 0.055f;
}

inline srgb lrgb2srgb(const lrgb &a)
{
  srgb o;
  o.r = lrgb2srgb1(a.r);
  o.g = lrgb2srgb1(a.g);
  o.b = lrgb2srgb1(a.b);
  return o;
}

inline srgb blend(const srgb &a, const srgb &b, float t)
{
  return lrgb2srgb(blend(srgb2lrgb(a), srgb2lrgb(b), t));
}

// srgb in 0..255

struct srgb8
{
  unsigned char r, g, b;
};

// http://pippin.gimp.org/a_dither/
inline srgb8 dither(const srgb &s, int x, int y)
{
  float mask[3];
  for (int c = 0; c < 3; ++c)
  {
    mask[c] = ((((x + c * 67) + y * 236) * 119) & 255) / 256.0f;
  }
  srgb8 o;
  o.r = floor(255.0f * s.r + mask[0]);
  o.g = floor(255.0f * s.g + mask[1]);
  o.b = floor(255.0f * s.b + mask[2]);
  return o;
}

// hsv in 0..1

struct hsv
{
  float h, s, v;
};

// HSV is a bit of a hack, assume it defines srgb for simplicity
inline srgb hsv2rgb(const hsv &a)
{
        float hue = a.h * 6.0f;
  float sat = a.s;
  float bri = a.v;
        int i = (int) floor(hue);
```

```
        float f = hue - i;
        if (! (i & 1))
                f = 1.0f - f;
        float m = bri * (1.0f - sat);
        float n = bri * (1.0f - sat * f);
    srgb cPos;
        switch (i)
        {
        case 6:
        case 0: cPos.b = bri;
                cPos.g = n;
                cPos.r = m;
                break;
        case 1: cPos.b = n;
                cPos.g = bri;
                cPos.r = m;
                break;
        case 2: cPos.b = m;
                cPos.g = bri;
                cPos.r = n;
                break;
        case 3: cPos.b = m;
                cPos.g = n;
                cPos.r = bri;
                break;
        case 4: cPos.b = n;
                cPos.g = m;
                cPos.r = bri;
                break;
        case 5: cPos.b = bri;
                cPos.g = m;
                cPos.r = n;
        }
    return cPos;
}

#endif
```

55 fraktal_sft/complex.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

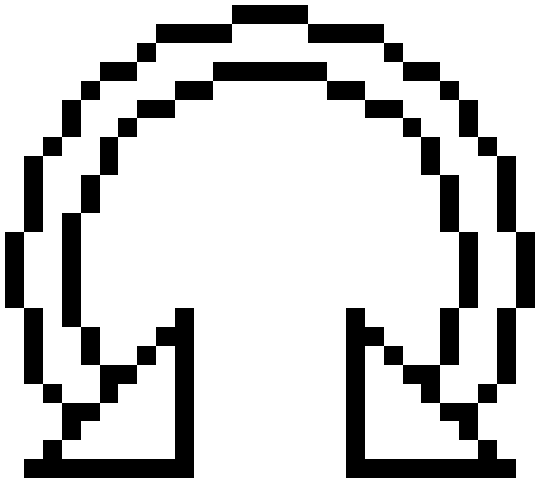
```
*/

#ifndef KF_COMPLEX_H
#define KF_COMPLEX_H

#include "../formula/formula.h"

#include <algorithm>

template <class tt> class complex
{
public:
        tt m_r, m_i;
        inline complex()
        {
                m_r = broadcast<tt>(0.);
                m_i = broadcast<tt>(0.);
        }
        template <typename ss> operator complex<ss>() const
        {
                return complex<ss>(broadcast<ss>(m_r), broadcast<ss>(m_i));
        }
        inline complex(const tt &r)
        {
                m_r = r;
                m_i = broadcast<tt>(0.);
        }
        inline complex(const tt &r, const tt &i)
        {
                m_r = r;
                m_i = i;
        }
        inline complex &operator =(const complex &a)
        {
                m_r = a.m_r;
                m_i = a.m_i;
                return *this;
        }
        inline complex operator *(const complex &a) const
        {
                complex<tt> r;
                r.m_r = m_r*a.m_r - m_i*a.m_i;
                r.m_i = m_r*a.m_i + m_i*a.m_r;
                return r;
        }
        inline complex &operator *=(const complex &a)
        {
                return *this = *this * a;
        }
        inline complex operator +(const int &a) const
        {
                complex<tt> r;
                r.m_r = m_r + a;
                r.m_i = m_i;
                return r;
        }
        inline complex operator +(const complex &a) const
```

```
75          {
                    complex<tt> r;
                    r.m_r = m_r + a.m_r;
                    r.m_i = m_i + a.m_i;
                    return r;
80          }
            inline complex operator -(const complex &a) const
            {
                    complex<tt> r;
                    r.m_r = m_r - a.m_r;
85                  r.m_i = m_i - a.m_i;
                    return r;
            }
            inline complex &operator +=(const complex &a)
            {
90                  m_r += a.m_r;
                    m_i += a.m_i;
                    return *this;
            }
            inline complex &operator -=(const complex &a)
95          {
                    m_r -= a.m_r;
                    m_i -= a.m_i;
                    return *this;
            }
100         inline complex operator ^(int exp) const
            {
                    complex<tt> r;
                    tt tmp;
                    if(exp==0){
105                         r.m_r=broadcast<tt>(1.);
                            r.m_i=broadcast<tt>(0.);
                            return r;
                    }
                    r.m_r = m_r;
110                 r.m_i = m_i;
                    int i;
                    for(i=1;i<exp;i++){
                            tmp = r.m_r*m_r - r.m_i*m_i;
                            r.m_i = r.m_r*m_i + r.m_i*m_r;
115                         r.m_r=tmp;
                    }
                    return r;
            }
            inline complex operator /(const complex &b) const
120         {
                    complex <tt> r;
                    tt div = (b.m_r*b.m_r + b.m_i*b.m_i);
    //              if(!(div==0)){
                            r.m_r = (m_r*b.m_r + m_i*b.m_i)/div;
125                         r.m_i = (m_i*b.m_r - m_r*b.m_i)/div;
    //              }
                    return r;
            }
            inline complex &operator /=(const complex &b)
130         {
                    return *this = *this / b;
```

```
        }
        __inline complex abs()
        {
                complex <tt> r;
                r.m_r=(m_r>0.?m_r:-m_r);
                r.m_i=(m_i>0.?m_i:-m_i);
                return r;
        }
        __inline complex abs_re()
        {
                complex <tt> r;
                r.m_r=(m_r>0.?m_r:-m_r);
                r.m_i=m_i;
                return r;
        }
        __inline complex abs_im()
        {
                complex <tt> r;
                r.m_r=m_r;
                r.m_i=(m_i>0.?m_i:-m_i);
                return r;
        }
        __inline complex re()
        {
                complex <tt> r;
                r.m_r=m_r;
                r.m_i=broadcast<tt>(0.);
                return r;
        }
        __inline complex im()
        {
                complex <tt> r;
                r.m_r=broadcast<tt>(0.);
                r.m_i=m_i;
                return r;
        }
};

template <class tt>
inline complex<tt> operator*(int a, const complex<tt> &b)
{
        return complex<tt>(a * b.m_r, a * b.m_i);
}

template <class tt>
inline complex<tt> operator*(const tt &a, const complex<tt> &b)
{
        return complex<tt>(a * b.m_r, a * b.m_i);
}

template <class tt>
inline complex<tt> operator/(const tt &a, const complex<tt> &b)
{
        return complex<tt>(a) / b;
}

template <typename ss, typename tt>
```

```
inline complex<tt> operator*(const complex<ss> &a, const complex<tt> &b)
{
        return complex<tt>(a) * b;
}

template <class tt>
inline complex<tt> operator-(int a, const complex<tt> &b)
{
        return complex<tt>(tt(a) - b.m_r, -b.m_i);
}

template <class tt>
inline tt norm(const complex<tt> &a)
{
        return a.m_r * a.m_r + a.m_i * a.m_i;
}

template <class tt>
inline tt abs(const complex<tt> &a)
{
        return sqrt(norm(a));
}

template <class tt>
inline complex<tt> sqrt(const complex<tt> &a)
{
        using std::sqrt;
        using std::max;
        tt r = abs(a);
        return complex<tt>(sqrt(max(0.0, (r + a.m_r)*0.5)), ((a.m_i >= 0) - (0 >
            a.m_i)) * sqrt(max(0.0, (r - a.m_r)*0.5)));
}

#endif
```

56 fraktal_sft/cursor1.cur



57 fraktal_sft/double_perturbation.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rumm
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "fraktal_sft.h"
#include "complex.h"
#include "../formula/formula.h"

void CFraktalSFT::MandelCalc()
{
  m_bIterChanged = TRUE;
  double yr, yi, dr, di;
  long double ldr = 0, ldi = 0;
  double epsilon(m_epsilon);
  int x, y, w, h;
```

```
  int64_t antal;

  // vectorization
  double16 Dr16, Di16, dbD0r16, dbD0i16, test116, test216, dr16, di16, daa16, ↙
      ↳ dab16, dba16, dbb16;
35  int16 antal16, bGlitch16, x16, y16, w16, h16;
  int k = 0;
  const int64_t chunksize = GetSIMDChunkSize();
  const int vectorsize = GetSIMDVectorSize();
  const bool derivatives = GetDerivatives();
40  const bool vectorized = (derivatives ? ! m_nScalingOffset : true) && (↙
      ↳ m_nFractalType == 0 ? ! (m_nPower > 10) : true) && vectorsize > 1;

  int64_t nMaxIter = (m_nGlitchIter<m_nMaxIter ? m_nGlitchIter : m_nMaxIter);
  while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
    int64_t nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
45    if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
      SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, w, h);
      continue;
    }
    if (GuessPixel(x, y, w, h))
50      continue;

    // Series approximation
    floatexp D0r = 0;
    floatexp D0i = 0;
55    floatexp daa0 = 1;
    floatexp dab0 = 0;
    floatexp dba0 = 0;
    floatexp dbb0 = 1;
    GetPixelCoordinates(x, y, D0r, D0i, daa0, dab0, dba0, dbb0);
60
    floatexp TDnr;
    floatexp TDni;
    floatexp dxa1, dxb1, dya1, dyb1;
    DoApproximation(antal, D0r, D0i, TDnr, TDni, dxa1, dxb1, dya1, dyb1);
65    floatexp TDDnr = dxa1;
    floatexp TDDni = dya1;

    double test1 = 0, test2 = 0;
    bool bGlitch = false;
70
    double dbD0r = D0r.todouble();
    double dbD0i = D0i.todouble();
    double Dr = TDnr.todouble();
    double Di = TDni.todouble();
75    dr = TDDnr.todouble();
    di = TDDni.todouble();

    if (m_nFractalType == 0 && m_nPower > 10)
    {
80      if (derivatives)
      {
        complex<double> d(dr, di);
        bool no_g = g_real == 1.0 && g_imag == 1.0;
        if (antal<nMaxIter && test1 <= m_nBailout2){
85        for (; antal<nMaxIter; antal++){
```

```
            yr = m_db_dxr[antal] + Dr;
            yi = m_db_dxi[antal] + Di;
            test2 = test1;
            if (no_g)
              test1 = yr*yr + yi*yi;
            else
              test1 = g_real*yr*yr + g_imag*yi*yi;
            if (test1<m_db_z[antal]){
              bGlitch = TRUE;
              if (! m_bNoGlitchDetection)
                break;
            }
            if (test1 > m_nBailout2)
            {
              break;
            }
            complex<double> yri(yr, yi);
            d = m_nPower * d * (yri ^ (m_nPower - 1)) + 1;
            complex<double> X(m_db_dxr[antal], m_db_dxi[antal]);
            complex<double> D(Dr, Di);
            complex<double> D0(dbD0r, dbD0i);
            complex<double> c(m_pnExpConsts[0], 0);
            int nXExp = m_nPower - 2, nDExp = 2, ci = 1;
            complex<double> Dn = c*(X^(m_nPower - 1))*D;
            while (nXExp){
              c.m_r = m_pnExpConsts[ci++];
              Dn += c*(X^nXExp)*(D^nDExp);
              nXExp--;
              nDExp++;
            }
            Dn += (D^m_nPower) + D0;
            Di = Dn.m_i;
            Dr = Dn.m_r;
          }
        }
        dr = d.m_r;
        di = d.m_i;
      }
      else
      {
        if (antal<nMaxIter && test1 <= m_nBailout2){
          bool no_g = g_real == 1.0 && g_imag == 1.0;
          for (; antal<nMaxIter; antal++){
            yr = m_db_dxr[antal] + Dr;
            yi = m_db_dxi[antal] + Di;
            test2 = test1;
            if (no_g)
              test1 = yr*yr + yi*yi;
            else
              test1 = g_real*yr*yr + g_imag*yi*yi;
            if (test1<m_db_z[antal]){
              bGlitch = TRUE;
              if (! m_bNoGlitchDetection)
                break;
            }
            if (test1 > m_nBailout2)
            {
```

```
            break;
          }
          complex<double> yri(yr, yi);
          complex<double> X(m_db_dxr[antal], m_db_dxi[antal]);
          complex<double> D(Dr, Di);
          complex<double> D0(dbD0r, dbD0i);
          complex<double> c(m_pnExpConsts[0], 0);
          int nXExp = m_nPower - 2, nDExp = 2, ci = 1;
          complex<double> Dn = c*(X^(m_nPower - 1))*D;
          while (nXExp){
            c.m_r = m_pnExpConsts[ci++];
            Dn += c*(X^nXExp)*(D^nDExp);
            nXExp--;
            nDExp++;
          }
          Dn += (D^m_nPower) + D0;
          Di = Dn.m_i;
          Dr = Dn.m_r;
        }
      }
    }
    dr *= m_lPixelSpacing;
    di *= m_lPixelSpacing;

  }
  else
  {
    if (vectorized)
    {
      x16[k] = x;
      y16[k] = y;
      w16[k] = w;
      h16[k] = h;
      Dr16[k] = m_nScalingOffset ? TDnr.todouble(m_nScalingOffset) : Dr;
      Di16[k] = m_nScalingOffset ? TDni.todouble(m_nScalingOffset) : Di;
      dbD0r16[k] = m_nScalingOffset ? D0r.todouble(m_nScalingOffset) : dbD0r;
      dbD0i16[k] = m_nScalingOffset ? D0i.todouble(m_nScalingOffset) : dbD0i;
      antal16[k] = antal;
      test116[k] = test1;
      test216[k] = test2;
      bGlitch16[k] = bGlitch;
      daa16[k] = daa0.todouble();
      dab16[k] = dab0.todouble();
      dba16[k] = dba0.todouble();
      dbb16[k] = dbb0.todouble();
      dr16[k] = dr * m_dPixelSpacing;
      di16[k] = di * m_dPixelSpacing;
      k = k + 1;
      if (k == vectorsize)
      {
        bool ok = false;
#define GO(intN,doubleN) \
          intN antalv, bGlitchv; \
          doubleN test1v, test2v, Drv, Div, dbD0rv, dbD0iv, drv, div, daav, ↙
              ↳ dabv, dbav, dbbv; \
          for (int q = 0; q < vectorsize; ++q) \
          { \
```

```
                antalv[q] = antal16[q]; \
                bGlitchv[q] = bGlitch16[q]; \
                test1v[q] = test116[q]; \
                test2v[q] = test216[q]; \
                Drv[q] = Dr16[q]; \
                Div[q] = Di16[q]; \
                dbD0rv[q] = dbD0r16[q]; \
                dbD0iv[q] = dbD0i16[q]; \
                if (derivatives) \
                { \
                  drv[q] = dr16[q]; \
                  div[q] = di16[q]; \
                  daav[q] = daa16[q]; \
                  dabv[q] = dab16[q]; \
                  dbav[q] = dba16[q]; \
                  dbbv[q] = dbb16[q]; \
                } \
              } \
              ok = m_nScalingOffset \
                ? perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                    ↳ m_db_z, antalv, test1v, test2v, bGlitchv, m_nBailout2, ↙
                    ↳ nMaxIter, m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, ↙
                    ↳ g_FactorAI, Drv, Div, dbD0rv, dbD0iv, chunksize, m_nScaling,↙
                    ↳  1 / m_nScaling) \
                : derivatives \
                ? perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                    ↳ m_db_z, antalv, test1v, test2v, bGlitchv, m_nBailout2, ↙
                    ↳ nMaxIter, m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, ↙
                    ↳ g_FactorAI, Drv, Div, dbD0rv, dbD0iv, drv, div, epsilon, ↙
                    ↳ m_dPixelSpacing, daav, dabv, dbav, dbbv, chunksize) \
                : perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                    ↳ m_db_z, antalv, test1v, test2v, bGlitchv, m_nBailout2, ↙
                    ↳ nMaxIter, m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, ↙
                    ↳ g_FactorAI, Drv, Div, dbD0rv, dbD0iv, chunksize) \
                ; \
              for (int q = 0; q < vectorsize; ++q) \
              { \
                antal16[q] = antalv[q]; \
                bGlitch16[q] = bGlitchv[q]; \
                test116[q] = test1v[q]; \
                test216[q] = test2v[q]; \
                Dr16[q] = Drv[q]; \
                Di16[q] = Div[q]; \
                dbD0r16[q] = dbD0rv[q]; \
                dbD0i16[q] = dbD0iv[q]; \
                if (derivatives) \
                { \
                  dr16[q] = drv[q]; \
                  di16[q] = div[q]; \
                  daa16[q] = daav[q]; \
                  dab16[q] = dabv[q]; \
                  dba16[q] = dbav[q]; \
                  dbb16[q] = dbbv[q]; \
                } \
              }
#if KF_SIMD >= 1
            if (vectorsize == 2)
```

```
245             {
                  GO(int2, double2)
                   assert(ok && "perturbation_double2");
                }
    #endif
250 #if KF_SIMD >= 2
                if (vectorsize == 4)
                {
                  GO(int4, double4)
                   assert(ok && "perturbation_double4");
255             }
    #endif
    #if KF_SIMD >= 3
                if (vectorsize == 8)
                {
260               GO(int8, double8)
                   assert(ok && "perturbation_double8");
                }
    #endif
    #if KF_SIMD >= 4
265             if (vectorsize == 16)
                {
                  GO(int16, double16)
                   assert(ok && "perturbation_double16");
                }
270 #endif
    #undef GO
                assert(ok && "valid vectorsize");
              }
            }
275         else
            {
              if (m_nScalingOffset)
              {
                Dr = TDnr.todouble(m_nScalingOffset);
280             Di = TDni.todouble(m_nScalingOffset);
                dbD0r = D0r.todouble(m_nScalingOffset);
                dbD0i = D0i.todouble(m_nScalingOffset);
                ldr = TDDnr.toLongDouble();
                ldi = TDDni.toLongDouble();
285             long double daa = daa0.toLongDouble();
                long double dab = dab0.toLongDouble();
                long double dba = dba0.toLongDouble();
                long double dbb = dbb0.toLongDouble();
                ldr *= m_lPixelSpacing;
290             ldi *= m_lPixelSpacing;
                bool ok = derivatives
                  ? perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z,↙
                      ↳  antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
                      ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, ↙
                      ↳ Dr, Di, dbD0r, dbD0i, ldr, ldi, (long double)(m_epsilon), ↙
                      ↳ m_lPixelSpacing, daa, dab, dba, dbb, m_nScaling, 1 / ↙
                      ↳ m_nScaling)
                  : perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z,↙
                      ↳  antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
                      ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, ↙
                      ↳ Dr, Di, dbD0r, dbD0i, m_nScaling, 1 / m_nScaling)
```

```
            ;
295         assert(ok && "perturbation_double_scaled");
            dr = ldr;
            di = ldi;
          }
          else
          {
            double daa = daa0.todouble();
            double dab = dab0.todouble();
            double dba = dba0.todouble();
            double dbb = dbb0.todouble();
305         dr *= m_dPixelSpacing;
            di *= m_dPixelSpacing;
            bool ok = derivatives
              ? perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z,↙
                  ↳ antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
                  ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, ↙
                  ↳ Dr, Di, dbD0r, dbD0i, dr, di, epsilon, m_dPixelSpacing, daa, ↙
                  ↳ dab, dba, dbb)
              : perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z,↙
                  ↳ antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
                  ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, ↙
                  ↳ Dr, Di, dbD0r, dbD0i)
310           ;
            assert(ok && "perturbation_double");
          }
        }
      }
315
      if (vectorized)
      {
        if (k == vectorsize)
        {
320       for (k = 0; k < vectorsize; ++k)
          {
            complex<double> z(Dr16[k], Di16[k]);
            complex<double> dc(dr16[k], di16[k]);
            complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;
325         OutputIterationData(x16[k], y16[k], w16[k], h16[k], bGlitch16[k], ↙
                ↳ antal16[k], test116[k], test216[k], de);
            InterlockedIncrement((LPLONG)&m_nDone);
            OutputPixelData(x16[k], y16[k], w16[k], h16[k], bGlitch16[k]);
          }
          k = 0;
330     }
      }
      else
      {
        complex<double> z(Dr, Di);
335     complex<double> dc(dr, di);
        complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;
        OutputIterationData(x, y, w, h, bGlitch, antal, test1, test2, de);
        InterlockedIncrement((LPLONG)&m_nDone);
        OutputPixelData(x, y, w, h, bGlitch);
340   }
    }
    if (vectorized)
```

```
    {
      int leftover = k;
345   for (k = 0; k < leftover; ++k)
      {
        x = x16[k];
        y = y16[k];
        w = w16[k];
350     h = h16[k];
        antal = antal16[k];
        double test1 = test116[k];
        double test2 = test216[k];
        bool bGlitch = bGlitch16[k];
355     double Dr = Dr16[k];
        double Di = Di16[k];
        double dbD0r = dbD0r16[k];
        double dbD0i = dbD0i16[k];
        double dr = dr16[k];
360     double di = di16[k];
        double daa = daa16[k];
        double dab = dab16[k];
        double dba = dba16[k];
        double dbb = dbb16[k];
365     bool ok = m_nScalingOffset
          ? perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z, ↙
              ↳ antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
              ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, Dr, ↙
              ↳ Di, dbD0r, dbD0i, m_nScaling, 1 / m_nScaling)
          : derivatives
          ? perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z, ↙
              ↳ antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
              ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, Dr, ↙
              ↳ Di, dbD0r, dbD0i, dr, di, epsilon, m_dPixelSpacing, daa, dab, dba,↙
              ↳  dbb)
          : perturbation(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, m_db_z, ↙
              ↳ antal, test1, test2, bGlitch, m_nBailout2, nMaxIter, ↙
              ↳ m_bNoGlitchDetection, g_real, g_imag, g_FactorAR, g_FactorAI, Dr, ↙
              ↳ Di, dbD0r, dbD0i)
370       ;
        assert(ok && "perturbation_double");
        complex<double> z(Dr16[k], Di16[k]);
        complex<double> dc(dr16[k], di16[k]);
        complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;
375     OutputIterationData(x, y, w, h, bGlitch, antal, test1, test2, de);
        InterlockedIncrement((LPLONG)&m_nDone);
        OutputPixelData(x, y, w, h, bGlitch);
      }
    }
380 }
```

58 fraktal_sft/double_reference.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
```

```
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "complex.h"
    #include "../formula/formula.h"

    void CFraktalSFT::CalculateReference()
25  {
        const bool derivatives = GetDerivatives();
        Precision prec(m_rref.m_f.precision());

        int64_t i;
30      if (m_db_dxr)
            delete[] m_db_dxr;
        m_db_dxr = new double[m_nMaxIter];
        if (m_db_dxi)
            delete[] m_db_dxi;
35      m_db_dxi = new double[m_nMaxIter];
        if (m_db_z)
            delete[] m_db_z;
        m_db_z = new double[m_nMaxIter];

40      CFixedFloat xr = g_SeedR, xi = g_SeedI, xin, xrn, sr = xr.Square(), si =↙
            ↳ xi.Square(), xrxid = 0;
        double terminate = SMOOTH_BAILOUT*SMOOTH_BAILOUT;

        if (m_Inflections.size() > 0)
        {
45          int inf;
            complex<CFixedFloat> c(m_rref,m_iref);
            for(inf=m_Inflections.size()-1;inf>=0;inf--){
                complex<CFixedFloat> d = c-m_Inflections[inf];
                c=m_Inflections[inf]+d*d;
50          }
            m_rref=c.m_r;
            m_iref=c.m_i;
        }

55      m_nGlitchIter = m_nMaxIter + 1;
        int64_t nMaxIter = m_nMaxIter;

        int64_t antal = 0;
        double test1 = 0;
60      double test2 = 0;
        double xxr = 0, xxi = 0;
```

```
	double dr = 1, di = 0;
	long double ldr = 1, ldi = 0;

	if (m_nFractalType == 0 && m_nPower > 10) // FIXME matrix derivatives, option to disable derivatives
	{

		double threashold = 0.0001;
		for (i = 7; i <= m_nPower; i += 2)
			threashold *= 10;
		if (GetGlitchLowTolerance()) {
			threashold = sqrt(threashold);
		}
		if (threashold>.5)
			threashold = .5;
		complex<CFixedFloat> r(m_rref, m_iref);
		complex<CFixedFloat> X(g_SeedR, g_SeedI);
		complex<double> d(1.0, 0.0);
		bool stored = false;
		double old_absval = 0;
		double abs_val = 0;
		for (i = 0; i<nMaxIter && !m_bStop; i++){
			complex<double> x(X.m_r.ToDouble(), X.m_i.ToDouble());
			d = m_nPower * d * (x^(m_nPower - 1)) + 1;
			X = (X^m_nPower) + r;
			m_db_dxr[i] = X.m_r.ToDouble();
			m_db_dxi[i] = X.m_i.ToDouble();
			old_absval = abs_val;
			abs_val = (g_real * m_db_dxr[i] * m_db_dxr[i] + g_imag * m_db_dxi[i] * m_db_dxi[i]);
			m_db_z[i] = abs_val*threashold;
			if (abs_val >= 4)
			{
				if (terminate == 4 && !stored)
				{
					stored = true;
					antal = i;
					test1 = abs_val;
					test2 = old_absval;
				}
			}
			if (abs_val >= terminate){
				if (terminate > 4 && !stored)
				{
					stored = true;
					antal = i;
					test1 = abs_val;
					test2 = old_absval;
				}
				if (nMaxIter == m_nMaxIter){
					nMaxIter = i + 3;
					if (nMaxIter>m_nMaxIter)
						nMaxIter = m_nMaxIter;
					m_nGlitchIter = nMaxIter;
				}
			}
			m_nRDone++;
```

```
                }
                dr = d.m_r;
                di = d.m_i;
                ldr = dr * m_lPixelSpacing;
                ldi = di * m_lPixelSpacing;

        }
        else if (m_nScalingOffset && scaling_supported(m_nFractalType, m_nPower,↙
            ↳ derivatives))
        {

                floatexp _x, _y, daa, dab, dba, dbb;
                GetPixelCoordinates(g_nAddRefX, g_nAddRefY, _x, _y, daa, dab, ↙
                    ↳ dba, dbb);
                long double ddaa = daa.todouble();
                long double ddab = dab.todouble();
                long double ddba = dba.todouble();
                long double ddbb = dbb.todouble();
                ldr *= m_lPixelSpacing;
                ldi *= m_lPixelSpacing;
                bool ok = derivatives
                  ? reference(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                      ↳ m_db_z, m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, ↙
                      ↳ m_rref, m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI↙
                      ↳ , terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                      ↳ antal, test1, test2, xxr, xxi, ldr, ldi, ddaa, ddab, ↙
                      ↳ ddba, ddbb)
                  : reference(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                      ↳ m_db_z, m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, ↙
                      ↳ m_rref, m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI↙
                      ↳ , terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                      ↳ antal, test1, test2, xxr, xxi)
                  ;
                assert(ok && "reference_scaled_double");

        }
        else
        {

                floatexp _x, _y, daa, dab, dba, dbb;
                GetPixelCoordinates(g_nAddRefX, g_nAddRefY, _x, _y, daa, dab, ↙
                    ↳ dba, dbb);
                double ddaa = daa.todouble();
                double ddab = dab.todouble();
                double ddba = dba.todouble();
                double ddbb = dbb.todouble();
                dr *= m_dPixelSpacing;
                di *= m_dPixelSpacing;
                bool ok = derivatives
                  ? reference(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                      ↳ m_db_z, m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, ↙
                      ↳ m_rref, m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI↙
                      ↳ , terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                      ↳ antal, test1, test2, xxr, xxi, dr, di, ddaa, ddab, ddba,↙
                      ↳  ddbb)
                  : reference(m_nFractalType, m_nPower, m_db_dxr, m_db_dxi, ↙
                      ↳ m_db_z, m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, ↙
```

```
                        ↪ m_rref, m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI↙
                        ↪ , terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                        ↪ antal, test1, test2, xxr, xxi)
                    ;
                assert(ok && "reference_double");
                ldr = dr;
                ldi = di;

        }

    complex<double> z((double(xxr)), (double(xxi)));
    complex<double> dc((double(ldr)), (double(ldi)));
    complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;

        if (0 <= g_nAddRefX && g_nAddRefX < m_nX && 0 <= g_nAddRefY && ↙
            ↪ g_nAddRefY < m_nY)
                OutputIterationData(g_nAddRefX, g_nAddRefY, 1, 1, false, antal ?↙
                    ↪  antal + 1 : m_nMaxIter, test1, test2, de);
}
```

59 fraktal_sft/exr.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <ImfNamespace.h>
#include <ImfMultiPartInputFile.h>
#include <ImfInputFile.h>
#include <ImfOutputFile.h>
#include <ImfHeader.h>
#include <ImfIntAttribute.h>
#include <ImfStringAttribute.h>
#include <ImfChannelList.h>
#include <ImfArray.h>
#include <ImfFrameBuffer.h>
#include <ImathBox.h>
#include <ImfPreviewImage.h>

#include <iostream>
#include <cstdlib>

```

```
#include "fraktal_sft.h"
#include "exr.h"

namespace IMF = OPENEXR_IMF_NAMESPACE;
using namespace IMF;
using namespace IMATH_NAMESPACE;

const char magic[] = "KallesFraktaler2+";
const int64_t BIAS = 1024;

extern int SaveEXR
( const std::string &filename
, const unsigned char *Data
, int nWidth
, int nHeight
, int nColors
, const std::string &comment
, int64_t maxiter
, int arrWidth
, int arrHeight
, const itercount_array &count
, const float *trans
, const float *rawdex
, const float *rawdey
)
{
  try
  {
    // prepare arrays with proper format
    uint32_t *n0 = new uint32_t[size_t(arrWidth) * arrHeight];
    uint32_t *n1 = (maxiter + BIAS >= 0xFFffFFffLL) ? new uint32_t[size_t(↙
        ↪ arrWidth) * arrHeight] : nullptr;
    float *nf = new float[size_t(arrWidth) * arrHeight];
    float *dex = rawdex ? new float[size_t(arrWidth) * arrHeight] : nullptr;
    float *dey = rawdey ? new float[size_t(arrWidth) * arrHeight] : nullptr;
    // TODO parallelize
    for (int i = 0; i < arrWidth; ++i)
    {
      for (int j = 0; j < arrHeight; ++j)
      {
        // [x][y], don't flip vertically!
        size_t k = j + arrHeight * i;
        size_t e = j + arrHeight * i;
        if (PIXEL_UNEVALUATED == count[i][j] || GET_TRANS_GLITCH(trans[k]))
        {
          // not calculated, or glitched
          if (n1) n1[e] = 0;
          n0[e] = 0;
          nf[e] = 0.0f;
          if (dex) dex[e] = 0.0f;
          if (dey) dey[e] = 0.0f;
        }
        else if (count[i][j] == maxiter)
        {
          // unescaped
          if (n1) n1[e] = 0xFFffFFffU;
          n0[e] = 0xFFffFFffU;
```

```
            nf[e] = 0.0f;
            if (dex) dex[e] = 0.0f;
            if (dey) dey[e] = 0.0f;
          }
          else
          {
            int64_t n = count[i][j] + BIAS;
            if (n1) n1[e] = (n >> 32) & 0xFFffFFffLL;
            n0[e] = n & 0xFFffFFffLL;
            nf[e] = 1.0f - trans[k];
            if (dex) dex[e] = rawdex[k];
            if (dey) dey[e] = rawdey[k];
          }
        }
      }
      // prepare preview image
      Header header(arrWidth, arrHeight);
      Array2D<PreviewRgba> preview;
      preview.resizeErase(nHeight, nWidth);
      // TODO parallelize
      for (int j = 0; j < nHeight; ++j)
      {
        for (int i = 0; i < nWidth; ++i)
        {
          size_t k = (j * size_t(nWidth) + i) * 3;
          PreviewRgba &o = preview[j][i];
          o.r = Data[k + 0];
          o.g = Data[k + 1];
          o.b = Data[k + 2];
          o.a = 255;
        }
      }
      header.setPreviewImage(PreviewImage(nWidth, nHeight, &preview[0][0]));
      // insert metadata
      header.insert(magic, StringAttribute(comment));
      header.insert("Iterations", IntAttribute(maxiter));
      header.insert("IterationsBias", IntAttribute(BIAS));
      // write image
      const half *rgb = g_SFT.GetArrayHalfColour();
      if (rgb)
      {
        header.channels().insert("R", Channel(IMF::HALF));
        header.channels().insert("G", Channel(IMF::HALF));
        header.channels().insert("B", Channel(IMF::HALF));
      }
      if (n1)
      {
        header.channels().insert("N0",  Channel(IMF::UINT));
        header.channels().insert("N1",  Channel(IMF::UINT));
      }
      else
        header.channels().insert("N",  Channel(IMF::UINT));
      header.channels().insert("NF", Channel(IMF::FLOAT));
      if (dex) header.channels().insert("DEX", Channel(IMF::FLOAT));
      if (dey) header.channels().insert("DEY", Channel(IMF::FLOAT));
      OutputFile of(filename.c_str(), header);
      FrameBuffer fb;
```

```
    if (rgb)
    {
      // [y][x]
      size_t row = g_SFT.GetArrayHalfColourStride();
      fb.insert("R", Slice(IMF::HALF, (char *) (rgb + 0), sizeof(*rgb) * 3, sizeof(*rgb) * row));
      fb.insert("G", Slice(IMF::HALF, (char *) (rgb + 1), sizeof(*rgb) * 3, sizeof(*rgb) * row));
      fb.insert("B", Slice(IMF::HALF, (char *) (rgb + 2), sizeof(*rgb) * 3, sizeof(*rgb) * row));
    }
    else
    {
      std::cerr << "no rgb" << std::endl;
    }
    // [x][y]
    if (n1)
    {
      fb.insert("N0", Slice(IMF::UINT,  (char *) n0, sizeof(*n0) * arrHeight, sizeof(*n0) * 1));
      fb.insert("N1", Slice(IMF::UINT,  (char *) n1, sizeof(*n1) * arrHeight, sizeof(*n1) * 1));
    }
    else
      fb.insert("N",  Slice(IMF::UINT,  (char *) n0, sizeof(*n0) * arrHeight, sizeof(*n0) * 1));
    fb.insert("NF", Slice(IMF::FLOAT, (char *) nf, sizeof(*nf) * arrHeight, sizeof(*nf) * 1));
    if (dex) fb.insert("DEX", Slice(IMF::FLOAT, (char *) dex, sizeof(*dex) * arrHeight, sizeof(*dex) * 1));
    if (dey) fb.insert("DEY", Slice(IMF::FLOAT, (char *) dey, sizeof(*dey) * arrHeight, sizeof(*dey) * 1));
    of.setFrameBuffer(fb);
    of.writePixels(arrHeight);
    delete[] nf;
    delete[] n0;
    delete[] n1;
    delete[] dex;
    delete[] dey;
    return 1;
  }
  catch (...)
  {
    return 0;
  }
}

extern std::string ReadEXRComment(const std::string &filename)
{
  try
  {
    MultiPartInputFile in(filename.c_str());
    for (int p = 0; p < in.parts(); ++p)
    {
      const Header &h = in.header(p);
      for (Header::ConstIterator i = h.begin(); i != h.end(); ++i)
      {
```

```
        std::string name(i.name());
        if (name == magic)
        {
200       const Attribute *a = &i.attribute();
          if (const StringAttribute *s = dynamic_cast<const StringAttribute *>(a↙
              ↳ ))
          {
            return std::string(s->value());
          }
205     }
      }
    }
    catch (...)
210 {
      return "";
    }
    return "";
  }
215
  extern bool ReadEXRMapFile(const std::string &filename)
  {
    bool retval = false;
    uint32_t *N0 = nullptr;
220 uint32_t *N1 = nullptr;
    float *NF = nullptr;
    float *DEX = nullptr;
    float *DEY = nullptr;
    try
225 {
      InputFile file(filename.c_str());
      int64_t maxiter = INT_MAX - 1 - BIAS;
      int64_t bias = BIAS;
      const Header &h = file.header();
230   for (Header::ConstIterator i = h.begin(); i != h.end(); ++i)
      {
        std::string name(i.name());
        if (name == "Iterations")
        {
235       const Attribute *a = &i.attribute();
          if (const IntAttribute *s = dynamic_cast<const IntAttribute *>(a))
          {
            maxiter = s->value();
          }
240       if (const StringAttribute *s = dynamic_cast<const StringAttribute *>(a))
          {
            maxiter = atoll(s->value().c_str());
          }
        }
245     if (name == "IterationsBias")
        {
          const Attribute *a = &i.attribute();
          if (const IntAttribute *s = dynamic_cast<const IntAttribute *>(a))
          {
250         bias = s->value();
          }
          if (const StringAttribute *s = dynamic_cast<const StringAttribute *>(a))
```

```
        {
          bias = atoll(s->value().c_str());
        }
      }
    }
    Box2i dw = h.dataWindow();
    size_t width = dw.max.x - dw.min.x + 1;
    size_t height = dw.max.y - dw.min.y + 1;
    // prepare arrays with proper format
    N0 = new uint32_t[width * height];
    N1 = maxiter + bias >= 0xFFffFFffLL ? new uint32_t[width * height] : nullptr;
    NF = new float[width * height];
    DEX = new float[width * height];
    DEY = new float[width * height];
    FrameBuffer fb;
    // [x][y]
    if (N1)
    {
      fb.insert("N0", Slice(IMF::UINT, (char *) (&N0[0] - dw.min.x - dw.min.y * width), sizeof(N0[0]) * height, sizeof(N0[0]) * 1, 1, 1, 0));
      fb.insert("N1", Slice(IMF::UINT, (char *) (&N1[0] - dw.min.x - dw.min.y * width), sizeof(N1[0]) * height, sizeof(N1[0]) * 1, 1, 1, 0));
    }
    else
      fb.insert("N" , Slice(IMF::UINT, (char *) (&N0[0] - dw.min.x - dw.min.y * width), sizeof(N0[0]) * height, sizeof(N0[0]) * 1, 1, 1, 0));
    fb.insert("NF", Slice(IMF::FLOAT, (char *) (&NF[0] - dw.min.x - dw.min.y * width), sizeof(NF[0]) * height, sizeof(NF[0]) * 1, 1, 1, 0.0f));
    fb.insert("DEX", Slice(IMF::FLOAT, (char *) (&DEX[0] - dw.min.x - dw.min.y * width), sizeof(DEX[0]) * height, sizeof(DEX[0]) * 1, 1, 1, 0.0f));
    fb.insert("DEY", Slice(IMF::FLOAT, (char *) (&DEY[0] - dw.min.x - dw.min.y * width), sizeof(DEY[0]) * height, sizeof(DEY[0]) * 1, 1, 1, 0.0f));
    file.setFrameBuffer(fb);
    file.readPixels(dw.min.y, dw.max.y);
    // copy to KF arrays
    g_SFT.SetIterations(maxiter);
    g_SFT.SetImageSize(width, height);
    itercount_array count = g_SFT.GetArrayCount();
    float *trans = g_SFT.GetArrayTrans();
    float *dex = g_SFT.GetArrayDEx();
    float *dey = g_SFT.GetArrayDEy();
    // TODO parallelize
    for (size_t i = 0; i < width; ++i)
    {
      for (size_t j = 0; j < height; ++j)
      {
        // [x][y], don't flip vertically!
        size_t e = j + height * i;
        size_t k = j + height * i;
        if ((N1 ? N1[e] == 0 : true) && N0[e] == 0 && NF[e] == 0.0f)
        {
          count[i][j] = PIXEL_UNEVALUATED;
          trans[k] = 0.0f;
          if (dex) dex[k] = 0.0f;
          if (dey) dey[k] = 0.0f;
        }
```

```
          else if ((N1 ? N1[e] == 0xFFffFFffU : true) && N0[e] == 0xFFffFFffU)
          {
305         count[i][j] = maxiter;
            trans[k] = 0.0f;
            if (dex) dex[k] = 0.0f;
            if (dey) dey[k] = 0.0f;
          }
310       else
          {
            count[i][j] = ((N1 ? int64_t(N1[e]) << 32 : 0) | N0[e]) - bias;
            trans[k] = 1.0f - NF[e];
            if (dex) dex[k] = DEX[e];
315         if (dey) dey[k] = DEY[e];
          }
        }
      }
      g_SFT.ReinitializeBitmap();
320   retval = true;
    }
    catch(...)
    {
      retval = false;
325 }
    delete[] N0;
    delete[] N1;
    delete[] NF;
    delete[] DEX;
330 delete[] DEY;
    return retval;
}
```

60 fraktal_sft/exr.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen
5

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20 #ifndef KF_EXR_H
#define KF_EXR_H 1

#include <string>
```

```
25  class itercount_array;

    extern int SaveEXR
    ( const std::string &filename
    , const unsigned char *Data
30  , int nWidth
    , int nHeight
    , int nColors
    , const std::string &comment
    , int64_t maxiter
35  , int arrWidth
    , int arrHeight
    , const itercount_array &count
    , const float *trans
    , const float *dex
40  , const float *dey
    );

    extern std::string ReadEXRComment(const std::string &filename);

45  extern bool ReadEXRMapFile(const std::string &filename);

    #endif
```

61 fraktal_sft/floatexp_approximation.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "complex.h"

    floatexp diffabs(const floatexp &c, const floatexp &d)
    {
25    if (c.val >= 0) if ((c + d).val >= 0) return d; else return -(c.mul2() + d);
      else if ((c + d).val > 0) return c.mul2() + d; else return -d;
    }

    void CFraktalSFT::CalculateApproximation(int nType)
30  {
        m_nApprox = 0;
```

```
        bool isC = GetApproximationType() == SeriesType_Complex;
        bool isR = GetApproximationType() == SeriesType_Real;

#if 0
        // disabled for now because of severe underskipping problems
        double nProbeRatio = 1.0 / 65536.0;
        double r = sqrt(nProbeRatio);
        int nProbeX = ceil(fabs(m_rApprox.right - m_rApprox.left) * r);
        int nProbeY = ceil(fabs(m_rApprox.bottom - m_rApprox.top) * r);
        if (nProbeX < 3) nProbeX = 3;
        if (nProbeY < 3) nProbeY = 3;
#else
        int nProbeX = 3;
        int nProbeY = 3;
#endif
        int nProbe = nProbeX * nProbeY - 1;
        floatexp _1 = 1;
        floatexp _3 = 3;
        CFixedFloat cr, ci;
        floatexp *dbTr = new floatexp[nProbe];
        floatexp *dbTi = new floatexp[nProbe];
        floatexp *dbTr0 = new floatexp[nProbe];
        floatexp *dbTi0 = new floatexp[nProbe];
        cr = m_rstart;
        ci = m_istart;

        POINT *p = new POINT[nProbe];
        int k = 0;
        for (int j = 0; j < nProbeY; ++j)
        {
                int y = m_rApprox.top + j * (m_rApprox.bottom - m_rApprox.top) /↙
                    ↳ (nProbeY - 1);
                if (y < m_rApprox.top) y = m_rApprox.top;
                if (y >= m_rApprox.bottom) y = m_rApprox.bottom - 1;
                for (int i = 0; i < nProbeX; ++i)
                {
                        if (i == nProbeX/2 && j == nProbeY/2)
                                continue;
                        int x = m_rApprox.left + i * (m_rApprox.right - ↙
                            ↳ m_rApprox.left) / (nProbeX - 1);
                        if (x < m_rApprox.left) x = m_rApprox.left;
                        if (x >= m_rApprox.right) x = m_rApprox.right - 1;
                        p[k].x = x;
                        p[k].y = y;
                        ++k;
                }
        }
        assert(k == nProbe);

        for (int j = 0; j<nProbe; j++){
                GetPixelCoordinates(p[j].x, p[j].y, dbTr0[j], dbTi0[j]);
                dbTr[j] = dbTr0[j];
                dbTi[j] = dbTi0[j];
        }
        m_nMaxApproximation = m_nMaxIter;
        //floatexp mindiff = (m_nBailout==2?0.000001:0.001);
        floatexp mindiff;
```

```
	if (GetApproxLowTolerance())
		mindiff = 0.00001;
	else
		mindiff = 0.001;
	//	if(dbTr[0]<0 && dbTi[0]<1e-16 && dbTi[0]>-1e-16)
	//		mindiff = 0.0000001;
	if (GetNoApprox())
		mindiff = 0;
	if (GetAutoApproxTerms()){
		int nT = sqrt((double)m_nTotal)*0.021;
		if ((GetFractalType() == 0 && GetPower() > 2) || (GetFractalType↙
		    ↳ () == 1 && GetPower() == 2))
			nT = 3;
		SetApproxTerms(nT);
	}
	int m_nTerms = GetApproxTerms();

	// clear approximation
	if (isC)
		for (int i = 0; i < m_nTerms; i++)
		{
			m_APr[i] = 0;
			m_APi[i] = 0;
		}
	if (isR)
		for (int i = 0; i <= m_nTerms; ++i)
			for (int j = 0; j <= m_nTerms - i; ++j)
			{
				m_APs->s[i][j] = 0;
				m_APs->t[i][j] = 0;
			}

	if (GetApproximationType() == SeriesType_None || m_nZoom<g_nRefZero){
		m_nMaxApproximation = 0;
		return;
	}

	floatexp xr;
	floatexp xi;
	floatexp *APr = isC ? new floatexp[m_nTerms] : nullptr;
	floatexp *APi = isC ? new floatexp[m_nTerms] : nullptr;
	SeriesR2 *APs = isR ? new SeriesR2 : nullptr;

	// copy approximation
	if (isC)
		for (int i = 0; i < m_nTerms; i++)
		{
			APr[i] = m_APr[i];
			APi[i] = m_APi[i];
		}
	if (isR)
		for (int i = 0; i <= m_nTerms; ++i)
			for (int j = 0; j <= m_nTerms - i; ++j)
			{
				APs->s[i][j] = m_APs->s[i][j];
				APs->t[i][j] = m_APs->t[i][j];
			}
```

```
        for (int64_t iteration = 0; iteration<m_nMaxIter - 1 && !m_bStop; ↙
            ↳ iteration++){
145             m_nApprox++;

                // get reference
                int64_t n = iteration - 1;
                if (iteration == 0)
150                     xr = xi = 0;
                else if (nType == 0){
                        xr = m_db_dxr[n];
                        xi = m_db_dxi[n];
                }
155             else if (nType == 1){
                        xr = m_ldxr[n];
                        xi = m_ldxi[n];
                }
                else{
160                     xr = m_dxr[n];
                        xi = m_dxi[n];
                }

                // copy approximation
165             if (isC)
                        for (int i = 0; i < m_nTerms; i++)
                        {
                                APr[i] = m_APr[i];
                                APi[i] = m_APi[i];
170                     }
                if (isR)
                        for (int i = 0; i <= m_nTerms; ++i)
                                for (int j = 0; j <= m_nTerms - i; ++j)
                                {
175                                     APs->s[i][j] = m_APs->s[i][j];
                                        APs->t[i][j] = m_APs->t[i][j];
                                }

                // step approximation
180             if (m_nFractalType == 0 && isC)
                {
                        if (m_nPower == 2){
                                m_APr[0] = (xr*APr[0] - xi*APi[0]).mul2() + _1;
                                m_APi[0] = (xi*APr[0] + xr*APi[0]).mul2();
185                             for (int k = 1; k<m_nTerms; k++){
                                        m_APr[k] = (xr*APr[k] - xi*APi[k]).mul2↙
                                            ↳ ();
                                        m_APi[k] = (xi*APr[k] + xr*APi[k]).mul2↙
                                            ↳ ();
                                        int n = k / 2;
                                        int q = n;
190                                     int f = 0;
                                        int r = k - 1;
                                        while (q){
                                                m_APr[k] += (APr[f] * APr[r] - ↙
                                                    ↳ APi[f] * APi[r]).mul2();
                                                m_APi[k] += (APi[f] * APr[r] + ↙
                                                    ↳ APr[f] * APi[r]).mul2();
```

```
195                                             q--;
                                                f++;
                                                r--;
                                        }
                                        if (k % 2){
200                                             m_APr[k] += APr[n] * APr[n] - ↙
                                                    ↳ APi[n] * APi[n];
                                                m_APi[k] += (APr[n] * APi[n]).↙
                                                    ↳ mul2();
                                        }
                                }
                        }
205                     else
                        {
                                int P = m_nPower;
                                complex<floatexp> X(xr, xi), A(APr[0], APi[0]), ↙
                                    ↳ B(APr[1], APi[1]), C(APr[2], APi[2]);
                                complex<floatexp> An = P * A * (X^(P-1)) + 1;
210                             complex<floatexp> Bn = P * B * (X^(P-1)) + ((P*(↙
                                    ↳ P-1))/2) * (A^2) * (X^(P-2));
                                complex<floatexp> Cn = P * C * (X^(P-1)) + (P*(P↙
                                    ↳ -1)) * A * B * (X^(P-2)) + ((P*(P-2)*(P-2)↙
                                    ↳ )/6) * (A^3) * (X^(P-3));
                                m_APr[0] = An.m_r;
                                m_APi[0] = An.m_i;
                                m_APr[1] = Bn.m_r;
215                             m_APi[1] = Bn.m_i;
                                m_APr[2] = Cn.m_r;
                                m_APi[2] = Cn.m_i;
                        }
                }
220             else if (m_nFractalType == 1 && isR)
                {
                        if (m_nPower == 2)
                        {

225     enum { Q0, Q1, Q2, Q3 } q;
        if ((xr * xi).val >= 0) q = Q0; else q = Q3;
        for (int i = 0; i <= m_nTerms; ++i)
        {
                for (int j = 0; j <= m_nTerms - i; ++j)
                {
230                     // s
                        floatexp total = 0;
                        if (i == 1 && j == 0) total += 1;
                        total += 2 * (xr * APs->s[i][j] - xi * APs->t[i][j]);
235                     for (int k = 0; k <= i; ++k)
                        {
                                const int ik = i - k;
                                for (int l = 0; l <= j; ++l)
                                {
240                                     const int jl = j - l;
                                        total += APs->s[k][l] * APs->s[ik][jl] -↙
                                            ↳  APs->t[k][l] * APs->t[ik][jl];
                                }
                        }
                        m_APs->s[i][j] = total;
```

```
245                             // t;
                                total = 0;
                                total += xr * APs->t[i][j] + xi * APs->s[i][j];
                                for (int k = 0; k <= i; ++k)
                                {
250                                     const int ik = i - k;
                                        for (int l = 0; l <= j; ++l)
                                        {
                                                const int jl = j - l;
                                                total += APs->s[k][l] * APs->t[ik][jl];
255                                     }
                                }
                                if (i == 0 && j == 0 && (q == Q1 || q == Q2)) total += 2↙
                                    ↪  * xr * xi;
                                total = total.mul2();
                                if (q == Q1 || q == Q3) total = -total;
260                             if (i == 0 && j == 1) total += 1;
                                m_APs->t[i][j] = total;
                        }
                }

265                             }
                        }

                                /*
                                An+1 = 2XnAn + 1
270                             Bn+1 = 2XnBn + An2
                                Cn+1 = 2XnCn + 2AnBn (33097)
                                Dn+1 = 2XnDn + 2AnCn + Bn2
                                En+1 = 2XnEn + 2AnDn + 2BnCn (38613)
                                Fn+1 = 2XnFn + 2AnEn + 2BnDn + Cn2 (38613)
275                             Gn+1 = 2XnGn + 2AnFn + 2BnEn + 2CnDn
                                Hn+1 = 2XnHn + 2AnGn + 2BnFn + 2CnEn + Dn2
                                In+1 = 2XnIn + 2AnHn + 2BnGn + 2CnFn + 2DnEn
                                */
                                if (iteration<m_nMaxApproximation){
280                                     floatexp dxr, dxi;
                                        if (nType == 0){
                                                dxr = m_db_dxr[iteration];
                                                dxi = m_db_dxi[iteration];
                                        }
285                                     else if (nType == 1){
                                                dxr = m_ldxr[iteration];
                                                dxi = m_ldxi[iteration];
                                        }
                                        else{
290                                             dxr = m_dxr[iteration];
                                                dxi = m_dxi[iteration];
                                        }
                                        int j = 0;
                                        for (j = 0; j<nProbe; j++){
295
                                                // do approximation
                                                int64_t antal_;
                                                floatexp Dnr, Dni, DDnr_, DDni_;
                                                if (isC)
300                                                     DoApproximation(antal_, dbTr0[j↙
```

```
                ↳ ], dbTi0[j], Dnr, Dni, ↙
                ↳ DDnr_, DDni_);
        if (isR)
                DoApproximation(dbTr0[j], dbTi0[↙
                    ↳ j], Dnr, Dni);

        // compare with probe point
305     floatexp diff = (Dnr - dbTr[j]) / dbTr[j↙
            ↳ ];
        if (diff>mindiff || diff<-mindiff){
                m_nMaxApproximation = iteration;
                break;
        }
310     diff = (Dni - dbTi[j]) / dbTi[j];
        if (diff>mindiff || diff<-mindiff){
                m_nMaxApproximation = iteration;
                break;
        }
315     double yr = (dxr + Dnr).todouble();
        double yi = (dxi + Dni).todouble();
        if (g_real*yr*yr + g_imag*yi*yi>↙
            ↳ m_nBailout2){
                m_nMaxApproximation = iteration;
                break;
320     }

        // step probe point using perturbation
        if (m_nFractalType == 0)
        {
325
                if (m_nPower == 2){
                        Dnr = (dxr*dbTr[j] - dxi↙
                            ↳ *dbTi[j]).mul2() +↙
                            ↳  dbTr[j] * dbTr[j]↙
                            ↳  - dbTi[j] * dbTi[↙
                            ↳ j] + dbTr0[j];
                        Dni = (dxr*dbTi[j] + dxi↙
                            ↳ *dbTr[j] + dbTr[j]↙
                            ↳  * dbTi[j]).mul2()↙
                            ↳  + dbTi0[j];
                }
330             else if (m_nPower == 3){
                        Dnr = -3*((dxr*dxr - dxi↙
                            ↳ *dxi)*dbTr[j] + ↙
                            ↳ dxr*(dbTr[j] * ↙
                            ↳ dbTr[j] - dbTi[j] ↙
                            ↳ * dbTi[j]) - dbTi[↙
                            ↳ j] * (dxi*(dxr + ↙
                            ↳ dbTr[j]).mul2() + ↙
                            ↳ dbTr[j] * dbTi[j])↙
                            ↳ ) + dbTr[j] * dbTr↙
                            ↳ [j] * dbTr[j] + ↙
                            ↳ dbTr0[j];
                        Dni = -3*((dxr*dxr - dxi↙
                            ↳ *dxi)*dbTi[j] + ↙
                            ↳ dxi*(dbTr[j] * ↙
                            ↳ dbTr[j] - dbTi[j] ↙
```

```
                    ↪ * dbTi[j]) + dbTr[↙
                    ↪ j] * (dxr*(dxi + ↙
                    ↪ dbTi[j]).mul2() + ↙
                    ↪ dbTr[j] * dbTi[j])↙
                    ↪ ) - dbTi[j] * dbTi↙
                    ↪ [j] * dbTi[j] + ↙
                    ↪ dbTi0[j];
            }
            else if (m_nPower == 4){
                complex<floatexp> X(dxr,↙
                    ↪  dxi);
                complex<floatexp> D(dbTr↙
                    ↪ [j], dbTi[j]);
                complex<floatexp> D0(↙
                    ↪ dbTr0[j], dbTi0[j↙
                    ↪ ]);
                complex<floatexp> _4(4, ↙
                    ↪ 0), _6(6, 0);
                complex<floatexp> Dn = ↙
                    ↪ _4*(X ^ 3)*D + _6↙
                    ↪ *(X ^ 2)*(D ^ 2) +↙
                    ↪  _4*X*(D ^ 3) + (D↙
                    ↪  ^ 4) + D0;
                Dni = Dn.m_i;
                Dnr = Dn.m_r;
            }
            else if (m_nPower == 5){
                complex<floatexp> X(dxr,↙
                    ↪  dxi);
                complex<floatexp> D(dbTr↙
                    ↪ [j], dbTi[j]);
                complex<floatexp> D0(↙
                    ↪ dbTr0[j], dbTi0[j↙
                    ↪ ]);
                complex<floatexp> _5(5, ↙
                    ↪ 0), _10(10, 0);
                complex<floatexp> Dn = ↙
                    ↪ _5*(X ^ 4)*D + _10↙
                    ↪ *(X ^ 3)*(D ^ 2) +↙
                    ↪  _10*(X ^ 2)*(D ^ ↙
                    ↪ 3) + _5*X*(D ^ 4) ↙
                    ↪ + (D ^ 5) + D0;
                Dni = Dn.m_i;
                Dnr = Dn.m_r;
            }
            else if (m_nPower == 6){
                complex<floatexp> X(dxr,↙
                    ↪  dxi);
                complex<floatexp> D(dbTr↙
                    ↪ [j], dbTi[j]);
                complex<floatexp> D0(↙
                    ↪ dbTr0[j], dbTi0[j↙
                    ↪ ]);
                complex<floatexp> _6(6, ↙
                    ↪ 0), _15(15, 0), ↙
                    ↪ _20(20, 0);
                complex<floatexp> Dn = ↙
```

```
                    ↪ _6*(X ^ 5)*D + _15↙
                    ↪ *(X ^ 4)*(D ^ 2) +↙
                    ↪  _20*(X ^ 3)*(D ^ ↙
                    ↪ 3) + _15*(X ^ 2)*(↙
                    ↪ D ^ 4) + _6*X*(D ^↙
                    ↪  5) + (D ^ 6) + D0↙
                    ↪ ;
                Dni = Dn.m_i;
                Dnr = Dn.m_r;
360         }
            else if (m_nPower == 7){
                complex<floatexp> X(dxr,↙
                    ↪  dxi);
                complex<floatexp> D(dbTr↙
                    ↪ [j], dbTi[j]);
                complex<floatexp> D0(↙
                    ↪ dbTr0[j], dbTi0[j↙
                    ↪ ]);
365             complex<floatexp> _7(7, ↙
                    ↪ 0), _21(21, 0), ↙
                    ↪ _35(35, 0);
                complex<floatexp> Dn = ↙
                    ↪ _7*(X ^ 6)*D + _21↙
                    ↪ *(X ^ 5)*(D ^ 2) +↙
                    ↪  _35*(X ^ 4)*(D ^ ↙
                    ↪ 3) + _35*(X ^ 3)*(↙
                    ↪ D ^ 4) + _21*(X ^ ↙
                    ↪ 2)*(D ^ 5) + _7*X↙
                    ↪ *(D ^ 6) + (D ^ 7)↙
                    ↪  + D0;
                Dni = Dn.m_i;
                Dnr = Dn.m_r;
            }
370         else if (m_nPower == 8){
                complex<floatexp> X(dxr,↙
                    ↪  dxi);
                complex<floatexp> D(dbTr↙
                    ↪ [j], dbTi[j]);
                complex<floatexp> D0(↙
                    ↪ dbTr0[j], dbTi0[j↙
                    ↪ ]);
                complex<floatexp> _8(8, ↙
                    ↪ 0), _28(28, 0), ↙
                    ↪ _56(56, 0), _70↙
                    ↪ (70, 0);
375             complex<floatexp> Dn = ↙
                    ↪ _8*(X ^ 7)*D + _28↙
                    ↪ *(X ^ 6)*(D ^ 2) +↙
                    ↪  _56*(X ^ 5)*(D ^ ↙
                    ↪ 3) + _70*(X ^ 4)*(↙
                    ↪ D ^ 4) + _56*(X ^ ↙
                    ↪ 3)*(D ^ 5) + _28*(↙
                    ↪ X ^ 2)*(D ^ 6) + ↙
                    ↪ _8*X*(D ^ 7) + (D ↙
                    ↪ ^ 8) + D0;
                Dni = Dn.m_i;
                Dnr = Dn.m_r;
```

```
                }
                else if (m_nPower == 9){
380                     complex<floatexp> X(dxr,↙
                            ↳  dxi);
                        complex<floatexp> D(dbTr↙
                            ↳ [j], dbTi[j]);
                        complex<floatexp> D0(↙
                            ↳ dbTr0[j], dbTi0[j↙
                            ↳ ]);
                        complex<floatexp> _9(9, ↙
                            ↳ 0), _36(36, 0), ↙
                            ↳ _84(84, 0), _126↙
                            ↳ (126, 0);
                        complex<floatexp> Dn = ↙
                            ↳ _9*(X ^ 8)*D + _36↙
                            ↳ *(X ^ 7)*(D ^ 2) +↙
                            ↳  _84*(X ^ 6)*(D ^ ↙
                            ↳ 3) + _126*(X ^ 5)↙
                            ↳ *(D ^ 4) + _126*(X↙
                            ↳  ^ 4)*(D ^ 5) + ↙
                            ↳ _84*(X ^ 3)*(D ^ ↙
                            ↳ 6) + _36*(X ^ 2)*(↙
                            ↳ D ^ 7) + _9*X*(D ^↙
                            ↳  8) + (D ^ 9) + D0↙
                            ↳ ;
385                     Dni = Dn.m_i;
                        Dnr = Dn.m_r;
                }
                else if (m_nPower == 10){
                        complex<floatexp> X(dxr,↙
                            ↳  dxi);
390                     complex<floatexp> D(dbTr↙
                            ↳ [j], dbTi[j]);
                        complex<floatexp> D0(↙
                            ↳ dbTr0[j], dbTi0[j↙
                            ↳ ]);
                        complex<floatexp> _10↙
                            ↳ (10, 0), _45(45, ↙
                            ↳ 0), _120(120, 0), ↙
                            ↳ _210(210, 0), _252↙
                            ↳ (252, 0);
                        complex<floatexp> Dn = ↙
                            ↳ _10*(X ^ 9)*D + ↙
                            ↳ _45*(X ^ 8)*(D ^ ↙
                            ↳ 2) + _120*(X ^ 7)↙
                            ↳ *(D ^ 3) + _210*(X↙
                            ↳  ^ 6)*(D ^ 4) + ↙
                            ↳ _252*(X ^ 5)*(D ^ ↙
                            ↳ 5) + _210*(X ^ 4)↙
                            ↳ *(D ^ 6) + _120*(X↙
                            ↳  ^ 3)*(D ^ 7) + ↙
                            ↳ _45*(X ^ 2)*(D ^ ↙
                            ↳ 8) + _10*X*(D ^ 9)↙
                            ↳  + (D ^ 10) + D0;
                        Dni = Dn.m_i;
395                     Dnr = Dn.m_r;
                }
```

```
                                        else{
                                                complex<floatexp> X(dxr,↙
                                                    ↪  dxi);
                                                complex<floatexp> D(dbTr↙
                                                    ↪ [j], dbTi[j]);
400                                             complex<floatexp> D0(↙
                                                    ↪ dbTr0[j], dbTi0[j↙
                                                    ↪ ]);
                                                complex<floatexp> c(↙
                                                    ↪ m_pnExpConsts[0], ↙
                                                    ↪ 0);
                                                int nXExp = m_nPower - ↙
                                                    ↪ 2, nDExp = 2, ci =↙
                                                    ↪  1;
                                                complex<floatexp> Dn = c↙
                                                    ↪ *(X^(m_nPower - 1)↙
                                                    ↪ )*D;
                                                while (nXExp){
405                                                     c.m_r = ↙
                                                            ↪ m_pnExpConsts↙
                                                            ↪ [ci++];
                                                        Dn += c*(X^nXExp↙
                                                            ↪ )*(D^nDExp↙
                                                            ↪ );
                                                        nXExp--;
                                                        nDExp++;
                                                }
410                                             Dn += (D^m_nPower) + D0;
                                                Dni = Dn.m_i;
                                                Dnr = Dn.m_r;
                                        }

415                                 }
                                    else if (m_nFractalType == 1)
                                    {

                                        if (m_nPower == 2)
420                                     {
                                                Dnr = (dxr*dbTr[j] - dxi↙
                                                    ↪ *dbTi[j]).mul2() +↙
                                                    ↪  dbTr[j] * dbTr[j]↙
                                                    ↪  - dbTi[j] * dbTi[↙
                                                    ↪ j] + dbTr0[j];
                                                Dni = diffabs(dxr * dxi,↙
                                                    ↪  dxr*dbTi[j] + dxi↙
                                                    ↪ *dbTr[j] + dbTr[j]↙
                                                    ↪  * dbTi[j]).mul2()↙
                                                    ↪  + dbTi0[j];
                                        }

425                                 }
                                    dbTr[j] = Dnr;
                                    dbTi[j] = Dni;
                                }
                            if (j<nProbe)
430                             break;
                    }
```

```
        }

        // store last-but-one approximation
        if (isC)
                for (int i = 0; i<m_nTerms; i++){
                        m_APr[i] = APr[i];
                        m_APi[i] = APi[i];
                }
        if (isR)
        {
//              bool isZero = true;
                for (int i = 0; i <= m_nTerms; ++i)
                        for (int j = 0; j <= m_nTerms - i; ++j)
                        {
                                m_APs->s[i][j] = APs->s[i][j];
                                m_APs->t[i][j] = APs->t[i][j];
//                              isZero &= m_APs->s[i][j] == 0;
//                              isZero &= m_APs->t[i][j] == 0;
                        }
/*
                for (int i = 0; i <= m_nTerms; ++i)
                {
                        for (int j = 0; j <= m_nTerms - i; ++j)
                                std::cerr << m_APs->s[i][j].toLongDouble() << "\
                                    ↪ t";
                        std::cerr << std::endl;
                }
                for (int i = 0; i <= m_nTerms; ++i)
                {
                        for (int j = 0; j <= m_nTerms - i; ++j)
                                std::cerr << m_APs->t[i][j].toLongDouble() << "\
                                    ↪ t";
                        std::cerr << std::endl;
                }
*/
        }

        if (GetNoApprox())
                m_nMaxApproximation = 0;
        if (isC)
        {
                delete[] APr;
                delete[] APi;
        }
        if (isR)
                delete APs;
        delete[] dbTr;
        delete[] dbTi;
        delete[] dbTr0;
        delete[] dbTi0;
        delete[] p;
}

void CFraktalSFT::DoApproximation(int64_t &antal, const floatexp &D0r, const \
    ↪ floatexp &D0i, floatexp &TDnr, floatexp &TDni, floatexp &TDDnr, floatexp &\
    ↪ TDDni)
{
```

```
485         if (m_nMaxApproximation)
            {
                    antal = m_nMaxApproximation - 1;
                    TDnr = 0; TDni = 0; TDDnr = 0; TDDni = 0;
                    for (int k = GetApproxTerms() - 1; k >= 0; --k)
490                 {
                            floatexp tr = TDnr * D0r - TDni * D0i + m_APr[k];
                            floatexp ti = TDnr * D0i + TDni * D0r + m_APi[k];
                            TDnr = tr;
                            TDni = ti;
495                         tr = TDDnr * D0r - TDDni * D0i + m_APr[k] * floatexp(k +↙
                                ↳ 1.0);
                            ti = TDDnr * D0i + TDDni * D0r + m_APi[k] * floatexp(k +↙
                                ↳ 1.0);
                            TDDnr = tr;
                            TDDni = ti;
                    }
500                 floatexp tr = TDnr * D0r - TDni * D0i;
                    floatexp ti = TDnr * D0i + TDni * D0r;
                    TDnr = tr;
                    TDni = ti;
            }
505         else
            {
                    antal = 0;
                    TDnr = D0r;
                    TDni = D0i;
510                 TDDnr = 1.0;
                    TDDni = 0.0;
            }
    }

515 void CFraktalSFT::DoApproximation
    ( int64_t &antal
    , const floatexp &a
    , const floatexp &b
    , floatexp &x
520 , floatexp &y
    , floatexp &dxa
    , floatexp &dxb
    , floatexp &dya
    , floatexp &dyb
525 )
    {
            switch (GetApproximationType())
            {
                    case SeriesType_Complex:
530                 {
                            floatexp dx, dy;
                            DoApproximation(antal, a, b, x, y, dx, dy);
                            // Cauchy-Riemann
                            dxa =  dx;
535                         dxb = -dy;
                            dya =  dy;
                            dyb =  dx;
                            break;
                    }
```

```
540                     case SeriesType_Real:
                        {
                                if (m_nMaxApproximation)
                                {
                                        antal = m_nMaxApproximation - 1;
545                                     auto order = GetApproxTerms();
                                        floatexp an[order + 1];
                                        floatexp bn[order + 1];
                                        floatexp dan[order + 1];
                                        floatexp dbn[order + 1];
550                                     an[0] = 1;
                                        bn[0] = 1;
                                        dan[0] = 0;
                                        dbn[0] = 0;
                                        for (auto i = 1; i <= order; ++i)
555                                     {
                                                an[i] = an[i - 1] * a;
                                                bn[i] = bn[i - 1] * b;
                                                dan[i] = i * an[i - 1];
                                                dbn[i] = i * bn[i - 1];
560                                     }
                                        // x   = sum ApproxSeriesR2X[i][j]   a^i       b↙
                                            ↳ ^j
                                        // dxa = sum ApproxSeriesR2X[i][j] i a^{i-1}   b↙
                                            ↳ ^j
                                        // dxb = sum ApproxSeriesR2X[i][j]   a^i     j b↙
                                            ↳ ^{j-1}
                                        floatexp totalX = 0;
565                                     floatexp totalY = 0;
                                        floatexp totalDXA = 0;
                                        floatexp totalDXB = 0;
                                        floatexp totalDYA = 0;
                                        floatexp totalDYB = 0;
570                                     for (auto i = 0; i <= order; ++i)
                                        {
                                                floatexp subtotalX = 0;
                                                floatexp subtotalY = 0;
                                                floatexp subtotalDXB = 0;
575                                             floatexp subtotalDYB = 0;
                                                for (auto j = 0; j <= order - i; ++j)
                                                {
                                                        subtotalX   += m_APs->s[i][j] * ↙
                                                            ↳ bn[j];
                                                        subtotalY   += m_APs->t[i][j] * ↙
                                                            ↳ bn[j];
580                                                     subtotalDXB += m_APs->s[i][j] * ↙
                                                            ↳ dbn[j];
                                                        subtotalDYB += m_APs->t[i][j] * ↙
                                                            ↳ dbn[j];
                                                }
                                                totalX   += subtotalX   *  an[i];
                                                totalY   += subtotalY   *  an[i];
585                                             totalDXA += subtotalX   * dan[i];
                                                totalDYA += subtotalY   * dan[i];
                                                totalDXB += subtotalDXB *  an[i];
                                                totalDYB += subtotalDYB *  an[i];
                                        }
```

```
590                                     x   = totalX;
                                        y   = totalY;
                                        dxa = totalDXA;
                                        dxb = totalDXB;
                                        dya = totalDYA;
595                                     dyb = totalDYB;
                                }
                                else
                                {
                                        antal = 0;
600                                     x = a;
                                        y = b;
                                        dxa = 1.0;
                                        dxb = 0.0;
                                        dya = 0.0;
605                                     dyb = 1.0;
                                }
                                break;
                        }
                        case SeriesType_None:
610                     {
                                antal = 0;
                                x = a;
                                y = b;
                                dxa = 1.0;
615                             dxb = 0.0;
                                dya = 0.0;
                                dyb = 1.0;
                        }
                }
620     }

        void CFraktalSFT::DoApproximation
        ( const floatexp &a
        , const floatexp &b
625     , floatexp &x
        , floatexp &y
        )
        {
                int order = GetApproxTerms();
630             floatexp an[order + 1];
                floatexp bn[order + 1];
                an[0] = 1;
                bn[0] = 1;
                for (int i = 1; i <= order; ++i)
635             {
                        an[i] = an[i - 1] * a;
                        bn[i] = bn[i - 1] * b;
                }
                floatexp totalX = 0;
640             floatexp totalY = 0;
                for (int i = 0; i <= order; ++i)
                {
                        floatexp subtotalX = 0;
                        floatexp subtotalY = 0;
645                     for (int j = 0; j <= order - i; ++j)
                        {
```

```
                        subtotalX += m_APs->s[i][j] * bn[j];
                        subtotalY += m_APs->t[i][j] * bn[j];
                }
                totalX += subtotalX * an[i];
                totalY += subtotalY * an[i];
        }
        x = totalX;
        y = totalY;
}
```

62 fraktal_sft/floatexp.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_FLOATEXP_H
#define KF_FLOATEXP_H

#include <math.h>
#include <stdint.h>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <limits>
#include "CFixedFloat.h"

#define MAX_PREC 1020
// this has two fewer 0 than you might expect, this is to give headroom for
// avoiding overflow in + and other functions. it is the exponent for 0.0
#define EXP_MIN (-0x80000000000000LL)

class floatexp
{
public:
        double val;
        int64_t exp;
        inline void align()
        {
                if (val != 0)
                {
                        union { double d; int64_t i; } u;
```

```
                        u.d = val;
                        exp += ((u.i & 0x7FF0000000000000LL) >> 52) - 1023;
                        u.i = (u.i & 0x800FFFFFFFFFFFFFLL) | 0x3FF0000000000000LL;
                        val = u.d;
                }
                else
                {
                        val = 0;
                        exp = EXP_MIN;
                }
        }
        inline floatexp &abs()
        {
                if(val<0)
                        val=-val;
                return *this;
        }
        inline void initFromDouble(double a)
        {
                val=a;
                exp=0;
                align();
        }
        inline void initFromLongDouble(long double a)
        {
                using std::frexp;
                int e = 0;
                a = frexp(a, &e);
                val=a;
                exp=e;
                align();
        }
        inline double setExp(double newval,int64_t newexp) const
        {
//              int64_t tmpval = (*((int64_t*)&newval) & 0x800FFFFFFFFFFFFF) | ((newexp+1023)<<52);
//              memcpy(&newval,&tmpval,sizeof(double));
//              return newval;
                union { double d; int64_t i; } u;
                u.d = newval;
                u.i = (u.i & 0x800FFFFFFFFFFFFFLL) | ((newexp + 1023) << 52);
                newval = u.d;
                return newval;
        }
        inline floatexp()
        {
                val = 0;
                exp = EXP_MIN;
        }
        inline floatexp(int a)
        {
                initFromDouble(a);
        }
        inline floatexp(int64_t a)
        {
                initFromDouble(a);
```

```
	}
	inline floatexp(double a)
	{
		initFromDouble(a);
	}
	inline floatexp(double a, int64_t e)
	{
		val = a;
		exp = e;
		align();
	}
	inline floatexp(double a, int64_t e, int dummy)
	{
		(void) dummy;
		val = a;
		exp = e;
	}
	inline floatexp(long double a)
	{
		initFromLongDouble(a);
	}
	inline floatexp &operator =(const floatexp &a)
	{
		val=a.val;
		exp=a.exp;
		return *this;
	}
	inline floatexp &operator =(int a)
	{
		initFromDouble((double)a);
		return *this;
	}
	inline floatexp &operator =(double a)
	{
		initFromDouble(a);
		return *this;
	}
	inline floatexp &operator =(long double a)
	{
		initFromLongDouble(a);
		return *this;
	}
	inline floatexp operator *(const floatexp &a) const
	{
		floatexp r;
		r.val = a.val*val;
		r.exp = a.exp+exp;
		r.align();
		return r;
	}
	inline floatexp operator /(const floatexp &a) const
	{
		floatexp r;
		r.val = val/a.val;
		r.exp = exp - a.exp;
		r.align();
		return r;
```

```
        }
        __attribute__ ((warn_unused_result))
        inline floatexp mul2() const
        {
                floatexp r;
                r.val = val;
                r.exp = exp + 1;
                return r;
        }
        __attribute__ ((warn_unused_result))
        inline floatexp mul4() const
        {
                floatexp r;
                r.val = val;
                r.exp = exp + 2;
                return r;
        }
        inline floatexp operator +(const floatexp &a) const
        {
                floatexp r;
                int64_t diff;
                if(exp>a.exp){
                        diff = exp-a.exp;
                        r.exp = exp;
                        if(diff>MAX_PREC)
                                r.val=val;
                        else{
                                double aval = setExp(a.val,-diff);
                                r.val = val+aval;
                        }
                }
                else{
                        diff = a.exp-exp;
                        r.exp = a.exp;
                        if(diff>MAX_PREC)
                                r.val=a.val;
                        else{
                                double aval = setExp(val,-diff);
                                r.val = a.val+aval;
                        }
                }
                r.align();
                return r;
        }
        inline floatexp operator -() const
        {
                floatexp r=*this;
                r.val=-r.val;
                return r;
        }
        inline floatexp &operator +=(const floatexp &a)
        {
                *this = *this+a;
                return *this;
        }
        inline floatexp operator -(const floatexp &a) const
        {
```

```
215                 floatexp r;
                    int64_t diff;
                    if(exp>a.exp){
                            diff = exp-a.exp;
                            r.exp = exp;
220                         if(diff>MAX_PREC)
                                    r.val = val;
                            else{
                                    double aval = setExp(a.val,-diff);
                                    r.val = val-aval;
225                         }
                    }
                    else{
                            diff = a.exp-exp;
                            r.exp = a.exp;
230                         if(diff>MAX_PREC)
                                    r.val=-a.val;
                            else{
                                    double aval = setExp(val,-diff);
                                    r.val = aval-a.val;
235                         }
                    }
                    r.align();
                    return r;
            }
240         inline floatexp &operator -=(const floatexp &a)
            {
                    *this = *this-a;
                    return *this;
            }
245         inline bool operator >(const floatexp &a) const
            {
                    if(val>0){
                            if(a.val<0)
                                    return true;
250                         if(exp>a.exp)
                                    return true;
                            else if(exp<a.exp)
                                    return false;
                            return val>a.val;
255                 }
                    else{
                            if(a.val>0)
                                    return false;
                            if(exp>a.exp)
260                                 return false;
                            else if(exp<a.exp)
                                    return true;
                            return val>a.val;
                    }
265         }
            inline bool operator <(const floatexp &a) const
            {
                    if(val>0){
                            if(a.val<0)
270                                 return false;
                            if(exp>a.exp)
```

```
                                return false;
                        else if(exp<a.exp)
                                return true;
                        return val<a.val;
                }
                else{
                        if(a.val>0)
                                return true;
                        if(exp>a.exp)
                                return true;
                        else if(exp<a.exp)
                                return false;
                        return val<a.val;
                }
        }
        inline bool operator <=(const floatexp &a) const
        {
                return (*this<a || *this==a);
        }
        inline bool operator >=(const floatexp &a) const
        {
                return (*this>a || *this==a);
        }
        inline bool operator <=(const int a) const
        {
                return (*this<a || *this==a);
        }
        inline bool operator ==(const floatexp &a) const
        {
                if(exp!=a.exp)
                        return false;
                return val==a.val;
        }
        inline bool iszero() const
        {
                return (val==0 && exp==0);
        }
        inline double todouble() const
        {
                if(exp<-MAX_PREC || exp>MAX_PREC)
                        return 0;
                return setExp(val,exp);
        }
        inline explicit operator double () const
        {
                return todouble();
        }
        inline double todouble(int nScaling) const
        {
                if(!nScaling)
                        return todouble();
                floatexp ret = *this;
                while(nScaling>9){
                        ret.val*=1e10;
                        ret.align();
                        nScaling-=10;
                }
```

```
                while(nScaling>2){
330                     ret.val*=1e3;
                        ret.align();
                        nScaling-=3;
                }
                while(nScaling>0){
335                     ret.val*=1e1;
                        ret.align();
                        nScaling--;
                }
                while(nScaling<-9){
340                     ret.val/=1e10;
                        ret.align();
                        nScaling+=10;
                }
                while(nScaling<-2){
345                     ret.val/=1e3;
                        ret.align();
                        nScaling+=3;
                }
                while(nScaling<0){
350                     ret.val/=1e1;
                        ret.align();
                        nScaling++;
                }
                if(ret.exp<-MAX_PREC || ret.exp>MAX_PREC)
355                     return 0;
                return setExp(ret.val,ret.exp);
        }
        inline floatexp &operator /=(double a)
        {
360             val/=a;
                align();
                return *this;
        }
        inline floatexp &operator *=(double a)
365     {
                val*=a;
                align();
                return *this;
        }
370     inline floatexp &operator *=(floatexp a)
        {
                return *this = *this * a;
        }
        inline floatexp &operator *=(long double a)
375     {
                return *this *= floatexp(a);
        }

        inline floatexp &operator =(const CFixedFloat &a)
380     {
                signed long int e = 0;
                val = mpfr_get_d_2exp(&e, a.m_f.backend().data(), MPFR_RNDN);
                exp = e;
                align();
385             return *this;
```

```
	}
	inline floatexp(const CFixedFloat &a)
	{
		*this = a;
	}
	inline floatexp(const CDecNumber &a)
	{
		signed long int e = 0;
		val = mpfr_get_d_2exp(&e, a.m_dec.backend().data(), MPFR_RNDN);
		exp = e;
		align();
	}
	inline void ToFixedFloat(CFixedFloat &a) const
	{
		a = val;
		if (exp >= 0)
			mpfr_mul_2ui(a.m_f.backend().data(), a.m_f.backend().↙
			    ↳ data(), exp, MPFR_RNDN);
		else
			mpfr_div_2ui(a.m_f.backend().data(), a.m_f.backend().↙
			    ↳ data(), -exp, MPFR_RNDN);
	}

	inline floatexp setLongDouble(long double a)
	{
		int e = 0;
		val = std::frexp(a, &e);
		exp = e;
		align();
		return *this;
	}
	inline long double toLongDouble() const
	{
		if (val == 0.0L)
			return 0.0L;
		if (exp >= INT_MAX)
			return (val / 0.0L); // infinity
		if (exp <= INT_MIN)
			return (val * 0.0L); // zero
		return std::ldexp((long double) val, exp);
	}
	inline long double toLongDouble(int nScaling) const
	{
		if(!nScaling)
			return toLongDouble();
		floatexp ret = *this;
		// FIXME risky to go higher than this? 1e300 might be ok?
		while(nScaling>99){
			ret.val*=1e100;
			ret.align();
			nScaling-=100;
		}
		while(nScaling>29){
			ret.val*=1e30;
			ret.align();
			nScaling-=30;
		}
```

```
                while(nScaling>9){
                        ret.val*=1e10;
                        ret.align();
                        nScaling-=10;
                }
                while(nScaling>2){
                        ret.val*=1e3;
                        ret.align();
                        nScaling-=3;
                }
                while(nScaling){
                        ret.val*=1e1;
                        ret.align();
                        nScaling--;
                }
                return ret.toLongDouble();
        }
        inline explicit operator long double () const
        {
                return toLongDouble();
        }

  inline std::string toString(int digits = 0) const
  {
                /*
                  f = val 2^exp
                  log10 f = log10 val + exp log10 2
                  e10 = floor(log10 f)
                  d10 = 10^(log10 f - e10)
                  d10 \in [1, 10)
                */
                double lf = std::log10(std::abs(val)) + exp * std::log10(2.0);
                int64_t e10 = std::floor(lf);
                double d10 = std::pow(10, lf - e10) * ((val > 0) - (val < 0));
                if (val == 0) { d10 = 0; e10 = 0; }
                std::ostringstream os; os
                  << std::setprecision(digits ? digits : (std::numeric_limits<
                      ↪ double>::digits10 + 1))
                  << std::fixed
                  << d10 << 'E' << e10;
                return os.str();
        }
};

inline std::ostream& operator<<(std::ostream& a, const floatexp& b)
{
        return a << b.toString();
}

inline floatexp operator*(double a, floatexp b)
{
        return floatexp(a) * b;
}

inline floatexp operator*(floatexp b, double a)
{
        return floatexp(a) * b;
```

```
}

inline floatexp operator*(long double a, floatexp b)
{
        return floatexp(a) * b;
}

inline floatexp operator*(floatexp b, long double a)
{
        return floatexp(a) * b;
}

inline floatexp operator+(double a, floatexp b)
{
        return floatexp(a) + b;
}

inline floatexp operator+(floatexp b, double a)
{
        return floatexp(a) + b;
}

inline floatexp operator*(int a, floatexp b)
{
        return double(a) * b;
}

inline floatexp operator*(floatexp b, int a)
{
        return double(a) * b;
}

inline floatexp abs(floatexp a)
{
        return a.abs();
}

inline floatexp sqrt(floatexp a)
{
  return floatexp
    ( std::sqrt((a.exp & 1) ? 2.0 * a.val : a.val)
    , (a.exp & 1) ? (a.exp - 1) / 2 : a.exp / 2
    );
}

inline floatexp log(floatexp a)
{
        return floatexp(std::log(a.val) + std::log(2.0) * a.exp);
}

inline floatexp log2(floatexp a)
{
        return floatexp(std::log2(a.val) + a.exp);
}

inline floatexp mpfr_get_fe(const mpfr_t value)
{
```

```
        signed long int e = 0;
        double l = mpfr_get_d_2exp(&e, value, MPFR_RNDN);
        return floatexp(l, e);
}

inline void mpfr_set_fe(mpfr_t value, floatexp fe)
{
        mpfr_set_d(value, fe.val, MPFR_RNDN);
        if (fe.exp >= 0)
        {
                mpfr_mul_2ui(value, value, fe.exp, MPFR_RNDN);
        }
        else
        {
                mpfr_div_2ui(value, value, -fe.exp, MPFR_RNDN);
        }
}

inline long double mpfr_get_ld(const mpfr_t value)
{
        using std::ldexp;
        signed long int e = 0;
        long double l = mpfr_get_ld_2exp(&e, value, MPFR_RNDN);
        if (l == 0.0L)
                return 0.0L;
        if (e >= INT_MAX)
                return l / 0.0L;
        if (e <= INT_MIN)
                return l * 0.0L;
        l = ldexp(l, e);
        return l;
}

#endif
```

63 fraktal_sft/floatexp_perturbation.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "fraktal_sft.h"
```

```
#include <float.h>
#include "complex.h"

#include "../formula/formula.h"

BOOL ISFLOATOK(double a);
extern double g_real;
extern double g_imag;
extern double g_FactorAR;
extern double g_FactorAI;
#define _abs(a) ((_abs_val=(a))>0?_abs_val:-_abs_val)
floatexp lb_abs_exp(const floatexp &c, const floatexp &d)
{
        floatexp abs_val, _abs_val, _2=2;
        if (c>0){
                if (c + d>0)
                        abs_val = d;
                else if (d == -c)
                        abs_val = d;
                else if (d<-c)
                        abs_val = -d - _2 * c;
        }
        else if (c == 0)
                abs_val = _abs(d);
        else if (c < 0){
                if (c + d>0)
                        abs_val = d + _2 * c;
                else if (d == -c)
                        abs_val = -d;
                else if (d < -c)
                        abs_val = -d;
        }
        return abs_val;
}

void CFraktalSFT::MandelCalcEXP()
{
        m_bIterChanged = TRUE;
        floatexp Dnr, Dni, yr, yi;
        int64_t antal;
        int x, y, w, h;
        floatexp real(g_real), imag(g_imag), _abs_val;
        floatexp epsilon(m_epsilon);
        const bool derivatives = GetDerivatives();

        while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
                int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
                if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
                        SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, ↙
                            ↳ w, h);
                        if (m_bMirrored)
                                Mirror(x, y);
                        continue;
                }
      if (GuessPixel(x, y, w, h))
        continue;
```

```
                // Series approximation
                floatexp D0r = 0;
80              floatexp D0i = 0;
                floatexp daa = 1;
                floatexp dab = 0;
                floatexp dba = 0;
                floatexp dbb = 1;
85              GetPixelCoordinates(x, y, D0r, D0i, daa, dab, dba, dbb);

                floatexp Dr;
                floatexp Di;
                floatexp dxa1, dxb1, dya1, dyb1;
90              DoApproximation(antal, D0r, D0i, Dr, Di, dxa1, dxb1, dya1, dyb1)↙
                    ↳ ;
                floatexp dr = dxa1;
                floatexp di = dya1;

                double test1 = 0, test2 = 0;
95              bool bGlitch = FALSE;
                int64_t nMaxIter = (m_nGlitchIter<m_nMaxIter ? m_nGlitchIter : ↙
                    ↳ m_nMaxIter);

        if (m_nFractalType == 0 && m_nPower > 10) // FIXME matrix derivatives
                {
100                     if (derivatives)
                        {
                        complex<floatexp> d(dr, di);
                        if (antal<nMaxIter && test1 <= m_nBailout2){
                                for (; antal<nMaxIter; antal++){
105                                     yr = m_dxr[antal] + Dr;
                                        yi = m_dxi[antal] + Di;
                                        test2 = test1;
                                        test1 = (real*yr*yr + imag*yi*yi).↙
                                            ↳ todouble();
                                        if (test1<m_db_z[antal]){
110                                             bGlitch = TRUE;
                                                if (! m_bNoGlitchDetection)
                                                        break;
                                        }
                                        if (test1 > m_nBailout2)
115                                     {
                                                break;
                                        }
                                        complex<floatexp> y(yr, yi);
                                        d = m_nPower * d * (y ^ (m_nPower - 1)) ↙
                                            ↳ + 1;
120                                     complex<floatexp> X(m_dxr[antal], m_dxi[↙
                                            ↳ antal]);
                                        complex<floatexp> D(Dr, Di);
                                        complex<floatexp> D0(D0r, D0i);
                                        complex<floatexp> c(m_pnExpConsts[0], 0)↙
                                            ↳ ;
                                        int nXExp = m_nPower - 2, nDExp = 2, ci ↙
                                            ↳ = 1;
125                                     complex<floatexp> Dn = c*(X^(m_nPower - ↙
                                            ↳ 1))*D;
```

```
                                        while (nXExp){
                                                c.m_r = m_pnExpConsts[ci++];
                                                Dn += c*(X^nXExp)*(D^nDExp);
                                                nXExp--;
130                                             nDExp++;
                                        }
                                        Dn += (D^m_nPower) + D0;
                                        Di = Dn.m_i;
                                        Dr = Dn.m_r;
135                             }
                                Dr = yr;
                                Di = yi;
                        }
                        dr = d.m_r * m_fPixelSpacing;
140                     di = d.m_i * m_fPixelSpacing;
                        } else {
                        if (antal<nMaxIter && test1 <= m_nBailout2){
                                for (; antal<nMaxIter; antal++){
                                        yr = m_dxr[antal] + Dr;
145                                     yi = m_dxi[antal] + Di;
                                        test2 = test1;
                                        test1 = (real*yr*yr + imag*yi*yi).↙
                                            ↪ todouble();
                                        if (test1<m_db_z[antal]){
                                                bGlitch = TRUE;
150                                             if (! m_bNoGlitchDetection)
                                                        break;
                                        }
                                        if (test1 > m_nBailout2)
                                        {
155                                             break;
                                        }
                                        complex<floatexp> y(yr, yi);
                                        complex<floatexp> X(m_dxr[antal], m_dxi[↙
                                            ↪ antal]);
                                        complex<floatexp> D(Dr, Di);
160                                     complex<floatexp> D0(D0r, D0i);
                                        complex<floatexp> c(m_pnExpConsts[0], 0)↙
                                            ↪ ;
                                        int nXExp = m_nPower - 2, nDExp = 2, ci ↙
                                            ↪ = 1;
                                        complex<floatexp> Dn = c*(X^(m_nPower - ↙
                                            ↪ 1))*D;
                                        while (nXExp){
165                                             c.m_r = m_pnExpConsts[ci++];
                                                Dn += c*(X^nXExp)*(D^nDExp);
                                                nXExp--;
                                                nDExp++;
                                        }
170                                     Dn += (D^m_nPower) + D0;
                                        Di = Dn.m_i;
                                        Dr = Dn.m_r;
                                }
                                Dr = yr;
175                             Di = yi;
                        }
                        }
```

```
                }
        else
        {

                        dr *= m_fPixelSpacing;
                        di *= m_fPixelSpacing;
                        bool ok = GetDerivatives()
                          ? perturbation(m_nFractalType, m_nPower, m_dxr, m_dxi,↙
                              ↳  m_db_z, antal, test1, test2, bGlitch, ↙
                              ↳ m_nBailout2, nMaxIter, m_bNoGlitchDetection, ↙
                              ↳ g_real, g_imag, g_FactorAR, g_FactorAI, Dr, Di, ↙
                              ↳ D0r, D0i, dr, di, epsilon, m_fPixelSpacing, daa,↙
                              ↳  dab, dba, dbb)
                          : perturbation(m_nFractalType, m_nPower, m_dxr, m_dxi,↙
                              ↳  m_db_z, antal, test1, test2, bGlitch, ↙
                              ↳ m_nBailout2, nMaxIter, m_bNoGlitchDetection, ↙
                              ↳ g_real, g_imag, g_FactorAR, g_FactorAI, Dr, Di, ↙
                              ↳ D0r, D0i)
                          ;
                        assert(ok && "perturbation_floatexp()");

                }

        complex<double> z((double(Dr)), (double(Di)));
        complex<double> dc((double(dr)), (double(di)));
        complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;

                OutputIterationData(x, y, w, h, bGlitch, antal, test1, test2, de↙
                    ↳ );

                InterlockedIncrement((LPLONG)&m_nDone);
        OutputPixelData(x, y, w, h, bGlitch);
            }
    }
```

64 fraktal_sft/floatexp_reference.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

    #include "fraktal_sft.h"
```

```
#include <float.h>
#include "complex.h"

#include "../formula/formula.h"

extern double g_real;
extern double g_imag;
DWORD WINAPI ThMC2(MC2 *pMC);
DWORD WINAPI ThMC(MC *pMC);
BOOL ISFLOATOK(double a);
extern double g_SeedR;
extern double g_SeedI;
extern double g_FactorAR;
extern double g_FactorAI;

#include "../common/barrier.h"

struct mcthread_common
{
        barrier *barrier;
        mpfr_t xr, xi, xrn, xin, xrn1, xin1, xrxid, xrxid1, sr, si, cr, ci;
        floatexp *m_dxr, *m_dxi;
        double *m_db_z, *terminate, *glitch_threshold;
        int64_t *m_nMaxIter, *m_nGlitchIter, *nMaxIter, *m_nRDone;
        int64_t *antal;
        double *test1;
        double *test2;
        volatile bool *stop;
        floatexp dr, di;
};

struct mcthread
{
        int nType;
        HANDLE hDone;
        mcthread_common *common;
};

static DWORD WINAPI mcthreadfunc(mcthread *p0)
{
        SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
        SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
        bool stored = false;
        double old_absval = 0;
        double abs_val = 0;
        int64_t antal = 0;
        double test1 = 0;
        double test2 = 0;
        const floatexp real(g_real);
        const floatexp imag(g_imag);
        mcthread_common *p = p0->common;
        floatexp dr = p->dr;
        floatexp di = p->di;
        double glitch_threshold = *p->glitch_threshold;
        int64_t i = 0;
        switch (p0->nType)
        {
```

```
                case 0:
                {
80                      for (i = 0; i < *p->nMaxIter; i++)
                        {
                                mpfr_sub(p->xrn1, p->sr, p->si, MPFR_RNDN);
                                mpfr_add(p->xrn, p->cr, p->xrn1, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
85                              mpfr_set(p->xr, p->xrn, MPFR_RNDN);
                                mpfr_sqr(p->sr, p->xrn, MPFR_RNDN);
                                p->m_dxr[i] = mpfr_get_fe(p->xrn);
                                if (p->barrier->wait(p->stop)) break;
                        }
90              }
                break;
                case 1:
                {
                        for (i = 0; i < *p->nMaxIter; i++)
95                      {
                                mpfr_add(p->xin, p->sr, p->si, MPFR_RNDN);
                                mpfr_sub(p->xin1, p->ci, p->xin, MPFR_RNDN);
                                mpfr_add(p->xin, p->xin1, p->xrxid, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
100                             mpfr_set(p->xi, p->xin, MPFR_RNDN);
                                mpfr_sqr(p->si, p->xin, MPFR_RNDN);
                                p->m_dxi[i] = mpfr_get_fe(p->xin);
                                if (p->barrier->wait(p->stop)) break;
                        }
105             }
                break;
                case 2:
                {
                        for (i = 0; i < *p->nMaxIter; i++)
110                     {
                                if (i > 0)
                                {
                                        const floatexp lr = p->m_dxr[i-1];
                                        const floatexp li = p->m_dxi[i-1];
115                                     floatexp drn = 2 * (lr * dr - li * di) +↙
                                            ↳ 1;
                                        floatexp din = 2 * (lr * di + li * dr);
                                        dr = drn;
                                        di = din;
                                        old_absval = abs_val;
120                                     abs_val = (real * lr * lr + imag * li * ↙
                                            ↳ li).todouble();
                                        p->m_db_z[i-1] = abs_val * ↙
                                            ↳ glitch_threshold;
                                        if (abs_val >= 4)
                                        {
                                                if (*p->terminate == 4 && !↙
                                                    ↳ stored)
125                                             {
                                                        stored = true;
                                                        antal = i;
                                                        test1 = abs_val;
                                                        test2 = old_absval;
130                                             }
```

```
                                        }
                                        if (abs_val >= *p->terminate){
                                                if (*p->terminate > 4 && !stored
                                                    ↪ )
                                                {
                                                        stored = true;
                                                        antal = i;
                                                        test1 = abs_val;
                                                        test2 = old_absval;
                                                }
                                                if (*p->nMaxIter == *p->
                                                    ↪ m_nMaxIter)
                                                {
                                                        *p->nMaxIter = i-1 + 3;
                                                        if (*p->nMaxIter > *p->
                                                            ↪ m_nMaxIter)
                                                                *p->nMaxIter = *
                                                                    ↪ p->
                                                                    ↪ m_nMaxIter
                                                                    ↪ ;
                                                        *p->m_nGlitchIter = *p->
                                                            ↪ nMaxIter;
                                                }
                                        }
                                        (*p->m_nRDone)++;
                                }
                                if (p->barrier->wait(p->stop)) break;
                                mpfr_add(p->xrxid1, p->xrn, p->xin, MPFR_RNDN);
                                mpfr_sqr(p->xrxid, p->xrxid1, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
                        }
                }
                break;
        }
        if (p->barrier->wait(p->stop))
        {
                SetEvent(p0->hDone);
                return 0;
        }
        if (p0->nType == 2)
        {
                if (i > 0)
                {
                        const floatexp lr = p->m_dxr[i-1];
                        const floatexp li = p->m_dxi[i-1];
                        floatexp drn = 2 * (lr * dr - li * di) + 1;
                        floatexp din = 2 * (lr * di + li * dr);
                        dr = drn;
                        di = din;
                        old_absval = abs_val;
                        abs_val = (real * lr * lr + imag * li * li).todouble();
                        p->m_db_z[i-1] = abs_val * glitch_threshold;
                        if (abs_val >= 4)
                        {
                                if (*p->terminate == 4 && !stored)
                                {
                                        stored = true;
```

```
                                        antal = i;
                                        test1 = abs_val;
                                        test2 = old_absval;
                                }
185                             }
                                if (abs_val >= *p->terminate){
                                        if (*p->terminate > 4 && !stored)
                                        {
                                                stored = true;
190                                             antal = i;
                                                test1 = abs_val;
                                                test2 = old_absval;
                                        }
                                        if (*p->nMaxIter == *p->m_nMaxIter)
195                                     {
                                                *p->nMaxIter = i-1 + 3;
                                                if (*p->nMaxIter > *p->m_nMaxIter)
                                                        *p->nMaxIter = *p->m_nMaxIter;
                                                *p->m_nGlitchIter = *p->nMaxIter;
200                                     }
                                }
                                (*p->m_nRDone)++;
                        }
                        floatexp xr; xr = mpfr_get_fe(p->xr);
205                     floatexp xi; xi = mpfr_get_fe(p->xi);
                        for (; i < *p->nMaxIter && !*p->stop; i++)
                        {
                                p->m_dxr[i] = xr;
                                p->m_dxi[i] = xi;
210                     }
                        *p->antal = antal;
                        *p->test1 = test1;
                        *p->test2 = test2;
                        p->dr = dr;
215                     p->di = di;
                }
                SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_END);
                SetEvent(p0->hDone);
                mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
220             return 0;
        }


        void CFraktalSFT::CalculateReferenceEXP()
225     {
                const bool derivatives = GetDerivatives();
                Precision prec(m_rref.m_f.precision());

                int64_t i;
230             if (m_dxr)
                        delete[] m_dxr;
                m_dxr = new floatexp[m_nMaxIter];
                if (m_dxi)
                        delete[] m_dxi;
235             m_dxi = new floatexp[m_nMaxIter];
                if (m_db_z)
                        delete[] m_db_z;
```

```
        m_db_z = new double [m_nMaxIter];

240
        floatexp real(g_real);
        floatexp imag(g_imag);

        int64_t antal = 0;
245     double test1 = 0;
        double test2 = 0;
        double xxr = 0, xxi = 0;

        floatexp dr = 1, di = 0;
250
        double terminate = SMOOTH_BAILOUT*SMOOTH_BAILOUT;
        m_nGlitchIter = m_nMaxIter + 1;
        int64_t nMaxIter = m_nMaxIter;
        if (m_nFractalType == 0 && m_nPower == 2 && GetThreadedReference()) // ↙
            ↳ FIXME matrix derivatives, option to disable derivatives
255     {
                double glitch_threshold = 0.0000001;
                if (GetGlitchLowTolerance()) {
                        glitch_threshold = sqrt(glitch_threshold);
                }
260
                mcthread mc[3];
                barrier barrier(3);
                HANDLE hDone[3];
                mcthread_common co;
265       co.barrier = &barrier;
                mp_bitcnt_t bits = mpfr_get_prec(m_rref.m_f.backend().data());
                mpfr_init2(co.xr, bits);
                mpfr_init2(co.xi, bits);
                mpfr_init2(co.xrn, bits);
270             mpfr_init2(co.xin, bits);
                mpfr_init2(co.xrn1, bits);
                mpfr_init2(co.xin1, bits);
                mpfr_init2(co.xrxid, bits);
                mpfr_init2(co.xrxid1, bits);
275             mpfr_init2(co.sr, bits);
                mpfr_init2(co.si, bits);
                mpfr_init2(co.cr, bits);
                mpfr_init2(co.ci, bits);
                mpfr_set(co.cr, m_rref.m_f.backend().data(), MPFR_RNDN);
280             mpfr_set(co.ci, m_iref.m_f.backend().data(), MPFR_RNDN);
                mpfr_set_d(co.xr, g_SeedR, MPFR_RNDN);
                mpfr_set_d(co.xi, g_SeedI, MPFR_RNDN);
                mpfr_sqr(co.sr, co.xr, MPFR_RNDN);
                mpfr_sqr(co.si, co.xi, MPFR_RNDN);
285             mpfr_set_d(co.xrxid, 0, MPFR_RNDN);
                co.m_dxr = m_dxr;
                co.m_dxi = m_dxi;
                co.m_db_z = m_db_z;
                co.terminate = &terminate;
290             co.glitch_threshold = &glitch_threshold;
                co.m_nMaxIter = &m_nMaxIter;
                co.m_nGlitchIter = &m_nGlitchIter;
                co.nMaxIter = &nMaxIter;
```

```
                co.m_nRDone = &m_nRDone;
                co.antal = &antal;
                co.test1 = &test1;
                co.test2 = &test2;
                co.stop = &m_bStop;
                co.dr = dr;
                co.di = di;
                // spawn threads
                for (i = 0; i < 3; i++)
                {
                        mc[i].nType = i;
                        hDone[i] = mc[i].hDone = CreateEvent(NULL, 0, 0, NULL);
                        mc[i].common = &co;
                        HANDLE hThread = CreateThread(NULL, 0, (↙
                            ↳ LPTHREAD_START_ROUTINE) mcthreadfunc, (LPVOID)&mc[↙
                            ↳ i], 0, NULL);
                        CloseHandle(hThread);
                }
                // wait for completion
                WaitForMultipleObjects(3, hDone, TRUE, INFINITE);
                for (i = 0; i < 3; i++){
                        CloseHandle(hDone[i]);
                }
                mpfr_clear(co.xr);
                mpfr_clear(co.xi);
                mpfr_clear(co.xrn);
                mpfr_clear(co.xin);
                mpfr_clear(co.xrn1);
                mpfr_clear(co.xin1);
                mpfr_clear(co.xrxid);
                mpfr_clear(co.xrxid1);
                mpfr_clear(co.sr);
                mpfr_clear(co.si);
                mpfr_clear(co.cr);
                mpfr_clear(co.ci);
        dr = co.dr;
        di = co.di;
                floatexp pixel_spacing = m_fPixelSpacing;
                dr = dr * pixel_spacing;
                di = di * pixel_spacing;
            }
            else if (m_nFractalType == 0 && m_nPower > 10) // FIXME matrix ↙
                ↳ derivatives, option to disable derivatives
            {
                bool stored = false;
                double old_absval = 0;
                double abs_val = 0;
                CFixedFloat xr = g_SeedR, xi = g_SeedI;
                double threashold = 0.0001;
                for (i = 7; i <= m_nPower; i += 2)
                        threashold *= 10;
                if (GetGlitchLowTolerance()) {
                        threashold = sqrt(threashold);
                }
                if (threashold>.5)
                        threashold = .5;
                complex<floatexp> d(1.0, 0.0);
```

```
                for (i = 0; i<nMaxIter && !m_bStop; i++){
                        complex<CFixedFloat> X(xr, xi), r(m_rref, m_iref);
                        complex<CFixedFloat> Xn = (X^m_nPower) + r;
                        floatexp xrf; xrf = xr;
                        floatexp xif; xif = xi;
                        complex<floatexp> x(xrf, xif);
                        d = m_nPower * d * (x ^ (m_nPower - 1)) + 1;
                        xr = Xn.m_r;
                        xi = Xn.m_i;
                        m_dxr[i] = xr;
                        m_dxi[i] = xi;
                        old_absval = abs_val;
                        abs_val = (real * m_dxr[i] * m_dxr[i] + imag * m_dxi[i] ↙
                            ↳ * m_dxi[i]).todouble();
                        m_db_z[i] = abs_val*threashold;
                        if (abs_val >= 4)
                        {
                                if (terminate == 4 && !stored)
                                {
                                        stored = true;
                                        antal = i;
                                        test1 = abs_val;
                                        test2 = old_absval;
                                }
                        }
                        if (abs_val >= terminate){
                                if (terminate > 4 && !stored)
                                {
                                        stored = true;
                                        antal = i;
                                        test1 = abs_val;
                                        test2 = old_absval;
                                }
                                if (nMaxIter == m_nMaxIter){
                                        nMaxIter = i + 3;
                                        if (nMaxIter>m_nMaxIter)
                                                nMaxIter = m_nMaxIter;
                                        m_nGlitchIter = nMaxIter;
                                }
                        }
                        m_nRDone++;
                }
                dr = d.m_r;
                di = d.m_i;
                floatexp pixel_spacing = m_fPixelSpacing;
                dr = dr * pixel_spacing;
                di = di * pixel_spacing;

        }
        else
        {

                floatexp _x, _y, daa, dab, dba, dbb;
                GetPixelCoordinates(g_nAddRefX, g_nAddRefY, _x, _y, daa, dab, ↙
                    ↳ dba, dbb);
                dr *= m_fPixelSpacing;
                di *= m_fPixelSpacing;
```

```
    bool ok = derivatives
      ? reference(m_nFractalType, m_nPower, m_dxr, m_dxi, m_db_z, m_bStop, ↙
          ↳ m_nRDone, m_nGlitchIter, m_nMaxIter, m_rref, m_iref, g_SeedR, ↙
          ↳ g_SeedI, g_FactorAR, g_FactorAI, terminate, g_real, g_imag, ↙
          ↳ GetGlitchLowTolerance(), antal, test1, test2, xxr, xxi, dr, di, daa,↙
          ↳  dab, dba, dbb)
405   : reference(m_nFractalType, m_nPower, m_dxr, m_dxi, m_db_z, m_bStop, ↙
          ↳ m_nRDone, m_nGlitchIter, m_nMaxIter, m_rref, m_iref, g_SeedR, ↙
          ↳ g_SeedI, g_FactorAR, g_FactorAI, terminate, g_real, g_imag, ↙
          ↳ GetGlitchLowTolerance(), antal, test1, test2, xxr, xxi)
      ;
    assert(ok && "reference_floatexp");

        }
410
    complex<double> z((double(xxr)), (double(xxi)));
    complex<double> dc((double(dr)), (double(di)));
    complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;

415     if (0 <= g_nAddRefX && g_nAddRefX < m_nX && 0 <= g_nAddRefY && ↙
            ↳ g_nAddRefY < m_nY)
            OutputIterationData(g_nAddRefX, g_nAddRefY, 1, 1, false, antal ?↙
                ↳  antal + 1 : m_nMaxIter, test1, test2, de);

}
```

65 fraktal_sft/fraktal_sft.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15
You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20  // Kalles Fraktaler 2
    //
    // © 2014-2015 Karl Runmo ,runmo@hotmail.com
    //
    // COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT ↙
        ↳ WARRANTY
25  // OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ↙
        ↳ WARRANTIES
    // THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR ↙
        ↳ PURPOSE
```

```
// OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED
// CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT
// THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY
// SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL
// PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER
// THIS DISCLAIMER.

#include "fraktal_sft.h"
#include "../common/parallell.h"
#include "../common/StringVector.h"
#include "../common/getimage.h"
#include <float.h>
#include <malloc.h>
#include "complex.h"
#include <cmath>
#include <string>
#include <iostream>
#include <set>
#include "../common/bitmap.h"
#include "../formula/formula.h"
#include "colour.h"
#include "jpeg.h"
#include "png.h"
#include "tiff.h"
#include "exr.h"
#include "main.h"
#include "gradient.h"
#include "newton.h"
#include "scale_bitmap.h"

double g_real=1;
double g_imag=1;
double g_SeedR=0;
double g_SeedI=0;
double g_FactorAR=1;
double g_FactorAI=0;
#define _abs(a) ((_abs_val=(a))>0?_abs_val:-_abs_val)
#define _SMOOTH_COLORS_
#define SMOOTH_TOLERANCE 256
int g_nLDBL = 600;
int g_nEXP = 9800;
int g_nRefZero = 3;
#define APPROX_GRID 19
#define TERM4
#define TERM5
//#define TERM6
//#define TERM7
int g_nAddRefX = -1, g_nAddRefY = -1;

double g_Degree = 0;
BOOL g_LDBL = TRUE;
#if 0
```

```
    void(SetParts)(double,double);
    int(SizeOfLD)();
    int(Version)();
    void(AssignInt)(void *p, int nValue);
    void(AssignLD)(void *p, void *ld);
    void(AssignFloatExp)(void *p, floatexp *fe);
    void(ToInt)(void *p, int *pnValue);
    void(ToDouble)(void *p, double *pnDouble);
    void(ToFloatExp)(void *p, floatexp *pnFloatExp);
    void(Multiply)(void *a, void *b, void *ret);
    double(SquareAdd)(void *a, void *b);
    void(Divide)(void *a, void *b, void *ret);
    void(Add)(void *a, void *b, void *ret);
    void(Subtract)(void *a, void *b, void *ret);
    int(GT)(void *a, void *b);
    int(LT)(void *a, void *b);
    int(Equal)(void *a, void *b);
    void(Print)(void *a, char *szRet);
    void(DLLConvertFromFixedFloat)(void *p, const mpfr_t value);
    #define ConvertFromFixedFloat(p,x) DLLConvertFromFixedFloat((p),(x).m_f.backend↙
        ↳ ().data())
    int(Perturbation4)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, void*↙
        ↳  pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, int ↙
        ↳ m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_3rd)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_4th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_5th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_6th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_7th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_8th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_9th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_10th)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch);
    int(Perturbation_Var)(int antal, void *pdxr, void *pdxi, void* pDr, void*pDi, ↙
        ↳ void* pD0r, void*pD0i, double *ptest1, double *ptest2, int m_nBailout2, ↙
        ↳ int m_nMaxIter, double *db_z, BOOL *pGlitch, int nPower, int *nExpConsts);
    int(LDBL_MandelCalc)(int nFractal, int nPower, int antal, void *pdxr, void *pdxi↙
        ↳ , void* pDr, void*pDi, void* pD0r, void*pD0i, double *ptest1, double *↙
        ↳ ptest2, int m_nBailout2, int m_nMaxIter, double *db_z, BOOL *pGlitch,↙
        ↳ double dbFactorAR,double dbFactorAI);
    #endif
```

```
void ErrorText()
{
        TCHAR szMsgBuf[500];

        DWORD dwError = GetLastError();
        FormatMessage(
                FORMAT_MESSAGE_FROM_SYSTEM,
                NULL,
                dwError,
                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                szMsgBuf,
                500,
                NULL );
        MessageBox(NULL,szMsgBuf,"Error",MB_ICONSTOP|MB_OK);
}

CFraktalSFT::CFraktalSFT()
: m_nPixels(0, 0, nullptr, nullptr) // invalid array
{
#ifdef KF_OPENCL
        clid = -1;
        cl = NULL;
#endif

        m_bTexture=FALSE;
        m_nImgMerge=1;
        m_nImgPower=200;
        m_nImgRatio=100;
        m_szTexture="";

        m_bSlopes = FALSE;
        m_nSlopePower = 50;
        m_nSlopeRatio = 50;
        m_nSlopeAngle = 45;
        m_bNoPostWhenDone = FALSE;
        m_scRatio.cx = 640;
        m_scRatio.cy = 360;
        m_nFractalType = 0;
        m_bMirrored = 0;
        m_bMW = 0;
        m_bBlend = 0;
        m_bNoGlitchDetection = FALSE;
        m_nPrevPower = m_nPower = 2;
        m_nMaxOldGlitches = OLD_GLITCH;

        int m_nTerms = GetApproxTerms();
        m_APr = new floatexp[m_nTerms];
        m_APi = new floatexp[m_nTerms];
        m_APs = new SeriesR2;

        m_hMutex = CreateMutex(NULL, 0, NULL);
        m_bRunning = FALSE;
        m_szPosition = NULL;
        m_rstart = -2;
        m_istart = -2;
        m_rstop = 2;
        m_istop = 2;
```

```
            m_dxr = NULL;
            m_dxi = NULL;
            m_ldxr = NULL;
            m_ldxi = NULL;
            m_nZoom = 0;
            m_nTrans = NULL;
            m_nDEx = nullptr;
            m_nDEy = nullptr;
            m_bTrans = TRUE;
            m_bITrans = FALSE;
            m_bAddReference = FALSE;
            m_nXPrev = m_nYPrev = -1;
            m_nSizeImage = -1;
            m_nTotal = -1;
            m_pnExpConsts = NULL;

            m_nBailout = SMOOTH_BAILOUT;
            m_nBailout2 = m_nBailout*m_nBailout;
            m_nSmoothMethod = SmoothMethod_Log;
            m_nColorMethod = ColorMethod_DistanceLog;
            m_nDifferences = Differences_Analytic;

            m_epsilon = 1.1102230246251565e-16 * (1 << 10);

            m_db_dxr = NULL;
            m_db_dxi = NULL;

            m_imageHalf = NULL;
            m_lpBits = NULL;
            m_row = 0;
            m_nMaxIter = 200;
            m_nIterDiv = 0.1;
            memset(m_pOldGlitch, -1, sizeof(m_pOldGlitch));

            m_bInhibitColouring = FALSE;
            m_bInteractive = true;
            GenerateColors(128, 1);
            OpenString(
    "Re: 0\r\n"
    "Im: 0\r\n"
    "Zoom: 1\r\n"
    "Iterations: 200\r\n"
    "IterDiv: 0.010000\r\n"
    "SmoothMethod: 0\r\n"
    "ColorMethod: 7\r\n"
    "Differences: 3\r\n"
    "ColorOffset: 0\r\n"
    "Rotate: 0.000000\r\n"
    "Ratio: 360.000000\r\n"
    "Colors: ↙
        ↳ 255,255,255,128,0,64,160,0,0,192,128,0,64,128,0,0,255,255,64,128,255,0,0,255,\↙
        ↳ r\n"
    "InteriorColor: 0,0,0,\r\n"
    "Smooth: 1\r\n"
    "MultiColor: 0\r\n"
    "BlendMC: 0\r\n"
    "MultiColors: \r\n"
```

```
	"Power: 2\r\n"
	"FractalType: 0\r\n"
	"Slopes: 1\r\n"
	"SlopePower: 50\r\n"
	"SlopeRatio: 20\r\n"
	"SlopeAngle: 45\r\n"
	"imag: 1\r\n"
	"real: 1\r\n"
	"SeedR: 0\r\n"
	"SeedI: 0\r\n"
	"FactorAR: 1\r\n"
	"FactorAI: 0\r\n"
	, FALSE);
		ApplyColors();
	}
	void CFraktalSFT::GenerateColors(int nParts, int nSeed)
	{
		m_nParts = nParts;
		m_nSeed = nSeed;
		if (m_nSeed == -1)
			m_nSeed = GetTickCount();
		srand(m_nSeed);
		m_cKeys[0].r = m_cKeys[0].g = m_cKeys[0].b = m_cKeys[1024].r = m_cKeys[1024].g = m_cKeys[1024].b = 0;
		int i;
		for (i = (m_nSeed == 1 ? 1 : 0); i<1024; i++){
			m_cKeys[i].r = rand() % 256;
			m_cKeys[i].g = rand() % 256;
			m_cKeys[i].b = rand() % 256;
		}
		m_cKeys[m_nParts].r = m_cKeys[0].r;
		m_cKeys[m_nParts].g = m_cKeys[0].g;
		m_cKeys[m_nParts].b = m_cKeys[0].b;
	}
	extern int MakePrime(int n)
	{
		int i;
		int nE = n / 2;
		while (1){
			BOOL bDone = TRUE;
			for (i = 2; i<nE; i++)
			if (n%i == 0){
				bDone = FALSE;
				break;
			}
			if (bDone)
				break;
			n++;
		}
		return n;
	}
	void CFraktalSFT::AddWave(int nColor, int nP, int nS)
	{
		srand(GetTickCount());
		int nPeriod;
		if (nP == -1){
			nPeriod = rand() % (m_nParts<4 ? m_nParts / 2 : m_nParts);
```

```
280                     if (nPeriod == 0)
                                nPeriod = 1;
                }
                else
                        nPeriod = nP;
285             if (nP == -1)
                        nPeriod = MakePrime(nPeriod);
                int nStart;
                if (nS == -1)
                        nStart = rand() % nPeriod;
290             else
                        nStart = nS;
                int i;
                for (i = 0; i<m_nParts; i++){
                        int val = 127 + 127 * sin((double)(i + nStart) * 2 * pi*((double↙
                            ↳ )nPeriod / (double)m_nParts));
295                     switch (nColor){
                        case 0:
                                m_cKeys[i].r = val;
                                if (nP == 0)
                                        m_cKeys[i].r = 0;
300                             break;
                        case 1:
                                m_cKeys[i].g = val;
                                if (nP == 0)
                                        m_cKeys[i].g = 0;
305                             break;
                        case 2:
                                m_cKeys[i].b = val;
                                if (nP == 0)
                                        m_cKeys[i].b = 0;
310                             break;
                        case 3:
                                if (nP){
                                        m_cKeys[i].r = (int)(m_cKeys[i].r + 1 * val) / ↙
                                            ↳ 2;
                                        m_cKeys[i].g = (int)(m_cKeys[i].g + 1 * val) / ↙
                                            ↳ 2;
315                                     m_cKeys[i].b = (int)(m_cKeys[i].b + 1 * val) / ↙
                                            ↳ 2;
                                }
                                break;
                        }
                }
320     }
        void CFraktalSFT::GenerateColors2(int nParts, int nSeed, int nWaves)
        {
                m_nParts = nParts;
                m_nSeed = nSeed;
325             if (m_nSeed == -1)
                        m_nSeed = GetTickCount();
                srand(m_nSeed);
                int i;
                for (i = 0; i<1024; i++)
330                     m_cKeys[i].r = m_cKeys[i].g = m_cKeys[i].b = 0;
                int nW;
                std::set<int> stPeriods;
```

```
        for (nW = 0; nW<nWaves; nW++){
                int nTests, nPeriod;
                for (nTests = 0; nTests<20; nTests++){
                        nPeriod = rand() % (nParts>4 ? nParts / 4 : nParts);
                        if (nPeriod == 0)
                                nPeriod = 1;
                        nPeriod = MakePrime(nPeriod);
                        if (stPeriods.count(nPeriod) != 0)
                                continue;
                        stPeriods.insert(nPeriod);
                }
                int nStart = rand() % nPeriod;
                int nColor = nW % 4;
                if (nW<4){
                        for (i = 0; i<1024; i++){
                                int val = 127 + 127 * sin((double)(i + nStart) *↙
                                    ↪  2 * pi*((double)nPeriod / (double)↙
                                    ↪ m_nParts));
                                switch (nColor){
                                case 0:
                                        m_cKeys[i].r += (int)(m_cKeys[i].r + val↙
                                            ↪ ) % 256;
                                        break;
                                case 1:
                                        m_cKeys[i].g += (int)(m_cKeys[i].g + val↙
                                            ↪ ) % 256;
                                        break;
                                case 2:
                                        m_cKeys[i].b += (int)(m_cKeys[i].b + val↙
                                            ↪ ) % 256;
                                        break;
                                case 3:
                                        m_cKeys[i].r = (int)(m_cKeys[i].r + val)↙
                                            ↪  / 2;
                                        m_cKeys[i].g = (int)(m_cKeys[i].g + val)↙
                                            ↪  / 2;
                                        m_cKeys[i].b = (int)(m_cKeys[i].b + val)↙
                                            ↪  / 2;
                                        break;
                                }
                        }
                }
                else{
                        for (i = 0; i<1024; i++){
                                int val = 127 + 127 * sin((double)(i + nStart) *↙
                                    ↪  2 * pi*((double)nPeriod / (double)↙
                                    ↪ m_nParts));
                                switch (nColor){
                                case 0:
                                        m_cKeys[i].r = (int)(m_cKeys[i].r + val)↙
                                            ↪  / 2;
                                        break;
                                case 1:
                                        m_cKeys[i].g = (int)(m_cKeys[i].g + val)↙
                                            ↪  / 2;
                                        break;
                                case 2:
```

```
                                        m_cKeys[i].b = (int)(m_cKeys[i].b + val)/
                                            \ / 2;
                                        break;
380                                 case 3:
                                        m_cKeys[i].r = (int)(m_cKeys[i].r + val)/
                                            \ / 2;
                                        m_cKeys[i].g = (int)(m_cKeys[i].g + val)/
                                            \ / 2;
                                        m_cKeys[i].b = (int)(m_cKeys[i].b + val)/
                                            \ / 2;
                                        break;
385                                 }
                            }
                        }
                }
        }
390 void CFraktalSFT::ChangeNumOfColors(int nParts)
    {
        m_nParts = nParts;
    }
    int CFraktalSFT::GetNumOfColors()
395 {
        return m_nParts;
    }
    void CFraktalSFT::ApplyIterationColors()
    {
400     if (m_nPixels && m_lpBits){
            int64_t nMin, nMax;
            GetIterations(nMin, nMax);
            if (nMin == nMax)
                nMax = nMin + 1;
405         int x, y;
            for (x = 0; x<m_nX; x++){
                for (y = 0; y<m_nY; y++){
                    int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*/
                        \ m_row;
                    m_lpBits[nIndex] = 255 * (m_nPixels[x][y] - nMin/
                        \ ) / (nMax - nMin);
410                 m_lpBits[nIndex + 1] = m_lpBits[nIndex];
                    m_lpBits[nIndex + 2] = m_lpBits[nIndex];
                }
            }
        }
415 }
    void CFraktalSFT::ApplySmoothColors()
    {
        if (m_nTrans && m_lpBits){
            int x, y;
420         for (x = 0; x<m_nX; x++){
                for (y = 0; y<m_nY; y++){
                    float tr = m_nTrans[x][y];
                    int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*/
                        \ m_row;
                    m_lpBits[nIndex] = 255 * tr;
425                 m_lpBits[nIndex + 1] = 255 * tr;
                    m_lpBits[nIndex + 2] = 255 * tr;
                }
```

```
                }
        }
}

extern void HSVToRGB(double hue, double sat, double bri, COLOR14 &cPos)
{
        hsv a;
        a.h = hue;
        a.s = sat;
        a.v = bri;
        srgb c = hsv2rgb(a);
        c.r *= 255.0;
        c.g *= 255.0;
        c.b *= 255.0;
        cPos.r = (byte) c.r;
        cPos.g = (byte) c.g;
        cPos.b = (byte) c.b;
}

HBITMAP CFraktalSFT::ShrinkBitmap(HBITMAP bmSrc,int nNewWidth,int nNewHeight,int mode)
{
        HDC hDC = GetDC(NULL);
        HBITMAP bmDst = create_bitmap(hDC,nNewWidth,nNewHeight);
  if (mode == 2) // best
        {
                BITMAP s, d;
                GetObject(bmSrc, sizeof(BITMAP), &s);
                GetObject(bmDst, sizeof(BITMAP), &d);
                bool ok = scale_bitmap_rgb8((unsigned char *) d.bmBits, d.bmWidth, d.bmHeight, (const unsigned char *)s.bmBits, s.bmWidth, s.bmHeight);
                //assert(ok && "scale_bitmap_rgb8");
        }
        else
        {
                BITMAP bm;
                GetObject(bmSrc,sizeof(BITMAP),&bm);
                HDC dcSrc = CreateCompatibleDC(hDC);
                HBITMAP bmOldSrc = (HBITMAP)SelectObject(dcSrc,bmSrc);
                HDC dcDst = CreateCompatibleDC(hDC);
                HBITMAP bmOldDst = (HBITMAP)SelectObject(dcDst,bmDst);
                if(mode == 1)
                        SetStretchBltMode(dcDst,HALFTONE);
                else // mode == 0
                        SetStretchBltMode(dcDst,COLORONCOLOR);
                StretchBlt(dcDst,0,0,nNewWidth,nNewHeight,dcSrc,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
                SelectObject(dcDst,bmOldDst);
                SelectObject(dcSrc,bmOldSrc);
                DeleteDC(dcDst);
                DeleteDC(dcSrc);
        }
        ReleaseDC(NULL,hDC);
        return bmDst;
}
void CFraktalSFT::SetTexture(int nIndex, int x, int y)
```

```
{
	int nImgOffs=m_nImgPower/64;
	if(m_lpTextureBits==NULL){
		HBITMAP bmBitmapIn = GetImage(m_szTexture);
485		SIZE scImg;
		scImg.cx = m_nX+(nImgOffs+m_nImgPower)/64;
		scImg.cy = m_nY+(m_nImgPower+nImgOffs)/64;
		HBITMAP bmBitmap = ShrinkBitmap(bmBitmapIn,scImg.cx,scImg.cy);

490		HDC hDC = GetDC(NULL);
		memset(&m_bmiBkg,0,sizeof(BITMAPINFOHEADER));
		m_bmiBkg.biSize=sizeof(BITMAPINFOHEADER);
		if(!GetDIBits(hDC,bmBitmap,0,0,NULL,(LPBITMAPINFO)&m_bmiBkg,↙
		    ↳ DIB_RGB_COLORS))
			Beep(1000,10);
495		m_bmiBkg.biCompression=m_bmiBkg.biClrUsed=m_bmiBkg.↙
		    ↳ biClrImportant=0;
		m_bmiBkg.biBitCount = 24;
		m_rowBkg = ((((m_bmiBkg.biWidth*(DWORD)m_bmiBkg.biBitCount)+31)↙
		    ↳ &~31) >> 3);
		m_bmiBkg.biSizeImage=m_rowBkg*m_bmiBkg.biHeight;
		m_lpTextureBits = new BYTE[m_bmiBkg.biSizeImage];
500		if(!GetDIBits(hDC,bmBitmap,0,m_bmiBkg.biHeight,m_lpTextureBits,
				(LPBITMAPINFO)&m_bmiBkg,DIB_RGB_COLORS))
			Beep(1000,10);
		DeleteObject(bmBitmap);
		DeleteObject(bmBitmapIn);
505		ReleaseDC(NULL,hDC);
	}
	double p1,p2;

	double diffx, diffy;
510	if(x){
		p1 = (double)m_nPixels[x - 1][y] + (double)1 - m_nTrans[x - 1][y↙
		    ↳ ];
		p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y];
	}
	else if(x<m_nX-1){
515		p1 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y];
		p2 = (double)m_nPixels[x+1][y] + (double)1 - m_nTrans[x+1][y];
	}
	else
		p1=p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y];
520	diffx = p1-p2;

	if(y){
		p1 = (double)m_nPixels[x][y-1] + (double)1 - m_nTrans[x][y-1];
		p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y];
525	}
	else if(y<m_nY-1){
		p1 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y];
		p2 = (double)m_nPixels[x][y+1] + (double)1 - m_nTrans[x][y+1];
	}
530	else
		p1=p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y];
	diffy = p1-p2;
```

```
            double diff = diffx*m_nSlopeX + diffy*m_nSlopeY;
535         //p1 = (double)Bmps.ppBits[y][x] + (double)1-Bmps.ppTrans[y][x];
            //p1 = (double)Bmps.ppBits[y][x] + (double)1-Bmps.ppTrans[y][x] - ↙
                ↳ g_nPrevMinIter;
            p1 = 1;
            diff = (p1+diff)/p1;
            diffx = (p1+diffx)/p1;
540         diffy = (p1+diffy)/p1;

            double dbScale=1;
            double nImgPower = m_nImgPower*dbScale;
            diff = pow(diff,nImgPower);
545         diffx = pow(diffx,nImgPower);
            diffy = pow(diffy,nImgPower);
            int nY=y;
            int nX=x;
            if(diff>1){
550             diff = (atan(diff)-pi/4)/(pi/4);
                diff=diff*(double)m_nImgRatio/100;;
            }
            else{
                diff=1/diff;
555             diff = (atan(diff)-pi/4)/(pi/4);
                diff=diff*(double)m_nImgRatio/100;;
            }
            if(diffy>1){
                diffy = (atan(diffy)-pi/4)/(pi/4);
560             diffy=diffy*(double)m_nImgRatio/100;;
                nY = y+nImgOffs + nImgPower*diffy;
            }
            else{
                diffy=1/diffy;
565             diffy = (atan(diffy)-pi/4)/(pi/4);
                diffy=diffy*(double)m_nImgRatio/100;;
                nY = y+nImgOffs - nImgPower*diffy;
            }
            if(diffx>1){
570             diffx = (atan(diffx)-pi/4)/(pi/4);
                diffx=diffx*(double)m_nImgRatio/100;;
                nX = x+nImgOffs + nImgPower*diffx;
            }
            else{
575             diffx=1/diffx;
                diffx = (atan(diffx)-pi/4)/(pi/4);
                diffx=diffx*(double)m_nImgRatio/100;;
                nX = x+nImgOffs - nImgPower*diffx;
            }
580
            int mr = (m_bmiBkg.biWidth)/2+nImgOffs/2;
            int mi = (m_bmiBkg.biHeight)/2+nImgOffs/2;
            nX = (nX - mr)/dbScale + mr;
            nY = (nY - mi)/dbScale + mi;
585
            if(nY<0)
                nY=0;
            else if(nY>m_bmiBkg.biHeight-1)
                nY=m_bmiBkg.biHeight-1;
```

```
590             if(nX<0)
                        nX=0;
                else if(nX>m_bmiBkg.biWidth-1)
                        nX=m_bmiBkg.biWidth-1;
                int nIndexBkg = nX*3 + (m_bmiBkg.biHeight-1-nY)*m_rowBkg;
595             m_lpBits[nIndex] = m_lpTextureBits[nIndexBkg]*m_nImgMerge + m_lpBits[
                    ↪ nIndex]*(1-m_nImgMerge);
                m_lpBits[nIndex+1] = m_lpTextureBits[nIndexBkg+1]*m_nImgMerge + m_lpBits
                    ↪ [nIndex+1]*(1-m_nImgMerge);
                m_lpBits[nIndex+2] = m_lpTextureBits[nIndexBkg+2]*m_nImgMerge + m_lpBits
                    ↪ [nIndex+2]*(1-m_nImgMerge);
        }

600     static inline double sqr(double x) { return x * x; }
        static inline double hypot2(double x, double y) { return x * x + y * y; }
        static inline double hypot1(double x, double y) { return sqrt(x * x + y * y); }

        void CFraktalSFT::SetColor(int nIndex, const int64_t nIter0, double offs, int x,
            ↪ int y, int w, int h)
605     {
                if (m_bInhibitColouring) return;
                srgb s;
                s.r = 0;
                s.g = 0;
610             s.b = 0;
                if (!GetShowGlitches() && GET_TRANS_GLITCH(offs))
                        return;
                if (nIter0 == m_nMaxIter)
                {
615                     s.r = s.g = s.b = 1;
                }
                else{
                        ColorMethod method = m_nColorMethod;
                        Differences diffs = m_nDifferences;
620                     int64_t nIter = nIter0;

                        double iter = (double)nIter + (double)1 - offs;
                        /*              if(1){//DE
                        double p1, p2;
625                     if(x){
                        p1 = (double)m_nPixels[x-1][y] + (double)1-m_nTrans[x-1][y];
                        p2 = (double)m_nPixels[x][y] + (double)1-m_nTrans[x][y];
                        }
                        else if(x<m_nX-1){
630                     p1 = (double)m_nPixels[x][y] + (double)1-m_nTrans[x][y];
                        p2 = (double)m_nPixels[x+1][y] + (double)1-m_nTrans[x+1][y];
                        }
                        else
                        p1=p2 = (double)m_nPixels[x][y] + (double)1-m_nTrans[x][y];
635                     iter = (p1>p2?p1-p2:p2-p1);
                        }
                        */
                        if (method == ColorMethod_SquareRoot){
                                iter = sqrt(fmax(0, iter));
640                     }
                        else if (method == ColorMethod_CubicRoot){
                                iter = pow(fmax(0, iter), (double)1 / (double)3);
```

```
		}
		else if (method == ColorMethod_Logarithm){
645			iter = log(fmax(1, iter));
		}
		else if (method == ColorMethod_LogLog){
			iter = log(1 + log(1 + iter));
		}
650		else if (method == ColorMethod_ATan){
			iter = atan(iter);
		}
		else if (method == ColorMethod_Stretched){
			int64_t nMin, nMax;
655			GetIterations(nMin, nMax,NULL,NULL,TRUE);
			iter = (double)1024 * ((double)iter - (double)nMin) / ((↙
			    ↳ double)nMax - (double)nMin);
		}
		else if (method == ColorMethod_DistanceLinear ||
			 method == ColorMethod_DEPlusStandard ||
660			 method == ColorMethod_DistanceLog ||
			 method == ColorMethod_DistanceSqrt){
			iter=0;
			if (diffs == Differences_Analytic)
			{
665				iter = 0;
				if (m_nDEx && m_nDEy)
				{
					float dex = m_nDEx[x][y];
					float dey = m_nDEy[x][y];
670					iter = 1 / sqrt(dex * dex + dey * dey);
				}
			}
			else
			{
675				// load 3x3 stencil around the pixel
				int X = m_nX - 1;
				int Y = m_nY - 1;
				static const double ninf = 1.0 / 0.0;
				double p[3][3] = { { ninf, ninf, ninf }, { ninf,↙
				    ↳  ninf, ninf }, { ninf, ninf, ninf } };
680				double px[3][3] = { { 0, 0, 0 }, { 0, 0, 0 }, { ↙
				    ↳ 0, 0, 0 } };
				double py[3][3] = { { 0, 0, 0 }, { 0, 0, 0 }, { ↙
				    ↳ 0, 0, 0 } };
				if (w <= x && h <= y && m_nPixels[x - w][y - h] ↙
				    ↳ != PIXEL_UNEVALUATED){p[0][0] = m_nPixels[↙
				    ↳ x - w][y - h] + 1.0 - m_nTrans[x - w][y - ↙
				    ↳ h];GetPixelOffset(x - w, y - h, px[0][0], ↙
				    ↳ py[0][0]); px[0][0] -= w; py[0][0] -= h;}
				if (w <= x           && m_nPixels[x - w][y    ] ↙
				    ↳ != PIXEL_UNEVALUATED){p[0][1] = m_nPixels[↙
				    ↳ x - w][y    ] + 1.0 - m_nTrans[x - w][y ↙
				    ↳    ];GetPixelOffset(x - w, y    , px↙
				    ↳ [0][1], py[0][1]); px[0][1] -= w; ↙
				    ↳                }
				if (w <= x && y+h<=Y && m_nPixels[x - w][y + h] ↙
				    ↳ != PIXEL_UNEVALUATED){p[0][2] = m_nPixels[↙
				    ↳ x - w][y + h] + 1.0 - m_nTrans[x - w][y + ↙
```

```
    ↪ h];GetPixelOffset(x - w, y + h, px[0][2], ↙
    ↪ py[0][2]); px[0][2] -= w; py[0][2] += h;}
685 if (          h <= y && m_nPixels[x    ][y - h] ↙
    ↪ != PIXEL_UNEVALUATED){p[1][0] = m_nPixels[↙
    ↪ x    ][y - h] + 1.0 - m_nTrans[x    ][y - ↙
    ↪ h];GetPixelOffset(x    , y - h, px[1][0], ↙
    ↪ py[1][0]);                  py[1][0] -= h;}
    if (                    m_nPixels[x    ][y    ] ↙
    ↪ != PIXEL_UNEVALUATED){p[1][1] = m_nPixels[↙
    ↪ x    ][y    ] + 1.0 - m_nTrans[x    ][y ↙
    ↪    ];GetPixelOffset(x    , y    , px↙
    ↪ [1][1], py[1][1]); ↙
    ↪                                    }
    if (          y+h<=Y && m_nPixels[x    ][y + h] ↙
    ↪ != PIXEL_UNEVALUATED){p[1][2] = m_nPixels[↙
    ↪ x    ][y + h] + 1.0 - m_nTrans[x    ][y + ↙
    ↪ h];GetPixelOffset(x    , y + h, px[1][2], ↙
    ↪ py[1][2]);                  py[1][2] += h;}
    if (x+w<=X && h <= y && m_nPixels[x + w][y - h] ↙
    ↪ != PIXEL_UNEVALUATED){p[2][0] = m_nPixels[↙
    ↪ x + w][y - h] + 1.0 - m_nTrans[x + w][y - ↙
    ↪ h];GetPixelOffset(x + w, y - h, px[2][0], ↙
    ↪ py[2][0]); px[2][0] += w; py[2][0] -= h;}
    if (x+w<=X           && m_nPixels[x + w][y    ] ↙
    ↪ != PIXEL_UNEVALUATED){p[2][1] = m_nPixels[↙
    ↪ x + w][y    ] + 1.0 - m_nTrans[x + w][y ↙
    ↪    ];GetPixelOffset(x + w, y    , px↙
    ↪ [2][1], py[2][1]); px[2][1] += w; ↙
    ↪                 }
690 if (x+w<=X && y+h<=Y && m_nPixels[x + w][y + h] ↙
    ↪ != PIXEL_UNEVALUATED){p[2][2] = m_nPixels[↙
    ↪ x + w][y + h] + 1.0 - m_nTrans[x + w][y + ↙
    ↪ h];GetPixelOffset(x + w, y + h, px[2][2], ↙
    ↪ py[2][2]); px[2][2] += w; py[2][2] += h;}
    // reflect at boundaries if necessary
    // this will break (result is infinite or NaN) ↙
    ↪ for image size of 1 pixel
    p[1][1] *= 2.0;
    px[1][1] *= 2.0;
695 py[1][1] *= 2.0;
    if (ninf == p[0][0]) {p[0][0] = p[1][1] - p↙
    ↪ [2][2];px[0][0] = px[1][1] - px[2][2];py↙
    ↪ [0][0] = py[1][1] - py[2][2];}
    if (ninf == p[0][1]) {p[0][1] = p[1][1] - p↙
    ↪ [2][1];px[0][1] = px[1][1] - px[2][1];py↙
    ↪ [0][1] = py[1][1] - py[2][1];}
    if (ninf == p[0][2]) {p[0][2] = p[1][1] - p↙
    ↪ [2][0];px[0][2] = px[1][1] - px[2][0];py↙
    ↪ [0][2] = py[1][1] - py[2][0];}
    if (ninf == p[1][0]) {p[1][0] = p[1][1] - p↙
    ↪ [1][2];px[1][0] = px[1][1] - px[1][2];py↙
    ↪ [1][0] = py[1][1] - py[1][2];}
700 if (ninf == p[1][2]) {p[1][2] = p[1][1] - p↙
    ↪ [1][0];px[1][2] = px[1][1] - px[1][0];py↙
    ↪ [1][2] = py[1][1] - py[1][0];}
    if (ninf == p[2][0]) {p[2][0] = p[1][1] - p↙
    ↪ [0][2];px[2][0] = px[1][1] - px[0][2];py↙
```

```
                    ↪ [2][0] = py[1][1] - py[0][2];}
                if (ninf == p[2][1]) {p[2][1] = p[1][1] - p↙
                    ↪ [0][1];px[2][1] = px[1][1] - px[0][1];py↙
                    ↪ [2][1] = py[1][1] - py[0][1];}
                if (ninf == p[2][2]) {p[2][2] = p[1][1] - p↙
                    ↪ [0][0];px[2][2] = px[1][1] - px[0][0];py↙
                    ↪ [2][2] = py[1][1] - py[0][0];}
                p[1][1] *= 0.5;
705             px[1][1] *= 0.5;
                py[1][1] *= 0.5;
                // do the differencing
                switch (diffs)
                {
710             case Differences_Analytic:
                        //assert(!"analytic case reached, should↙
                            ↪ be unreachable");
                        break;
                    case Differences_Laplacian3x3:
                        {
715                             // gerrit: Laplacian is ↙
                                ↪ proportional to g^2: https↙
                                ↪ ://fractalforums.org/↙
                                ↪ kalles-fraktaler/15/↙
                                ↪ gaussian-jitter-for-moire-↙
                                ↪ reduction/891/msg5563#↙
                                ↪ msg5563
                                double L = 0;
                                L +=  1 * p[0][0];
                                L +=  4 * p[0][1];
                                L +=  1 * p[0][2];
720                             L +=  4 * p[1][0];
                                L -= 20 * p[1][1];
                                L +=  4 * p[1][2];
                                L +=  1 * p[2][0];
                                L +=  4 * p[2][1];
725                             L +=  1 * p[2][2];
                                L /=  6;
#define INV_LOG_2 1.4426950408889634
                                double g = sqrt(fabs(L * ↙
                                    ↪ INV_LOG_2));
#undef INV_LOG_2
730                             iter = g * 2.8284271247461903;
                        }
                        break;
                    case Differences_LeastSquares3x3:
                        {
735                             double gx = 0;
                                double gy = 0;
                                compute_gradient_3x3(p, px, py, ↙
                                    ↪ gx, gy);
                                double g = hypot1(gx, gy);
                                iter = g * 2.8284271247461903;
740                     }
                        break;
                    case Differences_LeastSquares2x2:
                        {
                                double gx = 0;
```

```
745                                     double gy = 0;
                                        compute_gradient_2x2(p, px, py, ↙
                                            ↳ gx, gy);
                                        double g = hypot1(gx, gy);
                                        iter = g * 2.8284271247461903;
                                    }
750                                 break;
                                case Differences_Central3x3:
                                    {
                                        // gerrit's central difference ↙
                                            ↳ formula
                                        double gx = sqr(p[2][1] - p↙
                                            ↳ [0][1]) / hypot2(px[2][1] ↙
                                            ↳ - px[0][1], py[2][1] - py↙
                                            ↳ [0][1]);
755                                     double gy = sqr(p[1][2] - p↙
                                            ↳ [1][0]) / hypot2(px[1][2] ↙
                                            ↳ - px[1][0], py[1][2] - py↙
                                            ↳ [1][0]);
                                        double g1 = sqr(p[2][2] - p↙
                                            ↳ [0][0]) / hypot2(px[2][2] ↙
                                            ↳ - px[0][0], py[2][2] - py↙
                                            ↳ [0][0]);
                                        double g2 = sqr(p[0][2] - p↙
                                            ↳ [2][0]) / hypot2(px[0][2] ↙
                                            ↳ - px[2][0], py[0][2] - py↙
                                            ↳ [2][0]);
                                        double g = sqrt(0.5 * (gx + gy +↙
                                            ↳  g1 + g2));
                                        iter = g * 2.8284271247461903;
760                                 }
                                    break;
                                case Differences_Forward3x3:
                                    {
                                        // forward differencing in 8 ↙
                                            ↳ directions from the target↙
                                            ↳  point
765                                     double gx0 = sqr(p[0][1] - p↙
                                            ↳ [1][1]) / hypot2(px[0][1] ↙
                                            ↳ - px[1][1], py[0][1] - py↙
                                            ↳ [1][1]);
                                        double gx2 = sqr(p[2][1] - p↙
                                            ↳ [1][1]) / hypot2(px[2][1] ↙
                                            ↳ - px[1][1], py[2][1] - py↙
                                            ↳ [1][1]);
                                        double gy0 = sqr(p[1][0] - p↙
                                            ↳ [1][1]) / hypot2(px[1][0] ↙
                                            ↳ - px[1][1], py[1][0] - py↙
                                            ↳ [1][1]);
                                        double gy2 = sqr(p[1][2] - p↙
                                            ↳ [1][1]) / hypot2(px[1][2] ↙
                                            ↳ - px[1][1], py[1][2] - py↙
                                            ↳ [1][1]);
                                        double gu0 = sqr(p[0][0] - p↙
                                            ↳ [1][1]) / hypot2(px[0][0] ↙
                                            ↳ - px[1][1], py[0][0] - py↙
                                            ↳ [1][1]);
```

```
770                                     double gu2 = sqr(p[2][2] - p↙
                                            ↳ [1][1]) / hypot2(px[2][2] ↙
                                            ↳ - px[1][1], py[2][2] - py↙
                                            ↳ [1][1]);
                                        double gv0 = sqr(p[2][0] - p↙
                                            ↳ [1][1]) / hypot2(px[2][0] ↙
                                            ↳ - px[1][1], py[2][0] - py↙
                                            ↳ [1][1]);
                                        double gv2 = sqr(p[0][2] - p↙
                                            ↳ [1][1]) / hypot2(px[0][2] ↙
                                            ↳ - px[1][1], py[0][2] - py↙
                                            ↳ [1][1]);
                                        double g = sqrt(0.25 * (gx0 + ↙
                                            ↳ gx2 + gy0 + gy2 + gu0 + ↙
                                            ↳ gu2 + gv0 + gv2));
                                        iter = g * 2.8284271247461903;
775                                 }
                                    break;
                                case Differences_Diagonal2x2: // aka Roberts ↙
                                    ↳ Cross
                                    {
                                        // forward differencing in 2 ↙
                                            ↳ diagonals of a 2x2 ↙
                                            ↳ substencil
780                                     if (GetJitterSeed() == 0)
                                        {
                                            double gu = sqr(p[0][0] ↙
                                                ↳ - p[1][1]) / ↙
                                                ↳ hypot2(px[0][0] - ↙
                                                ↳ px[1][1], py[0][0]↙
                                                ↳  - py[1][1]);
                                            double gv = sqr(p[0][1] ↙
                                                ↳ - p[1][0]) / ↙
                                                ↳ hypot2(px[0][1] - ↙
                                                ↳ px[1][0], py[0][1]↙
                                                ↳  - py[1][0]);
                                            double g = sqrt(gu + gv)↙
                                                ↳ ;
785                                         iter = g * ↙
                                                ↳ 2.8284271247461903;↙
                                                ↳
                                        }
                                        else
                                        {
                                            // with displacement ↙
                                                ↳ correction by ↙
                                                ↳ gerrit
790                                         double nux = px[0][0] - ↙
                                                ↳ px[1][1];
                                            double nuy = py[0][0] - ↙
                                                ↳ py[1][1];
                                            double nvx = px[1][0] - ↙
                                                ↳ px[0][1];
                                            double nvy = py[1][0] - ↙
                                                ↳ py[0][1];
                                            double nu = hypot1(nux, ↙
                                                ↳ nuy);
```

```
795                                                     double nv = hypot1(nvx, ↙
                                                            ↳ nvy);
                                                        nux /= nu;
                                                        nuy /= nu;
                                                        nvx /= nv;
                                                        nvy /= nv;
800                                                     double u = (p[0][0] - p↙
                                                            ↳ [1][1]) / nu;
                                                        double v = (p[1][0] - p↙
                                                            ↳ [0][1]) / nv;
                                                        double dotnunv = nux * ↙
                                                            ↳ nvx + nuy * nvy;
                                                        double crossnunv = nux *↙
                                                            ↳  nvy - nuy * nvx;
                                                        double g = sqrt((u * u +↙
                                                            ↳  v * v - 2 * u * v↙
                                                            ↳  * dotnunv) / sqr(↙
                                                            ↳ crossnunv));
805                                                     iter = g * ↙
                                                            ↳ 2.8284271247461903;↙
                                                            ↳
                                                }
                                        }
                                        break;
                                case Differences_Traditional:
810                                     {
                                                // traditional method reverse ↙
                                                    ↳ engineered from original ↙
                                                    ↳ code
                                                double gx = (p[0][1] - p[1][1]) ↙
                                                    ↳ * 1.414 / hypot(px[0][1] -↙
                                                    ↳  px[1][1], py[0][1] - py↙
                                                    ↳ [1][1]);
                                                double gy = (p[1][0] - p[1][1]) ↙
                                                    ↳ * 1.414 / hypot(px[1][0] -↙
                                                    ↳  px[1][1], py[1][0] - py↙
                                                    ↳ [1][1]);
                                                double gu = (p[0][0] - p[1][1]) ↙
                                                    ↳ * 1.414 / hypot(px[0][0] -↙
                                                    ↳  px[1][1], py[0][0] - py↙
                                                    ↳ [1][1]);
815                                             double gv = (p[0][2] - p[1][1]) ↙
                                                    ↳ * 1.414 / hypot(px[0][2] -↙
                                                    ↳  px[1][1], py[0][2] - py↙
                                                    ↳ [1][1]);
                                                double g = fabs(gx) + fabs(gy) +↙
                                                    ↳  fabs(gu) + fabs(gv);
                                                iter = g;
                                        }
                                        break;
820                             }
                        }
                        // post differencing transfer functions
    //                  iter/=4;
    //                  iter*=iter;
825                     iter*=(double)m_nX / (double)640;
                        if (method == ColorMethod_DistanceSqrt || method == ↙
```

```
                    ↳ ColorMethod_DEPlusStandard)
                        iter=sqrt(fmax(0, iter));
                else if (method == ColorMethod_DistanceLog)
                        iter=log(fmax(1, iter+1));
                /*iter=log(iter);
                if(iter<0)
                        iter=0;*/
                if(iter>1024)
                        iter=1024;
                if(method == ColorMethod_DEPlusStandard && iter>↙
                    ↳ m_nIterDiv)
                        iter = (double)nIter + (double)1 - offs;
        }
        if (m_nIterDiv != 1){
                iter /= m_nIterDiv;
        }
        if (m_nColorOffset)
                iter += m_nColorOffset;// = (nIter+m_nColorOffset)%1024;
        nIter = (int64_t)floor(iter);
        offs = 1 - (iter - (double)nIter);
        if (m_bITrans)
                offs = 1 - offs;
        if (mbMW){
                double nH = 0, nS = 0, nB = 0;
                int nDR = 0, nDG = 0, nDB = 0;
                int i;
                for (i = 0; i<mnMW; i++){
                        double nPeriod;
                        nPeriod = mMW[i].nPeriod;
                        double g;
                        if (m_bTrans)
                                g = sin((pi*iter) / nPeriod) / 2 + .5;
                        else
                                g = sin((pi*((int)iter)) / nPeriod) / 2 ↙
                                    ↳ + .5;
                        if (nPeriod<0)
                                g = -(double)nPeriod / (double)100;
                        if (mMW[i].nType == 0){
                                nH += g;
                                nDR++;
                        }
                        if (mMW[i].nType == 1){
                                nS += g;
                                nDG++;
                        }
                        if (mMW[i].nType == 2){
                                nB += g;
                                nDB++;
                        }
                }
                if (nDR)
                        nH /= nDR;
                if (nDG)
                        nS /= nDG;
                if (nDB)
                        nB /= nDB;
                hsv nHSV;
```

```
                        nHSV.h = nH;
                        nHSV.s = nS;
                        nHSV.v = nB;
                        srgb nRGB = hsv2rgb(nHSV);
885                     if (m_bBlend){
                                double nR, nG, nB;
                                if (m_bTrans && offs){
                                        double g1 = (1 - offs);
                                        int col = ((nIter % 1024) + 1024) % ↙
                                            ↳ 1024;
890                                     int ncol = (col + 1) % 1024;
                                        nR = m_cPos[col].r*offs + m_cPos[ncol].r↙
                                            ↳ *g1;
                                        nG = m_cPos[col].g*offs + m_cPos[ncol].g↙
                                            ↳ *g1;
                                        nB = m_cPos[col].b*offs + m_cPos[ncol].b↙
                                            ↳ *g1;
                                }
895                             else{
                                        int col = ((nIter % 1024) + 1024) % ↙
                                            ↳ 1024;
                                        nR = m_cPos[col].r;//+n;
                                        nG = m_cPos[col].g;//+n;
                                        nB = m_cPos[col].b;//+n;
900                             }
                                srgb nRGB2;
                                nRGB2.r = nR / 255.0f;
                                nRGB2.g = nG / 255.0f;
                                nRGB2.b = nB / 255.0f;
905                             s.r = (nRGB.r + nRGB2.r) * 0.5f;
                                s.g = (nRGB.g + nRGB2.g) * 0.5f;
                                s.b = (nRGB.b + nRGB2.b) * 0.5f;
                        }
                        else
910                     {
                                s = nRGB;
                        }
                }
                else{
915                     if (m_bTrans && offs){
                                double g1 = (1 - offs);
                                int col = ((nIter % 1024) + 1024) % 1024;
                                int ncol = (col + 1) % 1024;
                                s.r = (m_cPos[col].r*offs + m_cPos[ncol].r*g1) /↙
                                    ↳  255.0f;
920                             s.g = (m_cPos[col].g*offs + m_cPos[ncol].g*g1) /↙
                                    ↳  255.0f;
                                s.b = (m_cPos[col].b*offs + m_cPos[ncol].b*g1) /↙
                                    ↳  255.0f;
                        }
                        else{
                                int col = ((nIter % 1024) + 1024) % 1024;
925                             s.r = m_cPos[col].r / 255.0f;
                                s.g = m_cPos[col].g / 255.0f;
                                s.b = m_cPos[col].b / 255.0f;
                        }
                }
```

```
930             }
                if(m_bTexture)
                        SetTexture(nIndex,x,y);
                if (m_bSlopes){
                        double diffx, diffy;
935                     if (m_nDifferences == Differences_Analytic)
                        {
                                complex<float> de(m_nDEx[x][y], m_nDEy[x][y]);
                                complex<float> diff = complex<float>(1) / de;
                                diffx = diff.m_r;
940                             diffy = diff.m_i;
                        }
                        else
                        {
                        double p1, p2;
945                     /*              if(x && y){
                        p1 = (double)m_nPixels[x-1][y-1] + (double)1-m_nTrans[x-1][y-1];
                        p2 = (double)m_nPixels[x][y] + (double)1-m_nTrans[x][y];
                        }
                        else if(x<m_nX-1 && y<m_nY-1){
950                     p1 = (double)m_nPixels[x][y] + (double)1-m_nTrans[x][y];
                        p2 = (double)m_nPixels[x+1][y+1] + (double)1-m_nTrans[x+1][y+1];
                        }
                        else
                        p1=p2 = (double)m_nPixels[x][y] + (double)1-m_nTrans[x][y];
955                     double diffCompare = p1/p2;
                        */
                        if (w <= x && m_nPixels[x - w][y] != PIXEL_UNEVALUATED){
                                p1 = (double)m_nPixels[x - w][y] + (double)1 - m_nTrans[↙
                                    ↳ x - w][y];
                                p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y↙
                                    ↳ ];
960                     }
                        else if (x+w<m_nX && m_nPixels[x + w][y] != PIXEL_UNEVALUATED){
                                p1 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y↙
                                    ↳ ];
                                p2 = (double)m_nPixels[x + w][y] + (double)1 - m_nTrans[↙
                                    ↳ x + w][y];
                        }
965                     else
                                p1 = p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans↙
                                    ↳ [x][y];
                        diffx = (p1 - p2) / w;
                        if (h <= y && m_nPixels[x][y - h] != PIXEL_UNEVALUATED){
                                p1 = (double)m_nPixels[x][y - h] + (double)1 - m_nTrans[↙
                                    ↳ x][y - h];
970                             p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y↙
                                    ↳ ];
                        }
                        else if (y+h<m_nY && m_nPixels[x][y + h] != PIXEL_UNEVALUATED){
                                p1 = (double)m_nPixels[x][y] + (double)1 - m_nTrans[x][y↙
                                    ↳ ];
                                p2 = (double)m_nPixels[x][y + h] + (double)1 - m_nTrans[↙
                                    ↳ x][y + h];
975                     }
                        else
                                p1 = p2 = (double)m_nPixels[x][y] + (double)1 - m_nTrans↙
```

```
                        ↪ [x][y];
                diffy = (p1 - p2) / h;
        }
980             double diff = diffx*m_nSlopeX + diffy*m_nSlopeY;
                double p1 = fmax(1, (double)m_nPixels[x][y] + (double)1 - ↙
                    ↪ m_nTrans[x][y]);
                diff = (p1 + diff) / p1;
                diff = pow(diff, (double)m_nSlopePower*(double)(m_nZoom*1.75 + ↙
                    ↪ 1)*(double)m_nX / (double)640);
                if (diff>1){
985                     diff = (atan(diff) - pi / 4) / (pi / 4);
                        diff = diff*(double)m_nSlopeRatio / 100;;
                        s.r = (1 - diff)*s.r;
                        s.g = (1 - diff)*s.g;
                        s.b = (1 - diff)*s.b;
990             }
                else{
                        diff = 1 / diff;
                        diff = (atan(diff) - pi / 4) / (pi / 4);
                        diff = diff*(double)m_nSlopeRatio / 100;;
995                     s.r = (1 - diff)*s.r + diff;
                        s.g = (1 - diff)*s.g + diff;
                        s.b = (1 - diff)*s.b + diff;
                }
        }
1000    lrgb l;
        if (m_imageHalf)
                l = srgb2lrgb(s);
        srgb8 s8 = dither(s, x, y);
        if (nIter0 == m_nMaxIter)
1005    {
                s8.r = m_cInterior.r;
                s8.g = m_cInterior.g;
                s8.b = m_cInterior.b;
                if (m_imageHalf)
1010            {
                        s.r = s8.r / 255.0f;
                        s.g = s8.g / 255.0f;
                        s.b = s8.b / 255.0f;
                        l = srgb2lrgb(s);
1015            }
        }
        m_lpBits[nIndex    ] = s8.r;
        m_lpBits[nIndex + 1] = s8.g;
        m_lpBits[nIndex + 2] = s8.b;
1020    if (m_imageHalf)
        {
                // flip vertically and convert BGR to RGB
                size_t i3 = (nIndex % m_row);
                size_t j = nIndex / m_row;
1025            size_t nIndex2 = (m_nY - 1 - j) * m_row + i3;
                float c; // clamp to finite non-negative
                c = l.b; if (! (c > 0)) c = 0; if (! (c < 65504)) c = 65504; ↙
                    ↪ m_imageHalf[nIndex2    ] = c;
                c = l.g; if (! (c > 0)) c = 0; if (! (c < 65504)) c = 65504; ↙
                    ↪ m_imageHalf[nIndex2 + 1] = c;
                c = l.r; if (! (c > 0)) c = 0; if (! (c < 65504)) c = 65504; ↙
```

```
                        ↪ m_imageHalf[nIndex2 + 2] = c;
1030            }
        }

        // based on cache-oblivious matrix transpose by divide and conquer
        void CFraktalSFT::ApplyColors(int x0, int x1, int y0, int y1)
1035    {
                int w = x1 - x0;
                int h = y1 - y0;
                //assert(w > 0);
                //assert(h > 0);
1040            if (w <= 16 && h <= 16)
                {
                        for (int x = x0; x < x1; ++x)
                        for (int y = y0; y < y1; ++y)
                        {
1045                            int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
                                SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, ↙
                                    ↪ 1, 1);
                        }
                }
                else if (w <= h)
1050            {
                        int y = (y0 + y1) >> 1;
                        ApplyColors(x0, x1, y0, y);
                        ApplyColors(x0, x1, y, y1);
                }
1055            else
                {
                        int x = (x0 + x1) >> 1;
                        ApplyColors(x0, x, y0, y1);
                        ApplyColors(x, x1, y0, y1);
1060            }
        }

        static int ThApplyColors(TH_PARAMS *pMan)
        {
1065            pMan->p->ApplyColors(pMan->nXStart, pMan->nXStop, 0, pMan->p->GetHeight↙
                    ↪ ());
                return 0;
        }

        void CFraktalSFT::ApplyColors()
1070    {
                int i, p = 0;
                for (i = 0; i<1024; i++){
                        double temp = (double)i*(double)m_nParts / (double)1024;
                        p = (int)temp;
1075                    int pn = (p + 1) % m_nParts;
                        temp -= p;
                        temp = sin((temp - .5)*pi) / 2 + .5;
                        m_cPos[i].r = (unsigned char)(temp*m_cKeys[pn].r + (1 - temp)*↙
                            ↪ m_cKeys[p].r);
                        m_cPos[i].g = (unsigned char)(temp*m_cKeys[pn].g + (1 - temp)*↙
                            ↪ m_cKeys[p].g);
1080                    m_cPos[i].b = (unsigned char)(temp*m_cKeys[pn].b + (1 - temp)*↙
                            ↪ m_cKeys[p].b);
```

```
        }
        if (m_nPixels && m_lpBits && ! m_bInhibitColouring){
                SYSTEM_INFO sysinfo;
                GetSystemInfo(&sysinfo);
1085            int nParallel = GetThreadsPerCore() * sysinfo.↙
                    ↳ dwNumberOfProcessors - GetThreadsReserveCore();
                if (nParallel < 1) nParallel = 1;
                CParallell P(
#ifdef _DEBUG
                        1
1090 #else
                        nParallel
#endif
                        );
                TH_PARAMS *pMan = new TH_PARAMS[nParallel];
1095            int nXStart = 0;
                int nXStep = (m_nX + nParallel - 1) / nParallel;
                for (i = 0; i < nParallel; i++)
                {
                        pMan[i].p = this;
1100                    pMan[i].nXStart = nXStart;
                        nXStart += nXStep;
                        if (nXStart > m_nX) nXStart = m_nX;
                        pMan[i].nXStop = nXStart;
                        P.AddFunction((LPEXECUTE)ThApplyColors, &pMan[i]);
1105            }
                P.Execute();
                P.Reset();
                delete[] pMan;
        }
1110 }
    int CFraktalSFT::GetSeed()
    {
        return m_nSeed;
    }
1115 CFraktalSFT::~CFraktalSFT()
    {
        delete[] m_APr;
        delete[] m_APi;
    }
1120 std::string CFraktalSFT::ToZoom()
    {
        CFixedFloat div = m_istop - m_istart;
        return ToZoom((CDecNumber)4 / ((CDecNumber)div.m_f), m_nZoom);
    }
1125 std::string CFraktalSFT::ToZoom(const CDecNumber &z, int &zoom)
    {
        static char szRet[40];
        std::string sszZoom = z.ToText();
        const char *szZoom = sszZoom.c_str();
1130    *szRet = 0;
        for (m_nZoom = 0; szZoom[m_nZoom] && szZoom[m_nZoom] != '.'; m_nZoom++);
        m_nZoom--;
        if (m_nZoom <= 0){
                strncpy(szRet, szZoom, 3);
1135            szRet[3] = 0;
                m_nZooms = 1+log10(atof(szRet)) / log10(2.0);
```

```
                return szRet;
        }
        szRet[0] = szZoom[0];
        if (szZoom[1]){
                szRet[1] = '.';
                szRet[2] = szZoom[1];
                if (szZoom[2] && szZoom[2] != '.'){
                        szRet[3] = szZoom[2];
                        wsprintf(szRet + 4, "e%03d", zoom);
                }
                else
                        wsprintf(szRet + 3, "e%03d", zoom);
        }
        else
                szRet[1] = 0;
        zoom = m_nZoom;
        char szTmp[40];
        strcpy(szTmp, szRet);
        if (strlen(szTmp)>4)
                szTmp[4] = 0;
        m_nZooms = 1 + (log10(atof(szTmp)) + m_nZoom) / (log10(2.0));
        return szRet;
}

static BOOL ISFLOATOK(double a)
{
        if (a <= DBL_MAX && a >= -DBL_MAX)
                return TRUE;
        return 0;
}

void CFraktalSFT::Mirror(int x, int y)
{
        if (!m_bMirrored)
                return;
        int ty = m_nY - y - 1;
        if (ty<y)
                return;
        int tx = m_nX - x - 1;

        m_nPixels[tx][ty] = m_nPixels[x][y];
        m_nTrans[tx][ty] = m_nTrans[x][y];
        if (m_nDEx)
                m_nDEx[tx][ty] = -m_nDEx[x][y];
        if (m_nDEy)
                m_nDEy[tx][ty] = -m_nDEy[x][y];
        int nIndex1 = tx * 3 + (m_bmi->biHeight - 1 - ty)*m_row;
        SetColor(nIndex1, m_nPixels[tx][ty], m_nTrans[tx][ty], tx, ty, 1, 1);
}

#define GET_EXP(val) ((*((__int64*)&val) & 0x7FF0000000000000)>>52)

//#define HARD_GUESS_EXP
//60 2.5

void CFraktalSFT::DeleteArrays()
{
```

```
                if (m_nPixels_LSB)
                {
                        delete[] m_nPixels_LSB;
                        m_nPixels_LSB = nullptr;
                }
                if (m_nPixels_MSB)
                {
                        delete[] m_nPixels_MSB;
                        m_nPixels_MSB = nullptr;
                }
                m_nPixels = itercount_array(0, 0, nullptr, nullptr); // invalid
                if (m_nTrans)
                {
                        if (m_nTrans[0])
                                delete[] m_nTrans[0];
                        delete[] m_nTrans;
                        m_nTrans = NULL;
                }
                if (m_nDEx)
                {
                        if (m_nDEx[0])
                                delete[] m_nDEx[0];
                        delete[] m_nDEx;
                        m_nDEx = nullptr;
                }
                if (m_nDEy)
                {
                        if (m_nDEy[0])
                                delete[] m_nDEy[0];
                        delete[] m_nDEy;
                        m_nDEy = nullptr;
                }
}

void CFraktalSFT::SetPosition(const CFixedFloat &rstart, const CFixedFloat &rstop, const CFixedFloat &istart, const CFixedFloat &istop, int nX, int nY)
{
        m_rstart = rstart;
        m_rstop = rstop;
        m_istart = istart;
        m_istop = istop;
        CFixedFloat re = (rstop + rstart)*.5;
        if (m_nX != nX || m_nY != nY){
                DeleteArrays();
        }
        m_nX = nX;
        m_nY = nY;

        CFixedFloat d = ((CFixedFloat)nX*(CFixedFloat)((double)1 / nY))*(istop - istart)*.5;
        m_rstart = re - d;
        m_rstop = re + d;
}
void CFraktalSFT::SetPosition(const std::string &szR, const std::string &szI, const std::string &szZ)
{
```

```
	try
	{
		Precision pLo(20u);
		CDecNumber z(szZ); // throws on bad string
		CDecNumber di(2 / z);
		long e = 0;
		mpfr_get_d_2exp(&e, z.m_dec.backend().data(), MPFR_RNDN);
		unsigned digits10 = std::max(20L, long(20 + 0.30102999566398114 ↙
		    ↳ * e));
		Precision pHi(digits10);

		CDecNumber re(szR); // throws on bad string
		CDecNumber im(szI); // throws on bad string
		CDecNumber istart(im - di);
		CDecNumber istop(im + di);
		CDecNumber dr(((double)m_scRatio.cx / (double)(m_scRatio.cy))*(↙
		    ↳ istop - istart)*.5);
		CDecNumber rstart(re - dr);
		CDecNumber rstop(re + dr);

		m_rref.m_f.precision(digits10);
		m_iref.m_f.precision(digits10);
		m_rstart.m_f.precision(digits10);
		m_rstop.m_f.precision(digits10);
		m_istart.m_f.precision(digits10);
		m_istop.m_f.precision(digits10);
		m_rstart = rstart.m_dec;
		m_rstop = rstop.m_dec;
		m_istart = istart.m_dec;
		m_istop = istop.m_dec;
	}
	catch (...)
	{
		std::cerr << "ERROR: SetPosition(): couldn't parse float (↙
		    ↳ ignored)" << std::endl;
	}
}

#ifdef KF_OPENCL

void CFraktalSFT::RenderFractalOpenCL()
{
	int64_t antal = 0;
	if (m_nMaxApproximation)
	{
		antal = m_nMaxApproximation - 1;
	}
	std::cerr << antal << std::endl;
	size_t stride_y = &m_nTrans[0][1] - &m_nTrans[0][0];
	size_t stride_x = &m_nTrans[1][0] - &m_nTrans[0][0];
	size_t stride_offset = 0;
	cl->run
  (
	  // for pixel -> parameter mapping
	  m_nX,
	  m_nY,
	  GetJitterSeed(),
```

```
            GetJitterShape(),
            GetJitterScale(),
            m_pixel_step_x,
            m_pixel_step_y,
1305        m_pixel_center_x,
            m_pixel_center_y,
            m_C,
            m_S,
            // for result -> output mapping
1310        stride_y,
            stride_x,
            stride_offset,
            // for iteration control
            m_nBailout,
1315        m_nBailout2,
            log(m_nPower),
            m_nGlitchIter,
            m_nMaxIter,
            m_nMaxIter,
1320        antal,
            m_bNoGlitchDetection,
            m_bAddReference,
            m_nSmoothMethod,
            g_real,
1325        g_imag,
            g_FactorAR,
            g_FactorAI,
            m_epsilon,
            // for series approximation
1330        m_dPixelSpacing,
            m_fPixelSpacing,
            m_nMaxApproximation,
            GetApproxTerms(),
            GetApproximationType(),
1335        m_APr,
            m_APi,
            m_APs,

            // reference orbit
1340        m_db_dxr,
            m_db_dxi,
            m_db_z,
            antal,
            m_nMaxIter,
1345
            // formula selection
            0,
            m_nFractalType,
            m_nPower,
1350        GetDerivatives(),

            // output arrays
            m_nPixels_MSB,
            m_nPixels_LSB,
1355        &m_nTrans[0][0],
            &m_nDEx[0][0],
            &m_nDEy[0][0]
```

```
  );
}

void CFraktalSFT::RenderFractalOpenCLEXP()
{
        int64_t antal = 0;
        if (m_nMaxApproximation)
        {
                antal = m_nMaxApproximation - 1;
        }
        size_t stride_y = &m_nTrans[0][1] - &m_nTrans[0][0];
        size_t stride_x = &m_nTrans[1][0] - &m_nTrans[0][0];
        size_t stride_offset = 0;
        cl->run
  (
          // for pixel -> parameter mapping
          m_nX,
          m_nY,
          GetJitterSeed(),
          GetJitterShape(),
          GetJitterScale(),
          m_pixel_step_x,
          m_pixel_step_y,
          m_pixel_center_x,
          m_pixel_center_y,
          m_C,
          m_S,
          // for result -> output mapping
          stride_y,
          stride_x,
          stride_offset,
          // for iteration control
          m_nBailout,
          m_nBailout2,
          log(m_nPower),
          m_nGlitchIter,
          m_nMaxIter,
          m_nMaxIter,
          antal,
          m_bNoGlitchDetection,
          m_bAddReference,
          m_nSmoothMethod,
          g_real,
          g_imag,
          g_FactorAR,
          g_FactorAI,
          m_epsilon,
          // for series approximation
          m_dPixelSpacing,
          m_fPixelSpacing,
          m_nMaxApproximation,
          GetApproxTerms(),
          GetApproximationType(),
          m_APr,
          m_APi,
          m_APs,
```

```
1415            // reference orbit
                m_dxr,
                m_dxi,
                m_db_z,
                antal,
1420            m_nMaxIter,

                // formula selection
                2,
                m_nFractalType,
1425            m_nPower,
                GetDerivatives(),

                // output arrays
                m_nPixels_MSB,
1430            m_nPixels_LSB,
                &m_nTrans[0][0],
                &m_nDEx[0][0],
                &m_nDEy[0][0]
    );
1435 }

#endif

HBITMAP CFraktalSFT::GetBitmap()
1440 {
        WaitForSingleObject(m_hMutex, INFINITE);
        if (m_bmi && m_lpBits){
                HDC hDC = GetDC(NULL);
                if (!SetDIBits(hDC, m_bmBmp, 0, m_bmi->biHeight, m_lpBits,
1445                    (LPBITMAPINFO)m_bmi, DIB_RGB_COLORS))
                        Beep(1000, 10);
                ReleaseDC(NULL, hDC);
        }
        ReleaseMutex(m_hMutex);
1450    return m_bmBmp;
}
void CFraktalSFT::UpdateBitmap()
{
        WaitForSingleObject(m_hMutex, INFINITE);
1455    if (m_bmi && m_lpBits){
                HDC hDC = GetDC(NULL);
                if (!GetDIBits(hDC, m_bmBmp, 0, m_bmi->biHeight, m_lpBits,
                        (LPBITMAPINFO)m_bmi, DIB_RGB_COLORS))
                        { /*Beep(1000,10)*/ }
1460            ReleaseDC(NULL, hDC);
        }
        ReleaseMutex(m_hMutex);
}
int CFraktalSFT::GetWidth()
1465 {
        return m_nX;
}
int CFraktalSFT::GetHeight()
{
1470    return m_nY;
}
```

```
void CFraktalSFT::Stop(BOOL bNoPostWhenDone)
{
        m_bNoPostWhenDone = bNoPostWhenDone;
        m_bStop = TRUE;
        m_bAddReference = FALSE;
        if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches - 1].x == -1)
                m_bNoGlitchDetection = FALSE;
        else
                m_bNoGlitchDetection = TRUE;
        double counter = 0;
        while (m_bRunning)
        {
                Sleep(1);
                counter += 1;
        }
#ifdef KF_DEBUG_SLEEP
        if (counter > 0)
                std::cerr << "Stop() slept for " << counter << "ms" << std::endl;
#endif
        m_bStop = FALSE;
        m_bNoPostWhenDone=0;
}

void CFraktalSFT::Zoom(int nXPos, int nYPos, double nZoomSize, int nWidth, int nHeight, BOOL bReuseCenter, bool autorender)
{
        Stop(TRUE);
        int **Org;
        float **OrgT;
        float **OrgDEx;
        float **OrgDEy;
        int nOX, nOY;
        int i;
        m_C = cos(g_Degree);
        m_S = sin(g_Degree);

        double mr = m_nX / 2;
        double mi = m_nY / 2;
//      if(g_Degree!=0 && (nXPos!=mr || nYPos!=mi))
//              bReuseCenter=FALSE;
        //double ratio = ((double)m_nX*((double)m_nY/(double)m_nX)) / (double)m_scRatio.cy;
        //double ratio = ((double)360/(double)m_nY) * ((double)m_scRatio.cx*((double)m_scRatio.cy/(double)m_scRatio.cx)) / (double)m_scRatio.cy;
        //double ratio = ((double)360/(double)m_nY) * (double)m_nX*((double)m_nY/(double)m_nX) / (double)m_scRatio.cy;
        double ratio = (((double)m_nY/(double)m_nX)/(360.0/640.0)) * ((double)360 / (double)m_scRatio.cy);
        double xpos = (nXPos - mr)*ratio + mr;
        double dbD0r = mr + m_C*(xpos - mr) + m_S*(nYPos - mi);
        double dbD0i = mi - m_S*(xpos - mr) + m_C*(nYPos - mi);
        dbD0r = (dbD0r - mr) / ratio + mr;
//      nXPos = dbD0r;
//      nYPos = dbD0i;

        m_bAddReference = FALSE;
```

```
        if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
                m_bNoGlitchDetection = FALSE;
1525    else
                m_bNoGlitchDetection = TRUE;
        if (m_nX != nWidth || m_nY != nHeight){
                DeleteArrays();
        }
1530    else if (bReuseCenter && !GetNoReuseCenter() && nZoomSize<=1){
                m_bAddReference = 2;
                nOX = nWidth*nZoomSize;
                nOY = nHeight*nZoomSize;
                Org = new int*[nOX];
1535            for (i = 0; i<nOX; i++)
                        Org[i] = new int[nOY];
                OrgT = new float*[nOX];
                for (i = 0; i<nOX; i++)
                        OrgT[i] = new float[nOY];
1540            OrgDEx = new float*[nOX];
                for (i = 0; i<nOX; i++)
                        OrgDEx[i] = new float[nOY];
                OrgDEy = new float*[nOX];
                for (i = 0; i<nOX; i++)
1545                    OrgDEy[i] = new float[nOY];
                int x, y, a, b;
                int nX2 = m_nX/2;
                int nY2 = m_nY/2;
                if(0 && nZoomSize<1 && nZoomSize>.8){
1550                    for (x = 0; x<nOX; x++){
                                for (y = 0; y<nOY; y++){
                                        Org[x][y]=PIXEL_UNEVALUATED;
                                }
                        }
1555            }
                else{
                        for (x = 0; x<nOX; x++){
                                for (y = 0; y<nOY; y++){
                                        a = (x + nXPos-nX2) / nZoomSize;
1560                                    b = (y + nYPos-nY2) / nZoomSize;
                                        if (a >= 0 && a<m_nX && b >= 0 && b<m_nY↙
                                            ↳ ){
                                                Org[x][y] = m_nPixels[a][b];
                                                OrgT[x][y] = m_nTrans[a][b];
                                                OrgDEx[x][y] = m_nDEx[a][b];
1565                                            OrgDEy[x][y] = m_nDEy[a][b];
                                                if (Org[x][y]>m_nMaxIter)
                                                        Org[x][y] = m_nMaxIter;
                                        }
                                        else
1570                                    {
                                                Org[x][y] = PIXEL_UNEVALUATED;
                                                OrgT[x][y] = SET_TRANS_GLITCH(0)↙
                                                    ↳ ;
                                                OrgDEx[x][y] = 0;
                                                OrgDEy[x][y] = 0;
1575                                    }
                                }
                        }
```

```
                }
                a = (nWidth - nOX) / 2;
                b = (nHeight - nOY) / 2;
                for (x = 0; x<m_nX; x++){
                        for (y = 0; y<m_nY; y++){
                                if (x - a>=0 && x - a<nOX && y - b>=0 && y - b<nOY){
                                        m_nPixels[x][y] = Org[x - a][y - b];
                                        m_nTrans[x][y] = OrgT[x - a][y - b];
                                        m_nDEx[x][y] = OrgDEx[x - a][y - b];
                                        m_nDEy[x][y] = OrgDEy[x - a][y - b];
                                }
                                else
                                {
                                        m_nPixels[x][y] = PIXEL_UNEVALUATED;
                                        m_nTrans[x][y] = SET_TRANS_GLITCH(0);
                                        m_nDEx[x][y] = 0;
                                        m_nDEy[x][y] = 0;
                                }
                        }
                }
                for (i = 0; i<nOX; i++){
                        delete[] Org[i];
                        delete[] OrgT[i];
                        delete[] OrgDEx[i];
                        delete[] OrgDEy[i];
                }
                delete[] Org;
                delete[] OrgT;
                delete[] OrgDEx;
                delete[] OrgDEy;
        }
        m_nX = nWidth;
        m_nY = nHeight;
        unsigned digits10 = 20u;
        {
                using std::abs;
                using std::min;
                Precision pLo(20u);
                CFixedFloat pixelSpacing(min(abs((m_rstop - m_rstart) / m_nX), abs((m_rstop - m_rstart) / m_nY)));
                long e = 0;
                mpfr_get_d_2exp(&e, pixelSpacing.m_f.backend().data(), MPFR_RNDN);
                digits10 = std::max(20.0, 20 + 0.30102999566398114 * (log2(nZoomSize) - e));
        }
        Precision p(digits10);
        m_rstart.m_f.precision(digits10);
        m_rstop.m_f.precision(digits10);
        m_istart.m_f.precision(digits10);
        m_istop.m_f.precision(digits10);
        CFixedFloat nr = (m_rstop - m_rstart)*(CFixedFloat)((double)1 / ((double)nZoomSize * 2));
        CFixedFloat ni = (m_istop - m_istart)*(CFixedFloat)((double)1 / ((double)nZoomSize * 2));
        CFixedFloat offsr = (CFixedFloat)dbD0r*(m_rstop - m_rstart)*(CFixedFloat
```

```
            ↪ )((double)1 / m_nX) + m_rstart;
        CFixedFloat offsi = (CFixedFloat)dbD0i*(m_istop - m_istart)*(CFixedFloat↙
            ↪ )((double)1 / m_nY) + m_istart;
1630    m_rstart = offsr - nr;
        m_rstop = offsr + nr;
        m_istart = offsi - ni;
        m_istop = offsi + ni;
//      if (bReuseCenter && m_nZoom>g_nRefZero && !m_bReuseRef)
1635    //          AddReference(nXPos + m_nY/10 - 1, nYPos + m_nY/10 - 1);
//      else
        if (autorender)
            RenderFractal(m_nX, m_nY, m_nMaxIter, m_hWnd);
}
1640
extern int g_bAutoGlitch;
double CFraktalSFT::GetProgress(int *pnGuessed, int *pnRDone, int *pnAP, int *↙
    ↪ pnT)
{
        int64_t iters = m_nMaxIter;
1645    if ((GetUseNanoMB1() || GetUseNanoMB2()) && g_bAutoGlitch == 1)
            iters = g_period;
        if (iters <= 0)
            iters = 1;
        if (pnGuessed)
1650        *pnGuessed = m_nGuessed * 100.0 / (m_nDone ? m_nDone : 1);
        if (pnRDone)
            *pnRDone = m_nRDone * 100.0 / iters;
        if (pnAP)
            *pnAP = m_nApprox * 100.0 / iters;
1655    if (pnT)
            *pnT = m_nTotal;
        if (!m_bmi)
            return 0;
        if (!m_nTotal)
1660        return 100;
        return m_nDone * 100.0 / m_nTotal;
}

std::string CFraktalSFT::GetPosition()
1665 {
        CStringTable st;
        if (m_szPosition)
            st.DeleteToText(m_szPosition);
        m_szPosition = nullptr;
1670    st.AddRow();
        st.AddString(st.GetCount() - 1, "MANDELBROT:");
        st.AddRow();
        st.AddString(st.GetCount() - 1, "R-Start");
        st.AddString(st.GetCount() - 1, m_rstart.ToText());
1675    st.AddRow();
        st.AddString(st.GetCount() - 1, "R-Stop");
        st.AddString(st.GetCount() - 1, m_rstop.ToText());
        st.AddRow();
        st.AddString(st.GetCount() - 1, "I-Start");
1680    st.AddString(st.GetCount() - 1, m_istart.ToText());
        st.AddRow();
        st.AddString(st.GetCount() - 1, "I-Stop");
```

```
        st.AddString(st.GetCount() - 1, m_istop.ToText());
        st.AddRow();
        st.AddString(st.GetCount() - 1, "Max-Iter");
        st.AddInt(st.GetCount() - 1, m_nMaxIter);
        if (m_nPixels){
                int x, y;
                int nMax = PIXEL_UNEVALUATED, nMin = PIXEL_UNEVALUATED;
                for (x = 0; x<m_nX; x++){
                        for (y = 0; y<m_nY; y++){
                                if (nMin == PIXEL_UNEVALUATED || nMin>m_nPixels[x][y])
                                        nMin = m_nPixels[x][y];
                                if (nMax == PIXEL_UNEVALUATED || nMax<m_nPixels[x][y])
                                        nMax = m_nPixels[x][y];
                        }
                }
                st.AddRow();
                st.AddString(st.GetCount() - 1, "Iter-Min");
                st.AddInt(st.GetCount() - 1, nMin);
                st.AddRow();
                st.AddString(st.GetCount() - 1, "Iter-Max");
                st.AddInt(st.GetCount() - 1, nMax);
        }
        m_szPosition = st.ToText(":", "\n");
        return m_szPosition;
}
int CFraktalSFT::CountFrames(int nProcent)
{
        double z = std::stod(ToZoom().substr(0, 4));
        return (int)((log10(z) + m_nZoom) / (log10(1 + (double)2 * (double)nProcent / (double)100))) + 1;
}
void CFraktalSFT::GetIterations(int64_t &nMin, int64_t &nMax, int *pnCalculated, int *pnType, BOOL bSkipMaxIter)
{
        if (m_bIterChanged || pnCalculated){
                int64_t nMA = (m_nMaxApproximation == m_nMaxIter ? 0 : m_nMaxApproximation);
                int64_t nCalc1 = 0;
                int64_t nCalc2 = 0;
                if (pnType)
                        *pnType = 0;
                if (m_nPixels){
                        int x, y;
                        nMax = PIXEL_UNEVALUATED;
                        nMin = PIXEL_UNEVALUATED;
                        for (x = 0; x<m_nX; x++){
                                for (y = 0; y<m_nY; y++){
                                        if (bSkipMaxIter && m_nPixels[x][y] >= m_nMaxIter - 1) // FIXME what about glitches?
                                                continue;
                                        if (m_nPixels[x][y] != PIXEL_UNEVALUATED && (nMin == PIXEL_UNEVALUATED || nMin>m_nPixels[x][y]))
                                                nMin = m_nPixels[x][y];
```

```
                                        if (m_nPixels[x][y] != PIXEL_UNEVALUATED
                                            && (nMax == PIXEL_UNEVALUATED ||
                                            nMax<m_nPixels[x][y]))
                                                nMax = m_nPixels[x][y];
                                        if (pnCalculated && m_nPixels[x][y] !=
                                            PIXEL_UNEVALUATED){
                                                nCalc1 += m_nPixels[x][y] - nMA;
1735                                            if (nCalc1>1000000){
                                                        nCalc2 += nCalc1 /
                                                            1000000;
                                                        nCalc1 %= 1000000;
                                                }
                                        }
1740                                }
                        }
                }
                m_bIterChanged = FALSE;
                m_nMinI = nMin;
1745            m_nMaxI = nMax;
                if (pnCalculated){
                        if (nCalc2>2000){
                                if (pnType)
                                        *pnType = 1;
1750                            *pnCalculated = nCalc2;
                        }
                        else{
                                *pnCalculated = nCalc2 * 1000000 + nCalc1;
                        }
1755            }
        }
        else{
                nMin = m_nMinI;
                nMax = m_nMaxI;
1760    }
}
int64_t CFraktalSFT::GetIterations()
{
        return m_nMaxIter;
1765 }
void CFraktalSFT::SetIterations(int64_t nIterations)
{
        m_nMaxIter = nIterations;
}
1770 std::string CFraktalSFT::GetRe()
{
        CFixedFloat re = (m_rstop + m_rstart)*.5;
        return re.ToText();
}
1775 std::string CFraktalSFT::GetRe(int nXPos, int nYPos, int width, int height)
{
        double mr = width / 2;
        double mi = height / 2;
        double ratio = (((double)height/(double)width)/(360.0/640.0)) * ((double
            )360 / (double)m_scRatio.cy);
1780    double xpos = (nXPos - mr)*ratio + mr;
        double dbD0r = mr + m_C*(xpos - mr) + m_S*(nYPos - mi);
        dbD0r = (dbD0r - mr) / ratio + mr;
```

```
        CFixedFloat re = m_rstart + ((double)dbD0r/(double)width)*(m_rstop - ↙
            ↳ m_rstart);
1785    return re.ToText();
}
std::string CFraktalSFT::GetIm()
{
        CFixedFloat im = (m_istop + m_istart)*.5;
1790    return im.ToText();
}
std::string CFraktalSFT::GetIm(int nXPos, int nYPos, int width, int height)
{
        double mr = width / 2;
1795    double mi = height / 2;
        double ratio = (((double)height/(double)width)/(360.0/640.0)) * ((double↙
            ↳ )360 / (double)m_scRatio.cy);
        double xpos = (nXPos - mr)*ratio + mr;
        double dbD0r = mr + m_C*(xpos - mr) + m_S*(nYPos - mi);
        double dbD0i = mi - m_S*(xpos - mr) + m_C*(nYPos - mi);
1800    dbD0r = (dbD0r - mr) / ratio + mr;

        CFixedFloat im = m_istart + ((double)dbD0i/(double)height)*(m_istop - ↙
            ↳ m_istart);
        return im.ToText();
}
1805 std::string CFraktalSFT::GetZoom()
{
        CDecNumber zoomDec = (CDecNumber)4 / ((CDecNumber)m_istop.ToText() - (↙
            ↳ CDecNumber)m_istart.ToText());
        floatexp zoomFE; zoomFE = CFixedFloat(zoomDec.m_dec);
        return zoomFE.toString();
1810 }
BOOL CFraktalSFT::HighestIteration(int &rx, int &ry)
{
        if (!m_nPixels)
                return FALSE;
1815    int x, y;
        int nMax = PIXEL_UNEVALUATED;
        for (x = 0; x<m_nX; x++)
        for (y = 0; y<m_nY; y++)
        if (m_nPixels[x][y] != PIXEL_UNEVALUATED && m_nPixels[x][y]>nMax){
1820            nMax = m_nPixels[x][y];
                rx = x;
                ry = y;
        }
        return nMax != PIXEL_UNEVALUATED;
1825 }
BOOL CFraktalSFT::Center(int &rx, int &ry, BOOL bSkipM, BOOL bQuick)
{
        if (!m_nPixels)
                return FALSE;
1830    int x, y;
        int nSegmentX = m_nX / 4;
        int nSegmentY = m_nY / 4;
        int nHalfX = m_nX / 2;
        int nHalfY = m_nY / 2;
1835    int tx, ty;
```

```
	double val, minval = 0;
	BOOL bFirst = TRUE;
	uint64_t nStep0 = uint64_t(bQuick ? 6 : 4)*uint64_t(m_nX)*uint64_t(m_nX)/
	    ↪ / uint64_t(640 * 640);
	if (nStep0 <= 0)
1840		nStep0 = 1;
	if (nStep0 > 1 << 30)
		nStep0 = 1 << 30;
	int nStep = nStep0;
	GetBitmap();
1845	int64_t nMin, nMax;
	GetIterations(nMin, nMax, NULL, NULL, TRUE);
	int64_t nMinIter = nMin + (nMax - nMin) / 4;

	for (tx = nSegmentX; tx<nHalfX + nSegmentX; tx++){
1850		if (bSkipM && tx>nHalfX - nSegmentX / 3 && tx<nHalfX + nSegmentX/
		    ↪ / 3)
			continue;
		for (ty = nSegmentY; ty<nHalfY + nSegmentY; ty++){
			if(m_nPixels[tx][ty]<nMinIter)
				continue;
1855			val = 0;
			for (x = 0; x<nSegmentX; x += nStep){
				for (y = 0; y<nSegmentY; y += nStep){
					int x1 = tx - x;
					int x2 = tx + x;
1860					int y1 = ty - y;
					int y2 = ty + y;
					int nIndex1 = x1 * 3 + (m_bmi->biHeight /
					    ↪ - 1 - y1)*m_row;
					int nIndex2 = x2 * 3 + (m_bmi->biHeight /
					    ↪ - 1 - y2)*m_row;
					int     t = (m_lpBits[nIndex1]+m_lpBits[/
					    ↪ nIndex1+1]+m_lpBits[nIndex1+2])-(/
					    ↪ m_lpBits[nIndex2]+m_lpBits[nIndex2/
					    ↪ +1]+m_lpBits[nIndex2+2]);
1865					val += (t<0 ? -t : t);

					x1 = tx - x;
					x2 = tx + x;
					y1 = ty + y;
1870					y2 = ty - y;
					nIndex1 = x1 * 3 + (m_bmi->biHeight - 1 /
					    ↪ - y1)*m_row;
					nIndex2 = x2 * 3 + (m_bmi->biHeight - 1 /
					    ↪ - y2)*m_row;
					if(((unsigned int)(nIndex2))>m_bmi->/
					    ↪ biSizeImage-3)
						continue;
1875					t = (m_lpBits[nIndex1]+m_lpBits[nIndex1/
					    ↪ +1]+m_lpBits[nIndex1+2])-(m_lpBits/
					    ↪ [nIndex2]+m_lpBits[nIndex2+1]+/
					    ↪ m_lpBits[nIndex2+2]);
					val += (t<0 ? -t : t);
				}
			}
			if (bFirst || minval>val){
```

```
1880                                    rx = tx;
                                        ry = ty;
                                        minval = val;
                                        bFirst = FALSE;
                                }
1885                    }
                }
                if (minval == 0)
                        return FALSE;
                return TRUE;
1890    }

        COLOR14 CFraktalSFT::GetKeyColor(int i)
        {
                if (i<0 || i >= m_nParts)
1895                    return m_cKeys[0];
                else
                        return m_cKeys[i];
        }
        void CFraktalSFT::SetKeyColor(COLOR14 col, int i)
1900    {
                if (i<0 || i >= m_nParts)
                        return;
                m_cKeys[i] = col;
        }
1905    COLOR14 CFraktalSFT::GetColor(int i)
        {
                if (i<0 || i >= 1024)
                        return m_cPos[0];
                else
1910                    return m_cPos[i];
        }

        void CFraktalSFT::SetImageSize(int nx, int ny)
        {
1915            m_bResized = m_nXPrev != nx || m_nYPrev != ny;
                if (m_bResized && m_nPixels){
                        DeleteArrays();
                }
                m_nX = nx;
1920            m_nY = ny;
                m_nXPrev = m_nX;
                m_nYPrev = m_nY;
                bool two = GetIterations() >= INT_MAX;
                if (! m_nPixels_LSB)
1925            {
                        m_nPixels_LSB = new uint32_t[m_nX * m_nY];
                        m_nPixels_MSB = two ? new uint32_t[m_nX * m_nY] : nullptr;
                        m_nTrans = new float*[m_nX];
                        m_nDEx = new float*[m_nX];
1930                    m_nDEy = new float*[m_nX];
                        m_nTrans[0] = new float[m_nX * m_nY];
                        m_nDEx[0] = new float[m_nX * m_nY];
                        m_nDEy[0] = new float[m_nX * m_nY];
                        for (int x = 1; x<m_nX; x++){
1935                            m_nTrans[x] = m_nTrans[0] + x * m_nY;
                                m_nDEx[x] = m_nDEx[0] + x * m_nY;
```

```
                        m_nDEy[x] = m_nDEy[0] + x * m_nY;
                }
                m_nPixels = itercount_array(m_nY, 1, m_nPixels_LSB, ↙
                    ↳ m_nPixels_MSB);
        }
        memset(m_nPixels_LSB, 0, sizeof(*m_nPixels_LSB) * m_nX * m_nY);
        if (two)
                memset(m_nPixels_MSB, 0, sizeof(*m_nPixels_MSB) * m_nX * m_nY);
        memset(m_nTrans[0], 0, sizeof(float) * m_nX * m_nY);
        memset(m_nDEx[0], 0, sizeof(float) * m_nX * m_nY);
        memset(m_nDEy[0], 0, sizeof(float) * m_nX * m_nY);
        SetImageWidth(nx);
        SetImageHeight(ny);
}

bool CFraktalSFT::OpenMapEXR(const std::string &szfile)
{
        return ReadEXRMapFile(szfile);
}

BOOL CFraktalSFT::OpenMapB(const std::string &szFile, BOOL bReuseCenter, double ↙
    ↳ nZoomSize)
{
        int **Org = 0;
        float **OrgT = 0;
        float **OrgDEx = 0;
        float **OrgDEy = 0;
        int nOX = 0, nOY = 0;
        int i;
        int x, y, a = 0, b = 0;
        if (bReuseCenter && !GetNoReuseCenter() && nZoomSize<1){
                int i;
                nOX = m_nX*nZoomSize;
                nOY = m_nY*nZoomSize;
                Org = new int*[nOX];
                for (i = 0; i<nOX; i++)
                        Org[i] = new int[nOY];
                OrgT = new float*[nOX];
                for (i = 0; i<nOX; i++)
                        OrgT[i] = new float[nOY];
                OrgDEx = new float*[nOX];
                for (i = 0; i<nOX; i++)
                        OrgDEx[i] = new float[nOY];
                OrgDEy = new float*[nOX];
                for (i = 0; i<nOX; i++)
                        OrgDEy[i] = new float[nOY];
                for (x = 0; x<nOX; x++){
                        for (y = 0; y<nOY; y++){
                                a = x / nZoomSize;
                                b = y / nZoomSize;
                                if (a >= 0 && a<m_nX && b >= 0 && b<m_nY)
                                {
                                        Org[x][y] = m_nPixels[a][b];
                                        OrgT[x][y] = m_nTrans[a][b];
                                        OrgDEx[x][y] = m_nDEx[a][b];
                                        OrgDEy[x][y] = m_nDEy[a][b];
                                }
```

```
                        }
                }
                a = (m_nX - nOX) / 2;
1995            b = (m_nY - nOY) / 2;
        }
        //HANDLE hFile = CreateFile(szFile,GENERIC_READ,0,NULL,OPEN_EXISTING,0,↙
            ↳ NULL);
        FILE *hFile = fopen(szFile.c_str(), "rb");
        //if(hFile==INVALID_HANDLE_VALUE)
2000    if (hFile == NULL) return FALSE; // FIXME leaks Org arrays
        DWORD dw;
        char szId[3];
        //ReadFile(hFile,szId,3,&dw,NULL);
        fread(szId, 1, 3, hFile);
2005    BOOL bNewFormat = FALSE;
        if (!strncmp(szId, "KFC", 3))
                bNewFormat = 1;
        if (!strncmp(szId, "KFD", 3))
                bNewFormat = 2;
2010    else if (strncmp(szId, "KFB", 3)){
                //CloseHandle(hFile);
                fclose(hFile);
                return FALSE;
        }
2015    //ReadFile(hFile,&m_nX,sizeof(int),&dw,NULL);
        //ReadFile(hFile,&m_nY,sizeof(int),&dw,NULL);
        int nx = -1, ny = -1;
        fread(&nx, 1, sizeof(int), hFile);
        fread(&ny, 1, sizeof(int), hFile);
2020    SetImageSize(nx, ny);
        float *pLine = NULL;
        if (bNewFormat == 1)
                pLine = new float[m_nY];
        else if (bNewFormat == 2)
2025            pLine = new float[m_nX];
        if (bNewFormat == 2){
                for (y = 0; y<m_nY; y++){
                        //ReadFile(hFile,pLine,sizeof(float)*m_nX,&dw,NULL);
                        fread(pLine, 1, sizeof(float)*m_nX, hFile);
2030                    for (x = 0; x<m_nX; x++){
                                m_nPixels[x][y] = (int)pLine[x];
                                m_nTrans[x][y] = pLine[x] - (int)m_nPixels[x][y↙
                                    ↳ ];
                        }
                }
2035    }
        else{
                for (x = 0; x<m_nX; x++){
                        if (bNewFormat){
                                //ReadFile(hFile,pLine,sizeof(float)*m_nY,&dw,↙
                                    ↳ NULL);
2040                            fread(pLine, 1, sizeof(float)*m_nY, hFile);
                                for (y = 0; y<m_nY; y++){
                                        m_nPixels[x][y] = (int)pLine[y];
                                        m_nTrans[x][y] = pLine[y] - (int)↙
                                            ↳ m_nPixels[x][y];
                                }
```

```
2045                        }
                            else
                                    //ReadFile(hFile,m_nPixels[x],sizeof(int)*m_nY,&dw,NULL);
                                    fread(m_nPixels_LSB + x * m_nY, 1, sizeof(*m_nPixels_LSB)*m_nY, hFile);
                    }
2050        }
            if (pLine)
                    delete[] pLine;
            //ReadFile(hFile,&m_nIterDiv,sizeof(int),&dw,NULL);
            int div = 1;
2055        fread(&div, 1, sizeof(int), hFile);
    #if 0 // don't read IterDiv, it is stored as int (not float) due to historical reasons
            m_nIterDiv = div;
            if (m_nIterDiv == 0)
                    m_nIterDiv = 1;
2060    #endif
            //ReadFile(hFile,&m_nParts,sizeof(int),&dw,NULL);
            fread(&m_nParts, 1, sizeof(int), hFile);
            //ReadFile(hFile,m_cKeys,sizeof(COLOR14)*m_nParts,&dw,NULL);
            fread(m_cKeys, 1, sizeof(COLOR14)*m_nParts, hFile);
2065        int nTest;
            //ReadFile(hFile,&nTest,sizeof(int),&dw,NULL);
            dw = fread(&nTest, 1, sizeof(int), hFile);
            if (sizeof(int) == dw)
                    m_nMaxIter = nTest;
2070
            BOOL ok = TRUE;
            if (!bNewFormat){
                    for (x = 0; x<m_nX; x++){
                            ok &= (1 == fread(m_nTrans[x], sizeof(float)*m_nY, 1, hFile));
2075                }
                    off_t pos1 = ftello(hFile);
                    fseek(hFile, 0, SEEK_END);
                    off_t pos2 = ftello(hFile);
                    if (pos1 != pos2)
2080                {
                            fseeko(hFile, pos1, SEEK_SET);
                            for (x = 0; x<m_nX; x++){
                                    ok &= (1 == fread(m_nDEx[x], sizeof(float)*m_nY, 1, hFile));
                            }
2085                }
            }
            //CloseHandle(hFile);
            fclose(hFile);

2090        if (bReuseCenter && !GetNoReuseCenter() && nZoomSize<1){
                    for (x = 0; x<m_nX; x++){
                            for (y = 0; y<m_nY; y++){
                                    if (x - a>0 && x - a<nOX - 1 && y - b>0 && y - b<nOY - 1){
                                            m_nPixels[x][y] = Org[x - a][y - b];
2095                                        m_nTrans[x][y] = OrgT[x - a][y - b];
```

```
                                        m_nDEx[x][y] = OrgDEx[x - a][y - b];
                                        m_nDEy[x][y] = OrgDEy[x - a][y - b];
                                }
                        }
                }
        }
        for (i = 0; i<nOX; i++){
                delete[] Org[i];
                delete[] OrgT[i];
                delete[] OrgDEx[i];
                delete[] OrgDEy[i];
        }
        delete[] Org;
        delete[] OrgT;
        delete[] OrgDEx;
        delete[] OrgDEy;
    }
    ReinitializeBitmap();
    return ok;
}

void CFraktalSFT::ReinitializeBitmap()
{
    if (m_bmBmp)
        DeleteObject(m_bmBmp);
    HDC hDC = GetDC(NULL);
    m_bmBmp = create_bitmap(hDC, m_nX, m_nY);
    if (m_bmi)
        free(m_bmi);
    m_bmi = (BITMAPINFOHEADER *)malloc(sizeof(BITMAPINFOHEADER)+sizeof(RGBQUAD)* 256);
    memset(m_bmi, 0, sizeof(BITMAPINFOHEADER)+sizeof(RGBQUAD)* 256);
    m_bmi->biSize = sizeof(BITMAPINFOHEADER);
    if (!GetDIBits(hDC, m_bmBmp, 0, 0, NULL, (LPBITMAPINFO)m_bmi, DIB_RGB_COLORS))
        { /*Beep(1000,10)*/ }
    ReleaseDC(NULL, hDC);
    m_bmi->biCompression = m_bmi->biClrUsed = m_bmi->biClrImportant = 0;
    m_bmi->biBitCount = 24;
    if (m_bmi->biBitCount != 24)
        m_bmi->biClrUsed = 1 << m_bmi->biBitCount;
    m_row = ((((m_bmi->biWidth*(DWORD)m_bmi->biBitCount) + 31)&~31) >> 3);
    m_bmi->biSizeImage = m_row*m_bmi->biHeight;

    if (!m_lpBits || (int)m_bmi->biSizeImage != m_nSizeImage){
        m_nSizeImage = m_bmi->biSizeImage;
        if (m_lpBits)
            delete[] m_lpBits;
        m_lpBits = new BYTE[m_bmi->biSizeImage];

        if (!GetDIBits(hDC, m_bmBmp, 0, m_bmi->biHeight, m_lpBits,
            (LPBITMAPINFO)m_bmi, DIB_RGB_COLORS))
            { /*Beep(1000,10)*/ }
    }
}

double CFraktalSFT::GetIterDiv()
{
```

```
        return m_nIterDiv;
}
void CFraktalSFT::SetIterDiv(double nIterDiv)
{
        if (nIterDiv>0)
                m_nIterDiv = nIterDiv;
}

int CFraktalSFT::SaveJpg(const std::string &szFile, int nQuality, int nWidth, int nHeight)
{
        std::string comment1(ToText());
        std::string comment2(m_Settings.ToText());
        std::string comment = comment1 + comment2;
        if (nWidth == 0)
                nWidth = m_nX;
        if (nHeight == 0)
                nHeight = m_nY;
        if (m_nX == nWidth && m_nY == nHeight)
                return ::SaveImage(szFile, m_lpBits, m_nX, m_nY, nQuality, comment);
        else{
                HBITMAP bmSave = ShrinkBitmap(GetBitmap(), nWidth, nHeight, GetShrink());
                int nRet = ::SaveImage(szFile, bmSave, nQuality, comment);
                DeleteObject(bmSave);
                return nRet;
        }
}
int64_t CFraktalSFT::GetMaxApproximation()
{
        return m_nApprox;
}
int64_t CFraktalSFT::GetIterationOnPoint(int x, int y)
{
        WaitForSingleObject(m_hMutex, INFINITE);
        if (!m_nPixels || m_nXPrev != m_nX || m_nYPrev != m_nY){
                ReleaseMutex(m_hMutex);
                return PIXEL_UNEVALUATED;
        }
        if (x<0 || x >= m_nX || y<0 || y >= m_nY){
                ReleaseMutex(m_hMutex);
                return PIXEL_UNEVALUATED;
        }
        int64_t nRet = m_nPixels[x][y];
        ReleaseMutex(m_hMutex);
        return nRet;
}
double CFraktalSFT::GetTransOnPoint(int x, int y)
{
        if (x<0 || x >= m_nX || y<0 || y >= m_nY || !m_nTrans){
                return 0;
        }
        return 1 - m_nTrans[x][y];
}
static BOOL IsEqual(int a, int b, int nSpan = 2, BOOL bGreaterThan = FALSE)
{
```

```
2205            //      return a==b;
                if (a == PIXEL_UNEVALUATED || b == PIXEL_UNEVALUATED)
                        return 0;
                if (bGreaterThan)
                        return a>nSpan;
2210            int diff = a - b;
                if (diff<0)
                        diff = -diff;
                return diff<nSpan;
        }
2215
        #define KF_RERENDER_ONLY_ALL_GLITCHES
        BOOL CFraktalSFT::AddReference(int nXPos, int nYPos, BOOL bEraseAll, BOOL ↙
            ↳ bNoGlitchDetection, BOOL bResuming)
        {
                if (!m_nPixels || (m_nZoom<g_nRefZero && !bEraseAll))
2220                    return FALSE;
        g_nAddRefX=nXPos;g_nAddRefY=nYPos;

                m_C = cos(g_Degree);
                m_S = sin(g_Degree);
2225            m_bNoGlitchDetection = bNoGlitchDetection;
                if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
                        m_bNoGlitchDetection = FALSE;
                else
                        m_bNoGlitchDetection = TRUE;
2230    #ifdef KF_RERENDER_ONLY_ALL_GLITCHES
        #else
                int **Pixels = m_nPixels;
        #endif
                if (m_nZoom >= g_nRefZero){
2235                    double mr = m_nX / 2;
                        double mi = m_nY / 2;
                        double ratio = (((double)m_nY/(double)m_nX)/(360.0/640.0)) * ((↙
                            ↳ double)360 / (double)m_scRatio.cy);
                        //double ratio = ((double)m_nX*((double)m_nY/(double)m_nX)) / (↙
                            ↳ double)m_scRatio.cy;
                        //double ratio = ((double)360/(double)m_nY) * ((double)m_scRatio↙
                            ↳ .cx*((double)m_scRatio.cy/(double)m_scRatio.cx)) / (double↙
                            ↳ )m_scRatio.cy;
2240                    //double ratio = ((double)360/(double)m_nY) * (double)m_nX*((↙
                            ↳ double)m_nY/(double)m_nX) / (double)m_scRatio.cy;
                        double xpos = (nXPos - mr)*ratio + mr;
                        double dbD0r = mr + m_C*(xpos - mr) + m_S*(nYPos - mi);
                        double dbD0i = mi - m_S*(xpos - mr) + m_C*(nYPos - mi);
                        dbD0r = (dbD0r - mr) / ratio + mr;
2245                    m_rref = (CFixedFloat)dbD0r*(m_rstop - m_rstart)*(CFixedFloat)((↙
                            ↳ double)1 / m_nX) + m_rstart;
                        m_iref = (CFixedFloat)dbD0i*(m_istop - m_istart)*(CFixedFloat)((↙
                            ↳ double)1 / m_nY) + m_istart;
                        g_nAddRefX = nXPos;
                        g_nAddRefY = nYPos;
                }
2250            else{
                        m_rref = 0;
                        m_iref = 0;
                        g_nAddRefX = -1;
```

```
                g_nAddRefY = -1;
        }
        int x, y;
#ifdef KF_RERENDER_ONLY_ALL_GLITCHES
#else
        int i = 0;
#endif
        if(nXPos>=0 && nXPos<m_nX && nYPos>=0 && nYPos<m_nY)
#ifdef KF_RERENDER_ONLY_ALL_GLITCHES
                ;
#else
                i = Pixels[nXPos][nYPos];
#endif
        else
                bResuming=TRUE;

        int nCount = 0;
        if (bEraseAll){
                for (x = 0; x<m_nX; x++)
                for (y = 0; y<m_nY; y++)
                        m_nPixels[x][y] = PIXEL_UNEVALUATED;
        }
/*      else if (bNP){
                int **Node = new int*[m_nX];
                for (i = 0; i<m_nX; i++)
                        Node[i] = new int[m_nY];
                for (x = 0; x<m_nX; x++)
                for (y = 0; y<m_nY; y++)
                        Node[x][y] = m_nPixels[x][y];
                GetArea(Node, nXPos, nYPos, 2, itercount_array_invalid, -1);
                for (x = 0; x<m_nX; x++)
                for (y = 0; y<m_nY; y++)
                if (Node[x][y] == -1)
                        m_nPixels[x][y] = -1;
        }
*/      else if (!bResuming){
                for (x = 0; x<m_nX; x++){
                        for (y = 0; y<m_nY; y++){
#ifdef KF_RERENDER_ONLY_ALL_GLITCHES
                                // re-render all and only glitched pixels
                                if (GET_TRANS_GLITCH(m_nTrans[x][y]))
                                {
                                        m_nPixels[x][y] = PIXEL_UNEVALUATED;
                                        nCount++;
                                }
#else
                                // re-render all and only pixels with the same integer iteration count
                                if (IsEqual(i, Pixels[x][y], 1)){// && IsEqual(t, (int)(SMOOTH_TOLERANCE*m_nTrans[x][y]), 4)){
                                        m_nPixels[x][y] = PIXEL_UNEVALUATED;
                                        nCount++;
                                }
#endif
                        }
                }
```

```
        }

        m_bAddReference = TRUE;
        RenderFractal(m_nX, m_nY, m_nMaxIter, m_hWnd, m_hWnd == nullptr, FALSE);
        return TRUE;
}
#undef KF_RERENDER_ONLY_ALL_GLITCHES

#define SMOOTH_TO 7
int CFraktalSFT::GetArea(itercount_array &Node, int nXStart,int nYStart,int nEqSpan, itercount_array &Pixels, int nDone)
{
        int x, y;
        int nAreaC=0;
        int nTarget = m_nPixels[nXStart][nYStart];

        int nQSize = m_nX*m_nY;
        if(nQSize>230400)
                nQSize=230400;
        POINT *pQ = new POINT[nQSize];
        nQSize--;
        int nQ=1;
        pQ[nQ-1].x = nXStart;
        pQ[nQ-1].y = nYStart;
        int nValidate = nQSize*2;
        while(nQ && nValidate){
                x = pQ[nQ-1].x;
                y = pQ[nQ-1].y;
                nQ--;
                nValidate--;
                if(IsEqual(Node[x][y],nTarget,nEqSpan,Pixels?TRUE:FALSE) && nQ<nQSize && GET_TRANS_GLITCH(m_nTrans[x][y])){
                        nAreaC++;
                        Node[x][y]=nDone;
                        if(Pixels && GET_TRANS_GLITCH(m_nTrans[x][y]))
                                Pixels[x][y]=PIXEL_UNEVALUATED;
                        int w=x, e=x;
                        while(nValidate && w && IsEqual(Node[w-1][y],nTarget,nEqSpan,Pixels?TRUE:FALSE) && GET_TRANS_GLITCH(m_nTrans[w-1][y])){
                                nAreaC++;
                                nValidate--;
                                w--;
                                Node[w][y]=nDone;
                                if(Pixels && GET_TRANS_GLITCH(m_nTrans[w][y]))
                                        Pixels[w][y]=PIXEL_UNEVALUATED;
                                if(y &&
                                        IsEqual(Node[w][y-1],nTarget,nEqSpan,Pixels?TRUE:FALSE)
                                        && nQ<nQSize-1 && GET_TRANS_GLITCH(m_nTrans[w][y-1])){
                                        nQ++;
                                        pQ[nQ-1].x = w;
                                        pQ[nQ-1].y = y-1;
                                }
                                if(y<m_nY-1 &&
                                        IsEqual(Node[w][y+1],nTarget,nEqSpan,
```

```
                                        ↳ Pixels?TRUE:FALSE)
                                    && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                        ↳ m_nTrans[w][y+1])){
2360                                    nQ++;
                                        pQ[nQ-1].x = w;
                                        pQ[nQ-1].y = y+1;
                                }
                            }
2365                        while(nValidate && e<m_nX-1 && IsEqual(Node[e+1][y],↙
                                ↳ nTarget,nEqSpan,Pixels?TRUE:FALSE) && ↙
                                ↳ GET_TRANS_GLITCH(m_nTrans[e+1][y])){
                                nAreaC++;
                                nValidate--;
                                e++;
                                Node[e][y]=nDone;
2370                            if(Pixels && GET_TRANS_GLITCH(m_nTrans[e][y]))
                                        Pixels[e][y]=PIXEL_UNEVALUATED;
                                if(y &&
                                        IsEqual(Node[e][y-1],nTarget,nEqSpan,↙
                                            ↳ Pixels?TRUE:FALSE)
                                        && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                            ↳ m_nTrans[e][y-1])){
2375                                    nQ++;
                                        pQ[nQ-1].x = e;
                                        pQ[nQ-1].y = y-1;
                                }
                                if(y<m_nY-1 &&
2380                                    IsEqual(Node[e][y+1],nTarget,nEqSpan,↙
                                            ↳ Pixels?TRUE:FALSE)
                                        && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                            ↳ m_nTrans[e][y+1])){
                                        nQ++;
                                        pQ[nQ-1].x = e;
                                        pQ[nQ-1].y = y+1;
2385                            }
                            }
                            w=y;
                            e=y;
                            while(nValidate && w && IsEqual(Node[x][w-1],nTarget,↙
                                ↳ nEqSpan,Pixels?TRUE:FALSE) && GET_TRANS_GLITCH(↙
                                ↳ m_nTrans[x][w-1])){
2390                            nAreaC++;
                                nValidate--;
                                w--;
                                Node[x][w]=nDone;
                                if(Pixels && GET_TRANS_GLITCH(m_nTrans[x][w]))
2395                                    Pixels[x][w]=PIXEL_UNEVALUATED;
                                if(x &&
                                        IsEqual(Node[x-1][w],nTarget,nEqSpan,↙
                                            ↳ Pixels?TRUE:FALSE)
                                        && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                            ↳ m_nTrans[x-1][w])){
                                        nQ++;
2400                                    pQ[nQ-1].x = x-1;
                                        pQ[nQ-1].y = w;
                                }
                                if(x<m_nX-1 &&
```

```
                                        IsEqual(Node[x+1][w],nTarget,nEqSpan,↙
                                            ↳ Pixels?TRUE:FALSE)
2405                                    && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                            ↳ m_nTrans[x+1][w])){
                                        nQ++;
                                        pQ[nQ-1].x = x+1;
                                        pQ[nQ-1].y = w;
                                }
2410                        }
                            while(nValidate && e<m_nY-1 && IsEqual(Node[x][e+1],↙
                                ↳ nTarget,nEqSpan,Pixels?TRUE:FALSE) && ↙
                                ↳ GET_TRANS_GLITCH(m_nTrans[x][e+1])){
                                    nAreaC++;
                                    nValidate--;
                                    e++;
2415                                Node[x][e]=nDone;
                                    if(Pixels && GET_TRANS_GLITCH(m_nTrans[x][e]))
                                            Pixels[x][e]=PIXEL_UNEVALUATED;
                                    if(x &&
                                            IsEqual(Node[x-1][e],nTarget,nEqSpan,↙
                                                ↳ Pixels?TRUE:FALSE)
2420                                        && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                                ↳ m_nTrans[x-1][e])){
                                        nQ++;
                                        pQ[nQ-1].x = x-1;
                                        pQ[nQ-1].y = e;
                                    }
2425                                if(x<m_nX-1 &&
                                            IsEqual(Node[x+1][e],nTarget,nEqSpan,↙
                                                ↳ Pixels?TRUE:FALSE)
                                        && nQ<nQSize-1 && GET_TRANS_GLITCH(↙
                                            ↳ m_nTrans[x+1][e])){
                                        nQ++;
                                        pQ[nQ-1].x = x+1;
2430                                    pQ[nQ-1].y = e;
                                    }
                            }
                    }
            }
2435        delete [] pQ;
            return nAreaC;
    }

    // old method works better, so don't use the gradient descent for now...
2440 #undef KF_CENTER_VIA_TRANS
    int CFraktalSFT::FindCenterOfGlitch(int &ret_x, int &ret_y)
    {
            int x, y, i=0, io;
            int rx = -1, ry = -1;
2445
            itercount_array &Pixels = m_nPixels;

            uint32_t *lsb = new uint32_t[m_nX * m_nY];
            uint32_t *msb = m_nPixels_MSB ? new uint32_t[m_nX * m_nY] : nullptr;
2450        memcpy(lsb, m_nPixels_LSB, sizeof(*lsb) * m_nX * m_nY);
            if (msb)
                    memcpy(msb, m_nPixels_MSB, sizeof(*msb) * m_nX * m_nY);
```

```
        itercount_array Node(m_nY, 1, lsb, msb);

2455    int nDistance=-1;

        int nHeight = m_nY;
        if(m_bMirrored)
                nHeight=(nHeight+1)/2;
2460    for(x=0;x<m_nX;x++){
                for(y=0;y<nHeight;y++){
                        int nDone = -(x*m_nY+y);
                        if(Node[x][y]>0 && GET_TRANS_GLITCH(m_nTrans[x][y]) && ↙
                            ↳ Pixels[x][y]!=m_nMaxIter){
                                if (GetIsolatedGlitchNeighbourhood())
2465                            {
                                        const double inf = 1.0 / 0.0;
                                        double p[3][3] = { { inf, inf, inf }, { ↙
                                            ↳ inf, inf, inf }, { inf, inf, inf }↙
                                            ↳ };
                                        for (int dx = -1; dx <= 1; ++dx)
                                        {
2470                                            for (int dy = -1; dy <= 1; ++dy)
                                                {
                                                        int x2 = x + dx;
                                                        int y2 = y + dy;
                                                        if (x2 < 0 || m_nX <= x2↙
                                                            ↳ ) x2 = x - dx;
2475                                                    if (y2 < 0 || m_nY <= y2↙
                                                            ↳ ) y2 = y - dy;
                                                        p[dx+1][dy+1] = ↙
                                                            ↳ GET_TRANS_GLITCH(↙
                                                            ↳ m_nTrans[x2][y2]);
                                                }
                                        }
                                        int nMatch = 0;
2480                                    for (int dx = -1; dx <= 1; ++dx)
                                        {
                                                for (int dy = -1; dy <= 1; ++dy)
                                                {
                                                        if (↙
                                                            ↳ GetIsolatedGlitchNeighbourhood↙
                                                            ↳ () == 4 && dx && ↙
                                                            ↳ dy)
2485                                                    {
                                                                continue;
                                                        }
                                                        nMatch += p[dx+1][dy+1] ↙
                                                            ↳ == p[1][1];
                                                }
2490                                    }
                                        // this seems to try to "fix" single ↙
                                            ↳ pixel glitches by copying ↙
                                            ↳ neighbour
                                        // this doesn't seem to be desirable for↙
                                            ↳ high quality rendering
                                        // but some like it fast
                                        if(nMatch==1){
2495                                            int x2 = x ? x - 1 : x + 1;
```

```
                                                m_nTrans[x][y]=m_nTrans[x2][y];
                                                m_nPixels[x][y]=m_nPixels[x2][y↙
                                                    ↳ ];
                                                if(m_bMirrored)
                                                        Mirror(x,y);
2500                                            continue;
                                        }
                                }
                                itercount_array invalid(0, 0, nullptr, nullptr);
                                int nDist = GetArea(Node,x,y,1,invalid,nDone);
2505                            if(nDistance<nDist){
                                        nDistance=nDist;
                                        rx=x;
                                        ry=y;
                                }
2510                    }
                }
        }
        // now (rx,ry) is a point in the largest glitch of size (nDistance)
        // or (nDistance == -1) for no glitches
2515
        if(nDistance!=-1){
                for(io=0;io<m_nMaxOldGlitches;io++){
                        if(m_pOldGlitch[io].x==-1)
                                break;
                }
/*              if(io<m_nMaxOldGlitches){
                        m_pOldGlitch[io].x=rx;
                        m_pOldGlitch[io].y=ry;
                }
2525 */              ret_x=rx;
                ret_y=ry;
                int nMaxDist=0;
                int offs = 1 + nDistance/100000;
                if(io%2==0 && io>3){
2530                    // find the (rx, ry) that maximizes the minimum radius ↙
                            ↳ from the center to the edge of the glitch in 8 ↙
                            ↳ directions from the point
                        for(x=1;x<m_nX-1;x+=offs){
                                for(y=1;y<m_nY-1;y+=offs){
                                        if(Node[x][y]!=Node[ret_x][ret_y])
                                                continue;
2535                                    int tm=m_nX*m_nY, to;
                                        for(to=0;x-to>=0 && Node[x-to][y]==Node[↙
                                            ↳ ret_x][ret_y];to++);
                                        if(tm>to)
                                                tm=to;
                                        for(to=0;x+to<m_nX && Node[x+to][y]==↙
                                            ↳ Node[ret_x][ret_y];to++);
2540                                    if(tm>to)
                                                tm=to;
                                        for(to=0;y-to>=0 && Node[x][y-to]==Node[↙
                                            ↳ ret_x][ret_y];to++);
                                        if(tm>to)
                                                tm=to;
2545                                    for(to=0;y+to<m_nY && Node[x][y+to]==↙
                                            ↳ Node[ret_x][ret_y];to++);
```

```
                                        if(tm>to)
                                                tm=to;
                                        for(to=0;x-to>=0 && y-to>=0 && Node[x-to↙
                                            ↳ ][y-to]==Node[ret_x][ret_y];to++);
                                        if(tm>to)
2550                                            tm=to;
                                        for(to=0;x+to<m_nX && y+to<m_nY && Node[↙
                                            ↳ x+to][y+to]==Node[ret_x][ret_y];to↙
                                            ↳ ++);
                                        if(tm>to)
                                                tm=to;
                                        for(to=0;x-to>=0 && y+to<m_nY && Node[x-↙
                                            ↳ to][y+to]==Node[ret_x][ret_y];to↙
                                            ↳ ++);
2555                                    if(tm>to)
                                                tm=to;
                                        for(to=0;x+to<m_nX && y-to>=0 && Node[x+↙
                                            ↳ to][y-to]==Node[ret_x][ret_y];to↙
                                            ↳ ++);
                                        if(tm>to)
                                                tm=to;
2560                                    if(nMaxDist<tm){
                                                nMaxDist=tm;
                                                rx=x;
                                                ry=y;
                                        }
2565                                }
                        }
                        ret_x=rx;
                        ret_y=ry;
                }
2570             else{
                        // find the (rx, ry) that minimizes the total diameter ↙
                            ↳ between opposite ends of the glitch in 4 bi-↙
                            ↳ directions from the point
                        for(x=1;x<m_nX-1;x+=offs){
                                for(y=1;y<m_nY-1;y+=offs){
                                        if(Node[x][y]!=Node[ret_x][ret_y])
2575                                            continue;
                                        int t=0, to, c, ct;
                                        ct=c=0;
                                        for(to=0;x-to>=0 && Node[x-to][y]==Node[↙
                                            ↳ ret_x][ret_y];to++){
                                                t++;
2580                                            ct++;
                                        }
                                        for(to=0;x+to<m_nX && Node[x+to][y]==↙
                                            ↳ Node[ret_x][ret_y];to++){
                                                t++;
                                                ct--;
2585                                    }
                                        c+=(ct<0?-ct:ct);
                                        ct=0;
                                        for(to=0;y-to>=0 && Node[x][y-to]==Node[↙
                                            ↳ ret_x][ret_y];to++){
                                                t++;
2590                                            ct++;
```

```
                                                }
                                                for(to=0;y+to<m_nY && Node[x][y+to]==↙
                                                    ↳ Node[ret_x][ret_y];to++){
                                                        t++;
                                                        ct--;
2595                                            }

                                                c+=(ct<0?-ct:ct);
                                                ct=0;
                                                for(to=0;x-to>=0 && y-to>=0 && Node[x-to↙
                                                    ↳ ][y-to]==Node[ret_x][ret_y];to++){
2600                                                    t++;
                                                        ct++;
                                                }
                                                for(to=0;x+to<m_nX && y+to<m_nY && Node[↙
                                                    ↳ x+to][y+to]==Node[ret_x][ret_y];to↙
                                                    ↳ ++){
                                                        t++;
2605                                                    ct--;
                                                }
                                                c+=(ct<0?-ct:ct);
                                                ct=0;
                                                for(to=0;x-to>=0 && y+to<m_nY && Node[x-↙
                                                    ↳ to][y+to]==Node[ret_x][ret_y];to↙
                                                    ↳ ++){
2610                                                    t++;
                                                        ct++;
                                                }
                                                for(to=0;x+to<m_nX && y-to>=0 && Node[x+↙
                                                    ↳ to][y-to]==Node[ret_x][ret_y];to↙
                                                    ↳ ++){
                                                        t++;
2615                                                    ct--;
                                                }
                                                c+=(ct<0?-ct:ct);

                                                t-=c;
2620
                                                if(nMaxDist<t){
                                                        nMaxDist=t;
                                                        rx=x;
                                                        ry=y;
2625                                            }
                                        }
                                }
                                ret_x=rx;
                                ret_y=ry;
2630                    }
        }

#ifdef KF_CENTER_VIA_TRANS
        // find center of glitch via gradient descent
2635    if (nDistance != -1)
        {
          // arbitrary loop count limit to help termination guarantee
                for (int k = 0; k < nDistance; ++k)
                {
```

```
2640                            // find minimum of neighbourhood (with reflection at ↙
                                    ↳ image edges)
                                double m = 1.0 / 0.0;
                                int mx = rx;
                                int my = ry;
                                for (int dx = -1; dx <= 1; ++dx)
2645                            {
                                        int px = rx + dx;
                                        if (px < 0 || px >= m_nX) px = rx - dx;
                                        for (int dy = -1; dy <= 1; ++dy)
                                        {
2650                                            int py = ry + dy;
                                                if (py < 0 || py >= m_nY) py = ry - dy;
                                                if (m_nTrans[px][py] < m) // strict ↙
                                                    ↳ inequality for cycle prevention
                                                {
                                                        m = m_nTrans[px][py];
2655                                                    mx = px;
                                                        my = py;
                                                }
                                        }
                                }
2660                            if (mx == rx && my == ry)
                                {
                                        // found local minimum
                                        break;
                                }
2665                            rx = mx;
                                ry = my;
                        }
                        ret_x = rx;
                        ret_y = ry;
2670            }
        #endif

                if(nDistance!=-1){
                        for(io=0;io<m_nMaxOldGlitches;io++)
2675                            if(m_pOldGlitch[io].x==-1 || (m_pOldGlitch[io].x==ret_x ↙
                                    ↳ && m_pOldGlitch[io].y==ret_y))
                                        break;
                        if(io<1 && m_pOldGlitch[io].x==-1 && Center(rx,ry,FALSE,TRUE) &&↙
                            ↳  Pixels[ret_x][ret_y]==Pixels[rx][ry] && Pixels[rx][ry]!=↙
                            ↳ m_nMaxIter){
                                ret_x=rx;
                                ret_y=ry;
2680                    }
                        if(io<m_nMaxOldGlitches && m_pOldGlitch[io].x!=-1){
                                m_P.Init(m_nX, m_nY, m_bInteractive);
                                int w,h;
                                while(m_P.GetPixel(x,y,w,h)){
2685                                    if(Node[x][y]==Node[ret_x][ret_y]){
                                                for(i=0;i<m_nMaxOldGlitches && ↙
                                                    ↳ m_pOldGlitch[i].x!=-1 && !(↙
                                                    ↳ m_pOldGlitch[i].x==x && ↙
                                                    ↳ m_pOldGlitch[i].y==y);i++);
                                                if(i==m_nMaxOldGlitches || m_pOldGlitch[↙
                                                    ↳ i].x==-1){
```

```
                                                        ret_x=x;
                                                        ret_y=y;
2690                                                    break;
                                                }
                                        }
                                }
                        }
                        for(;io<m_nMaxOldGlitches && m_pOldGlitch[io].x!=-1;io++);
                        if(io<m_nMaxOldGlitches){
                                m_pOldGlitch[io].x=ret_x;
                                m_pOldGlitch[io].y=ret_y;
                        }
2700            }
                delete[] lsb;
                delete[] msb;
                return nDistance + 1; // -1 becomes 0
        }
2705    #undef KF_CENTER_VIA_TRANS

        BOOL CFraktalSFT::GetTransition()
        {
                return m_bTrans;
2710    }
        void CFraktalSFT::SetTransition(BOOL bTransition)
        {
                m_bTrans = bTransition;
        }
2715    BOOL CFraktalSFT::GetITransition()
        {
                return m_bITrans;
        }
        void CFraktalSFT::SetITransition(BOOL bITransition)
2720    {
                m_bITrans = bITransition;
        }

        int CFraktalSFT::GetColorIndex(int x, int y)
2725    {
                if (x<0 || x >= m_nX || y<0 || y >= m_nY || !m_nPixels)
                        return -1;
                return ((((int64_t)floor(m_nPixels[x][y] / m_nIterDiv)) % 1024) + 1024) ↙
                    ↳ % 1024;
        }
2730    void CFraktalSFT::SaveMap(const std::string &szFile)
        {
                if (!m_nPixels)
                        return;
                DWORD dw;
2735            HANDLE hFile = CreateFile(szFile.c_str(), GENERIC_WRITE, 0, NULL, ↙
                    ↳ CREATE_ALWAYS, 0, NULL);
                int x, y;
                char szNum[64];
                for (y = 0; y<m_nY; y++){
                        if (y)
2740                            WriteFile(hFile, "\r\n", 2, &dw, NULL);
                        for (x = 0; x<m_nX; x++){
                                if (x)
```

```
                                        WriteFile(hFile, " ", 1, &dw, NULL);
                                itoa(m_nPixels[x][y], szNum, 10);
2745                            WriteFile(hFile, szNum, strlen(szNum), &dw, NULL);
                        }
                }
                const char *szC0 = "\r\nColors: ";
                WriteFile(hFile, szC0, strlen(szC0), &dw, NULL);
2750            CStringTable stColors;
                int i;
                for (i = 0; i<m_nParts; i++){
                        stColors.AddRow();
                        stColors.AddInt(stColors.GetCount() - 1, m_cKeys[i].r);
2755                    stColors.AddRow();
                        stColors.AddInt(stColors.GetCount() - 1, m_cKeys[i].g);
                        stColors.AddRow();
                        stColors.AddInt(stColors.GetCount() - 1, m_cKeys[i].b);
                }
2760            char *szC = stColors.ToText("", ",");
                WriteFile(hFile, szC, strlen(szC), &dw, NULL);
                stColors.DeleteToText(szC);
                CloseHandle(hFile);
        }
2765    void CFraktalSFT::SaveMapB(const std::string &szFile)
        {
                // WONTFIX doesn't save m_nPixels_MSB
                if (!m_nPixels_LSB)
                        return;
2770            DWORD dw;
                HANDLE hFile = CreateFile(szFile.c_str(), GENERIC_WRITE, 0, NULL, ↙
                    ↳ CREATE_ALWAYS, 0, NULL);
                int x;
                WriteFile(hFile, "KFB", 3, &dw, NULL);
                WriteFile(hFile, &m_nX, sizeof(m_nX), &dw, NULL);
2775            WriteFile(hFile, &m_nY, sizeof(m_nY), &dw, NULL);
                WriteFile(hFile, m_nPixels_LSB, sizeof(*m_nPixels_LSB) * m_nX * m_nY, &↙
                    ↳ dw, NULL);
                int div = m_nIterDiv;
                WriteFile(hFile, &div, sizeof(m_nParts), &dw, NULL);
                WriteFile(hFile, &m_nParts, sizeof(m_nParts), &dw, NULL);
2780            WriteFile(hFile, m_cKeys, sizeof(COLOR14)*m_nParts, &dw, NULL);
                WriteFile(hFile, &m_nMaxIter, sizeof(int), &dw, NULL);
                for (x = 0; x<m_nX; x++)
                        WriteFile(hFile, m_nTrans[x], m_nY*sizeof(float), &dw, NULL);
                if (GetDerivatives() && m_nDEx)
2785            {
                        float *column = new float[m_nY];
                        for (x = 0; x<m_nX; x++)
                        {
                                for (int y = 0; y < m_nY; ++y)
2790                                    column[y] = std::hypot(m_nDEx[x][y], m_nDEy[x][y↙
                                            ↳ ]);
                                WriteFile(hFile, column, m_nY*sizeof(float), &dw, NULL);
                        }
                        delete[] column;
                }
2795            CloseHandle(hFile);
        }
```

```
SmoothMethod CFraktalSFT::GetSmoothMethod()
{
        return m_nSmoothMethod;
}
void CFraktalSFT::SetSmoothMethod(int nSmoothMethod)
{
        if (nSmoothMethod == 0){
                m_nSmoothMethod = SmoothMethod(nSmoothMethod);
                m_nBailout = SMOOTH_BAILOUT;
                m_nBailout2 = m_nBailout*m_nBailout;
        }
        else if (nSmoothMethod == 1){
                m_nSmoothMethod = SmoothMethod_Sqrt;
                m_nBailout = 2;
                m_nBailout2 = m_nBailout*m_nBailout;
        }
        else{
                m_nSmoothMethod = SmoothMethod_SqrtLow;
                m_nBailout = pow(2.0, 1.0 / (m_nPower - 1));
                m_nBailout2 = m_nBailout*m_nBailout;
        }
}
int CFraktalSFT::GetPower()
{
        return m_nPower;
}
void CFraktalSFT::SetPower(int nPower)
{
        m_nPower = nPower;
        if (m_nPower<2)
                m_nPower = 2;
        if (m_nPower>70)
                m_nPower = 70;
//      if (m_nFractalType>4 && m_nPower>3)
//              m_nPower = 3;
        if (g_nLDBL>100){
                if (m_nPower == 2 && scaling_supported(m_nFractalType, m_nPower,↙
                    ↳ GetDerivatives()))
                        g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_2;
                else if (m_nPower == 3 && scaling_supported(m_nFractalType, ↙
                    ↳ m_nPower, GetDerivatives()))
                        g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_3;
                else
                        g_nLDBL = LONG_DOUBLE_THRESHOLD_DEFAULT;
        }
        SetSmoothMethod(m_nSmoothMethod); // update bailout if necessary
}

void CFraktalSFT::SetDifferences(int nDifferences)
{
        if (nDifferences < 0) nDifferences = 0;
        if (nDifferences > 7) nDifferences = 0;
        m_nDifferences = Differences(nDifferences);
}
Differences CFraktalSFT::GetDifferences()
{
```

```
        return m_nDifferences;
}

void CFraktalSFT::SetColorMethod(int nColorMethod)
{
  if (nColorMethod < 0) nColorMethod = 0;
  if (nColorMethod > 10) nColorMethod = 0;
        m_nColorMethod = ColorMethod(nColorMethod);
}
ColorMethod CFraktalSFT::GetColorMethod()
{
        return m_nColorMethod;
}
void CFraktalSFT::SetColorOffset(int nColorOffset)
{
        while (nColorOffset<0)
                nColorOffset += 1024;
        while (nColorOffset >= 1024)
                nColorOffset -= 1024;
        m_nColorOffset = nColorOffset;
}
int CFraktalSFT::GetColorOffset()
{
        return m_nColorOffset;
}
void CFraktalSFT::ErasePixel(int x, int y)
{
        if (x >= 0 && y >= 0 && x<m_nX && y<m_nY){
                m_nPixels[x][y] = 1;
                m_nTrans[x][y] = 0;
                m_nDEx[x][y] = 0;
                m_nDEy[x][y] = 0;
                int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
                SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, 1, 1);
                m_nPixels[x][y] = PIXEL_UNEVALUATED;
        }
}
void CFraktalSFT::StoreLocation()
{
        m_storedr = m_rref;
        m_storedi = m_iref;
}
void CFraktalSFT::SetMW(BOOL bMW, BOOL bBlend)
{
        m_bMW = bMW;
        m_bBlend = bBlend;
}
int CFraktalSFT::GetMWCount()
{
        return m_nMW;
}
int CFraktalSFT::GetMW(BOOL *pbBlend)
{
        if (pbBlend)
                *pbBlend = m_bBlend;
        return m_bMW;
}
```

```
BOOL CFraktalSFT::GetMW(int nIndex, int &nPeriod, int &nStart, int &nType)
{
	if (nIndex<0 || nIndex >= m_nMW)
		return FALSE;
	nPeriod = m_MW[nIndex].nPeriod;
	nStart = m_MW[nIndex].nStart;
	nType = m_MW[nIndex].nType;
	return TRUE;
}
BOOL CFraktalSFT::AddMW(int nPeriod, int nStart, int nType)
{
	if (m_nMW == MULTIWAVE_MAX - 1)
		return FALSE;
	int i = m_nMW++;
	m_MW[i].nPeriod = nPeriod;
	m_MW[i].nStart = nStart;
	m_MW[i].nType = nType;
	return TRUE;
}
BOOL CFraktalSFT::UpdateMW(int nIndex, int nPeriod, int nStart, int nType)
{
	if (nIndex<0 || nIndex >= m_nMW)
		return FALSE;
	m_MW[nIndex].nPeriod = nPeriod;
	m_MW[nIndex].nStart = nStart;
	m_MW[nIndex].nType = nType;
	return TRUE;
}
BOOL CFraktalSFT::DeleteMW(int nIndex)
{
	int i;
	if (!m_nMW)
		return FALSE;
	m_nMW--;
	for (i = nIndex; i<m_nMW; i++)
		m_MW[i].nPeriod = m_MW[i + 1].nPeriod;
	return TRUE;
}
int64_t CFraktalSFT::GetMaxExceptCenter()
{
	int64_t nMax = 0;
	int x, y;
	int nXO = m_nX / 2 - 2;
	int nYO = m_nY / 2 - 2;
	for (x = 0; x<m_nX; x++)
	for (y = 0; y<m_nY; y++)
	if ((x<nXO || x>m_nX - nXO) && (y<m_nY - nYO || y>3 * nYO) && nMax<↙
	    ↳ m_nPixels[x][y])
		nMax = m_nPixels[x][y];
	return nMax;
}
void CFraktalSFT::SetFractalType(int nFractalType)
{
	m_nFractalType = nFractalType;
	if ((m_nFractalType == 1 || m_nFractalType == 2) && (m_nPower<2 || ↙
	    ↳ m_nPower>3))
		m_nPower = 2;
```

```
        if (m_nFractalType>2 && m_nPower>2)
2965            m_nPower = 2;

        if (g_nLDBL>100){
            if (m_nPower == 2 && scaling_supported(m_nFractalType, m_nPower,↙
                ↳ GetDerivatives()))
                g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_2;
2970        else if (m_nPower == 3 && scaling_supported(m_nFractalType, ↙
                ↳ m_nPower, GetDerivatives()))
                g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_3;
            else
                g_nLDBL = LONG_DOUBLE_THRESHOLD_DEFAULT;
        }
2975 }
    int CFraktalSFT::GetFractalType()
    {
        return m_nFractalType;
    }
2980
    int CFraktalSFT::GetExponent()
    {
        return m_nZoom;
    }
2985
    void CFraktalSFT::SetApproxTerms(int64_t nTerms)
    {
        m_Settings.SetApproxTerms(nTerms);
        int m_nTerms = GetApproxTerms();
2990    delete[] m_APr;
        delete[] m_APi;
        m_APr = new floatexp[m_nTerms];
        m_APi = new floatexp[m_nTerms];
    }
2995
    void CFraktalSFT::SetHalfColour(bool b)
    {
        m_Settings.SetHalfColour(b);
        b = m_Settings.GetHalfColour();
3000    if (b)
        {
            if (! m_imageHalf)
            {
                m_imageHalf = new half[m_nSizeImage];
3005        }
        }
        else
        {
            if (m_imageHalf)
3010        {
                delete[] m_imageHalf;
                m_imageHalf = nullptr;
            }
        }
3015 }

    double CFraktalSFT::GetRatioX()
    {
```

```
        return m_scRatio.cx;
}
double CFraktalSFT::GetRatioY()
{
        return m_scRatio.cy;
}
void CFraktalSFT::SetRatio(double x, double y)
{
        double xRatio = 640.0/x;
        m_scRatio.cx = 640;
        m_scRatio.cy = y*xRatio;
        CStringTable st;
        st.AddRow();
        st.AddString(0, GetRe());
        st.AddString(0, GetIm());
        SetPosition(st[0][0], st[0][1], GetZoom());
}

BOOL CFraktalSFT::GetSlopes(int &nSlopePower, int &nSlopeRatio, int &nSlopeAngle )
{
        nSlopePower = m_nSlopePower;
        nSlopeRatio = m_nSlopeRatio;
        nSlopeAngle = m_nSlopeAngle;
        return m_bSlopes;
}
void CFraktalSFT::SetSlopes(BOOL bSlope, int nSlopePower, int nSlopeRatio, int nSlopeAngle)
{
        m_bSlopes = bSlope;
        m_nSlopePower = nSlopePower;
        if (m_nSlopePower<1)
                m_nSlopePower = 1;
        m_nSlopeRatio = nSlopeRatio;
        if (m_nSlopeRatio<0)
                m_nSlopeRatio = 0;
        m_nSlopeAngle = nSlopeAngle;

        double lightAngleRadians = pi*(double)m_nSlopeAngle / 180;
        m_nSlopeX = cos(lightAngleRadians);
        m_nSlopeY = sin(lightAngleRadians);
}
BOOL CFraktalSFT::GetTexture(double &nImgMerge,double &nImgPower,int &nImgRatio,std::string &szTexture)
{
        nImgMerge = m_nImgMerge;
        nImgPower = m_nImgPower;
        nImgRatio = m_nImgRatio;
        szTexture = m_szTexture;
        return m_bTexture;
}
void CFraktalSFT::SetTexture(BOOL bTexture,double nImgMerge,double nImgPower,int nImgRatio,const std::string &szTexture)
{
        delete[] m_lpTextureBits;
        m_lpTextureBits=NULL;
        m_bTexture = bTexture;
```

```
        m_nImgMerge = nImgMerge;
        m_nImgPower = nImgPower;
        m_nImgRatio = nImgRatio;
        m_szTexture = szTexture;
}

void CFraktalSFT::AddInflectionPont(int nXPos, int nYPos)
{
        m_C = cos(g_Degree);
        m_S = sin(g_Degree);
        double mr = m_nX / 2;
        double mi = m_nY / 2;
        double ratio = (((double)m_nY/(double)m_nX)/(360.0/640.0)) * ((double)
            ↪ 360 / (double)m_scRatio.cy);
        double xpos = (nXPos - mr)*ratio + mr;
        double dbD0r = mr + m_C*(xpos - mr) + m_S*(nYPos - mi);
        double dbD0i = mi - m_S*(xpos - mr) + m_C*(nYPos - mi);
        dbD0r = (dbD0r - mr) / ratio + mr;
        complex<CFixedFloat> inflect;
        inflect.m_r = (CFixedFloat)dbD0r*(m_rstop - m_rstart)*(CFixedFloat)((
            ↪ double)1 / m_nX) + m_rstart;
        inflect.m_i = (CFixedFloat)dbD0i*(m_istop - m_istart)*(CFixedFloat)((
            ↪ double)1 / m_nY) + m_istart;
        m_Inflections.push_back(inflect);
}
void CFraktalSFT::RemoveInflectionPoint()
{
        if (m_Inflections.size() > 0)
                m_Inflections.pop_back();
}

#ifdef KF_OPENCL
int CFraktalSFT::GetOpenCLDeviceIndex()
{
        return clid;
}
void CFraktalSFT::SetOpenCLDeviceIndex(int i)
{
        if (i != clid)
        {
                if (cl)
                {
                        delete cl;
                        cl = NULL;
                        clid = -1;
                }
                if (0 <= i && i < cldevices.size())
                {
                        clid = i;
                        cl = new OpenCL(cldevices[i].pid, cldevices[i].did);
                }
        }
}
#endif

CPixels::CPixels()
{
```

```
        m_pPixels = NULL;
        m_hMutex = CreateMutex(NULL, 0, NULL);
}

struct CPixel
{
        uint16_t x, y; uint8_t w, h;
        CPixel(int x, int y, int w, int h) : x(x), y(y), w(w), h(h) { }
        CPixel() : CPixel(0, 0, 0, 0) { }
};

struct CPixelComparator {
        double x;
        double y;
        CPixelComparator(int x, int y) : x(x), y(y) { }
  bool operator() (const CPixel &a, const CPixel &b)
  {
                double dax = a.x - x;
                double day = a.y - y;
                double dbx = b.x - x;
                double dby = b.y - y;
                double da = dax * dax + day * day;
                double db = dbx * dbx + dby * dby;
                return da < db;
        }
};

void CPixels::Init(int width, int height, bool interactive)
{
        m_nNextPixel = -1;
        if (m_nX == width && m_nY == height && m_pPixels)
                return;
        m_nX = width;
        m_nY = height;
        m_nY2 = m_nY >> 1;
        m_nPixels = m_nX*m_nY;
        if (m_pPixels)
                delete[] m_pPixels;
        CPixel *pixels = m_pPixels = new CPixel[m_nPixels];

  // Adam7-style interlacing
  CPixelComparator cmp(width >> 1, height >> 1);
        int step = 1 << 7;
        int ix = 0;
        int begin = ix;
        for (int y = 0; y < height; y += step)
                for (int x = 0; x < width; x += step)
                        pixels[ix++] = CPixel(x, y, step, step);
        int end = ix;
        if (interactive) std::sort(&pixels[begin], &pixels[end], cmp);
  for (; step > 1; step >>= 1)
  {
                begin = ix;
                for (int y = 0;  y < height;  y += step)
                        for (int x = step >> 1; x < width; x += step)
                                pixels[ix++] = CPixel(x, y, step >> 1, step);
                end = ix;
```

```
                if (interactive) std::sort(&pixels[begin], &pixels[end], cmp);
                begin = ix;
3185            for (int y = step >> 1; y < height; y += step)
                        for (int x = 0; x < width; x += step >> 1)
                                pixels[ix++] = CPixel(x, y, step >> 1, step >> ↙
                                    ↳ 1);
                end = ix;
                if (interactive) std::sort(&pixels[begin], &pixels[end], cmp);
3190    }
        //assert(ix == width * height);
}

BOOL CPixels::GetPixel(int &rx, int &ry, int &rw, int &rh, BOOL bMirrored)
3195 {
        do{
                int nNext = InterlockedIncrement(&m_nNextPixel);
                if (nNext < m_nPixels){
                        rx = m_pPixels[nNext].x;
3200                    ry = m_pPixels[nNext].y;
                        rw = m_pPixels[nNext].w;
                        rh = m_pPixels[nNext].h;
                        if (bMirrored && ry>m_nY2)
                                continue;
3205                    return TRUE;
                }
                return FALSE;
        } while (bMirrored);
        return FALSE;
3210 }

void CFraktalSFT::OutputIterationData(int x, int y, int w, int h, bool bGlitch, ↙
    ↳ int64_t antal, double test1, double test2, const complex<double> &de)
{
                int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
3215            if (antal == m_nGlitchIter)
                        bGlitch = TRUE;
                if (antal >= m_nMaxIter){
                        m_nPixels[x][y] = antal;
                        m_nTrans[x][y] = 0;
3220                    m_nDEx[x][y] = 0;
                        m_nDEy[x][y] = 0;
                        m_lpBits[nIndex] = 0;
                        m_lpBits[nIndex + 1] = 0;
                        m_lpBits[nIndex + 2] = 0;
3225            }
                else{

                        m_nPixels[x][y] = antal;
                        if (m_nDEx)
3230                    {
                                m_nDEx[x][y] = de.m_r;
                        }
                        if (m_nDEy)
                        {
3235                            m_nDEy[x][y] = de.m_i;
                        }
```

```
                        if (!bGlitch && (m_nSmoothMethod == SmoothMethod_Sqrt ||
                            m_nSmoothMethod == SmoothMethod_SqrtLow)){
                                double div = sqrt(test1) - sqrt(test2);
                                if (div != 0)
                                        m_nTrans[x][y] = (sqrt(test1) -
                                            m_nBailout) / div;
                                else
                                        m_nTrans[x][y] = 0;
                        }

                        else if (!bGlitch && m_nSmoothMethod == SmoothMethod_Log
                            ){
                                double t = log(log(sqrt(test1))) / log((double)
                                    m_nPower);
                                if (!ISFLOATOK(t))
                                        t = 0;
                                int64_t i = floor(t);
                                auto r = m_nPixels[x][y];
                                r = r - i;
                                m_nTrans[x][y] = t - i;
                        }

                        if (bGlitch && !m_bNoGlitchDetection){
                                m_nTrans[x][y] = SET_TRANS_GLITCH(test1);
                        }

                        SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y,
                            w, h);
                }
                if (m_bMirrored)
                        Mirror(x, y);
}

void CFraktalSFT::OutputPixelData(int x, int y, int w, int h, bool bGlitch)
{
                if ((!bGlitch || GetShowGlitches()) && ! m_bInhibitColouring)
    {
      int nIndex = x * 3 + (m_bmi->biHeight - 1 - y) * m_row;
      for (int ty = 0; ty < h; ++ty)
      {
        int y2 = y + ty;
        if (y2 < m_nY)
        {
          for (int tx = 0; tx < w; ++tx)
          {
            int x2 = x + tx;
            if (x2 < m_nX)
            {
              if (m_nPixels[x2][y2] == PIXEL_UNEVALUATED)
              {
                int index2 = x2 * 3 + (m_bmi->biHeight - 1 - y2) * m_row;
                m_lpBits[index2    ] = m_lpBits[nIndex    ];
                m_lpBits[index2 + 1] = m_lpBits[nIndex + 1];
                m_lpBits[index2 + 2] = m_lpBits[nIndex + 2];
              }
            }
          }
```

```
3290                                    }
                                }
                        }
        }

3295    bool CFraktalSFT::GuessPixel(int x, int y, int x0, int y0, int x1, int y1)
        {
                if ( 0 <= x && x < m_nX && 0 <= x0 && x0 < m_nX && 0 <= x1 && x1 < m_nX ↙
                    ↳ &&
                     0 <= y && y < m_nY && 0 <= y0 && y0 < m_nY && 0 <= y1 && y1 < m_nY ↙
                         ↳ &&
                     m_nPixels[x0][y0] != PIXEL_UNEVALUATED &&
3300                 m_nPixels[x0][y0] == m_nPixels[x1][y1] &&
                     GET_TRANS_GLITCH(m_nTrans[x0][y0]) == GET_TRANS_GLITCH(m_nTrans[x1↙
                         ↳ ][y1])
                   )
                {
                        // NOTE cast to int64_t is required to avoid copying just the ↙
                            ↳ ref!
3305                    m_nPixels[x][y] = int64_t(m_nPixels[x0][y0]);
                        m_nTrans[x][y] = (m_nTrans[x0][y0] + m_nTrans[x1][y1])*.5;
                        // use geometric mean for directional DE guessing
        #ifdef KF_GUESS_DE_GEOMETRIC
                        complex<float> de0(m_nDEx[x0][y0], m_nDEy[x0][y0]);
3310                    complex<float> de1(m_nDEx[x1][y1], m_nDEy[x1][y1]);
                        complex<float> deA = 0.5*(de0 + de1);
                        complex<float> deG = sqrt(de0 * de1);
                        complex<float> de = (norm(deA - deG) < norm(deA + deG)) ? deG : ↙
                            ↳ -1.0*deG;
        #else
3315                    m_nDEx[x][y] = 0.5 * (m_nDEx[x0][y0] + m_nDEx[x1][y1]);
                        m_nDEy[x][y] = 0.5 * (m_nDEy[x0][y0] + m_nDEy[x1][y1]);
        #endif
                        int nIndex  = x  * 3 + (m_bmi->biHeight - 1 - y )*m_row;
                        int nIndex0 = x0 * 3 + (m_bmi->biHeight - 1 - y0)*m_row;
3320                    int nIndex1 = x1 * 3 + (m_bmi->biHeight - 1 - y1)*m_row;
                        m_lpBits[nIndex    ] = (m_lpBits[nIndex0    ] + m_lpBits[nIndex1↙
                            ↳     ]) / 2;
                        m_lpBits[nIndex + 1] = (m_lpBits[nIndex0 + 1] + m_lpBits[nIndex1↙
                            ↳ + 1]) / 2;
                        m_lpBits[nIndex + 2] = (m_lpBits[nIndex0 + 2] + m_lpBits[nIndex1↙
                            ↳ + 2]) / 2;
                        InterlockedIncrement((LPLONG)&m_nDone);
3325                    InterlockedIncrement((LPLONG)&m_nGuessed);
                        if (m_bMirrored)
                                Mirror(x, y);
                        return true;
                }
3330            return false;
        }

        bool CFraktalSFT::GuessPixel(int x, int y, int w, int h)
        {
3335            if (GetGuessing())
                {
                        if (w == 1 && h <= 2)
                                if (GuessPixel(x, y, x - 1, y    , x + 1, y    )) return↙
```

```
                                ↪ true;
                        if (w == 1 && h == 1) {
3340                            if (GuessPixel(x, y, x    , y - 1, x    , y + 1)) return↙
                                    ↪ true;
                                if (GuessPixel(x, y, x - 1, y - 1, x + 1, y + 1)) return↙
                                    ↪ true;
                                if (GuessPixel(x, y, x - 1, y + 1, x + 1, y - 1)) return↙
                                    ↪ true;
                  }
                }
3345            return false;
        }

        // http://www.burtleburtle.net/bob/hash/integer.html
        #if 0
3350    static uint32_t wang_hash(uint32_t a)
        {
            a = (a ^ 61) ^ (a >> 16);
            a = a + (a << 3);
            a = a ^ (a >> 4);
3355        a = a * 0x27d4eb2d;
            a = a ^ (a >> 15);
            return a;
        }
        #else
3360    static uint32_t burtle_hash(uint32_t a)
        {
            a = (a+0x7ed55d16) + (a<<12);
            a = (a^0xc761c23c) ^ (a>>19);
            a = (a+0x165667b1) + (a<<5);
3365        a = (a+0xd3a2646c) ^ (a<<9);
            a = (a+0xfd7046c5) + (a<<3);
            a = (a^0xb55a4f09) ^ (a>>16);
            return a;
        }
3370    #endif

        // uniform in [0,1)
        static double dither(uint32_t x, uint32_t y, uint32_t c)
        {
3375      return burtle_hash(x + burtle_hash(y + burtle_hash(c))) / (double) (0↙
              ↪ x100000000LL);
        }

        void CFraktalSFT::GetPixelOffset(const int i, const int j, double &x, double &y)↙
            ↪ const
        {
3380            int c = GetJitterSeed();
                if (c)
                {
                        double s = GetJitterScale();
                        double u = dither(i, j, 2 * c + 0);
3385                    double v = dither(i, j, 2 * c + 1);
                        switch (GetJitterShape())
                        {
                                default:
                                case 0: // uniform
```

```
3390                                    {
                                                x = s * (u - 0.5);
                                                y = s * (v - 0.5);
                                        }
                                        break;
3395                            case 1: // Gaussian
                                        {
                                                // FIXME cache slow trig functions for ↙
                                                    ↳ every pixel for every image?
                                                // https://en.wikipedia.org/wiki/Box%E2↙
                                                    ↳ %80%93Muller_transform
                                                double r = 0 < u && u < 1 ? sqrt(-2 * ↙
                                                    ↳ log(u)) : 0;
3400                                            double t = 2 * 3.141592653589793 * v;
                                                s *= 0.5;
                                                x = s * r * cos(t);
                                                y = s * r * sin(t);
                                        }
3405                                    break;
                        }
                }
                else
                {
3410                    x = 0.0;
                        y = 0.0;
                }
        }

3415    void CFraktalSFT::GetPixelCoordinates(const int i, const int j, floatexp &x, ↙
            ↳ floatexp &y) const
        {
                double di = 0;
                double dj = 0;
                GetPixelOffset(i, j, di, dj);
3420            floatexp u = (i - m_nX/2 + di) * m_pixel_step_x;
                floatexp v = (j - m_nY/2 + dj) * m_pixel_step_y;
                x = m_pixel_center_x + m_C * u + m_S * v;
                y = m_pixel_center_y - m_S * u + m_C * v;
        }
3425
        void CFraktalSFT::GetPixelCoordinates(const int i, const int j, floatexp &x, ↙
            ↳ floatexp &y, floatexp &daa, floatexp &dab, floatexp &dba, floatexp &dbb) ↙
            ↳ const
        {
                double di = 0;
                double dj = 0;
3430            GetPixelOffset(i, j, di, dj);
                floatexp u = (i - m_nX/2 + di) * m_pixel_step_x;
                floatexp v = (j - m_nY/2 + dj) * m_pixel_step_y;
                x = m_pixel_center_x + m_C * u + m_S * v;
                y = m_pixel_center_y - m_S * u + m_C * v;
3435            daa =  m_C * m_pixel_step_x;
                dab =  m_S * m_pixel_step_y;
                dba = -m_S * m_pixel_step_x;
                dbb =  m_C * m_pixel_step_y;
        }
```

66 fraktal_sft/fraktal_sft.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_FRAKTAL_SFT_H
#define KF_FRAKTAL_SFT_H 1

#include <windows.h>
#include <half.h>

#include "Settings.h"
#include "CFixedFloat.h"
#include "CDecNumber.h"
#include "complex.h"
#include "floatexp.h"
#include "../common/matrix.h"
#include "itercount_array.h"

#ifdef KF_OPENCL
#include "../cl/opencl.h"
extern std::vector<cldevice> cldevices;
#endif

struct NanoMB1_Reference;
struct NanoMB2_Reference;

struct CPixel;
class CPixels
{
        int m_nX;
        int m_nY;
        int m_nY2;
        CPixel *m_pPixels;
        int m_nPixels;
        LONG m_nNextPixel;
        HANDLE m_hMutex;
public:
        CPixels();
        void Init(int nX, int nY, bool interactive);
        BOOL GetPixel(int &x, int &y, int &w, int &h, BOOL bMirrored = 0);
```

```
#if 0
        BOOL GetPixels(int *px, int *py, int &nCount);
#endif
};

// magic value stored in m_nPixels[][] when pixel needs (re)computation
#define PIXEL_UNEVALUATED INT_MIN
// magic value stored in m_nTrans[][] when a glitch is detected
#if 0
#define SET_TRANS_GLITCH(x) (fmin(log2((x) + 2.2250738585072014e-308) - 1024.0, -1.0))
#else
#define SET_TRANS_GLITCH(x) (-1.0)
#endif
#define GET_TRANS_GLITCH(x) ((x) < 0.0f)

// thresholds for switching to floatexp iterations
#define FLOATEXP_THRESHOLD_POWER_2 9800
#define FLOATEXP_THRESHOLD_POWER_3 6533
#define FLOATEXP_THRESHOLD_DEFAULT 4900
// thresholds for switching to long double iterations
#define LONG_DOUBLE_THRESHOLD_POWER_2 590
#define LONG_DOUBLE_THRESHOLD_POWER_3 390
#define LONG_DOUBLE_THRESHOLD_DEFAULT 290
// threshold for switching to scaled double iterations
// this is lower than the theoretical maximum to avoid derivative overflow
#define SCALED_DOUBLE_THRESHOLD 290

#define SMOOTH_BAILOUT 10000
struct MC
{
        CFixedFloat *xr, *xi, *sr, *si, *xrxid;
        HANDLE hDone;
        HANDLE hWait;
        HANDLE hExit;
        int nType;
};
struct MC2
{
        CFixedFloat *xrn, *xin, *xrxid, *sr, *si, *m_iref, *m_rref;
        HANDLE hDone;
        HANDLE hWait;
        HANDLE hExit;
        int nType;
};


struct COLOR14 { unsigned char r, g, b; };

#if 0
typedef long double ldbl;

struct ldblexp {
        ldbl val;
        __int64 exp;
};
#endif
```

```
#define MULTIWAVE_MAX 30
struct MULTIWAVE
{
        int nPeriod;
        int nStart;
        int nType;
};

#if 0
template <class T> class CFileFloat
{
        HANDLE m_hFile;
        int m_nSize;
        int m_nStartPos;
        T *m_dbBuf;
        int m_nViewSize;
        char m_szFileName[256];
public:
        CFileFloat(int nSize)
        {
                m_nViewSize = 1000;
                m_dbBuf = new T[m_nViewSize];
                memset(m_dbBuf,0,m_nViewSize*sizeof(T));
                m_nSize = nSize;
                m_nStartPos=0;
                GetTempPath(sizeof(m_szFileName),m_szFileName);
                if(m_szFileName[strlen(m_szFileName)-1]!='\\')
                        strcat(m_szFileName,"\\");
                do{
                        wsprintf(strrchr(m_szFileName,'\\')+1,"%d",GetTickCount↙
                            ↪ ());
                        m_hFile = CreateFile(m_szFileName,GENERIC_READ|↙
                            ↪ GENERIC_WRITE,0,NULL,CREATE_NEW,0,NULL);
                }while(m_hFile==INVALID_HANDLE_VALUE);
                int nData=0;
                DWORD dw;
                while(nData<nSize){
                        WriteFile(m_hFile,m_dbBuf,m_nViewSize*sizeof(T),&dw,NULL↙
                            ↪ );
                        nData+=m_nViewSize;
                }
                SetFilePointer(m_hFile,0,0,FILE_BEGIN);
        }
        ~CFileFloat()
        {
                CloseHandle(m_hFile);
                DeleteFile(m_szFileName);
                delete m_dbBuf;
        }
        T &operator[] (int nIndex)
        {
                int nMapIndex = nIndex-m_nStartPos;
                int nPrevStart = m_nStartPos;
                BOOL bRemap=FALSE;
                while(nMapIndex<0){
                        m_nStartPos-=m_nViewSize;
```

```
                        nMapIndex = nIndex-m_nStartPos;
                        bRemap=TRUE;
                }
                while(nMapIndex>=m_nViewSize){
                        m_nStartPos+=m_nViewSize;
                        nMapIndex = nIndex-m_nStartPos;
                        bRemap=TRUE;
                }
                if(bRemap){
                        __int64 nStartPos = (__int64)nPrevStart*sizeof(T);
                        long *pPos = (long *)&nStartPos;
                        SetFilePointer(m_hFile,pPos[0],&pPos[1],FILE_BEGIN);
                        DWORD dw;
                        WriteFile(m_hFile,m_dbBuf,m_nViewSize*sizeof(T),&dw,NULL);
                        nStartPos = (__int64)m_nStartPos*sizeof(T);
                        pPos = (long *)&nStartPos;
                        SetFilePointer(m_hFile,pPos[0],&pPos[1],FILE_BEGIN);
                        ReadFile(m_hFile,m_dbBuf,m_nViewSize*sizeof(T),&dw,NULL);
                }
                return m_dbBuf[nMapIndex];
        }
};
#endif

enum SmoothMethod
{
        SmoothMethod_Log = 0,
        SmoothMethod_Sqrt = 1,
        SmoothMethod_SqrtLow = 2
};

enum ColorMethod
{
        ColorMethod_Standard = 0,
        ColorMethod_SquareRoot = 1,
        ColorMethod_CubicRoot = 2,
        ColorMethod_Logarithm = 3,
        ColorMethod_Stretched = 4,
        ColorMethod_DistanceLinear = 5,
        ColorMethod_DEPlusStandard = 6,
        ColorMethod_DistanceLog = 7,
        ColorMethod_DistanceSqrt = 8,
        ColorMethod_LogLog = 9,
        ColorMethod_ATan = 10
};

enum Differences
{
        Differences_Traditional = 0,
        Differences_Forward3x3 = 1,
        Differences_Central3x3 = 2,
        Differences_Diagonal2x2 = 3,
        Differences_LeastSquares2x2 = 4,
        Differences_LeastSquares3x3 = 5,
        Differences_Laplacian3x3 = 6,
```

```
        Differences_Analytic = 7
};

extern double g_Degree;

enum SeriesType
{
        SeriesType_None = 0,
        SeriesType_Complex = 1,
        SeriesType_Real = 2
};

struct SeriesR2
{
        floatexp s[MAX_APPROX_TERMS+1][MAX_APPROX_TERMS+1];
        floatexp t[MAX_APPROX_TERMS+1][MAX_APPROX_TERMS+1];
};

class CFraktalSFT
{
        Settings m_Settings;

        MULTIWAVE mMW[MULTIWAVE_MAX];
        int mnMW;
        BOOL mbMW;
        BOOL m_bBlend;
        HANDLE m_hMutex;
        CPixels m_P;
        CFixedFloat m_rstart, m_istart, m_rstop, m_istop, m_rref, m_iref;
        double m_dPixelSpacing;
        long double m_lPixelSpacing;
        floatexp m_fPixelSpacing;
        double m_epsilon;
        CFixedFloat m_storedr, m_storedi;
        POINT m_pOldGlitch[OLD_GLITCH];
        int m_nSizeImage;
        int m_nZoom;
        int64_t m_nMaxIter;
        int64_t m_nGlitchIter;
        int m_nTotal;
        HBITMAP m_bmBmp;
        COLOR14 m_cPos[1025], m_cKeys[1025], m_cInterior;
        int m_nParts;
        int m_nSeed;
        uint32_t *m_nPixels_LSB;
        uint32_t *m_nPixels_MSB; // may be nullptr
        itercount_array m_nPixels; // FIXME
        BOOL m_bTrans;
        BOOL m_bITrans;
        float **m_nTrans;
        float **m_nDEx;
        float **m_nDEy;
        BOOL m_bNoGlitchDetection;
        int m_nPower;
        int m_nPrevPower;
        int *m_pnExpConsts;
        int m_nMaxOldGlitches;
```

```
        double m_C, m_S;
        struct SIZE_F { double cx; double cy; };
        SIZE_F m_scRatio;
        BOOL m_bNoPostWhenDone;
        BOOL m_bSlopes;
        int m_nSlopePower;
        int m_nSlopeRatio;
        int m_nSlopeAngle;
        double m_nSlopeX, m_nSlopeY;
        double m_nZooms;

        double m_nBailout;
        double m_nBailout2;
        SmoothMethod m_nSmoothMethod;
        ColorMethod m_nColorMethod;
        Differences m_nDifferences;
        int m_nColorOffset;
        BOOL m_bIterChanged;
        int64_t m_nMinI, m_nMaxI;

        double m_xrd, m_xid;

        floatexp *m_dxr, *m_dxi;
        floatexp *m_APr;
        floatexp *m_APi;
        SeriesR2 *m_APs;

        BOOL m_bMirrored;
        int m_nFractalType;

        long double *m_ldxr, *m_ldxi;

        double m_nIterDiv;
        int64_t m_nMaxApproximation;
        int64_t m_nApprox;
        RECT m_rApprox;

        double *m_db_dxr;
        double *m_db_dxi;
        double *m_db_z;

        floatexp m_pixel_step_x, m_pixel_step_y;
        floatexp m_pixel_center_x, m_pixel_center_y;

        half *m_imageHalf; // for EXR export
        BYTE *m_lpBits;
        int m_row;
        BITMAPINFOHEADER *m_bmi;
        bool m_bResized;
        int m_nX, m_nXPrev;
        int m_nY, m_nYPrev;
        int m_nDone;
        int m_nGuessed;
        int64_t m_nRDone;
        bool m_bStop;
        HWND m_hWnd;
        char *m_szPosition;
```

```
335         BOOL m_bReuseRef;
            double m_nScaling;
            int m_nScalingOffset;
            long double m_nScalingL;
            int m_nScalingOffsetL;
340         int m_nStatus;
            int m_nFrameDone;
            BOOL m_bAddReference;
            BOOL m_bNoApproximation;

345         BOOL m_bTexture;
            double m_nImgMerge;
            double m_nImgPower;
            int m_nImgRatio;
            std::string m_szTexture;
350         BYTE *m_lpTextureBits;
            BITMAPINFOHEADER m_bmiBkg;
            int m_rowBkg;

            std::vector< complex<CFixedFloat> > m_Inflections;
355
            NanoMB1_Reference *m_NanoMB1Ref;
            NanoMB2_Reference *m_NanoMB2Ref;

            SeriesType GetApproximationType()
360         {
                    if (m_nFractalType == 0)
                            return SeriesType_Complex;
                    if (m_nFractalType == 1 && m_nPower == 2)
                            return SeriesType_Real;
365                 return SeriesType_None;
            }
            void CalculateApproximation(int nType);
            void DoApproximation(int64_t &antal, const floatexp &D0r, const floatexp↙
                ↳ &D0i, floatexp &TDnr, floatexp &TDni, floatexp &TDDnr, floatexp &↙
                ↳ TDDni);
            void DoApproximation(int64_t &antal, const floatexp &a, const floatexp &↙
                ↳ b, floatexp &x, floatexp &y, floatexp &dxa, floatexp &dxb, ↙
                ↳ floatexp &dya, floatexp &dyb);
370         void DoApproximation(const floatexp &a, const floatexp &b, floatexp &x, ↙
                ↳ floatexp &y); // for internal usage only, assumes isR
            void CalculateReference();
            void CalculateReferenceEXP();
            void CalculateReferenceLDBL();
            void CalculateReferenceNANOMB1();
375         void CalculateReferenceNANOMB2();
            void CalcStart();
            void CreateLists();
            std::string ToZoom(const CDecNumber &z, int &zoom);
            void RenderFractalEXP();
380         void RenderFractalLDBL();
            void RenderFractalNANOMB1();
            void RenderFractalNANOMB2();
#ifdef KF_OPENCL
            void RenderFractalOpenCL();
385         void RenderFractalOpenCLEXP();
#endif
```

```
        int GetArea(itercount_array &Node, int nXStart, int nXStop, int nEqSpan,↙
            ↳ itercount_array &Pixels, int nDone);

        void SetTexture(int nIndex, int x, int y);
        void SetColor(int nIndex, int64_t nIter, double offs = 0, int x = -1, ↙
            ↳ int y = -1, int w = 1, int h = 1);
        void DeleteArrays();

#ifdef KF_OPENCL
        int clid;
  OpenCL *cl;
#endif

        std::vector<std::string> m_undo;
        std::vector<std::string> m_redo;

public:
        BOOL m_bRunning;
        BOOL m_bInhibitColouring;
        bool m_bInteractive;
        int nPos;
        void MandelCalc();
        void MandelCalcEXP();
        void MandelCalcLDBL();
        void MandelCalcNANOMB1();
        void MandelCalcNANOMB2();

        CFraktalSFT();
        ~CFraktalSFT();

        inline void SetWindow(HWND hWnd) { m_hWnd = hWnd; };

        void SetPosition(const CFixedFloat &rstart, const CFixedFloat &rstop, ↙
            ↳ const CFixedFloat &istart, const CFixedFloat &istop, int nX, int ↙
            ↳ nY);
        void SetPosition(const std::string &szR, const std::string &szI, const ↙
            ↳ std::string &szZ);
        std::string ToZoom();
        void SetImageSize(int nx, int ny);
        void RenderFractal(int nX, int nY, int64_t nMaxIter, HWND hWnd, BOOL ↙
            ↳ bNoThread = FALSE, BOOL bResetOldGlitch = TRUE);
        void RenderFractal();
        void CalcStart(int x0, int x1, int y0, int y1);
        HBITMAP GetBitmap();
        HBITMAP ShrinkBitmap(HBITMAP bmSrc,int nNewWidth,int nNewHeight,int mode↙
            ↳ = 1);
        void UpdateBitmap();
        int GetWidth();
        int GetHeight();
        void Stop(BOOL bNoPostWhenDone = FALSE);
        int CountFrames(int nProcent);
        void Zoom(int nXPos, int nYPos, double nZoomSize, int nWidth, int ↙
            ↳ nHeight, BOOL bReuseCenter = FALSE, bool autoRender = true);
        BOOL Center(int &rx, int &ry, BOOL bSkipM = FALSE, BOOL bQuick = FALSE);
        double GetProgress(int *pnGuessed = NULL, int *pnRDone = NULL, int *pnAP↙
            ↳ = NULL, int *pnT = NULL);
        std::string GetPosition();
```

```
435         void GetIterations(int64_t &nMin, int64_t &nMax, int *pnCalculated = ↙
                ↳ NULL, int *pnType = NULL, BOOL bSkipMaxIter = FALSE);
            int64_t GetIterations();
            void SetIterations(int64_t nIterations);
            std::string GetRe();
            std::string GetRe(int nXPos, int nYPos, int width, int height);
440         std::string GetIm();
            std::string GetIm(int nXPos, int nYPos, int width, int height);
            std::string GetZoom();
            void GenerateColors(int nParts, int nSeed = -1);
            void GenerateColors2(int nParts, int nSeed = -1, int nWaves = 9);
445         void AddWave(int nCol, int nPeriod = -1, int nStart = -1);
            void ChangeNumOfColors(int nParts);
            int GetNumOfColors();
            void ApplyColors(int x0, int x1, int y0, int y1);
            void ApplyColors();
450         void ApplyIterationColors();
            void ApplySmoothColors();
            int GetSeed();
            COLOR14 GetKeyColor(int i);
            void SetKeyColor(COLOR14 col, int i);
455         COLOR14 GetColor(int i);
            COLOR14 GetInteriorColor() { return m_cInterior; };
            void SetInteriorColor(const COLOR14 &c) { m_cInterior = c; };
            BOOL OpenFile(const std::string &szFile, BOOL bNoLocation = FALSE);
            BOOL OpenString(const std::string &szText, BOOL bNoLocation = FALSE);
460         BOOL OpenMapB(const std::string &szFile, BOOL bReuseCenter = FALSE, ↙
                ↳ double nZoomSize = 1);
            bool OpenMapEXR(const std::string &szFile);
            std::string ToText();
            BOOL SaveFile(const std::string &szFile, bool overwrite);
            double GetIterDiv();
465         void SetIterDiv(double nIterDiv);
            int SaveJpg(const std::string &szFile, int nQuality, int nWidth = 0, int↙
                ↳  nHeight = 0);
            int64_t GetMaxApproximation();
            int64_t GetIterationOnPoint(int x, int y);
            double GetTransOnPoint(int x, int y);
470         BOOL AddReference(int x, int y, BOOL bEraseAll = FALSE, BOOL ↙
                ↳ bNoGlitchDetection = FALSE, BOOL bResuming = FALSE);
            BOOL HighestIteration(int &rx, int &ry);
            int FindCenterOfGlitch(int &rx, int &ry);
            int GetColorIndex(int x, int y);
            BOOL GetTransition();
475         void SetTransition(BOOL bTransition);
            BOOL GetITransition();
            void SetITransition(BOOL bITransition);

            void SaveMap(const std::string &szFile);
480         void SaveMapB(const std::string &szFile);

            SmoothMethod GetSmoothMethod();
            void SetSmoothMethod(int nSmoothMethod);
            int GetPower();
485         void SetPower(int nPower);
            void SetColorMethod(int nColorMethod);
            ColorMethod GetColorMethod();
```

```
        void SetDifferences(int nDifferences);
        Differences GetDifferences();
        void SetColorOffset(int nColorOffset);
        int GetColorOffset();
        void ErasePixel(int x, int y);

        void StoreLocation();
        void Mirror(int x, int y);

        int GetMWCount();
        void SetMW(BOOL bMW, BOOL bBlend);
        int GetMW(BOOL *pbBlend = NULL);
        BOOL GetMW(int nIndex, int &nPeriod, int &nStart, int &nType);
        BOOL AddMW(int nPeriod, int nStart, int nType);
        BOOL UpdateMW(int nIndex, int nPeriod, int nStart, int nType);
        BOOL DeleteMW(int nIndex);

        int64_t GetMaxExceptCenter();
        void SetFractalType(int nFractalType);
        int GetFractalType();

        int GetExponent();

        double GetRatioX();
        double GetRatioY();
        void SetRatio(double x, double y);

        BOOL GetSlopes(int &nSlopePower, int &nSlopeRatio, int &nSlopeAngle);
        void SetSlopes(BOOL bSlope, int nSlopePower, int nSlopeRatio, int ↙
            ↳ nSlopeAngle);

        BOOL GetTexture(double &nImgMerge,double &nImgPower,int &nImgRatio,std::↙
            ↳ string &szTexture);
        void SetTexture(BOOL bTexture,double nImgMerge,double nImgPower,int ↙
            ↳ nImgRatio,const std::string &szTexture);

        void AddInflectionPont(int x, int y);
        void RemoveInflectionPoint();

#ifdef KF_OPENCL
  int GetOpenCLDeviceIndex();
  void SetOpenCLDeviceIndex(int i);
#endif

        void OutputIterationData(int x, int y, int w, int h, bool bGlitch, ↙
            ↳ int64_t antal, double test1, double test2, const complex<double> &↙
            ↳ de);
        void OutputPixelData(int x, int y, int w, int h, bool bGlitch);
        bool GuessPixel(int x, int y, int x0, int y0, int x1, int y1);
        bool GuessPixel(int x, int y, int w, int h);

        inline bool OpenSettings(const std::string &filename) { return ↙
            ↳ m_Settings.OpenFile(filename); }
        inline bool SaveSettings(const std::string &filename, bool overwrite) ↙
            ↳ const { return m_Settings.SaveFile(filename, overwrite); }

        inline void SetTransformPolar(const polar2 &P)
```

```
        {
                using std::sqrt;
                g_Degree = -P.stretch_angle;
                SetRatio(m_nX, m_nY / (P.stretch_factor * P.stretch_factor));
                // FIXME need to scale image by P.scale * P.stretch_factor
                // FIXME kf does not support rotation after skew...
        }
        inline void SetTransformMatrix(const mat2 &M) { SetTransformPolar(↙
            ↪ polar_decomposition(M)); }
        inline mat2 GetTransformMatrix() const
        {
                double ratio = (((double)m_nY/(double)m_nX)/(360.0/640.0)) * ((↙
                    ↪ double)360 / (double)m_scRatio.cy);
                double k = sqrt(std::abs(ratio));
                return polar_composition(polar2(1.0, -g_Degree, k, g_Degree));
        }

#define DOUBLE(KEY) \
        inline double Get##KEY() const { return m_Settings.Get##KEY(); }; \
        inline void   Set##KEY(double x) { return m_Settings.Set##KEY(x); };
#define INT(KEY) \
        inline int64_t    Get##KEY() const { return m_Settings.Get##KEY(); }; \
        inline void   Set##KEY(int64_t x) { return m_Settings.Set##KEY(x); };
#define BOOL(KEY) \
        inline bool   Get##KEY() const { return m_Settings.Get##KEY(); }; \
        inline void   Set##KEY(bool x) { return m_Settings.Set##KEY(x); };
  DOUBLE(ZoomSize)
  INT(MaxReferences)
  BOOL(GlitchLowTolerance)
  BOOL(ApproxLowTolerance)
  BOOL(AutoApproxTerms)
        inline int64_t    GetApproxTerms() const { return m_Settings.↙
            ↪ GetApproxTerms(); };
             void   SetApproxTerms(int64_t t);
  INT(WindowWidth)
  INT(WindowHeight)
  INT(WindowTop)
  INT(WindowLeft)
  INT(WindowBottom)
  INT(WindowRight)
  INT(ImageWidth)
  INT(ImageHeight)
  DOUBLE(ThreadsPerCore)
  INT(ThreadsReserveCore)
  BOOL(AnimateZoom)
  BOOL(ArbitrarySize)
  BOOL(ReuseReference)
  BOOL(AutoSolveGlitches)
  BOOL(Guessing)
  BOOL(SolveGlitchNear)
  BOOL(NoApprox)
  BOOL(Mirror)
  BOOL(LongDoubleAlways)
  BOOL(FloatExpAlways)
  BOOL(AutoIterations)
  BOOL(ShowGlitches)
  BOOL(NoReuseCenter)
```

```
    INT(IsolatedGlitchNeighbourhood)
    INT(JitterSeed)
    INT(JitterShape)
    DOUBLE(JitterScale)
    BOOL(Derivatives)
    BOOL(ShowCrossHair)
    BOOL(UseNanoMB1)
    BOOL(UseNanoMB2)
    INT(OrderM)
    INT(OrderN)
    BOOL(InteriorChecking)
    DOUBLE(RadiusScale)
    INT(Shrink)
        inline bool   GetHalfColour() const { return m_Settings.GetHalfColour(); };
            void   SetHalfColour(bool b);
        BOOL(SaveOverwrites)
        BOOL(ThreadedReference)
    INT(SIMDVectorSize)
    INT(SIMDChunkSize)
#undef DOUBLE
#undef INT
#undef BOOL

        void GetPixelOffset(const int i, const int j, double &x, double &y) const;
        void GetPixelCoordinates(const int i, const int j, floatexp &x, floatexp &y) const;
        void GetPixelCoordinates(const int i, const int j, floatexp &x, floatexp &y, floatexp &daa, floatexp &dab, floatexp &dba, floatexp &dbb) const;

    void UndoStore() { m_undo.push_back(ToText()); m_redo.clear(); };
    void Undo() { if (!m_undo.empty()) { auto s = m_undo.back(); m_undo.pop_back(); m_redo.push_back(s); OpenString(s); } };
    void Redo() { if (!m_redo.empty()) { auto s = m_redo.back(); m_redo.pop_back(); m_undo.push_back(s); OpenString(s); } };

    // for EXR IO
    itercount_array GetArrayCount() { return m_nPixels; };
    float *GetArrayTrans() { return m_nTrans[0]; };
    float *GetArrayDEx() { return GetDerivatives() ? m_nDEx[0] : nullptr; };
    float *GetArrayDEy() { return GetDerivatives() ? m_nDEy[0] : nullptr; };
    half *GetArrayHalfColour() { return m_imageHalf; };
    size_t GetArrayHalfColourStride() { return m_row; };
        void ReinitializeBitmap();
};

struct TH_PARAMS
{
        int nXStart;
        int nXStop;
        CFraktalSFT *p;
};

extern int g_nAddRefX;
extern int g_nAddRefY;
```

```
extern double g_real;
extern double g_imag;
extern double g_SeedR;
extern double g_SeedI;
extern double g_FactorAR;
extern double g_FactorAI;

extern int g_nLDBL;
extern int g_nEXP;
extern int g_nRefZero;

extern BOOL g_LDBL;

const double pi = 3.141592653589793;

extern void HSVToRGB(double hue, double sat, double bri, COLOR14 &cPos);
extern int MakePrime(int n);

// singleton instance
extern CFraktalSFT g_SFT;

#endif
```

67 fraktal_sft/fraktal_sft.rc

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
```

```
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// English (United States) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)

//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_DIALOG11 DIALOGEX 0, 0, 117, 80
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Skew Animation"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    LTEXT           "Number of frames",IDC_STATIC,7,10,58,8
    EDITTEXT        IDC_EDIT1,69,7,40,14,ES_AUTOHSCROLL
    LTEXT           "To Stretch",IDC_STATIC,31,26,34,8
    EDITTEXT        IDC_EDIT3,69,23,40,14,ES_AUTOHSCROLL
    LTEXT           "To Rotate",IDC_STATIC,32,42,33,8
    EDITTEXT        IDC_EDIT4,69,39,40,14,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,7,59,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,60,59,50,14
END

IDD_DIALOG12 DIALOGEX 65386, 100, 175, 250
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Newton-Raphson Zooming"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    LTEXT           "Click on the fractal to specify the point of interest\nThe zoom size is applied on the area that is searched for minibrot.\nChoose the behaviour when clicking:",IDC_STATIC,7,7,160,33
    CONTROL         "Minibrot",IDC_RADIO1,"Button",BS_AUTORADIOBUTTON,7,42,41,10
    CONTROL         "3/4 (current pattern doubled)",IDC_RADIO2,"Button",BS_AUTORADIOBUTTON,7,52,111,10
    CONTROL         "7/8 (current pattern 4-folded)",IDC_RADIO3,"Button",BS_AUTORADIOBUTTON,7,62,112,10
    CONTROL         "15/16 (current pattern 8-folded)",IDC_RADIO4,"Button",BS_AUTORADIOBUTTON,7,72,120,10
    CONTROL         "Custom factor",IDC_RADIO5,"Button",BS_AUTORADIOBUTTON,7,82,120,10
    LTEXT           "Factor (less than 1.0)",IDC_STATIC,7,94,120,10
    EDITTEXT        IDC_EDIT2,7,106,120,10,ES_AUTOHSCROLL
    LTEXT           "Current pattern zoom",IDC_STATIC,7,118,120,10
    EDITTEXT        IDC_EDIT4,7,130,120,10,ES_AUTOHSCROLL
    PUSHBUTTON      "Capture",IDC_BUTTON2,125,130,40,10

    CONTROL         "Auto Skew (Newton)",IDC_AUTOSKEW,"Button",BS_AUTOCHECKBOX | WS_TABSTOP,7,142,80,12
    PUSHBUTTON      "Auto Skew (Escape)",IDC_AUTOSKEW_BUTTON,7,162,80,12
```

```
    CONTROL         "Use d/dZ",IDC_AUTOSKEW_USEDZ,"Button",BS_AUTOCHECKBOX | ↙
        ↳ WS_TABSTOP,90,162,70,12

    PUSHBUTTON      "Stop",IDCANCEL2,139,61,29,14,NOT WS_VISIBLE
    CONTROL         "Display result",IDC_CHECK9,"Button",BS_AUTOCHECKBOX | ↙
        ↳ WS_TABSTOP,109,182,59,10
    LTEXT           "Status",IDC_STATIC,7,182,23,10
    EDITTEXT        IDC_EDIT1,7,192,181,19,ES_MULTILINE | ES_AUTOHSCROLL | ↙
        ↳ ES_READONLY
    PUSHBUTTON      "Close",IDCANCEL,118,212,50,14
    EDITTEXT        IDC_EDIT3,31,212,67,12,ES_MULTILINE | ES_AUTOHSCROLL | ↙
        ↳ ES_READONLY
    LTEXT           "Period:",IDC_STATIC,7,212,23,10
    CONTROL         "Specify",IDC_CHECK1,"Button",BS_AUTOCHECKBOX | NOT ↙
        ↳ WS_VISIBLE | WS_TABSTOP,82,212,39,10

    CONTROL         "Use ball period detection method",IDC_BALL_PERIOD,"Button",↙
        ↳ BS_AUTOCHECKBOX | WS_TABSTOP,7,230,160,12
END


#ifdef KF_OPENCL
// OpenCL
IDD_DIALOG_OPENCL DIALOGEX 0, 0, 300, 100
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | ↙
    ↳ WS_SYSMENU
CAPTION "OpenCL Settings"
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    LTEXT           "OpenCL Device",IDC_STATIC,7,10,250,8
    COMBOBOX        IDC_COMBO_OPENCL_DEVICE,25,20,250,125,CBS_DROPDOWNLIST | ↙
        ↳ WS_VSCROLL | WS_TABSTOP
    DEFPUSHBUTTON   "OK",IDOK,7,50,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,60,50,50,14
END
#endif


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_DIALOG11, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 110
        TOPMARGIN, 7
        BOTTOMMARGIN, 73
    END

    IDD_DIALOG12, DIALOG
    BEGIN
        LEFTMARGIN, 7
```

```
        RIGHTMARGIN, 168
        TOPMARGIN, 7
        BOTTOMMARGIN, 131
    END
END
#endif    // APSTUDIO_INVOKED

#endif    // English (United States) resources
/////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////
// Swedish (Sweden) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_SVE)
LANGUAGE LANG_SWEDISH, SUBLANG_SWEDISH
#pragma code_page(1252)

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_DIALOG1 DIALOG 0, 0, 272, 28
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_EDIT1,7,7,258,14,ES_AUTOHSCROLL | ES_READONLY
END
```

```
IDD_DIALOG2 DIALOGEX 0, 0, 116, 420
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | ↙
    ↳ WS_SYSMENU
CAPTION "Iterations"
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    EDITTEXT        IDC_EDIT1,7,7,102,14,ES_AUTOHSCROLL | ES_NUMBER
    COMBOBOX        IDC_COMBO2,25,85,84,92,CBS_DROPDOWNLIST | WS_VSCROLL | ↙
        ↳ WS_TABSTOP
    COMBOBOX        IDC_COMBO3,25,109,84,94,CBS_DROPDOWNLIST | WS_VSCROLL | ↙
        ↳ WS_TABSTOP
    COMBOBOX        IDC_COMBO5,25,134,84,125,CBS_DROPDOWNLIST | WS_VSCROLL | ↙
        ↳ WS_TABSTOP
    EDITTEXT        IDC_EDIT3,61,154,40,14,ES_AUTOHSCROLL | ES_NUMBER
    CONTROL         "Glitch low tolerance",IDC_GLITCHLOWTOLERANCE,"Button",↙
        ↳ BS_AUTOCHECKBOX | WS_TABSTOP,7,169,104,10
    CONTROL         "Approximation low tolerance",IDC_CHECK1,"Button",↙
        ↳ BS_AUTOCHECKBOX | WS_TABSTOP,7,181,104,10
    COMBOBOX        IDC_COMBO6,25,213,84,94,CBS_DROPDOWN | WS_VSCROLL | ↙
        ↳ WS_TABSTOP
    LTEXT           "Min",IDC_STATIC,7,30,12,8
    EDITTEXT        IDC_EDIT2,25,28,84,14,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Max",IDC_STATIC,7,45,14,8
    EDITTEXT        IDC_EDIT5,25,43,84,14,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Appr",IDC_STATIC,7,61,16,8
    EDITTEXT        IDC_EDIT7,25,59,84,14,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Calculations",IDC_STATIC,7,229,39,8
    EDITTEXT        IDC_EDIT8,25,227,84,14,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Smooth method",IDC_STATIC,7,76,50,8
    LTEXT           "Power",IDC_STATIC,7,100,21,8
    LTEXT           "Fractal type",IDC_STATIC,7,125,38,8
    LTEXT           "Max references",IDC_STATIC,7,155,50,8
    LTEXT           "Approximation terms",IDC_STATIC,7,204,64,8
    CONTROL         "Auto approximation terms",IDC_CHECK2,"Button",↙
        ↳ BS_AUTOCHECKBOX | WS_TABSTOP,7,193,95,10
    CONTROL         "Real",IDC_CHECK3,"Button",BS_AUTOCHECKBOX | WS_TABSTOP↙
        ↳ ,7,260,31,10
    CONTROL         "Imag",IDC_CHECK5,"Button",BS_AUTOCHECKBOX | WS_TABSTOP↙
        ↳ ,38,260,31,10
    LTEXT           "Seed",IDC_STATIC,7,271,18,8
    EDITTEXT        IDC_EDIT4,17,281,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT9,69,281,40,14,ES_AUTOHSCROLL
    LTEXT           "I:",IDC_STATIC,59,284,8,8
    LTEXT           "R:",IDC_STATIC,7,284,8,8
    LTEXT           "Factor a (for TheRedShifter)",IDC_STATIC,7,296,89,8
    EDITTEXT        IDC_EDIT28,17,306,40,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_EDIT10,69,306,40,14,ES_AUTOHSCROLL
    LTEXT           "I:",IDC_STATIC,59,309,8,8
    LTEXT           "R:",IDC_STATIC,7,309,8,8
    LTEXT           "Jitter Seed",IDC_STATIC,7,327,64,8
    EDITTEXT        IDC_JITTERSEED,69,325,40,14,ES_AUTOHSCROLL
    LTEXT           "Jitter Scale",IDC_STATIC,7,342,64,8
    EDITTEXT        IDC_JITTERSCALE,69,340,40,14,ES_AUTOHSCROLL
    CONTROL         "Gaussian Jitter",IDC_JITTERSHAPE,"Button",BS_AUTOCHECKBOX |↙
        ↳ WS_TABSTOP,7,357,100,10
    CONTROL         "Derivatives",IDC_DERIVATIVES,"Button",BS_AUTOCHECKBOX | ↙
        ↳ WS_TABSTOP,7,374,100,10
```

```
    DEFPUSHBUTTON   "OK",IDOK,7,397,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,59,397,50,14

END

IDD_DIALOG3 DIALOGEX 0, 0, 397, 273
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Set Colors"
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    GROUPBOX        "",IDC_STATIC,60,44,65,56
    LTEXT           "Number of key colors",IDC_STATIC,7,7,39,17
    EDITTEXT        IDC_EDIT1,7,26,38,14,ES_AUTOHSCROLL | ES_NUMBER
    CONTROL         "Spin1","IDC_SPIN1","msctls_updown32",UDS_SETBUDDYINT | UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,39,35,11,14
    LTEXT           "Divide iteration",IDC_STATIC,46,7,39,17
    EDITTEXT        IDC_EDIT3,46,26,38,14,ES_AUTOHSCROLL
    CONTROL         "Spin2","IDC_SPIN2","msctls_updown32",UDS_SETBUDDYINT | UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,91,30,11,14
    LTEXT           "Color\noffset",IDC_STATIC,85,7,39,17
    EDITTEXT        IDC_EDIT12,85,26,38,14,ES_AUTOHSCROLL | ES_NUMBER
    CONTROL         "Spin2",IDC_SPIN5,"msctls_updown32",UDS_SETBUDDYINT | UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,131,30,11,14
    EDITTEXT        IDC_EDIT2,134,13,46,14,ES_AUTOHSCROLL
    PUSHBUTTON      "&Generate",IDC_BUTTON2,143,27,36,10
    PUSHBUTTON      "&Seed",IDC_BUTTON13,143,38,36,10
    PUSHBUTTON      "&Generate",IDC_BUTTON12,352,137,34,12
    PUSHBUTTON      "Seed",IDC_BUTTON14,352,152,34,12
    PUSHBUTTON      "Save palette",IDC_BUTTON1,7,116,55,10
    PUSHBUTTON      "Open palette",IDC_BUTTON5,7,127,55,10
    PUSHBUTTON      "Expand Double",IDC_BUTTON6,65,105,55,10
    PUSHBUTTON      "Expand All",IDC_BUTTON7,65,116,55,10
    PUSHBUTTON      "Double",IDC_BUTTON10,65,127,55,10
    CONTROL         "Show index (move cursor over the fractal and the color will be selected in the list)",IDC_CHECK1,
                    "Button",BS_AUTOCHECKBOX | BS_MULTILINE | WS_TABSTOP,7,138,114,23
    CONTROL         "Smooth color transition",IDC_CHECK2,"Button",BS_AUTOCHECKBOX | WS_TABSTOP,7,161,87,10
    COMBOBOX        IDC_COMBO1,7,182,116,71,CBS_DROPDOWNLIST | WS_VSCROLL | WS_TABSTOP
    COMBOBOX        IDC_DIFFERENCES,7,202,116,71,CBS_DROPDOWNLIST | WS_VSCROLL | WS_TABSTOP
    GROUPBOX        "Merge Colors",IDC_STATIC,129,57,56,113
    EDITTEXT        IDC_EDIT9,137,105,40,14,ES_AUTOHSCROLL | ES_NUMBER
    CONTROL         "Spin2",IDC_SPIN3,"msctls_updown32",UDS_SETBUDDYINT | UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,177,110,11,14
    LTEXT           "Rate percent",IDC_STATIC,137,123,42,8
    EDITTEXT        IDC_EDIT10,137,133,40,14,ES_AUTOHSCROLL | ES_NUMBER
    CONTROL         "Spin2",IDC_SPIN4,"msctls_updown32",UDS_SETBUDDYINT | UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,177,136,11,14
```

```
    PUSHBUTTON      "Color",IDC_BUTTON8,135,150,47,15
    LISTBOX         IDC_LIST1,189,26,68,145,LBS_SORT | LBS_OWNERDRAWFIXED | ↙
        ↳ LBS_HASSTRINGS | LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
    CONTROL         "Enable",IDC_AUTOCOLOUR,"Button",BS_AUTOCHECKBOX | ↙
        ↳ WS_TABSTOP,200,258,50,10
    DEFPUSHBUTTON   "Apply",IDOK,255,255,43,14
    PUSHBUTTON      "Close",IDCANCEL,347,255,43,14
    LTEXT           "Double click to edit\nRight click for options",IDC_STATIC↙
        ↳ ,188,7,69,18
    LTEXT           "",IDC_SHOW,7,222,383,32,0,WS_EX_STATICEDGE
    GROUPBOX        "Random",IDC_STATIC,129,2,56,49
    GROUPBOX        "Palette waves",IDC_STATIC,262,114,128,78
    CONTROL         "Inverse Smooth color transition",IDC_CHECK3,"Button",↙
        ↳ BS_AUTOCHECKBOX | WS_TABSTOP,7,171,113,10
    LISTBOX         IDC_LIST6,263,67,82,43,LBS_USETABSTOPS | ↙
        ↳ LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
    GROUPBOX        "",IDC_FRAME6,260,8,130,105
    LTEXT           "Period",Colors2,273,37,21,8
    EDITTEXT        IDC_EDIT23,300,32,32,14,ES_AUTOHSCROLL
    PUSHBUTTON      "Add",IDC_BUTTON26,347,68,30,12
    PUSHBUTTON      "Update",IDC_BUTTON27,347,81,30,12
    PUSHBUTTON      "Remove",IDC_BUTTON28,347,94,30,12
    LTEXT           "Type",IDC_STATIC,271,21,17,8
    COMBOBOX        IDC_COMBO4,300,18,48,102,CBS_DROPDOWNLIST | WS_VSCROLL | ↙
        ↳ WS_TABSTOP
    CONTROL         "Infinite waves",IDC_CHECK6,"Button",BS_AUTOCHECKBOX | ↙
        ↳ WS_TABSTOP,265,7,59,10
    EDITTEXT        IDC_EDIT24,268,37,14,14,ES_AUTOHSCROLL | NOT WS_VISIBLE
    PUSHBUTTON      "Make prime",IDC_BUTTON29,334,32,43,14
    LTEXT           "Start",Colors3,273,53,16,8,NOT WS_VISIBLE
    EDITTEXT        IDC_EDIT25,300,48,32,14,ES_AUTOHSCROLL | ES_NUMBER | NOT ↙
        ↳ WS_VISIBLE
    LTEXT           "Red",IDC_STATIC,273,132,14,8
    LTEXT           "Green",IDC_STATIC,267,147,20,8
    LTEXT           "Blue",IDC_STATIC,272,161,15,8
    LTEXT           "Period",IDC_STATIC,291,122,21,8
    LTEXT           "Start",IDC_STATIC,325,122,16,8
    EDITTEXT        IDC_EDIT11,290,130,22,14,ES_AUTOHSCROLL | ES_NUMBER
    EDITTEXT        IDC_EDIT13,324,130,22,14,ES_AUTOHSCROLL | ES_NUMBER
    PUSHBUTTON      "P",IDC_BUTTON16,313,130,10,14
    EDITTEXT        IDC_EDIT14,290,144,22,14,ES_AUTOHSCROLL | ES_NUMBER
    EDITTEXT        IDC_EDIT15,324,144,22,14,ES_AUTOHSCROLL | ES_NUMBER
    PUSHBUTTON      "P",IDC_BUTTON19,313,144,10,14
    EDITTEXT        IDC_EDIT16,290,159,22,14,ES_AUTOHSCROLL | ES_NUMBER
    EDITTEXT        IDC_EDIT17,324,159,22,14,ES_AUTOHSCROLL | ES_NUMBER
    PUSHBUTTON      "P",IDC_BUTTON20,313,159,10,14
    LTEXT           "B/W",IDC_STATIC,271,177,16,8
    EDITTEXT        IDC_EDIT18,290,174,22,14,ES_AUTOHSCROLL | ES_NUMBER
    EDITTEXT        IDC_EDIT19,324,174,22,14,ES_AUTOHSCROLL | ES_NUMBER
    PUSHBUTTON      "P",IDC_BUTTON21,313,174,10,14
    CONTROL         "Step",IDC_RADIO4,"Button",BS_AUTORADIOBUTTON,137,72,31,10
    CONTROL         "Wave period",IDC_RADIO5,"Button",BS_AUTORADIOBUTTON | ↙
        ↳ BS_MULTILINE,137,85,36,17
    PUSHBUTTON      "More contrast",IDC_BUTTON17,7,51,50,10
    PUSHBUTTON      "Less contrast",IDC_BUTTON18,7,61,50,10
    PUSHBUTTON      "From image",IDC_BUTTON22,7,105,55,10
    PUSHBUTTON      "Cancel",IDCLOSE,301,255,43,14
```

```
320         CONTROL         "Blend",IDC_CHECK7,"Button",BS_AUTOCHECKBOX | WS_TABSTOP
                ↪ ,356,7,34,10
            CONTROL         "Show slopes",IDC_CHECK4,"Button",BS_AUTOCHECKBOX |
                ↪ WS_TABSTOP,62,43,56,10
            EDITTEXT        IDC_EDIT20,67,53,38,14,ES_AUTOHSCROLL | ES_NUMBER
            CONTROL         "Spin1",IDC_SPIN6,"msctls_updown32",UDS_SETBUDDYINT |
                ↪ UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS
                ↪ ,99,66,11,14
            EDITTEXT        IDC_EDIT21,67,68,38,14,ES_AUTOHSCROLL | ES_NUMBER
325         CONTROL         "Spin1",IDC_SPIN7,"msctls_updown32",UDS_SETBUDDYINT |
                ↪ UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS
                ↪ ,99,77,11,14
            LTEXT           "^",IDC_STATIC,107,57,8,8
            LTEXT           "%",IDC_STATIC,107,72,8,8
            EDITTEXT        IDC_EDIT22,67,83,38,14,ES_AUTOHSCROLL | ES_NUMBER
            CONTROL         "Spin1",IDC_SPIN8,"msctls_updown32",UDS_SETBUDDYINT |
                ↪ UDS_ALIGNRIGHT | UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS
                ↪ ,99,90,11,14
330         LTEXT           "°",IDC_STATIC,108,86,8,8
            GROUPBOX        "",IDC_STATIC,130,172,127,44
            CONTROL         "Texture",IDC_CHECK8,"Button",BS_AUTOCHECKBOX | WS_TABSTOP
                ↪ ,133,172,40,10
            EDITTEXT        IDC_EDIT26,135,183,30,14,ES_AUTOHSCROLL | ES_NUMBER
            EDITTEXT        IDC_EDIT27,178,183,30,14,ES_AUTOHSCROLL | ES_NUMBER
335         LTEXT           "^",IDC_STATIC,168,188,8,8
            LTEXT           "%",IDC_STATIC,212,186,8,8
            PUSHBUTTON      "...",IDC_BUTTON23,237,199,14,14
            EDITTEXT        IDC_EDIT29,135,199,102,14,ES_AUTOHSCROLL
            PUSHBUTTON      "Transfer to palette",IDC_BUTTON30,300,50,77,14
340         PUSHBUTTON      "Interior",IDC_INTERIORCOLOR,265,199,50,15
    END

    IDD_DIALOG4 DIALOGEX 0, 0, 84, 76
    STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION
345 EXSTYLE WS_EX_TOOLWINDOW
    CAPTION "Crosshair"
    FONT 8, "MS Sans Serif", 0, 0, 0x1
    BEGIN
        LTEXT           "",IDC_PREV,0,0,88,62,0,WS_EX_CLIENTEDGE
350 END

    IDD_DIALOG5 DIALOG 0, 0, 186, 140
    STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION |
        ↪ WS_SYSMENU
    CAPTION "Location"
355 FONT 8, "MS Sans Serif"
    BEGIN
        LTEXT           "Re: ",IDC_STATIC,7,9,14,8
        EDITTEXT        IDC_EDIT1,33,7,146,14,ES_AUTOHSCROLL
        LTEXT           "Im:",IDC_STATIC,7,25,10,8
360     EDITTEXT        IDC_EDIT3,33,22,146,14,ES_AUTOHSCROLL
        LTEXT           "Zoom:",IDC_STATIC,7,43,21,8
        EDITTEXT        IDC_EDIT4,33,40,146,14,ES_AUTOHSCROLL
        LTEXT           "Min",IDC_STATIC,15,61,12,8
        EDITTEXT        IDC_EDIT2,33,59,84,14,ES_AUTOHSCROLL | ES_READONLY
365     LTEXT           "Max",IDC_STATIC,15,76,14,8
        EDITTEXT        IDC_EDIT5,33,74,84,14,ES_AUTOHSCROLL | ES_READONLY
```

```
    LTEXT           "Period:",IDC_STATIC,7,91,21,8
    EDITTEXT        IDC_LOCATION_PERIOD,33,89,146,14,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "Set",IDOK,75,121,50,14
    PUSHBUTTON      "Close",IDCANCEL,129,121,50,14
END

IDD_DIALOG6 DIALOG 0, 0, 186, 133
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Create zoom sequence"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "Store in folder",IDC_STATIC,7,7,44,8
    EDITTEXT        IDC_EDIT1,7,18,155,14,ES_AUTOHSCROLL
    PUSHBUTTON      "...",IDC_BUTTON1,163,18,16,14
    LTEXT           "Percent zoom-out",IDC_STATIC,7,40,56,8
    EDITTEXT        IDC_EDIT2,67,39,50,14,ES_AUTOHSCROLL | ES_NUMBER
    LTEXT           "Number of frames",IDC_STATIC,7,56,56,8
    EDITTEXT        IDC_EDIT5,67,55,50,14,ES_AUTOHSCROLL | ES_NUMBER
    PUSHBUTTON      "Calculate to zoom 1",IDC_BUTTON3,121,43,58,21,BS_MULTILINE
    LTEXT           "Width",IDC_STATIC,7,79,20,8
    EDITTEXT        IDC_EDIT3,31,77,30,14,ES_AUTOHSCROLL | ES_NUMBER
    PUSHBUTTON      "Open incomplete",IDC_BUTTON4,7,112,62,14
    DEFPUSHBUTTON   "Start",IDOK,77,112,50,14
    PUSHBUTTON      "Close",IDCANCEL,129,112,50,14
    EDITTEXT        IDC_EDIT4,91,77,30,14,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Height",IDC_STATIC,66,79,22,8
    EDITTEXT        IDC_EDIT6,7,95,172,14,ES_AUTOHSCROLL | ES_READONLY
    CONTROL         "Alt. method",IDC_CHECK1,"Button",BS_AUTOCHECKBOX | WS_TABSTOP,129,78,51,10
END

IDD_DIALOG7 DIALOG 0, 0, 117, 79
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Jpeg properties"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "Width",IDC_STATIC,7,9,20,8
    EDITTEXT        IDC_EDIT1,34,7,48,14,ES_AUTOHSCROLL
    LTEXT           "Height",IDC_STATIC,7,25,22,8
    EDITTEXT        IDC_EDIT3,34,23,48,14,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Quality",IDC_QUALITY_LABEL,7,41,22,8
    EDITTEXT        IDC_EDIT4,34,39,48,14,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,7,58,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,60,58,50,14
END

IDD_DIALOG8 DIALOGEX 0, 0, 153, 161
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Examine zoom sequence"
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    CONTROL         "Add references",IDC_RADIO1,"Button",BS_AUTORADIOBUTTON,7,7,64,10
    CONTROL         "Set main reference",IDC_RADIO2,"Button",BS_AUTORADIOBUTTON,7,28,75,10
```

```
    CONTROL         "Eraser",IDC_RADIO3,"Button",BS_AUTORADIOBUTTON,7,41,36,10
    CONTROL         "Use near pixel method",IDC_CHECK1,"Button",BS_AUTOCHECKBOX ↙
        ↳ | NOT WS_VISIBLE | WS_TABSTOP,13,17,86,10
    PUSHBUTTON      "Save and &Previous",IDC_BUTTON1,8,64,68,14
    DEFPUSHBUTTON   "Save and &Next",IDOK,78,64,68,14
    PUSHBUTTON      "Save and Previous - reuse &center",IDC_BUTTON5,7,79,116,14
    PUSHBUTTON      "Cancel changes",IDC_BUTTON2,7,140,58,14
    EDITTEXT        IDC_EDIT1,112,17,34,14,ES_AUTOHSCROLL
    PUSHBUTTON      "Set",IDC_BUTTON3,112,33,34,11
    PUSHBUTTON      "Close",IDCANCEL,96,140,50,14
    LTEXT           "Frame",IDC_STATIC,112,7,20,8
    EDITTEXT        IDC_EDIT2,112,46,34,14,ES_AUTOHSCROLL | ES_READONLY
    EDITTEXT        IDC_EDIT3,7,51,93,12,ES_AUTOHSCROLL | ES_READONLY | NOT ↙
        ↳ WS_BORDER
    PUSHBUTTON      "Auto solve glitches",IDC_BUTTON4,7,96,73,14
    EDITTEXT        IDC_EDIT4,82,96,64,20,ES_MULTILINE | ES_AUTOVSCROLL | ↙
        ↳ ES_READONLY
    LTEXT           "Max refs",IDC_STATIC,8,120,50,14
    EDITTEXT        IDC_EDITMAXREFS,64,120,50,14,ES_AUTOHSCROLL
END

IDD_DIALOG9 DIALOGEX 0, 0, 186, 46
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | ↙
    ↳ WS_SYSMENU
CAPTION "Find Minibrot"
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    EDITTEXT        IDC_EDIT1,68,7,57,14,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,40,25,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,96,25,50,14
    LTEXT           "Stop at exponent",IDC_LABEL,7,9,55,8
END

IDD_DIALOG10 DIALOGEX 0, 0, 116, 79
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Skew location"
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    LTEXT           "Stretch",IDC_STATIC,7,10,24,8
    EDITTEXT        IDC_EDIT1,35,7,60,14,ES_AUTOHSCROLL
    CONTROL         "Spin1",IDC_SPIN1,"msctls_updown32",UDS_ALIGNRIGHT | ↙
        ↳ UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,81,7,11,14
    RTEXT           "Rotate",IDC_STATIC,7,28,22,8
    EDITTEXT        IDC_EDIT2,35,25,60,14,ES_AUTOHSCROLL
    CONTROL         "Spin1",IDC_SPIN2,"msctls_updown32",UDS_ALIGNRIGHT | ↙
        ↳ UDS_AUTOBUDDY | UDS_ARROWKEYS | UDS_NOTHOUSANDS,85,25,11,14
    DEFPUSHBUTTON   "OK",IDOK,7,58,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,59,58,50,14
    LTEXT           "%",IDC_STATIC,101,10,8,8
    LTEXT           "°",IDC_STATIC,101,28,8,8
    PUSHBUTTON      "Reset",IDC_BUTTON1,59,42,50,14
    CONTROL         "Help lines",IDC_CHECK1,"Button",BS_AUTOCHECKBOX | ↙
        ↳ BS_PUSHLIKE | WS_TABSTOP,7,42,50,14
END

IDD_STOREZOOM DIALOGEX 0, 0, 116, 140
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | ↙
```

```
    ↪ WS_SYSMENU
CAPTION "Store Zoom Sequence"
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    CONTROL         "Store KFB",IDC_STOREZOOM_KFB,"Button",BS_AUTOCHECKBOX | ↙
        ↪ WS_TABSTOP,7,6,50,14
    CONTROL         "Store PNG",IDC_STOREZOOM_PNG,"Button",BS_AUTOCHECKBOX | ↙
        ↪ WS_TABSTOP,7,22,50,14
    CONTROL         "Store JPG",IDC_STOREZOOM_JPG,"Button",BS_AUTOCHECKBOX | ↙
        ↪ WS_TABSTOP,7,38,50,14
    CONTROL         "Store TIF",IDC_STOREZOOM_TIF,"Button",BS_AUTOCHECKBOX | ↙
        ↪ WS_TABSTOP,7,54,50,14
    CONTROL         "Store EXR",IDC_STOREZOOM_EXR,"Button",BS_AUTOCHECKBOX | ↙
        ↪ WS_TABSTOP,7,70,50,14
    CONTROL         "Stop after n images",IDC_STOREZOOM_COUNTAUTO,"Button",↙
        ↪ BS_AUTOCHECKBOX | WS_TABSTOP,7,86,100,14
    EDITTEXT        IDC_STOREZOOM_COUNT,7,102,30,14,ES_AUTOHSCROLL | ES_NUMBER
    DEFPUSHBUTTON   "OK",IDOK,7,118,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,59,118,50,14
END

/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_DIALOG1, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 265
        TOPMARGIN, 7
        BOTTOMMARGIN, 21
    END

    IDD_DIALOG2, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 109
        VERTGUIDE, 38
        TOPMARGIN, 7
        BOTTOMMARGIN, 325
    END

    IDD_DIALOG3, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 390
        TOPMARGIN, 7
        BOTTOMMARGIN, 269
    END

    IDD_DIALOG4, DIALOG
    BEGIN
        RIGHTMARGIN, 80
```

```
        BOTTOMMARGIN, 74
    END

    IDD_DIALOG5, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 110
    END

    IDD_DIALOG6, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 126
    END

    IDD_DIALOG7, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 110
        TOPMARGIN, 7
        BOTTOMMARGIN, 72
    END

    IDD_DIALOG8, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 146
        TOPMARGIN, 7
        BOTTOMMARGIN, 134
    END

    IDD_DIALOG9, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 39
    END

    IDD_DIALOG10, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 109
        TOPMARGIN, 7
        BOTTOMMARGIN, 72
    END
END
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// Menu
```

```
//

IDR_MENU1 MENUEX
BEGIN
    POPUP "&File",                                  65535,MFT_STRING,MFS_ENABLED
    BEGIN
        MENUITEM "&Open\tCtrl+O",                   ID_FILE_OPEN_,MFT_STRING,
            ↳ MFS_ENABLED
        MENUITEM "Open map...",                     ID_FILE_OPENMAP,MFT_STRING,
            ↳ MFS_ENABLED
        MENUITEM "&Save\tCtrl+S",                   ID_FILE_SAVE_,MFT_STRING,
            ↳ MFS_ENABLED
        POPUP "Save &As",65535,MFT_STRING,MFS_ENABLED
        BEGIN
            MENUITEM "KFR\tCtrl+A",                 ID_FILE_SAVEAS_,MFT_STRING,
                ↳ MFS_ENABLED
            MENUITEM "&PNG...\tCtrl+P",             ID_FILE_SAVEASPNG,MFT_STRING,
                ↳ MFS_ENABLED
            MENUITEM "&JPEG...\tCtrl+J",            ID_FILE_SAVEASJPEG,MFT_STRING,
                ↳ MFS_ENABLED
            MENUITEM "TIFF...",                     ID_FILE_SAVEASTIF,MFT_STRING,
                ↳ MFS_ENABLED
            MENUITEM "EXR...",                      ID_FILE_SAVEASEXR,MFT_STRING,
                ↳ MFS_ENABLED
            MENUITEM "KFB...",                      ID_FILE_SAVEMAP,MFT_STRING,
                ↳ MFS_ENABLED
        END
        MENUITEM "Store Zoom-out images...\tCtrl+Q", ID_FILE_STOREZOOMOUTIMAGES,
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Examine Zoom sequence",           ID_FILE_EXAMINEZOOMSEQUENCE,
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Resume Zoom sequence",            ID_FILE_RESUMEZOOMSEQUENCE,
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Open Settings...",                ID_FILE_OPENSETTINGS,MFT_STRING,
            ↳ MFS_ENABLED
        MENUITEM "Save Settings...",                ID_FILE_SAVESETTINGS,MFT_STRING,
            ↳ MFS_ENABLED
        MENUITEM "Check for update...",             ID_FILE_CHECKFORUPDATE,
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "&Exit",                           ID_ACTIONS_EXIT,MFT_STRING,
            ↳ MFS_ENABLED
    END

    POPUP "Fra&ktal",65535,MFT_STRING,MFS_ENABLED
    BEGIN
        POPUP "Presets",65535,MFT_STRING,MFS_ENABLED
        BEGIN
            MENUITEM "Fast",                                ID_SPECIAL_PRESET_FAST,
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "Best",                                ID_SPECIAL_PRESET_BEST,
                ↳ MFT_STRING,MFS_ENABLED
        END
        MENUITEM "&Location...\tCtrl+L",            ID_ACTIONS_POSITION,MFT_STRING,
            ↳ MFS_ENABLED
        MENUITEM "&Iterations...\tCtrl+I",          ID_ACTIONS_ITERATIONS,MFT_STRING
            ↳ ,MFS_ENABLED
        MENUITEM "Set &Colors...\tCtrl+C",          ID_ACTIONS_SETCOLORS,MFT_STRING,
```

```
            ↳ MFS_ENABLED
        MENUITEM "Refresh\tF5",                 ID_ACTIONS_REFRESH,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Cancel rendering\tESC",       ID_ACTIONS_CANCELRENDERING,↙
            ↳ MFT_STRING,MFS_ENABLED
615     MENUITEM "Rese&t\tCtrl+N",              ID_ACTIONS_RESET,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Undo\tCtrl+Z",                ID_ACTIONS_UNDO,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Redo\tCtrl+Y",                ID_ACTIONS_REDO,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Copy\tCtrl+X",                ID_ACTIONS_COPY,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Paste\tCtrl+V",               ID_ACTIONS_PASTE,MFT_STRING,↙
            ↳ MFS_ENABLED
620 END

    POPUP "&View",65535,MFT_STRING,MFS_ENABLED
    BEGIN
        POPUP "&Zoom size",                     65535,MFT_STRING,MFS_ENABLED
625     BEGIN
            MENUITEM "1",                           ID_ACTIONS_ZOOMSIZE_1,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "&2\tCtrl+2",                  ID_ACTIONS_ZOOMSIZE_2,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "&4\tCtrl+4",                  ID_ACTIONS_ZOOMSIZE_4,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "&8\tCtrl+8",                  ID_ACTIONS_ZOOMSIZE_8,↙
                ↳ MFT_STRING,MFS_ENABLED
630         MENUITEM "&16\tCtrl+1",                 ID_ACTIONS_ZOOMSIZE_16,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "&32\tCtrl+3",                 ID_ACTIONS_ZOOMSIZE_32,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "&64\tCtrl+6",                 ID_ACTIONS_ZOOMSIZE_64,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "128",                         ID_ACTIONS_ZOOMSIZE_128,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "Custom...",                   ID_ZOOMSIZE_CUSTOM,↙
                ↳ MFT_STRING,MFS_ENABLED
635     END
        MENUITEM "Animate zoom",                ID_ACTIONS_ANIMATEZOOM,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Set &window size\tCtrl+W",    ID_ACTIONS_SETWINDOWSIZE,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Set image &size\tCtrl+Shift+W",ID_ACTIONS_SETIMAGESIZE,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Arbitrary size",              ID_SPECIAL_ARBITRARYSIZE,↙
            ↳ MFT_STRING,MFS_ENABLED
640     MENUITEM "Rotate\tCtrl+T",              ID_ACTIONS_ROTATE,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Reset Rotation\tCtrl+Shift+T",ID_ACTIONS_RESETROTATION,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "S&kew\tCtrl+K",               ID_ACTIONS_SKEW,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Set ratio",                   ↙
            ↳ ID_ACTIONS_SPECIAL_SPECIAL_SETRATIO,MFT_STRING,MFS_ENABLED
        MENUITEM "Reset ratio\tCtrl+Shift+K",   ↙
```

```
            ↳ ID_ACTIONS_SPECIAL_SPECIAL_RESETRATIO,MFT_STRING,MFS_ENABLED
        MENUITEM "Skew animation",                  ID_SPECIAL_SKEWANIMATION,↙
            ↳ MFT_STRING,MFS_ENABLED
        POPUP "Shrink quality",                     65535,MFT_STRING,MFS_ENABLED
        BEGIN
            MENUITEM "Fast",                            ID_IMAGE_SHRINK_FAST,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "Default",                         ID_IMAGE_SHRINK_DEFAULT,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "Best",                            ID_IMAGE_SHRINK_BEST,↙
                ↳ MFT_STRING,MFS_ENABLED
        END
    END

    POPUP "&Navigation",65535,MFT_STRING,MFS_ENABLED
    BEGIN
        MENUITEM "Newton-Raphson zooming\tCtrl+D", ID_SPECIAL_NEWTON,MFT_STRING,↙
            ↳ MFS_ENABLED
        MENUITEM "Find &Minibrot\tCtrl+M",      ID_ACTIONS_FINDMINIBROT,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Non-exact find Minibrot",     ID_SPECIAL_NONEXACTFINDMINIBROT,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Center c&ursor\tCtrl+U",      ID_ACTIONS_CENTERCURSOR,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Find center of glitch (Colors)\tCtrl+F", ↙
            ↳ ID_ACTIONS_FINDCENTEROFGLITCH,MFT_STRING,MFS_ENABLED
        MENUITEM "Find highest iteration",      ID_ACTIONS_FINDHIGHESTITERATION,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "S&how Inflection\tCtrl+H",    ID_ACTIONS_SHOWINFLECTION,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Show crosshair window",       ID_SPECIAL_SHOWCROSSHAIR,↙
            ↳ MFT_STRING,MFS_ENABLED
    END

    POPUP "&Advanced",                          65535,MFT_STRING,MFS_ENABLED
    BEGIN
        MENUITEM "Auto solve glitches",         ID_ACTIONS_AUTOSOLVEGLITCHES,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "Solve glitch with near pixels method", ↙
            ↳ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD,MFT_STRING,↙
            ↳ MFS_ENABLED
        POPUP "Ignore isolated small glitches", 65535,MFT_STRING,MFS_ENABLED
        BEGIN
            MENUITEM "Never",                           ID_ACTIONS_ISOLATED_0,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "When alone in 8-neighbourhood",   ID_ACTIONS_ISOLATED_8,↙
                ↳ MFT_STRING,MFS_ENABLED
            MENUITEM "When alone in 4-neighbourhood",   ID_ACTIONS_ISOLATED_4,↙
                ↳ MFT_STRING,MFS_ENABLED
        END
        MENUITEM "Set main reference",          ↙
            ↳ ID_ACTIONS_SPECIAL_SETMAINREFERENCE,MFT_STRING,MFS_ENABLED
        MENUITEM "Add Reference (Colors)\tCtrl+R", ID_ACTIONS_ADDREFERENCE,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "R&euse reference\tCtrl+E",    ID_ACTIONS_REUSEREFERENCE,↙
            ↳ MFT_STRING,MFS_ENABLED
        MENUITEM "NO reuse center",             ID_ACTIONS_SPECIAL_NOREUSECENTER↙
```

```
                ↪ ,MFT_STRING,MFS_ENABLED
680             MENUITEM "Show Iterations",                 ID_ACTIONSSHOWITERATIONS, ↙
                ↪ MFT_STRING,MFS_ENABLED
                MENUITEM "Show smooth transition colors", ↙
                ↪ ID_ACTIONS_SPECIAL_SHOWSMOOTHTRANSITIONCOLORS,MFT_STRING, ↙
                ↪ MFS_ENABLED
                MENUITEM "Show glitches",                   ID_SPECIAL_SHOWGLITCHES, ↙
                ↪ MFT_STRING,MFS_ENABLED
                MENUITEM "Mirror",                          ↙
                ↪ ID_ACTIONS_SPECIAL_SPECIAL_MIRROR1,MFT_STRING,MFS_ENABLED
                MENUITEM "NO series approximation",         ↙
                ↪ ID_ACTIONS_SPECIAL_NOAPPROXIMATION,MFT_STRING,MFS_ENABLED
685             MENUITEM "Use long double always",          ↙
                ↪ ID_ACTIONS_SPECIAL_USELONGDOUBLEFROMSTART,MFT_STRING,MFS_ENABLED
                MENUITEM "Use floatexp always",             ↙
                ↪ ID_ACTIONS_SPECIAL_USEFLOATEXPALWAYS,MFT_STRING,MFS_ENABLED
                MENUITEM "Multi-threaded reference calcs",ID_SPECIAL_THREADED_REFERENCE, ↙
                ↪ MFT_STRING,MFS_ENABLED
                MENUITEM "Use NanoMB1 (experimental)",  ID_ACTIONS_SPECIAL_USENANOMB1, ↙
                ↪ MFT_STRING,MFS_ENABLED
                MENUITEM "Use NanoMB2 (experimental)",  ID_ACTIONS_SPECIAL_USENANOMB2, ↙
                ↪ MFT_STRING,MFS_ENABLED
690             MENUITEM "Interior checking (NanoMB only)", ↙
                ↪ ID_ACTIONS_SPECIAL_INTERIORCHECKING,MFT_STRING,MFS_ENABLED
                MENUITEM "Use auto iterations",             ID_ACTIONS_SPECIAL_AUTOITERATION↙
                ↪ ,MFT_STRING,MFS_ENABLED
                MENUITEM "Use guessing",                    ID_ACTIONS_SPECIAL_GUESSING, ↙
                ↪ MFT_STRING,MFS_ENABLED
                MENUITEM "Half-float image buffer",         ID_SPECIAL_HALFCOLOUR,MFT_STRING↙
                ↪ ,MFS_ENABLED
                MENUITEM "'Save' overwrites existing file",ID_SPECIAL_SAVE_OVERWRITES, ↙
                ↪ MFT_STRING,MFS_ENABLED
695             POPUP "SIMD vector size",                   65535,MFT_STRING,MFS_ENABLED
                BEGIN
                    MENUITEM "1",  ID_SIMD_VECTOR_SIZE_1,MFT_STRING,MFS_ENABLED
    #if KF_SIMD >= 1
                    MENUITEM "2",  ID_SIMD_VECTOR_SIZE_2,MFT_STRING,MFS_ENABLED
700 #endif
    #if KF_SIMD >= 2
                    MENUITEM "4",  ID_SIMD_VECTOR_SIZE_4,MFT_STRING,MFS_ENABLED
    #endif
    #if KF_SIMD >= 3
705                 MENUITEM "8",  ID_SIMD_VECTOR_SIZE_8,MFT_STRING,MFS_ENABLED
    #endif
    #if KF_SIMD >= 4
                    MENUITEM "16", ID_SIMD_VECTOR_SIZE_16,MFT_STRING,MFS_ENABLED
    #endif
710             END
                POPUP "SIMD chunk size",                    65535,MFT_STRING,MFS_ENABLED
                BEGIN
                    MENUITEM "1",   ID_SIMD_CHUNK_SIZE_1,MFT_STRING,MFS_ENABLED
                    MENUITEM "2",   ID_SIMD_CHUNK_SIZE_2,MFT_STRING,MFS_ENABLED
715                 MENUITEM "4",   ID_SIMD_CHUNK_SIZE_4,MFT_STRING,MFS_ENABLED
                    MENUITEM "8",   ID_SIMD_CHUNK_SIZE_8,MFT_STRING,MFS_ENABLED
                    MENUITEM "16",  ID_SIMD_CHUNK_SIZE_16,MFT_STRING,MFS_ENABLED
                    MENUITEM "32",  ID_SIMD_CHUNK_SIZE_32,MFT_STRING,MFS_ENABLED
                    MENUITEM "64",  ID_SIMD_CHUNK_SIZE_64,MFT_STRING,MFS_ENABLED
```

```
720                 MENUITEM "128", ID_SIMD_CHUNK_SIZE_128,MFT_STRING,MFS_ENABLED
                    MENUITEM "256", ID_SIMD_CHUNK_SIZE_256,MFT_STRING,MFS_ENABLED
                    MENUITEM "512", ID_SIMD_CHUNK_SIZE_512,MFT_STRING,MFS_ENABLED
                    MENUITEM "1024",ID_SIMD_CHUNK_SIZE_1024,MFT_STRING,MFS_ENABLED
                END
725             POPUP "Threads per CPU",                    65535,MFT_STRING,MFS_ENABLED
                BEGIN
                    MENUITEM "1/4", ID_ACTIONS_THREADS_1_4,MFT_STRING,MFS_ENABLED
                    MENUITEM "1/2", ID_ACTIONS_THREADS_1_2,MFT_STRING,MFS_ENABLED
                    MENUITEM "1",   ID_ACTIONS_THREADS_1,MFT_STRING,MFS_ENABLED
730                 MENUITEM "2",   ID_ACTIONS_THREADS_2,MFT_STRING,MFS_ENABLED
                    MENUITEM "4",   ID_ACTIONS_THREADS_4,MFT_STRING,MFS_ENABLED
                    MENUITEM "Reduce thread count by 1", ID_ACTIONS_THREADS_RESERVE_CORE↙
                        ↳ ,MFT_STRING,MFS_ENABLED
                END
    #ifdef KF_OPENCL
735             MENUITEM "OpenCL",                          ID_SPECIAL_OPENCL,MFT_STRING,↙
                    ↳ MFS_ENABLED
    #endif
        END
        MENUITEM "?", ID_MENU_ABOUT,MFT_STRING | MFT_RIGHTJUSTIFY,MFS_ENABLED
    END
740

    /////////////////////////////////////////////////////////////////////////////
    //
    // Icon
745 //

    // Icon with lowest ID value placed first to ensure application icon
    // remains consistent on all systems.
    IDI_ICON1               ICON                        "icon.ico"
750
    /////////////////////////////////////////////////////////////////////////////
    //
    // Cursor
    //
755
    IDC_CURSOR1             CURSOR                      "cursor1.cur"
    #endif    // Swedish (Sweden) resources
    /////////////////////////////////////////////////////////////////////////////


760


    #ifndef APSTUDIO_INVOKED
    /////////////////////////////////////////////////////////////////////////////
    //
765 // Generated from the TEXTINCLUDE 3 resource.
    //


    /////////////////////////////////////////////////////////////////////////////
770 #endif    // not APSTUDIO_INVOKED
```

68 fraktal_sft/gradient.cpp

```
/*
```

```
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <gsl/gsl_linalg.h>

#if 0

// neighbourhood size
#define S 8

// https://github.com/numpy/numpy/blob/020f2ad4e133b7500c7062d89368d38f19e836fd/
    ↳ numpy/linalg/linalg.py#L1681
// https://www.gnu.org/software/gsl/doc/html/linalg.html#c.gsl_linalg_SV_decomp
static void pinv(double out[S][S], const double in[S][S])
{
  double work_data[S];
  double u[S][S], s[S], v[S][S];
  // u = in
  for (int i = 0; i < S; ++i)
  for (int j = 0; j < S; ++j)
    u[i][j] = in[i][j];
  // initialize GSL
  gsl_vector work = { S, 1, &work_data[0], nullptr, 0 };
  gsl_matrix m_u = { S, S, S, &u[0][0], nullptr, 0 };
  gsl_matrix m_v = { S, S, S, &v[0][0], nullptr, 0 };
  gsl_vector v_s = { S, 1, &s[0], nullptr, 0 };
  // u, s, v = svd(a)
  gsl_linalg_SV_decomp(&m_u, &m_v, &v_s, &work);
  // s = 1/s
  double smax = s[0];
  const double epsilon = 1e-15;
  double cutoff = epsilon * S * smax;
  for (int i = 0; i < S; ++i)
    if (s[i] > cutoff)
      s[i] = 1 / s[i];
    else
      s[i] = 0;
  // su = multiply(s[..., newaxis], transpose(u)))
  double su[S][S];
  for (int i = 0; i < S; ++i)
  for (int j = 0; j < S; ++j)
    su[i][j] = s[i] * u[j][i]; // FIXME check
```

```
  // res = matmul(v, su)
  for (int i = 0; i < S; ++i)
  for (int j = 0; j < S; ++j)
  {
    out[i][j] = 0;
    for (int k = 0; k < S; ++k)
      out[i][j] += v[i][k] * su[k][j];
  }
}

// https://stackoverflow.com/questions/40338386/calculating-a-3d-gradient-with-unevenly-spaced-points/40520133#40520133
extern void compute_gradient_3x3(const double p[3][3], const double px[3][3], const double py[3][3], double &dx_out, double &dy_out)
{
  // inputs
  double C[2] = { px[1][1], py[1][1] };
  double Tc = p[1][1];
  double X[S][2]
    = { { px[0][0], py[0][0] }
      , { px[0][1], py[0][1] }
      , { px[0][2], py[0][2] }
      , { px[1][0], py[1][0] }
      , { px[1][2], py[1][2] }
      , { px[2][0], py[2][0] }
      , { px[2][1], py[2][1] }
      , { px[2][2], py[2][2] }
      };
  double T[S]
    = { p[0][0]
      , p[0][1]
      , p[0][2]
      , p[1][0]
      , p[1][2]
      , p[2][0]
      , p[2][1]
      , p[2][2]
      };
  // dX = X - C
  double dX[S][2];
  for (int i = 0; i < S; ++i)
  for (int j = 0; j < 2; ++j)
    dX[i][j] = X[i][j] - C[j];
  // G = dot(dX, dX.T)
  double G[S][S];
  for (int i = 0; i < S; ++i)
  for (int j = 0; j < S; ++j)
    G[i][j] = dX[i][0] * dX[j][0] + dX[i][1] * dX[j][1];
  // F = multiply(G, G)
  double F[S][S];
  for (int i = 0; i < S; ++i)
  for (int j = 0; j < S; ++j)
    F[i][j] = G[i][j] * G[i][j];
  // v = diag(G)
  double v[S];
  for (int i = 0; i < S; ++i)
    v[i] = G[i][i];
```

```
    // G1 = pinv(G)
    double G1[S][S];
    pinv(G1, G);
    // N = dot(G1, G)
    double N[S][S];
    for (int i = 0; i < S; ++i)
  {
    for (int j = 0; j < S; ++j)
    {
      N[i][j] = 0;
      for (int k = 0; k < S; ++k)
        N[i][j] += G1[i][k] * G[k][j];
  }
    }
    // N = eye(k) - N
    for (int i = 0; i < S; ++i)
    for (int j = 0; j < S; ++j)
      N[i][j] = (i == j ? 1 : 0) - N[i][j];
    // FN = dot(F, N)
    double FN[S][S];
    for (int i = 0; i < S; ++i)
    for (int j = 0; j < S; ++j)
    for (int k = 0; k < S; ++k)
    {
      FN[i][j] = 0;
      for (int k = 0; k < S; ++k)
        FN[i][j] += F[i][k] * N[k][j];
    }
    // FN1 = pinv(FN)
    double FN1[S][S];
    pinv(FN1, FN);
    // M = dot(N, FN1)
    double M[S][S];
    for (int i = 0; i < S; ++i)
    for (int j = 0; j < S; ++j)
    {
      M[i][j] = 0;
      for (int k = 0; k < S; ++k)
        M[i][j] += N[i][k] * FN1[k][j];
    }
    // a = dot(M, v)
    double a[S];
    for (int i = 0; i < S; ++i)
    {
      a[i] = 0;
      for (int j = 0; j < S; ++j)
        a[i] += M[i][j] * v[j];
    }
    // dT = T - Tc
    double dT[S];
    for (int i = 0; i < S; ++i)
      dT[i] = T[i] - Tc;
    // dT = multiply(a, dT)
    for (int i = 0; i < S; ++i)
      dT[i] = a[i] * dT[i];
    // y = dot(dX.T, a)
    double y[2];
```

```
    for (int i = 0; i < 2; ++i)
    {
      y[i] = 0;
      for (int j = 0; j < S; ++j)
        y[i] += dX[j][i] * dT[j];
    }
    // output
    dx_out = y[0];
    dy_out = y[1];
  }

  #endif

  extern void compute_gradient_2x2(const double p[3][3], const double px[3][3], ↙
      ↳ const double py[3][3], double &dx_out, double &dy_out)
  {
    // find weighted average of function values
    double num = 0, den = 0;
    for (int i = 0; i < 2; ++i)
      for (int j = 0; j < 2; ++j)
      {
        double qx = px[i][j] + 0.5;
        double qy = py[i][j] + 0.5;
        double w = 1 / (qx * qx + qy * qy + 1e-100);
        num += w * p[i][j];
        den += w;
      }
    double p0 = num / den;
    // initialize system A x = b for solving x = [ dF/dx ; dF/dy ]
    double A[4][2];
    double b[4];
    int k = 0;
    for (int i = 0; i < 2; ++i)
      for (int j = 0; j < 2; ++j)
      {
        A[k][0] = px[i][j] + 0.5;
        A[k][1] = py[i][j] + 0.5;
        b[k] = p[i][j] - p0;
        ++k;
      }
    // least-squares solve overdetermined A x = b via QR decomposition
    double tau[2] = { 0, 0 };
    double x[2] = { 0, 0 };
    double e[4] = { 0, 0, 0, 0 };
    gsl_matrix A_m   = { 4, 2, 2, &A[0][0], nullptr, 0 };
    gsl_vector b_v   = { 4, 1, &b[0], nullptr, 0 };
    gsl_vector tau_v = { 2, 1, &tau[0], nullptr, 0 };
    gsl_vector x_v   = { 2, 1, &x[0], nullptr, 0 };
    gsl_vector e_v   = { 4, 1, &e[0], nullptr, 0 };
    gsl_linalg_QR_decomp(&A_m, &tau_v);
    gsl_linalg_QR_lssolve(&A_m, &tau_v, &b_v, &x_v, &e_v);
    // output
    dx_out = x[0];
    dy_out = x[1];
  }

  extern void compute_gradient_3x3(const double p[3][3], const double px[3][3], ↙
```

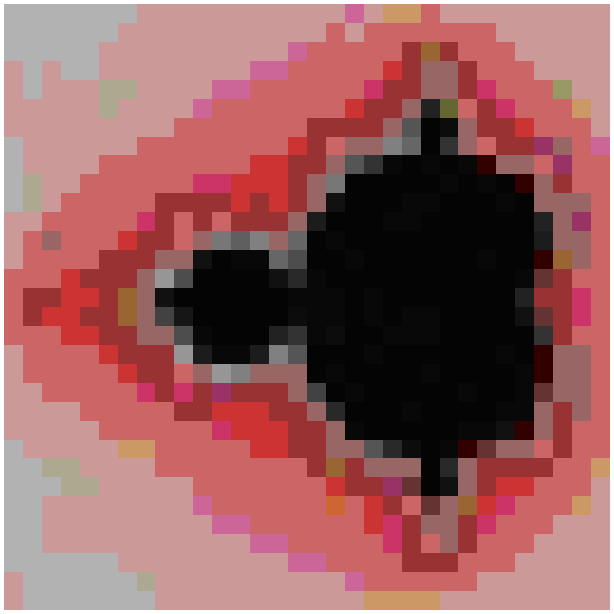
```
    ↳ const double py[3][3], double &dx_out, double &dy_out)
{
  double p0 = p[1][1];
  // initialize system A x = b for solving x = [ dF/dx ; dF/dy ]
  double A[8][2];
  double b[8];
  int k = 0;
  for (int i = 0; i < 3; ++i)
    for (int j = 0; j < 3; ++j)
    {
      if (i == 1 && j == 1) continue;
      A[k][0] = px[i][j] - px[1][1];
      A[k][1] = py[i][j] - py[1][1];
      b[k] = p[i][j] - p0;
      ++k;
    }
  // least-squares solve overdetermined A x = b via QR decomposition
  double tau[2] = { 0, 0 };
  double x[2] = { 0, 0 };
  double e[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };
  gsl_matrix A_m   = { 8, 2, 2, &A[0][0], nullptr, 0 };
  gsl_vector b_v   = { 8, 1, &b[0], nullptr, 0 };
  gsl_vector tau_v = { 2, 1, &tau[0], nullptr, 0 };
  gsl_vector x_v   = { 2, 1, &x[0], nullptr, 0 };
  gsl_vector e_v   = { 8, 1, &e[0], nullptr, 0 };
  gsl_linalg_QR_decomp(&A_m, &tau_v);
  gsl_linalg_QR_lssolve(&A_m, &tau_v, &b_v, &x_v, &e_v);
  // output
  dx_out = x[0];
  dy_out = x[1];
}
```

69 fraktal_sft/gradient.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_GRADIENT_H
#define KF_GRADIENT_H 1

extern void compute_gradient_3x3(const double p[3][3], const double px[3][3], ↙
```

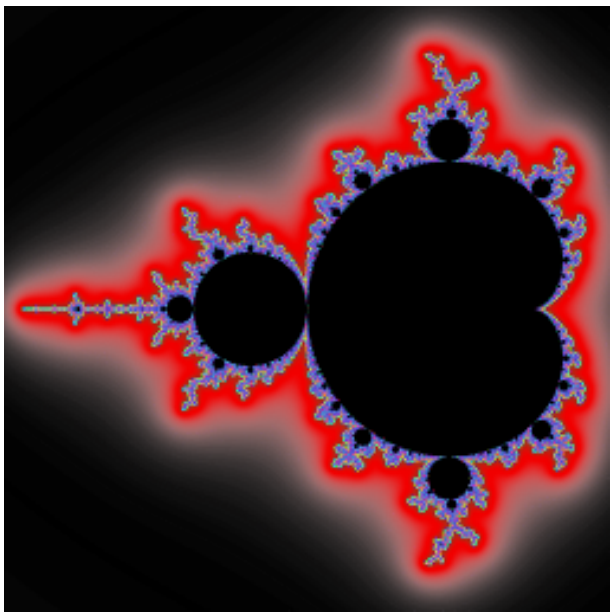
```
    ↪ const double py[3][3], double &dx_out, double &dy_out);
extern void compute_gradient_2x2(const double p[3][3], const double px[3][3], ↙
    ↪ const double py[3][3], double &dx_out, double &dy_out);

#endif
```

70 fraktal_sft/icon1.ico

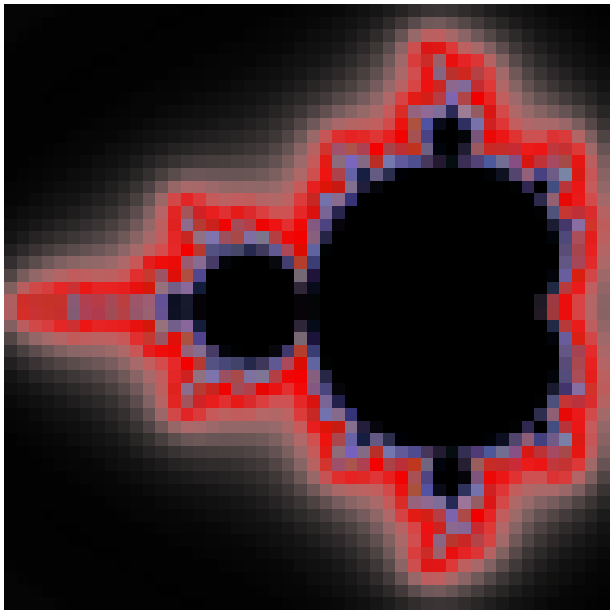


71 fraktal_sft/icon.ico

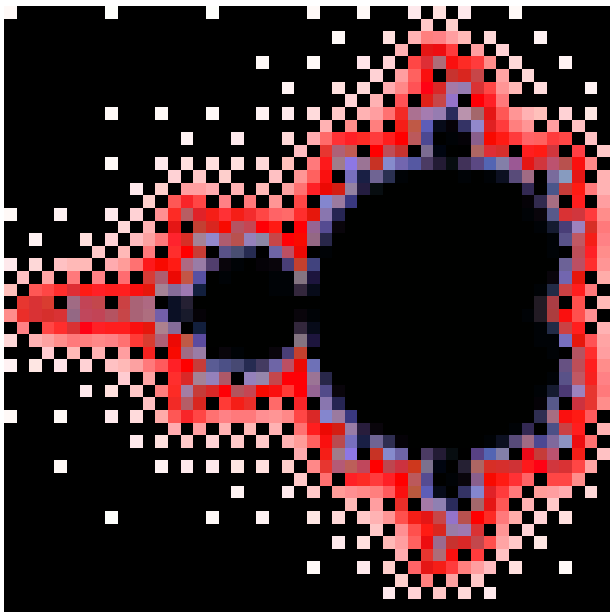
71.1 Layer 0



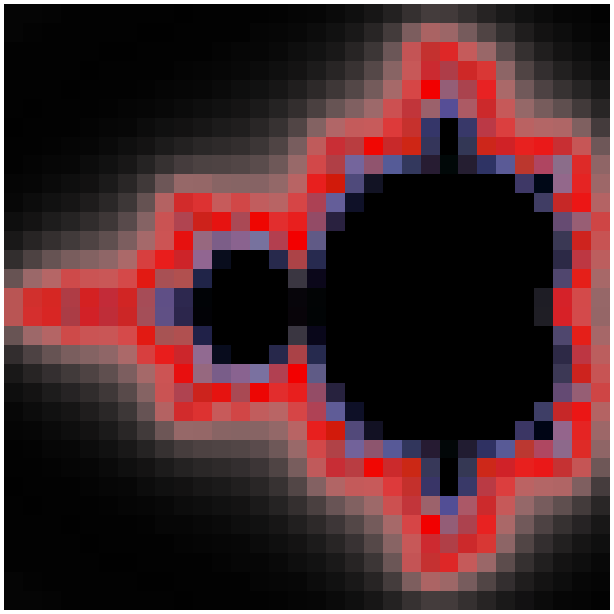
71.2 Layer 1



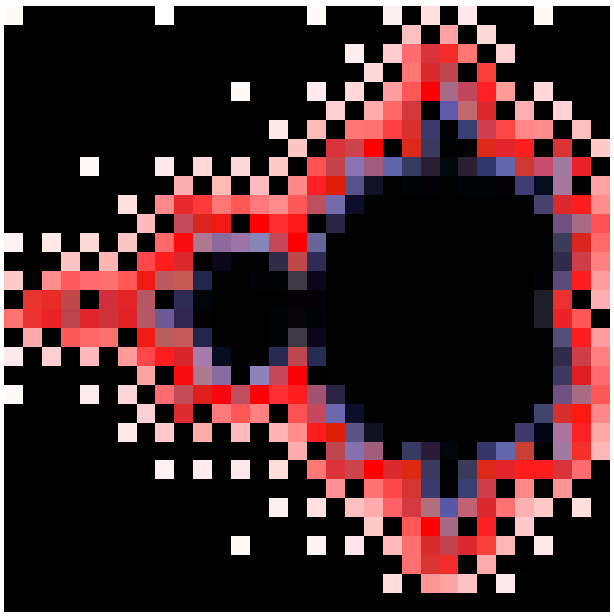
71.3 Layer 2



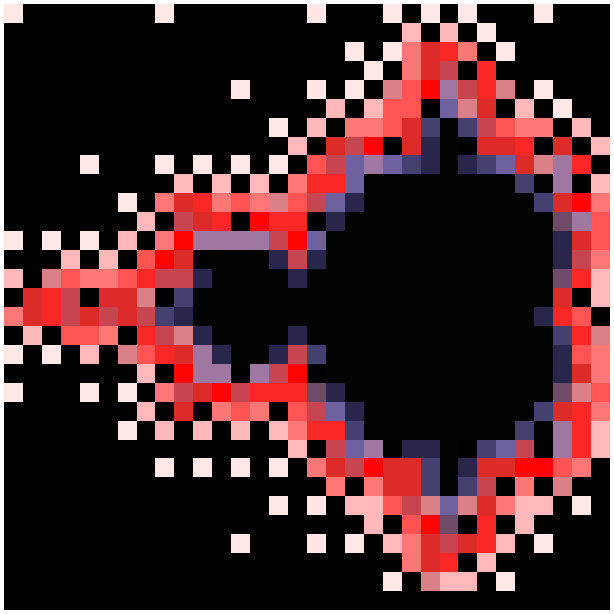
71.4 Layer 3



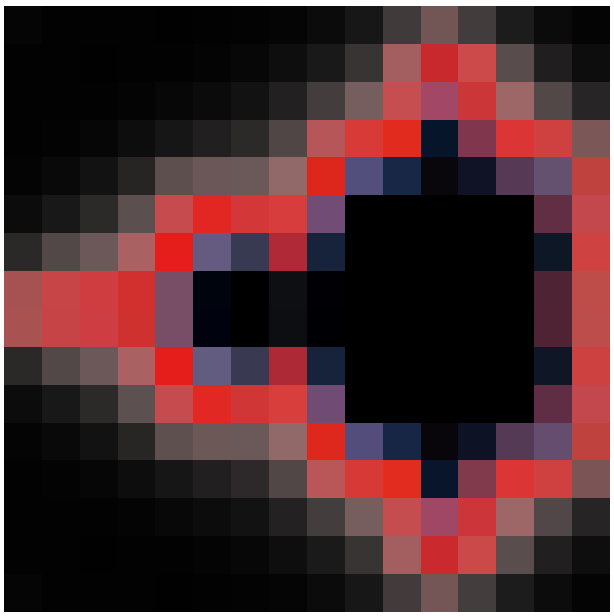
71.5 Layer 4



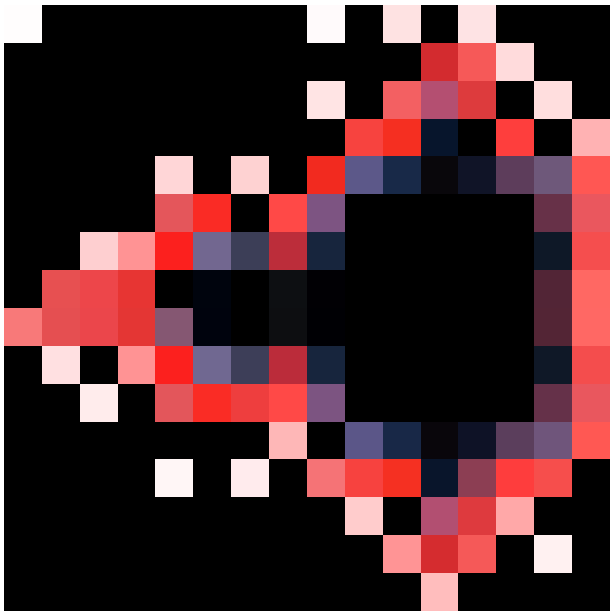
71.6 Layer 5



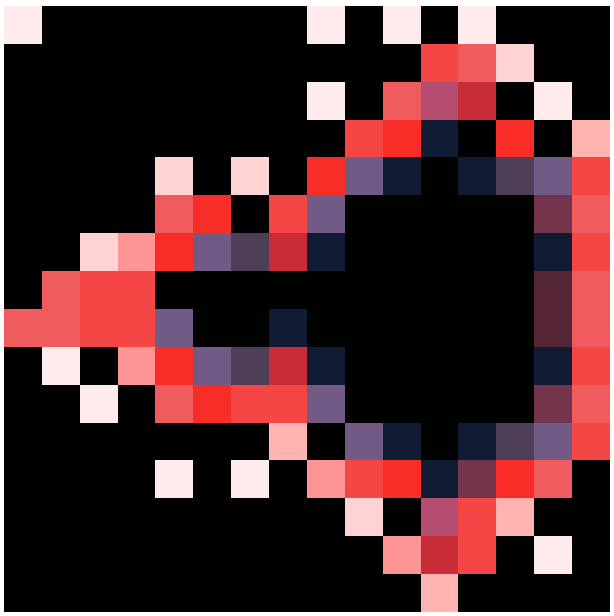
71.7 Layer 6



71.8 Layer 7



71.9 Layer 8



72 fraktal_sft/itercount_array.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
```

```
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #ifndef KF_ITERCOUNT_ARRAY_H
    #define KF_ITERCOUNT_ARRAY_H 1

    // a[x][y] behaves like int64_t& but with potentially less memory usage

25  struct itercount_array_item_ref
    {
    private:
      uint32_t *lsb;
      uint32_t *msb; // may be nullptr
30  public:
      inline itercount_array_item_ref(uint32_t *lsb, uint32_t *msb = nullptr)
      : lsb(lsb)
      , msb(msb)
      {
35    }
      inline itercount_array_item_ref &operator=(const int64_t &n)
      {
        if (msb)
          *msb = (n >> 32) & 0xFFffFFffU;
40      *lsb = n & 0xFFffFFffU;
        return *this;
      }
      inline operator int64_t() const
      {
45      int64_t n = *lsb;
        if (msb)
          n |= int64_t(*msb) << 32;
        else
          // sign extend
50        n = int64_t(int32_t(n));
        return n;
      }
    };

55  struct itercount_array_column_ref
    {
    private:
      ptrdiff_t stridey;
      uint32_t *lsb;
60    uint32_t *msb; // may be nullptr
    public:
      inline itercount_array_column_ref(ptrdiff_t stridey, uint32_t *lsb, uint32_t *↙
          ↳ msb = nullptr)
```

```
  : stridey(stridey)
  , lsb(lsb)
  , msb(msb)
  {
  }
  inline const itercount_array_item_ref operator[](int i) const
  {
    ptrdiff_t y = i * stridey;
    return itercount_array_item_ref(lsb + y, msb ? msb + y : msb);
  }
  inline itercount_array_item_ref operator[](int i)
  {
    ptrdiff_t y = i * stridey;
    return itercount_array_item_ref(lsb + y, msb ? msb + y : msb);
  }
};

struct itercount_array
{
private:
  ptrdiff_t stridex;
  ptrdiff_t stridey;
  uint32_t *lsb;
  uint32_t *msb; // may be nullptr
public:
  inline itercount_array(ptrdiff_t stridex, ptrdiff_t stridey, uint32_t *lsb, ↙
      ↳ uint32_t *msb = nullptr)
  : stridex(stridex)
  , stridey(stridey)
  , lsb(lsb)
  , msb(msb)
  {
  }
  inline const itercount_array_column_ref operator[](int i) const
  {
    ptrdiff_t x = i * stridex;
    return itercount_array_column_ref(stridey, lsb + x, msb ? msb + x : msb);
  }
  inline itercount_array_column_ref operator[](int i)
  {
    ptrdiff_t x = i * stridex;
    return itercount_array_column_ref(stridey, lsb + x, msb ? msb + x : msb);
  }
  inline operator bool() const
  {
    return lsb;
  }
};

#endif
```

73 fraktal_sft/jpeg.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen
```

```
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include <assert.h>
    #include <stdio.h>
    #include <string.h>
    #include <string>
    extern "C"
25  {
    #include <jpeglib.h>
    }
    #include <setjmp.h>
    #include <memory.h>
30  #include <malloc.h>

    #include "jpeg.h"

    #if 0
35
    #if defined(__GNUC__) && !defined(__clang__)
    #pragma GCC diagnostic ignored "-Wclobbered"
    #endif

40  struct my_error_mgr {
      struct jpeg_error_mgr pub;    /* "public" fields */

      jmp_buf setjmp_buffer;        /* for return to caller */
    };
45
    typedef struct my_error_mgr * my_error_ptr;
    METHODDEF(void)
    my_error_exit (j_common_ptr cinfo)
    {
50    /* cinfo->err really points to a my_error_mgr struct, so coerce pointer */
      my_error_ptr myerr = (my_error_ptr) cinfo->err;

      /* Always display the message. */
      /* We could postpone this until after returning, if we chose. */
55    (*cinfo->err->output_message) (cinfo);

      /* Return control to the setjmp point */
      longjmp(myerr->setjmp_buffer, 1);
    }
60
    int ReadJPG (char * filename,char **ppData, int *pnWidth, int *pnHeight,int *↙
```

```
    ↪ pnComponents)
{
  /* This struct contains the JPEG decompression parameters and pointers to
   * working space (which is allocated as needed by the JPEG library).
   */
  struct jpeg_decompress_struct cinfo;
  int nBuffSize=0;
  char *ReturnBuff=NULL;
  /* We use our private extension JPEG error handler.
   * Note that this struct must live as long as the main JPEG parameter
   * struct, to avoid dangling-pointer problems.
   */
  struct my_error_mgr jerr;
  /* More stuff */
  FILE * infile;                /* source file */
  JSAMPARRAY buffer;            /* Output row buffer */
  int row_stride;               /* physical row width in output buffer */

  /* In this example we want to open the input file before doing anything else,
   * so that the setjmp() error recovery below can assume the file is open.
   * VERY IMPORTANT: use "b" option to fopen() if you are on a machine that
   * requires it in order to read binary files.
   */

  if ((infile = fopen(filename, "rb")) == NULL)
    return 0;

  /* Step 1: allocate and initialize JPEG decompression object */

  /* We set up the normal JPEG error routines, then override error_exit. */
  cinfo.err = jpeg_std_error(&jerr.pub);
  jerr.pub.error_exit = my_error_exit;
  /* Establish the setjmp return context for my_error_exit to use. */
  if (setjmp(jerr.setjmp_buffer)) {
    /* If we get here, the JPEG code has signaled an error.
     * We need to clean up the JPEG object, close the input file, and return.
     */
    jpeg_destroy_decompress(&cinfo);
    fclose(infile);
    return 0;
  }
  /* Now we can initialize the JPEG decompression object. */
  jpeg_create_decompress(&cinfo);

  /* Step 2: specify data source (eg, a file) */

  jpeg_stdio_src(&cinfo, infile);

  /* Step 3: read file parameters with jpeg_read_header() */

  (void) jpeg_read_header(&cinfo, TRUE);
  *pnWidth = cinfo.image_width;
  *pnHeight = cinfo.image_height;
  *pnComponents = cinfo.num_components;
  /* We can ignore the return value from jpeg_read_header since
   *   (a) suspension is not possible with the stdio data source, and
   *   (b) we passed TRUE to reject a tables-only JPEG file as an error.
```

```
     * See libjpeg.doc for more info.
     */

    /* Step 4: set parameters for decompression */

    /* In this example, we don't need to change any of the defaults set by
     * jpeg_read_header(), so we do nothing here.
     */

    /* Step 5: Start decompressor */

    (void) jpeg_start_decompress(&cinfo);
    /* We can ignore the return value since suspension is not possible
     * with the stdio data source.
     */

    /* We may need to do some setup of our own at this point before reading
     * the data.  After jpeg_start_decompress() we have the correct scaled
     * output image dimensions available, as well as the output colormap
     * if we asked for color quantization.
     * In this example, we need to make an output work buffer of the right size.
     */
    /* JSAMPLEs per row in output buffer */
    row_stride = cinfo.output_width * cinfo.output_components;
    /* Make a one-row-high sample array that will go away when done with image */
    buffer = (*cinfo.mem->alloc_sarray)
                ((j_common_ptr) &cinfo, JPOOL_IMAGE, row_stride, 1);

    /* Step 6: while (scan lines remain to be read) */
    /*           jpeg_read_scanlines(...); */

    /* Here we use the library's state variable cinfo.output_scanline as the
     * loop counter, so that we don't have to keep track ourselves.
     */
    while (cinfo.output_scanline < cinfo.output_height) {
      /* jpeg_read_scanlines expects an array of pointers to scanlines.
       * Here the array is only one element long, but you could ask for
       * more than one scanline at a time if that's more convenient.
       */
      (void) jpeg_read_scanlines(&cinfo, buffer, 1);
      /* Assume put_scanline_someplace wants a pointer and sample count. */
          ReturnBuff = (char*)realloc(ReturnBuff,nBuffSize+row_stride);
      memcpy(&ReturnBuff[nBuffSize],buffer[0],row_stride);
          nBuffSize+=row_stride;
    }
    *ppData = ReturnBuff;

    /* Step 7: Finish decompression */

    (void) jpeg_finish_decompress(&cinfo);
    /* We can ignore the return value since suspension is not possible
     * with the stdio data source.
     */

    /* Step 8: Release JPEG decompression object */

    /* This is an important step since it will release a good deal of memory. */
```

```
175   jpeg_destroy_decompress(&cinfo);

      /* After finish_decompress, we can close the input file.
       * Here we postpone it until after no more JPEG errors are possible,
       * so as to simplify the setjmp error logic above.  (Actually, I don't
180    * think that jpeg_destroy can do an error exit, but why assume anything...)
       */
      fclose(infile);

      /* At this point you may want to check to see whether any corrupt-data
185    * warnings occurred (test whether jerr.pub.num_warnings is nonzero).
       */

      /* And we're done! */
      return 1;
190 }

    #endif


195 int SaveJPG(const std::string &szFileName, char *Data, int nHeight, int nWidth, ↙
        ↳ int nColors, int nQuality, const std::string &comment)
    {
      if (nColors != 3)
        return 0;
      struct jpeg_compress_struct cinfo;
200   struct jpeg_error_mgr jerr;

      FILE * outfile;               /* target file */
      JSAMPROW row_pointer[1];      /* pointer to JSAMPLE row[s] */
      int row_stride;               /* physical row width in image buffer */
205
      cinfo.err = jpeg_std_error(&jerr);
      int nStructSize = sizeof(jpeg_compress_struct);
      jpeg_CreateCompress(&cinfo,JPEG_LIB_VERSION,nStructSize);
      if ((outfile = fopen(szFileName.c_str(), "wb")) == NULL) {
210     return 0;//exit(1);
      }
      jpeg_stdio_dest(&cinfo, outfile);
      cinfo.image_width = nWidth;   /* image width and height, in pixels */
      cinfo.image_height = nHeight;
215   cinfo.input_components = 3;           /* # of color components per pixel */
      cinfo.in_color_space = JCS_RGB;       /* colorspace of input image */
      jpeg_set_defaults(&cinfo);
      jpeg_set_quality(&cinfo, nQuality, TRUE /* limit to baseline-JPEG values */);
      jpeg_start_compress(&cinfo, TRUE);
220   size_t length = comment.length();
      const char *comment_str = comment.c_str();
      do {
        size_t wlength = length;
        if (wlength > 65533)
225       wlength = 65533;
        if (wlength > 0)
          jpeg_write_marker(&cinfo, JPEG_COM, (const unsigned char *) comment_str, ↙
              ↳ wlength);
        comment_str += wlength;
        length -= wlength;
```

```
    } while (length > 0);
    row_stride = nWidth * 3;       /* JSAMPLEs per row in image_buffer */
    while (cinfo.next_scanline < cinfo.image_height) {
      row_pointer[0] = (unsigned char*)& Data[cinfo.next_scanline * row_stride];
      (void) jpeg_write_scanlines(&cinfo, row_pointer, 1);
    }

    jpeg_finish_compress(&cinfo);
    fclose(outfile);
    jpeg_destroy_compress(&cinfo);
    return 1;
}


std::string ReadJPEGComment(const std::string &filename)
{
    std::string comment = "";
    FILE *file = fopen(filename.c_str(), "rb");
    if (! file)
    {
      return "";
    }
    int c = fgetc(file);
    if (c != 0xFF)
    {
      fclose(file);
      return "";
    }
    c = fgetc(file);
    if (c != 0xD8) // SOI
    {
      fclose(file);
      return "";
    }
    do {
      do {
        c = fgetc(file);
      } while (0 <= c && c <= 0xFF && c != 0xFF);
      if (! (0 <= c && c <= 0xFF))
        break;
      // c is 0xFF
      do {
        c = fgetc(file);
      } while (0 <= c && c <= 0xFF && c == 0xFF);
      if (! (0 <= c && c <= 0xFF))
        break;
      // c is now the marker byte
      if (c == 0xFE) // COM
      {
        // read 2 byte length
        int l1 = fgetc(file);
        if (! (0 <= l1 && l1 <= 0xFF))
          break;
        int l2 = fgetc(file);
        if (! (0 <= l2 && l2 <= 0xFF))
          break;
        int length = 0;
```

```
      length = ((l1 << 8) | l2) - 2; // length includes itself
      if (length > 0)
      {
        // read comment
        char *buffer = (char *) malloc(length + 1);
        if (! buffer)
        {
          fclose(file);
          return "";
        }
        buffer[length] = 0;
        if (1 != fread(buffer, length, 1, file))
        {
          free(buffer);
          fclose(file);
          return "";
        }
        comment += buffer;
        free(buffer);
      }
      else if (length < 0)
      {
        fclose(file);
        return "";
      }
    }
    else if (c == 0xD9 || c == 0xDA) // EOI SOS
    {
      // end of file, start of compressed data, or so
      break;
    }
    else
    {
      // read 2 byte length
      int l1 = fgetc(file);
      if (! (0 <= l1 && l1 <= 0xFF))
        break;
      int l2 = fgetc(file);
      if (! (0 <= l2 && l2 <= 0xFF))
        break;
      int length = 0;
      length = ((l1 << 8) | l2) - 2;
      if (length >= 0)
      {
        if (0 != fseek(file, length, SEEK_CUR))
        {
          fclose(file);
          return "";
        }
      }
      else
      {
        fclose(file);
        return "";
      }
    }
  } while(true);
```

```
  fclose(file);
  return comment;
}
```

74 fraktal_sft/jpeg.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_JPEG_H
#define KF_JPEG_H 1

#include <string>

extern int SaveJPG(const std::string &szFileName, char *Data, int nHeight, int ↙
    ↳ nWidth, int nColors, int nQuality, const std::string &comment);
extern std::string ReadJPEGComment(const std::string &filename);

#endif
```

75 fraktal_sft/listbox.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/
```

```
20  #include <windows.h>
    #include "listbox.h"
    #include "../common/getimage.h"

    #ifdef _WIN64
25  #define GCL_WNDPROC -24
    #define GWL_WNDPROC -4
    #define GWL_USERDATA -21
    #endif

30  int WINAPI SubclassListProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
    {
            if(uMsg==WM_COMMAND){
                    CListBox*pList = (CListBox*)GetWindowLongPtr(hWnd,GWLP_USERDATA)↙
                        ↳ ;
                    if(pList)
35                          pList->OnCommand((HWND)lParam);
            }
            else if(uMsg==WM_SIZE || uMsg==WM_VSCROLL){
                    CListBox*pList = (CListBox*)GetWindowLongPtr(hWnd,GWLP_USERDATA)↙
                        ↳ ;
                    if(pList)
40                          pList->ArrangeButtons(uMsg==WM_VSCROLL);
            }
            else if(uMsg==WM_DRAWITEM){
                    CListBox*pList = (CListBox*)GetWindowLongPtr(hWnd,GWLP_USERDATA)↙
                        ↳ ;
                    if(pList)
45                          pList->DrawButton((LPDRAWITEMSTRUCT)lParam);
            }
            return CallWindowProc((WNDPROC)GetClassLongPtr(hWnd,GCLP_WNDPROC),hWnd,↙
                ↳ uMsg,wParam,lParam);
    }
    char *CListBox::CopyString(char *szString)
50  {
            int nLen = strlen(szString);
            char *szRet = new char[nLen+1];
            strcpy(szRet,szString);
            return szRet;
55  }
    void CListBox::DrawItem(LPDRAWITEMSTRUCT lpdis)
    {
            if(lpdis->itemID==(UINT)(-1))
                    return;
60          int i;
            COLORREF col = (lpdis->itemID%2?m_colRow2:m_colRow1);
            if(m_bShowSelection && (lpdis->itemState & ODS_SELECTED)){
                    char *szCol = (char*)&col;
                    col = RGB(~szCol[0],~szCol[1],~szCol[2]);
65                  SetTextColor(lpdis->hDC,RGB(255,255,255));
            }
            else
                    SetTextColor(lpdis->hDC,RGB(0,0,0));
            HBRUSH br = CreateSolidBrush(col);
70          FillRect(lpdis->hDC,&lpdis->rcItem,br);
            DeleteObject(br);
```

```
        SetBkMode(lpdis->hDC,TRANSPARENT);
        HFONT hfOld = (HFONT)SelectObject(lpdis->hDC, m_hfBold);
        TextOutA(lpdis->hDC,2,lpdis->rcItem.top+2,m_pRows[lpdis->itemID].szBoldText,strlen(m_pRows[lpdis->itemID].szBoldText));
        SelectObject(lpdis->hDC, m_hfNormal);
        TextOutA(lpdis->hDC,2,lpdis->rcItem.top+4+m_nHeight,m_pRows[lpdis->itemID].szNormalText,strlen(m_pRows[lpdis->itemID].szNormalText));
        SelectObject(lpdis->hDC, hfOld);
        for(i=0;i<m_pRows[lpdis->itemID].nButtons;i++){
                InvalidateRect(m_pRows[lpdis->itemID].phwButtons[i],NULL,FALSE);
                UpdateWindow(m_pRows[lpdis->itemID].phwButtons[i]);
        }
}
void CListBox::ArrangeButtons(BOOL bVScroll)
{
        int i, j;
        RECT r;
        int nLeft=0;
        BOOL bInvalidate=FALSE;
        for(i=0;i<m_nRows;i++){
                SendMessageA(m_hwList,LB_GETITEMRECT,i,(LPARAM)&r);
                for(j=0;j<m_pRows[i].nButtons;j++)
                        r.right-=m_pButtons[j].nWidth+2;
                if(!nLeft)
                        nLeft = r.right-2;
                for(j=0;j<m_pRows[i].nButtons;j++){
                        RECT wr;
                        GetWindowRect(m_pRows[i].phwButtons[j],&wr);
                        ScreenToClient(m_hwList,(LPPOINT)&wr);
                        if(wr.left!=r.right || wr.top!=r.top+2)
                                bInvalidate=TRUE;
                        MoveWindow(m_pRows[i].phwButtons[j],r.right,r.top+2,m_pButtons[j].nWidth,m_pButtons[j].nHeight,TRUE);
                        r.right+=m_pButtons[j].nWidth+2;
                }
        }
        if(bVScroll || bInvalidate || 1){
                GetClientRect(m_hwList,&r);
                r.left=nLeft;
                InvalidateRect(m_hwList,&r,FALSE);
        }
}
void CListBox::OnCommand(HWND hWnd)
{
        int i, j;
        for(i=0;i<m_nRows;i++)
                for(j=0;j<m_pRows[i].nButtons;j++)
                        if(m_pRows[i].phwButtons[j]==hWnd){
                                m_pButtons[j].lpfnButtonProc(m_pButtons[j].nCommand,i,m_pButtons[j].lParam);
                                return;
                        }
}

CListBox::CListBox(HWND hwParent,HWND hwList,char *szFontFace, int nSize,COLORREF colRow1,COLORREF colRow2,BOOL bShowSelection)
```

```
{
125         m_hwParent = hwParent;
            m_hwList = hwList;
            SetWindowLongPtr(m_hwList,GWLP_USERDATA,(LONG_PTR)this);
            SetWindowLongPtr(m_hwList,GWLP_WNDPROC,(LONG_PTR)SubclassListProc);
            m_hfBold = CreateFontA(nSize,0,0,0,FW_BOLD,0,0,0,0,0,0,0,0,szFontFace);
130         m_hfNormal = CreateFontA(nSize,0,0,0,FW_NORMAL,0,0,0,0,0,0,0,0,↙
                ↳ szFontFace);
            m_nButtons = 0;
            m_pButtons = NULL;
            m_pRows = NULL;
            m_nRows = 0;
135
            HDC dcTest = GetDC(m_hwList);
            HFONT hfOld = (HFONT)SelectObject(dcTest,m_hfNormal);
            SIZE sc;
            GetTextExtentPointA(dcTest,"1",1,&sc);
140         m_nHeight = sc.cy;
            SelectObject(dcTest,hfOld);
            ReleaseDC(m_hwList,dcTest);
            SendMessageA(m_hwList,LB_SETITEMHEIGHT,0,m_nHeight*2+6);
            m_colRow1 = colRow1;
145         m_colRow2 = colRow2;
            m_bShowSelection = bShowSelection;
}
CListBox::~CListBox()
{
150         int i, j;
            for(i=0;i<m_nButtons;i++){
                    if(m_pButtons[i].szTitle)
                            delete m_pButtons[i].szTitle;
            }
155         if(m_pButtons)
                    free(m_pButtons);
            for(i=0;i<m_nRows;i++){
                    if(m_pRows[i].szBoldText)
                            delete m_pRows[i].szBoldText;
160                 if(m_pRows[i].szNormalText)
                            delete m_pRows[i].szNormalText;
                    for(j=0;j<m_pRows[i].nButtons;j++)
                            DestroyWindow(m_pRows[i].phwButtons[j]);
                    if(m_pRows[i].phwButtons)
165                         delete m_pRows[i].phwButtons;
            }
            if(m_pRows)
                    free(m_pRows);
            DeleteObject(m_hfBold);
170         DeleteObject(m_hfNormal);
}

void CListBox::AddButton(int nType,char *szTitle,HBITMAP bmBmp,int nCommand,↙
    ↳ LPBUTTONPROC lpfnButtonProc,LPARAM lParam)
{
175         int i = m_nButtons++;
            m_pButtons = (BUTTON*)realloc(m_pButtons,sizeof(BUTTON)*m_nButtons);
            memset(&m_pButtons[i],0,sizeof(BUTTON));
            if(nType==BUTTONSTRING){
```

```
                m_pButtons[i].szTitle = CopyString(szTitle);
                HDC dcTest = GetDC(m_hwList);
                HFONT hfOld = (HFONT)SelectObject(dcTest,(HFONT)GetStockObject(↙
                    ↳ ANSI_VAR_FONT));
                SIZE sc;
                GetTextExtentPointA(dcTest,szTitle,strlen(szTitle),&sc);
                m_pButtons[i].nHeight = sc.cy+8;
                m_pButtons[i].nWidth = sc.cx+8;
                SelectObject(dcTest,hfOld);
                ReleaseDC(m_hwList,dcTest);
        }
        else{
                m_pButtons[i].bmBmp = (HBITMAP)bmBmp;
                BITMAP bm;
                GetObject(bmBmp,sizeof(BITMAP),&bm);
                m_pButtons[i].nHeight = bm.bmHeight+6;
                m_pButtons[i].nWidth = bm.bmWidth+6;
        }
        m_pButtons[i].lpfnButtonProc = lpfnButtonProc;
        m_pButtons[i].nCommand = nCommand;
        m_pButtons[i].lParam = lParam;
}
void CListBox::AddRow(char *szBoldText, char *szNormalText)
{
        RECT r;
        int j, i = m_nRows++;
        m_pRows = (ROWS*)realloc(m_pRows,sizeof(ROWS)*m_nRows);
        memset(&m_pRows[i],0,sizeof(ROWS));
        m_pRows[i].szBoldText = CopyString(szBoldText);
        m_pRows[i].szNormalText = CopyString(szNormalText);

        i = SendMessageA(m_hwList,LB_ADDSTRING,0,(LPARAM)"");

        SendMessageA(m_hwList,LB_GETITEMRECT,i,(LPARAM)&r);
        m_pRows[i].nButtons = m_nButtons;
        m_pRows[i].phwButtons = new HWND[m_nButtons];
        for(j=0;j<m_pRows[i].nButtons;j++)
                r.right-=m_pButtons[j].nWidth+2;
        for(j=0;j<m_pRows[i].nButtons;j++){
                m_pRows[i].phwButtons[j] = CreateWindowA("button",m_pButtons[j].↙
                    ↳ szTitle,
                        (m_pButtons[j].bmBmp?BS_OWNERDRAW:0) | WS_CHILD|↙
                            ↳ WS_VISIBLE,r.right,r.top+2,m_pButtons[j].nWidth,
                        m_pButtons[j].nHeight,m_hwList,NULL,GetModuleHandle(NULL↙
                            ↳ ),0);
                SendMessageA(m_pRows[i].phwButtons[j],WM_SETFONT,(WPARAM)↙
                    ↳ GetStockObject(ANSI_VAR_FONT),0);
                r.right+=m_pButtons[j].nWidth+2;
        }
        ArrangeButtons();
}
void CListBox::DeleteRow(int nItem)
{
        int i, j;
        if(nItem<0 || nItem>m_nRows)
                return;
        if(m_pRows[nItem].szBoldText)
```

```
                delete m_pRows[nItem].szBoldText;
        if(m_pRows[nItem].szNormalText)
                delete m_pRows[nItem].szNormalText;
        for(j=0;j<m_pRows[nItem].nButtons;j++)
                DestroyWindow(m_pRows[nItem].phwButtons[j]);
        if(m_pRows[nItem].nButtons)
                delete m_pRows[nItem].phwButtons;
        m_nRows--;
        for(i=nItem;i<m_nRows;i++)
                m_pRows[i] = m_pRows[i+1];
        SendMessageA(m_hwList,LB_DELETESTRING,nItem,0);
        ArrangeButtons();
}
void CListBox::ClearList()
{
        while(m_nRows)
                DeleteRow(0);
}
int CListBox::ProcessMessage(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        (void) hWnd;
        if(uMsg==WM_DRAWITEM && wParam==(UINT)GetDlgCtrlID(m_hwList)){
                DrawItem((LPDRAWITEMSTRUCT)lParam);
                if(!m_bShowSelection)
                        return 1;
        }
        return 0;
}


CListBoxEdit::CListBoxEdit(HWND hwAdd, HWND hwUpdate, HWND hwRemove, HWND hwEdit
    ↪ , HWND hwList,HWND *phwEdits, int nEdits)
{
        m_hwAdd = hwAdd;
        m_hwUpdate = hwUpdate;
        m_hwRemove = hwRemove;
        m_hwEdit = hwEdit;
        m_hwList = hwList;
        int i;
        for(i=0;i<nEdits;i++)
                m_stEdits.AddInt((intptr_t)phwEdits[i]);
}
int CListBoxEdit::ProcessMessage(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam
    ↪ )
{
        (void) hWnd;
        if(uMsg==WM_COMMAND){
                if(lParam==(LPARAM)m_hwAdd){
                        int nLen = GetWindowTextLength(m_hwEdit);
                        int i;
                        for(i=0;i<m_stEdits.GetCount();i++)
                        {
                                intptr_t h = 0;
                                sscanf(m_stEdits[i], "%" SCNdPTR, &h);
                                nLen+=GetWindowTextLength((HWND)h)+1;
                        }
                        char *szTmp = new char[nLen+1];
```

```
                        GetWindowTextA(m_hwEdit,szTmp,nLen+1);
                        char szClass[256];
                        for(i=0;i<m_stEdits.GetCount();i++){
                                strcat(szTmp,"\t");
290                             intptr_t h = 0;
                                sscanf(m_stEdits[i], "%" SCNdPTR, &h);
                                GetClassName((HWND)h,szClass,sizeof(szClass));
                                if(!stricmp(szClass,"button"))
                                        itoa(SendMessage((HWND)h,BM_GETCHECK↙
                                            ↳ ,0,0),szTmp+strlen(szTmp),10);
295                             else
                                        GetWindowTextA((HWND)h,szTmp+strlen(↙
                                            ↳ szTmp),nLen+1);
                        }
                        SendMessageA(m_hwList,LB_ADDSTRING,0,(LPARAM)szTmp);
                        delete[] szTmp;
300                     SetFocus(m_hwEdit);
                        SendMessageA(m_hwEdit,EM_SETSEL,0,-1);
                }
                else if(lParam==(LPARAM)m_hwUpdate){
                        int nSel = SendMessageA(m_hwList,LB_GETCURSEL,0,0);
305                     if(nSel==-1)
                                return 0;
                        SendMessageA(m_hwList,LB_DELETESTRING,nSel,0);
                        int nLen = GetWindowTextLength(m_hwEdit);
                        int i;
310                     for(i=0;i<m_stEdits.GetCount();i++)
                        {
                                intptr_t h = 0;
                                sscanf(m_stEdits[i], "%" SCNdPTR, &h);
                                nLen+=GetWindowTextLength((HWND)h)+1;
315                     }
                        char *szTmp = new char[nLen+1];
                        GetWindowTextA(m_hwEdit,szTmp,nLen+1);
                        char szClass[256];
                        for(i=0;i<m_stEdits.GetCount();i++){
320                             strcat(szTmp,"\t");
                                intptr_t h = 0;
                                sscanf(m_stEdits[i], "%" SCNdPTR, &h);
                                GetClassName((HWND)h,szClass,sizeof(szClass));
                                if(!stricmp(szClass,"button"))
325                                     itoa(SendMessage((HWND)h,BM_GETCHECK↙
                                            ↳ ,0,0),szTmp+strlen(szTmp),10);
                                else
                                        GetWindowTextA((HWND)h,szTmp+strlen(↙
                                            ↳ szTmp),nLen+1);
                        }
                        SendMessageA(m_hwList,LB_INSERTSTRING,nSel,(LPARAM)szTmp↙
                            ↳ );
330                     delete[] szTmp;
                        SetFocus(m_hwEdit);
                        SendMessageA(m_hwEdit,EM_SETSEL,0,-1);
                }
                else if(lParam==(LPARAM)m_hwRemove){
335                     int i = SendMessageA(m_hwList,LB_GETCURSEL,0,0);
                        if(i==-1)
                                return 0;
```

```
                        SendMessageA(m_hwList,LB_DELETESTRING,i,0);
                        SetFocus(m_hwEdit);
340                     SendMessageA(m_hwEdit,EM_SETSEL,0,-1);
                }
                else if(lParam==(LPARAM)m_hwList && HIWORD(wParam)==↙
                    ↪ LBN_SELCHANGE){
                        int i = SendMessageA(m_hwList,LB_GETCURSEL,0,0);
                        if(i==-1)
345                             return 0;
                        int nLen = SendMessageA(m_hwList,LB_GETTEXTLEN,i,0);
                        char *szTmp = new char[nLen+1];
                        SendMessageA(m_hwList,LB_GETTEXT,i,(LPARAM)szTmp);
                        char szClass[256];
350                     if(m_stEdits.GetCount()){
                                CStringTable stT(szTmp,"\t","");
                                GetClassName(m_hwEdit,szClass,sizeof(szClass));
                                SetWindowTextA(m_hwEdit,stT[0][0]);
                                for(i=0;i<m_stEdits.GetCount();i++){
355                                     intptr_t h = 0;
                                        sscanf(m_stEdits[i], "%" SCNdPTR, &h);
                                        GetClassName((HWND)h,szClass,sizeof(↙
                                            ↪ szClass));
                                        if(!stricmp(szClass,"button"))
                                                SendMessage((HWND)h,BM_SETCHECK,↙
                                                    ↪ atoi(stT[0][i+1]),0);
360                                     else
                                                SetWindowTextA((HWND)h,stT[0][i↙
                                                    ↪ +1]);
                                }
                        }
                        else{
365                             GetClassName(m_hwEdit,szClass,sizeof(szClass));
                                if(!stricmp(szClass,"combobox")){
                                        int li;
                                        for(li=0;li<SendMessage(m_hwEdit,↙
                                            ↪ CB_GETCOUNT,0,0);li++){
                                                int nL = SendMessage(m_hwEdit,↙
                                                    ↪ CB_GETLBTEXTLEN,li,0);
370                                             char *szL = new char[nL+1];
                                                SendMessage(m_hwEdit,↙
                                                    ↪ CB_GETLBTEXT,li,(LPARAM)↙
                                                    ↪ szL);
                                                if(!strcmp(szTmp,szL)){
                                                        delete[] szL;
                                                        break;
375                                             }
                                                delete[] szL;
                                        }
                                        SendMessage(m_hwEdit,CB_SETCURSEL,li,0);
                                }
380                             else
                                        SetWindowTextA(m_hwEdit,szTmp);
                        }
                        delete[] szTmp;
                }
385     }
        return 0;
```

```
}
int CListBoxEdit::GetStrings(CStringVektor *psv)
{
        psv->Clean();
        int nLen = SendMessageA(m_hwList,LB_GETCOUNT,0,0);
        int i;
        for(i=0;i<nLen;i++){
                int nLen = SendMessageA(m_hwList,LB_GETTEXTLEN,i,0);
                char *szTmp = new char[nLen+1];
                SendMessageA(m_hwList,LB_GETTEXT,i,(LPARAM)szTmp);
                psv->AddString(szTmp);
                delete szTmp;
        }
        return nLen;
}
void CListBox::DrawButton(LPDRAWITEMSTRUCT lpdis)
{
        int i, j = 0;
        for(i=0;i<m_nRows;i++){
                for(j=0;j<m_pRows[i].nButtons;j++)
                        if(m_pRows[i].phwButtons[j]==lpdis->hwndItem)
                                break;
                if(j<m_pRows[i].nButtons)
                        break;
        }
        HDC dcBmp = CreateCompatibleDC(lpdis->hDC);
        HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,m_pButtons[j].bmBmp);
        BITMAP bm;
        GetObject(m_pButtons[j].bmBmp,sizeof(BITMAP),&bm);
        COLORREF col = GetPixel(dcBmp,0,0);
        HBRUSH br = CreateSolidBrush(col);
        FillRect(lpdis->hDC,&lpdis->rcItem,br);
        DeleteObject(br);
#ifndef NO_GETIMAGE
        RECT r = lpdis->rcItem;
        if(lpdis->itemState & ODS_SELECTED){
                SkuggadRect(lpdis->hDC,r,FALSE,TRUE,0);
                r.left++;
                r.top++;
                r.right--;
                r.bottom--;
                SkuggadRect(lpdis->hDC,r,FALSE,FALSE,0);
                BitBlt(lpdis->hDC,4,4,bm.bmWidth,bm.bmHeight,dcBmp,0,0,SRCCOPY);
        }
        else{
                SkuggadRect(lpdis->hDC,r,TRUE,TRUE,0);
                r.left++;
                r.top++;
                r.right--;
                r.bottom--;
                SkuggadRect(lpdis->hDC,r,TRUE,FALSE,0);
                BitBlt(lpdis->hDC,2,2,bm.bmWidth,bm.bmHeight,dcBmp,0,0,SRCCOPY);
        }
#endif
        SelectObject(dcBmp,bmOld);
        DeleteDC(dcBmp);
}
```

76 fraktal_sft/listbox.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef __LIST_H__
#define __LIST_H__

#include "../common/StringVector.h"

typedef int (*LPBUTTONPROC)(int nCommand, int nItem,LPARAM lParam);

enum BUTTONTYPES{
        BUTTONSTRING,
        BUTTONBITMAP
};
class CListBox
{
        HWND m_hwList;
        HWND m_hwParent;
        int m_nHeight;

        HFONT m_hfBold;
        HFONT m_hfNormal;

        COLORREF m_colRow1, m_colRow2;
        BOOL m_bShowSelection;

        struct BUTTON{
                char *szTitle;
                HBITMAP bmBmp;
                LPBUTTONPROC lpfnButtonProc;
                int nHeight;
                int nWidth;
                int nCommand;
                LPARAM lParam;
        }*m_pButtons;
        int m_nButtons;

        struct ROWS{
                char *szBoldText;
```

```
                char *szNormalText;
                HWND *phwButtons;
                int nButtons;
        }*m_pRows;
        int m_nRows;

        char *CopyString(char *szString);
        void DrawItem(LPDRAWITEMSTRUCT lpdis);
public:
        CListBox(HWND hwParent,HWND hwList,char *szFontFace, int nSize,COLORREF colRow1,COLORREF colRow2,BOOL bShowSelection);
        ~CListBox();

        void ArrangeButtons(BOOL bVScroll=FALSE);
        void OnCommand(HWND hWnd);
        void DrawButton(LPDRAWITEMSTRUCT lpdis);
        void AddButton(int nType,char *szTitle,HBITMAP bmBmp,int nCommand,LPBUTTONPROC lpfnButtonProc,LPARAM lParam);
        void AddRow(char *szBoldText, char *szNormalText);
        void DeleteRow(int nItem);
        void ClearList();
        int GetCount() {return m_nRows;}
        int ProcessMessage(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
};

class CListBoxEdit
{
        HWND m_hwAdd, m_hwUpdate, m_hwRemove;
        HWND m_hwEdit, m_hwList;
        CStringVektor m_stEdits;
public:
        CListBoxEdit(HWND hwAdd, HWND hwUpdate, HWND hwRemove, HWND hwEdit, HWND hwList,HWND *phwEdits=NULL, int nEdits=0);
        int ProcessMessage(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
        int GetStrings(CStringVektor *psv);
};
#endif//__LIST_H__
```

77 fraktal_sft/long_double_perturbation.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

```
*/

#include "fraktal_sft.h"
#include "floatexp.h"
#include "complex.h"
#include "../formula/formula.h"

static int Perturbation_Var(int64_t antal,const long double *dxr,const long double *dxi, long double Dr, long double Di, long double D0r, long double D0i,double &test1, double &test2, int m_nBailout2, int64_t m_nMaxIter,const double *m_db_z,bool &bGlitch,int m_nPower,const int *m_pnExpConsts, const bool m_bNoGlitchDetection)
{
  long double yr, yi;
  bGlitch=FALSE;
  if(antal<m_nMaxIter && test1 <= m_nBailout2){
    for(;antal<m_nMaxIter;antal++){
      yr=dxr[antal]+Dr;
      yi=dxi[antal]+Di;
      test2=test1;
      test1 = g_real*yr*yr + g_imag*yi*yi;
      if(test1<m_db_z[antal]){
        bGlitch=TRUE;
        if (! m_bNoGlitchDetection)
          break;
      }
      complex<long double> y(yr, yi);
      complex<long double> X(dxr[antal],dxi[antal]);
      complex<long double> D(Dr,Di);
      complex<long double> D0(D0r,D0i);
      complex<long double> c(m_pnExpConsts[0],0);
      int nXExp=m_nPower-2, nDExp=2, ci=1;
      complex<long double> Dn = c*(X^(m_nPower-1))*D;
      while(nXExp){
        c.m_r = m_pnExpConsts[ci++];
        Dn += c*(X^nXExp)*(D^nDExp);
        nXExp--;
        nDExp++;
      }
      Di = Dn.m_i;
      Dr = Dn.m_r;
    }
  }
  return antal;
}

static int Perturbation_Var(int64_t antal,const long double *dxr,const long double *dxi, long double Dr, long double Di, long double D0r, long double D0i,double &test1, double &test2, int m_nBailout2, int64_t m_nMaxIter,const double *m_db_z,bool &bGlitch,int m_nPower,const int *m_pnExpConsts, long double &dr, long double &di, const bool m_bNoGlitchDetection)
{
  complex<long double> d(dr, di);
  long double yr, yi;
  bGlitch=FALSE;
  if(antal<m_nMaxIter && test1 <= m_nBailout2){
    for(;antal<m_nMaxIter;antal++){
```

```
        yr=dxr[antal]+Dr;
        yi=dxi[antal]+Di;
        test2=test1;
        test1 = g_real*yr*yr + g_imag*yi*yi;
        if(test1<m_db_z[antal]){
          bGlitch=TRUE;
          if (! m_bNoGlitchDetection)
            break;
        }
        complex<long double> y(yr, yi);
        d = m_nPower * d * (y ^ (m_nPower - 1)) + 1;
        complex<long double> X(dxr[antal],dxi[antal]);
        complex<long double> D(Dr,Di);
        complex<long double> D0(D0r,D0i);
        complex<long double> c(m_pnExpConsts[0],0);
        int nXExp=m_nPower-2, nDExp=2, ci=1;
        complex<long double> Dn = c*(X^(m_nPower-1))*D;
        while(nXExp){
          c.m_r = m_pnExpConsts[ci++];
          Dn += c*(X^nXExp)*(D^nDExp);
          nXExp--;
          nDExp++;
        }
        Di = Dn.m_i;
        Dr = Dn.m_r;
      }
    }
    dr = d.m_r;
    di = d.m_i;
    return antal;
  }

  void CFraktalSFT::MandelCalcLDBL()
  {
    m_bIterChanged = TRUE;
    long double dr = 0, di = 0;
    floatexp ldr = 0, ldi = 0;
    int64_t antal;
    int x, y, w, h;
    bool derivatives = GetDerivatives();

    int64_t nMaxIter = (m_nGlitchIter<m_nMaxIter ? m_nGlitchIter : m_nMaxIter);
    while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
      int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
      if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
        SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, w, h);
        continue;
      }
      if (GuessPixel(x, y, w, h))
        continue;

      // Series approximation
      floatexp D0r = 0;
      floatexp D0i = 0;
      floatexp daa0 = 1;
      floatexp dab0 = 0;
      floatexp dba0 = 0;
```

```
    floatexp dbb0 = 1;
    GetPixelCoordinates(x, y, D0r, D0i, daa0, dab0, dba0, dbb0);

    floatexp TDnr;
    floatexp TDni;
    floatexp dxa1, dxb1, dya1, dyb1;
    DoApproximation(antal, D0r, D0i, TDnr, TDni, dxa1, dxb1, dya1, dyb1);
    floatexp TDDnr = dxa1;
    floatexp TDDni = dya1;

    double test1 = 0, test2 = 0;
    bool bGlitch = false;

    long double dbD0r = D0r.toLongDouble();
    long double dbD0i = D0i.toLongDouble();
    long double Dr = TDnr.toLongDouble();
    long double Di = TDni.toLongDouble();
    dr = TDDnr.toLongDouble();
    di = TDDni.toLongDouble();

    if (m_nFractalType == 0 && m_nPower > 10) // FIXME matrix derivatives
    { // FIXME check this is still ok around long double vs scaled double zoom
      threshold e600
      antal = derivatives
        ? Perturbation_Var(antal, m_ldxr, m_ldxi, Dr, Di, dbD0r, dbD0i, test1,
            test2, m_nBailout2, nMaxIter, m_db_z, bGlitch, m_nPower,
            m_pnExpConsts, dr, di, m_bNoGlitchDetection)
        : Perturbation_Var(antal, m_ldxr, m_ldxi, Dr, Di, dbD0r, dbD0i, test1,
            test2, m_nBailout2, nMaxIter, m_db_z, bGlitch, m_nPower,
            m_pnExpConsts, m_bNoGlitchDetection)
        ;
      long double pixel_spacing = m_lPixelSpacing;
      dr *= pixel_spacing;
      di *= pixel_spacing;
      ldr = dr;
      ldi = di;
    }
    else if (m_nScalingOffsetL)
    {
      Dr = TDnr.toLongDouble(m_nScalingOffsetL);
      Di = TDni.toLongDouble(m_nScalingOffsetL);
      dbD0r = D0r.toLongDouble(m_nScalingOffsetL);
      dbD0i = D0i.toLongDouble(m_nScalingOffsetL);
      ldr = TDDnr;
      ldi = TDDni;
      ldr *= m_fPixelSpacing;
      ldi *= m_fPixelSpacing;
      bool ok = derivatives
        ? perturbation(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, antal,
            test1, test2, bGlitch, m_nBailout2, nMaxIter, m_bNoGlitchDetection
            , g_real, g_imag, g_FactorAR, g_FactorAI, Dr, Di, dbD0r, dbD0i,
            ldr, ldi, (floatexp)(m_epsilon), m_fPixelSpacing, daa0, dab0, dba0
            , dbb0, m_nScalingL, 1 / m_nScalingL)
        : perturbation(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, antal,
            test1, test2, bGlitch, m_nBailout2, nMaxIter, m_bNoGlitchDetection
            , g_real, g_imag, g_FactorAR, g_FactorAI, Dr, Di, dbD0r, dbD0i,
            m_nScalingL, 1 / m_nScalingL)
```

```
        ;
170     assert(ok && "perturbation_long_double_scaled");
      }
      else
      {
        long double daa = daa0.toLongDouble();
175     long double dab = dab0.toLongDouble();
        long double dba = dba0.toLongDouble();
        long double dbb = dbb0.toLongDouble();
        dr *= m_lPixelSpacing;
        di *= m_lPixelSpacing;
180     bool ok = derivatives
          ? perturbation(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, antal, ↙
              ↳ test1, test2, bGlitch, m_nBailout2, nMaxIter, m_bNoGlitchDetection↙
              ↳ , g_real, g_imag, g_FactorAR, g_FactorAI, Dr, Di, dbD0r, dbD0i, dr↙
              ↳ , di, (long double)(m_epsilon), m_lPixelSpacing, daa, dab, dba, ↙
              ↳ dbb)
          : perturbation(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, antal, ↙
              ↳ test1, test2, bGlitch, m_nBailout2, nMaxIter, m_bNoGlitchDetection↙
              ↳ , g_real, g_imag, g_FactorAR, g_FactorAI, Dr, Di, dbD0r, dbD0i)
          ;
        assert(ok && "perturbation_long_double");
185     ldr = dr;
        ldi = di;
      }
      complex<double> z((double(Dr)), (double(Di)));
      complex<double> dc((double(ldr)), (double(ldi)));
190   complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;
      OutputIterationData(x, y, w, h, bGlitch, antal, test1, test2, de);
      InterlockedIncrement((LPLONG)&m_nDone);
      OutputPixelData(x, y, w, h, bGlitch);
    }
195 }
```

78 fraktal_sft/long_double_reference.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen
5

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20 #include "fraktal_sft.h"
#include <float.h>
```

```
#include "complex.h"

#include "../formula/formula.h"

extern double g_real;
extern double g_imag;
extern double g_FactorAR;
extern double g_FactorAI;

DWORD WINAPI ThMC2(MC2 *pMC);
DWORD WINAPI ThMC(MC *pMC);
extern double g_SeedR;
extern double g_SeedI;

#include "../common/barrier.h"

struct mcthread_common
{
        barrier *barrier;
        mpfr_t xr, xi, xrn, xin, xrn1, xin1, xrxid, xrxid1, sr, si, cr, ci;
        long double *m_ldxr, *m_ldxi;
        double *m_db_z, *terminate, *glitch_threshold;
        int64_t *m_nMaxIter, *m_nGlitchIter, *nMaxIter, *m_nRDone;
        int64_t *antal;
        double *test1;
        double *test2;
        volatile bool *stop;
        long double dr, di;
};

struct mcthread
{
        int nType;
        HANDLE hDone;
        mcthread_common *common;
};

static DWORD WINAPI mcthreadfunc(mcthread *p0)
{
        SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
        SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
        int64_t antal = 0;
        double test1 = 0;
        double test2 = 0;
        bool stored = false;
        double old_absval = 0;
        double abs_val = 0;

        mcthread_common *p = p0->common;
        long double dr = p->dr;
        long double di = p->di;
        const double glitch_threshold = *p->glitch_threshold;
        int64_t i = 0;
        switch (p0->nType)
        {
                case 0:
                {
```

```
                        for (i = 0; i < *p->nMaxIter; i++)
80                      {
                                mpfr_sub(p->xrn1, p->sr, p->si, MPFR_RNDN);
                                mpfr_add(p->xrn, p->cr, p->xrn1, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
                                mpfr_set(p->xr, p->xrn, MPFR_RNDN);
85                              mpfr_sqr(p->sr, p->xrn, MPFR_RNDN);
                                p->m_ldxr[i] = mpfr_get_ld(p->xrn, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
                        }
                }
90              break;
                case 1:
                {
                        for (i = 0; i < *p->nMaxIter; i++)
                        {
95                              mpfr_add(p->xin, p->sr, p->si, MPFR_RNDN);
                                mpfr_sub(p->xin1, p->ci, p->xin, MPFR_RNDN);
                                mpfr_add(p->xin, p->xin1, p->xrxid, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
                                mpfr_set(p->xi, p->xin, MPFR_RNDN);
100                             mpfr_sqr(p->si, p->xin, MPFR_RNDN);
                                p->m_ldxi[i] = mpfr_get_ld(p->xin, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
                        }
                }
105             break;
                case 2:
                {
                        for (i = 0; i < *p->nMaxIter; i++)
                        {
110                             if (i > 0)
                                {
                                        const long double lr = p->m_ldxr[i-1];
                                        const long double li = p->m_ldxi[i-1];
                                        long double drn = 2 * (lr * dr - li * di↙
                                            ↳ ) + 1;
115                                     long double din = 2 * (lr * di + li * dr↙
                                            ↳ );
                                        dr = drn;
                                        di = din;
                                        old_absval = abs_val;
                                        abs_val = g_real * lr * lr + g_imag * li↙
                                            ↳  * li;
120                                     p->m_db_z[i-1] = abs_val * ↙
                                            ↳ glitch_threshold;
                                        if (abs_val >= 4)
                                        {
                                                if (*p->terminate == 4 && !↙
                                                    ↳ stored)
                                                {
125                                                     stored = true;
                                                        antal = i;
                                                        test1 = abs_val;
                                                        test2 = old_absval;
                                                }
130                                     }
```

```
                                        if (abs_val >= *p->terminate){
                                                if (*p->terminate > 4 && !stored
                                                    ↪ )
                                                {
                                                        stored = true;
135                                                     antal = i;
                                                        test1 = abs_val;
                                                        test2 = old_absval;
                                                }
                                                if (*p->nMaxIter == *p->
                                                    ↪ m_nMaxIter)
140                                             {
                                                        *p->nMaxIter = i-1 + 3;
                                                        if (*p->nMaxIter > *p->
                                                            ↪ m_nMaxIter)
                                                                *p->nMaxIter = *
                                                                    ↪ p->
                                                                    ↪ m_nMaxIter
                                                                    ↪ ;
                                                        *p->m_nGlitchIter = *p->
                                                            ↪ nMaxIter;
145                                             }
                                        }
                                        (*p->m_nRDone)++;
                                }
                                if (p->barrier->wait(p->stop)) break;
150                             mpfr_add(p->xrxid1, p->xrn, p->xin, MPFR_RNDN);
                                mpfr_sqr(p->xrxid, p->xrxid1, MPFR_RNDN);
                                if (p->barrier->wait(p->stop)) break;
                        }
                }
155             break;
        }
        if (p->barrier->wait(p->stop))
        {
                SetEvent(p0->hDone);
160             return 0;
        }
        if (p0->nType == 2)
        {
                if (i > 0)
165             {
                        const long double lr = p->m_ldxr[i-1];
                        const long double li = p->m_ldxi[i-1];
                        long double drn = 2 * (lr * dr - li * di) + 1;
                        long double din = 2 * (lr * di + li * dr);
170                     dr = drn;
                        di = din;
                        old_absval = abs_val;
                        abs_val = g_real * lr * lr + g_imag * li * li;
                        p->m_db_z[i-1] = abs_val * glitch_threshold;
175                     if (abs_val >= 4)
                        {
                                if (*p->terminate == 4 && !stored)
                                {
                                        stored = true;
180                                     antal = i;
```

```
                                        test1 = abs_val;
                                        test2 = old_absval;
                                }
                        }
                        if (abs_val >= *p->terminate){
                                if (*p->terminate > 4 && !stored)
                                {
                                        stored = true;
                                        antal = i;
                                        test1 = abs_val;
                                        test2 = old_absval;
                                }
                                if (*p->nMaxIter == *p->m_nMaxIter)
                                {
                                        *p->nMaxIter = i-1 + 3;
                                        if (*p->nMaxIter > *p->m_nMaxIter)
                                                *p->nMaxIter = *p->m_nMaxIter;
                                        *p->m_nGlitchIter = *p->nMaxIter;
                                }
                        }
                        (*p->m_nRDone)++;
                }
                p->dr = dr;
                p->di = di;
                const long double xr = mpfr_get_ld(p->xr, MPFR_RNDN);
                const long double xi = mpfr_get_ld(p->xi, MPFR_RNDN);
                for (; i < *p->nMaxIter && !*p->stop; i++)
                {
                        p->m_ldxr[i] = xr;
                        p->m_ldxi[i] = xi;
                }
                *p->antal = antal;
                *p->test1 = test1;
                *p->test2 = test2;
        }
        SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_END);
        SetEvent(p0->hDone);
        mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
        return 0;
}

void CFraktalSFT::CalculateReferenceLDBL()
{
        const bool derivatives = GetDerivatives();
        Precision prec(m_rref.m_f.precision());

        int64_t i;
        if (m_ldxr)
                delete[] m_ldxr;
        m_ldxr = new long double[m_nMaxIter];
        if (m_ldxi)
                delete[] m_ldxi;
        m_ldxi = new long double[m_nMaxIter];
        if (m_db_z)
                delete[] m_db_z;
        m_db_z = new double [m_nMaxIter];
```

```
        int64_t antal = 0;
        double test1 = 0;
        double test2 = 0;
        double xxr = 0, xxi = 0;

        long double dr = 1, di = 0;
        floatexp ldr = 1, ldi = 0;

        double terminate = SMOOTH_BAILOUT*SMOOTH_BAILOUT;
        m_nGlitchIter = m_nMaxIter + 1;
        int64_t nMaxIter = m_nMaxIter;

        if (m_nFractalType == 0 && m_nPower == 2 && GetThreadedReference()){ // ↙
            ↳ FIXME matrix derivatives, option to disable derivatives
                double glitch_threshold = 0.0000001;
                if (GetGlitchLowTolerance()) {
                        glitch_threshold = sqrt(glitch_threshold);
                }

                mcthread mc[3];
                barrier barrier(3);
                HANDLE hDone[3];

                mcthread_common co;
          co.barrier = &barrier;
                mp_bitcnt_t bits = mpfr_get_prec(m_rref.m_f.backend().data());
                mpfr_init2(co.xr, bits);
                mpfr_init2(co.xi, bits);
                mpfr_init2(co.xrn, bits);
                mpfr_init2(co.xin, bits);
                mpfr_init2(co.xrn1, bits);
                mpfr_init2(co.xin1, bits);
                mpfr_init2(co.xrxid, bits);
                mpfr_init2(co.xrxid1, bits);
                mpfr_init2(co.sr, bits);
                mpfr_init2(co.si, bits);
                mpfr_init2(co.cr, bits);
                mpfr_init2(co.ci, bits);
                mpfr_set(co.cr, m_rref.m_f.backend().data(), MPFR_RNDN);
                mpfr_set(co.ci, m_iref.m_f.backend().data(), MPFR_RNDN);
                mpfr_set_d(co.xr, g_SeedR, MPFR_RNDN);
                mpfr_set_d(co.xi, g_SeedI, MPFR_RNDN);
                mpfr_sqr(co.sr, co.xr, MPFR_RNDN);
                mpfr_sqr(co.si, co.xi, MPFR_RNDN);
                mpfr_set_d(co.xrxid, 0, MPFR_RNDN);
                co.m_ldxr = m_ldxr;
                co.m_ldxi = m_ldxi;
                co.m_db_z = m_db_z;
                co.terminate = &terminate;
                co.glitch_threshold = &glitch_threshold;
                co.m_nMaxIter = &m_nMaxIter;
                co.m_nGlitchIter = &m_nGlitchIter;
                co.nMaxIter = &nMaxIter;
                co.m_nRDone = &m_nRDone;
                co.stop = &m_bStop;
                co.antal = &antal;
                co.test1 = &test1;
```

```
                co.test2 = &test2;
                co.dr = dr;
                co.di = di;
                // spawn threads
                for (i = 0; i < 3; i++)
                {
                        mc[i].nType = i;
                        hDone[i] = mc[i].hDone = CreateEvent(NULL, 0, 0, NULL);
                        mc[i].common = &co;
                        HANDLE hThread = CreateThread(NULL, 0, (↙
                            ↳ LPTHREAD_START_ROUTINE) mcthreadfunc, (LPVOID)&mc[↙
                            ↳ i], 0, NULL);
                        CloseHandle(hThread);
                }
                // wait for completion
                WaitForMultipleObjects(3, hDone, TRUE, INFINITE);
                for (i = 0; i < 3; i++){
                        CloseHandle(hDone[i]);
                }

                mpfr_clear(co.xr);
                mpfr_clear(co.xi);
                mpfr_clear(co.xrn);
                mpfr_clear(co.xin);
                mpfr_clear(co.xrn1);
                mpfr_clear(co.xin1);
                mpfr_clear(co.xrxid);
                mpfr_clear(co.xrxid1);
                mpfr_clear(co.sr);
                mpfr_clear(co.si);
                mpfr_clear(co.cr);
                mpfr_clear(co.ci);
                dr = co.dr;
                di = co.di;
                long double pixel_spacing = m_lPixelSpacing;
                dr *= pixel_spacing;
                di *= pixel_spacing;
                ldr = dr;
                ldi = di;

        }
        else if (m_nFractalType == 0 && m_nPower > 10) // FIXME matrix ↙
            ↳ derivatives, option to disable derivatives
        {

                bool stored = false;
                double old_absval = 0;
                double abs_val = 0;
                CFixedFloat xr = g_SeedR, xi = g_SeedI;
                double threashold = 0.0001;
                for (i = 7; i <= m_nPower; i += 2)
                        threashold *= 10;
    if (GetGlitchLowTolerance()) {
                        threashold = sqrt(threashold);
                }
                if (threashold>.5)
                        threashold = .5;
```

```
                complex<long double> d(1.0, 0.0);
                for (i = 0; i<nMaxIter && !m_bStop; i++){
350                     complex<CFixedFloat> X(xr, xi), r(m_rref, m_iref);
                        complex<CFixedFloat> Xn = (X^m_nPower) + r;
                        floatexp xrf; xrf = xr;
                        floatexp xif; xif = xi;
                        complex<long double> x(xrf.toLongDouble(), xif.↙
                            ↳ toLongDouble());
355                     d = m_nPower * d * (x ^ (m_nPower - 1)) + 1;
                        xr = Xn.m_r;
                        xi = Xn.m_i;
                        m_ldxr[i] = mpfr_get_ld(xr.m_f.backend().data(), ↙
                            ↳ MPFR_RNDN);
                        m_ldxi[i] = mpfr_get_ld(xi.m_f.backend().data(), ↙
                            ↳ MPFR_RNDN);
360                     old_absval = abs_val;
                        abs_val = g_real * m_ldxr[i] * m_ldxr[i] + g_imag * ↙
                            ↳ m_ldxi[i] * m_ldxi[i];
                        m_db_z[i] = abs_val*threashold;
                        if (abs_val >= 4)
                        {
365                             if (terminate == 4 && !stored)
                                {
                                        stored = true;
                                        antal = i;
                                        test1 = abs_val;
370                                     test2 = old_absval;
                                }
                        }
                        if (abs_val >= terminate){
                                if (terminate > 4 && !stored)
375                             {
                                        stored = true;
                                        antal = i;
                                        test1 = abs_val;
                                        test2 = old_absval;
380                             }
                                if (nMaxIter == m_nMaxIter){
                                        nMaxIter = i + 3;
                                        if (nMaxIter>m_nMaxIter)
                                                nMaxIter = m_nMaxIter;
385                                     m_nGlitchIter = nMaxIter;
                                }
                        }
                        m_nRDone++;
                }
390             dr = d.m_r;
                di = d.m_i;
                long double pixel_spacing = m_lPixelSpacing;
                dr *= pixel_spacing;
                di *= pixel_spacing;
395             ldr = dr;
                ldi = di;

        }
        else if (m_nScalingOffset && scaling_supported(m_nFractalType, m_nPower,↙
            ↳ derivatives))
```

```
400         {

                    floatexp _x, _y, daa, dab, dba, dbb;
                    GetPixelCoordinates(g_nAddRefX, g_nAddRefY, _x, _y, daa, dab, ↙
                        ↳ dba, dbb);
                    ldr *= m_fPixelSpacing;
405                 ldi *= m_fPixelSpacing;
                    bool ok = derivatives
                      ? reference(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, ↙
                          ↳ m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, m_rref, ↙
                          ↳ m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI, ↙
                          ↳ terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                          ↳ antal, test1, test2, xxr, xxi, ldr, ldi, daa, dab, dba, ↙
                          ↳ dbb)
                      : reference(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, ↙
                          ↳ m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, m_rref, ↙
                          ↳ m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI, ↙
                          ↳ terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                          ↳ antal, test1, test2, xxr, xxi)
                      ;
410                 assert(ok && "reference_scaled_long_double");

        }
        else
        {
415
                    floatexp _x, _y, daa, dab, dba, dbb;
                    GetPixelCoordinates(g_nAddRefX, g_nAddRefY, _x, _y, daa, dab, ↙
                        ↳ dba, dbb);
                    long double ldaa = daa.toLongDouble();
                    long double ldab = dab.toLongDouble();
420                 long double ldba = dba.toLongDouble();
                    long double ldbb = dbb.toLongDouble();
                    dr *= m_lPixelSpacing;
                    di *= m_lPixelSpacing;
                    bool ok = derivatives
425                   ? reference(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, ↙
                          ↳ m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, m_rref, ↙
                          ↳ m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI, ↙
                          ↳ terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                          ↳ antal, test1, test2, xxr, xxi, dr, di, ldaa, ldab, ldba,↙
                          ↳  ldbb)
                      : reference(m_nFractalType, m_nPower, m_ldxr, m_ldxi, m_db_z, ↙
                          ↳ m_bStop, m_nRDone, m_nGlitchIter, m_nMaxIter, m_rref, ↙
                          ↳ m_iref, g_SeedR, g_SeedI, g_FactorAR, g_FactorAI, ↙
                          ↳ terminate, g_real, g_imag, GetGlitchLowTolerance(), ↙
                          ↳ antal, test1, test2, xxr, xxi)
                      ;
          assert(ok && "reference_long_double");
                    ldr = dr;
430                 ldi = di;

        }

        complex<double> z((double(xxr)), (double(xxi)));
435     complex<double> dc((double(ldr)), (double(ldi)));
        complex<double> de = derivatives ? abs(z) * log(abs(z)) / dc : 0;
```

```
        if (0 <= g_nAddRefX && g_nAddRefX < m_nX && 0 <= g_nAddRefY && ↙
            ↳ g_nAddRefY < m_nY)
                OutputIterationData(g_nAddRefX, g_nAddRefY, 1, 1, false, antal ?↙
                    ↳  antal + 1 : m_nMaxIter, test1, test2, de);
}
```

79 fraktal_sft/main_color.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "main.h"
#include "main_color.h"
#include "fraktal_sft.h"
#include "resource.h"
#include "../common/tooltip.h"
#include "../common/FolderBrowser.h"
#include "../common/StringVector.h"
#include "../common/getimage.h"
#include "listbox.h"

#include <commctrl.h>

static char g_szTmpFile[MAX_PATH];

static RECT g_rShow;
static COLOR14 g_colCopy={0};
static BOOL g_bCaptureMouse=FALSE;
static BOOL g_bInitColorDialog=FALSE;
static CListBoxEdit *g_pWaves=NULL;
static BOOL g_AutoColour = TRUE;
static int g_AutoUpdate = 0;

extern int WINAPI ColorProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        static COLORREF colCust[16]={0};
        if(uMsg==WM_SHOWWINDOW && wParam){
                GetTempPath(sizeof(g_szTmpFile),g_szTmpFile);
                GetTempFileName(g_szTmpFile,"KFR",TRUE,g_szTmpFile);
                g_SFT.SaveFile(g_szTmpFile, true);
```

```
        }
        if(uMsg==WM_INITDIALOG)
        {
                SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,0);
                SendDlgItemMessage(hWnd, IDC_AUTOCOLOUR, BM_SETCHECK, ↙
                    ↳ g_AutoColour, 0);
                DragAcceptFiles(hWnd, TRUE);
        }
        if(uMsg==WM_INITDIALOG || uMsg==WM_USER+99 || (uMsg==WM_SHOWWINDOW && ↙
            ↳ wParam) || (uMsg==WM_COMMAND && (wParam==IDC_CHECK6 || wParam==↙
            ↳ IDC_CHECK7))){
                std::string szTexture;
                double nRatio, nPower;
                int nMerge;
                BOOL bTexture = g_SFT.GetTexture(nRatio,nPower,nMerge,szTexture)↙
                    ↳ ;
                SetDlgItemText(hWnd,IDC_EDIT29,szTexture.c_str());
                SendDlgItemMessage(hWnd,IDC_CHECK8,BM_SETCHECK,bTexture,0);
                SetDlgItemFloat(hWnd,IDC_EDIT26,nPower);
                SetDlgItemInt(hWnd,IDC_EDIT27,nRatio*100,FALSE);
                if(!g_pWaves){
                        HWND hwnds[2]={GetDlgItem(hWnd,IDC_EDIT23),GetDlgItem(↙
                            ↳ hWnd,IDC_EDIT25)};
                        g_pWaves = new CListBoxEdit(GetDlgItem(hWnd,IDC_BUTTON26↙
                            ↳ ), GetDlgItem(hWnd,IDC_BUTTON27), GetDlgItem(hWnd,↙
                            ↳ IDC_BUTTON28), GetDlgItem(hWnd,IDC_EDIT24), ↙
                            ↳ GetDlgItem(hWnd,IDC_LIST6),hwnds, 2);

                        SendDlgItemMessage(hWnd,IDC_COMBO4,CB_ADDSTRING,0,(↙
                            ↳ LPARAM)"Hue");
                        SendDlgItemMessage(hWnd,IDC_COMBO4,CB_ADDSTRING,0,(↙
                            ↳ LPARAM)"Saturation");
                        SendDlgItemMessage(hWnd,IDC_COMBO4,CB_ADDSTRING,0,(↙
                            ↳ LPARAM)"Brightness");
                }
                if(uMsg==WM_COMMAND && (wParam==IDC_CHECK6 || wParam==IDC_CHECK7↙
                    ↳ )){
                        g_SFT.SetMW(SendDlgItemMessage(hWnd,IDC_CHECK6,↙
                            ↳ BM_GETCHECK,0,0),SendDlgItemMessage(hWnd,↙
                            ↳ IDC_CHECK7,BM_GETCHECK,0,0));
                        g_AutoUpdate++;
                        SendMessage(hWnd,WM_COMMAND,IDOK,0);
                        g_AutoUpdate--;
                }
                else{
                        BOOL bBlend=FALSE;
                        SendDlgItemMessage(hWnd,IDC_CHECK6,BM_SETCHECK,g_SFT.↙
                            ↳ GetMW(&bBlend),0);
                        SendDlgItemMessage(hWnd,IDC_CHECK7,BM_SETCHECK,bBlend,0)↙
                            ↳ ;
                }
                EnableWindow(GetDlgItem(hWnd,IDC_COMBO4),g_SFT.GetMW());
                EnableWindow(GetDlgItem(hWnd,IDC_EDIT23),g_SFT.GetMW());
                EnableWindow(GetDlgItem(hWnd,IDC_CHECK7),g_SFT.GetMW());
                EnableWindow(GetDlgItem(hWnd,IDC_BUTTON29),g_SFT.GetMW());
```

```
90                  EnableWindow(GetDlgItem(hWnd,IDC_BUTTON30),g_SFT.GetMW());
                    EnableWindow(GetDlgItem(hWnd,IDC_EDIT25),g_SFT.GetMW());
                    EnableWindow(GetDlgItem(hWnd,IDC_LIST6),g_SFT.GetMW());
                    EnableWindow(GetDlgItem(hWnd,IDC_BUTTON26),g_SFT.GetMW());
                    EnableWindow(GetDlgItem(hWnd,IDC_BUTTON27),g_SFT.GetMW());
95                  EnableWindow(GetDlgItem(hWnd,IDC_BUTTON28),g_SFT.GetMW());
            }
            else if(HIWORD(wParam)==LBN_SELCHANGE && LOWORD(wParam)==IDC_LIST6){
                    char szTmp[256];
                    int i = SendDlgItemMessage(hWnd,IDC_LIST6,LB_GETCURSEL,0,0);
100                 if(i!=-1){
                            SendDlgItemMessage(hWnd,IDC_LIST6,LB_GETTEXT,i,(LPARAM)↙
                                ↳ szTmp);
                            if(*szTmp=='H')
                                    SendDlgItemMessage(hWnd,IDC_COMBO4,CB_SETCURSEL↙
                                        ↳ ,0,0);
                            else if(*szTmp=='S')
105                                 SendDlgItemMessage(hWnd,IDC_COMBO4,CB_SETCURSEL↙
                                        ↳ ,1,0);
                            else if(*szTmp=='B')
                                    SendDlgItemMessage(hWnd,IDC_COMBO4,CB_SETCURSEL↙
                                        ↳ ,2,0);
                    }
            }
110         else if(HIWORD(wParam)==CBN_SELCHANGE && LOWORD(wParam)==IDC_COMBO4){
                    int i = SendDlgItemMessage(hWnd,IDC_COMBO4,CB_GETCURSEL,0,0);
                    if(i==0)
                            SetDlgItemText(hWnd,IDC_EDIT24,"H");
                    else if(i==1)
115                         SetDlgItemText(hWnd,IDC_EDIT24,"S");
                    else if(i==2)
                            SetDlgItemText(hWnd,IDC_EDIT24,"B");
            }
            if(uMsg==WM_INITDIALOG || uMsg==WM_USER+99 || (uMsg==WM_SHOWWINDOW && ↙
                ↳ wParam)){
120                 g_bInitColorDialog=FALSE;

                    SendDlgItemMessage(hWnd,IDC_LIST6,LB_RESETCONTENT,0,0);
                    int i;
                    char szTmp[256];
125                 for(i=0;i<g_SFT.GetMWCount();i++){
                            int nPeriod, nStart, nType;
                            g_SFT.GetMW(i,nPeriod, nStart, nType);
                            if(nType==0)
                                    strcpy(szTmp,"H\t");
130                         else if(nType==1)
                                    strcpy(szTmp,"S\t");
                            else if(nType==2)
                                    strcpy(szTmp,"B\t");
                            itoa(nPeriod,szTmp+strlen(szTmp),10);
135                         strcat(szTmp,"\t");
                            itoa(nStart,szTmp+strlen(szTmp),10);
                            SendDlgItemMessage(hWnd,IDC_LIST6,LB_ADDSTRING,0,(LPARAM↙
                                ↳ )szTmp);
                    }

140                 SetWindowText(hWnd,"Number of Colors");
```

```
                SetDlgItemInt(hWnd,IDC_EDIT1,g_SFT.GetNumOfColors(),FALSE);
                SetDlgItemInt(hWnd,IDC_EDIT2,g_SFT.GetSeed(),FALSE);
                sprintf(szTmp,"%f",g_SFT.GetIterDiv());
                int e = strlen(szTmp);
145             while(e && szTmp[e-1]=='0')
                        e--;
                szTmp[e]=0;
                SetDlgItemText(hWnd,IDC_EDIT3,szTmp);
                SetDlgItemInt(hWnd,IDC_EDIT12,g_SFT.GetColorOffset(),FALSE);
150             SendDlgItemMessage(hWnd,IDC_SPIN1,UDM_SETRANGE,0,MAKELONG↙
                    ↪ (1024,2));
                SendDlgItemMessage(hWnd,IDC_SPIN2,UDM_SETRANGE,0,MAKELONG↙
                    ↪ (1000000,1));
                SendDlgItemMessage(hWnd,IDC_SPIN5,UDM_SETRANGE,0,MAKELONG↙
                    ↪ (1023,0));
                SendDlgItemMessage(hWnd,IDC_SPIN6,UDM_SETRANGE,0,MAKELONG(100,1)↙
                    ↪ );
                SendDlgItemMessage(hWnd,IDC_SPIN7,UDM_SETRANGE,0,MAKELONG(100,1)↙
                    ↪ );
155             SendDlgItemMessage(hWnd,IDC_SPIN8,UDM_SETRANGE,0,MAKELONG↙
                    ↪ (360,-360));
                int nSP, nSC, nSA;
                SendDlgItemMessage(hWnd,IDC_CHECK4,BM_SETCHECK,g_SFT.GetSlopes(↙
                    ↪ nSP,nSC,nSA),0);
                SetDlgItemInt(hWnd,IDC_EDIT20,nSP,FALSE);
                SetDlgItemInt(hWnd,IDC_EDIT21,nSC,FALSE);
160             SetDlgItemInt(hWnd,IDC_EDIT22,nSA,FALSE);
                EnableWindow(GetDlgItem(hWnd,IDC_EDIT20),SendDlgItemMessage(hWnd↙
                    ↪ ,IDC_CHECK4,BM_GETCHECK,0,0));
                EnableWindow(GetDlgItem(hWnd,IDC_EDIT21),SendDlgItemMessage(hWnd↙
                    ↪ ,IDC_CHECK4,BM_GETCHECK,0,0));
                EnableWindow(GetDlgItem(hWnd,IDC_EDIT22),SendDlgItemMessage(hWnd↙
                    ↪ ,IDC_CHECK4,BM_GETCHECK,0,0));

165             SendDlgItemMessage(hWnd,IDC_LIST1,WM_SETREDRAW,0,0);
                SendDlgItemMessage(hWnd,IDC_LIST1,LB_RESETCONTENT,0,0);
                for(i=0;i<g_SFT.GetNumOfColors();i++)
                        SendDlgItemMessage(hWnd,IDC_LIST1,LB_ADDSTRING,0,(LPARAM↙
                            ↪ )"");
                SendDlgItemMessage(hWnd,IDC_LIST1,WM_SETREDRAW,1,0);
170             HWND hw = GetDlgItem(hWnd,IDC_SHOW);
                if(hw){
    //          BOOL b = SetWindowLong(hw,GWL_WNDPROC,(LONG)ShowProc);
                        GetWindowRect(hw,&g_rShow);
                        g_rShow.right-=g_rShow.left;
175                     g_rShow.bottom-=g_rShow.top;
                        ScreenToClient(hWnd,(LPPOINT)&g_rShow);
                        DestroyWindow(hw);
                }
                SendDlgItemMessage(hWnd,IDC_CHECK2,BM_SETCHECK,g_SFT.↙
                    ↪ GetTransition(),0);
180             SendDlgItemMessage(hWnd,IDC_CHECK3,BM_SETCHECK,g_SFT.↙
                    ↪ GetITransition(),0);
                if(uMsg==WM_INITDIALOG || (uMsg==WM_SHOWWINDOW && wParam)){
                        SetDlgItemInt(hWnd,IDC_EDIT9,2,0);
                        SetDlgItemInt(hWnd,IDC_EDIT11,4,0);
                        SendDlgItemMessage(hWnd,IDC_SPIN3,UDM_SETRANGE,0,↙
```

```
                    ↪ MAKELONG(1,1024));
185                 SetDlgItemInt(hWnd,IDC_EDIT10,50,0);
                    SendDlgItemMessage(hWnd,IDC_SPIN4,UDM_SETRANGE,0,↙
                        ↪ MAKELONG(1,100));

                    if(SendDlgItemMessage(hWnd,IDC_COMBO1,CB_GETCOUNT,0,0)↙
                        ↪ ==0){
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Standard");
190                         SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Square root");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Cubic root");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Logarithm");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Stretched");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Distance (Linear)");
195                         SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"DE+Standard");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Distance (Logarithm)");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"Distance (Square Root)");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"LogLog");
                            SendDlgItemMessage(hWnd,IDC_COMBO1,CB_ADDSTRING↙
                                ↪ ,0,(LPARAM)"ATan");
200                 }

                    if(SendDlgItemMessage(hWnd,IDC_DIFFERENCES,CB_GETCOUNT↙
                        ↪ ,0,0)==0){
                            SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Traditional");
                            SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Forward 3x3");
205                         SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Central 3x3");
                            SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Diagonal 2x2");
                            SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Least Squares 2x2↙
                                ↪ ");
                            SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Least Squares 3x3↙
                                ↪ ");
                            SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Laplacian 3x3");
210                         SendDlgItemMessage(hWnd,IDC_DIFFERENCES,↙
                                ↪ CB_ADDSTRING,0,(LPARAM)"Analytic");
                    }

                    if(uMsg==WM_INITDIALOG){
                            SendDlgItemMessage(hWnd,IDC_RADIO4,BM_SETCHECK↙
                                ↪ ,1,0);
215                         SendDlgItemMessage(hWnd,IDC_RADIO5,BM_SETCHECK↙
```

```
                                        ↪ ,0,0);
                        }
                }
                else
                        InvalidateRect(hWnd,NULL,FALSE);
220             SendDlgItemMessage(hWnd,IDC_COMBO1,CB_SETCURSEL,g_SFT.↙
                    ↪ GetColorMethod(),0);
                SendDlgItemMessage(hWnd,IDC_DIFFERENCES,CB_SETCURSEL,g_SFT.↙
                    ↪ GetDifferences(),0);
                g_bInitColorDialog=TRUE;
                return 1;
        }
225     else if(uMsg==WM_PAINT){
                PAINTSTRUCT ps;
                BeginPaint(hWnd,&ps);
                int i;
                RECT rb={g_rShow.left,g_rShow.top,g_rShow.right+g_rShow.left,↙
                    ↪ g_rShow.top+g_rShow.bottom+1};
230             FillRect(ps.hdc,&rb,(HBRUSH)GetStockObject(WHITE_BRUSH));
                for(i=0;i<1024;i++){
                        RECT r={i*g_rShow.right/1024 + g_rShow.left,g_rShow.top↙
                            ↪ ,(i+1)*g_rShow.right/1024+g_rShow.left,g_rShow.top↙
                            ↪ +g_rShow.bottom/4};
                        COLOR14 c = g_SFT.GetColor(i);
                        HBRUSH br = CreateSolidBrush(RGB(c.b,c.g,c.r));
235                     FillRect(ps.hdc,&r,br);
                        DeleteObject(br);
                }
                HPEN pn = CreatePen(0,1,RGB(255,0,0));
                HPEN pnOld = (HPEN)SelectObject(ps.hdc,pn);
240             MoveToEx(ps.hdc,g_rShow.left,g_rShow.top+g_rShow.bottom/2 - ↙
                    ↪ g_SFT.GetColor(0).b*(g_rShow.bottom/4)/255,NULL);
                for(i=1;i<1024;i++)
                        LineTo(ps.hdc,i*g_rShow.right/1024 + g_rShow.left,↙
                            ↪ g_rShow.top+g_rShow.bottom/2 - g_SFT.GetColor(i).b↙
                            ↪ *(g_rShow.bottom/4)/255);
                SelectObject(ps.hdc,pnOld);
                DeleteObject(pn);
245
                pn = CreatePen(0,1,RGB(0,255,0));
                pnOld = (HPEN)SelectObject(ps.hdc,pn);
                MoveToEx(ps.hdc,g_rShow.left,g_rShow.top+3*g_rShow.bottom/4 - ↙
                    ↪ g_SFT.GetColor(0).g*(g_rShow.bottom/4)/255,NULL);
                for(i=1;i<1024;i++)
250                     LineTo(ps.hdc,i*g_rShow.right/1024 + g_rShow.left,↙
                            ↪ g_rShow.top+3*g_rShow.bottom/4 - g_SFT.GetColor(i)↙
                            ↪ .g*(g_rShow.bottom/4)/255);
                SelectObject(ps.hdc,pnOld);
                DeleteObject(pn);

                pn = CreatePen(0,1,RGB(0,0,255));
255             pnOld = (HPEN)SelectObject(ps.hdc,pn);
                MoveToEx(ps.hdc,g_rShow.left,g_rShow.top+g_rShow.bottom - g_SFT.↙
                    ↪ GetColor(0).r*(g_rShow.bottom/4)/255,NULL);
                for(i=1;i<1024;i++)
                        LineTo(ps.hdc,i*g_rShow.right/1024 + g_rShow.left,↙
                            ↪ g_rShow.top+g_rShow.bottom - g_SFT.GetColor(i).r*(↙
```

```
                                    ↳ g_rShow.bottom/4)/255);
                    SelectObject(ps.hdc,pnOld);
260                 DeleteObject(pn);

                    RECT r =
                      { g_rShow.right + g_rShow.left - 96
                            , g_rShow.top - 32
265                         , g_rShow.right + g_rShow.left
                            , g_rShow.top - 16
                            };
        COLOR14 c = g_SFT.GetInteriorColor();
        HBRUSH br = CreateSolidBrush(RGB(c.b, c.g, c.r));
270     FillRect(ps.hdc, &r, br);
        DeleteObject(br);

                    EndPaint(hWnd,&ps);
                    return 0;
275         }
    /*      else if(uMsg==WM_NOTIFY){
                    LPNMUPDOWN lpnmud = (LPNMUPDOWN)lParam;
                    if(lpnmud->hdr.code==UDN_DELTAPOS && (lpnmud->hdr.idFrom==↙
                        ↳ IDC_SPIN5 || lpnmud->hdr.idFrom==IDC_SPIN1 || lpnmud->hdr.↙
                        ↳ idFrom==IDC_SPIN2))
                            SendMessage(hWnd,WM_COMMAND,IDOK,0);
280         }
    */      else if(uMsg==WM_COMMAND){
                    EnableWindow(GetDlgItem(hWnd,IDC_EDIT20),SendDlgItemMessage(hWnd↙
                        ↳ ,IDC_CHECK4,BM_GETCHECK,0,0));
                    EnableWindow(GetDlgItem(hWnd,IDC_EDIT21),SendDlgItemMessage(hWnd↙
                        ↳ ,IDC_CHECK4,BM_GETCHECK,0,0));
                    EnableWindow(GetDlgItem(hWnd,IDC_EDIT22),SendDlgItemMessage(hWnd↙
                        ↳ ,IDC_CHECK4,BM_GETCHECK,0,0));
285                 if((g_bInitColorDialog && (HIWORD(wParam)==EN_UPDATE && (LOWORD(↙
                        ↳ wParam)==IDC_EDIT1 || LOWORD(wParam)==IDC_EDIT3 || LOWORD(↙
                        ↳ wParam)==IDC_EDIT12 || LOWORD(wParam)==IDC_EDIT20 || ↙
                        ↳ LOWORD(wParam)==IDC_EDIT21 || LOWORD(wParam)==IDC_EDIT22))↙
                        ↳ ) || (HIWORD(wParam)==CBN_SELCHANGE && (LOWORD(wParam)==↙
                        ↳ IDC_COMBO1 || LOWORD(wParam)==IDC_DIFFERENCES))){
                            g_bInitColorDialog=FALSE;
                            g_AutoUpdate++;
                            SendMessage(hWnd,WM_COMMAND,IDOK,0);
                            g_AutoUpdate--;
290                         return 0;
                    }
                    if(wParam==IDCANCEL){
                            ShowWindow(hWnd,SW_HIDE);
                    }
295                 else if(wParam==IDCLOSE){
                            g_SFT.OpenFile(g_szTmpFile,TRUE);
                            ShowWindow(hWnd,SW_HIDE);
                    }
                    else if(wParam==IDC_BUTTON23){
300                         char buffer[1024] = {0};
                            GetDlgItemText(hWnd,IDC_EDIT29,buffer,sizeof(buffer));
                            std::string szFile = buffer;
                            if(BrowseFile(hWnd,TRUE,"Select texture","Jpg\0*.jpg↙
                                ↳ \0\0",szFile)){
```

```
                                SetDlgItemText(hWnd,IDC_EDIT29,szFile.c_str());
305                             g_AutoUpdate++;
                                SendMessage(hWnd,WM_COMMAND,IDOK,0);
                                g_AutoUpdate--;
                        }
                }
310             else if (wParam == IDC_AUTOCOLOUR)
                {
                        g_AutoColour = SendDlgItemMessage(hWnd, IDC_AUTOCOLOUR, ↙
                            ↳ BM_GETCHECK, 0, 0);
                        if (g_AutoColour)
                        {
315                             g_AutoUpdate++;
                                SendMessage(hWnd, WM_COMMAND, IDOK, 0);
                                g_AutoUpdate--;
                        }
                }
320             else if(wParam==IDOK){
                        if (! g_AutoUpdate)
                                g_SFT.UndoStore();
                        char szTexture[1024];
                        double nPower;
325                     int nRatio;
                        GetDlgItemText(hWnd,IDC_EDIT29,szTexture,sizeof(↙
                            ↳ szTexture));
                        BOOL bTexture = SendDlgItemMessage(hWnd,IDC_CHECK8,↙
                            ↳ BM_GETCHECK,0,0);
                        nPower = GetDlgItemFloat(hWnd,IDC_EDIT26);
                        nRatio = GetDlgItemInt(hWnd,IDC_EDIT27,NULL,0);
330                     g_SFT.SetTexture(bTexture,(double)nRatio/100,nPower,100,↙
                            ↳ szTexture);

                        int nColors = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0);
                        if(nColors<=0 || nColors>1024){
                                nColors=1024;
335                             SetDlgItemInt(hWnd,IDC_EDIT1,nColors,FALSE);
                        }
                        g_SFT.ChangeNumOfColors(nColors);
                        char szDiv[256];
                        GetDlgItemText(hWnd,IDC_EDIT3,szDiv,sizeof(szDiv));
340                     double nDiv = atof(szDiv);
                        if(nDiv==0)
                                nDiv=1;
                        g_SFT.SetIterDiv(nDiv);
                        int nO = GetDlgItemInt(hWnd,IDC_EDIT12,NULL,0);
345                     g_SFT.SetColorOffset(nO);
                        if(nO!=g_SFT.GetColorOffset())
                                SetDlgItemInt(hWnd,IDC_EDIT12,g_SFT.↙
                                    ↳ GetColorOffset(),FALSE);
                        g_SFT.SetColorMethod(SendDlgItemMessage(hWnd,IDC_COMBO1,↙
                            ↳ CB_GETCURSEL,0,0));
                        g_SFT.SetDifferences(SendDlgItemMessage(hWnd,↙
                            ↳ IDC_DIFFERENCES,CB_GETCURSEL,0,0));
350                     g_SFT.SetTransition(SendDlgItemMessage(hWnd,IDC_CHECK2,↙
                            ↳ BM_GETCHECK,0,0));
                        g_SFT.SetITransition(SendDlgItemMessage(hWnd,IDC_CHECK3,↙
                            ↳ BM_GETCHECK,0,0));
```

```
                        if(nColors!=SendDlgItemMessage(hWnd,IDC_LIST1,↙
                            ↳ LB_GETCOUNT,0,0)){
                                SendDlgItemMessage(hWnd,IDC_LIST1,WM_SETREDRAW↙
                                    ↳ ,0,0);
                                SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                    ↳ LB_RESETCONTENT,0,0);
355                             int i;
                                for(i=0;i<nColors;i++)
                                        SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                            ↳ LB_ADDSTRING,0,(LPARAM)"");
                                SendDlgItemMessage(hWnd,IDC_LIST1,WM_SETREDRAW↙
                                    ↳ ,1,0);
                        }
360                     CStringVektor sv;
                        g_pWaves->GetStrings(&sv);
                        char *szSV = sv.ToText("\n");
                        CStringTable stMW(szSV,"\t","\n");
                        sv.DeleteToText(szSV);
365                     int i;
                        for(i=0;i<stMW.GetCount();i++){
                                int nType = 0;
                                if(*stMW[i][0]=='H')
                                        nType=0;
370                             else if(*stMW[i][0]=='S')
                                        nType=1;
                                else if(*stMW[i][0]=='B')
                                        nType=2;
                                int nPeriod = atoi(stMW[i][1]);
375                             int nStart = atoi(stMW[i][2]);
                                if(i==g_SFT.GetMWCount())
                                        g_SFT.AddMW(nPeriod,nStart,nType);
                                else
                                        g_SFT.UpdateMW(i,nPeriod,nStart,nType);
380                     }
                        while(i<g_SFT.GetMWCount())
                                g_SFT.DeleteMW(i);

                        g_SFT.SetSlopes(SendDlgItemMessage(hWnd,IDC_CHECK4,↙
                            ↳ BM_GETCHECK,0,0),GetDlgItemInt(hWnd,IDC_EDIT20,↙
                            ↳ NULL,FALSE),GetDlgItemInt(hWnd,IDC_EDIT21,NULL,↙
                            ↳ FALSE),GetDlgItemInt(hWnd,IDC_EDIT22,NULL,TRUE));
385
                        if (g_AutoColour || ((! g_AutoColour) && g_AutoUpdate ==↙
                            ↳ 0))
                        {
                                g_SFT.ApplyColors();
                                InvalidateRect(GetParent(hWnd),NULL,FALSE);
390                             UpdateWindow(GetParent(hWnd));
                        }
                        InvalidateRect(hWnd,NULL,FALSE);
                        g_bInitColorDialog=TRUE;
                }
395             else if(wParam==IDC_CHECK2 || wParam==IDC_CHECK3 || wParam==↙
                    ↳ IDC_CHECK4)
                {
                        g_AutoUpdate++;
                        SendMessage(hWnd,WM_COMMAND,IDOK,0);
```

```
                g_AutoUpdate--;
400         }
            else if(wParam==IDC_BUTTON1){
                std::string szFile;
                if(BrowseFile(hWnd,FALSE,"Save palette","Palette\0*.kfp↙
                    ↳ \0\0",szFile))
                    g_SFT.SaveFile(szFile, true);
405         }
            else if(wParam==IDC_BUTTON29){
                int val = GetDlgItemInt(hWnd,IDC_EDIT23,NULL,FALSE);
                val = MakePrime(val);
                SetDlgItemInt(hWnd,IDC_EDIT23,val,FALSE);
410         }
            else if(wParam==IDC_BUTTON30){
                srand(GetTickCount());
                int nColors = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0);
                if(nColors>1024)
415                 nColors=1024;
                int c, w;
                for(c=0;c<nColors;c++){
                    double nH = 0, nS = 0, nB = 0;
                    int nDR = 0, nDG = 0, nDB = 0;
420                 for(w=0;w<100;w++){
                        int nPeriod,nStart,nType;
                        if(!g_SFT.GetMW(w,nPeriod,nStart,nType))
                            break;

425                     double col = (double)c*(double)1024/(↙
                            ↳ double)nColors;
                        double g = sin((pi*((int)col)) / nPeriod↙
                            ↳ ) / 2 + .5;
                        if (nPeriod<0)
                            g = -(double)nPeriod / (double)↙
                                ↳ 100;
                        if (nType == 0){
430                         nH += g;
                            nDR++;
                        }
                        if (nType == 1){
                            nS += g;
435                         nDG++;
                        }
                        if (nType == 2){
                            nB += g;
                            nDB++;
440                     }
                    }
                    if (nDR)
                        nH /= nDR;
                    if (nDG)
445                     nS /= nDG;
                    if (nDB)
                        nB /= nDB;
                    COLOR14 cPos;
                    HSVToRGB(nH, nS, nB, cPos);
450                 g_SFT.SetKeyColor(cPos,c);
                }
```

```
                        if (g_AutoColour) g_SFT.ApplyColors();
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                        InvalidateRect(GetParent(hWnd),NULL,FALSE);
455                     InvalidateRect(hWnd,NULL,FALSE);
                }
                else if(wParam==IDC_BUTTON6)
                {
                        int nParts = g_SFT.GetNumOfColors();
460                     int nPParts = nParts;
                        int j=nParts-1;
                        nParts*=2;
                        if(nParts>1024)
                                return ColorProc(hWnd,uMsg,IDC_BUTTON7,lParam);
465                     g_SFT.ChangeNumOfColors(nParts);
                        int i;
                        for(i=nParts-1;i>=0;i-=2){
                                COLOR14 n;
                                if(j>=nPParts-1)
470                                     n=g_SFT.GetKeyColor(0);
                                else
                                        n = g_SFT.GetKeyColor(j+1);
                                COLOR14 c = g_SFT.GetKeyColor(j--);
                                n.r = (n.r+c.r)/2;
475                             n.g = (n.g+c.g)/2;
                                n.b = (n.b+c.b)/2;
                                g_SFT.SetKeyColor(n,i);
                                g_SFT.SetKeyColor(c,i-1);
                        }
480     //                  g_SFT.ChangeNumOfColors(nParts-1);
                        SendMessage(hWnd,WM_USER+99,0,0);
                        if (g_AutoColour) g_SFT.ApplyColors();
                }
                else if(wParam==IDC_BUTTON7){
485                     int i;
                        if (g_AutoColour) g_SFT.ApplyColors();
                        COLOR14 c[1024];
                        for(i=0;i<1024;i++)
                                c[i] = g_SFT.GetColor(i);
490                     g_SFT.ChangeNumOfColors(1024);
                        for(i=0;i<1024;i++)
                                g_SFT.SetKeyColor(c[i],i);
                        SendMessage(hWnd,WM_USER+99,0,0);
                        if (g_AutoColour) g_SFT.ApplyColors();
495             }
                else if(wParam==IDC_BUTTON17){
                        int i, nParts = g_SFT.GetNumOfColors();
                        COLOR14 n;
                        for(i=0;i<nParts;i++){
500                             n=g_SFT.GetKeyColor(i);
                                int col;
                                for(col=0;col<3;col++){
                                        unsigned char &c = (col==0?n.r:col==1?n.↙
                                            ↳ g:n.b);
                                        int diff = c - 128;
505                                     if(diff>0){
                                                diff = 255-c;
                                                diff = 10*diff/100;
```

```
                                        c+=diff;
                                    }
                                    else{
                                        diff=c;
                                        diff = 10*diff/100;
                                        c-=diff;
                                    }
                                }
                                g_SFT.SetKeyColor(n,i);
                            }
                            SendMessage(hWnd,WM_USER+99,0,0);
                            g_AutoUpdate++;
                            SendMessage(hWnd,WM_COMMAND,IDOK,0);
                            g_AutoUpdate--;
                        }
                        else if(wParam==IDC_BUTTON18){
                            int i, nParts = g_SFT.GetNumOfColors();
                            COLOR14 n;
                            for(i=0;i<nParts;i++){
                                n=g_SFT.GetKeyColor(i);
                                int col;
                                for(col=0;col<3;col++){
                                    unsigned char &c = (col==0?n.r:col==1?n.↙
                                        ↪ g:n.b);
                                    int diff = c - 128;
                                    if(diff>0){
                                        diff = c-128;
                                        diff = 10*diff/100;
                                        c-=diff;
                                    }
                                    else{
                                        diff=128-c;
                                        diff = 10*diff/100;
                                        c+=diff;
                                    }
                                }
                                g_SFT.SetKeyColor(n,i);
                            }
                            SendMessage(hWnd,WM_USER+99,0,0);
                            g_AutoUpdate++;
                            SendMessage(hWnd,WM_COMMAND,IDOK,0);
                            g_AutoUpdate--;
                        }
                        else if(wParam==IDC_BUTTON22){
                            std::string szFile;
                            if(!BrowseFile(hWnd,TRUE,"Open image","Image\0*.bmp;*.jp↙
                                ↪ *g;*.gif\0\0",szFile))
                                return 0;
                            HBITMAP bmBmp = GetImage(szFile.c_str());
                            HDC hDC = GetDC(NULL);
                            BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
                            int row;
                            GetDIBits(hDC,bmBmp,0,0,NULL,(LPBITMAPINFO)&bmi,↙
                                ↪ DIB_RGB_COLORS);
                            bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
                            bmi.biBitCount = 24;
                            row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >>↙
```

```
                        ↪ 3);
                    bmi.biSizeImage=row*bmi.biHeight;
                    BYTE *lpBits = new BYTE[bmi.biSizeImage];
                    GetDIBits(hDC,bmBmp,0,bmi.biHeight,lpBits,
565                             (LPBITMAPINFO)&bmi,DIB_RGB_COLORS);

                    double woffs = (double)bmi.biWidth/(double)1024;
                    int nParts = bmi.biWidth;
                    if(nParts>1024)
570                         nParts=1024;
                    else
                            woffs=1;
                    double hoffs = (double)bmi.biHeight/(double)nParts;
                    g_SFT.ChangeNumOfColors(nParts);
575                 int i;
                    for(i=0;i<nParts;i++){
                            int x = i*woffs;
                            int y = i*hoffs;
                            if(x>=bmi.biWidth)
580                             x=bmi.biWidth;
                            if(y>=bmi.biHeight)
                                    y=bmi.biHeight;
                            int nIndex = x*3 + (bmi.biHeight-1-y)*row;
                            COLOR14 c = g_SFT.GetKeyColor(i);
585                         c.r = lpBits[nIndex];
                            c.g = lpBits[nIndex+1];
                            c.b = lpBits[nIndex+2];
                            g_SFT.SetKeyColor(c,i);
                    }
590                 ReleaseDC(NULL,hDC);
                    DeleteObject(bmBmp);
                    delete[] lpBits;
                    if(nParts!=SendDlgItemMessage(hWnd,IDC_LIST1,LB_GETCOUNT↙
                        ↪ ,0,0)){
                            SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                ↪ LB_RESETCONTENT,0,0);
595                         int i;
                            for(i=0;i<nParts;i++)
                                    SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                        ↪ LB_ADDSTRING,0,(LPARAM)"");
                    }
                    InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
600             }
                else if(wParam==IDC_CHECK1){
                    if(SendDlgItemMessage(hWnd,IDC_CHECK1,BM_GETCHECK,0,0)){
                            SetCapture(hWnd);
                            SetTimer(hWnd,0,200,NULL);
605                 }
                    else{
                            ReleaseCapture();
                            KillTimer(hWnd,0);
                    }
610             }
                else if(wParam==IDC_BUTTON5){
                    std::string szFile;
                    if(BrowseFile(hWnd,TRUE,"Open palette","Palette\0*.kfp↙
                        ↪ \0\0",szFile)){
```

```
                                g_SFT.OpenFile(szFile, TRUE);
                                SendMessage(hWnd,WM_USER+99,0,0);
                                if (g_AutoColour) g_SFT.ApplyColors();
                                g_AutoUpdate++;
                                SendMessage(hWnd,WM_COMMAND,IDOK,0);
                                g_AutoUpdate--;
                        }
                }
                else if(wParam==IDC_BUTTON2){
                        int nColors = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0);
                        if(nColors>1024){
                                nColors=1024;
                                SetDlgItemInt(hWnd,IDC_EDIT1,nColors,FALSE);
                        }
                        int nSeed = GetDlgItemInt(hWnd,IDC_EDIT2,NULL,0);
                        g_SFT.GenerateColors(nColors,nSeed);
                        if (g_AutoColour) g_SFT.ApplyColors();
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                        InvalidateRect(GetParent(hWnd),NULL,FALSE);
                        if(nColors!=SendDlgItemMessage(hWnd,IDC_LIST1,↙
                            ↳ LB_GETCOUNT,0,0)){
                                SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                    ↳ LB_RESETCONTENT,0,0);
                                int i;
                                for(i=0;i<nColors;i++)
                                        SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                            ↳ LB_ADDSTRING,0,(LPARAM)"");
                        }
                        InvalidateRect(hWnd,NULL,FALSE);
                }
                else if(wParam==IDC_BUTTON12 || wParam==IDC_BUTTON14){
                        if(wParam==IDC_BUTTON14){
                                int nCols = g_SFT.GetNumOfColors()/2;
                                srand(GetTickCount());
                                int nVal = rand()%nCols;
                                if(nVal==0)
                                        nVal=1;
                                SetDlgItemInt(hWnd,IDC_EDIT11,nVal = MakePrime(↙
                                    ↳ nVal),FALSE);
                                SetDlgItemInt(hWnd,IDC_EDIT13,rand()%nVal,FALSE)↙
                                    ↳ ;
                                nVal = rand()%nCols;
                                if(nVal==0)
                                        nVal=1;
                                SetDlgItemInt(hWnd,IDC_EDIT14,nVal = MakePrime(↙
                                    ↳ nVal),FALSE);
                                SetDlgItemInt(hWnd,IDC_EDIT15,rand()%nVal,FALSE)↙
                                    ↳ ;
                                nVal = rand()%nCols;
                                if(nVal==0)
                                        nVal=1;
                                SetDlgItemInt(hWnd,IDC_EDIT16,nVal = MakePrime(↙
                                    ↳ nVal),FALSE);
                                SetDlgItemInt(hWnd,IDC_EDIT17,rand()%nVal,FALSE)↙
                                    ↳ ;
                                nVal = rand()%nCols;
                                if(nVal==0)
```

```
                                        nVal=1;
                                    SetDlgItemInt(hWnd,IDC_EDIT18,nVal = MakePrime(nVal),FALSE);
                                    SetDlgItemInt(hWnd,IDC_EDIT19,rand()%nVal,FALSE);
                                }
                                int nPeriod = GetDlgItemInt(hWnd,IDC_EDIT11,NULL,FALSE);
                                int nStart = GetDlgItemInt(hWnd,IDC_EDIT13,NULL,FALSE);
                                g_SFT.AddWave(2,nPeriod,nStart);
                                nPeriod = GetDlgItemInt(hWnd,IDC_EDIT14,NULL,FALSE);
                                nStart = GetDlgItemInt(hWnd,IDC_EDIT15,NULL,FALSE);
                                g_SFT.AddWave(1,nPeriod,nStart);
                                nPeriod = GetDlgItemInt(hWnd,IDC_EDIT16,NULL,FALSE);
                                nStart = GetDlgItemInt(hWnd,IDC_EDIT17,NULL,FALSE);
                                g_SFT.AddWave(0,nPeriod,nStart);
                                nPeriod = GetDlgItemInt(hWnd,IDC_EDIT18,NULL,FALSE);
                                nStart = GetDlgItemInt(hWnd,IDC_EDIT19,NULL,FALSE);
                                g_SFT.AddWave(3,nPeriod,nStart);

                                if(g_AutoColour) g_SFT.ApplyColors();
                                InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                                InvalidateRect(GetParent(hWnd),NULL,FALSE);
                                InvalidateRect(hWnd,NULL,FALSE);
                            }
                            else if(wParam==IDC_BUTTON16){
                                int nPeriod = MakePrime(GetDlgItemInt(hWnd,IDC_EDIT11,NULL,FALSE));
                                SetDlgItemInt(hWnd,IDC_EDIT11,nPeriod,FALSE);
                            }
                            else if(wParam==IDC_BUTTON19){
                                int nPeriod = MakePrime(GetDlgItemInt(hWnd,IDC_EDIT14,NULL,FALSE));
                                SetDlgItemInt(hWnd,IDC_EDIT14,nPeriod,FALSE);
                            }
                            else if(wParam==IDC_BUTTON20){
                                int nPeriod = MakePrime(GetDlgItemInt(hWnd,IDC_EDIT16,NULL,FALSE));
                                SetDlgItemInt(hWnd,IDC_EDIT16,nPeriod,FALSE);
                            }
                            else if(wParam==IDC_BUTTON21){
                                int nPeriod = MakePrime(GetDlgItemInt(hWnd,IDC_EDIT18,NULL,FALSE));
                                SetDlgItemInt(hWnd,IDC_EDIT18,nPeriod,FALSE);
                            }
                            else if(HIWORD(wParam)==LBN_DBLCLK && LOWORD(wParam)==IDC_LIST1){
                                int i = SendDlgItemMessage(hWnd,IDC_LIST1,LB_GETCURSEL,0,0);
                                if(i==-1)
                                    return 0;
                                CHOOSECOLOR col={sizeof(CHOOSECOLOR)};
                                col.hwndOwner = hWnd;
                                col.lpCustColors = colCust;
                                col.Flags = CC_RGBINIT;
                                COLOR14 c = g_SFT.GetKeyColor(i);
                                col.rgbResult = RGB(c.b,c.g,c.r);
                                if(ChooseColor(&col)){
```

```
                                char *cc = (char*)&col.rgbResult;
                                c.b = cc[0];
                                c.g = cc[1];
                                c.r = cc[2];
715                             g_SFT.SetKeyColor(c,i);
                                InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,↙
                                    ↳ FALSE);
                        }
                }
                else if(wParam==IDC_BUTTON8){
720                     COLOR14 c;
                        CHOOSECOLOR col={sizeof(CHOOSECOLOR)};
                        col.hwndOwner = hWnd;
                        col.lpCustColors = colCust;
                        col.Flags = CC_RGBINIT;
725                     int nStep = GetDlgItemInt(hWnd,IDC_EDIT9,NULL,FALSE);
                        if(nStep==0)
                                nStep =1;
                        int nPercent = GetDlgItemInt(hWnd,IDC_EDIT10,NULL,FALSE)↙
                            ↳ ;
                        if(nPercent>100)
730                             nPercent=100;
                        if(ChooseColor(&col)){
                                char *cc = (char*)&col.rgbResult;
                                c.b = cc[0];
                                c.g = cc[1];
735                             c.r = cc[2];
                                int i = SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                    ↳ LB_GETCURSEL,0,0);
                                if(i==-1)
                                        i=0;
                                if(SendDlgItemMessage(hWnd,IDC_RADIO4,↙
                                    ↳ BM_GETCHECK,0,0)){
740                                     for(;i<g_SFT.GetNumOfColors();i+=nStep){
                                                COLOR14 n = g_SFT.GetKeyColor(i)↙
                                                    ↳ ;
                                                n.r = ((100-nPercent)*n.r+↙
                                                    ↳ nPercent*c.r)/100;
                                                n.g = ((100-nPercent)*n.g+↙
                                                    ↳ nPercent*c.g)/100;
                                                n.b = ((100-nPercent)*n.b+↙
                                                    ↳ nPercent*c.b)/100;
745                                             g_SFT.SetKeyColor(n,i);
                                        }
                                }
                                else{
                                        for(;i<g_SFT.GetNumOfColors();i++){
750                                             double g = sin((pi*i*2)/nStep)↙
                                                    ↳ /2+.5;
                                                g = (double)nPercent*g/(double)↙
                                                    ↳ 100;
                                                COLOR14 n = g_SFT.GetKeyColor(i)↙
                                                    ↳ ;
                                                n.r = ((1-g)*n.r+g*c.r);
                                                n.g = ((1-g)*n.g+g*c.g);
755                                             n.b = ((1-g)*n.b+g*c.b);
                                                g_SFT.SetKeyColor(n,i);
```

```
                                        }
                                }
                        }
                        SendMessage(hWnd,WM_USER+99,0,0);
                        if (g_AutoColour) g_SFT.ApplyColors();
                }
                else if(wParam==IDC_BUTTON9){
                        COLOR14 c;
                        CHOOSECOLOR col={sizeof(CHOOSECOLOR)};
                        col.hwndOwner = hWnd;
                        col.lpCustColors = colCust;
                        col.Flags = CC_RGBINIT;
                        if(ChooseColor(&col)){
                                char *cc = (char*)&col.rgbResult;
                                c.b = cc[0];
                                c.g = cc[1];
                                c.r = cc[2];
                                int i = SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                    ↳ LB_GETCURSEL,0,0);
                                if(i==-1)
                                        i=0;
                                for(;i<g_SFT.GetNumOfColors();i+=3){
                                        COLOR14 n = g_SFT.GetKeyColor(i);
                                        n.r = (n.r+c.r)/2;
                                        n.g = (n.g+c.g)/2;
                                        n.b = (n.b+c.b)/2;
                                        g_SFT.SetKeyColor(n,i);
                                }
                        }
                        SendMessage(hWnd,WM_USER+99,0,0);
                        if (g_AutoColour) g_SFT.ApplyColors();
                }
                else if(wParam==IDC_BUTTON11){
                        COLOR14 c;
                        CHOOSECOLOR col={sizeof(CHOOSECOLOR)};
                        col.hwndOwner = hWnd;
                        col.lpCustColors = colCust;
                        col.Flags = CC_RGBINIT;
                        if(ChooseColor(&col)){
                                char *cc = (char*)&col.rgbResult;
                                c.b = cc[0];
                                c.g = cc[1];
                                c.r = cc[2];
                                int i = SendDlgItemMessage(hWnd,IDC_LIST1,↙
                                    ↳ LB_GETCURSEL,0,0);
                                if(i==-1)
                                        i=0;
                                for(;i<g_SFT.GetNumOfColors();i+=4){
                                        COLOR14 n = g_SFT.GetKeyColor(i);
                                        n.r = (n.r+c.r)/2;
                                        n.g = (n.g+c.g)/2;
                                        n.b = (n.b+c.b)/2;
                                        g_SFT.SetKeyColor(n,i);
                                }
                        }
                        SendMessage(hWnd,WM_USER+99,0,0);
                        if (g_AutoColour) g_SFT.ApplyColors();
```

```
		}
		else if(wParam==IDC_INTERIORCOLOR){
			COLOR14 c;
815			CHOOSECOLOR col={sizeof(CHOOSECOLOR)};
			col.hwndOwner = hWnd;
			col.lpCustColors = colCust;
			col.Flags = CC_RGBINIT;
			if(ChooseColor(&col)){
820				char *cc = (char*)&col.rgbResult;
				c.b = cc[0];
				c.g = cc[1];
				c.r = cc[2];
				g_SFT.SetInteriorColor(c);
825			}
			SendMessage(hWnd,WM_USER+99,0,0);
			if (g_AutoColour) g_SFT.ApplyColors();
		}
		else if(wParam==IDC_BUTTON10){
830			int nParts = g_SFT.GetNumOfColors();
			int nPParts = nParts;
			nParts*=2;
			if(nParts>1024)
				nParts=1024;
835			g_SFT.ChangeNumOfColors(nParts);
			int i, j=0;
			for(i=nPParts;i<nParts;i++){
				COLOR14 c = g_SFT.GetKeyColor(j++);
				g_SFT.SetKeyColor(c,i);
840			}
			SendMessage(hWnd,WM_USER+99,0,0);
			if (g_AutoColour) g_SFT.ApplyColors();
		}
		else if(wParam==IDC_BUTTON13){
845			srand(GetTickCount());
			int nColors = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0);
			if(nColors>1024){
				nColors=1024;
				SetDlgItemInt(hWnd,IDC_EDIT1,nColors,FALSE);
850			}
			int nSeed = rand();
			SetDlgItemInt(hWnd,IDC_EDIT2,nSeed,0);
			g_SFT.GenerateColors(nColors,nSeed);
			if (g_AutoColour) g_SFT.ApplyColors();
855			InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
			InvalidateRect(GetParent(hWnd),NULL,FALSE);
			if(nColors!=SendDlgItemMessage(hWnd,IDC_LIST1,↙
			    ↳ LB_GETCOUNT,0,0)){
				SendDlgItemMessage(hWnd,IDC_LIST1,↙
				    ↳ LB_RESETCONTENT,0,0);
				int i;
860				for(i=0;i<nColors;i++)
					SendDlgItemMessage(hWnd,IDC_LIST1,↙
					    ↳ LB_ADDSTRING,0,(LPARAM)"");
			}
			InvalidateRect(hWnd,NULL,FALSE);
		}
865	}
```

```
        else if(uMsg==WM_USER+88){
                POINT p;
                GetCursorPos(&p);
                ScreenToClient(GetDlgItem(hWnd,IDC_LIST1),&p);
870             int i = SendDlgItemMessage(hWnd,IDC_LIST1,LB_ITEMFROMPOINT,0,↙
                    ↳ MAKELONG(p.x,p.y));
                SendDlgItemMessage(hWnd,IDC_LIST1,LB_SETCURSEL,i,0);
                HMENU hMen = CreatePopupMenu();
                AppendMenu(hMen,MF_STRING,1,"Copy color");
                AppendMenu(hMen,MF_STRING,2,"Paste color");
875             AppendMenu(hMen,MF_SEPARATOR,0,"");
                AppendMenu(hMen,MF_STRING,3,"Paste every second");
                AppendMenu(hMen,MF_STRING,4,"Paste every third");
                AppendMenu(hMen,MF_STRING,5,"Paste every forth");
                AppendMenu(hMen,MF_STRING,6,"Paste every fifth");
880             AppendMenu(hMen,MF_STRING,7,"Paste every sixth");
                AppendMenu(hMen,MF_STRING,8,"Paste every seventh");
                AppendMenu(hMen,MF_STRING,9,"Paste every eighth");
                AppendMenu(hMen,MF_SEPARATOR,0,"");
                AppendMenu(hMen,MF_STRING,10,"Capture from mouse");
885             GetCursorPos(&p);
                int rc = TrackPopupMenu(hMen,TPM_LEFTALIGN|TPM_RETURNCMD,p.x,p.y↙
                    ↳ ,0,hWnd,NULL);
                DestroyMenu(hMen);
                if(rc==0)
                        return 0;
890             if(rc==1)
                        g_colCopy = g_SFT.GetKeyColor(i);
                else if(rc==2){
                        g_SFT.SetKeyColor(g_colCopy,i);
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
895             }
                else if(rc==3){
                        do{
                                g_SFT.SetKeyColor(g_colCopy,i);
                                i+=2;
900                     }while(i<g_SFT.GetNumOfColors());
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
                else if(rc==4){
                        do{
905                             g_SFT.SetKeyColor(g_colCopy,i);
                                i+=3;
                        }while(i<g_SFT.GetNumOfColors());
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
910             else if(rc==5){
                        do{
                                g_SFT.SetKeyColor(g_colCopy,i);
                                i+=4;
                        }while(i<g_SFT.GetNumOfColors());
915                     InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
                else if(rc==6){
                        do{
                                g_SFT.SetKeyColor(g_colCopy,i);
920                             i+=5;
```

```
                        }while(i<g_SFT.GetNumOfColors());
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
                else if(rc==7){
                        do{
                                g_SFT.SetKeyColor(g_colCopy,i);
                                i+=6;
                        }while(i<g_SFT.GetNumOfColors());
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
                else if(rc==8){
                        do{
                                g_SFT.SetKeyColor(g_colCopy,i);
                                i+=7;
                        }while(i<g_SFT.GetNumOfColors());
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
                else if(rc==9){
                        do{
                                g_SFT.SetKeyColor(g_colCopy,i);
                                i+=8;
                        }while(i<g_SFT.GetNumOfColors());
                        InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
                }
                else if(rc==10){
                        SetCapture(hWnd);
                        g_bCaptureMouse=TRUE;
                }
        }
        else if(uMsg==WM_DRAWITEM){
                LPDRAWITEMSTRUCT lpdis = (LPDRAWITEMSTRUCT)lParam;
                COLOR14 c = g_SFT.GetKeyColor(lpdis->itemID);
                HBRUSH br = CreateSolidBrush(RGB(c.b,c.g,c.r));
                FillRect(lpdis->hDC,&lpdis->rcItem,br);
                DeleteObject(br);
                if(lpdis->itemState & ODS_SELECTED){
                        SkuggadRect(lpdis->hDC,lpdis->rcItem,FALSE,0,0);
                        lpdis->rcItem.left++;
                        lpdis->rcItem.top++;
                        lpdis->rcItem.right--;
                        lpdis->rcItem.bottom--;
                        SkuggadRect(lpdis->hDC,lpdis->rcItem,FALSE,TRUE,0);
                }
        }
        else if(uMsg==WM_TIMER){
                POINT p;
                GetCursorPos(&p);
                ScreenToClient(GetParent(hWnd),&p);
                RECT rc;
                GetClientRect(GetParent(hWnd),&rc);
                RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
                int x = p.x*g_SFT.GetWidth()/rc.right;
                int y = p.y*g_SFT.GetHeight()/rc.bottom;
                int i = g_SFT.GetColorIndex(x,y);
```

```
                i/=(1024/g_SFT.GetNumOfColors());
                SendDlgItemMessage(hWnd,IDC_LIST1,LB_SETCURSEL,i,0);
        }
        else if(uMsg==WM_CAPTURECHANGED){
                g_bCaptureMouse=FALSE;
                SendDlgItemMessage(hWnd,IDC_CHECK1,BM_SETCHECK,0,0);
                KillTimer(hWnd,0);
        }
        else if(uMsg==WM_LBUTTONDOWN && SendDlgItemMessage(hWnd,IDC_CHECK1,↙
            ↳ BM_GETCHECK,0,0)){
                POINT p;
                GetCursorPos(&p);
                RECT rC;
                GetWindowRect(hWnd,&rC);
                if(p.x>=rC.left && p.x<=rC.right && p.y>=rC.top && p.y<=rC.↙
                    ↳ bottom){
                        KillTimer(hWnd,0);
                        ReleaseCapture();
                        return 0;
                }
                ScreenToClient(GetParent(hWnd),&p);

                RECT rc;
                GetClientRect(GetParent(hWnd),&rc);
                RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
                int x = p.x*g_SFT.GetWidth()/rc.right;
                int y = p.y*g_SFT.GetHeight()/rc.bottom;

                int i = g_SFT.GetColorIndex(x,y);
                i/=(1024/g_SFT.GetNumOfColors());
                SendDlgItemMessage(hWnd,IDC_LIST1,LB_SETCURSEL,i,0);
                ReleaseCapture();
                SendDlgItemMessage(hWnd,IDC_CHECK1,BM_SETCHECK,0,0);
                KillTimer(hWnd,0);
                if(i!=-1)
                        SendMessage(hWnd,WM_COMMAND,MAKELONG(IDC_LIST1,↙
                            ↳ LBN_DBLCLK),(LPARAM)GetDlgItem(hWnd,IDC_LIST1));
        }
        else if(uMsg==WM_LBUTTONDOWN && g_bCaptureMouse){
                g_bCaptureMouse=FALSE;
                ReleaseCapture();
                POINT p;
                GetCursorPos(&p);
                HDC hDC = GetDC(NULL);
                COLORREF col = GetPixel(hDC,p.x,p.y);
                ReleaseDC(NULL,hDC);

                int i = SendDlgItemMessage(hWnd,IDC_LIST1,LB_GETCURSEL,0,0);
                if(i==-1)
                        return 0;
                COLOR14 c = g_SFT.GetKeyColor(i);
                char *cc = (char*)&col;
                c.b = cc[0];
                c.g = cc[1];
```

```
                c.r = cc[2];
                g_SFT.SetKeyColor(c,i);
                InvalidateRect(GetDlgItem(hWnd,IDC_LIST1),NULL,FALSE);
        }
    else if(uMsg==WM_DROPFILES)
    {
                HDROP hDrop = (HDROP) wParam;
                if (hDrop)
                {
                        UINT len = DragQueryFile(hDrop, 0, 0, 0);
                        if (len > 0)
                        {
                                char *buffer = (char *) calloc(1, 2 * len + 1);
                                if (buffer)
                                {
                                        UINT ok = DragQueryFile(hDrop, 0, buffer↙
                                            ↳ , 2 * len);
                                        if (ok)
                                        {
                                                std::string file(buffer);
                                                g_SFT.OpenFile(file, TRUE);
                                                SendMessage(hWnd,WM_USER+99,0,0)↙
                                                    ↳ ;
                                                if (g_AutoColour) g_SFT.↙
                                                    ↳ ApplyColors();
                                                g_AutoUpdate++;
                                                SendMessage(hWnd,WM_COMMAND,IDOK↙
                                                    ↳ ,0);
                                                g_AutoUpdate--;
                                        }
                                        free(buffer);
                                }
                        }
                        DragFinish(hDrop);
                }
        }

        if(g_pWaves)
                g_pWaves->ProcessMessage(hWnd,uMsg,wParam,lParam);
        if(uMsg==WM_COMMAND && (wParam==IDC_BUTTON26 || wParam==IDC_BUTTON27 || ↙
            ↳ wParam==IDC_BUTTON28))
        {
                g_AutoUpdate++;
                SendMessage(hWnd,WM_COMMAND,IDOK,0);
                g_AutoUpdate--;
        }
        return 0;
}

extern const char *ColorToolTip(int nID)
{
        switch(nID){
        case IDC_EDIT1:
        case IDC_SPIN1:
                return "Number of Key Colors\nThese colors will be spread out on↙
                    ↳ the 1024 color palette\nInteger value";
        case IDC_EDIT3:
```

```
        case IDC_SPIN2:
            return "Division of iteration mapping\nThis value can be a float↙
                ↳  value but not negative";
1085    case IDC_EDIT12:
        case IDC_SPIN5:
            return "Color offset\nOffset the colors in the palette,\nvalid ↙
                ↳ values are 0-1024";
        case IDC_EDIT2:
            return "Seed value for making a random palette";
1090    case IDC_BUTTON2:
            return "Generate the given number of key colors\nfrom the seed ↙
                ↳ value";
        case IDC_BUTTON13:
            return "Generate the given number of key colors\nfrom a random ↙
                ↳ seed value";
        case IDC_BUTTON12:
1095        return "Generate RGB key colors\nfrom the given values";
        case IDC_BUTTON14:
            return "Generate RGB key colors\nfrom random values";
        case IDC_BUTTON1:
            return "Save the current palette";
1100    case IDC_BUTTON5:
            return "Open a previosly saved palette";
        case IDC_CHECK4:
            return "Apply 3D-like shadows based on changes in iteration ↙
                ↳ values";
        case IDC_EDIT20:
1105    case IDC_SPIN6:
            return "Slope shadow depth";
        case IDC_EDIT21:
        case IDC_SPIN7:
            return "Slope shadow strength";
1110    case IDC_EDIT22:
        case IDC_SPIN8:
            return "Slope shadow angle (0-360)";
        case IDC_BUTTON22:
            return "Select an image from which colors will be fetched";
1115    case IDC_BUTTON6:
            return "Double the number of Key Colors\nby spreading out the ↙
                ↳ current colors";
        case IDC_BUTTON7:
            return "Expand the number of Key Colors to 1024\nby spreading ↙
                ↳ out the current colors";
        case IDC_BUTTON10:
1120        return "Double by repeating the Key Colors";
        case IDC_CHECK1:
            return "Move the cursor over the fractal to select the Key Color↙
                ↳  in the list.\nWill only work if color offset is zero";
        case IDC_CHECK2:
            return "Make colors smooth";
1125    case IDC_CHECK3:
            return "Inverse color transition";
        case IDC_COMBO1:
            return "Color method. Available methods are\nStandard: Standard ↙
                ↳ iteration band coloring\nSquare root: Iterations are ↙
                ↳ squared before colors are appplied\nCubic root: Cube root ↙
                ↳ is applied before colors\nLogarithm: Logarithm is applied ↙
```

```
                    ↪ before colors\nStretched: The palette is stretched over ↙
                    ↪ min-max iteration values\nDistance (Linear): Distance ↙
                    ↪ Estimation with linear transfer (2.11.1 compatible)\nDE+↙
                    ↪ Standard: hybrid mode\nDistance (Logarithm) DE with log ↙
                    ↪ transfer\nDistance (Square Root) DE with sqrt transfer ↙
                    ↪ (2.11.1+gmp.DATE compatible)";
            case IDC_DIFFERENCES:
1130            return "Derivative differencing calculation method for distance ↙
                    ↪ colouring";
            case IDC_RADIO4:
                return "Colors are merged on distinct steps";
            case IDC_RADIO5:
                return "Colors are merged by sine function with given period ↙
                    ↪ length";
1135        case IDC_EDIT9:
            case IDC_SPIN3:
                return "Length of period";
            case IDC_EDIT10:
            case IDC_SPIN4:
1140            return "Rate in percent of merged color";
            case IDC_BUTTON8:
                return "Select the color to be merged, which will be applied ↙
                    ↪ when OK is clicked in the color dialog";
            case IDC_LIST1:
                return "List of Key Colors.\nEach Key Color can be edited by ↙
                    ↪ double click.\nAdditional functions are available by right↙
                    ↪  click on a Key Color";
1145        case IDC_CHECK6:
                return "Activate sine waves on HSB coloring";
            case IDC_CHECK7:
                return "Blend Infinite Colors and the Color Palette";
            case IDC_COMBO4:
1150            return "Select type of wave:\nHue, Saturation or Brightness";
            case IDC_EDIT23:
                return "Period length of the wave";
            case IDC_BUTTON29:
                return "Change the period wave to the nearest higher prime value↙
                    ↪ ";
1155        case IDC_BUTTON30:
                return "Fill the palette with the colors from infinte waves";
            case IDC_BUTTON26:
                return "Add a new wave with the given values";
            case IDC_BUTTON27:
1160            return "Update the selected wave";
            case IDC_BUTTON28:
                return "Remove the selected wave";
            case IDC_EDIT11:
                return "Period lenght of Red color";
1165        case IDC_EDIT14:
                return "Period lenght of Green color";
            case IDC_EDIT16:
                return "Period lenght of Blue color";
            case IDC_EDIT18:
1170            return "Period lenght of Black and White color";
            case IDC_BUTTON16:
                return "Change the period wave to the nearest higher prime value↙
                    ↪  for Red color";
```

```
            case IDC_BUTTON19:
                return "Change the period wave to the nearest higher prime value↙
                    ↳ for Green color";
1175        case IDC_BUTTON20:
                return "Change the period wave to the nearest higher prime value↙
                    ↳ for Blue color";
            case IDC_BUTTON21:
                return "Change the period wave to the nearest higher prime value↙
                    ↳ for Black and White color";
            case IDC_EDIT13:
1180            return "Change the start position of the wave of Red color";
            case IDC_EDIT15:
                return "Change the start position of the wave of Green color";
            case IDC_EDIT17:
                return "Change the start position of the wave of Blue color";
1185        case IDC_EDIT19:
                return "Change the start position of the wave of Black and White↙
                    ↳ color";
            case IDC_LIST6:
                return "List of Infinite Waves";
            case IDC_BUTTON17:
1190            return "Apply more contrast on the palette";
            case IDC_BUTTON18:
                return "Apply less contrast on the palette";
            case IDC_AUTOCOLOUR:
                return "Automatically apply palette on change";
1195        case IDOK:
                return "Apply current palette";
            case IDCLOSE:
                return "Close the dialog and undo all changes";
            case IDCANCEL:
1200            return "Close the dialog";
            case 1051:
                return "Enable texture";
            case 1052:
                return "Texture depth";
1205        case 1053:
                return "Texture strength/ratio";
            case 1054:
                return "Browse for image";
            case 1055:
1210            return "Texture image";
      default:
        static char tooltip[100];
        snprintf(tooltip, 100, "%d", nID);
        return tooltip;
1215    }
    }
```

80 fraktal_sft/main_color.h

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Runmo
    Copyright (C) 2017-2018 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
```

```
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

    #ifndef KF_MAIN_COLOR_H
    #define KF_MAIN_COLOR_H 1

    #include <windows.h>

    extern int WINAPI ColorProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
    extern const char *ColorToolTip(int nID);

    #endif
```

81 fraktal_sft/main.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Runmo
    Copyright (C) 2017-2019 Claude Heiland-Allen

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

    // Kalles Fraktaler 2
    //
    // © 2014-2015 Karl Runmo ,runmo@hotmail.com
    //
    // COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT ↙
        ↘ WARRANTY
    // OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ↙
        ↘ WARRANTIES
    // THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR ↙
        ↘ PURPOSE
    // OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE ↙
        ↘ COVERED
    // CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU↙
```

```
    ↪ (NOT
// THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY ↙
    ↪ NECESSARY
// SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ↙
    ↪ ESSENTIAL
// PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER ↙
    ↪ EXCEPT UNDER
// THIS DISCLAIMER.

#include <windows.h>
#include <math.h>
#include <commctrl.h>
#include <math.h>
#include "../common/parallell.h"
#include "../common/getimage.h"
#include "../common/StringVector.h"
#include "../common/FolderBrowser.h"
#include "listbox.h"
#include "../common/tooltip.h"
#include "resource.h"
#include "fraktal_sft.h"
#include "newton.h"
#include <malloc.h>
#include "../formula/formula.h"
#ifdef KF_OPENCL
#include "../cl/opencl.h"
#endif
#include "../common/bitmap.h"
#include <png.h>
#include <zlib.h>
#include <gsl/gsl_version.h>
#include <pixman.h>
#include <IlmBaseConfig.h>
#include <OpenEXRConfig.h>
#include "check_for_update.h"
#include "jpeg.h"
#include "png.h"
#include "tiff.h"
#include "exr.h"
#include "main.h"
#include "main_color.h"
#include "main_examine.h"
#include "main_iterations.h"
#include "main_position.h"
#include "cmdline.h"
#include <iostream>
#include <sstream>

#ifdef KF_OPENCL
std::vector<cldevice> cldevices;
#endif

POINT g_pInflections[10];
int g_nInflection=0;
extern double g_SeedR;
extern double g_SeedI;
extern double g_FactorAR;
```

```
extern double g_FactorAI;

//#define PARAM_ANIMATION

BOOL g_bNewton=FALSE;
HWND g_hwNewton=NULL;
BOOL g_bResizing=FALSE;
BOOL g_bTrackSelect=FALSE;
POINT g_pTrackStart;
HICON g_hIcon;

bool g_bAddReference=false;
bool g_bEraser=false;

#ifdef KF_OPENCL
HWND g_hwOpenCL = NULL;
#endif


double g_nMinDiff=0;

extern BOOL g_LDBL;
extern int g_nLDBL;
extern int g_nEXP;
#ifdef _WIN64
#define GCL_WNDPROC -24
#define GWL_WNDPROC -4
#define GWL_USERDATA -21
#endif

extern double g_Degree;
extern double g_real;
extern double g_imag;
extern int g_bAutoGlitch;

BOOL ISFLOATOK(double a);

CFraktalSFT g_SFT;

bool g_bAddMainReference=false;
int g_bAutoGlitch = 1;
BOOL g_bRotate=FALSE;
BOOL g_bMove=FALSE;
BOOL g_bShowInflection=FALSE;
BOOL g_bShowSkew=FALSE;
double g_nSkewStretch;
double g_nSkewRotate;
double g_StartDegree=0;
BOOL bXSign=FALSE;
double g_MoveDegree=0;
SIZE g_scSize = {640,360};
BOOL g_bAnimateEachFrame=FALSE;

int g_nPrevGlitchX=-1;
int g_nPrevGlitchY=-1;
BOOL g_bStoreZoom=FALSE;
BOOL g_bStoreZoomMap=FALSE;
```

```
BOOL g_bStoreZoomJpg=FALSE;
BOOL g_bStoreZoomPng=FALSE;
BOOL g_bStoreZoomTif=FALSE;
BOOL g_bStoreZoomExr=FALSE;
int g_nStoreZoomCount = 0;
int g_nStoreZoomLimit = 0;

bool g_bWaitRead=false;
int g_nStopAtExponent=0;

const char *g_pszStatus[] = {
                        "Generate reference",
                        "Render first image",
                        "Check done images",
                        "Center",
                        "Top",
                        "Bottom",
                        "Top left",
                        "Bottom left",
                        "Top right",
                        "Bottom right",
                        "Left",
                        "Right"};
int g_nStatus=0;
BOOL g_bFirstDone=TRUE;
BOOL g_bSaveJpeg=FALSE;
BOOL g_bSavePng=FALSE;
BOOL g_bSaveTif=FALSE;
BOOL g_bSaveExr=FALSE;
BOOL g_bSaveMap=FALSE;
BOOL g_bInteractive=TRUE;
const CommandLineArguments *g_args = 0;

char g_szRecovery[256];

BOOL g_bResetReference=FALSE;
BOOL g_bFindMinibrot=FALSE;
BOOL g_bFindMinibrotCount=0;
int g_bFindMinibrotPos=0;

BOOL g_bZoomRunning=FALSE;
BOOL g_bZoomStop=FALSE;

static void bmp2rgb(BYTE *rgb, const BYTE *bmp, int height, int width, int ↙
    ↪ stride, int bytes)
{
        // TODO add support for strict aliasing optimisations, "restrict" etc
        for (int y = height; y >= 0; --y)
        {
                int k = y * stride;
                for (int x = 0; x < width; ++x)
                        if (k + 2 < bytes)
                        {
                                k += 3;
                                *rgb++ = bmp[--k];
                                *rgb++ = bmp[--k];
                                *rgb++ = bmp[--k];
```

```
195                                     k += 3;
                                }
        }
}
extern int SaveImage(const std::string &szFileName,HBITMAP bmBmp,int nQuality, ↙
    ↪ const std::string &comment)
200 {
        int row;
        BYTE *lpBits, *lpJeg;
        BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
        HDC hDC = GetDC(NULL);
205     if(!GetDIBits(hDC,bmBmp,0,0,NULL,(LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                Beep(1000,10);
        bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
        bmi.biBitCount = 24;
        row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
210     bmi.biSizeImage=row*bmi.biHeight;
        lpBits = new BYTE[bmi.biSizeImage];
        lpJeg = new BYTE[bmi.biSizeImage];
        if(!GetDIBits(hDC,bmBmp,0,bmi.biHeight,lpBits,
                        (LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
215             Beep(1000,10);
        bmp2rgb(lpJeg, lpBits, bmi.biHeight, bmi.biWidth, row, bmi.biSizeImage);
        int nRet;
        if (nQuality == -1)
                nRet = SavePNG(szFileName,(char*)lpJeg,bmi.biHeight,bmi.biWidth↙
                    ↪ ,3,comment);
220     else if (nQuality == -2)
                nRet = SaveTIFF(szFileName,(char*)lpJeg,bmi.biHeight,bmi.biWidth↙
                    ↪ ,3,comment);
        else if (nQuality == -3)
        {
                bool allocate = ! g_SFT.GetHalfColour();
225             if (allocate)
                {
                        g_SFT.SetHalfColour(true);
                        g_SFT.ApplyColors();
                }
230             nRet = SaveEXR
                  ( szFileName
                  , lpJeg
                  , bmi.biWidth
                  , bmi.biHeight
235               , 3
                  , comment
                  , g_SFT.GetIterations()
                  , g_SFT.GetWidth()
                  , g_SFT.GetHeight()
240               , g_SFT.GetArrayCount()
                  , g_SFT.GetArrayTrans()
                  , g_SFT.GetArrayDEx()
                  , g_SFT.GetArrayDEy()
                  );
245             if (allocate)
                {
                        g_SFT.SetHalfColour(false);
                }
```

```
        }
        else
                nRet = SaveJPG(szFileName,(char*)lpJeg,bmi.biHeight,bmi.biWidth,3,nQuality,comment);
        delete [] lpJeg;
        delete [] lpBits;
        ReleaseDC(NULL,hDC);
        return nRet;
}
// this version doesn't go via a bitmap structure, avoiding dreaded blank images ...
extern int SaveImage(const std::string &szFileName, const BYTE *lpBits, int biWidth, int biHeight, int nQuality, const std::string &comment)
{
        assert(lpBits);
        assert(biWidth);
        assert(biHeight);
        int biBitCount = 24;
        int row = ((((biWidth*(DWORD)biBitCount)+31)&~31) >> 3);
        int biSizeImage=row*biHeight;
        BYTE *lpJeg = new BYTE[biSizeImage];
        bmp2rgb(lpJeg, lpBits, biHeight, biWidth, row, biSizeImage);
        int nRet;
        if (nQuality == -1)
                nRet = SavePNG(szFileName,(char*)lpJeg,biHeight,biWidth,3,comment);
        else if (nQuality == -2)
                nRet = SaveTIFF(szFileName,(char*)lpJeg,biHeight,biWidth,3,comment);
        else if (nQuality == -3)
        {
                bool allocate = ! g_SFT.GetHalfColour();
                if (allocate)
                {
                        g_SFT.SetHalfColour(true);
                        g_SFT.ApplyColors();
                }
                nRet = SaveEXR
                  ( szFileName
                  , lpJeg
                  , biWidth
                  , biHeight
                  , 3
                  , comment
                  , g_SFT.GetIterations()
                  , g_SFT.GetWidth()
                  , g_SFT.GetHeight()
                  , g_SFT.GetArrayCount()
                  , g_SFT.GetArrayTrans()
                  , g_SFT.GetArrayDEx()
                  , g_SFT.GetArrayDEy()
                  );
                if (allocate)
                {
                        g_SFT.SetHalfColour(false);
                }
        }
```

```
        else
                nRet = SaveJPG(szFileName,(char*)lpJeg,biHeight,biWidth,3,
                    ↪ nQuality,comment);
        delete [] lpJeg;
        return nRet;
}
POINT g_pSelect, g_pStart;
int g_bSelect=0;
HWND g_hwStatus=NULL;
__int64 g_nTStart;
BOOL g_bRunning=FALSE;
HWND g_hwHair;
HWND g_hwColors=NULL;
std::string g_szFile;
std::string g_szSettingsFile;

extern double GetDlgItemFloat(HWND hWnd,int nID)
{
        char szText[256];
        GetDlgItemText(hWnd,nID,szText,sizeof(szText));
        return atof(szText);
}

extern void SetDlgItemFloat(HWND hWnd,int nID,double val)
{
        char szText[256];
        snprintf(szText,256,"%.4f",val);
        SetDlgItemText(hWnd,nID,szText);
}

extern int FileExists(const std::string &szFind)
{
        WIN32_FIND_DATA wf;
        HANDLE hFind = FindFirstFile(szFind.c_str(),&wf);
        if(hFind==INVALID_HANDLE_VALUE)
                return 0;
        FindClose(hFind);
        return 1;
}

static void EnableUnsafeMenus(HWND hWnd, bool enable=true)
{
        HMENU hMenu = GetMenu(hWnd);
#define M(x) EnableMenuItem(hMenu, x, MF_BYCOMMAND | (enable ? MF_ENABLED : 
    ↪ MF_GRAYED))
        M(ID_ACTIONS_THREADS_1_4);
        M(ID_ACTIONS_THREADS_1_2);
        M(ID_ACTIONS_THREADS_1);
        M(ID_ACTIONS_THREADS_2);
        M(ID_ACTIONS_THREADS_4);
        M(ID_ACTIONS_THREADS_RESERVE_CORE);
#undef M
        DrawMenuBar(hWnd);
}

static void DisableUnsafeMenus(HWND hWnd, bool disable=true)
{
```

```
        return EnableUnsafeMenus(hWnd, !disable);
}

const char *GetToolText_const(int nID,LPARAM lParam)
{
        if(lParam==0){
                return const_cast<char *>(ColorToolTip(nID));
        }
        else if(lParam==1){
                return const_cast<char *>(IterationToolTip(nID));
        }
        else if(lParam==2){
                return const_cast<char *>(PositionToolTip(nID));
        }
        else if(lParam==3){
                switch(nID){
                case IDC_EDIT1:
                case IDC_SPIN1:
                        return "Stretch in percent";
                case IDC_EDIT2:
                case IDC_SPIN2:
                        return "Rotation in degrees";
                case IDC_BUTTON1:
                        return "Reset to default values";
                case IDC_CHECK1:
                        return "When this button is clicked, it will stay down\↙
                            ↳ nWhile down, add 4 points in the Fractal view\↙
                            ↳ nthese points will be connected 2 and 2 by lines\↙
                            ↳ nThe program will automatically attempt to\nmake ↙
                            ↳ the angle perpendicular";
                case IDOK:
                        return "Apply and close";
                case IDCANCEL:
                        return "Close and undo";
                }
        }
        else if(lParam==4){
                return const_cast<char *>(ExamineToolTip(nID));
        }
        else if(lParam==5){
                switch (nID)
                {
                        case IDC_STOREZOOM_KFB: return "Save KFB map files for ↙
                            ↳ each frame";
                        case IDC_STOREZOOM_EXR: return "Save EXR image + map ↙
                            ↳ files for each frame";
                        case IDC_STOREZOOM_TIF: return "Save TIF image files for↙
                            ↳  each frame";
                        case IDC_STOREZOOM_PNG: return "Save PNG image files for↙
                            ↳  each frame";
                        case IDC_STOREZOOM_JPG: return "Save JPEG image files ↙
                            ↳ for each frame";
                        case IDC_STOREZOOM_COUNTAUTO: return "Render only a ↙
                            ↳ limited number of frames";
                        case IDC_STOREZOOM_COUNT: return "Render this many ↙
                            ↳ frames (if above is checked)";
                        case IDOK: return "Apply and close";
```

```
                        case IDCANCEL: return "Close and undo";
                    }
                }
                static char szTmp[1024];
                wsprintf(szTmp,"nID=%d, lParam=%d",nID,lParam);
                return szTmp;
#if 0
#ifdef KF_OPENCL
                case IDC_COMBO_OPENCL_DEVICE:
                    return "Select the OpenCL device to use for per-pixel iteration calculations";
#endif
#endif
}

extern char *GetToolText(int nID, LPARAM lParam)
{
        return const_cast<char *>(GetToolText_const(nID, lParam));
}

// settings update

static void UpdateShrink(HWND hWnd)
{
        int s = g_SFT.GetShrink();
        CheckMenuItem(GetMenu(hWnd), ID_IMAGE_SHRINK_FAST,    MF_BYCOMMAND | (s == 0 ? MF_CHECKED : MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd), ID_IMAGE_SHRINK_DEFAULT, MF_BYCOMMAND | (s == 1 ? MF_CHECKED : MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd), ID_IMAGE_SHRINK_BEST,    MF_BYCOMMAND | (s == 2 ? MF_CHECKED : MF_UNCHECKED));
}

static void UpdateZoomSize(HWND hWnd)
{
        double z = g_SFT.GetZoomSize();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_2,MF_BYCOMMAND|(z==2?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_4,MF_BYCOMMAND|(z==4?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_8,MF_BYCOMMAND|(z==8?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_16,MF_BYCOMMAND|(z==16?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_32,MF_BYCOMMAND|(z==32?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_64,MF_BYCOMMAND|(z==64?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_128,MF_BYCOMMAND|(z==128?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateThreadsPerCore(HWND hWnd)
{
        double z = g_SFT.GetThreadsPerCore();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_THREADS_1_4,MF_BYCOMMAND|(z
```

```
            ↪ ==1./4?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_THREADS_1_2,MF_BYCOMMAND|(z↙
            ↪ ==1./2?MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_THREADS_1,MF_BYCOMMAND|(z==1?↙
            ↪ MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_THREADS_2,MF_BYCOMMAND|(z==2?↙
            ↪ MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_THREADS_4,MF_BYCOMMAND|(z==4?↙
            ↪ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateThreadsReserveCore(HWND hWnd)
{
        int z = g_SFT.GetThreadsReserveCore();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_THREADS_RESERVE_CORE,MF_BYCOMMAND↙
            ↪ |(z>0?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateAnimateZoom(HWND hWnd)
{
        bool b = g_SFT.GetAnimateZoom();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ANIMATEZOOM,MF_BYCOMMAND|(b?↙
            ↪ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateArbitrarySize(HWND hWnd)
{
        bool b = g_SFT.GetArbitrarySize();
        CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_ARBITRARYSIZE,MF_BYCOMMAND|(b?↙
            ↪ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateReuseReference(HWND hWnd)
{
        bool b = g_SFT.GetReuseReference();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_REUSEREFERENCE,MF_BYCOMMAND|(b?↙
            ↪ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateAutoSolveGlitches(HWND hWnd)
{
        bool b = g_SFT.GetAutoSolveGlitches();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_AUTOSOLVEGLITCHES,MF_BYCOMMAND|(b↙
            ↪ ?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateSolveGlitchNear(HWND hWnd)
{
        bool b = g_SFT.GetSolveGlitchNear();
        CheckMenuItem(GetMenu(hWnd),↙
            ↪ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD,MF_BYCOMMAND|(b↙
            ↪ ?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateNoApprox(HWND hWnd)
{
        bool b = g_SFT.GetNoApprox();
```

```
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_NOAPPROXIMATION,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateMirror(HWND hWnd)
{
        bool b = g_SFT.GetMirror();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_SPECIAL_MIRROR1,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateLongDoubleAlways(HWND hWnd)
{
        bool b = g_SFT.GetLongDoubleAlways();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_USELONGDOUBLEFROMSTART,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateFloatExpAlways(HWND hWnd)
{
        bool b = g_SFT.GetFloatExpAlways();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_USEFLOATEXPALWAYS,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateUseNanoMB1(HWND hWnd)
{
        bool b = g_SFT.GetUseNanoMB1();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_USENANOMB1,MF_BYCOMMAND|(↙
            ↳ b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateUseNanoMB2(HWND hWnd)
{
        bool b = g_SFT.GetUseNanoMB2();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_USENANOMB2,MF_BYCOMMAND|(↙
            ↳ b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateInteriorChecking(HWND hWnd)
{
        bool b = g_SFT.GetInteriorChecking();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_INTERIORCHECKING,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateAutoIterations(HWND hWnd)
{
        bool b = g_SFT.GetAutoIterations();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_AUTOITERATION,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateGuessing(HWND hWnd)
{
        bool b = g_SFT.GetGuessing();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_GUESSING,MF_BYCOMMAND|(b?↙
```

```
            ↳ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateShowGlitches(HWND hWnd)
{
        bool b = g_SFT.GetShowGlitches();
        CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_SHOWGLITCHES,MF_BYCOMMAND|(b?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateNoReuseCenter(HWND hWnd)
{
        bool b = g_SFT.GetNoReuseCenter();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_NOREUSECENTER,↙
            ↳ MF_BYCOMMAND|(b?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateIsolatedGlitchNeighbourhood(HWND hWnd)
{
        double z = g_SFT.GetIsolatedGlitchNeighbourhood();
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ISOLATED_0,MF_BYCOMMAND|(z==0?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ISOLATED_4,MF_BYCOMMAND|(z==4?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
        CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ISOLATED_8,MF_BYCOMMAND|(z==8?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateShowCrossHair(HWND hWnd)
{
        bool b = g_SFT.GetShowCrossHair();
        CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_SHOWCROSSHAIR,MF_BYCOMMAND|(b?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
        ShowWindow(g_hwHair, b ? SW_SHOWNA : SW_HIDE);
}

static void UpdateHalfColour(HWND hWnd)
{
        bool b = g_SFT.GetHalfColour();
        CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_HALFCOLOUR,MF_BYCOMMAND|(b?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateSaveOverwrites(HWND hWnd)
{
        bool b = g_SFT.GetSaveOverwrites();
        CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_SAVE_OVERWRITES,MF_BYCOMMAND|(b?↙
            ↳ MF_CHECKED:MF_UNCHECKED));
}

static void UpdateThreadedReference(HWND hWnd)
{
        bool b = g_SFT.GetThreadedReference();
        CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_THREADED_REFERENCE,MF_BYCOMMAND|(↙
            ↳ b?MF_CHECKED:MF_UNCHECKED));
}
```

```
static void UpdateSIMDVectorSize(HWND hWnd)
{
	int z = g_SFT.GetSIMDVectorSize();
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_VECTOR_SIZE_1,MF_BYCOMMAND|(z==1?MF_CHECKED:MF_UNCHECKED));
#if KF_SIMD >= 1
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_VECTOR_SIZE_2,MF_BYCOMMAND|(z==2?MF_CHECKED:MF_UNCHECKED));
#endif
#if KF_SIMD >= 2
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_VECTOR_SIZE_4,MF_BYCOMMAND|(z==4?MF_CHECKED:MF_UNCHECKED));
#endif
#if KF_SIMD >= 3
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_VECTOR_SIZE_8,MF_BYCOMMAND|(z==8?MF_CHECKED:MF_UNCHECKED));
#endif
#if KF_SIMD >= 4
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_VECTOR_SIZE_16,MF_BYCOMMAND|(z==16?MF_CHECKED:MF_UNCHECKED));
#endif
}

static void UpdateSIMDChunkSize(HWND hWnd)
{
	int z = g_SFT.GetSIMDChunkSize();
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_1,MF_BYCOMMAND|(z==1?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_2,MF_BYCOMMAND|(z==2?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_4,MF_BYCOMMAND|(z==4?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_8,MF_BYCOMMAND|(z==8?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_16,MF_BYCOMMAND|(z==16?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_32,MF_BYCOMMAND|(z==32?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_64,MF_BYCOMMAND|(z==64?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_128,MF_BYCOMMAND|(z==128?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_256,MF_BYCOMMAND|(z==256?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_512,MF_BYCOMMAND|(z==512?MF_CHECKED:MF_UNCHECKED));
	CheckMenuItem(GetMenu(hWnd),ID_SIMD_CHUNK_SIZE_1024,MF_BYCOMMAND|(z==1024?MF_CHECKED:MF_UNCHECKED));
}

static void UpdateMenusFromSettings(HWND hWnd)
{
	UpdateShrink(hWnd);
	UpdateZoomSize(hWnd);
	UpdateThreadsPerCore(hWnd);
	UpdateThreadsReserveCore(hWnd);
	UpdateAnimateZoom(hWnd);
```

```
630             UpdateArbitrarySize(hWnd);
                UpdateReuseReference(hWnd);
                UpdateAutoSolveGlitches(hWnd);
                UpdateSolveGlitchNear(hWnd);
                UpdateNoApprox(hWnd);
635             UpdateMirror(hWnd);
                UpdateLongDoubleAlways(hWnd);
                UpdateFloatExpAlways(hWnd);
                UpdateUseNanoMB1(hWnd);
                UpdateUseNanoMB2(hWnd);
640             UpdateInteriorChecking(hWnd);
                UpdateAutoIterations(hWnd);
                UpdateGuessing(hWnd);
                UpdateShowGlitches(hWnd);
                UpdateNoReuseCenter(hWnd);
645             UpdateIsolatedGlitchNeighbourhood(hWnd);
                UpdateShowCrossHair(hWnd);
                UpdateHalfColour(hWnd);
                UpdateSaveOverwrites(hWnd);
                UpdateThreadedReference(hWnd);
650             UpdateSIMDVectorSize(hWnd);
                UpdateSIMDChunkSize(hWnd);
        }

        static void UpdateWindowSize(HWND hWnd)
655     {
                int l = g_SFT.GetWindowLeft();
                int t = g_SFT.GetWindowTop();
                int r = g_SFT.GetWindowRight();
                int b = g_SFT.GetWindowBottom();
660             MoveWindow(hWnd,l,t,r,b,TRUE);
                int w = g_SFT.GetWindowWidth();
                int widths[3] = { (180 * w) / 640, (430 * w) / 640, -1 };
                SendMessage(g_hwStatus,SB_SETPARTS,3,(LPARAM)&widths);
        }
665
        #if 0
        static long WINAPI ShowProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
        {
                if(uMsg==WM_PAINT){
670                     PAINTSTRUCT ps;
                        BeginPaint(hWnd,&ps);
                        int i;
                        RECT cr;
                        GetClientRect(hWnd,&cr);
675                     for(i=0;i<1024;i++){
                                RECT r={i*cr.right/1024,0,(i+1)*cr.right/1024,cr.bottom↙
                                    ↪ };
                                COLOR14 c = g_SFT.GetColor(i);
                                HBRUSH br = CreateSolidBrush(RGB(c.b,c.g,c.r));
                                FillRect(ps.hdc,&r,br);
680                             DeleteObject(br);
                        }
                        EndPaint(hWnd,&ps);
                        return 0;
                }
685             return CallWindowProc((WNDPROC)GetClassLongPtr(hWnd,GCLP_WNDPROC),hWnd,↙
```

```
        ↪ uMsg,lParam,wParam);
}
#endif


690 static int WINAPI CrossHairProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        (void) wParam;
        (void) lParam;
        if(uMsg==WM_INITDIALOG){
695             SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                SetTimer(hWnd,1,50,NULL);

                HWND hwPrev = GetDlgItem(hWnd,IDC_PREV);
700             RECT r, r2={0,0,16,16};
                GetWindowRect(hwPrev,&r);
                ScreenToClient(hWnd,(LPPOINT)&r);
                r.right=r2.right*8;
                r.bottom=r2.bottom*8;
705             MoveWindow(hwPrev,r.left,r.top,r.right,r.bottom,TRUE);
        }
        else if(uMsg==WM_USER+112){
                HWND hwPrev = GetDlgItem(hWnd,IDC_PREV);
                RECT rc;
710             GetClientRect(hWnd,&rc);
                MoveWindow(hwPrev,0,0,rc.right,rc.bottom,TRUE);
        }
        else if(uMsg==WM_TIMER){
                POINT p;
715             GetCursorPos(&p);
                if(hWnd==WindowFromPoint(p) || hWnd==GetParent(WindowFromPoint(p↙
                    ↪ )))
                        return 0;
                HDC dcScreen = GetDC(NULL);

720             HWND hwPrev = GetDlgItem(hWnd,IDC_PREV);
                RECT r, r2={0,0,16,16};
                GetWindowRect(hwPrev,&r);
                ScreenToClient(hWnd,(LPPOINT)&r);
                r.right=r2.right*8;
725             r.bottom=r2.bottom*8;

                HDC dcPrev = GetDC(hwPrev);
                r2.left = p.x-r2.right/2+1;
                r2.top = p.y-r2.bottom/2+1;
730             StretchBlt(dcPrev,0,0,r.right,r.bottom,dcScreen,r2.left,r2.top,↙
                    ↪ r2.right,r2.bottom,SRCCOPY);

                SetROP2(dcPrev,R2_NOT);
                MoveToEx(dcPrev,r.right/2-4,0,NULL);
                LineTo(dcPrev,r.right/2-4,r.bottom);
735             MoveToEx(dcPrev,0,r.bottom/2-4,NULL);
                LineTo(dcPrev,r.right,r.bottom/2-4);

                ReleaseDC(hwPrev,dcPrev);
                ReleaseDC(NULL,dcScreen);
```

```
740             }
                return 0;
}
/*
int g_nFrames=0;
745 int WINAPI ZoomProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        if(uMsg==WM_INITDIALOG){
                RECT r;
                GetClientRect(GetParent(hWnd),&r);
750             SetDlgItemInt(hWnd,IDC_EDIT3,r.right,0);
                return 1;
        }
        if(uMsg==WM_COMMAND){
                if(wParam==IDC_BUTTON1){
755                     char szFolder[256];
                        GetDlgItemText(hWnd,IDC_EDIT1,szFolder,sizeof(szFolder))↙
                            ↳ ;
                        if(Browse(hWnd,szFolder,sizeof(szFolder)))
                                SetDlgItemText(hWnd,IDC_EDIT1,szFolder);
                }
760             else if(wParam==IDC_BUTTON3){
                        int nProcent = GetDlgItemInt(hWnd,IDC_EDIT2,NULL,FALSE);
                        if(!nProcent)
                                return MessageBox(hWnd,"No percent given","Error↙
                                    ↳ ",MB_OK|MB_ICONSTOP);
                        SetDlgItemInt(hWnd,IDC_EDIT5,g_SFT.CountFrames(↙
                            ↳ SendDlgItemMessage(hWnd,IDC_CHECK1,BM_GETCHECK↙
                            ↳ ,0,0)?25:nProcent),FALSE);
765             }
                else if(LOWORD(wParam)==IDC_EDIT3 && HIWORD(wParam)==EN_CHANGE){
                        int nHeight = GetDlgItemInt(hWnd,IDC_EDIT3,NULL,0)↙
                            ↳ *360/640;
                        SetDlgItemInt(hWnd,IDC_EDIT4,nHeight,0);
                }
770             else if(wParam==IDOK){
                        if(g_bZoomRunning){
                                g_bZoomStop=TRUE;
                                while(g_bZoomRunning)
                                        Sleep(10);
775                             SetDlgItemText(hWnd,IDOK,"Start");
                                return 0;
                        }
                        char szFolder[256];
                        GetDlgItemText(hWnd,IDC_EDIT1,szFolder,sizeof(szFolder))↙
                            ↳ ;
780                     if(!*szFolder){
                                SetFocus(GetDlgItem(hWnd,IDC_EDIT1));
                                return MessageBox(hWnd,"No folder given","Error↙
                                    ↳ ",MB_OK|MB_ICONSTOP);
                        }
                        int nWidth = GetDlgItemInt(hWnd,IDC_EDIT3,NULL,0);
785                     if(!nWidth){
                                SetFocus(GetDlgItem(hWnd,IDC_EDIT3));
                                return MessageBox(hWnd,"No width given","Error",↙
                                    ↳ MB_OK|MB_ICONSTOP);
                        }
```

```
                int nHeight = GetDlgItemInt(hWnd,IDC_EDIT4,NULL,0);
790             int nPercent = GetDlgItemInt(hWnd,IDC_EDIT2,NULL,0);
                if(!nPercent){
                        SetFocus(GetDlgItem(hWnd,IDC_EDIT2));
                        return MessageBox(hWnd,"No percent given","Error↙
                            ↳ ",MB_OK|MB_ICONSTOP);
                }
795             g_nFrames = GetDlgItemInt(hWnd,IDC_EDIT5,NULL,0);
                if(!g_nFrames){
                        SetFocus(GetDlgItem(hWnd,IDC_EDIT5));
                        return MessageBox(hWnd,"No frames given","Error↙
                            ↳ ",MB_OK|MB_ICONSTOP);
                }
800             BOOL bKfr = SendDlgItemMessage(hWnd,IDC_CHECK1,↙
                    ↳ BM_GETCHECK,0,0);
                char szParams[256];
                strcpy(szParams,szFolder);
                if(szParams[strlen(szParams)-1]!='\\')
                        strcat(szParams,"\\");
805             strcat(szParams,"Parameters.kfz");
                DWORD dw;
                if(g_SFT.SaveFile(szParams)){
                        CStringTable stParameters;
                        stParameters.AddRow();
810                     stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,"Folder");
                        stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,szFolder);
                        stParameters.AddRow();
                        stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,"Width");
                        stParameters.AddInt(stParameters.GetCount()-1,↙
                            ↳ nWidth);
815                     stParameters.AddRow();
                        stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,"Height");
                        stParameters.AddInt(stParameters.GetCount()-1,↙
                            ↳ nHeight);
                        stParameters.AddRow();
                        stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,"Frames");
820                     stParameters.AddInt(stParameters.GetCount()-1,↙
                            ↳ g_nFrames);
                        stParameters.AddRow();
                        stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,"Percent");
                        stParameters.AddInt(stParameters.GetCount()-1,↙
                            ↳ nPercent);
                        stParameters.AddRow();
825                     stParameters.AddString(stParameters.GetCount()↙
                            ↳ -1,"Kfr");
                        stParameters.AddInt(stParameters.GetCount()-1,↙
                            ↳ bKfr);
                        HANDLE hFile = CreateFile(szParams,GENERIC_READ|↙
                            ↳ GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL)↙
                            ↳ ;
                        if(hFile==INVALID_HANDLE_VALUE)
```

```
                                        return MessageBox(hWnd,"Could not write ↙
                                            ↳ in given folder","Error",MB_OK|↙
                                            ↳ MB_ICONSTOP);
830                                 SetFilePointer(hFile,0,NULL,FILE_END);
                                    char *szParameters = stParameters.ToText(": ","\↙
                                        ↳ r\n");
                                    WriteFile(hFile,szParameters,strlen(szParameters↙
                                        ↳ ),&dw,NULL);
                                    CloseHandle(hFile);
                                    stParameters.DeleteToText(szParameters);
835                             }
                                SetDlgItemText(hWnd,IDOK,"Stop");
                                SetTimer(hWnd,0,500,NULL);
                                RENDER_FRAKTAL *prf = new RENDER_FRAKTAL;
                                char *sz = g_SFT.GetPosition();
840                             CStringTable stP(sz,":","\n");
                                prf->rstart = stP[stP.FindString(0,"R-Start")][1];
                                prf->rstop = stP[stP.FindString(0,"R-Stop")][1];
                                prf->istart = stP[stP.FindString(0,"I-Start")][1];
                                prf->istop = stP[stP.FindString(0,"I-Stop")][1];
845                             prf->nFrames = g_nFrames;
                                prf->nProcent = (bKfr?25:nPercent);
                                prf->nMaxIter = g_SFT.GetIterations();
                                prf->r.left=prf->r.top=0;
                                prf->r.right = nWidth;
850                             prf->r.bottom = nHeight;
                                prf->hWnd = hWnd;
                                prf->kfrParts = (int)(bKfr?(0.5 + log10(1.5)/log10(1 + ↙
                                    ↳ 2*(double)nPercent/100)):0);
                                if(prf->kfrParts<1 && bKfr)
                                        prf->kfrParts=2;
855                             strcpy(prf->szFileName,szFolder);
                                if(prf->szFileName[strlen(prf->szFileName)-1]!='\\')
                                        strcat(prf->szFileName,"\\");
                                strcat(prf->szFileName,"000000.jpg");
                                prf->bCheckZoom=TRUE;
860                             g_bZoomRunning=TRUE;
                                g_bZoomStop=FALSE;
                                HANDLE hThread = CreateThread(NULL,150000000,(↙
                                    ↳ LPTHREAD_START_ROUTINE)RenderFraktal,(LPVOID)prf↙
                                    ↳ ,0,&dw);
                                CloseHandle(hThread);
//                              g_SFT.CreateZoomSequence(szFolder,nWidth,nHeight,↙
    ↳ g_nFrames,nPercent);
865                         }
                            else if(wParam==IDC_BUTTON4){
                                static char szFile[256]={0};
                                if(BrowseFile(hWnd,TRUE,"Open Incomplete Zoom Sequence↙
                                    ↳ ","Zoom parameters\0*.kfz\0",szFile,sizeof(szFile)↙
                                    ↳ )){
                                    DWORD dw;
870                                 HANDLE hFile = CreateFile(szFile,GENERIC_READ,0,↙
                                        ↳ NULL,OPEN_EXISTING,0,NULL);
                                    if(hFile==INVALID_HANDLE_VALUE)
                                            return MessageBox(hWnd,"Could not open ↙
                                                ↳ file","Error",MB_OK|MB_ICONSTOP);
                                    int nLen = GetFileSize(hFile,NULL);
```

```
                                        char *szData = new char[nLen+1];
875                                     ReadFile(hFile,szData,nLen,&dw,NULL);
                                        CloseHandle(hFile);
                                        szData[nLen]=0;
                                        CStringTable stParameters(szData,": ","\r\n");
                                        delete [] szData;
880                                     SetDlgItemText(hWnd,IDC_EDIT1,stParameters[
                                            ↪ stParameters.FindString(0,"Folder")][1]);
                                        SetDlgItemText(hWnd,IDC_EDIT2,stParameters[
                                            ↪ stParameters.FindString(0,"Percent")][1]);
                                        SetDlgItemText(hWnd,IDC_EDIT5,stParameters[
                                            ↪ stParameters.FindString(0,"Frames")][1]);
                                        SetDlgItemText(hWnd,IDC_EDIT3,stParameters[
                                            ↪ stParameters.FindString(0,"Width")][1]);
                                        SetDlgItemText(hWnd,IDC_EDIT4,stParameters[
                                            ↪ stParameters.FindString(0,"Height")][1]);
885                                     SendDlgItemMessage(hWnd,IDC_CHECK1,BM_SETCHECK,
                                            ↪ atoi(stParameters[stParameters.FindString
                                            ↪ (0,"Kfr")][1]),0);
                                        if(g_hwColors)
                                                SendMessage(g_hwColors,WM_USER+99,0,0);
                                        g_SFT.OpenFile(szFile);
                                }
890             }
                else if(wParam==IDCANCEL){
                        if(g_bZoomRunning){
                                g_bZoomStop=TRUE;
                                while(g_bZoomRunning)
895                                     Sleep(10);
                        }
                        EndDialog(hWnd,0);
                }
        }
900     else if(uMsg==WM_TIMER){
                char szDisplay[256];
                static char szPrev[256]="";
                wsprintf(szDisplay,"%d%% G:%d%% (%d of %d) %s",g_nDone*100/(
                    ↪ g_nMax?g_nMax:1),g_nGuessed*100/(!g_nDone?1:g_nDone),
                    ↪ g_nFrameDone,g_nFrameMax,g_pszStatus[g_nStatus]);
                if(strcmp(szDisplay,szPrev)){
905                     strcpy(szPrev,szDisplay);
                        SetDlgItemText(hWnd,IDC_EDIT6,szDisplay);
                }
        }
/*      else if(uMsg==WM_TIMER){
910             static char szPev[256]={0};
                char szStatus[256];
                if(g_nFrames==g_SFT.GetFramesDone()){
                        strcpy(szStatus,"Done");
                        g_bZoomRunning=FALSE;
915                     SetDlgItemText(hWnd,IDOK,"Start");
                }
                else
                        wsprintf(szStatus,"Frame %d (%s)",g_SFT.GetFramesDone(),
                            ↪ g_SFT.GetZoomStatus());
                if(strcmp(szPev,szStatus)){
920                     strcpy(szPev,szStatus);
```

```
                        SetDlgItemText(hWnd,IDC_EDIT6,szStatus);
                }
        }
        return 0;
}
*/
struct JPEG_PARAMS
{
        int nWidth;
        int nHeight;
        int nQuality;
        int lParam;
}g_JpegParams;
static int WINAPI JpegProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        if(uMsg==WM_INITDIALOG){
                SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                g_JpegParams.lParam = lParam;
                SetDlgItemInt(hWnd,IDC_EDIT1,g_JpegParams.nWidth,0);
                SetDlgItemInt(hWnd,IDC_EDIT3,g_JpegParams.nHeight,0);
                if(lParam){
                        ShowWindow(GetDlgItem(hWnd,IDC_EDIT4),FALSE);
                        ShowWindow(GetDlgItem(hWnd,IDC_QUALITY_LABEL),FALSE);
                        SetWindowText(hWnd,"Size");
                }
                else
                        SetDlgItemInt(hWnd,IDC_EDIT4,g_JpegParams.nQuality,0);
                if(lParam==1){
                        SetWindowText(hWnd,"Image Size");
                }
                else if(lParam==2){
                        SetWindowText(hWnd,"Set Ratio");
                        SendDlgItemMessage(hWnd,IDC_EDIT1,EM_SETREADONLY,TRUE,0)
                            ↘ ;
                        SendDlgItemMessage(hWnd,IDC_EDIT3,EM_SETREADONLY,FALSE
                            ↘ ,0);
                        SetFocus(GetDlgItem(hWnd,IDC_EDIT3));
                }
                else if(lParam==3){
                        SetWindowText(hWnd,"Window Size");
                        if (g_SFT.GetArbitrarySize())
                                SendDlgItemMessage(hWnd,IDC_EDIT3,EM_SETREADONLY
                                    ↘ ,FALSE,0);
                }
                return 1;
        }
        else if(uMsg==WM_COMMAND){
                if(wParam==IDCANCEL)
                        EndDialog(hWnd,0);
                else if(wParam==IDOK){
                        g_JpegParams.nWidth = GetDlgItemInt(hWnd,IDC_EDIT1,NULL
                            ↘ ,0);
                        g_JpegParams.nHeight = GetDlgItemInt(hWnd,IDC_EDIT3,NULL
                            ↘ ,0);
                        g_JpegParams.nQuality = GetDlgItemInt(hWnd,IDC_EDIT4,
                            ↘ NULL,0);
```

```
                        if (g_JpegParams.lParam == 2)
                        {
                                EndDialog(hWnd, 1);
                                return 0;
                        }
                        if (0 < g_JpegParams.nWidth && g_JpegParams.nWidth <= 65536 &&
                            0 < g_JpegParams.nHeight && g_JpegParams.nHeight <= 65536 &&
                            (uint64_t(g_JpegParams.nWidth) * uint64_t(g_JpegParams.nHeight) * uint64_t(3)) < (uint64_t(1) << uint64_t(31)))
                        {
                                EndDialog(hWnd,1);
                        }
                        else
                        {
                                MessageBox(hWnd, "Invalid dimensions\n\nMaximum width or height is 65536\nMaximum bitmap size is 2GiB\nIn pixels this makes:\n26754x26754 (1:1)\n35673x20066 (16:9)\n",
                                    "Error", MB_OK);
                        }
                }
                else if(LOWORD(wParam)==IDC_EDIT1 && HIWORD(wParam)==EN_CHANGE){
                        //int nHeight = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0)*(double)g_SFT.GetRatioY()/(double)g_SFT.GetRatioX();
                        int nHeight = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0)*(double)g_SFT.GetHeight()/(double)g_SFT.GetWidth();
                        SetDlgItemInt(hWnd,IDC_EDIT3,nHeight,0);
                }
        }
        return 0;
}
struct ANIM
{
        HBITMAP bmBmp;
        HWND hWnd;
        POINT pOffs;
        BOOL bZoomOut;
        BOOL bZoomOne;
        int nZoomSize;
        int nPos;
};
int g_nAnim=0;
bool g_bAnim=false;
static void UpdateBkpImage(ANIM *pAnim)
{
        double zoomDiff = pAnim->nZoomSize;
        if(pAnim->bZoomOut)
                zoomDiff = 1/zoomDiff;
        if(pAnim->bZoomOne)
                zoomDiff = 1;
        RECT r;
        GetClientRect(pAnim->hWnd,&r);
        RECT sr;
```

```
        GetWindowRect(g_hwStatus,&sr);
        sr.bottom-=sr.top;
        r.bottom-=sr.bottom;
        int nToXStart = pAnim->pOffs.x-r.right/(zoomDiff*2);
        int nToYStart = pAnim->pOffs.y-r.bottom/(zoomDiff*2);
        int nToXStop = r.right - (pAnim->pOffs.x+r.right/(zoomDiff*2));
        int nToYStop = r.bottom - (pAnim->pOffs.y+r.bottom/(zoomDiff*2));

        HDC hDC = GetDC(NULL);
        HDC dcBmp = CreateCompatibleDC(hDC);
        HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,pAnim->bmBmp);
        HDC dcBkg = CreateCompatibleDC(hDC);
        HBITMAP bmBkgOld = (HBITMAP)SelectObject(dcBkg,g_SFT.GetBitmap());
        if(pAnim->bZoomOut)
                SetStretchBltMode(dcBkg,HALFTONE);
        StretchBlt(dcBkg,0,0,g_SFT.GetWidth(),g_SFT.GetHeight(),dcBmp,nToXStart,
            ↪ nToYStart,r.right-nToXStop-nToXStart,r.bottom-nToYStop-nToYStart,
            ↪ SRCCOPY);
        SelectObject(dcBkg,bmBkgOld);
        g_SFT.UpdateBitmap();
        DeleteDC(dcBkg);
        SelectObject(dcBmp,bmOld);
        DeleteDC(dcBmp);
        ReleaseDC(NULL,hDC);
}
static int WINAPI ThAnim_(ANIM *pAnim)
{
        int nParts = 10;// * log((double)g_SFT.GetZoomSize())/log((double)2);
        g_bAnim=true;
        int pMyID = InterlockedIncrement((LPLONG)&g_nAnim);
        HDC hDC = GetDC(pAnim->hWnd);
        SetStretchBltMode(hDC,HALFTONE);
        HDC dcBmp = CreateCompatibleDC(hDC);
        HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,pAnim->bmBmp);
        double zoomDiff = pAnim->nZoomSize;
        if(pAnim->bZoomOut)
                zoomDiff = 1/zoomDiff;
        if(pAnim->bZoomOne)
                zoomDiff = 1;
        RECT r;
        GetClientRect(pAnim->hWnd,&r);
        RECT sr;
        GetWindowRect(g_hwStatus,&sr);
        sr.bottom-=sr.top;
        r.bottom-=sr.bottom;
        int nToXStart = pAnim->pOffs.x-r.right/(zoomDiff*2);
        double nToXStartOffs = (double)nToXStart/nParts;//pow(10,log10((double)
            ↪ nToXStart)/(nParts));
        int nToYStart = pAnim->pOffs.y-r.bottom/(zoomDiff*2);
        double nToYStartOffs = (double)nToYStart/nParts;//pow(10,log10((double)
            ↪ nToYStart)/(nParts));
        int nToXStop = r.right - (pAnim->pOffs.x+r.right/(zoomDiff*2));
        double nToXStopOffs = (double)nToXStop/nParts;//pow(10,log10((double)
            ↪ nToXStop)/(nParts));
        int nToYStop = r.bottom - (pAnim->pOffs.y+r.bottom/(zoomDiff*2));
        double nToYStopOffs = (double)nToYStop/nParts;//pow(10,log10((double)
            ↪ nToYStop)/(nParts));
```

```
        double nToXStartPos=0;
        double nToYStartPos=0;
        double nToXStopPos=0;
        double nToYStopPos=0;

        int k;
        for(k=0;pMyID==g_nAnim && k<=nParts;k++){
                StretchBlt(hDC,0,0,r.right,r.bottom,dcBmp,nToXStartPos,↙
                    ↳ nToYStartPos,r.right-nToXStopPos-nToXStartPos,r.bottom-↙
                    ↳ nToYStopPos-nToYStartPos,SRCCOPY);
                if(k==nParts)
                        break;
                nToXStartPos+=nToXStartOffs;
                nToYStartPos+=nToYStartOffs;
                nToXStopPos+=nToXStopOffs;
                nToYStopPos+=nToYStopOffs;
                Sleep(7);
        }
        if(pMyID==g_nAnim && !g_SFT.m_bRunning){ // Render is done during ↙
            ↳ animation
//              g_SFT.ApplyColors();
                InvalidateRect(pAnim->hWnd,NULL,FALSE);
        }
        if(pMyID==g_nAnim)
                g_bAnim=false;
        SelectObject(dcBmp,bmOld);
        DeleteDC(dcBmp);
        ReleaseDC(pAnim->hWnd,hDC);
        DeleteObject(pAnim->bmBmp);
        delete pAnim;
        return 0;
}
static int WINAPI ThAnim(ANIM *pAnim)
{
//#ifndef _DEBUG
        try{
//#endif
                return ThAnim_(pAnim);
//#ifndef _DEBUG
        }catch(...){
                char szPos[1024];
                wsprintf(szPos,"Krash: %d",pAnim->nPos);
                MessageBox(pAnim->hWnd,szPos,"Krash",MB_OK);
        }
//#endif
        mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
        return 0;
}

extern std::string replace_path_filename(const std::string &path, const std::↙
    ↳ string &file)
{
        size_t slash = path.rfind('\\');
        if (slash == std::string::npos)
        {
                return file;
        }
```

```
		else
		{
			return path.substr(0, slash + 1) + file;
		}
	}

	extern std::string replace_path_extension(const std::string &path, const std::↙
	    ↳ string &ext)
	{
		size_t dot = path.rfind('.');
		if (dot == std::string::npos)
		{
			return path + "." + ext;
		}
		else
		{
			return path.substr(0, dot + 1) + ext;
		}
	}

	extern std::string get_filename_path(const std::string &path)
	{
		size_t slash = path.rfind('\\');
		if (slash == std::string::npos)
		{
			return "";
		}
		else
		{
			return path.substr(0, slash + 1);
		}
	}

	extern std::string get_filename_extension(const std::string &path)
	{
		size_t dot = path.rfind('.');
		if (dot == std::string::npos)
		{
			return "";
		}
		else
		{
			return path.substr(dot + 1);
		}
	}

	extern std::string get_filename_zoom_string(const std::string &file)
	{
		size_t begin = file.rfind('_');
		size_t end = file.rfind('.');
		assert(begin != std::string::npos);
		assert(end != std::string::npos);
		assert(begin < end);
		return file.substr(begin + 1, end - (begin + 1));
	}

	extern std::string store_zoom_filename(int n, const std::string &z, const std::↙
```

```
    ↪ string &ext)
{
        std::ostringstream os;
1180    os << std::setfill('0') << std::setw(5) << n << "_" << z << "." << ext;
        return os.str();
}

static void AutoIterations()
1185 {
        if(g_SFT.GetAutoIterations()){
                int64_t nMin, nMax, nIter;
                g_SFT.GetIterations(nMin,nMax);
                // sanity check, abort if no pixels have been calculated
1190            if (nMax == PIXEL_UNEVALUATED)
                        return;
                // sanity increase
                nIter = g_SFT.GetIterations();
                if(nIter<nMax)
1195                    g_SFT.SetIterations(nMax);
                // increase
                nIter = g_SFT.GetIterations();
                if(nIter<nMin+nMin+2000)
                        g_SFT.SetIterations(nMin+nMin+3000);
1200            // decrease
                nMax = g_SFT.GetMaxExceptCenter();
                // sanity check, abort if no pixels have been calculated
                if (nMax == PIXEL_UNEVALUATED)
                        return;
1205            if(nMax<g_SFT.GetIterations()/3)
                        g_SFT.SetIterations(nMax*3>1000?nMax*3:1000);
        }
}

1210 static int ResumeZoomSequence(HWND hWnd)
{
        g_szFile = "";
        if(!BrowseFile(hWnd,TRUE,"Open location","Kalle's fraktaler\0*.kfr\0\0",↙
            ↪ g_szFile))
                return 0;
1215    if(!g_SFT.OpenFile(g_szFile))
                return MessageBox(hWnd,"Could not open file","Error",MB_OK|↙
                    ↪ MB_ICONSTOP);
        g_SFT.StoreLocation();
        std::string File = replace_path_filename(g_szFile, "*_*.jpg");
        WIN32_FIND_DATA fd;
1220    HANDLE hFind = FindFirstFile(File.c_str(),&fd);
        int countJpg = 0;
        if(hFind!=INVALID_HANDLE_VALUE){
                g_bStoreZoomJpg=1;
                do{
1225                    countJpg++;
                }while(FindNextFile(hFind,&fd));
                FindClose(hFind);
        }
        else
1230            g_bStoreZoomJpg=0;
        File = replace_path_filename(g_szFile, "*_*.tif");
```

```
	hFind = FindFirstFile(File.c_str(),&fd);
	int countTif = 0;
	if(hFind!=INVALID_HANDLE_VALUE){
1235		g_bStoreZoomTif=1;
		do{
			countTif++;
		}while(FindNextFile(hFind,&fd));
		FindClose(hFind);
1240	}
	else
		g_bStoreZoomTif=0;
	File = replace_path_filename(g_szFile, "*_*.png");
	hFind = FindFirstFile(File.c_str(),&fd);
1245	int countPng = 0;
	if(hFind!=INVALID_HANDLE_VALUE){
		g_bStoreZoomPng=1;
		do{
			countPng++;
1250		}while(FindNextFile(hFind,&fd));
		FindClose(hFind);
	}
	else
		g_bStoreZoomPng=0;
1255	File = replace_path_filename(g_szFile, "*_*.exr");
	hFind = FindFirstFile(File.c_str(),&fd);
	int countExr = 0;
	if(hFind!=INVALID_HANDLE_VALUE){
		g_bStoreZoomExr=1;
1260		do{
			countExr++;
		}while(FindNextFile(hFind,&fd));
		FindClose(hFind);
	}
1265	else
		g_bStoreZoomExr=0;
	File = replace_path_filename(g_szFile, "*_*.kfb");
	hFind = FindFirstFile(File.c_str(),&fd);
	if(hFind!=INVALID_HANDLE_VALUE){
1270		g_bStoreZoomMap=1;
	}
	else
		g_bStoreZoomMap=0;

1275	std::vector<std::string> stExamine;
	int countMap = 0;
	if(hFind!=INVALID_HANDLE_VALUE){
		do{
			countMap++;
1280			File = replace_path_filename(g_szFile, fd.cFileName);
			stExamine.push_back(File);
		}while(FindNextFile(hFind,&fd));
		FindClose(hFind);
	}
1285	std::sort(stExamine.begin(), stExamine.end());
	std::reverse(stExamine.begin(), stExamine.end());
	BOOL bRecoveryFile=FALSE;
```

```
        File = replace_path_filename(g_szFile, "recovery.kfb");
1290    hFind = FindFirstFile(File.c_str(),&fd);
        if(hFind!=INVALID_HANDLE_VALUE){
                FindClose(hFind);
                g_SFT.OpenMapB(File);
                bRecoveryFile=TRUE;
1295    }
        else{
                File = replace_path_filename(g_szFile, "last.kfb");
                hFind = FindFirstFile(File.c_str(),&fd);
                if(hFind!=INVALID_HANDLE_VALUE){
1300                    FindClose(hFind);
                        g_SFT.OpenMapB(File);
                        bRecoveryFile=FALSE;
                }
                else{
1305                    if(stExamine.size())
                                g_SFT.OpenMapB(stExamine[0]);
                        else
                                return MessageBox(hWnd,"Could not browse kfb files","Error",MB_OK|MB_ICONSTOP);
                }
1310    }

        int zoomCount = countMap ? countMap
                      : countExr ? countExr
                      : countTif ? countTif
1315                  : countPng ? countPng
                      : countJpg ? countJpg
                      : stExamine.size();
        if(zoomCount){
                CDecNumber A = CDecNumber(g_SFT.GetZoom())/(CDecNumber(g_SFT.GetZoomSize())^(zoomCount-(bRecoveryFile?0:1)));
1320            std::string szRe = g_SFT.GetRe();
                std::string szIm = g_SFT.GetIm();
                g_SFT.SetPosition(szRe,szIm,A.ToText());
        }

1325    g_nStoreZoomCount = 0;
        // g_nStoreZoomLimit = 0; // FIXME should there be another dialog?
        g_bStoreZoom=zoomCount+1;
        g_JpegParams.nWidth = g_SFT.GetWidth();
        g_JpegParams.nHeight = g_SFT.GetHeight();
1330    g_JpegParams.nQuality = 100;
        //g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.GetIterations(),hWnd);
        AutoIterations();
        if(bRecoveryFile){
                g_SFT.ToZoom();
1335            g_SFT.AddReference(g_JpegParams.nWidth/2,g_JpegParams.nHeight/2,FALSE,FALSE,TRUE);
        }
        else
        {
                bool bReuseCenter = (g_SFT.GetZoomSize() == round(g_SFT.GetZoomSize()));
1340            g_SFT.Zoom(g_JpegParams.nWidth/2,g_JpegParams.nHeight/2,1/(
```

```
                        ↪ double)g_SFT.GetZoomSize(),g_JpegParams.nWidth,↙
                        ↪ g_JpegParams.nHeight,bReuseCenter/* TRUE */ /* !↙
                        ↪ g_bAutoGlitch */);
        }
        SetTimer(hWnd,0,500,NULL);
        return 0;
}
int g_nHandleDone=0;
int g_bSkewAnimation=0;
double g_dbSkewToRatio=0;
double g_dbSkewToAnim=0;
double g_dbSkewRatio=0;
double g_dbSkewAnim=0;
double g_dbRotateRatio=0;
double g_dbRotateStart=0;
double g_dbRotateAnim=0;
int g_nSkewFrames=300;

static double CompareBitmaps(HBITMAP bm1, HBITMAP bm2)
{
        BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
        BITMAPINFOHEADER bmi2={sizeof(BITMAPINFOHEADER)};
        HDC hDC = GetDC(NULL);
        if(!GetDIBits(hDC,bm1,0,0,NULL,(LPBITMAPINFO)&bmi,DIB_RGB_COLORS))
                return 0;
        if(!GetDIBits(hDC,bm2,0,0,NULL,(LPBITMAPINFO)&bmi2,DIB_RGB_COLORS))
                return 0;
        if(bmi.biWidth!=bmi2.biWidth || bmi.biHeight!=bmi2.biHeight)
                return 0;
        bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
        bmi.biBitCount = 24;
        int row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
        bmi.biSizeImage=row*bmi.biHeight;
        BYTE *lpBits1 = new BYTE[bmi.biSizeImage];
        BYTE *lpBits2 = new BYTE[bmi.biSizeImage];
        if(!GetDIBits(hDC,bm1,0,bmi.biHeight,lpBits1,
                        (LPBITMAPINFO)&bmi,DIB_RGB_COLORS)){
                delete[] lpBits1;
                delete[] lpBits2;
                return 0;
        }
        if(!GetDIBits(hDC,bm2,0,bmi.biHeight,lpBits2,
                        (LPBITMAPINFO)&bmi,DIB_RGB_COLORS)){
                delete[] lpBits1;
                delete[] lpBits2;
                return 0;
        }
        int x, y;
        int nLM=0;
        double ret=0;
        for(y=0;y<bmi.biHeight;y++){
                for(x=0;x<bmi.biWidth;x++){
                        int nIndex = x*3 + (bmi.biHeight-1-y)*row;
                        int C1 = (lpBits1[nIndex] + lpBits1[nIndex+1] + lpBits1[↙
                            ↪ nIndex+2])/3;
                        int C2 = (lpBits2[nIndex] + lpBits2[nIndex+1] + lpBits2[↙
                            ↪ nIndex+2])/3;
```

```
                                nLM+=(C1-C2)*(C1-C2);
                                if(nLM>1000000){
1395                                    int n = (nLM/1000000)*1000000;
                                        ret+=n;
                                        nLM-=n;
                                }
                        }
1400            }
                ret+=nLM;
                delete[] lpBits1;
                delete[] lpBits2;
                return ret;
1405    }

        #if 0
        static int KRYield(HWND hWnd)
        {
1410            UpdateWindow(hWnd);
                MSG msg;
                while(PeekMessage(&msg,hWnd,0,0,PM_REMOVE)){
                        if(msg.message==WM_KEYDOWN && msg.wParam==VK_ESCAPE)
                                return 0;
1415                    if(IsDialogMessage(hWnd,&msg))
                                continue;
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                }
1420            return 1;
        }
        #endif

        static HBITMAP ShrinkBitmap2(HBITMAP bmBmp,int nX, int nY)
1425    {
                HDC hDC = GetDC(NULL);
                HDC dcBmp = CreateCompatibleDC(hDC);
                HDC dcResult = CreateCompatibleDC(hDC);
                BITMAP bm;
1430            GetObject(bmBmp,sizeof(BITMAP),&bm);
                HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
                HBITMAP bmResult = create_bitmap(hDC,nX,nY);
                HBITMAP bmOldResult = (HBITMAP)SelectObject(dcResult,bmResult);
                SetStretchBltMode(dcResult,HALFTONE);
1435            StretchBlt(dcResult,0,0,nX,nY,dcBmp,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
                SelectObject(dcBmp,bmOld);
                SelectObject(dcResult,bmOldResult);
                DeleteDC(dcBmp);
                DeleteDC(dcResult);
1440            ReleaseDC(NULL,hDC);
                return bmResult;
        }
        double g_length=0;
        double g_degree=0;
1445    HBITMAP g_bmSaveZoomBuff=NULL;
        SIZE g_scSaveZoomBuff;
        static void SaveZoomImg(const std::string &szFile, const std::string &comment)
        {
                HBITMAP bmSave;
```

```
1450        HDC hDC = GetDC(NULL);
            HDC dcBmp = CreateCompatibleDC(hDC);
            HBITMAP bmBmp = g_SFT.GetBitmap();
            HBITMAP bmOldBmp = (HBITMAP)SelectObject(dcBmp,bmBmp);
            BITMAP bm;
1455        GetObject(bmBmp, sizeof(BITMAP),&bm);

            SIZE scNextZoom = {LONG(g_scSaveZoomBuff.cx*g_SFT.GetZoomSize()),LONG(↙
                ↳ g_scSaveZoomBuff.cy*g_SFT.GetZoomSize())};
            if(!g_bmSaveZoomBuff){
                    scNextZoom.cx = g_scSaveZoomBuff.cx = bm.bmWidth;
1460                scNextZoom.cy = g_scSaveZoomBuff.cy = bm.bmHeight;
            }
            BOOL bScaled=FALSE;
            if(scNextZoom.cx/bm.bmWidth>2){
                    scNextZoom.cx = bm.bmWidth;
1465                scNextZoom.cy = bm.bmHeight;
                    bScaled=TRUE;
            }

            HBITMAP bmTmp = create_bitmap(hDC,scNextZoom.cx,scNextZoom.cy);
1470        HDC dcSaveZoom = CreateCompatibleDC(hDC);
            HBITMAP dcOldSaveZoom = (HBITMAP)SelectObject(dcSaveZoom,↙
                ↳ g_bmSaveZoomBuff);
            HDC dcTmp = CreateCompatibleDC(hDC);
            HBITMAP bmOldTmp = (HBITMAP)SelectObject(dcTmp,bmTmp);
            SetStretchBltMode(dcTmp,HALFTONE);
1475        StretchBlt(dcTmp,0,0,scNextZoom.cx,scNextZoom.cy,dcBmp,0,0,bm.bmWidth,bm↙
                ↳ .bmHeight,SRCCOPY);

            if(g_bmSaveZoomBuff){
                    SIZE offs={LONG((scNextZoom.cx - scNextZoom.cx/g_SFT.GetZoomSize↙
                        ↳ ())/2),LONG((scNextZoom.cy - scNextZoom.cy/g_SFT.↙
                        ↳ GetZoomSize())/2)};
                    if(bScaled)
1480                        StretchBlt(dcTmp, offs.cx, offs.cy, scNextZoom.cx-2*offs.cx↙
                            ↳ ,scNextZoom.cy-2*offs.cy,
                                dcSaveZoom,0,0,g_scSaveZoomBuff.cx,↙
                                    ↳ g_scSaveZoomBuff.cy,SRCCOPY);
                    else
                            BitBlt(dcTmp, offs.cx, offs.cy, scNextZoom.cx-2*offs.cx,↙
                                ↳ scNextZoom.cy-2*offs.cy,
                                dcSaveZoom,0,0,SRCCOPY);
1485        }
            SelectObject(dcSaveZoom,dcOldSaveZoom);
            DeleteDC(dcSaveZoom);
            SelectObject(dcTmp,bmOldTmp);
            DeleteDC(dcTmp);
1490        SelectObject(dcBmp,bmOldBmp);
            DeleteDC(dcBmp);
            DeleteObject(g_bmSaveZoomBuff);
            ReleaseDC(NULL,hDC);
            g_bmSaveZoomBuff = bmTmp;
1495        bmSave = ShrinkBitmap2(bmTmp,bm.bmWidth,bm.bmHeight);
            SaveImage(szFile,bmSave,100,comment);
            //SaveJpg(szFile,bmTmp,99);
            DeleteObject(bmSave);
```

```
		g_scSaveZoomBuff.cx = scNextZoom.cx;
		g_scSaveZoomBuff.cy = scNextZoom.cy;
}
static int HandleDone(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam,int &nPos)
{
		if(g_bStoreZoom){
			std::string File = replace_path_filename(g_szFile, "recovery.kfb");
			if(uMsg==WM_USER+199)
				DeleteFile(File.c_str());
			else{
				g_nHandleDone++;
				if(g_nHandleDone>60){
					g_nHandleDone=0;
					g_SFT.SaveMapB(File);
				}
			}
		}
nPos=0;
		int nG, nR, nA;
		double nPd = g_SFT.GetProgress(&nG,&nR,&nA);
		int nP = nPd;
		if(!wParam && uMsg==WM_USER+199 && (!g_bAnim || !g_SFT.GetAnimateZoom())){
			g_SFT.ApplyColors();
			InvalidateRect(hWnd,NULL,FALSE);
		}

nPos=1;
		char szTmp[1024];
		wsprintf(szTmp,"%d%% R:%d%% G:%d%% A:%d%%",nP,nR,nG,nA);
		SendMessage(g_hwStatus,SB_SETTEXT,0,(LPARAM)szTmp);
		SYSTEMTIME st;
		__int64 nTStop;
		GetLocalTime(&st);
		SystemTimeToFileTime(&st,(LPFILETIME)&nTStop);
		nTStop-=g_nTStart;
		FileTimeToSystemTime((LPFILETIME)&nTStop,&st);
		if(st.wDay>1)
			st.wHour+=(st.wDay-1)*24;
		std::string z = g_SFT.ToZoom();
		wsprintf(szTmp,"Zoom:%s T:%02d:%02d:%02d.%03d",z.c_str(),st.wHour,st.
		    wMinute,st.wSecond,st.wMilliseconds);
nPos=9;
		if(g_bAutoGlitch){
			wsprintf(szTmp+strlen(szTmp)," Ref: %d",g_bAutoGlitch);
		}
nPos=2;
		if(!g_hwExamine && uMsg==WM_USER+199 && !wParam){
nPos=3;
			if(g_bAutoGlitch && g_bAutoGlitch-1<g_SFT.GetMaxReferences() &&
			    g_SFT.GetAutoSolveGlitches()){
				g_bAutoGlitch++;
nPos=4;
				int x, y, d;

				if((d = g_SFT.FindCenterOfGlitch(x, y))){
```

```
nPos=5;
                                        if(g_nPrevGlitchX!=x || g_nPrevGlitchY!=y){
nPos=6;
1555                                            if (! g_bInteractive)
                                                {
                                                        std::cerr << "add reference " <<↙
                                                            ↳  g_bAutoGlitch << " at (" ↙
                                                            ↳ << x << "," << y << ") ↙
                                                            ↳ area " << (d - 1) << std::↙
                                                            ↳ endl;
                                                        SendMessage(g_hwStatus,↙
                                                            ↳ SB_SETTEXT,1,(LPARAM)szTmp↙
                                                            ↳ );
                                                }
1560                                            if(g_SFT.AddReference(x, y,FALSE,g_SFT.↙
                                                    ↳ GetSolveGlitchNear(),g_bAutoGlitch↙
                                                    ↳ ==g_SFT.GetMaxReferences())){
nPos=7;
                                                        return 0;
                                                }
                                        }
1565                                }
                                    else
                                        g_bAutoGlitch--;
nPos=8;
                        }
1570            }
                if(g_bAutoGlitch){
                        wsprintf(szTmp+strlen(szTmp)," %s", uMsg==WM_USER+199?"Done":"")↙
                            ↳ ;
nPos=10;
                        SendMessage(g_hwStatus,SB_SETTEXT,1,(LPARAM)szTmp);
1575                    if (uMsg==WM_USER+199)
                                g_bAutoGlitch=1;
                }
                if(g_hwExamine && uMsg==WM_USER+199)
                        PostMessage(g_hwExamine,uMsg,wParam,lParam);
1580            if(nPd > 0.0 && (!g_bAnim || !g_SFT.GetAnimateZoom()))
                        InvalidateRect(hWnd,NULL,FALSE);
nPos=11;
                if(uMsg==WM_USER+199){
                        KillTimer(hWnd,0);
1585    //                          g_nAnim++;
    //                          g_bAnim=FALSE;
                        g_SFT.ApplyColors();
                        if(!g_bAnim || !g_SFT.GetAnimateZoom()){
                                InvalidateRect(hWnd,NULL,FALSE);
1590                            UpdateWindow(hWnd);
                        }
nPos=12;
                        if(g_bFindMinibrot){
                                PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_FINDMINIBROT,1);
1595                    }
                        if(g_bStoreZoom){
nPos=13;
                                std::string szZ = g_SFT.ToZoom();
                                if(g_bStoreZoomJpg){
```

```
1600                                    std::string File = replace_path_filename(↙
                                            ↳ g_szFile, store_zoom_filename(g_bStoreZoom↙
                                            ↳ , szZ, "jpg"));
                                        if(g_SFT.GetZoomSize()<2 && !g_bAnimateEachFrame↙
                                            ↳ )
                                                SaveZoomImg(File, "KF2");
                                        else
                                                g_SFT.SaveJpg(File,100);
1605                                }
                                    if(g_bStoreZoomPng){
                                        std::string File = replace_path_filename(↙
                                            ↳ g_szFile, store_zoom_filename(g_bStoreZoom↙
                                            ↳ , szZ, "png"));
                                        if(g_SFT.GetZoomSize()<2 && !g_bAnimateEachFrame↙
                                            ↳ )
                                                SaveZoomImg(File, "KF2");
1610                                    else
                                                g_SFT.SaveJpg(File,-1);
                                    }
                                    if(g_bStoreZoomTif){
                                        std::string File = replace_path_filename(↙
                                            ↳ g_szFile, store_zoom_filename(g_bStoreZoom↙
                                            ↳ , szZ, "tif"));
1615                                    if(g_SFT.GetZoomSize()<2 && !g_bAnimateEachFrame↙
                                            ↳ )
                                                SaveZoomImg(File, "KF2");
                                        else
                                                g_SFT.SaveJpg(File,-2);
                                    }
1620                                if(g_bStoreZoomExr){
                                        std::string File = replace_path_filename(↙
                                            ↳ g_szFile, store_zoom_filename(g_bStoreZoom↙
                                            ↳ , szZ, "exr"));
                                        if(g_SFT.GetZoomSize()<2 && !g_bAnimateEachFrame↙
                                            ↳ )
                                                SaveZoomImg(File, "KF2");
                                        else
1625                                            g_SFT.SaveJpg(File,-3);
                                    }
                                    std::string File = replace_path_filename(g_szFile, ↙
                                        ↳ store_zoom_filename(g_bStoreZoom, szZ, "kfb"));
                                    if(!g_bAnimateEachFrame && g_bStoreZoomMap)
                                        g_SFT.SaveMapB(File);
1630
                                    File = replace_path_filename(g_szFile, "last.kfb");
                                    if(!g_bAnimateEachFrame && !g_bStoreZoomMap)
                                        g_SFT.SaveMapB(File);
    nPos=14;
1635                                g_bStoreZoom++;
                                    g_nStoreZoomCount++;
                                    double szZd = 0.0;
                                    try
                                    {
1640                                    szZd = std::stod(szZ);
                                    }
                                    catch (std::out_of_range &e)
                                    {
```

```
                szZd = 1;
1645        }
            catch (std::invalid_argument &e)
            {
                szZd = 0.0;
            }
1650        if(0.0 == szZd || (g_nStoreZoomLimit && ↙
                ↳ g_nStoreZoomCount >= g_nStoreZoomLimit)){
                g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
                g_bmSaveZoomBuff=NULL;
            }
1655        else{
                AutoIterations();

                if(g_bSkewAnimation){
                    if(g_bSkewAnimation==1){
1660                    g_bSkewAnimation=2;
                        g_dbSkewAnim = g_SFT.GetRatioY()↙
                            ↳ ;

                        g_dbSkewAnim = 100*g_SFT.↙
                            ↳ GetRatioY()/360;
                        SIZE size;
1665                    size.cx = g_SFT.GetWidth();
                        size.cy = g_SFT.GetHeight();
                        double xRatio = 640.0/size.cx;
                        size.cx = 640;
                        size.cy = size.cy*xRatio;
1670                    xRatio = (double)360/(double)↙
                            ↳ size.cy;
                        g_dbSkewAnim*=xRatio;

                        g_dbSkewRatio = pow(10,log10((↙
                            ↳ double)100/g_dbSkewAnim)/(↙
                            ↳ double)g_nSkewFrames);
                        g_dbSkewAnim = g_SFT.GetRatioY()↙
                            ↳ ;
1675                    //g_dbSkewRatio = ((double)360*↙
                            ↳ g_dbSkewToRatio/100-(↙
                            ↳ double)g_dbSkewAnim)/(↙
                            ↳ double)(g_nSkewFrames-1);
                        g_dbRotateRatio = (pi*↙
                            ↳ g_dbSkewToAnim/180-↙
                            ↳ g_Degree)/g_nSkewFrames;
                        /*g_dbRotateStart = g_Degree;
                        g_dbRotateAnim = (g_Degree<0?-↙
                            ↳ g_Degree:g_Degree)+1;
                        g_dbRotateRatio = pow(10,log10(↙
                            ↳ g_dbRotateAnim)/(double)↙
                            ↳ g_nSkewFrames);
1680                    g_dbRotateAnim=g_dbRotateRatio↙
                            ↳ ;*/
                    }
                    g_dbSkewAnim*=g_dbSkewRatio;
                    //g_dbSkewAnim+=g_dbSkewRatio;
                    g_Degree+=g_dbRotateRatio;
```

```
1685                                        /*if(g_Degree>0)
                                                    g_Degree = g_dbRotateStart-(↙
                                                        ↳ g_dbRotateAnim-1);
                                            else
                                                    g_Degree = g_dbRotateStart+(↙
                                                        ↳ g_dbRotateAnim-1);
                                                    */
1690                                        g_dbRotateAnim *= g_dbRotateRatio;
                                            g_nSkewFrames--;
                                            if(g_nSkewFrames==0)
                                                    return 0;
                                            g_SFT.SetRatio(640,g_dbSkewAnim);
1695                                        DisableUnsafeMenus(hWnd);
                                            g_SFT.RenderFractal(g_SFT.GetWidth(),↙
                                                ↳ g_SFT.GetHeight(),g_SFT.↙
                                                ↳ GetIterations(),hWnd);
                                            SetTimer(hWnd,0,500,NULL);
                                    }
                                    else{
1700    //                                  if(g_bAnimateEachFrame)
        //                                          g_Degree+=0.01;
                                            bool bReuseCenter = (g_SFT.GetZoomSize()↙
                                                ↳  == round(g_SFT.GetZoomSize()));
                                            g_SFT.Zoom(g_SFT.GetZoomSize()==1?-↙
                                                ↳ g_JpegParams.nWidth/2:g_JpegParams↙
                                                ↳ .nWidth/2,g_JpegParams.nHeight↙
                                                ↳ /2,1/(double)g_SFT.GetZoomSize(),↙
                                                ↳ g_JpegParams.nWidth,g_JpegParams.↙
                                                ↳ nHeight,!g_bAnimateEachFrame && ↙
                                                ↳ bReuseCenter);
                                    }
1705                                SetTimer(hWnd,0,500,NULL);
                                    return 0;
                            }
        nPos=15;
                    }
1710                if(g_bFirstDone){
        nPos=16;
                            if(!g_LDBL)
                                    MessageBox(hWnd,"The library "
                                    "kf.dll"
1715                                " could not be loaded. You may continue to use ↙
                                        ↳ this application, but the speed of ↙
                                        ↳ rendering images on depths between 1e600 ↙
                                        ↳ and 1e4900 will be significantly slower ↙
                                        ↳ without this library","Missing file",MB_OK↙
                                        ↳ |MB_ICONWARNING);

        nPos=17;
                            if(g_LDBL==2){
                                    MessageBox(hWnd,"The library "
1720                                "kf.dll"
                                    " is old. Please update this file.","Old file",↙
                                        ↳ MB_OK|MB_ICONWARNING);
                            }
                            g_bFirstDone=FALSE;
        nPos=18;
```

```
1725                            HANDLE hFile = CreateFile(g_szRecovery,GENERIC_READ,↙
                                    ↳ FILE_SHARE_READ|FILE_SHARE_WRITE,0,OPEN_EXISTING↙
                                    ↳ ,0,NULL);
                                if(hFile!=INVALID_HANDLE_VALUE && MessageBox(hWnd,"Kalle↙
                                    ↳ 's Fraktaler was not closed properly last session.↙
                                    ↳  Do you want to recover your last location?","↙
                                    ↳ Kalle's Fraktaler",MB_YESNO)==IDYES){
        nPos=19;
                                    CloseHandle(hFile);
                                    g_SFT.Stop();
1730                                g_bAnim=false;
                                    g_SFT.OpenFile(g_szRecovery);
                                    PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                                }
                                else if(hFile!=INVALID_HANDLE_VALUE)
1735                                CloseHandle(hFile);
        nPos=20;
                        }
                        else{
        nPos=21;
1740                        g_SFT.SaveFile(g_szRecovery, true);
                        }

        nPos=22;
                        g_SFT.ApplyColors();
1745                    if(g_bSaveJpeg){
        nPos=23;
                            if (g_bInteractive)
                            {
                                g_bSaveJpeg=FALSE;
1750                            std::string szFile;
                                if(BrowseFile(hWnd,FALSE,"Save as Jpeg","Jpeg↙
                                    ↳ \0*.jpg\0\0",szFile)){
                                    if(!g_SFT.SaveJpg(szFile,g_JpegParams.↙
                                        ↳ nQuality))
                                        MessageBox(hWnd,"File could not↙
                                            ↳ be saved","Error",MB_OK|↙
                                            ↳ MB_ICONSTOP);
                                    PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
1755                            }
                            }
                            else if (uMsg==WM_USER+199 && wParam==0)
                            {
                                g_bSaveJpeg=FALSE;
1760                            char szFile[1024]={0};
                                strncpy(szFile, g_args->sSaveJPG.c_str(), sizeof↙
                                    ↳ (szFile)-1);
                                std::cerr << "saving JPG " << szFile << std::↙
                                    ↳ endl;
                                if(!g_SFT.SaveJpg(szFile,100))
                                    std::cerr << "ERROR in save jpg: " << ↙
                                        ↳ szFile << std::endl;
1765                        }
                        }
                        if(g_bSavePng){
        nPos=24;
                            if (g_bInteractive)
```

```
1770                            {
                                        g_bSavePng=FALSE;
                                        std::string szFile;
                                        if(BrowseFile(hWnd,FALSE,"Save as PNG","PNG\0*.↙
                                            ↳ png\0\0",szFile)){
                                                if(!g_SFT.SaveJpg(szFile,-1))
1775                                                    MessageBox(hWnd,"File could not ↙
                                                        ↳ be saved","Error",MB_OK|↙
                                                        ↳ MB_ICONSTOP);
                                                PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                                        }
                                }
                                else if (uMsg==WM_USER+199 && wParam==0)
1780                            {
                                        g_bSavePng=FALSE;
                                        char szFile[1024]={0};
                                        strncpy(szFile, g_args->sSavePNG.c_str(), sizeof↙
                                            ↳ (szFile)-1);
                                        std::cerr << "saving PNG " << szFile << std::↙
                                            ↳ endl;
1785                                    if(!g_SFT.SaveJpg(szFile,-1))
                                                std::cerr << "ERROR in save png: " << ↙
                                                    ↳ szFile << std::endl;
                                }
                        }
                        if(g_bSaveTif){
1790    nPos=24;
                                if (g_bInteractive)
                                {
                                        g_bSaveTif=FALSE;
                                        std::string szFile;
1795                                    if(BrowseFile(hWnd,FALSE,"Save as TIFF","TIFF↙
                                            ↳ \0*.tif\0\0",szFile)){
                                                if(!g_SFT.SaveJpg(szFile,-2))
                                                        MessageBox(hWnd,"File could not ↙
                                                            ↳ be saved","Error",MB_OK|↙
                                                            ↳ MB_ICONSTOP);
                                                PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                                        }
1800                            }
                                else if (uMsg==WM_USER+199 && wParam==0)
                                {
                                        g_bSaveTif=FALSE;
                                        char szFile[1024]={0};
1805                                    strncpy(szFile, g_args->sSaveTIF.c_str(), sizeof↙
                                            ↳ (szFile)-1);
                                        std::cerr << "saving TIFF " << szFile << std::↙
                                            ↳ endl;
                                        if(!g_SFT.SaveJpg(szFile,-2))
                                                std::cerr << "ERROR in save TIFF: " << ↙
                                                    ↳ szFile << std::endl;
                                }
1810                    }
                        if(g_bSaveExr){
    nPos=24;
                                if (g_bInteractive)
                                {
```

```
1815                                    g_bSaveExr=FALSE;
                                        std::string szFile;
                                        if(BrowseFile(hWnd,FALSE,"Save as EXR","EXR\0*.↙
                                            ↳ exr\0\0",szFile)){
                                                if(!g_SFT.SaveJpg(szFile,-3))
                                                        MessageBox(hWnd,"File could not ↙
                                                            ↳ be saved","Error",MB_OK|↙
                                                            ↳ MB_ICONSTOP);
1820                                            PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                                        }
                                }
                                else if (uMsg==WM_USER+199 && wParam==0)
                                {
1825                                    g_bSaveExr=FALSE;
                                        char szFile[1024]={0};
                                        strncpy(szFile, g_args->sSaveEXR.c_str(), sizeof↙
                                            ↳ (szFile)-1);
                                        std::cerr << "saving EXR " << szFile << std::↙
                                            ↳ endl;
                                        if(!g_SFT.SaveJpg(szFile,-3))
1830                                            std::cerr << "ERROR in save EXR: " << ↙
                                                ↳ szFile << std::endl;
                                }
                        }
                        if(g_bSaveMap){
                                if (g_bInteractive)
1835                            {
                                        g_bSaveMap = FALSE;
                                }
                                else if (uMsg==WM_USER+199 && wParam==0)
                                {
1840                                    g_bSaveMap=FALSE;
                                        char szFile[1024]={0};
                                        strncpy(szFile, g_args->sSaveMap.c_str(), sizeof↙
                                            ↳ (szFile)-1);
                                        std::cerr << "saving KFB " << szFile << std::↙
                                            ↳ endl;
                                        g_SFT.SaveMapB(szFile);
1845                            }
                        }
                        if (uMsg==WM_USER+199 && wParam==0 && !g_bInteractive)
                        {
                                PostQuitMessage(0);
1850                    }
/*                              int nMin, nMax;
                        g_SFT.GetIterations(nMin,nMax);
                        if(nMax<g_SFT.GetIterations())
                                g_SFT.SetIterations(nMax+nMax/3);
1855 */
                        EnableUnsafeMenus(hWnd);
                }
nPos=25;
                return 0;
1860 }
static int HandleDoneSEH(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
                int nPos=0;
```

```
#ifndef _DEBUG
1865            try{
#endif
                        return HandleDone(hWnd,uMsg,wParam,lParam,nPos);
#ifndef _DEBUG
            }catch(...){
1870            }
#endif
            return 0;
}

1875 static int WINAPI CustomZoomSize(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam↙
    ↪ ){
            if(uMsg==WM_INITDIALOG){
                    SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                    SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                    SetDlgItemText(hWnd,IDC_LABEL,"Zoom size(float value)");
1880                SetWindowLongPtr(hWnd,GWLP_USERDATA,lParam);
                    return 1;
            }
            if(uMsg==WM_COMMAND){
                    if(wParam==IDCANCEL)
1885                        EndDialog(hWnd,0);
                    else if(wParam==IDOK){
                            char *szTmp = (char*)GetWindowLongPtr(hWnd,GWLP_USERDATA↙
                                ↪ );
                            GetDlgItemText(hWnd,IDC_EDIT1,szTmp,25);
                            EndDialog(hWnd,1);
1890                    }
            }
            return 0;
}
static int WINAPI StopAtProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
1895 {
            if(uMsg==WM_INITDIALOG){
                    SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                    SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                    if(lParam){
1900                        char *szTitle = (char*)lParam;
                            SetWindowText(hWnd,szTitle);
                            SetDlgItemText(hWnd,IDC_LABEL,szTitle);
                    }
                    return 1;
1905            }
            if(uMsg==WM_COMMAND){
                    if(wParam==IDCANCEL)
                            EndDialog(hWnd,0);
                    else if(wParam==IDOK)
1910                        EndDialog(hWnd,GetDlgItemInt(hWnd,IDC_EDIT1,NULL,0));
            }
            return 0;
}
static void RotateImageAroundPoint(HBITMAP bmBkg,POINT pm)
1915 {
            HDC hDC = GetDC(NULL);
            BYTE *lpBits=NULL;
            BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
```

```
        int row;
1920    GetDIBits(hDC,bmBkg,0,0,NULL,(LPBITMAPINFO)&bmi,DIB_RGB_COLORS);
        bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
        bmi.biBitCount = 24;
        row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
        bmi.biSizeImage=row*bmi.biHeight;
1925    BYTE *lpOrgBits = new BYTE[bmi.biSizeImage];
        GetDIBits(hDC,bmBkg,0,bmi.biHeight,lpOrgBits,
                        (LPBITMAPINFO)&bmi,DIB_RGB_COLORS);
        lpBits = new BYTE[bmi.biSizeImage];
        memset(lpBits,0,bmi.biSizeImage);
1930    int x, y;
        double diagonal = sqrt((double)(bmi.biWidth*bmi.biWidth/4+bmi.biHeight*↙
            ↳ bmi.biHeight/4));
        for(x=0;x<bmi.biWidth;x++){
                for(y=0;y<bmi.biHeight;y++){
                        double ratio = (double)(y-pm.y)/(double)(x-pm.x);
1935                    double degree = -atan(ratio);
                        if(x==pm.x){
                                if(y<pm.y)
                                        degree=-pi/2;
                                else
1940                                    degree=pi/2;
                        }
                        if(x>pm.x)
                                degree-=pi;
                        double dist = sqrt((double)((x-pm.x)*(x-pm.x)+(y-pm.y)*(↙
                            ↳ y-pm.y)))/diagonal;
1945                    int dx = (double)pm.x + (double)(x-pm.x)*cos(degree)*↙
                            ↳ dist
                                + (double)(y-pm.y)*sin(degree)*dist;
                        int dy = (double)pm.y + (double)(y-pm.y)*cos(degree)*↙
                            ↳ dist
                                - (double)(x-pm.x)*sin(degree)*dist;
                        if(dx<=-1 || dy<=-1 || dx>=bmi.biWidth || dy>=bmi.↙
                            ↳ biHeight)
1950                            continue;
                        int nIndex = x*3 + (bmi.biHeight-1-y)*row;
                        int nDIndex = dx*3 + (bmi.biHeight-1-dy)*row;
                        lpBits[nIndex]=lpOrgBits[nDIndex];
                        lpBits[nIndex+1]=lpOrgBits[nDIndex+1];
1955                    lpBits[nIndex+2]=lpOrgBits[nDIndex+2];

                }
        }
        SetDIBits(hDC,bmBkg,0,bmi.biHeight,lpBits,
1960                    (LPBITMAPINFO)&bmi,DIB_RGB_COLORS);
        ReleaseDC(NULL,hDC);
        delete lpOrgBits;
        delete lpBits;
}
1965 static void RotateImage(HBITMAP bmBkg,HBITMAP bmBkgDraw,POINT pm,double nDegree)
{

        double s = sin(nDegree);
        double c = cos(nDegree);
1970    HDC hDC = GetDC(NULL);
```

```
        BYTE *lpBits=NULL;
        BITMAPINFOHEADER bmi={sizeof(BITMAPINFOHEADER)};
        int row;
        GetDIBits(hDC,bmBkg,0,0,NULL,(LPBITMAPINFO)&bmi,DIB_RGB_COLORS);
1975    bmi.biCompression=bmi.biClrUsed=bmi.biClrImportant=0;
        bmi.biBitCount = 24;
        row = ((((bmi.biWidth*(DWORD)bmi.biBitCount)+31)&~31) >> 3);
        bmi.biSizeImage=row*bmi.biHeight;
        BYTE *lpOrgBits = new BYTE[bmi.biSizeImage];
1980    GetDIBits(hDC,bmBkg,0,bmi.biHeight,lpOrgBits,
                  (LPBITMAPINFO)&bmi,DIB_RGB_COLORS);

        BITMAPINFOHEADER bmiDraw={sizeof(BITMAPINFOHEADER)};
        GetDIBits(hDC,bmBkgDraw,0,0,NULL,(LPBITMAPINFO)&bmiDraw,DIB_RGB_COLORS);
1985    bmiDraw.biCompression=bmiDraw.biClrUsed=bmiDraw.biClrImportant=0;
        bmiDraw.biBitCount = 24;
        int rowDraw = ((((bmiDraw.biWidth*(DWORD)bmiDraw.biBitCount)+31)&~31) >>↙
            ↪ 3);
        bmiDraw.biSizeImage=rowDraw*bmiDraw.biHeight;
        lpBits = new BYTE[bmiDraw.biSizeImage];
1990    memset(lpBits,50,bmiDraw.biSizeImage);

        POINT p1, p2;
        for(p1.y=0;p1.y<bmiDraw.biHeight;p1.y++){
                for(p1.x=0;p1.x<bmiDraw.biWidth;p1.x++){
1995                    lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*rowDraw] = ↙
                            ↪ lpOrgBits[0];
                        lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*rowDraw+1] = ↙
                            ↪ lpOrgBits[1];
                        lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*rowDraw+2] = ↙
                            ↪ lpOrgBits[2];

                        p2 = p1;
2000
                        double dx = (double)pm.x + (double)(p2.x-pm.x)*c
                                + (double)(p2.y-pm.y)*s;
                        double dy = (double)pm.y + (double)(p2.y-pm.y)*c
                                - (double)(p2.x-pm.x)*s;
2005                    if(dx<=-1 || dy<=-1)
                                continue;
                        if(dx<0)
                                dx=0;
                        if(dy<0)
2010                            dy=0;
                        p2.x = (int)dx;
                        p2.y = (int)dy;
                        if(p2.x<0 || p2.x>=bmi.biWidth || p2.y<0 || p2.y>=bmi.↙
                            ↪ biHeight)
                                continue;
2015                    int R, Rx, Ry, Rz, G, Gx, Gy, Gz, B, Bx, By, Bz;
                        double X, Xn, Y, Yn, Z, Zn;
                        Xn = dx - (int)dx;
                        X = 1 - Xn;
                        Yn = dy - (int)dy;
2020                    Y = 1 - Yn;
                        Zn = (Xn+Yn)/2;
                        Z = 1 - Zn;
```

```
                R = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*row];
                G = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*row + 1];
2025            B = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*row + 2];
                p2.x++;
                if(p2.x<0 || p2.x>=bmi.biWidth || p2.y<0 || p2.y>=bmi.↙
                    ↳ biHeight){
                    Rx = R;
                    Gx = G;
2030                Bx = B;
                }
                else{
                    Rx = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row];
                    Gx = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row + 1];
2035                Bx = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row + 2];
                }
                p2.x--;
                p2.y++;
                if(p2.x<0 || p2.x>=bmi.biWidth || p2.y<0 || p2.y>=bmi.↙
                    ↳ biHeight){
2040                Ry = R;
                    Gy = G;
                    By = B;
                }
                else{
2045                Ry = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row];
                    Gy = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row + 1];
                    By = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row + 2];
                }
                p2.x++;
2050            if(p2.x<0 || p2.x>=bmi.biWidth || p2.y<0 || p2.y>=bmi.↙
                    ↳ biHeight){
                    Rz = R;
                    Gz = G;
                    Bz = B;
                }
2055            else{
                    Rz = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row];
                    Gz = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row + 1];
                    Bz = lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*↙
                        ↳ row + 2];
                }
2060

                R = (int)(((double)R*X + (double)Rx*Xn + (double)R*Y + (↙
                    ↳ double)Ry*Yn + (double)R*Z + (double)Rz*Zn)/3);
                G = (int)(((double)G*X + (double)Gx*Xn + (double)G*Y + (↙
                    ↳ double)Gy*Yn + (double)G*Z + (double)Gz*Zn)/3);
                B = (int)(((double)B*X + (double)Bx*Xn + (double)B*Y + (↙
                    ↳ double)By*Yn + (double)B*Z + (double)Bz*Zn)/3);
                lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*rowDraw] = R;
```

```
2065                                lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*rowDraw+1] = G;
                                        ↪ ;
                                    lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*rowDraw+2] = B;
                                        ↪ ;

    //                              memcpy(&lpBits[p1.x*3 + (bmiDraw.biHeight-p1.y-1)*;
        ↪ rowDraw],&lpOrgBits[p2.x*3 + (bmi.biHeight-p2.y-1)*row],3);
                            }
2070            }
                SetDIBits(hDC,bmBkgDraw,0,bmiDraw.biHeight,lpBits,
                                    (LPBITMAPINFO)&bmiDraw,DIB_RGB_COLORS);
                delete lpBits;
                delete lpOrgBits;
2075            ReleaseDC(NULL,hDC);
    }
    static void SkewImage(HBITMAP bmBmp)
    {
                HDC hDC = GetDC(NULL);
2080            BITMAP bm;
                GetObject(bmBmp, sizeof(BITMAP),&bm);
                HBITMAP bmNew = create_bitmap(hDC,bm.bmWidth,bm.bmHeight);
                POINT pm = {bm.bmWidth/2,bm.bmHeight/2};
                double r=pi*(double)g_nSkewRotate/180;
2085            RotateImage(bmBmp,bmNew,pm,r);

                int nWidth = g_nSkewStretch*bm.bmWidth/100;
                HDC dcNew = CreateCompatibleDC(hDC);
                HBITMAP bmOldNew = (HBITMAP)SelectObject(dcNew,bmNew);
2090            HDC dcBmp = CreateCompatibleDC(hDC);
                HBITMAP bmOldBmp = (HBITMAP)SelectObject(dcBmp,bmBmp);
                SetStretchBltMode(dcBmp,HALFTONE);

                RECT rc = {0,0,bm.bmWidth,bm.bmHeight};
2095            FillRect(dcBmp,&rc,(HBRUSH)GetStockObject(BLACK_BRUSH));
                StretchBlt(dcBmp,(bm.bmWidth-nWidth)/2,0,nWidth,bm.bmHeight,dcNew,0,0,bm;
                    ↪ .bmWidth,bm.bmHeight,SRCCOPY);
                SelectObject(dcNew,bmOldNew);
                SelectObject(dcBmp,bmOldBmp);
                DeleteDC(dcNew);
2100            DeleteDC(dcBmp);
                DeleteObject(bmNew);
                ReleaseDC(NULL,hDC);
    }
    static void UnSkewImage(HBITMAP bmBmp)
2105{
                HDC hDC = GetDC(NULL);
                BITMAP bm;
                GetObject(bmBmp, sizeof(BITMAP),&bm);
                HBITMAP bmNew = create_bitmap(hDC,bm.bmWidth,bm.bmHeight);
2110
                int nWidth = (10000/g_nSkewStretch)*bm.bmWidth/100;
                HDC dcNew = CreateCompatibleDC(hDC);
                HBITMAP bmOldNew = (HBITMAP)SelectObject(dcNew,bmNew);
                HDC dcBmp = CreateCompatibleDC(hDC);
2115            HBITMAP bmOldBmp = (HBITMAP)SelectObject(dcBmp,bmBmp);
                SetStretchBltMode(dcNew,HALFTONE);
```

```
            RECT rc = {0,0,bm.bmWidth,bm.bmHeight};
            FillRect(dcNew,&rc,(HBRUSH)GetStockObject(BLACK_BRUSH));
2120        StretchBlt(dcNew,(bm.bmWidth-nWidth)/2,0,nWidth,bm.bmHeight,dcBmp,0,0,bm↙
                ↳ .bmWidth,bm.bmHeight,SRCCOPY);
            SelectObject(dcNew,bmOldNew);
            SelectObject(dcBmp,bmOldBmp);
            DeleteDC(dcNew);
            DeleteDC(dcBmp);
2125        POINT pm = {bm.bmWidth/2,bm.bmHeight/2};
            double r=-pi*(double)g_nSkewRotate/180;
            RotateImage(bmNew,bmBmp,pm,r);

            DeleteObject(bmNew);
2130        ReleaseDC(NULL,hDC);
    }
    BOOL g_DialogInit=0;
    POINT g_Cross[4];
    int g_nCrossPos=0;
2135 double g_nTestDegree;
    double g_nTestRatio;
    static int WINAPI SkewProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
    {
        if(uMsg==WM_INITDIALOG){
2140            SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,3);
                SetWindowLongPtr(hWnd,GWLP_USERDATA,lParam);
                SendDlgItemMessage(hWnd,IDC_SPIN1,UDM_SETRANGE,0,MAKELONG↙
                    ↳ (10000,1));
2145            SetDlgItemFloat(hWnd,IDC_EDIT1,g_nSkewStretch);
                SendDlgItemMessage(hWnd,IDC_SPIN2,UDM_SETRANGE,0,MAKELONG↙
                    ↳ (360,-360));
                SetDlgItemFloat(hWnd,IDC_EDIT2,g_nSkewRotate);
                g_DialogInit=1;
                g_nCrossPos=0;
2150            return 1;
        }
        if(uMsg==WM_NOTIFY){
            LPNMUPDOWN lpnmud = (LPNMUPDOWN) lParam;
            if(lpnmud->hdr.code!=UDN_DELTAPOS)
2155            return 0;
            if(lpnmud->hdr.idFrom==IDC_SPIN1)
                SetDlgItemFloat(hWnd,IDC_EDIT1,GetDlgItemFloat(hWnd,↙
                    ↳ IDC_EDIT1)+lpnmud->iDelta);
            else if(lpnmud->hdr.idFrom==IDC_SPIN2)
                SetDlgItemFloat(hWnd,IDC_EDIT2,GetDlgItemFloat(hWnd,↙
                    ↳ IDC_EDIT2)+lpnmud->iDelta);
2160        return 1;
        }
        if(uMsg==WM_COMMAND){
            if(wParam==IDOK)
                EndDialog(hWnd,1);
2165        else if(wParam==IDCANCEL)
                EndDialog(hWnd,0);
            else if(wParam==IDC_BUTTON1){
                SetDlgItemInt(hWnd,IDC_EDIT1,100,FALSE);
                SetDlgItemInt(hWnd,IDC_EDIT2,0,TRUE);
```

```
2170                    }
                        else if(wParam==IDC_CHECK1){
                                if(SendDlgItemMessage(hWnd,IDC_CHECK1,BM_GETCHECK,0,0)){
                                        g_nCrossPos=0;
                                        SetCapture(hWnd);
2175                            }
                                else
                                        ReleaseCapture();
                        }
                        else if(g_DialogInit){
2180                            g_nSkewStretch = GetDlgItemFloat(hWnd,IDC_EDIT1);
                                g_nSkewRotate = GetDlgItemFloat(hWnd,IDC_EDIT2);
                               HWND hwParent = (HWND)GetWindowLongPtr(hWnd,↙
                                    ↳ GWLP_USERDATA);
                                if(hwParent){
                                        InvalidateRect(hwParent,NULL,FALSE);
2185                                    UpdateWindow(hwParent);
                                        RECT rC;
                                        GetClientRect(hwParent,&rC);
                                        POINT pm;
                                        pm.x = rC.right/2;
2190                                    pm.y = rC.bottom/2;

                                        HDC hDC = GetDC(hwParent);
                                        HPEN pn = CreatePen(0,2,RGB(0,0,255));
                                        HPEN pnOld = (HPEN)SelectObject(hDC,pn);
2195                                    int i;
                                        double nDegree = -pi*(double)g_nSkewRotate/(↙
                                            ↳ double)180;
                                        double s = sin(nDegree);
                                        double c = cos(nDegree);
                                        double nStrecth = (double)g_nSkewStretch/(double↙
                                            ↳ )100;
2200                                    POINT p[4];
                                        for(i=0;i<g_nCrossPos;i++){
                                                POINT pt = g_Cross[i];
                                                p[i].x = (double)pm.x + (double)(pt.x-pm↙
                                                    ↳ .x)*c
                                                        + (double)(pt.y-pm.y)*s;
2205                                            p[i].y = (double)pm.y + (double)(pt.y-pm↙
                                                    ↳ .y)*c
                                                        - (double)(pt.x-pm.x)*s;
                                                p[i].x = (p[i].x-pm.x)*nStrecth + pm.x;

                                                MoveToEx(hDC,p[i].x-2,p[i].y-2,NULL);
2210                                            LineTo(hDC,p[i].x+2,p[i].y+2);
                                                MoveToEx(hDC,p[i].x+2,p[i].y-2,NULL);
                                                LineTo(hDC,p[i].x-2,p[i].y+2);
                                                if(i%2==1){
                                                        POINT pt = g_Cross[i-1];
2215                                                    p[i-1].x = (double)pm.x + (↙
                                                            ↳ double)(pt.x-pm.x)*c
                                                                + (double)(pt.y-pm.y)*s;
                                                        p[i-1].y = (double)pm.y + (↙
                                                            ↳ double)(pt.y-pm.y)*c
                                                                - (double)(pt.x-pm.x)*s;
                                                        p[i-1].x = (p[i-1].x-pm.x)*↙
```

```
                                                    ↪ nStrecth + pm.x;
2220                                                MoveToEx(hDC,p[i-1].x,p[i-1].y,↙
                                                    ↪ NULL);
                                                    LineTo(hDC,p[i].x,p[i].y);
                                            }
                                    }
                                    if(g_nCrossPos==4){
2225                                        double b1 =  (double)(p[0].y - p[1].y) /↙
                                            ↪  (double)(p[0].x - p[1].x==0?1:p↙
                                            ↪ [0].x - p[1].x);
                                            double b2 =  (double)(p[2].y - p[3].y) /↙
                                            ↪  (double)(p[2].x - p[3].x==0?1:p↙
                                            ↪ [2].x - p[3].x);
                                            double a1 = p[0].y - b1*p[0].x;
                                            double a2 = p[2].y - b2*p[2].x;
                                            POINT pM;
2230                                        pM.x = - (a1 - a2) / (b1 - b2) + .5;
                                            pM.y = a1 + b1*pM.x + .5;

                                            double distance1 = sqrt((double)(p[0].x-↙
                                            ↪ p[1].x)*(p[0].x-p[1].x) + (double)↙
                                            ↪ (p[0].y-p[1].y)*(p[0].y-p[1].y));
                                            double distance2 = sqrt((double)(p[2].x-↙
                                            ↪ p[3].x)*(p[2].x-p[3].x) + (double)↙
                                            ↪ (p[2].y-p[3].y)*(p[2].y-p[3].y));
2235
                                            double ratio = (double)(p[0].y-pM.y)/(↙
                                            ↪ double)(p[0].x-pM.x);
                                            double nDegree1 = atan(ratio);
                                            if((p[0].x-pM.x>=0 && p[0].y-pM.y<0) || ↙
                                            ↪ (p[0].x-pM.x<0 && p[0].y-pM.y>0))
                                                nDegree1+=pi;
2240                                        ratio = (double)(p[2].y-pM.y)/(double)(p↙
                                            ↪ [2].x-pM.x);
                                            double nDegree2 = atan(ratio);
                                            if((p[2].x-pM.x>=0 && p[2].y-pM.y<0) || ↙
                                            ↪ (p[2].x-pM.x<0 && p[2].y-pM.y>0))
                                                nDegree2+=pi;
                                            double nDegree = 180*(nDegree1-nDegree2)↙
                                            ↪ /pi;
2245                                        if(nDegree<0)
                                                nDegree=-nDegree;
                                            char szDegree[130];
                                            sprintf(szDegree,"Degree: %.2f, Ratio: ↙
                                            ↪ %.2f",nDegree,distance1/distance2)↙
                                            ↪ ;
                                            TextOut(hDC,0,0,szDegree,strlen(szDegree↙
                                            ↪ ));
2250                                        g_nTestDegree = nDegree;
                                            g_nTestRatio = distance1/distance2;
                                    }

                                    SelectObject(hDC,pnOld);
2255                                DeleteObject(pn);
                                    ReleaseDC(hwParent,hDC);
                                }
                            }
```

```
        }
        else if(uMsg==WM_TIMER){
                SkewProc(hWnd,WM_COMMAND,0,0);
                double nTestDegree = g_nTestDegree;
#ifdef _TEST_RATIO
                double nTestRatio = g_nTestRatio;
#endif
                double nSkewStretch = GetDlgItemFloat(hWnd,IDC_EDIT1);
                double nSkewRotate = GetDlgItemFloat(hWnd,IDC_EDIT2);
                double nTestStretch = nSkewStretch;
                double nTestRotate = nSkewRotate;
                double nBestStretch = nSkewStretch;
                double nBestRotate = nSkewRotate;
                double nStretchOffs = nTestStretch/40;
                double nRotateOffs = .2;
                int bSet=0;
                if(nStretchOffs==0)
                        nStretchOffs=1;
                BOOL bSetValue=0;

                if(g_nTestDegree>85 && g_nTestDegree<95){
                        nStretchOffs/=2;
                        nRotateOffs/=2;
                }
                if(g_nTestDegree>89 && g_nTestDegree<91){
                        nStretchOffs/=2;
                        nRotateOffs/=2;
                }

                nTestStretch-=nStretchOffs;
                SetDlgItemFloat(hWnd,IDC_EDIT1,nTestStretch);
                SkewProc(hWnd,WM_COMMAND,0,0);

                bSet=0;
                if(nTestDegree<90 && g_nTestDegree<90 && g_nTestDegree>↙
                    ↳ nTestDegree)
                        bSet=1;
                if(nTestDegree>90 && g_nTestDegree>90 && g_nTestDegree<↙
                    ↳ nTestDegree)
                        bSet=1;
#ifdef _TEST_RATIO
                if(nTestRatio<1 && g_nTestRatio>nTestRatio)
                        bSet++;
                if(nTestRatio>1 && g_nTestRatio<nTestRatio)
                        bSet++;
#endif
                if(bSet){
                        bSetValue=TRUE;
                        nTestDegree = g_nTestDegree;
                        nBestStretch = nTestStretch;
                        nBestRotate = nTestRotate;
                }

                nTestRotate-=nRotateOffs;
                SetDlgItemFloat(hWnd,IDC_EDIT2,nTestRotate);
                SkewProc(hWnd,WM_COMMAND,0,0);
                bSet=0;
```

```
                    if(nTestDegree<90 && g_nTestDegree<90 && g_nTestDegree>↙
                        ↳ nTestDegree)
2315                        bSet=1;
                    if(nTestDegree>90 && g_nTestDegree>90 && g_nTestDegree<↙
                        ↳ nTestDegree)
                            bSet=1;
    #ifdef _TEST_RATIO
                    if(nTestRatio<1 && g_nTestRatio>nTestRatio)
2320                        bSet++;
                    if(nTestRatio>1 && g_nTestRatio<nTestRatio)
                            bSet++;
    #endif
                    if(bSet){
2325                        bSetValue=TRUE;
                            nTestDegree = g_nTestDegree;
                            nBestStretch = nTestStretch;
                            nBestRotate = nTestRotate;
                    }
2330
                    nTestStretch+=2*nStretchOffs;
                    SetDlgItemFloat(hWnd,IDC_EDIT1,nTestStretch);
                    SkewProc(hWnd,WM_COMMAND,0,0);
                    bSet=0;
2335                if(nTestDegree<90 && g_nTestDegree<90 && g_nTestDegree>↙
                        ↳ nTestDegree)
                            bSet=1;
                    if(nTestDegree>90 && g_nTestDegree>90 && g_nTestDegree<↙
                        ↳ nTestDegree)
                            bSet=1;
    #ifdef _TEST_RATIO
2340                if(nTestRatio<1 && g_nTestRatio>nTestRatio)
                            bSet++;
                    if(nTestRatio>1 && g_nTestRatio<nTestRatio)
                            bSet++;
    #endif
2345                if(bSet){
                            bSetValue=TRUE;
                            nTestDegree = g_nTestDegree;
                            nBestStretch = nTestStretch;
                            nBestRotate = nTestRotate;
2350                }

                    nTestRotate+=2*nRotateOffs;
                    SetDlgItemFloat(hWnd,IDC_EDIT2,nTestRotate);
                    SkewProc(hWnd,WM_COMMAND,0,0);
2355                bSet=0;
                    if(nTestDegree<90 && g_nTestDegree<90 && g_nTestDegree>↙
                        ↳ nTestDegree)
                            bSet=1;
                    if(nTestDegree>90 && g_nTestDegree>90 && g_nTestDegree<↙
                        ↳ nTestDegree)
                            bSet=1;
2360    #ifdef _TEST_RATIO
                    if(nTestRatio<1 && g_nTestRatio>nTestRatio)
                            bSet++;
                    if(nTestRatio>1 && g_nTestRatio<nTestRatio)
                            bSet++;
```

```
2365 #endif
                    if(bSet){
                            bSetValue=TRUE;
                            nTestDegree = g_nTestDegree;
                            nBestStretch = nTestStretch;
2370                        nBestRotate = nTestRotate;
                    }

                    if(bSetValue){
                            SetDlgItemFloat(hWnd,IDC_EDIT1,nBestStretch);
2375                        SetDlgItemFloat(hWnd,IDC_EDIT2,nBestRotate);
                            SkewProc(hWnd,WM_COMMAND,0,0);
                    }
                    else{
                            SetDlgItemFloat(hWnd,IDC_EDIT1,nSkewStretch);
2380                        SetDlgItemFloat(hWnd,IDC_EDIT2,nSkewRotate);
                            KillTimer(hWnd,0);
                            SetDlgItemText(hWnd,IDC_CHECK1,"Help lines");
                    }
                    if(g_nTestDegree>89.5 && g_nTestDegree<91.5){
2385                        KillTimer(hWnd,0);
                            SetDlgItemText(hWnd,IDC_CHECK1,"Help lines");
                    }
            }
            else if(uMsg==WM_CAPTURECHANGED && SendDlgItemMessage(hWnd,IDC_CHECK1,↙
                ↪ BM_GETCHECK,0,0)){
2390            SendDlgItemMessage(hWnd,IDC_CHECK1,BM_SETCHECK,0,0);
            }
            else if(uMsg==WM_LBUTTONDOWN && SendDlgItemMessage(hWnd,IDC_CHECK1,↙
                ↪ BM_GETCHECK,0,0)){
                HWND hwParent = (HWND)GetWindowLongPtr(hWnd,GWLP_USERDATA);
                g_Cross[g_nCrossPos].x = (short)LOWORD(lParam);
2395            g_Cross[g_nCrossPos].y = (short)HIWORD(lParam);
                ClientToScreen(hWnd,&g_Cross[g_nCrossPos]);
                ScreenToClient(hwParent,&g_Cross[g_nCrossPos]);

                HDC hDC = GetDC(hwParent);
2400            HPEN pn = CreatePen(0,2,RGB(0,0,255));
                HPEN pnOld = (HPEN)SelectObject(hDC,pn);
                MoveToEx(hDC,g_Cross[g_nCrossPos].x-2,g_Cross[g_nCrossPos].y-2,↙
                    ↪ NULL);
                LineTo(hDC,g_Cross[g_nCrossPos].x+2,g_Cross[g_nCrossPos].y+2);
                MoveToEx(hDC,g_Cross[g_nCrossPos].x+2,g_Cross[g_nCrossPos].y-2,↙
                    ↪ NULL);
2405            LineTo(hDC,g_Cross[g_nCrossPos].x-2,g_Cross[g_nCrossPos].y+2);
                if(g_nCrossPos%2==1){
                        MoveToEx(hDC,g_Cross[g_nCrossPos-1].x,g_Cross[↙
                            ↪ g_nCrossPos-1].y,NULL);
                        LineTo(hDC,g_Cross[g_nCrossPos].x,g_Cross[g_nCrossPos].y↙
                            ↪ );
                }
2410
                g_nCrossPos++;
                if(g_nCrossPos==4){
                        SendDlgItemMessage(hWnd,IDC_CHECK1,BM_SETCHECK,0,0);
                        ReleaseCapture();
2415                    POINT pM;
```

```
                                double b1 =  (double)(g_Cross[0].y - g_Cross[1].y) / (↙
                                    ↳ double)(g_Cross[0].x - g_Cross[1].x);
                                double b2 =  (double)(g_Cross[2].y - g_Cross[3].y) / (↙
                                    ↳ double)(g_Cross[2].x - g_Cross[3].x);
                                double a1 = g_Cross[0].y - b1*g_Cross[0].x;
2420                            double a2 = g_Cross[2].y - b2*g_Cross[2].x;

                                pM.x = - (a1 - a2) / (b1 - b2) + .5;
                                pM.y = a1 + b1*pM.x + .5;

2425                            SelectObject(hDC,pnOld);
                                DeleteObject(pn);
                                pn = CreatePen(0,2,RGB(255,0,0));
                                pnOld = (HPEN)SelectObject(hDC,pn);
                                MoveToEx(hDC,pM.x-2,pM.y-2,NULL);
2430                            LineTo(hDC,pM.x+2,pM.y+2);
                                MoveToEx(hDC,pM.x+2,pM.y-2,NULL);
                                LineTo(hDC,pM.x-2,pM.y+2);

                                double nStrecth = (double)g_nSkewStretch/(double)100;
2435                            double nDegree = pi*(double)g_nSkewRotate/(double)180;
                                double s = sin(nDegree);
                                double c = cos(nDegree);
                                RECT rC;
                                GetClientRect(hwParent,&rC);
2440                            POINT pm;
                                pm.x = rC.right/2;
                                pm.y = rC.bottom/2;
                                int i;
                                for(i=0;i<g_nCrossPos;i++){
2445                                    POINT pt = g_Cross[i];
                                        pt.x = (pt.x-pm.x)/nStrecth + pm.x;
                                        g_Cross[i].x = (double)pm.x + (double)(pt.x-pm.x↙
                                            ↳ )*c
                                                + (double)(pt.y-pm.y)*s;
                                        g_Cross[i].y = (double)pm.y + (double)(pt.y-pm.y↙
                                            ↳ )*c
2450                                            - (double)(pt.x-pm.x)*s;
                                }
                                SetTimer(hWnd,0,100,NULL);
                                SetDlgItemText(hWnd,IDC_CHECK1,"Working");
                        }
2455                    SelectObject(hDC,pnOld);
                        DeleteObject(pn);
                        ReleaseDC(hwParent,hDC);
                }
                return 0;
2460    }
        static int WINAPI SkewAnimateProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM ↙
            ↳ lParam)
        {
                (void) lParam;
                if(uMsg==WM_INITDIALOG){
2465                    SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                        SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                        SetDlgItemText(hWnd,IDC_EDIT1,"400");
```

```
                SetDlgItemText(hWnd,IDC_EDIT3,"100");
                SetDlgItemText(hWnd,IDC_EDIT4,"0");
2470            return 1;
        }
        if(uMsg==WM_COMMAND){
                if(wParam==IDCANCEL)
                        EndDialog(hWnd,0);
2475            else if(wParam==IDOK){
                        g_nSkewFrames = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,FALSE)↙
                            ↳ ;
                        if(g_nSkewFrames<1)
                                g_nSkewFrames=1;
                        g_dbSkewToRatio = GetDlgItemFloat(hWnd,IDC_EDIT3);
2480                    g_dbSkewToAnim = GetDlgItemFloat(hWnd,IDC_EDIT4);
                        EndDialog(hWnd,1);
                }
        }
        return 0;
2485 }

#ifdef KF_OPENCL
LRESULT CALLBACK OpenCLProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
2490   switch(msg)
       {
         case WM_INITDIALOG:
                {
                        SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon↙
                            ↳ ));
2495                    SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon))↙
                            ↳ ;
                        InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,1);

                        SendDlgItemMessage(hWnd, IDC_COMBO_OPENCL_DEVICE, ↙
                            ↳ CB_ADDSTRING, 0, (LPARAM) "(none)");
                        for (int i = 0; i < cldevices.size(); ++i)
2500                    {
                                cldevice d = cldevices[i];
                                SendDlgItemMessage(hWnd, IDC_COMBO_OPENCL_DEVICE↙
                                    ↳ , CB_ADDSTRING, 0, (LPARAM) d.name.c_str()↙
                                    ↳ );
                        }

2505                    SIZE sc;
                        HDC hDC = GetDC(NULL);
                        HFONT hfOld = (HFONT)SelectObject(hDC,(HFONT)↙
                            ↳ GetStockObject(ANSI_VAR_FONT));
                        int i, nMaxWidth=0;
                        for(i=0;i<SendDlgItemMessage(hWnd,↙
                            ↳ IDC_COMBO_OPENCL_DEVICE,CB_GETCOUNT,0,0);i++){
2510                        int n = SendDlgItemMessage(hWnd,↙
                                ↳ IDC_COMBO_OPENCL_DEVICE,CB_GETLBTEXTLEN,i↙
                                ↳ ,0);
                            char *szT = new char[n+1];
                            SendDlgItemMessage(hWnd,IDC_COMBO_OPENCL_DEVICE,↙
                                ↳ CB_GETLBTEXT,i,(LPARAM)szT);
                            GetTextExtentPoint32A(hDC,szT,strlen(szT),&sc);
```

```
                                        if(sc.cx>nMaxWidth)
2515                                            nMaxWidth = sc.cx;
                                        delete szT;
                                }
                                SendDlgItemMessage(hWnd,IDC_COMBO_OPENCL_DEVICE,↙
                                    ↳ CB_SETDROPPEDWIDTH,nMaxWidth+8+GetSystemMetrics(↙
                                    ↳ SM_CXHTHUMB),0);

2520                            SendDlgItemMessage(hWnd, IDC_COMBO_OPENCL_DEVICE, ↙
                                    ↳ CB_SETCURSEL, g_SFT.GetOpenCLDeviceIndex() + 1, 0)↙
                                    ↳ ;
                          break;
                        }
                        case WM_COMMAND:
                        {
2525                            if (wParam == IDOK)
                                {
                                        g_SFT.SetOpenCLDeviceIndex(SendDlgItemMessage(↙
                                            ↳ hWnd, IDC_COMBO_OPENCL_DEVICE, ↙
                                            ↳ CB_GETCURSEL, 0, 0) - 1);
                                        ExitToolTip(hWnd);
                                        EndDialog(hWnd, 0);
2530                            }
                                else if(wParam == IDCANCEL)
                                {
                                        ExitToolTip(hWnd);
                                        EndDialog(hWnd, 0);
2535                            }
                                break;
                        }
                }
                return 0;
2540    }
        #endif

        static long OpenFile(HWND hWnd, bool &ret)
        {
2545                                    g_SFT.UndoStore();
                                        g_SFT.Stop();
                                        g_bAnim=false;
                                        if(!g_SFT.OpenFile(g_szFile))
                                        {
2550                                            ret = true;
                                                if (hWnd)
                                                        return MessageBox(hWnd,"Invalid ↙
                                                            ↳ parameter file","Error",↙
                                                            ↳ MB_OK|MB_ICONSTOP);
                                                else
                                                        return 0;
2555                                    }
                                        else{
                                                std::string extension = ↙
                                                    ↳ get_filename_extension(g_szFile);
                                                if (extension != "kfr")
                                                {
2560                                                    // prevent ctrl-s save ↙
                                                            ↳ overwriting a file with ↙
```

```
                ↳ the wrong extension
            g_szFile += ".kfr";
        }
        if(g_hwColors)
            SendMessage(g_hwColors,WM_USER↙
                ↳ +99,0,0);
2565    char szTitle[1024];
        wsprintf(szTitle,"Kalle's Fraktaler 2 - ↙
            ↳ %s",g_szFile.c_str());
        if (hWnd)
        {
            SetWindowText(hWnd, szTitle);
2570    }
        if (g_SFT.GetDifferences() == ↙
            ↳ Differences_Analytic && !g_SFT.↙
            ↳ GetDerivatives())
        {
            if (hWnd)
            {
2575            if (IDOK == MessageBox(↙
                    ↳ hWnd,
                    "This parameter ↙
                        ↳ file ↙
                        ↳ requests ↙
                        ↳ analytic ↙
                        ↳ DE ↙
                        ↳ colouring↙
                        ↳ ,\n"
                    "but derivative ↙
                        ↳ calculations↙
                        ↳  are ↙
                        ↳ disabled.\↙
                        ↳ n"
                    "Derivative ↙
                        ↳ calculations↙
                        ↳  are ↙
                        ↳ slower but↙
                        ↳  needed ↙
                        ↳ for ↙
                        ↳ analytic ↙
                        ↳ DE.\n"
                    "You may switch ↙
                        ↳ to non-↙
                        ↳ analytic ↙
                        ↳ DE in the ↙
                        ↳ Colors ↙
                        ↳ dialog.\n"
2580                "You may control↙
                        ↳  ↙
                        ↳ derivatives↙
                        ↳  in the ↙
                        ↳ Iterations↙
                        ↳  dialog.\n↙
                        ↳ "
                    "\n"
                    "Enable ↙
                        ↳ derivatives↙
```

```
                                                                    ↪  ↙
                                                                    ↪ calculation↙
                                                                    ↪  now?",
                                                                "Kalle's ↙
                                                                    ↪ Fraktaler↙
                                                                    ↪ ",
                                                                MB_OKCANCEL))
2585                                                        {
                                                                g_SFT.↙
                                                                    ↪ SetDerivatives↙
                                                                    ↪ (true);
                                                            }
                                                    }
                                                    else
2590                                                {
                                                            output_log_message(Warn,↙
                                                                ↪  "automatically ↙
                                                                ↪ enabling ↙
                                                                ↪ derivatives for ↙
                                                                ↪ analytic DE");
                                                            g_SFT.SetDerivatives(↙
                                                                ↪ true);
                                                    }
                                            }
2595                                        else if (g_SFT.GetDifferences() != ↙
                                                ↪ Differences_Analytic && g_SFT.↙
                                                ↪ GetDerivatives())
                                            {
                                                    if (hWnd)
                                                    {
                                                            if (IDOK == MessageBox(↙
                                                                ↪ hWnd,
2600                                                            "Derivatives ↙
                                                                    ↪ calculations↙
                                                                    ↪  are ↙
                                                                    ↪ enabled, ↙
                                                                    ↪ but\n"
                                                                "this parameter ↙
                                                                    ↪ file does ↙
                                                                    ↪ not ↙
                                                                    ↪ request ↙
                                                                    ↪ analytic ↙
                                                                    ↪ DE ↙
                                                                    ↪ colouring↙
                                                                    ↪ .\n"
                                                                "Derivatives ↙
                                                                    ↪ calculations↙
                                                                    ↪  are ↙
                                                                    ↪ slower but↙
                                                                    ↪  are ↙
                                                                    ↪ needed for↙
                                                                    ↪  analytic ↙
                                                                    ↪ DE,\n"
                                                                "if you wish to ↙
                                                                    ↪ switch to ↙
                                                                    ↪ it in the ↙
                                                                    ↪ Colors ↙
```

```
                                                                    ↳ dialog.\n"
                                                                "You may control↙
                                                                    ↳  ↙
                                                                    ↳ derivatives↙
                                                                    ↳  in the ↙
                                                                    ↳ Iterations↙
                                                                    ↳  dialog.\n↙
                                                                    ↳ "
2605                                                                "\n"
                                                                "Disable ↙
                                                                    ↳ derivatives↙
                                                                    ↳  ↙
                                                                    ↳ calculation↙
                                                                    ↳  now?",
                                                                "Kalle's ↙
                                                                    ↳ Fraktaler↙
                                                                    ↳ ",
                                                                MB_OKCANCEL))
                                                        {
2610                                                            g_SFT.↙
                                                                    ↳ SetDerivatives↙
                                                                    ↳ (false);
                                                        }
                                                    }
                                                    else
                                                    {
2615                                                            output_log_message(Warn,↙
                                                                ↳  "derivatives are ↙
                                                                ↳ enabled but no ↙
                                                                ↳ analytic DE ↙
                                                                ↳ requested (slow)")↙
                                                                ↳ ;
                                                    }
                                                }
                                                if (hWnd)
                                                {
2620                                                    PostMessage(hWnd,WM_KEYDOWN,↙
                                                        ↳ VK_F5,0);
                                                }
                                                ret = false;
                                                return 0;
                                            }
2625    }

        static long OpenMap(HWND hWnd, bool &ret, const std::string &szFile)
        {
                g_SFT.Stop();
2630            g_bAnim = false;
                if (!(g_SFT.OpenMapB(szFile) || g_SFT.OpenMapEXR(szFile)))
                {
                        ret = true;
                        if (hWnd)
2635                            return MessageBox(hWnd,"Invalid map file","Error",MB_OK|↙
                                ↳ MB_ICONSTOP);
                        else
                                return 0;
                }
```

```
		else
		{
			if(g_hwColors)
				SendMessage(g_hwColors,WM_USER+99,0,0);
			g_SFT.ApplyColors();
			if (hWnd)
			{
				InvalidateRect(hWnd,NULL,FALSE);
				UpdateWindow(hWnd);
			}
		}
		ret = false;
		return 0;
	}

	static long OpenSettings(HWND hWnd, bool &ret)
	{
				g_SFT.Stop();
				g_bAnim=false;
				if(!g_SFT.OpenSettings(g_szSettingsFile))
				{
					ret = true;
					if (hWnd)
						return MessageBox(hWnd,"Invalid settings file","Error",MB_OK|MB_ICONSTOP);
					else
						return 0;
				}
				else{
					if (hWnd)
						UpdateMenusFromSettings(hWnd);
					if (hWnd)
						UpdateWindowSize(hWnd);
					g_SFT.SetImageSize(g_SFT.GetImageWidth(), g_SFT.GetImageHeight());
					g_SFT.SetApproxTerms(g_SFT.GetApproxTerms());
					if (hWnd)
						PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
					ret = false;
					return 0;
				}
	}

	static long WINAPI StoreZoomProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
	{
		(void) lParam;
		switch (uMsg)
		{
			case WM_INITDIALOG:
			{
				SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
				SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
```

```
                                ↪ ;
                        InitToolTip(hWnd, GetModuleHandle(NULL), GetToolText, 5);
2690                    break;
                }
                case WM_COMMAND:
                {
                        if (wParam == IDOK)
2695                    {
                                g_bStoreZoomMap = SendDlgItemMessage(hWnd, ↙
                                    ↪ IDC_STOREZOOM_KFB, BM_GETCHECK, 0, 0) != ↙
                                    ↪ 0;
                                g_bStoreZoomExr = SendDlgItemMessage(hWnd, ↙
                                    ↪ IDC_STOREZOOM_EXR, BM_GETCHECK, 0, 0) != ↙
                                    ↪ 0;
                                g_bStoreZoomTif = SendDlgItemMessage(hWnd, ↙
                                    ↪ IDC_STOREZOOM_TIF, BM_GETCHECK, 0, 0) != ↙
                                    ↪ 0;
                                g_bStoreZoomPng = SendDlgItemMessage(hWnd, ↙
                                    ↪ IDC_STOREZOOM_PNG, BM_GETCHECK, 0, 0) != ↙
                                    ↪ 0;
2700                            g_bStoreZoomJpg = SendDlgItemMessage(hWnd, ↙
                                    ↪ IDC_STOREZOOM_JPG, BM_GETCHECK, 0, 0) != ↙
                                    ↪ 0;
                                g_nStoreZoomCount = 0;
                                g_nStoreZoomLimit = GetDlgItemInt(hWnd, ↙
                                    ↪ IDC_STOREZOOM_COUNT, NULL, FALSE);
                                if (g_nStoreZoomLimit <= 0 || !↙
                                    ↪ SendDlgItemMessage(hWnd, ↙
                                    ↪ IDC_STOREZOOM_COUNTAUTO, BM_GETCHECK, 0, ↙
                                    ↪ 0))
                                {
2705                                    g_nStoreZoomLimit = 0;
                                }
                                ExitToolTip(hWnd);
                                EndDialog(hWnd, g_bStoreZoomMap || ↙
                                    ↪ g_bStoreZoomExr || g_bStoreZoomTif || ↙
                                    ↪ g_bStoreZoomPng || g_bStoreZoomJpg);
                        }
2710                    else if(wParam == IDCANCEL)
                        {
                                ExitToolTip(hWnd);
                                EndDialog(hWnd, 0);
                        }
2715                    break;
                }
        }
        return 0;
}
2720
static long WINAPI MainProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        if(uMsg==WM_CREATE){
                g_hwStatus = CreateStatusWindow(WS_CHILD|WS_VISIBLE,"",hWnd,0);
2725            RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                int widths[3]={180,430,-1};
```

```
	SendMessage(g_hwStatus,SB_SETPARTS,3,(LPARAM)&widths);
	SendMessage(g_hwStatus,SB_SETTEXT,0,(LPARAM)"");
	SendMessage(g_hwStatus,SB_SETTEXT,1,(LPARAM)"1");
	SendMessage(g_hwStatus,SB_SETTEXT,2,(LPARAM)"");
	RECT wr, cr;
	GetWindowRect(hWnd,&wr);
	wr.right-=wr.left;
	wr.bottom-=wr.top;
	GetClientRect(hWnd,&cr);
	int nXOffs = 640-cr.right;
	wr.right+=nXOffs;
	int nYOffs = 360-cr.bottom;
	wr.bottom+=nYOffs+sr.bottom;
	CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ZOOMSIZE_4,MF_BYCOMMAND|MF_CHECKED);

	RECT r;
	GetClientRect(GetDesktopWindow(),&r);
	g_SFT.SetWindowLeft(r.right/2-wr.right/2);
	g_SFT.SetWindowTop(r.bottom/2-wr.bottom/2);
	g_SFT.SetWindowRight(wr.right);
	g_SFT.SetWindowBottom(wr.bottom);
	UpdateWindowSize(hWnd);

	g_SFT.SetPosition((CFixedFloat)-2,(CFixedFloat)2,(CFixedFloat)-2,(CFixedFloat)2,640,360);
	SetTimer(hWnd,0,500,NULL);
	SYSTEMTIME st;
	GetLocalTime(&st);
	SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);

	g_hwHair = CreateDialog(GetModuleHandle(NULL),MAKEINTRESOURCE(IDD_DIALOG4),hWnd,(DLGPROC)CrossHairProc);
	GetWindowRect(hWnd,&wr);
	GetWindowRect(g_hwHair,&r);
	r.right-=r.left;
	r.bottom-=r.top;
	MoveWindow(g_hwHair,wr.left-r.right-3,wr.top,128,144,TRUE);
	PostMessage(g_hwHair,WM_USER+112,0,0);
	ShowWindow(g_hwHair,SW_SHOWNA);

	UpdateMenusFromSettings(hWnd);
	DragAcceptFiles(hWnd, TRUE);

	if (g_args->bLoadSettings)
	{
		bool ret;
		g_szSettingsFile = g_args->sLoadSettings;
		std::cerr << "loading settings: " << g_szSettingsFile << std::endl;
		OpenSettings(hWnd, ret);
	}
	if (g_args->bLoadLocation)
	{
		bool ret;
		g_szFile = g_args->sLoadLocation;
		std::cerr << "loading location: " << g_szFile << std::
```

```
                        ↪ endl;
                    OpenFile(hWnd, ret);
                }
                g_bSaveJpeg = g_args->bSaveJPG;
2785            g_bSaveTif = g_args->bSaveTIF;
                g_bSavePng = g_args->bSavePNG;
                g_bSaveExr = g_args->bSaveEXR;
                g_bSaveMap = g_args->bSaveMap;
                g_bInteractive = !(g_args->bSaveJPG || g_args->bSaveTIF ||↙
                    ↪ g_args->bSavePNG || g_args->bSaveEXR || g_args->bSaveMap);
2790            if (! g_bInteractive)
                {
                    std::cerr << "rendering at " << g_SFT.GetImageWidth() <<↙
                        ↪ "x" << g_SFT.GetImageHeight() << std::endl;
                }
                DisableUnsafeMenus(hWnd);
2795            g_SFT.RenderFractal(g_SFT.GetImageWidth(),g_SFT.GetImageHeight()↙
                    ↪ ,g_SFT.GetIterations(),hWnd);
        }
        else if(uMsg==WM_CLOSE)
            PostQuitMessage(0);
        else if(uMsg==WM_PAINT){
2800            RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                RECT rc;
                GetClientRect(hWnd,&rc);
2805            rc.bottom-=sr.bottom;
                int width = rc.right;
                int height = rc.bottom;
                PAINTSTRUCT ps;
                BeginPaint(hWnd,&ps);
2810            SetStretchBltMode(ps.hdc,HALFTONE);
                HDC dcBmp = CreateCompatibleDC(ps.hdc);
                HBITMAP bmBmp = g_SFT.ShrinkBitmap(g_SFT.GetBitmap(), width, ↙
                    ↪ height, g_SFT.GetShrink());
                HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
                if(g_bShowInflection){
2815                POINT p;
                    int i;
                    for(i=0;i<g_nInflection;i++){
                        p = g_pInflections[i];
                        RotateImageAroundPoint(bmBmp,p);
2820                }
                    GetCursorPos(&p);
                    ScreenToClient(hWnd,&p);
                    RotateImageAroundPoint(bmBmp,p);
                }
2825            else if(g_bShowSkew){
                    SelectObject(dcBmp,bmOld);
                    SkewImage(bmBmp);
                    bmOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
                }
2830            StretchBlt(ps.hdc,0,0,rc.right,rc.bottom,dcBmp,0,0,width,height,↙
                    ↪ SRCCOPY);
                SelectObject(dcBmp,bmOld);
                DeleteObject(bmBmp);
```

```
                DeleteDC(dcBmp);
                EndPaint(hWnd,&ps);
2835            return 0;
        }
        else if(uMsg==WM_RBUTTONDOWN && HIWORD(GetKeyState(VK_CONTROL))){
                g_bTrackSelect=TRUE;
                g_pSelect.x = (short)LOWORD(lParam);
2840            g_pSelect.y = (short)HIWORD(lParam);
                g_pTrackStart.x = (short)LOWORD(lParam);
                g_pTrackStart.y = (short)HIWORD(lParam);
                SetCapture(hWnd);
                return 0;
2845    }
        else if(uMsg==WM_LBUTTONDOWN){
                if(g_bNewton){
                        static RECT r;
                        GetClientRect(hWnd,&r);
2850                    RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
                        r.bottom-=sr.bottom;
                        r.left = (short)LOWORD(lParam);
2855                    r.top = (short)HIWORD(lParam);
                        PostMessage(g_hwNewton,WM_USER+1,0,(LPARAM)&r);
                        return 0;
                }
                if(!HIWORD(GetKeyState(VK_CONTROL)) && !g_bRotate && !↙
                    ↪ g_bFindMinibrot && g_SFT.GetAnimateZoom() && !↙
                    ↪ g_bAddReference && !g_bEraser && !g_hwExamine && !↙
                    ↪ g_bAddMainReference){
2860                    while(g_bAnim){
                                Sleep(3);
                        }
                        g_SFT.Stop();
                        g_bMove=TRUE;
2865                    SetCapture(hWnd);
                        g_pSelect.x = (short)LOWORD(lParam);
                        g_pSelect.y = (short)HIWORD(lParam);
                        return 0;
                }
2870            if(g_bRotate){
                        g_SFT.Stop();
                        g_bAnim=false;
                        RECT r;
                        GetClientRect(hWnd,&r);
2875                        RECT sr;
                            GetWindowRect(g_hwStatus,&sr);
                            sr.bottom-=sr.top;
                            r.bottom-=sr.bottom;
                        POINT pm={r.right/2,r.bottom/2};
2880                    POINT p;
                        GetCursorPos(&p);
                        ScreenToClient(hWnd,&p);
                        double ratio = (double)(p.y-pm.y)/(double)(p.x-pm.x);
                        g_StartDegree = atan(ratio);
2885                    if(p.x-pm.x>0)
                                bXSign=TRUE;
```

```
                        else
                                bXSign=FALSE;
                        g_MoveDegree = 0;
2890                    g_bRotate=2;
                        SetCapture(hWnd);
                        return 0;
                }
                if(g_bWaitRead)
2895                    return 0;
                MSG msg;
                int nButtons=0;
                g_SFT.Stop();
                g_bAnim=false;
2900            while(PeekMessage(&msg,hWnd,WM_LBUTTONDOWN,WM_LBUTTONDOWN,↙
                    ↳ PM_REMOVE)){
                        nButtons++;
                        Sleep(10);
                }
                g_bFindMinibrot=FALSE;
2905            g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
                g_bmSaveZoomBuff=NULL;

                InvalidateRect(hWnd,NULL,FALSE);
2910            UpdateWindow(hWnd);
                RECT rc;
                GetClientRect(hWnd,&rc);
                RECT sr;
                GetWindowRect(g_hwStatus,&sr);
2915            sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
                HDC hDC = GetDC(hWnd);
                SetROP2(hDC,R2_NOT);
                g_pSelect.x = (short)LOWORD(lParam);
2920            g_pSelect.y = (short)HIWORD(lParam);
                MoveToEx(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2),NULL);
                LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
                LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
2925            LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
                ReleaseDC(hWnd,hDC);
                g_bSelect=1;
                SetCapture(hWnd);
        }
2930    else if(uMsg==WM_MOUSEMOVE){
                if(g_bTrackSelect==1){
                        HDC hDC = GetDC(hWnd);
                        SetROP2(hDC,R2_NOT);
                        MoveToEx(hDC,g_pTrackStart.x,g_pTrackStart.y,NULL);
2935
                        LineTo(hDC,g_pSelect.x,g_pTrackStart.y);
                        LineTo(hDC,g_pSelect.x,g_pSelect.y);
```

```
                        LineTo(hDC, g_pTrackStart.x, g_pSelect.y);
                        LineTo(hDC, g_pTrackStart.x, g_pTrackStart.y);
2940
                        g_pSelect.x = (short)LOWORD(lParam);
                        g_pSelect.y = (short)HIWORD(lParam);
                        double yoffs = (double)(g_pSelect.y - g_pTrackStart.y)/(↙
                            ↳ double)g_SFT.GetHeight();
                        double xoffs = (double)(g_pSelect.x - g_pTrackStart.x)/(↙
                            ↳ double)g_SFT.GetWidth();
2945                    if((yoffs<0?-yoffs:yoffs)>(xoffs<0?-xoffs:xoffs)){
                                double offs = yoffs;
                                if((yoffs<0 && xoffs>0) || (yoffs>0 && xoffs<0))
                                        offs=-offs;
                                g_pSelect.x = g_pTrackStart.x + g_SFT.GetWidth()↙
                                    ↳ *offs;
2950                    }
                        else{
                                double offs = xoffs;
                                if((yoffs<0 && xoffs>0) || (yoffs>0 && xoffs<0))
                                        offs=-offs;
2955                            g_pSelect.y = g_pTrackStart.y + g_SFT.GetHeight↙
                                    ↳ ()*offs;
                        }

                        MoveToEx(hDC, g_pTrackStart.x, g_pTrackStart.y,NULL);
                        LineTo(hDC, g_pSelect.x, g_pTrackStart.y);
2960                    LineTo(hDC, g_pSelect.x, g_pSelect.y);
                        LineTo(hDC, g_pTrackStart.x, g_pSelect.y);
                        LineTo(hDC, g_pTrackStart.x, g_pTrackStart.y);
                        ReleaseDC(hWnd,hDC);
                }
2965            if(g_bShowInflection)
                        InvalidateRect(hWnd,NULL,FALSE);
                if(g_bMove){
                        SetCursor(LoadCursor(NULL,IDC_SIZEALL));
                        RECT rc;
2970                    GetClientRect(hWnd,&rc);
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
                        rc.bottom-=sr.bottom;
2975                    POINT p;
                        p.x = (short)LOWORD(lParam) - g_pSelect.x;
                        p.y = (short)HIWORD(lParam) - g_pSelect.y;
                        HDC hDC = GetDC(hWnd);
                        HDC dcBmp = CreateCompatibleDC(hDC);
2980                    HDC dcBuff = CreateCompatibleDC(hDC);
                        HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,g_SFT.↙
                            ↳ GetBitmap());
                        HBITMAP bmBuff = create_bitmap(hDC,rc.right,rc.bottom);
                        HBITMAP bmOldBuff = (HBITMAP)SelectObject(dcBuff,bmBuff)↙
                            ↳ ;
                        SetStretchBltMode(dcBuff,HALFTONE);
2985                    StretchBlt(dcBuff,p.x,p.y,rc.right,rc.bottom,dcBmp,0,0,↙
                            ↳ g_SFT.GetWidth(),g_SFT.GetHeight(),SRCCOPY);
                        SetStretchBltMode(hDC,HALFTONE);
                        BitBlt(hDC,0,0,rc.right,rc.bottom,dcBuff,0,0,SRCCOPY);
```

```
                /*
2990            HDC dcBkg = CreateCompatibleDC(hDC);
                SetStretchBltMode(dcBkg,HALFTONE);
                HBITMAP bmBkgOld = (HBITMAP)SelectObject(dcBkg,g_SFT.GetBitmap()↙
                    ↳ );
                StretchBlt(dcBkg,0,0,g_SFT.GetWidth(),g_SFT.GetHeight(),dcBuff↙
                    ↳ ,0,0,rc.right,rc.bottom,SRCCOPY);
                SelectObject(dcBkg,bmBkgOld);
2995            DeleteDC(dcBkg);
                g_SFT.UpdateBitmap();
                */

                        SelectObject(dcBmp,bmOld);
3000                    DeleteDC(dcBmp);
                        SelectObject(dcBuff,bmOldBuff);
                        DeleteDC(dcBuff);
                        DeleteObject(bmBuff);
                        ReleaseDC(hWnd,hDC);
3005                    return 0;
                }
                if(g_bRotate){
                        SetCursor(LoadCursor(GetModuleHandle(NULL),↙
                            ↳ MAKEINTRESOURCE(IDC_CURSOR1)));
                        if(g_bRotate==2){
3010                            HDC hDC = GetDC(hWnd);
                                RECT r;
                                GetClientRect(hWnd,&r);
                                RECT sr;
                                GetWindowRect(g_hwStatus,&sr);
3015                            sr.bottom-=sr.top;
                                r.bottom-=sr.bottom;
                                POINT pm={r.right/2,r.bottom/2};
                                HBITMAP bmBmp = create_bitmap(hDC,g_SFT.GetWidth↙
                                    ↳ (),g_SFT.GetHeight());
                                POINT p;
3020                            GetCursorPos(&p);
                                ScreenToClient(hWnd,&p);
                                double ratio = (double)(p.y-pm.y)/(double)(p.x-↙
                                    ↳ pm.x);
                                g_MoveDegree = atan(ratio);
                                if((p.x-pm.x>0 && !bXSign) || (p.x-pm.x<0 && ↙
                                    ↳ bXSign))
3025                                    g_MoveDegree+=pi;
                                g_MoveDegree -= g_StartDegree;
                                pm.x = g_SFT.GetWidth()/2;
                                pm.y = g_SFT.GetHeight()/2;
                                RotateImage(g_SFT.GetBitmap(),bmBmp,pm,↙
                                    ↳ g_MoveDegree);
3030                            HDC dcBmp = CreateCompatibleDC(hDC);
                                HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,↙
                                    ↳ bmBmp);
                                SetStretchBltMode(hDC,HALFTONE);
                                StretchBlt(hDC,0,0,r.right,r.bottom,dcBmp,0,0,↙
                                    ↳ g_SFT.GetWidth(),g_SFT.GetHeight(),SRCCOPY↙
                                    ↳ );
                                SelectObject(dcBmp,bmOld);
```

```
3035                                    DeleteObject(dcBmp);
                                        DeleteObject(bmBmp);
                                        ReleaseDC(hWnd,hDC);
                                }
                                return 0;
3040                    }
                        if(g_bSelect){
                                RECT rc;
                                GetClientRect(hWnd,&rc);
                                RECT sr;
3045                            GetWindowRect(g_hwStatus,&sr);
                                sr.bottom-=sr.top;
                                rc.bottom-=sr.bottom;
                                HDC hDC = GetDC(hWnd);
                                SetROP2(hDC,R2_NOT);
3050                            MoveToEx(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()↙
                                    ↳ *2),g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2),↙
                                    ↳ NULL);
                                LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
                                LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
3055
                                g_pSelect.x = (short)LOWORD(lParam);
                                g_pSelect.y = (short)HIWORD(lParam);
                                MoveToEx(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()↙
                                    ↳ *2),g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2),↙
                                    ↳ NULL);
                                LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
3060                            LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
                                ReleaseDC(hWnd,hDC);
                        }
3065                    char szI[1024];
                        strcpy(szI,"I:");

                        RECT rc;
                        GetClientRect(hWnd,&rc);
3070                    RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
                        rc.bottom-=sr.bottom;
                        int x = (short)LOWORD(lParam)*g_SFT.GetWidth()/rc.right;
3075                    int y = (short)HIWORD(lParam)*g_SFT.GetHeight()/rc.bottom;
                        int i = g_SFT.GetIterationOnPoint(x,y);
                        if(i != PIXEL_UNEVALUATED){
                                snprintf(szI+strlen(szI),100,"%d",i);
                                snprintf(szI+strlen(szI),100," <%d,%d> S:%f",(short)↙
```

```
                                ↪ LOWORD(lParam),(short)HIWORD(lParam),g_SFT.↙
                                ↪ GetTransOnPoint(x,y));
3080                        SendMessage(g_hwStatus,SB_SETTEXT,2,(LPARAM)szI);
                    }
                }
                else if(uMsg==WM_LBUTTONUP && g_bMove){
                    g_bMove=FALSE;
3085                ReleaseCapture();
                    POINT p;
                    p.x = (short)LOWORD(lParam) - g_pSelect.x;
                    p.y = (short)HIWORD(lParam) - g_pSelect.y;
                    RECT rc;
3090                GetClientRect(hWnd,&rc);
                    RECT sr;
                    GetWindowRect(g_hwStatus,&sr);
                    sr.bottom-=sr.top;
                    rc.bottom-=sr.bottom;
3095
                    if(1){
                        HDC hDC = GetDC(hWnd);
                        HDC dcBmp = CreateCompatibleDC(hDC);
                        HDC dcBuff = CreateCompatibleDC(hDC);
3100                    HBITMAP bmBuff = ShrinkBitmap2(g_SFT.GetBitmap(),g_SFT.↙
                            ↪ GetWidth(),g_SFT.GetHeight());
                        HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,g_SFT.↙
                            ↪ GetBitmap());
                        HBITMAP bmOldBuff = (HBITMAP)SelectObject(dcBuff,bmBuff)↙
                            ↪ ;
                        int x = p.x*g_SFT.GetWidth()/rc.right;
                        int y = p.y*g_SFT.GetHeight()/rc.bottom;
3105                    BitBlt(dcBmp,x,y,g_SFT.GetWidth(),g_SFT.GetHeight(),↙
                            ↪ dcBuff,0,0,SRCCOPY);
                        SelectObject(dcBmp,bmOld);
                        SelectObject(dcBuff,bmOldBuff);
                        DeleteObject(bmBuff);
                        DeleteDC(dcBmp);
3110                    DeleteDC(dcBuff);
                        ReleaseDC(NULL,hDC);
                        g_SFT.UpdateBitmap();
                    }

3115                p.x = (short)(rc.right/2-p.x)*g_SFT.GetWidth()/rc.right;
                    p.y = (short)(rc.bottom/2-p.y)*g_SFT.GetHeight()/rc.bottom;

                    SYSTEMTIME st;
                    GetLocalTime(&st);
3120                SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
                    if(g_bAutoGlitch)
                        g_bAutoGlitch=1;

                    g_SFT.UndoStore();
3125                g_SFT.Zoom(p.x,p.y,1,g_SFT.GetWidth(),g_SFT.GetHeight(),TRUE);
                    SetTimer(hWnd,0,500,NULL);
                    return 0;
                }
                else if(uMsg==WM_LBUTTONUP && g_bRotate){
3130                ReleaseCapture();
```

```
                g_bRotate=FALSE;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ROTATE,MF_BYCOMMAND|(g_bRotate?MF_CHECKED:MF_UNCHECKED));

                                HDC hDC = GetDC(hWnd);
3135                            RECT r;
                                GetClientRect(hWnd,&r);
                                RECT sr;
                                GetWindowRect(g_hwStatus,&sr);
                                sr.bottom-=sr.top;
3140                            r.bottom-=sr.bottom;
                                POINT pm={r.right/2,r.bottom/2};
                                HBITMAP bmBmp = create_bitmap(hDC,g_SFT.GetWidth(),g_SFT.GetHeight());
                                POINT p;
                                GetCursorPos(&p);
3145                            ScreenToClient(hWnd,&p);
                                double ratio = (double)(p.y-pm.y)/(double)(p.x-pm.x);
                                g_MoveDegree = atan(ratio);
                                if((p.x-pm.x>0 && !bXSign) || (p.x-pm.x<0 && bXSign))
                                    g_MoveDegree+=pi;
3150                            g_MoveDegree -= g_StartDegree;
                                pm.x = g_SFT.GetWidth()/2;
                                pm.y = g_SFT.GetHeight()/2;
                                RotateImage(g_SFT.GetBitmap(),bmBmp,pm,g_MoveDegree);
                                HDC dcBmp = CreateCompatibleDC(hDC);
3155                            HDC dcSft = CreateCompatibleDC(hDC);
                                HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
                                HBITMAP bmOldSft = (HBITMAP)SelectObject(dcSft,g_SFT.GetBitmap());
                                SetStretchBltMode(hDC,HALFTONE);
                                SetStretchBltMode(dcSft,HALFTONE);
3160                            StretchBlt(hDC,0,0,r.right,r.bottom,dcBmp,0,0,g_SFT.GetWidth(),g_SFT.GetHeight(),SRCCOPY);
                                BitBlt(dcSft,0,0,g_SFT.GetWidth(),g_SFT.GetHeight(),dcBmp,0,0,SRCCOPY);
                                SelectObject(dcBmp,bmOld);
                                DeleteObject(dcBmp);
                                DeleteObject(bmBmp);
3165                            SelectObject(dcSft,bmOldSft);
                                g_SFT.UpdateBitmap();
                                DeleteObject(dcSft);
                                ReleaseDC(hWnd,hDC);

3170            g_Degree += g_MoveDegree;
                SYSTEMTIME st;
                GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
                SetTimer(hWnd,0,500,NULL);
3175    //      std::cerr << "WMLBUTTONUP && g_bRotate" << std::endl;
                g_SFT.UndoStore();
                DisableUnsafeMenus(hWnd);
```

```
                g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.↙
                    ↳ GetIterations(),hWnd);
        }
3180    else if(g_bTrackSelect==1 && (uMsg==WM_CAPTURECHANGED || uMsg==↙
            ↳ WM_RBUTTONUP || (uMsg==WM_KEYDOWN && wParam==VK_ESCAPE))){
                g_bTrackSelect=FALSE;
                ReleaseCapture();
                HDC hDC = GetDC(hWnd);
                SetROP2(hDC,R2_NOT);
3185            MoveToEx(hDC,g_pTrackStart.x,g_pTrackStart.y,NULL);
                LineTo(hDC,g_pSelect.x,g_pTrackStart.y);
                LineTo(hDC,g_pSelect.x,g_pSelect.y);
                LineTo(hDC,g_pTrackStart.x,g_pSelect.y);
                LineTo(hDC,g_pTrackStart.x,g_pTrackStart.y);
3190            ReleaseDC(hWnd,hDC);
                if(uMsg==WM_KEYDOWN && wParam==VK_ESCAPE){
                        g_bTrackSelect=2;
                        return 0;
                }
3195            if(g_pSelect.y==g_pTrackStart.y || g_pSelect.x==g_pTrackStart.x)
                        return 0;

                RECT rc;
                GetClientRect(hWnd,&rc);
3200            RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
                double nZoom = (double)rc.bottom/(double)(g_pSelect.y-↙
                    ↳ g_pTrackStart.y);
3205            if(nZoom<0)
                        nZoom=-nZoom;
                POINT p;
                p.x = (g_pSelect.x+g_pTrackStart.x)/2;
                p.y = (g_pSelect.y+g_pTrackStart.y)/2;
3210
                if(g_SFT.GetAnimateZoom()){
                        g_pSelect.x = -g_SFT.GetWidth()/2;
                        g_pSelect.y = g_SFT.GetHeight()/2;
                        ANIM* pAnim = new ANIM;
3215                    if(lParam==9)
                                pAnim->nZoomSize = 2;
                        else
                                pAnim->nZoomSize = nZoom;
                        pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),rc.right,↙
                            ↳ rc.bottom);
3220                    pAnim->hWnd = hWnd;
                        pAnim->pOffs = p;
                        pAnim->bZoomOut = FALSE;
                        pAnim->bZoomOne = FALSE;
                        DWORD dw;
3225                    HANDLE hThread = CreateThread(NULL,0,(↙
                            ↳ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                            ↳ ;
                        CloseHandle(hThread);
                }
                AutoIterations();
```

```
                SYSTEMTIME st;
3230            GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
                p.x = (short)(p.x)*g_SFT.GetWidth()/rc.right;
                p.y = (short)(p.y)*g_SFT.GetHeight()/rc.bottom;
                g_SFT.UndoStore();
3235            g_SFT.Zoom(p.x,p.y,nZoom,g_SFT.GetWidth(),g_SFT.GetHeight(),↙
                    ↳ FALSE);
                SetTimer(hWnd,0,500,NULL);
            }
            else if(!g_bWaitRead && g_bSelect && ((uMsg==WM_KEYDOWN && wParam==↙
                ↳ VK_ESCAPE) || uMsg==WM_LBUTTONUP || uMsg==WM_CAPTURECHANGED)){
                g_bSelect=FALSE;
3240            ReleaseCapture();
                if(g_bRotate){
                        g_bRotate=FALSE;
                        return 0;
                }
3245            RECT rc;
                GetClientRect(hWnd,&rc);
                RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
3250            rc.bottom-=sr.bottom;
                HDC hDC = GetDC(hWnd);
                SetROP2(hDC,R2_NOT);
                MoveToEx(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2),NULL);
                LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
3255            LineTo(hDC,g_pSelect.x+rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y+rc.bottom/(g_SFT.GetZoomSize()*2));
                LineTo(hDC,g_pSelect.x-rc.right/(g_SFT.GetZoomSize()*2),↙
                    ↳ g_pSelect.y-rc.bottom/(g_SFT.GetZoomSize()*2));
                ReleaseDC(hWnd,hDC);
                if(uMsg==WM_LBUTTONUP || uMsg==WM_CAPTURECHANGED){
3260                    if(g_SFT.GetAnimateZoom() && !g_bAddMainReference && !↙
                            ↳ g_bAddReference && !g_bEraser){
                                ANIM* pAnim = new ANIM;
                                pAnim->nZoomSize = g_SFT.GetZoomSize();
                                pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),↙
                                    ↳ rc.right,rc.bottom);
                                pAnim->hWnd = hWnd;
3265                            pAnim->pOffs = g_pSelect;
                                pAnim->bZoomOut = FALSE;
                                pAnim->bZoomOne = FALSE;
                                DWORD dw;
                                HANDLE hThread = CreateThread(NULL,0,(↙
                                    ↳ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)↙
                                    ↳ pAnim,0,&dw);
3270                            CloseHandle(hThread);
                        }

                        if(!g_bAddMainReference && !g_bAddReference && !↙
                            ↳ g_bEraser){
```

```
                HDC hDC = GetDC(NULL);
3275            HDC dcBmp = CreateCompatibleDC(hDC);
                HBITMAP bmBmp = g_SFT.GetBitmap();

                HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,↙
                    ↳ bmBmp);
                HDC dcSBmp = CreateCompatibleDC(hDC);
3280            SIZE sc, sb;
                POINT pSelect;
                sb.cx = g_SFT.GetWidth();
                sb.cy = g_SFT.GetHeight();
                pSelect.x = g_pSelect.x*sb.cx/rc.right;
3285            pSelect.y = g_pSelect.y*sb.cy/rc.bottom;
                sc.cx = sb.cx/(g_SFT.GetZoomSize());
                sc.cy = sb.cy/(g_SFT.GetZoomSize());
                HBITMAP bmSBmp = create_bitmap(hDC, sc.cx, sc.cy);
                HBITMAP bmSOld = (HBITMAP)SelectObject(dcSBmp,↙
                    ↳ bmSBmp);
3290            SetStretchBltMode(dcSBmp,HALFTONE);
                SetStretchBltMode(dcBmp,HALFTONE);
                BitBlt(dcSBmp,0,0,sc.cx,sc.cy,dcBmp,pSelect.x-sb↙
                    ↳ .cx/(g_SFT.GetZoomSize()*2),pSelect.y-sb.↙
                    ↳ cy/(g_SFT.GetZoomSize()*2),SRCCOPY);
                StretchBlt(dcBmp,0,0,sb.cx,sb.cy,dcSBmp,0,0,sc.↙
                    ↳ cx,sc.cy,SRCCOPY);
                SelectObject(dcBmp,bmOld);
3295            SelectObject(dcSBmp,bmSOld);
                DeleteObject(bmSBmp);
                DeleteDC(dcBmp);
                DeleteDC(dcSBmp);
                ReleaseDC(NULL,hDC);
3300            g_SFT.UpdateBitmap();
                if(!g_SFT.GetAnimateZoom()){
                        InvalidateRect(hWnd,NULL,FALSE);
                        UpdateWindow(hWnd);
                }
3305
                AutoIterations();
        }

        SYSTEMTIME st;
3310    GetLocalTime(&st);
        SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);

        int x = (short)LOWORD(lParam)*g_SFT.GetWidth()/rc.right;
        int y = (short)HIWORD(lParam)*g_SFT.GetHeight()/rc.↙
            ↳ bottom;
3315

        if(g_bAddMainReference){
                if(!g_hwExamine)
                        g_bAddMainReference=false;
3320            if(g_hwExamine){
                        g_bExamineDirty=TRUE;
                        SetFocus(g_hwExamine);
                }
                CheckMenuItem(GetMenu(hWnd),↙
```

```
                                    ↪ ID_ACTIONS_SPECIAL_SETMAINREFERENCE,↙
                                    ↪ MF_BYCOMMAND|MF_UNCHECKED);
3325                                int x = (short)LOWORD(lParam)*g_SFT.GetWidth()/↙
                                    ↪ rc.right;
                                    int y = (short)HIWORD(lParam)*g_SFT.GetHeight()/↙
                                    ↪ rc.bottom;
                                    g_SFT.AddReference(x,y,TRUE);
                                    SetTimer(hWnd,0,500,NULL);
                                    return 0;
3330                            }
                                else if(g_bAddReference){
                                    if(!g_hwExamine)
                                            g_bAddReference=FALSE;
                                    if(g_hwExamine){
3335                                        g_bExamineDirty=TRUE;
                                            SetFocus(g_hwExamine);
                                    }
                                    CheckMenuItem(GetMenu(hWnd),↙
                                    ↪ ID_ACTIONS_ADDREFERENCE,MF_BYCOMMAND|↙
                                    ↪ MF_UNCHECKED);
                                    int x = (short)LOWORD(lParam)*g_SFT.GetWidth()/↙
                                    ↪ rc.right;
3340                                int y = (short)HIWORD(lParam)*g_SFT.GetHeight()/↙
                                    ↪ rc.bottom;
                                    if(g_SFT.AddReference(x,y,FALSE,g_SFT.↙
                                    ↪ GetSolveGlitchNear()))
                                            SetTimer(hWnd,0,500,NULL);
                                    return 0;
                                }
3345                            else if(g_bEraser){
                                    RECT rE = {LONG(g_pSelect.x-rc.right/(g_SFT.↙
                                    ↪ GetZoomSize()*2)),LONG(g_pSelect.y-rc.↙
                                    ↪ bottom/(g_SFT.GetZoomSize()*2)),LONG(↙
                                    ↪ g_pSelect.x+rc.right/(g_SFT.GetZoomSize()↙
                                    ↪ *2)),LONG(g_pSelect.y+rc.bottom/(g_SFT.↙
                                    ↪ GetZoomSize()*2))};
                                    rE.left = g_SFT.GetWidth()*rE.left/rc.right;
                                    rE.top = g_SFT.GetHeight()*rE.top/rc.bottom;
                                    rE.right = g_SFT.GetWidth()*rE.right/rc.right;
3350                                rE.bottom = g_SFT.GetHeight()*rE.bottom/rc.↙
                                    ↪ bottom;
                                    int x, y;
                                    for(x=rE.left;x<rE.right;x++)
                                            for(y=rE.top;y<rE.bottom;y++)
                                                    g_SFT.ErasePixel(x,y);
3355                                g_bExamineDirty=TRUE;
                                    g_SFT.ApplyColors();
                                    InvalidateRect(hWnd,NULL,FALSE);
                                    UpdateWindow(hWnd);
                                    return 0;
3360                            }
                                else
                                {
                                    g_SFT.UndoStore();
                                    g_SFT.Zoom(x,y,g_SFT.GetZoomSize(),g_SFT.↙
                                    ↪ GetWidth(),g_SFT.GetHeight(),g_SFT.↙
                                    ↪ GetZoomSize()==1);
```

```
3365                            }
                                SetTimer(hWnd,0,500,NULL);
                        }
                }
                else if(uMsg==WM_KEYDOWN && wParam==VK_F5){
3370                    if(g_hwExamine)
                                g_bExamineDirty=TRUE;
                        g_nPrevGlitchX=g_nPrevGlitchY=-1;
                        g_SFT.Stop();
                        g_bAnim=false;
3375                    g_bFindMinibrot=FALSE;
                        g_bStoreZoom=FALSE;
                        DeleteObject(g_bmSaveZoomBuff);
                        g_bmSaveZoomBuff=NULL;
                        SetTimer(hWnd,0,500,NULL);
3380                    RECT r;
                        GetClientRect(hWnd,&r);
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
3385                    r.bottom-=sr.bottom;
                        SYSTEMTIME st;
                        GetLocalTime(&st);
                        SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
#if 0
3390                    if(g_SFT.GetWidth()<r.right || g_SFT.GetHeight()<r.bottom)
                        {
//                              std::cerr << "WM_KEYDOWN && wParam==VK_F5 && small" << std::endl;
                                DisableUnsafeMenus(hWnd);
                                g_SFT.RenderFractal(r.right,r.bottom,g_SFT.GetIterations(),hWnd);
3395                    }
                        else
#endif
                        if(g_SFT.GetArbitrarySize()){
                                SIZE sc;
3400                            sc.cy = g_SFT.GetHeight();
                                sc.cx = (double)r.right*((double)sc.cy/(double)r.bottom);
//                              std::cerr << "WM_KEYDOWN && wParam==VK_F5 && arbitrary" << std::endl;
                                DisableUnsafeMenus(hWnd);
                                g_SFT.RenderFractal(sc.cx,sc.cy,g_SFT.GetIterations(),hWnd);
3405                    }
                        else
                        {
//                              std::cerr << "WM_KEYDOWN && wParam==VK_F5 && otherwise" << std::endl;
                                DisableUnsafeMenus(hWnd);
3410                            g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.GetIterations(),hWnd);
                        }
                }
                else if(uMsg==WM_KEYDOWN && wParam==VK_ESCAPE){
                        if(g_bAddReference){
```

```
3415                            g_bAddReference=FALSE;
                                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ADDREFERENCE,↙
                                    ↳ MF_BYCOMMAND|MF_UNCHECKED);
                        }
                        if(g_bAddMainReference){
                                g_bAddMainReference=false;
3420                            CheckMenuItem(GetMenu(hWnd),↙
                                    ↳ ID_ACTIONS_SPECIAL_SETMAINREFERENCE,MF_BYCOMMAND|↙
                                    ↳ MF_UNCHECKED);
                        }
                        g_SFT.Stop();
                        g_bAnim=false;
                        g_bFindMinibrot=FALSE;
3425                    g_bStoreZoom=FALSE;
                        DeleteObject(g_bmSaveZoomBuff);
                        g_bmSaveZoomBuff=NULL;
                }

3430            else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_COPY) || (uMsg==↙
                    ↳ WM_KEYDOWN && wParam=='X' && HIWORD(GetKeyState(VK_CONTROL))))
                {
                                        std::string str = g_SFT.ToText();
                                        if (!OpenClipboard(hWnd))
                                                return 0;
3435                                    EmptyClipboard();
                                        HANDLE hclip = GlobalAlloc(GMEM_MOVEABLE, (str.↙
                                            ↳ length() + 1) * sizeof(TCHAR));
                                        if (! hclip)
                                        {
                                                CloseClipboard();
3440                                            return 0;
                                        }
                                        char *s = (char *) GlobalLock(hclip);
                                        memcpy(s, str.c_str(), str.length() + 1);
                                        GlobalUnlock(hclip);
3445                                    SetClipboardData(CF_TEXT, hclip);
                                        CloseClipboard();
                }

                else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_PASTE) || (uMsg==↙
                    ↳ WM_KEYDOWN && wParam=='V' && HIWORD(GetKeyState(VK_CONTROL))))
3450            {
                                        if(!OpenClipboard(hWnd))
                                                return 0;
                                        HANDLE hTemp;
                                        const char *szTemp=0;
3455                                    if((hTemp = GetClipboardData(CF_TEXT)))
                                                szTemp = (const char*)GlobalLock(hTemp);
                                        if(!szTemp || !*szTemp){
                                                GlobalUnlock(hTemp);
                                                CloseClipboard();
3460                                            return 0;
                                        }
                                        std::string str(szTemp);
                                        GlobalUnlock(hTemp);
                                        CloseClipboard();
3465                                    g_SFT.UndoStore();
```

```
                                        BOOL ok = g_SFT.OpenString(str, FALSE);
                                        if (ok)
                                        {
                                                g_SFT.Stop();
3470                                            g_bAnim=false;
                                                g_bFindMinibrot=FALSE;
                                                g_bStoreZoom=FALSE;
                                                DeleteObject(g_bmSaveZoomBuff);
                                                g_bmSaveZoomBuff=NULL;
3475                                            PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                                        }
        }

    else if(uMsg==WM_DROPFILES)
3480    {
                HDROP hDrop = (HDROP) wParam;
                if (hDrop)
                {
                        UINT len = DragQueryFile(hDrop, 0, 0, 0);
3485                    if (len > 0)
                        {
                                char *buffer = (char *) calloc(1, 2 * len + 1);
                                if (buffer)
                                {
3490                                    UINT ok = DragQueryFile(hDrop, 0, buffer↙
                                            ↳ , 2 * len);
                                        if (ok)
                                        {
                                                std::string file(buffer);
                                                // call function OpenFile() (not↙
                                                    ↳  the similarly named ↙
                                                    ↳ method)
3495                                            // so that the window title is ↙
                                                    ↳ set correctly
                                                g_szFile = file;
                                                bool err = false;
                                                g_SFT.UndoStore();
                                                OpenFile(hWnd, err);
3500                                            if (! err)
                                                {
                                                        // don't know if this ↙
                                                            ↳ code is necessary
                                                        // probably it would be ↙
                                                            ↳ better to stop *↙
                                                            ↳ before* opening?
                                                        g_SFT.Stop();
3505                                                    g_bAnim=false;
                                                        g_bFindMinibrot=FALSE;
                                                        g_bStoreZoom=FALSE;
                                                        DeleteObject(↙
                                                            ↳ g_bmSaveZoomBuff);
                                                        g_bmSaveZoomBuff=NULL;
3510                                                    PostMessage(hWnd,↙
                                                            ↳ WM_KEYDOWN,VK_F5↙
                                                            ↳ ,0);
                                                }
                                        }
```

```
                                        free(buffer);
                                }
3515                    }
                        DragFinish(hDrop);
                }
        }

3520    else if(uMsg==WM_KEYDOWN && wParam==VK_PRIOR)
        {
                // page up = zoom in
                g_SFT.UndoStore();
                g_SFT.Zoom(g_SFT.GetWidth()/2,g_SFT.GetHeight()/2, 1*g_SFT.↙
                    ↪ GetZoomSize(), g_SFT.GetWidth(),g_SFT.GetHeight(),FALSE);
3525            PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
        }
        else if(uMsg==WM_KEYDOWN && wParam==VK_NEXT)
        {
                // page down = zoom out
3530            g_SFT.UndoStore();
                g_SFT.Zoom(g_SFT.GetWidth()/2,g_SFT.GetHeight()/2, 1/g_SFT.↙
                    ↪ GetZoomSize(), g_SFT.GetWidth(),g_SFT.GetHeight(),FALSE);
                PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
        }

3535    else if(uMsg==WM_KEYDOWN && wParam==187){
                lParam=9;
                g_bAddReference=FALSE;
                g_bAddMainReference=false;
                g_bFindMinibrot=FALSE;
3540            g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
                g_bmSaveZoomBuff=NULL;
                while(g_bAnim){
                        Sleep(3);
3545            }
                POINT p;
                GetCursorPos(&p);
                ScreenToClient(hWnd,&p);
                RECT rc;
3550            GetClientRect(hWnd,&rc);
                RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
3555            if(!g_SFT.GetAnimateZoom()){
                        SendMessage(hWnd,WM_KEYDOWN,VK_ESCAPE,0);
                        SendMessage(hWnd,WM_LBUTTONDOWN,0,MAKELONG(p.x,p.y));
                        PostMessage(hWnd,WM_LBUTTONUP,0,MAKELONG(p.x,p.y));
                        return 0;
3560            }
                if(lParam==9){
                        p.x=rc.right/2 + (p.x-rc.right/2)/2;
                        p.y=rc.bottom/2 + (p.y-rc.bottom/2)/2;
                }
3565            AutoIterations();
                g_pSelect.x = -g_SFT.GetWidth()/2;
                g_pSelect.y = g_SFT.GetHeight()/2;
```

```
                ANIM* pAnim = new ANIM;
                if(lParam==9)
3570                    pAnim->nZoomSize = 2;
                else
                        pAnim->nZoomSize = g_SFT.GetZoomSize();
                pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),rc.right,rc.↙
                    ↪ bottom);
                pAnim->hWnd = hWnd;
3575            pAnim->pOffs = p;
                pAnim->bZoomOut = FALSE;
                pAnim->bZoomOne = FALSE;
                UpdateBkpImage(pAnim);
                if(g_SFT.GetAnimateZoom()){
3580                    DWORD dw;
                        HANDLE hThread = CreateThread(NULL,0,(↙
                            ↪ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                            ↪ ;
                        CloseHandle(hThread);
                }
                else
3585                    delete pAnim;

                SYSTEMTIME st;
                GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
3590            p.x = (short)(p.x)*g_SFT.GetWidth()/rc.right;
                p.y = (short)(p.y)*g_SFT.GetHeight()/rc.bottom;
                g_SFT.UndoStore();
                g_SFT.Zoom(p.x,p.y,(lParam==9?2:g_SFT.GetZoomSize()),g_SFT.↙
                    ↪ GetWidth(),g_SFT.GetHeight(),FALSE);
                SetTimer(hWnd,0,500,NULL);
3595            MSG msg;
                while(PeekMessage(&msg,hWnd,WM_KEYDOWN,WM_KEYDOWN,PM_REMOVE));
        }
        else if(uMsg==WM_KEYDOWN && wParam==189){
                lParam=9;
3600            g_bAddReference=FALSE;
                g_bAddMainReference=false;
                g_bFindMinibrot=FALSE;
                g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
3605            g_bmSaveZoomBuff=NULL;
                while(g_bAnim){
                        Sleep(3);
                }
                POINT p;
3610            GetCursorPos(&p);
                ScreenToClient(hWnd,&p);
                RECT rc;
                GetClientRect(hWnd,&rc);
                RECT sr;
3615            GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
                if(!g_SFT.GetAnimateZoom()){
                        SendMessage(hWnd,WM_KEYDOWN,VK_ESCAPE,0);
3620                    SendMessage(hWnd,WM_RBUTTONDOWN,0,MAKELONG(p.x,p.y));
```

```
                        PostMessage(hWnd,WM_RBUTTONUP,0,MAKELONG(p.x,p.y));
                        return 0;
                }
                if(lParam==9){
                        p.x=rc.right/2 - (p.x-rc.right/2);
                        p.y=rc.bottom/2 - (p.y-rc.bottom/2);
                }

                g_pSelect.x = -g_SFT.GetWidth()/2;
                g_pSelect.y = g_SFT.GetHeight()/2;
                ANIM* pAnim = new ANIM;
                if(lParam==9)
                        pAnim->nZoomSize = 2;
                else
                        pAnim->nZoomSize = g_SFT.GetZoomSize();
                pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),rc.right,rc.↙
                    ↪ bottom);
                pAnim->hWnd = hWnd;
                pAnim->pOffs = p;
                pAnim->bZoomOut = TRUE;
                pAnim->bZoomOne = FALSE;
                UpdateBkpImage(pAnim);
                if(g_SFT.GetAnimateZoom()){
                        DWORD dw;
                        HANDLE hThread = CreateThread(NULL,0,(↙
                            ↪ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                            ↪ ;
                        CloseHandle(hThread);
                }
                else
                        delete pAnim;

                p.x = (short)(p.x)*g_SFT.GetWidth()/rc.right;
                p.y = (short)(p.y)*g_SFT.GetHeight()/rc.bottom;
                g_SFT.UndoStore();
                g_SFT.Zoom(p.x,p.y,(lParam==9?.5:(double)1/(double)g_SFT.↙
                    ↪ GetZoomSize()),g_SFT.GetWidth(),g_SFT.GetHeight(),FALSE);
                SetTimer(hWnd,0,500,NULL);
                MSG msg;
                while(PeekMessage(&msg,hWnd,WM_KEYDOWN,WM_KEYDOWN,PM_REMOVE));
        }
        else if(uMsg==0x020A){//WM_MOUSEWHEEL
                if((short)HIWORD(wParam)>0)
                        SendMessage(hWnd,WM_KEYDOWN,187,9);
                else
                        SendMessage(hWnd,WM_KEYDOWN,189,9);
                MSG msg;
                while(PeekMessage(&msg,hWnd,0x020A,0x020A,PM_REMOVE));
        }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SHOWITERATIONS){
                g_SFT.ApplyIterationColors();
                InvalidateRect(hWnd,NULL,FALSE);
        }
        else if(uMsg==WM_COMMAND && wParam==↙
            ↪ ID_ACTIONS_SPECIAL_SHOWSMOOTHTRANSITIONCOLORS){
                g_SFT.ApplySmoothColors();
                InvalidateRect(hWnd,NULL,FALSE);
```

```
            }
            else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_ANIMATEZOOM){
3675                g_SFT.SetAnimateZoom(!g_SFT.GetAnimateZoom());
                    UpdateAnimateZoom(hWnd);
            }
            else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_ARBITRARYSIZE){
                    if(g_SFT.GetArbitrarySize()){
3680                        g_SFT.SetRatio(640,360);
                            g_SFT.SetArbitrarySize(false);
                            SendMessage(hWnd,WM_SIZE,0,0);
                            g_SFT.SetArbitrarySize(true);
                            SendMessage(hWnd,WM_KEYDOWN,VK_F5,0);
3685                    }
                    g_SFT.SetArbitrarySize(!g_SFT.GetArbitrarySize());
                    CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_ARBITRARYSIZE,↙
                        ↳ MF_BYCOMMAND|(g_SFT.GetArbitrarySize()?MF_CHECKED:↙
                        ↳ MF_UNCHECKED));
            }
            else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_NEWTON){
3690                g_bNewton=!g_bNewton;
                    CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_NEWTON,MF_BYCOMMAND|(↙
                        ↳ g_bNewton?MF_CHECKED:MF_UNCHECKED));
                    if(!g_bNewton && g_hwNewton){
                            if(g_bNewtonRunning){
                                    g_bNewtonStop=TRUE;
3695                                while(g_bNewtonRunning)
                                            Sleep(10);
                            }
                            DestroyWindow(g_hwNewton);
                            g_hwNewton=NULL;
3700                    }
                    else{
                            g_hwNewton = CreateDialog(GetModuleHandle(NULL),↙
                                ↳ MAKEINTRESOURCE(IDD_DIALOG12),hWnd,(DLGPROC)↙
                                ↳ NewtonProc);
                            ShowWindow(g_hwNewton,SW_SHOW);
                    }
3705        }

#ifdef KF_OPENCL
            else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_OPENCL){
                    if(g_hwOpenCL){
3710                        DestroyWindow(g_hwOpenCL);
                            g_hwOpenCL=NULL;
                    }
                    g_hwOpenCL = CreateDialog(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                        ↳ IDD_DIALOG_OPENCL),hWnd,(DLGPROC)OpenCLProc);
                    ShowWindow(g_hwOpenCL,SW_SHOW);
3715        }
#endif

            else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SKEW){
                    g_bShowSkew=TRUE;
3720                g_DialogInit=0;
                    SYSTEMTIME st;

                    g_SFT.Stop();
```

```
                g_nSkewStretch = 100*g_SFT.GetRatioY()/360;
3725            SIZE size;
                size.cx = g_SFT.GetWidth();
                size.cy = g_SFT.GetHeight();
                double xRatio = 640.0/size.cx;
                size.cx = 640;
3730            size.cy = size.cy*xRatio;
                xRatio = (double)360/(double)size.cy;
                g_nSkewStretch*=xRatio;

                g_nSkewRotate = 180*g_Degree/pi;
3735            HBITMAP bmBmp = g_SFT.GetBitmap();
                UnSkewImage(bmBmp);
                g_SFT.UpdateBitmap();

                if(DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                    ↪ IDD_DIALOG10),hWnd,(DLGPROC)SkewProc,(LPARAM)hWnd)){
3740                    g_bShowSkew=FALSE;

                        SIZE size;
                        size.cx = g_SFT.GetWidth();
                        size.cy = g_SFT.GetHeight();
3745                    double xRatio = 640.0/size.cx;
                        size.cx = 640;
                        size.cy = size.cy*xRatio;
                        xRatio = (double)size.cy/(double)360;
                        g_nSkewStretch*=xRatio;
3750
                        g_SFT.SetRatio(640,360*g_nSkewStretch/100);
                        g_Degree = pi*(double)(g_nSkewRotate)/180;
                        GetLocalTime(&st);
                        SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
3755                    g_SFT.UpdateBitmap();
                        g_SFT.UndoStore();
                        DisableUnsafeMenus(hWnd);
                        g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),↙
                            ↪ g_SFT.GetIterations(),hWnd);
                        SetTimer(hWnd,0,500,NULL);
3760                }
                else{
                        g_bShowSkew=FALSE;
                        g_SFT.ApplyColors();
                        InvalidateRect(hWnd,NULL,FALSE);
3765                }

        }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SHOWINFLECTION){
                g_bShowInflection=!g_bShowInflection;
3770            g_nInflection=0;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SHOWINFLECTION,↙
                    ↪ MF_BYCOMMAND|(g_bShowInflection?MF_CHECKED:MF_UNCHECKED));
                InvalidateRect(hWnd,NULL,FALSE);
        }
        else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_SHOWGLITCHES){
3775            g_SFT.SetShowGlitches(!g_SFT.GetShowGlitches());
                CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_SHOWGLITCHES,MF_BYCOMMAND↙
                    ↪ |(g_SFT.GetShowGlitches()?MF_CHECKED:MF_UNCHECKED));
```

```
	}
	else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_NOREUSECENTER){
		g_SFT.SetNoReuseCenter(!g_SFT.GetNoReuseCenter());
		CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_NOREUSECENTER,↙
		    ↳ MF_BYCOMMAND|(g_SFT.GetNoReuseCenter()?MF_CHECKED:↙
		    ↳ MF_UNCHECKED));
	}
	else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_SHOWCROSSHAIR){
		g_SFT.SetShowCrossHair(!g_SFT.GetShowCrossHair());
		UpdateShowCrossHair(hWnd);
	}
	else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_ROTATE){
		g_bRotate=!g_bRotate;
		CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ROTATE,MF_BYCOMMAND|(↙
		    ↳ g_bRotate?MF_CHECKED:MF_UNCHECKED));
	}
	else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_RESETROTATION){
		g_SFT.Stop();
		g_bAnim=false;
		g_Degree=0;
		SetTimer(hWnd,0,500,NULL);
		g_SFT.UndoStore();
		DisableUnsafeMenus(hWnd);
		g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.↙
		    ↳ GetIterations(),hWnd);
	}
	else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_SPECIAL_MIRROR1){
		if(g_SFT.GetMirror()==1)
			g_SFT.SetMirror(0);
		else
			g_SFT.SetMirror(1);
		UpdateMirror(hWnd);
	}
	else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_SPECIAL_SETRATIO)↙
	    ↳ {
		g_JpegParams.nWidth = g_SFT.GetRatioX();
		g_JpegParams.nHeight = g_SFT.GetRatioY();
		if(!DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
		    ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,2))
			return 0;
		g_SFT.UndoStore();
		g_SFT.SetRatio(g_JpegParams.nWidth,g_JpegParams.nHeight);
		SetTimer(hWnd,0,500,NULL);
		DisableUnsafeMenus(hWnd);
		g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.↙
		    ↳ GetIterations(),hWnd);
	}
	else if(uMsg==WM_COMMAND && wParam==↙
	    ↳ ID_ACTIONS_SPECIAL_SPECIAL_RESETRATIO){
		g_SFT.UndoStore();
		g_SFT.SetRatio(640,360);
		SetTimer(hWnd,0,500,NULL);
		DisableUnsafeMenus(hWnd);
		g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.↙
		    ↳ GetIterations(),hWnd);
	}
	else if(uMsg==WM_COMMAND && wParam==ID_FILE_SAVEMAP){
```

```
3825                    std::string file = "";
                        if(BrowseFile(hWnd,FALSE,"Save Map","Kalle's fraktaler\0*.kfb↙
                            ↳ \0\0",file))
                                g_SFT.SaveMapB(file);
                }
                else if(uMsg==WM_COMMAND && (wParam==ID_FILE_STOREZOOMOUTIMAGES)){
3830                    MainProc(hWnd,WM_COMMAND,ID_FILE_SAVEAS_,0);
                        g_JpegParams.nWidth = g_SFT.GetWidth();
                        g_JpegParams.nHeight = g_SFT.GetHeight();
                        g_JpegParams.nQuality = 100;
                        if(!DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                            ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,0))
3835                            return 0;
                        if(!DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                            ↳ IDD_STOREZOOM),hWnd,(DLGPROC)StoreZoomProc,0))
                                return 0;
                        std::string path = get_filename_path(g_szFile);
                        if(!Browse(hWnd,path))
3840                            return 0;
                        if (path[path.length() - 1] != '\\')
                                path += "\\";
                        SetTimer(hWnd,0,500,NULL);
                        g_bStoreZoom=1;
3845                    std::string szFile = path;
                        szFile = replace_path_filename(szFile, store_zoom_filename(↙
                            ↳ g_bStoreZoom, "*", "kfb"));
                        while(FileExists(szFile)){
                                g_bStoreZoom++;
                                szFile = replace_path_filename(szFile, ↙
                                    ↳ store_zoom_filename(g_bStoreZoom, "*", "kfb"));
3850                    }
                        g_SFT.StoreLocation();
                        DisableUnsafeMenus(hWnd);
                        g_SFT.RenderFractal(g_JpegParams.nWidth,g_JpegParams.nHeight,↙
                            ↳ g_SFT.GetIterations(),hWnd);
                }
3855            else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_CENTERCURSOR) || (uMsg==↙
                    ↳ WM_KEYDOWN && wParam=='U' && HIWORD(GetKeyState(VK_CONTROL)))){
                        POINT p;
                        int p_x, p_y;
                        if(g_SFT.Center(p_x, p_y)){
                                p.x = p_x;
3860                            p.y = p_y;
                                RECT rc;
                                GetClientRect(hWnd,&rc);
                                RECT sr;
                                GetWindowRect(g_hwStatus,&sr);
3865                            sr.bottom-=sr.top;
                                rc.bottom-=sr.bottom;
                                p.x = p.x*rc.right/g_SFT.GetWidth();
                                p.y = p.y*rc.bottom/g_SFT.GetHeight();

3870                            ClientToScreen(hWnd,&p);
                                SetCursorPos(p.x,p.y);
                        }
                        else
                                MessageBox(hWnd,"Could not find center","Kalle's ↙
```

```
                    ↳ Fraktaler",MB_OK|MB_ICONINFORMATION);
3875        }
            else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_NOAPPROXIMATION){
                g_SFT.SetNoApprox(!g_SFT.GetNoApprox());
                UpdateNoApprox(hWnd);
            }
3880
            else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_UNDO) || (uMsg==↙
                ↳ WM_KEYDOWN && wParam=='Z' && HIWORD(GetKeyState(VK_CONTROL)))){
                g_SFT.Stop();
                g_bAnim=false;
                g_nPrevGlitchX=g_nPrevGlitchY=-1;
3885            g_bFindMinibrot=FALSE;
                g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
                g_bmSaveZoomBuff=NULL;
                SYSTEMTIME st;
3890            GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
                g_nPrevGlitchX=g_nPrevGlitchY=-1;
                SetTimer(hWnd,0,500,NULL);
                g_SFT.Undo();
3895            DisableUnsafeMenus(hWnd);
                g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.↙
                    ↳ GetIterations(),hWnd);
            }
            else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_REDO) || (uMsg==↙
                ↳ WM_KEYDOWN && wParam=='Y' && HIWORD(GetKeyState(VK_CONTROL)))){
                g_SFT.Stop();
3900            g_bAnim=false;
                g_nPrevGlitchX=g_nPrevGlitchY=-1;
                g_bFindMinibrot=FALSE;
                g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
3905            g_bmSaveZoomBuff=NULL;
                SYSTEMTIME st;
                GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
                g_nPrevGlitchX=g_nPrevGlitchY=-1;
3910            SetTimer(hWnd,0,500,NULL);
                g_SFT.Redo();
                DisableUnsafeMenus(hWnd);
                g_SFT.RenderFractal(g_SFT.GetWidth(),g_SFT.GetHeight(),g_SFT.↙
                    ↳ GetIterations(),hWnd);
            }
3915
            else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_SETIMAGESIZE) || (uMsg==↙
                ↳ WM_KEYDOWN && wParam=='W' && HIWORD(GetKeyState(VK_CONTROL)) && ↙
                ↳ HIWORD(GetKeyState(VK_SHIFT)))){
                g_JpegParams.nWidth = g_SFT.GetWidth();
                g_JpegParams.nHeight = g_SFT.GetHeight();
                if(!DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                    ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,1))
3920                return 0;
                g_SFT.Stop();
                g_bAnim=false;
                g_nPrevGlitchX=g_nPrevGlitchY=-1;
```

```
                g_bFindMinibrot=FALSE;
3925            g_bStoreZoom=FALSE;
                DeleteObject(g_bmSaveZoomBuff);
                g_bmSaveZoomBuff=NULL;
                SYSTEMTIME st;
                GetLocalTime(&st);
3930            SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
                g_nPrevGlitchX=g_nPrevGlitchY=-1;
                SetTimer(hWnd,0,500,NULL);
                DisableUnsafeMenus(hWnd);
                g_SFT.RenderFractal(g_JpegParams.nWidth,g_JpegParams.nHeight,↙
                    ↳ g_SFT.GetIterations(),hWnd);
3935        }
            else if((uMsg==WM_COMMAND && wParam==ID_ACTIONS_SETWINDOWSIZE) || (uMsg↙
                ↳ ==WM_KEYDOWN && wParam=='W' && HIWORD(GetKeyState(VK_CONTROL)) && ↙
                ↳ !HIWORD(GetKeyState(VK_SHIFT)))){
                RECT wr, cr;
                GetClientRect(hWnd,&cr);
                g_scSize.cx = cr.right;
3940            g_scSize.cy = cr.bottom;
                RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                g_JpegParams.nWidth = cr.right;
3945            g_JpegParams.nHeight = cr.bottom-sr.bottom;
                if(!DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                    ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,3))
                        return 0;
                GetWindowRect(hWnd,&wr);
                wr.right-=wr.left;
3950            wr.bottom-=wr.top;
                int nXOffs = g_JpegParams.nWidth-cr.right;
                wr.right+=nXOffs;
                wr.left-=nXOffs/2;
                int nYOffs = g_JpegParams.nHeight-cr.bottom;
3955            wr.bottom+=nYOffs+sr.bottom;
                wr.top-=nYOffs/2+sr.bottom/2;
                g_SFT.SetWindowWidth(g_JpegParams.nWidth);
                g_SFT.SetWindowHeight(g_JpegParams.nHeight);
                g_SFT.SetWindowLeft(wr.left);
3960            g_SFT.SetWindowTop(wr.top);
                g_SFT.SetWindowRight(wr.right);
                g_SFT.SetWindowBottom(wr.bottom);
                UpdateWindowSize(hWnd);
            }
3965        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_FINDHIGHESTITERATION){
                POINT p;
                int p_x, p_y;
                if(g_SFT.HighestIteration(p_x, p_y)){
                        p.x = p_x;
3970                    p.y = p_y;
                        RECT rc;
                        GetClientRect(hWnd,&rc);
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
3975                    sr.bottom-=sr.top;
                        rc.bottom-=sr.bottom;
```

```
                        p.x = p.x*rc.right/g_SFT.GetWidth();
                        p.y = p.y*rc.bottom/g_SFT.GetHeight();
                        ClientToScreen(hWnd,&p);
3980                    SetCursorPos(p.x,p.y);
                }
        }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_ADDREFERENCE){
                g_bAddReference=TRUE;
3985            CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ADDREFERENCE,MF_BYCOMMAND↙
                    ↪ |MF_CHECKED);

                g_bAddMainReference=false;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_SETMAINREFERENCE,↙
                    ↪ MF_BYCOMMAND|MF_UNCHECKED);
        }
3990    else if(uMsg==WM_COMMAND && wParam==↙
            ↪ ID_ACTIONS_SPECIAL_ADDREFERENCEERRORS){
                g_bAddReference=FALSE;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ADDREFERENCE,MF_BYCOMMAND↙
                    ↪ |MF_UNCHECKED);
                g_bAddMainReference=false;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_SETMAINREFERENCE,↙
                    ↪ MF_BYCOMMAND|MF_UNCHECKED);
3995    }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_SETMAINREFERENCE)↙
            ↪ {
                g_bAddMainReference=true;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_SETMAINREFERENCE,↙
                    ↪ MF_BYCOMMAND|MF_CHECKED);

4000            g_bAddReference=FALSE;
                CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_ADDREFERENCE,MF_BYCOMMAND↙
                    ↪ |MF_UNCHECKED);
        }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_FINDCENTEROFGLITCH){
                POINT p;
4005            int p_x, p_y;
                if(g_SFT.FindCenterOfGlitch(p_x, p_y)){
                        RECT rc;
                        GetClientRect(hWnd,&rc);
                        RECT sr;
4010                    GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
                        rc.bottom-=sr.bottom;
                        p.x = p.x*rc.right/g_SFT.GetWidth();
                        p.y = p.y*rc.bottom/g_SFT.GetHeight();
4015                    ClientToScreen(hWnd,&p);
                        SetCursorPos(p.x,p.y);
                }
                else
                        MessageBox(hWnd,"Could not find any glitches","Kalle's ↙
                            ↪ Fraktaler",MB_OK|MB_ICONINFORMATION);
4020    }
        else if(uMsg==WM_COMMAND && wParam==ID_FILE_EXAMINEZOOMSEQUENCE){
                bool ok = Examine(hWnd);
                if (! ok)
                        return 0;
```

```
4025            }
                else if(uMsg==WM_COMMAND && wParam==ID_FILE_RESUMEZOOMSEQUENCE){
                    return ResumeZoomSequence(hWnd);
                }
                else if(uMsg==WM_COMMAND && wParam==↙
                    ↳ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD){
4030                g_SFT.SetSolveGlitchNear(! g_SFT.GetSolveGlitchNear());
                    UpdateSolveGlitchNear(hWnd);
                }
                else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_AUTOSOLVEGLITCHES){
                    g_bAutoGlitch=!g_bAutoGlitch;
4035                g_nPrevGlitchX=g_nPrevGlitchY=-1;
                    g_SFT.SetAutoSolveGlitches(g_bAutoGlitch);
                    UpdateAutoSolveGlitches(hWnd);
                    if(g_bAutoGlitch){
                            g_SFT.SetReuseReference(!g_bAutoGlitch);
4040                        UpdateReuseReference(hWnd);
                    }
                    else{
                            g_SFT.SetSolveGlitchNear(false);
                            CheckMenuItem(GetMenu(hWnd),↙
                                ↳ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD↙
                                ↳ ,MF_BYCOMMAND|(g_SFT.GetSolveGlitchNear()?↙
                                ↳ MF_CHECKED:MF_UNCHECKED));
4045                }
                }
                else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_AUTOITERATION){
                    g_SFT.SetAutoIterations(! g_SFT.GetAutoIterations());
                    CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_AUTOITERATION,↙
                        ↳ MF_BYCOMMAND|(g_SFT.GetAutoIterations()?MF_CHECKED:↙
                        ↳ MF_UNCHECKED));
4050            }
                else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_GUESSING){
                    g_SFT.SetGuessing(!g_SFT.GetGuessing());
                    CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_SPECIAL_GUESSING,↙
                        ↳ MF_BYCOMMAND|(g_SFT.GetGuessing()?MF_CHECKED:MF_UNCHECKED)↙
                        ↳ );
                }
4055            else if(uMsg==WM_COMMAND && wParam==↙
                    ↳ ID_ACTIONS_SPECIAL_USELONGDOUBLEFROMSTART){
                    if(g_nLDBL>100)
                    {
                            g_nLDBL=0;
                    }
4060                else{
                            if(scaling_supported(g_SFT.GetFractalType(), g_SFT.↙
                                ↳ GetPower(), g_SFT.GetDerivatives()))
                            {
                                if (g_SFT.GetPower() == 2)
                                {
4065                                g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_2;
                                    g_nEXP = FLOATEXP_THRESHOLD_POWER_2;
                                }
                                if (g_SFT.GetPower() == 3)
                                {
4070                                g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_3;
                                    g_nEXP = FLOATEXP_THRESHOLD_POWER_3;
```

```
                                }
                        }
                        else
4075                    {
                                g_nLDBL = LONG_DOUBLE_THRESHOLD_DEFAULT;
                                g_nEXP = FLOATEXP_THRESHOLD_DEFAULT;
                        }
                }
4080            g_SFT.SetFloatExpAlways(false);
                g_SFT.SetLongDoubleAlways(g_nLDBL < 100);
                UpdateLongDoubleAlways(hWnd);
                UpdateFloatExpAlways(hWnd);
        }
4085    else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_USEFLOATEXPALWAYS↙
            ↳ ){
                if(g_nEXP>100){
                        g_nLDBL=2;
                        g_nEXP=2;
                }
4090            else{
                        if(scaling_supported(g_SFT.GetFractalType(), g_SFT.↙
                            ↳ GetPower(), g_SFT.GetDerivatives()))
                        {
                                if (g_SFT.GetPower() == 2)
                                {
4095                                    g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_2;
                                        g_nEXP = FLOATEXP_THRESHOLD_POWER_2;
                                }
                                if (g_SFT.GetPower() == 3)
                                {
4100                                    g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_3;
                                        g_nEXP = FLOATEXP_THRESHOLD_POWER_3;
                                }
                        }
                        else
4105                    {
                                g_nLDBL = LONG_DOUBLE_THRESHOLD_DEFAULT;
                                g_nEXP = FLOATEXP_THRESHOLD_DEFAULT;
                        }
                }
4110            g_SFT.SetLongDoubleAlways(false);
                g_SFT.SetFloatExpAlways(g_nEXP < 3);
                UpdateLongDoubleAlways(hWnd);
                UpdateFloatExpAlways(hWnd);
        }
4115    else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_USENANOMB1){
                g_SFT.SetUseNanoMB1(! g_SFT.GetUseNanoMB1());
                if (g_SFT.GetUseNanoMB1()) g_SFT.SetUseNanoMB2(false);
                else g_SFT.SetInteriorChecking(false);
                UpdateUseNanoMB1(hWnd);
4120            UpdateUseNanoMB2(hWnd);
                UpdateInteriorChecking(hWnd);
        }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_USENANOMB2){
                g_SFT.SetUseNanoMB2(! g_SFT.GetUseNanoMB2());
4125            if (g_SFT.GetUseNanoMB2()) g_SFT.SetUseNanoMB1(false);
                else g_SFT.SetInteriorChecking(false);
```

```
                UpdateUseNanoMB1(hWnd);
                UpdateUseNanoMB2(hWnd);
                UpdateInteriorChecking(hWnd);
        }
        else if(uMsg==WM_COMMAND && wParam==ID_ACTIONS_SPECIAL_INTERIORCHECKING)↙
            ↳ {
                g_SFT.SetInteriorChecking(!g_SFT.GetInteriorChecking());
                if (!(g_SFT.GetUseNanoMB1() || g_SFT.GetUseNanoMB2()))
                    g_SFT.SetInteriorChecking(false);
                UpdateInteriorChecking(hWnd);
        }
        else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_HALFCOLOUR){
                g_SFT.SetHalfColour(!g_SFT.GetHalfColour());
                CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_HALFCOLOUR,MF_BYCOMMAND|(↙
                    ↳ g_SFT.GetHalfColour()?MF_CHECKED:MF_UNCHECKED));
        }
        else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_SAVE_OVERWRITES){
                g_SFT.SetSaveOverwrites(!g_SFT.GetSaveOverwrites());
                CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_SAVE_OVERWRITES,↙
                    ↳ MF_BYCOMMAND|(g_SFT.GetSaveOverwrites()?MF_CHECKED:↙
                    ↳ MF_UNCHECKED));
        }
        else if(uMsg==WM_COMMAND && wParam==ID_SPECIAL_THREADED_REFERENCE){
                g_SFT.SetThreadedReference(!g_SFT.GetThreadedReference());
                CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_THREADED_REFERENCE,↙
                    ↳ MF_BYCOMMAND|(g_SFT.GetThreadedReference()?MF_CHECKED:↙
                    ↳ MF_UNCHECKED));
        }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_VECTOR_SIZE_1){ g_SFT.↙
            ↳ SetSIMDVectorSize(1); UpdateSIMDVectorSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_VECTOR_SIZE_2){ g_SFT.↙
            ↳ SetSIMDVectorSize(2); UpdateSIMDVectorSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_VECTOR_SIZE_4){ g_SFT.↙
            ↳ SetSIMDVectorSize(4); UpdateSIMDVectorSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_VECTOR_SIZE_8){ g_SFT.↙
            ↳ SetSIMDVectorSize(8); UpdateSIMDVectorSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_VECTOR_SIZE_16){ g_SFT.↙
            ↳ SetSIMDVectorSize(16); UpdateSIMDVectorSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_1){ g_SFT.↙
            ↳ SetSIMDChunkSize(1); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_2){ g_SFT.↙
            ↳ SetSIMDChunkSize(2); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_4){ g_SFT.↙
            ↳ SetSIMDChunkSize(4); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_8){ g_SFT.↙
            ↳ SetSIMDChunkSize(8); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_16){ g_SFT.↙
            ↳ SetSIMDChunkSize(16); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_32){ g_SFT.↙
            ↳ SetSIMDChunkSize(32); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_64){ g_SFT.↙
            ↳ SetSIMDChunkSize(64); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_128){ g_SFT.↙
            ↳ SetSIMDChunkSize(128); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_256){ g_SFT.↙
            ↳ SetSIMDChunkSize(256); UpdateSIMDChunkSize(hWnd); }
        else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_512){ g_SFT.↙
```

```
        ↳ SetSIMDChunkSize(512); UpdateSIMDChunkSize(hWnd); }
    else if(uMsg==WM_COMMAND && wParam==ID_SIMD_CHUNK_SIZE_1024){ g_SFT.↙
        ↳ SetSIMDChunkSize(1024); UpdateSIMDChunkSize(hWnd); }
4165    else if(uMsg==WM_KEYDOWN && wParam==VK_LEFT && HIWORD(GetKeyState(↙
        ↳ VK_CONTROL))){
            RECT r;
            GetClientRect(hWnd,&r);
            RECT sr;
            GetWindowRect(g_hwStatus,&sr);
4170        r.bottom-=(sr.bottom-sr.top);
            if(g_SFT.GetAnimateZoom()){
                    g_pSelect.x = -g_SFT.GetWidth()/2;
                    g_pSelect.y = g_SFT.GetHeight()/2;
                    ANIM* pAnim = new ANIM;
4175                pAnim->nZoomSize = g_SFT.GetZoomSize();
                    pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),r.right,r↙
                        ↳ .bottom);
                    pAnim->hWnd = hWnd;
                    pAnim->pOffs.x = -r.right/2;
                    pAnim->pOffs.y = r.bottom/2;
4180                pAnim->bZoomOut = FALSE;
                    pAnim->bZoomOne = TRUE;
                    DWORD dw;
                    HANDLE hThread = CreateThread(NULL,0,(↙
                        ↳ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                        ↳ ;
                    CloseHandle(hThread);
4185        }
            g_SFT.UndoStore();
            g_SFT.Zoom(g_pSelect.x,g_pSelect.y,1,g_SFT.GetWidth(),g_SFT.↙
                ↳ GetHeight(),FALSE);
            SetTimer(hWnd,0,500,NULL);
    }
4190    else if(uMsg==WM_KEYDOWN && wParam==VK_RIGHT && HIWORD(GetKeyState(↙
        ↳ VK_CONTROL))){
            RECT r;
            GetClientRect(hWnd,&r);
            RECT sr;
            GetWindowRect(g_hwStatus,&sr);
4195        r.bottom-=(sr.bottom-sr.top);
            if(g_SFT.GetAnimateZoom()){
                    g_pSelect.x = g_SFT.GetWidth()+g_SFT.GetWidth()/2;
                    g_pSelect.y = g_SFT.GetHeight()/2;
                    ANIM* pAnim = new ANIM;
4200                pAnim->nZoomSize = g_SFT.GetZoomSize();
                    pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),r.right,r↙
                        ↳ .bottom);
                    pAnim->hWnd = hWnd;
                    pAnim->pOffs.x = r.right+r.right/2;
                    pAnim->pOffs.y = r.bottom/2;
4205                pAnim->bZoomOut = FALSE;
                    pAnim->bZoomOne = TRUE;
                    DWORD dw;
                    HANDLE hThread = CreateThread(NULL,0,(↙
                        ↳ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                        ↳ ;
                    CloseHandle(hThread);
```

```
4210                    }
                        g_SFT.UndoStore();
                        g_SFT.Zoom(g_pSelect.x,g_pSelect.y,1,g_SFT.GetWidth(),g_SFT.↙
                            ↪ GetHeight(),FALSE);
                        SetTimer(hWnd,0,500,NULL);
                }
4215            else if(uMsg==WM_KEYDOWN && wParam==VK_UP && HIWORD(GetKeyState(↙
                    ↪ VK_CONTROL))){
                        RECT r;
                        GetClientRect(hWnd,&r);
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
4220                    r.bottom-=(sr.bottom-sr.top);
                        if(g_SFT.GetAnimateZoom()){
                                g_pSelect.x = g_SFT.GetWidth()/2;
                                g_pSelect.y = -g_SFT.GetHeight()/2;
                                ANIM* pAnim = new ANIM;
4225                            pAnim->nZoomSize = g_SFT.GetZoomSize();
                                pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),r.right,r↙
                                    ↪ .bottom);
                                pAnim->hWnd = hWnd;
                                pAnim->pOffs.x = r.right/2;
                                pAnim->pOffs.y = -r.bottom/2;
4230                            pAnim->bZoomOut = FALSE;
                                pAnim->bZoomOne = TRUE;
                                DWORD dw;
                                HANDLE hThread = CreateThread(NULL,0,(↙
                                    ↪ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                                    ↪ ;
                                CloseHandle(hThread);
4235                    }
                        g_SFT.UndoStore();
                        g_SFT.Zoom(g_pSelect.x,g_pSelect.y,1,g_SFT.GetWidth(),g_SFT.↙
                            ↪ GetHeight(),FALSE);
                        SetTimer(hWnd,0,500,NULL);
                }
4240            else if(uMsg==WM_KEYDOWN && wParam==VK_DOWN && HIWORD(GetKeyState(↙
                    ↪ VK_CONTROL))){
                        RECT r;
                        GetClientRect(hWnd,&r);
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
4245                    r.bottom-=(sr.bottom-sr.top);
                        if(g_SFT.GetAnimateZoom()){
                                g_pSelect.x = g_SFT.GetWidth()/2;
                                g_pSelect.y = g_SFT.GetHeight()+g_SFT.GetHeight()/2;
                                ANIM* pAnim = new ANIM;
4250                            pAnim->nZoomSize = g_SFT.GetZoomSize();
                                pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),r.right,r↙
                                    ↪ .bottom);
                                pAnim->hWnd = hWnd;
                                pAnim->pOffs.x = r.right/2;
                                pAnim->pOffs.y = r.bottom+r.bottom/2;
4255                            pAnim->bZoomOut = FALSE;
                                pAnim->bZoomOne = TRUE;
                                DWORD dw;
                                HANDLE hThread = CreateThread(NULL,0,(↙
```

```
                                    ↪ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                                    ↪ ;
                                CloseHandle(hThread);
4260                        }
                            g_SFT.UndoStore();
                            g_SFT.Zoom(g_pSelect.x,g_pSelect.y,1,g_SFT.GetWidth(),g_SFT.↙
                                ↪ GetHeight(),FALSE);
                            SetTimer(hWnd,0,500,NULL);
                    }
4265                else if(uMsg==WM_KEYDOWN && wParam=='O' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_FILE_OPEN_,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='S' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_FILE_SAVE_,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='N' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
4270                        PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_RESET,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='N' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_FILE_SAVEAS_,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='2' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ZOOMSIZE_2,0);
4275                else if(uMsg==WM_KEYDOWN && wParam=='4' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ZOOMSIZE_4,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='8' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ZOOMSIZE_8,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='1' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
4280                        PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ZOOMSIZE_16,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='3' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ZOOMSIZE_32,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='6' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ZOOMSIZE_64,0);
4285                else if(uMsg==WM_KEYDOWN && wParam=='L' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_POSITION,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='C' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_SETCOLORS,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='D' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
4290                        PostMessage(hWnd,WM_COMMAND,ID_SPECIAL_NEWTON,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='I' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ITERATIONS,0);
                    else if(uMsg==WM_KEYDOWN && wParam=='E' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_REUSEREFERENCE,0);
4295                else if(uMsg==WM_KEYDOWN && wParam=='Q' && HIWORD(GetKeyState(VK_CONTROL↙
                        ↪ )))
                            PostMessage(hWnd,WM_COMMAND,ID_FILE_STOREZOOMOUTIMAGES,0);
```

```
        else if(uMsg==WM_KEYDOWN && wParam=='M' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_FINDMINIBROT,0);
        else if(uMsg==WM_KEYDOWN && wParam=='K' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )) &&  HIWORD(GetKeyState(VK_SHIFT)))
            PostMessage(hWnd,WM_COMMAND,↙
                ↳ ID_ACTIONS_SPECIAL_SPECIAL_RESETRATIO,0);
        else if(uMsg==WM_KEYDOWN && wParam=='K' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )) && !HIWORD(GetKeyState(VK_SHIFT)))
            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_SKEW,0);
        else if(uMsg==WM_KEYDOWN && wParam=='H' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_SHOWINFLECTION,0);
        else if(uMsg==WM_KEYDOWN && wParam=='H' && HIWORD(GetKeyState(VK_SHIFT))↙
            ↳ ){
            POINT p;
            GetCursorPos(&p);
            ScreenToClient(hWnd,&p);
            g_SFT.AddInflectionPont(p.x,p.y);
            PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
        }
        else if(uMsg==WM_KEYDOWN && wParam=='G' && HIWORD(GetKeyState(VK_SHIFT))↙
            ↳ ){
            g_SFT.RemoveInflectionPoint();
            PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
        }
#if 0
        else if(uMsg==WM_KEYDOWN && wParam=='X' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ ))){
            if(size_t(g_nInflection)<sizeof(g_pInflections)/sizeof(POINT)){
                GetCursorPos(&g_pInflections[g_nInflection]);
                ScreenToClient(hWnd,&g_pInflections[g_nInflection]);
                g_nInflection++;
            }
        }
#endif
        else if((uMsg==WM_KEYDOWN && wParam=='B' && HIWORD(GetKeyState(↙
            ↳ VK_CONTROL))) || (uMsg==WM_COMMAND && wParam==↙
            ↳ ID_SPECIAL_SKEWANIMATION)){
            g_bSkewAnimation=!g_bSkewAnimation;
            if(g_bSkewAnimation)
                g_bSkewAnimation=DialogBoxParam(GetModuleHandle(NULL),↙
                    ↳ MAKEINTRESOURCE(IDD_DIALOG11),hWnd,(DLGPROC)↙
                    ↳ SkewAnimateProc,(LPARAM)0);
            CheckMenuItem(GetMenu(hWnd),ID_SPECIAL_SKEWANIMATION,↙
                ↳ MF_BYCOMMAND|(g_bSkewAnimation?MF_CHECKED:MF_UNCHECKED));
            return 0;
        }
        else if(uMsg==WM_KEYDOWN && wParam=='J' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
            PostMessage(hWnd,WM_COMMAND,ID_FILE_SAVEASJPEG,0);
        else if(uMsg==WM_KEYDOWN && wParam=='P' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
            PostMessage(hWnd,WM_COMMAND,ID_FILE_SAVEASPNG,0);
        else if(uMsg==WM_KEYDOWN && wParam=='R' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
            PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ADDREFERENCE,0);
```

```
        else if(uMsg==WM_KEYDOWN && wParam=='T' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )) &&   HIWORD(GetKeyState(VK_SHIFT)))
                PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_RESETROTATION,0);
4340    else if(uMsg==WM_KEYDOWN && wParam=='T' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )) && ! HIWORD(GetKeyState(VK_SHIFT)))
                PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_ROTATE,0);
        else if(uMsg==WM_KEYDOWN && wParam=='F' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
                PostMessage(hWnd,WM_COMMAND,ID_ACTIONS_FINDCENTEROFGLITCH,0);
        else if(uMsg==WM_KEYDOWN && wParam=='A' && HIWORD(GetKeyState(VK_CONTROL↙
            ↳ )))
4345            PostMessage(hWnd,WM_COMMAND,ID_FILE_SAVEAS_,0);
        else if(uMsg==WM_RBUTTONUP && !g_bWaitRead){
                if(g_bTrackSelect==2)
                        g_bTrackSelect=0;
                else{
4350                    g_SFT.Stop();
                        g_bAnim=false;
                        SYSTEMTIME st;
                        GetLocalTime(&st);
                        SystemTimeToFileTime(&st,(LPFILETIME)&g_nTStart);
4355                    g_bFindMinibrot=FALSE;
                        g_bStoreZoom=FALSE;
                        DeleteObject(g_bmSaveZoomBuff);
                        g_bmSaveZoomBuff=NULL;
                        PostMessage(hWnd,WM_USER+299,wParam,lParam);
4360            }
        }
        else if(uMsg==WM_USER+299){
                RECT rc;
                GetClientRect(hWnd,&rc);
4365            RECT sr;
                GetWindowRect(g_hwStatus,&sr);
                sr.bottom-=sr.top;
                rc.bottom-=sr.bottom;
                g_pSelect.x = (short)LOWORD(lParam)*g_SFT.GetWidth()/rc.right;
4370            g_pSelect.y = (short)HIWORD(lParam)*g_SFT.GetHeight()/rc.bottom;
                if(g_SFT.GetAnimateZoom()){
                        ANIM* pAnim = new ANIM;
                        pAnim->nZoomSize = g_SFT.GetZoomSize();
                        pAnim->bmBmp = ShrinkBitmap2(g_SFT.GetBitmap(),rc.right,↙
                            ↳ rc.bottom);
4375                    pAnim->hWnd = hWnd;
                        pAnim->pOffs.x = (short)LOWORD(lParam);
                        pAnim->pOffs.y = (short)HIWORD(lParam);
                        pAnim->bZoomOut = TRUE;
                        pAnim->bZoomOne = FALSE;
4380                    DWORD dw;
                        HANDLE hThread = CreateThread(NULL,0,(↙
                            ↳ LPTHREAD_START_ROUTINE)ThAnim,(LPVOID)pAnim,0,&dw)↙
                            ↳ ;
                        CloseHandle(hThread);

                        RECT r = {0,0,g_SFT.GetWidth(),g_SFT.GetHeight()};
4385                    double zoomDiff = (double)1/(double)g_SFT.GetZoomSize();
                        int nToXStart = g_pSelect.x-r.right/(zoomDiff*2);
                        int nToYStart = g_pSelect.y-r.bottom/(zoomDiff*2);
```

```
                                int nToXStop = r.right - (g_pSelect.x+r.right/(zoomDiff↙
                                    ↳ *2));
                                int nToYStop = r.bottom - (g_pSelect.y+r.bottom/(↙
                                    ↳ zoomDiff*2));
4390                            HDC hDC = GetDC(NULL);
                                HDC dcBmp = CreateCompatibleDC(hDC);
                                HBITMAP bmBmp = g_SFT.GetBitmap();
                                HBITMAP bmBmp2 = ShrinkBitmap2(bmBmp, r.right, r.bottom);
                                HBITMAP bmOld = (HBITMAP)SelectObject(dcBmp,bmBmp);
4395                            SetStretchBltMode(dcBmp,HALFTONE);
                                HDC dcBmp2 = CreateCompatibleDC(hDC);
                                HBITMAP bmOld2 = (HBITMAP)SelectObject(dcBmp2,bmBmp2);
                                FillRect(dcBmp,&r,(HBRUSH)GetStockObject(BLACK_BRUSH));
                                StretchBlt(dcBmp,0,0,r.right,r.bottom,dcBmp2,nToXStart,↙
                                    ↳ nToYStart,r.right-nToXStop-nToXStart,r.bottom-↙
                                    ↳ nToYStop-nToYStart,SRCCOPY);
4400                            SelectObject(dcBmp,bmOld);
                                DeleteDC(dcBmp);
                                SelectObject(dcBmp2,bmOld2);
                                DeleteDC(dcBmp2);
                                DeleteObject(bmBmp2);
4405                            ReleaseDC(NULL,hDC);
                                g_SFT.UpdateBitmap();
                        }
                        bool bReuseCenter = (g_SFT.GetZoomSize() == round(g_SFT.↙
                            ↳ GetZoomSize()));
                        if(!g_bAutoGlitch && g_SFT.GetReuseReference() && g_pSelect.x==↙
                            ↳ g_SFT.GetWidth()/2 && g_pSelect.y==g_SFT.GetHeight()/2)
4410                            g_SFT.Zoom(g_pSelect.x,g_pSelect.y,1/(double)g_SFT.↙
                                    ↳ GetZoomSize(),g_SFT.GetWidth(),g_SFT.GetHeight(),↙
                                    ↳ bReuseCenter);
                        else
                                g_SFT.Zoom(g_pSelect.x,g_pSelect.y,1/(double)g_SFT.↙
                                    ↳ GetZoomSize(),g_SFT.GetWidth(),g_SFT.GetHeight());
                        SetTimer(hWnd,0,500,NULL);
                }
4415            else if(!g_SFT.GetArbitrarySize() && uMsg==WM_SIZING){
                        RECT sr, cr;
                        LPRECT pwr = (LPRECT)lParam;
                        pwr->right-=pwr->left;
                        pwr->bottom-=pwr->top;
4420                    GetClientRect(hWnd,&cr);
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
                        cr.bottom-=sr.bottom;
                        int nXOffs = 16*cr.bottom/9 - cr.right;
4425                    pwr->right+=nXOffs;

    /*                  MoveWindow(hWnd,pwr->left,pwr->top,pwr->right,pwr->bottom,TRUE);
                        SendMessage(g_hwStatus,uMsg,wParam,lParam);
                        InvalidateRect(hWnd,NULL,FALSE);
4430    */
                        pwr->right+=pwr->left;
                        pwr->bottom+=pwr->top;
                        return TRUE;
                }
4435            else if(uMsg==WM_SIZE){
```

```
                if(!g_SFT.GetArbitrarySize()){
                        RECT wr, cr;
                        GetWindowRect(hWnd,&wr);
                        wr.right-=wr.left;
4440                    wr.bottom-=wr.top;
                        GetClientRect(hWnd,&cr);
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
4445                    cr.bottom-=sr.bottom;
                        int nXOffs = 16*cr.bottom/9 - cr.right;
                        wr.right+=nXOffs;
                        MoveWindow(hWnd,wr.left,wr.top,wr.right,wr.bottom,TRUE);
                }
4450            InvalidateRect(hWnd,NULL,TRUE);
                SendMessage(g_hwStatus,uMsg,wParam,lParam);
                if(g_SFT.GetArbitrarySize() && !g_bResizing && (wParam==↙
                    ↳ SIZE_MAXIMIZED || wParam==SIZE_RESTORED))
                        SendMessage(hWnd,WM_EXITSIZEMOVE,0,0);
        }
4455    else if(uMsg==WM_ENTERSIZEMOVE && g_SFT.GetArbitrarySize()){
                g_bResizing=TRUE;
                RECT cr;
                GetClientRect(hWnd,&cr);
                g_scSize.cx = cr.right;
4460            g_scSize.cy = cr.bottom;
        }
        else if(uMsg==WM_EXITSIZEMOVE && g_SFT.GetArbitrarySize() && !↙
            ↳ g_bFirstDone){
                g_bResizing=FALSE;
                RECT cr;
4465            GetClientRect(hWnd,&cr);
                if(cr.right!=g_scSize.cx || cr.bottom!=g_scSize.cy){
                        RECT sr;
                        GetWindowRect(g_hwStatus,&sr);
                        sr.bottom-=sr.top;
4470                    cr.bottom-=sr.bottom;
                        g_scSize.cy-=sr.bottom;
                        double xRatio = 640.0/cr.right;
                        cr.right = 640;
                        cr.bottom = cr.bottom*xRatio;
4475                    xRatio = 640.0/g_scSize.cx;
                        g_scSize.cx = 640;
                        g_scSize.cy = g_scSize.cy*xRatio;
                        xRatio = (double)g_SFT.GetRatioY()/(double)g_scSize.cy;
                        g_SFT.SetRatio(cr.right,cr.bottom*xRatio);
4480                    PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                }
                GetClientRect(hWnd,&cr);
                g_scSize.cx = cr.right;
                g_scSize.cy = cr.bottom;
4485    }
        else if((uMsg==WM_USER+199 || uMsg==WM_TIMER) && !g_bMove){
                return HandleDoneSEH(hWnd,uMsg,wParam,lParam);
        }
        else if(uMsg==WM_COMMAND){
4490            if(wParam==ID_ACTIONS_POSITION){
```

```
//                          MessageBox(NULL,g_SFT.GetPosition(),g_SFT.ToZoom(),MB_OK↙
    ↳ );
                            if(DialogBox(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                                ↳ IDD_DIALOG5),hWnd,(DLGPROC)PositionProc))
                            {
                                    g_SFT.UndoStore();
4495                                PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                            }
                        }
                        else if(wParam==ID_ACTIONS_EXIT)
                            PostQuitMessage(0);
4500                    else if(wParam==ID_ACTIONS_REFRESH)
                            PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                        else if(wParam==ID_ACTIONS_CANCELRENDERING)
                            PostMessage(hWnd,WM_KEYDOWN,VK_ESCAPE,0);
                        else if(wParam==ID_ACTIONS_ITERATIONS){
4505                        INT_PTR n = DialogBoxParam(GetModuleHandle(NULL),↙
                                ↳ MAKEINTRESOURCE(IDD_DIALOG2),hWnd,IterationProc,0)↙
                                ↳ ;
                            if(n > 0){
                                    SetTimer(hWnd,0,500,NULL);
                                    RECT r;
                                    GetClientRect(hWnd,&r);
4510                                RECT sr;
                                    GetWindowRect(g_hwStatus,&sr);
                                    sr.bottom-=sr.top;
                                    r.bottom-=sr.bottom;
                                    g_SFT.Stop();
4515                                g_bAnim=false;
                                    g_SFT.UndoStore();
                                    g_SFT.SetIterations(g_SFT.GetIterations());
                                    PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                            }
4520                    }
                        else if(wParam==ID_ACTIONS_SETCOLORS){
                            if(!g_hwColors)
                                    g_hwColors = CreateDialog(GetModuleHandle(NULL),↙
                                        ↳ MAKEINTRESOURCE(IDD_DIALOG3),hWnd,(DLGPROC↙
                                        ↳ )ColorProc);
                            ShowWindow(g_hwColors,SW_SHOW);
4525                    }
                        else if(wParam==ID_ACTIONS_ZOOMSIZE_1 || wParam==↙
                            ↳ ID_ACTIONS_ZOOMSIZE_2 ||
                            wParam==ID_ACTIONS_ZOOMSIZE_4 ||
                            wParam==ID_ACTIONS_ZOOMSIZE_8 ||
                            wParam==ID_ACTIONS_ZOOMSIZE_16 ||
4530                        wParam==ID_ACTIONS_ZOOMSIZE_32 ||
                            wParam==ID_ACTIONS_ZOOMSIZE_64 ||
                            wParam==ID_ACTIONS_ZOOMSIZE_128 ||
                            wParam==ID_ZOOMSIZE_CUSTOM){
                            if(wParam==ID_ACTIONS_ZOOMSIZE_1)
4535                                g_SFT.SetZoomSize(1);
                            if(wParam==ID_ACTIONS_ZOOMSIZE_2)
                                    g_SFT.SetZoomSize(2);
                            else if(wParam==ID_ACTIONS_ZOOMSIZE_4)
                                    g_SFT.SetZoomSize(4);
4540                        else if(wParam==ID_ACTIONS_ZOOMSIZE_8)
```

```
                        g_SFT.SetZoomSize(8);
                    else if(wParam==ID_ACTIONS_ZOOMSIZE_16)
                        g_SFT.SetZoomSize(16);
                    else if(wParam==ID_ACTIONS_ZOOMSIZE_32)
                        g_SFT.SetZoomSize(32);
                    else if(wParam==ID_ACTIONS_ZOOMSIZE_64)
                        g_SFT.SetZoomSize(64);
                    else if(wParam==ID_ACTIONS_ZOOMSIZE_128)
                        g_SFT.SetZoomSize(128);
                    else if(wParam==ID_ZOOMSIZE_CUSTOM){
                        char szTmp[25];
                        if(DialogBoxParam(GetModuleHandle(NULL),↙
                            ↳ MAKEINTRESOURCE(IDD_DIALOG9),hWnd,(DLGPROC↙
                            ↳ )CustomZoomSize,(LPARAM)szTmp)){
                                g_SFT.SetZoomSize(atof(szTmp));
                                if(!g_SFT.GetZoomSize())
                                        g_SFT.SetZoomSize(2);
                        }
                    }
                    UpdateZoomSize(hWnd);
                }
                else if(wParam==ID_IMAGE_SHRINK_FAST ||
                    wParam==ID_IMAGE_SHRINK_DEFAULT ||
                    wParam==ID_IMAGE_SHRINK_BEST){
                        if(wParam==ID_IMAGE_SHRINK_FAST)    g_SFT.SetShrink↙
                            ↳ (0);
                    else if(wParam==ID_IMAGE_SHRINK_DEFAULT) g_SFT.SetShrink↙
                        ↳ (1);
                    else if(wParam==ID_IMAGE_SHRINK_BEST)    g_SFT.SetShrink↙
                        ↳ (2);
                    UpdateShrink(hWnd);
                    InvalidateRect(hWnd,NULL,FALSE);
                }
                else if(wParam==ID_SPECIAL_PRESET_FAST){
                    g_SFT.SetIsolatedGlitchNeighbourhood(4);
                    UpdateIsolatedGlitchNeighbourhood(hWnd);
                    g_SFT.SetGuessing(true);
                    UpdateGuessing(hWnd);
                    g_SFT.SetGlitchLowTolerance(false);
                    g_SFT.SetApproxLowTolerance(false);
                    if (g_SFT.GetDifferences() != Differences_Analytic)
                    {
                            g_SFT.SetDerivatives(false);
                    }
                }
                else if(wParam==ID_SPECIAL_PRESET_BEST){
                    g_SFT.SetIsolatedGlitchNeighbourhood(0);
                    UpdateIsolatedGlitchNeighbourhood(hWnd);
                    g_SFT.SetGuessing(false);
                    UpdateGuessing(hWnd);
                    g_SFT.SetGlitchLowTolerance(true);
                    g_SFT.SetApproxLowTolerance(true);
                    g_SFT.SetJitterSeed(1);
                }
                else if(wParam==ID_ACTIONS_THREADS_1_4){
                    g_SFT.SetThreadsPerCore(1./4);
                    UpdateThreadsPerCore(hWnd);
```

```
		}
		else if(wParam==ID_ACTIONS_THREADS_1_2){
4595			g_SFT.SetThreadsPerCore(1./2);
			UpdateThreadsPerCore(hWnd);
		}
		else if(wParam==ID_ACTIONS_THREADS_1){
			g_SFT.SetThreadsPerCore(1);
4600			UpdateThreadsPerCore(hWnd);
		}
		else if(wParam==ID_ACTIONS_THREADS_2){
			g_SFT.SetThreadsPerCore(2);
			UpdateThreadsPerCore(hWnd);
4605		}
		else if(wParam==ID_ACTIONS_THREADS_4){
			g_SFT.SetThreadsPerCore(4);
			UpdateThreadsPerCore(hWnd);
		}
4610		else if(wParam==ID_ACTIONS_THREADS_RESERVE_CORE){
			g_SFT.SetThreadsReserveCore(!g_SFT.↙
			    ↪ GetThreadsReserveCore());
			UpdateThreadsReserveCore(hWnd);
		}
		else if(wParam==ID_ACTIONS_ISOLATED_0){
4615			g_SFT.SetIsolatedGlitchNeighbourhood(0);
			UpdateIsolatedGlitchNeighbourhood(hWnd);
		}
		else if(wParam==ID_ACTIONS_ISOLATED_4){
			g_SFT.SetIsolatedGlitchNeighbourhood(4);
4620			UpdateIsolatedGlitchNeighbourhood(hWnd);
		}
		else if(wParam==ID_ACTIONS_ISOLATED_8){
			g_SFT.SetIsolatedGlitchNeighbourhood(8);
			UpdateIsolatedGlitchNeighbourhood(hWnd);
4625		}
		else if(wParam==ID_ACTIONS_REUSEREFERENCE){
			g_SFT.SetReuseReference(!g_SFT.GetReuseReference());
			CheckMenuItem(GetMenu(hWnd),ID_ACTIONS_REUSEREFERENCE,↙
			    ↪ MF_BYCOMMAND|(g_SFT.GetReuseReference()?MF_CHECKED↙
			    ↪ :MF_UNCHECKED));
			if(g_SFT.GetReuseReference()){
4630				g_SFT.SetAutoSolveGlitches(!g_SFT.↙
				    ↪ GetReuseReference());
				CheckMenuItem(GetMenu(hWnd),↙
				    ↪ ID_ACTIONS_AUTOSOLVEGLITCHES,MF_BYCOMMAND↙
				    ↪ |(g_SFT.GetAutoSolveGlitches()?MF_CHECKED:↙
				    ↪ MF_UNCHECKED));
			}
		}
		else if(wParam==ID_FILE_OPENSETTINGS){
4635			if(BrowseFile(hWnd,TRUE,"Open Settings","Kalle's ↙
			    ↪ fraktaler\0*.kfs\0Image files\0*.png;*.jpg;*.jpeg↙
			    ↪ ;*.tif;*.tiff;*.exr\0\0",g_szSettingsFile)){
				bool ret;
				long r = OpenSettings(hWnd, ret);
				if (ret) return r;
			}
4640		}
```

```
		else if(wParam==ID_FILE_SAVESETTINGS){
			if(BrowseFile(hWnd,FALSE,"Save Settings","Kalle's ↙
			    ↳ fraktaler\0*.kfs\0\0",g_szSettingsFile)){
				if(!g_SFT.SaveSettings(g_szSettingsFile, true))
					return MessageBox(hWnd,"Could not save ↙
					    ↳ settings","Error",MB_OK|↙
					    ↳ MB_ICONSTOP);
			}
		}
		else if(wParam==ID_FILE_OPEN_){
			if(BrowseFile(hWnd,TRUE,"Open Location Parameters","↙
			    ↳ Kalle's fraktaler\0*.kfr\0Image files\0*.png;*.jpg↙
			    ↳ ;*.jpeg;*.tif;*.tiff;*.exr\0\0",g_szFile)){
				bool ret;
				g_SFT.UndoStore();
				long r = OpenFile(hWnd, ret);
				if (ret) return r;
			}
		}
		else if(wParam==ID_FILE_OPENMAP){
			std::string szfile;
			if(BrowseFile(hWnd,TRUE,"Open Map","Kalle's fraktaler↙
			    ↳ \0*.kfb\0EXR files\0*.exr\0\0",szfile)){
				bool ret;
				long r = OpenMap(hWnd, ret, szfile);
				if (ret) return r;
			}
		}
		else if(wParam==ID_FILE_SAVE_){
			if(g_szFile == "")
				PostMessage(hWnd,WM_COMMAND,ID_FILE_SAVEAS_,0);
			else if (g_SFT.GetSaveOverwrites()){
				if(!g_SFT.SaveFile(g_szFile, true))
				  return MessageBox(hWnd,"Could not save ↙
				      ↳ parameters","Error",MB_OK|MB_ICONSTOP);
			}
			else if (!g_SFT.SaveFile(g_szFile, false)){
				SYSTEMTIME now = { 0 };
				GetSystemTime(&now);
				char date[100];
				snprintf(date, 100 - 1, "%04d-%02d-%02dT%02d-%02↙
				    ↳ d-%02d-%03d.%s", now.wYear, now.wMonth, ↙
				    ↳ now.wDay, now.wHour, now.wMinute, now.↙
				    ↳ wSecond, now.wMilliseconds, "kfr");
				if (! g_SFT.SaveFile(replace_path_extension(↙
				    ↳ g_szFile, date), false))
					return MessageBox(hWnd,"Could not save ↙
					    ↳ parameters","Error",MB_OK|↙
					    ↳ MB_ICONSTOP);
			}
		}
		else if(wParam==ID_FILE_SAVEASJPEG){
			g_JpegParams.nWidth = g_SFT.GetWidth();
			g_JpegParams.nHeight = g_SFT.GetHeight();
			g_JpegParams.nQuality = 100;
			if(DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
			    ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,0)){
```

```
                                std::string szFile;
4685                            if(g_JpegParams.nWidth>g_SFT.GetWidth()){
                                        g_bSaveJpeg=TRUE;
                                        SetTimer(hWnd,0,500,NULL);
                                        DisableUnsafeMenus(hWnd);
                                        g_SFT.RenderFractal(g_JpegParams.nWidth,↙
                                            ↳ g_JpegParams.nHeight,g_SFT.↙
                                            ↳ GetIterations(),hWnd);
4690                                    return 0;
                                }
                                if(BrowseFile(hWnd,FALSE,"Save as Jpeg","Jpeg↙
                                    ↳ \0*.jpg\0\0",szFile)){
                                        if(!g_SFT.SaveJpg(szFile,g_JpegParams.↙
                                            ↳ nQuality,g_JpegParams.nWidth,↙
                                            ↳ g_JpegParams.nHeight))
                                                MessageBox(hWnd,"File could not ↙
                                                    ↳ be saved","Error",MB_OK|↙
                                                    ↳ MB_ICONSTOP);
4695                                    szFile = replace_path_extension(szFile, ↙
                                            ↳ "kfb");
                                        if(FileExists(szFile) && MessageBox(hWnd↙
                                            ↳ ,"Found a map file (.kfb) with the↙
                                            ↳  same name, do you want to replace↙
                                            ↳  it?","Kalle's Fraktaler",MB_YESNO↙
                                            ↳ )==IDYES)
                                                g_SFT.SaveMapB(szFile);
                                        szFile = replace_path_extension(szFile, ↙
                                            ↳ "exr");
                                        if(FileExists(szFile) && MessageBox(hWnd↙
                                            ↳ ,"Found a map file (.exr) with the↙
                                            ↳  same name, do you want to replace↙
                                            ↳  it?","Kalle's Fraktaler",MB_YESNO↙
                                            ↳ )==IDYES)
4700                                            g_SFT.SaveJpg(szFile,-3,↙
                                                    ↳ g_JpegParams.nWidth,↙
                                                    ↳ g_JpegParams.nHeight);
                                }
                        }
                }
                else if(wParam==ID_FILE_SAVEASPNG){
4705                    g_JpegParams.nWidth = g_SFT.GetWidth();
                        g_JpegParams.nHeight = g_SFT.GetHeight();
                        g_JpegParams.nQuality = 100;
                        if(DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                            ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,1)){
                                std::string szFile;
4710                            if(g_JpegParams.nWidth>g_SFT.GetWidth()){
                                        g_bSavePng=TRUE;
                                        SetTimer(hWnd,0,500,NULL);
                                        DisableUnsafeMenus(hWnd);
                                        g_SFT.RenderFractal(g_JpegParams.nWidth,↙
                                            ↳ g_JpegParams.nHeight,g_SFT.↙
                                            ↳ GetIterations(),hWnd);
4715                                    return 0;
                                }
                                if(BrowseFile(hWnd,FALSE,"Save as PNG","PNG\0*.↙
                                    ↳ png\0\0",szFile)){
```

```
                                        if(!g_SFT.SaveJpg(szFile,-1,g_JpegParams.nWidth,g_JpegParams.nHeight))
                                            MessageBox(hWnd,"File could not be saved","Error",MB_OK|MB_ICONSTOP);
4720                                    szFile = replace_path_extension(szFile, "kfb");
                                        if(FileExists(szFile) && MessageBox(hWnd,"Found a map file (.kfb) with the same name, do you want to replace it?","Kalle's Fraktaler",MB_YESNO)==IDYES)
                                            g_SFT.SaveMapB(szFile);
                                        szFile = replace_path_extension(szFile, "exr");
                                        if(FileExists(szFile) && MessageBox(hWnd,"Found a map file (.exr) with the same name, do you want to replace it?","Kalle's Fraktaler",MB_YESNO)==IDYES)
4725                                        g_SFT.SaveJpg(szFile,-3,g_JpegParams.nWidth,g_JpegParams.nHeight);
                                    }
                                }
                            }
                            else if(wParam==ID_FILE_SAVEASTIF){
4730                            g_JpegParams.nWidth = g_SFT.GetWidth();
                                g_JpegParams.nHeight = g_SFT.GetHeight();
                                g_JpegParams.nQuality = 100;
                                if(DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,1)){
                                    std::string szFile;
4735                                if(g_JpegParams.nWidth>g_SFT.GetWidth()){
                                        g_bSaveTif=TRUE;
                                        SetTimer(hWnd,0,500,NULL);
                                        DisableUnsafeMenus(hWnd);
                                        g_SFT.RenderFractal(g_JpegParams.nWidth,g_JpegParams.nHeight,g_SFT.GetIterations(),hWnd);
4740                                    return 0;
                                    }
                                    if(BrowseFile(hWnd,FALSE,"Save as TIFF","TIFF\0*.tif\0\0",szFile)){
                                        if(!g_SFT.SaveJpg(szFile,-2,g_JpegParams.nWidth,g_JpegParams.nHeight))
                                            MessageBox(hWnd,"File could not be saved","Error",MB_OK|MB_ICONSTOP);
4745                                    szFile = replace_path_extension(szFile, "kfb");
                                        if(FileExists(szFile) && MessageBox(hWnd,"Found a map file (.kfb) with the same name, do you want to replace it?","Kalle's Fraktaler",MB_YESNO)==IDYES)
                                            g_SFT.SaveMapB(szFile);
```

```
                                        szFile = replace_path_extension(szFile, ↙
                                            ↳ "exr");
                                        if(FileExists(szFile) && MessageBox(hWnd↙
                                            ↳ ,"Found a map file (.exr) with the↙
                                            ↳  same name, do you want to replace↙
                                            ↳  it?","Kalle's Fraktaler",MB_YESNO↙
                                            ↳ )==IDYES)
4750                                        g_SFT.SaveJpg(szFile,-3,↙
                                                ↳ g_JpegParams.nWidth,↙
                                                ↳ g_JpegParams.nHeight);
                                    }
                                }
                            }
                            else if(wParam==ID_FILE_SAVEASEXR){
4755                            g_JpegParams.nWidth = g_SFT.GetWidth();
                                g_JpegParams.nHeight = g_SFT.GetHeight();
                                g_JpegParams.nQuality = 100;
                                if(DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                                    ↳ IDD_DIALOG7),hWnd,(DLGPROC)JpegProc,1)){
                                    std::string szFile;
4760                                if(g_JpegParams.nWidth>g_SFT.GetWidth()){
                                        g_bSaveExr=TRUE;
                                        SetTimer(hWnd,0,500,NULL);
                                        DisableUnsafeMenus(hWnd);
                                        g_SFT.RenderFractal(g_JpegParams.nWidth,↙
                                            ↳ g_JpegParams.nHeight,g_SFT.↙
                                            ↳ GetIterations(),hWnd);
4765                                    return 0;
                                    }
                                    if(BrowseFile(hWnd,FALSE,"Save as EXR","EXR\0*.↙
                                        ↳ exr\0\0",szFile)){
                                        if(!g_SFT.SaveJpg(szFile,-3,g_JpegParams↙
                                            ↳ .nWidth,g_JpegParams.nHeight))
                                            MessageBox(hWnd,"File could not ↙
                                                ↳ be saved","Error",MB_OK|↙
                                                ↳ MB_ICONSTOP);
4770                                    }
                                }
                            }
                            else if(wParam==ID_FILE_SAVEAS_){
                                if(BrowseFile(hWnd,FALSE,"Save Location Parameters","↙
                                    ↳ Kalle's fraktaler\0*.kfr\0\0",g_szFile)){
4775                                if(!g_SFT.SaveFile(g_szFile, true))
                                        return MessageBox(hWnd,"Could not save ↙
                                            ↳ parameters","Error",MB_OK|↙
                                            ↳ MB_ICONSTOP);
                                    char szTitle[1024];
                                    wsprintf(szTitle,"Kalle's Fraktaler 2 - %s",↙
                                        ↳ g_szFile.c_str());
                                    SetWindowText(hWnd,szTitle);
4780                            }
                            }
/*                          else if(wParam==ID_ACTIONS_CREATEZOOMSEQUENCE){
                                DialogBoxParam(GetModuleHandle(NULL),MAKEINTRESOURCE(↙
                                    ↳ IDD_DIALOG6),hWnd,(DLGPROC)ZoomProc,0);
                            }
4785 */                      else if(wParam==ID_ACTIONS_RESET){
```

```
                        g_szFile="";
                        SetWindowText(hWnd,"Kalle's Fraktaler 2");
                        g_SFT.UndoStore();
                        g_SFT.SetPosition("0","0","1");
4790                    PostMessage(hWnd,WM_KEYDOWN,VK_F5,0);
                }
                else if(wParam==ID_SPECIAL_NONEXACTFINDMINIBROT){
                        if(!g_bFindMinibrotCount){
#if 0
4795                            char *e = g_SFT.GetZoom();
                                e = stristr(e,"e");
                                if(e)
                                        e++;
                                else e = "1";
4800 #endif
                                g_bFindMinibrotPos=1;// + 2*log(atof(e))/log((↙
                                    ↳ double)2)/3;
                                g_bFindMinibrotCount=1;
                        }
                        else
4805                            g_bFindMinibrotCount=0;
                        CheckMenuItem(GetMenu(hWnd),↙
                            ↳ ID_SPECIAL_NONEXACTFINDMINIBROT,MF_BYCOMMAND|(↙
                            ↳ g_bFindMinibrotCount?MF_CHECKED:MF_UNCHECKED));
                }
                else if(wParam==ID_ACTIONS_FINDMINIBROT){
                        if(!lParam && HIWORD(GetKeyState(VK_CONTROL)))
4810                            g_nStopAtExponent = DialogBox(GetModuleHandle(↙
                                    ↳ NULL),MAKEINTRESOURCE(IDD_DIALOG9),hWnd,(↙
                                    ↳ DLGPROC)StopAtProc);
                        else if(!lParam)
                                g_nStopAtExponent=0;
                        if(g_nStopAtExponent && g_nStopAtExponent<=g_SFT.↙
                            ↳ GetExponent())
                                return MessageBox(hWnd,"Done","Error",MB_OK|↙
                                    ↳ MB_ICONINFORMATION);
4815                    g_SFT.Stop();
                        g_SFT.UndoStore();
                        g_bAnim=false;
                        g_bStoreZoom=FALSE;
                        DeleteObject(g_bmSaveZoomBuff);
4820                    g_bmSaveZoomBuff=NULL;
                        int x, y;
                        if(g_bFindMinibrotCount && g_bFindMinibrotCount==↙
                            ↳ g_bFindMinibrotPos){
                                int64_t nMin,nMax;
                                g_SFT.GetIterations(nMin,nMax);
4825                            if(nMax<nMin+3){
                                        SYSTEMTIME st;
                                        GetLocalTime(&st);
                                        SystemTimeToFileTime(&st,(LPFILETIME)&↙
                                            ↳ g_nTStart);
                                        g_SFT.Zoom(0,0,1.0/g_SFT.GetZoomSize(),↙
                                            ↳ g_SFT.GetWidth(),g_SFT.GetHeight()↙
                                            ↳ );
4830                                    SetTimer(hWnd,0,500,NULL);
                                        return 0;
```

```
                }
                int64_t nIter;
                g_SFT.GetIterations(nMin,nMax,NULL,NULL,TRUE);
4835            int diff = (nMax-nMin)/6;
                int nTries=0;
                do{
                        x = rand()%g_SFT.GetWidth();
                        y = rand()%g_SFT.GetHeight();
4840                    nIter = g_SFT.GetIterationOnPoint(x,y);
                        if(nTries++>20 && diff>10){
                                nTries=0;
                                diff--;
                        }
4845            }while(nIter<nMin + diff || nIter==g_SFT.↙
                    ↳ GetIterations());
        }
        else if(!g_SFT.Center(x,y,g_bFindMinibrotCount && ↙
            ↳ g_bFindMinibrotCount==g_bFindMinibrotPos)){
                g_bFindMinibrot=FALSE;
                return MessageBox(hWnd,"Cannot find center","↙
                    ↳ Error",MB_OK|MB_ICONSTOP);
4850    }
        if(g_bFindMinibrotCount){
                int64_t nMin,nMax;
                g_SFT.GetIterations(nMin,nMax);
                int64_t nIter=g_SFT.GetIterationOnPoint(x,y);
4855            if(nIter<nMin+8)
                        g_SFT.HighestIteration(x,y);
        }
        if(g_bFindMinibrotCount){
                if(g_bFindMinibrotPos==g_bFindMinibrotCount){
4860                    g_bFindMinibrotCount=0;
                        g_bFindMinibrotPos=1;//+=(↙
                            ↳ g_bFindMinibrotPos==1?1:↙
                            ↳ g_bFindMinibrotPos/2);
                }
                g_bFindMinibrotCount++;
                char szTitle[1024];
4865            wsprintf(szTitle,"Kalle's Fraktaler 2 - %d",↙
                    ↳ g_bFindMinibrotPos-g_bFindMinibrotCount);
                SetWindowText(hWnd,szTitle);
        }
        else
                SetWindowText(hWnd,"Kalle's Fraktaler 2");
4870    int64_t nMin, nMax, nIter;
        g_SFT.GetIterations(nMin,nMax);
        nIter = g_SFT.GetIterations();
        if(nIter<nMin+nMin/2+2000)
        {
4875            // ignore auto iterations setting, otherwise it ↙
                    ↳ could stop early
                g_SFT.SetIterations(nMin+nMin/2+3000);
        }
        if(nIter==nMax && !g_bFindMinibrotCount){
                g_bFindMinibrot=FALSE;
4880            MessageBox(hWnd,"Done","Kalle's Fraktaler",MB_OK↙
                    ↳ );
```

```
                                }
                                RECT rc;
                                GetClientRect(hWnd,&rc);
                                RECT sr;
4885                            GetWindowRect(g_hwStatus,&sr);
                                sr.bottom-=sr.top;
                                rc.bottom-=sr.bottom;
                                x = x*rc.right/g_SFT.GetWidth();
                                y = y*rc.bottom/g_SFT.GetHeight();
4890                            while(g_bAnim){
                                        Sleep(10);
                                }
                                SendMessage(hWnd,WM_LBUTTONDOWN,0,MAKELONG(x,y));
                                PostMessage(hWnd,WM_LBUTTONUP,0,MAKELONG(x,y));
4895                            g_bFindMinibrot=TRUE;
                        }
                        else if(wParam==ID_MENU_ABOUT){
                                char szMsg[4096];
                                SYSTEM_INFO sysinfo;
4900                            GetSystemInfo( &sysinfo );  //©
                                snprintf(szMsg, 4000,
                                        "version %s\n"
                                        "©2013-2017 Karl Runmo\n"
                                        "©2017-2019 Claude Heiland-Allen\n"
4905                                    "License: GNU AGPL v3+\n\n"
                                        "Processors: %d\n"
                                        "Compiled for %s\n"
                                        "Precision: %d bits (%d decimal digits)\n"
                                        "\nLibraries:\n"
4910                                    "- JPEG 6b2 <https://jpegclub.org/support>\n"
                                        "- TIFF 4.0.10 <https://www.simplesystems.org/↙
                                            ↳ libtiff/>\n"
                                        "- PNG %s <https://libpng.org>\n"
                                        "- ZLIB %s <https://zlib.net>\n"
                                        "- GMP %d.%d.%d <https://gmplib.org>\n"
4915                                    "- MPFR %s <https://mpfr.org>\n"
                                        "- GSL %s <https://www.gnu.org/software/gsl>\n"
                                        "- PIXMAN %s <https://pixman.org>\n"
                                        "- MINGW-STD-THREADS <https://github.com/meganz/↙
                                            ↳ mingw-std-threads>\n"
                                        "- ILMBASE %s <https://openexr.com>\n"
4920                                    "- OPENEXR %s <https://openexr.com>\n"
                                        "- GLM %d.%d.%d.%d <https://glm.g-truc.net>\n"
                                        "- BOOST %d.%d.%d <https://boost.org>\n"
#ifdef KF_OPENCL
                                        "- CLEW git.50751dd <https://github.com/↙
                                            ↳ martijnberger/clew>\n"
4925 #endif
#ifdef __GNUC__
                "\nCompiler:\n"
                "- MINGW/G++ %d.%d.%d <https://gcc.gnu.org/>\n"
#endif
4930                                    "\nThanks to:\n"
                                        " - K.I.Martin for applying Perturbation and ↙
                                            ↳ Series Approximation on the Mandelbrot set↙
                                            ↳  and sharing theory and source code!\n"
                                        " - Pauldelbrot for reliable glitch detection ↙
```

```
                                        ↳ method\n"
                                    " - Botond Kósa and knighty for extensions of ↙
                                        ↳ Series Approximation\n"
                                    " - laser blaster for Burning ship formula\n"
4935                                " - stardust4ever for other fractal types\n"
                                    " - claude for Newton-Raphson method\n"
                                    " - gerrit for differencing variations\n"
                                    " - Dinkydau, Fractal universe, CFJH, Foxxie and↙
                                        ↳  others for bug reports\n"
                                    " - Chillheimer for hosting <http://www.↙
                                        ↳ chillheimer.de/kallesfraktaler>\n"
4940                                " - Karl for releasing the source code under a ↙
                                        ↳ Free Software license\n\n"
                                    "Homepage: <https://mathr.co.uk/kf/kf.html>",
                                    version.c_str(),
                                    int(sysinfo.dwNumberOfProcessors),
                                    sizeof(void*)==4?"32-bit":"64-bit",
4945                                int(MPFR_PREC_MAX),
                                    int(MPFR_PREC_MAX * log10(2.0)),
                                    png_libpng_ver,
                                    zlib_version,
                                    __GNU_MP_VERSION, __GNU_MP_VERSION_MINOR, ↙
                                        ↳ __GNU_MP_VERSION_PATCHLEVEL,
4950                                MPFR_VERSION_STRING,
                                    GSL_VERSION,
                                    PIXMAN_VERSION_STRING,
                                    ILMBASE_VERSION_STRING,
                                    OPENEXR_VERSION_STRING,
4955                                GLM_VERSION_MAJOR, GLM_VERSION_MINOR, ↙
                                        ↳ GLM_VERSION_PATCH, GLM_VERSION_REVISION,
                                    BOOST_VERSION / 100000, BOOST_VERSION / 100 % ↙
                                        ↳ 1000, BOOST_VERSION % 100
#ifdef __GNUC__
                                    , __GNUC__, __GNUC_MINOR__, __GNUC_PATCHLEVEL__
#endif
4960                                );
                        szMsg[4096-1] = 0;
                        return MessageBox(hWnd,szMsg,"Kalle's Fraktaler 2",MB_OK↙
                            ↳ );
                }
                else if (wParam == ID_FILE_CHECKFORUPDATE)
4965            {
                        std::string latest = CheckForUpdate();
                        std::ostringstream os;
                        os << "Latest version: " << latest << "\n";
                        os << "Current version: " << version << "\n";
4970                    if (latest == "unknown")
                          os << "Couldn't retrieve information!\n";
                        else if (latest == version)
                          os << "You are up to date!\n";
                        else
4975                      os << "There is a newer version!\n";
            return MessageBox(hWnd, os.str().c_str(), "Update Checker", MB_OK);
                }
        }
        return DefWindowProc(hWnd,uMsg,wParam,lParam);
4980 }
```

```
#if 0
static int Test()
{
4985            CFileFloat <double>tmp(3000);
                int i;
                for(i=0;i<3000;i++)
                        tmp[i] = i;
                for(i=0;i<3000;i++)
4990                    if(tmp[i]!=i)
                                break;
                if(i<3000)
                        return MessageBox(NULL,"Error","Error",MB_OK|MB_ICONSTOP);
                else
4995                    return MessageBox(NULL,"Success","Success",MB_OK);
                return 0;
}

static int Test2()
5000 {
                return 0;
}

static int Test1()
5005 {
                CFixedFloat xr = 0, xi = 0, xin, xrn, sr = 0, si = 0, xrxid = 0;
                CFixedFloat m_rref = 0.25, m_iref=0;
                int antal;
                SYSTEMTIME st;
5010            __int64 t1, t2;
                MessageBox(NULL,"Start","Debug",MB_OK);
                GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&t1);
                for(antal=0;antal<10000;antal++){
5015                    xrn = sr - si + m_rref;
                        xrxid = 2 * xr*xi;
//                      xrxid = CFixedFloat_Multiply((CFixedFloat)2,xr);
//                      xrxid = CFixedFloat_Multiply(xrxid,xi);
                        xin = xrxid.Abs() + m_iref;
5020                    xr = xrn;
                        xi = xin;
                        sr = xr.Square();
                        si = xi.Square();

5025            }
                GetLocalTime(&st);
                SystemTimeToFileTime(&st,(LPFILETIME)&t2);
                t2-=t1;
                FileTimeToSystemTime((LPFILETIME)&t2,&st);
5030            __int64 td = 10000000;
                char szRes[1024];
                wsprintf(szRes,"%02d:%02d:%02d.%07d",t2/(td*3600),(t2/(td*60))%60,(t2/td↙
                    ↳ )%60,t2%td);
//              wsprintf(szRes+strlen(szRes),"\n%02d:%02d:%02d.%03d",st.wHour,st.wMinute↙
    ↳ ,st.wSecond,st.wMilliseconds);
                return MessageBox(NULL,szRes,"Res",MB_OK);
5035 }
```

```
#endif

// detailed progress reporting thread for command line rendering
static volatile bool ThReportProgress_running = true;
DWORD ThReportProgress(LPVOID arg)
{
(void) arg;
	while (ThReportProgress_running)
	{
		Sleep(1000);
		int nG, nR, nA, nT;
		int nP = g_SFT.GetProgress(&nG,&nR,&nA,&nT);
		std::ostringstream status;
		status << " P " << std::setw(3) << nP << "%  G " << std::setw(3)↙
		    ↳ << nG << "%  R " << std::setw(3) << nR << "%  A " << std↙
		    ↳ ::setw(3) << nA << "%  T " << std::setw(8) << nT << "\r";
		std::cerr << status.str();
	}
	return 0;
}

static bool save_frame(int frame)
{
	output_log_message(Info, "colouring final image");
	g_SFT.m_bInhibitColouring = FALSE;
	g_SFT.ApplyColors();
	//  save the result
	bool ok = true;
	if (g_args->bSaveEXR)
	{
		char fn[1000];
		snprintf(fn, 1000, g_args->sSaveEXR.c_str(), frame);
		output_log_message(Info, "saving EXR " << fn);
		if (! g_SFT.SaveJpg(fn, -3))
		{
			ok = false;
			output_log_message(Error, "saving EXR " << fn << " ↙
			    ↳ FAILED");
		}
	}
	if (g_args->bSaveTIF)
	{
		char fn[1000];
		snprintf(fn, 1000, g_args->sSaveTIF.c_str(), frame);
		output_log_message(Info, "saving TIFF " << fn);
		if (! g_SFT.SaveJpg(fn, -2))
		{
			ok = false;
			output_log_message(Error, "saving TIFF " << fn << " ↙
			    ↳ FAILED");
		}
	}
	if (g_args->bSavePNG)
	{
		char fn[1000];
		snprintf(fn, 1000, g_args->sSavePNG.c_str(), frame);
		output_log_message(Info, "saving PNG " << fn);
```

```
                if (! g_SFT.SaveJpg(fn, -1))
5090            {
                        ok = false;
                        output_log_message(Error, "saving PNG " << fn << " ↙
                            ↳ FAILED");
                }
        }
5095    if (g_args->bSaveJPG)
        {
                char fn[1000];
                snprintf(fn, 1000, g_args->sSaveJPG.c_str(), frame);
                output_log_message(Info, "saving JPG " << fn);
5100            if (! g_SFT.SaveJpg(fn, 100))
                {
                        ok = false;
                        output_log_message(Error, "saving JPG " << fn << " ↙
                            ↳ FAILED");
                }
5105    }
        if (g_args->bSaveMap)
        {
                char fn[1000];
                snprintf(fn, 1000, g_args->sSaveMap.c_str(), frame);
5110            output_log_message(Info, "saving KFB " << fn);
                g_SFT.SaveMapB(fn);
        }
        return ok;
}
5115
static bool render_frame(int frame)
{
        output_log_message(Info, "reference " << 1);
        g_SFT.m_bInhibitColouring = TRUE;
5120    g_SFT.m_bInteractive = false;
        if (frame == 0)
        {
                g_SFT.RenderFractal(g_SFT.GetImageWidth(), g_SFT.GetImageHeight↙
                    ↳ (), g_SFT.GetIterations(), nullptr, true, true);
        }
5125    else
        {
                int j = g_SFT.GetJitterSeed();
                if (j)
                {
5130                    g_SFT.SetJitterSeed(j + 1);
                }
                AutoIterations();
                g_SFT.Zoom(g_SFT.GetWidth()/2, g_SFT.GetHeight()/2, 1/g_SFT.↙
                    ↳ GetZoomSize(), g_SFT.GetWidth(), g_SFT.GetHeight(), FALSE,↙
                    ↳  false);
                g_SFT.RenderFractal(g_SFT.GetImageWidth(), g_SFT.GetImageHeight↙
                    ↳ (), g_SFT.GetIterations(), nullptr, true, true);
5135    }
        for (int r = 2; r < g_SFT.GetMaxReferences(); ++r)
        {
                int x = -1, y = -1;
                int n = g_SFT.FindCenterOfGlitch(x, y);
```

```
5140                    if (! n)
                        {
                                output_log_message(Info, "no more glitches");
                                break;
                        }
5145                    output_log_message(Info, "reference " << r << " at (" << x << ↙
                            ↳ "," << y << ") size " << (n - 1) << " ");
                        g_SFT.AddReference(x, y);
                }
                ThReportProgress_running = false;
                return save_frame(frame);
5150    }

        extern int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE,LPSTR commandline,int)
        {
        //      return Test();
5155
                CommandLineArguments args(commandline);
                if (args.bError)
                {
                        std::cerr << "ERROR: bad command line arguments" << std::endl;
5160            }
                if (args.bVersion)
                {
                        std::cout << version << std::endl;
                }
5165            if (args.bHelp)
                {
                        std::cout << usage;
                }
                if (args.bVersion || args.bHelp || args.bError)
5170            {
                        return 0;
                }
                g_args = &args;

5175    #ifdef KF_OPENCL
                cldevices = initialize_opencl();
        #endif


5180            bool interactive = !(g_args->bSaveJPG || g_args->bSaveTIF || g_args->↙
                    ↳ bSavePNG || g_args->bSaveEXR || g_args->bSaveMap);
                if (interactive)
                {
                        GetModuleFileName(GetModuleHandle(NULL),g_szRecovery,sizeof(↙
                            ↳ g_szRecovery));
                        strcpy(strrchr(g_szRecovery,'.'),".rec");
5185
                        WNDCLASS wc={0};
                        wc.hInstance = hInstance;
                        wc.lpszClassName = "FRAKTAL_SFT";
                        wc.lpfnWndProc = (WNDPROC)MainProc;
5190                    wc.hCursor = LoadCursor(NULL,IDC_CROSS);
                        wc.hIcon = LoadIcon(hInstance,MAKEINTRESOURCE(IDI_ICON1));
                        g_hIcon = wc.hIcon;
                        RegisterClass(&wc);
```

```
		HWND hWnd = CreateWindowEx(WS_EX_CLIENTEDGE, wc.lpszClassName,"Kalle's Fraktaler 2",WS_OVERLAPPEDWINDOW|WS_VISIBLE,0,0,640,360,NULL,LoadMenu(hInstance,MAKEINTRESOURCE(IDR_MENU1)),hInstance,0);
		g_SFT.SetWindow(hWnd);
		ShowWindow(hWnd,SW_SHOW);

		MSG msg;
		while(GetMessage(&msg,NULL,0,0)){
			if(GetDlgCtrlID(msg.hwnd)==IDC_LIST1 && msg.message==WM_RBUTTONDOWN)
				SendMessage(GetParent(msg.hwnd),WM_USER+88,GetDlgCtrlID(msg.hwnd),0);
			if(g_hwColors && IsDialogMessage(g_hwColors,&msg))
				continue;
			if(g_hwExamine && IsDialogMessage(g_hwExamine,&msg))
				continue;
			if(g_hwNewton && IsDialogMessage(g_hwNewton,&msg))
				continue;
			TranslateMessage(&msg);
			DispatchMessage(&msg);
		}
		DeleteFile(g_szRecovery);
	}
	else
	{
		// prepare
		output_log_message(Info, "kf " << version << " (c) 2013-2017 Karl Runmo, (c) 2017-2019 Claude Heiland-Allen");
		if (g_args->bLoadSettings)
		{
			bool ret;
			g_szSettingsFile = g_args->sLoadSettings;
			output_log_message(Info, "loading settings " << g_szSettingsFile);
			OpenSettings(nullptr, ret);
			if (ret)
			{
				output_log_message(Error, "loading settings " << g_szSettingsFile << " FAILED");
				return 1;
			}
		}
		if (g_args->bLoadLocation)
		{
			bool ret;
			g_szFile = g_args->sLoadLocation;
			output_log_message(Info, "loading location " << g_szFile);
			OpenFile(nullptr, ret);
			if (ret)
			{
				output_log_message(Error, "loading location " << g_szFile << " FAILED");
				return 1;
			}
		}
```

```
                if (g_args->bLoadMap)
                {
                        bool ret;
                        g_szFile = g_args->sLoadMap;
5245                    output_log_message(Info, "loading map " << g_szFile);
                        ret = ! g_SFT.OpenMapB(g_szFile);
                        if (ret)
                        {
                                ret = ! g_SFT.OpenMapEXR(g_szFile);
5250                            if (ret)
                                {
                                        output_log_message(Error, "loading map "↙
                                            ↳ << g_szFile << " FAILED");
                                        return 1;
                                }
5255              }
                }
                if (g_args->bLoadPalette)
                {
                        bool ret;
5260                    g_szFile = g_args->sLoadPalette;
                        output_log_message(Info, "loading palette " << g_szFile)↙
                            ↳ ;
                        ret = ! g_SFT.OpenFile(g_szFile, TRUE);
                        if (ret)
                        {
5265                            output_log_message(Error, "loading palette " << ↙
                                    ↳ g_szFile << " FAILED");
                                return 1;
                        }
                }
        if (g_args->bLoadMap)
5270    {
                        save_frame(0);
                }
                else
            {
5275                    output_log_message(Info, "rendering at " << g_SFT.↙
                            ↳ GetImageWidth() << "x" << g_SFT.GetImageHeight());
                        // render the image (add reference calls render fractal↙
                            ↳ ...)
                        if (LogLevel_Status >= g_log_level)
                        {
                                HANDLE hProgress = CreateThread(0,0,(↙
                                    ↳ LPTHREAD_START_ROUTINE)ThReportProgress↙
                                    ↳ ,0,0,0);
5280                            CloseHandle(hProgress);
                        }
                        bool ok = true;
                        if (g_args->bZoomOut)
                        {
5285                            for (int frame = 0; g_args->nZoomOut < 0 || ↙
                                    ↳ frame < g_args->nZoomOut; ++frame)
                                {
                                        output_log_message(Info, "frame " << ↙
                                            ↳ frame << " of " << g_args->↙
                                            ↳ nZoomOut);
```

```
                                        ok = render_frame(frame);
                                        if (! ok)
5290                                    {
                                                break;
                                        }
                                        if (atof(g_SFT.ToZoom().c_str()) < 1e-3)
                                        {
5295                                            break;
                                        }
                                }
                        }
                        else
5300                    {
                                ok = render_frame(0);
                        }
                        if (! ok)
                        {
5305                            output_log_message(Error, "FAILED");
                        }
                        else
                        {
                                output_log_message(Info, "all done, exiting");
5310                    }
                        return ok ? 0 : 1;
                }
        }

5315    return 0;
}
```

82 fraktal_sft/main_examine.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15
You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20  #include "main.h"
    #include "main_examine.h"
    #include "fraktal_sft.h"
    #include "resource.h"
    #include "../common/tooltip.h"
25  #include "../common/StringVector.h"
```

```
#include "../common/FolderBrowser.h"

static int g_nPrevAutoGlitchNP;
static int g_bAutoSolveGlitch=0;
static int g_nAutoSolveGlitchLimit=10;

static std::string g_szExamine;
static std::vector<std::string> g_stExamine;
static int g_nExamine=-1;
static int g_nExamineZoom=-1;

HWND g_hwExamine=NULL;

bool g_bExamineDirty=false;

static int kGetDlgItemInt(HWND hWnd,int nID)
{
	char szText[256];
	GetDlgItemText(hWnd,nID,szText,sizeof(szText));
	return atoi(szText);
}

static void kSetDlgItemInt(HWND hWnd,int nID,int val)
{
	char szText[256];
	sprintf(szText,"%d",val);
	SetDlgItemText(hWnd,nID,szText);
}

extern bool Examine(HWND hWnd)
{
	g_szExamine = "";
	if(!BrowseFile(hWnd,TRUE,"Open location","Kalle's fraktaler\0*.kfr\0\0",↙
	    ↳ g_szExamine))
		return false;
	if(g_hwExamine)
		SetFocus(g_hwExamine);
	else{
		g_hwExamine = CreateDialog(GetModuleHandle(NULL),MAKEINTRESOURCE↙
		    ↳ (IDD_DIALOG8),hWnd,(DLGPROC)ExamineProc);
		ShowWindow(g_hwExamine,SW_SHOW);
	}
	return true;
}

extern int WINAPI ExamineProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
	(void) lParam;
	if(uMsg==WM_INITDIALOG){
		SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
		SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
		InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,4);
		g_nPrevAutoGlitchNP=g_SFT.GetSolveGlitchNear();
		g_SFT.SetSolveGlitchNear(false);
		CheckMenuItem(GetMenu(GetParent(hWnd)),↙
		    ↳ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD,↙
		    ↳ MF_BYCOMMAND|(g_SFT.GetSolveGlitchNear()?MF_CHECKED:↙
```

```
                        ↪ MF_UNCHECKED));
                g_bAddReference=TRUE;
80              g_bEraser=FALSE;
                g_stExamine.resize(0);
                SendDlgItemMessage(hWnd,IDC_RADIO1,BM_SETCHECK,1,0);
                WIN32_FIND_DATA fd;
                std::string szExamine = replace_path_filename(g_szExamine, "*_*.↙
                    ↪ kfb");
85              HANDLE hFind = FindFirstFile(szExamine.c_str(),&fd);
                if(hFind==INVALID_HANDLE_VALUE){
                        if(hFind)
                                FindClose(hFind);
                        MessageBox(hWnd,"Could not browse kfb files","Error",↙
                            ↪ MB_OK|MB_ICONSTOP);
90                      DestroyWindow(hWnd);
                        g_hwExamine=NULL;
                        return 0;
                }
                do{
95                      szExamine = replace_path_filename(szExamine, fd.↙
                            ↪ cFileName);
                        g_stExamine.push_back(szExamine);
                }while(FindNextFile(hFind,&fd));
                FindClose(hFind);
                std::sort(g_stExamine.begin(), g_stExamine.end());
100             std::reverse(g_stExamine.begin(), g_stExamine.end());
                g_SFT.OpenFile(g_szExamine);
                g_SFT.OpenMapB(g_stExamine[0]);
                g_SFT.ApplyColors();
                g_nExamine=0;
105             SetDlgItemInt(hWnd,IDC_EDIT1,g_nExamine,FALSE);
                g_bExamineDirty=FALSE;
                int last = g_stExamine.size() - 1;
                CDecNumber A(get_filename_zoom_string(g_stExamine[last]));
                CDecNumber B(get_filename_zoom_string(g_stExamine[last - 1]));
110             g_nExamineZoom = (A/B+CDecNumber(0.5)).ToInt();
                A = CDecNumber(g_SFT.GetZoom())/(CDecNumber(g_nExamineZoom)^(↙
                    ↪ g_stExamine.size()-g_nExamine-1));
                std::string szRe = g_SFT.GetRe();
                std::string szIm = g_SFT.GetIm();
                g_SFT.SetPosition(szRe,szIm,A.ToText());
115
                PostMessage(GetParent(hWnd),WM_USER+199,0,0);
                SetTimer(hWnd,0,100,NULL);
        }
        else if(uMsg==WM_COMMAND){
120             if(wParam==IDCANCEL){
                        DestroyWindow(hWnd);
                        g_hwExamine=NULL;
                        g_SFT.SetSolveGlitchNear(g_nPrevAutoGlitchNP);
                        CheckMenuItem(GetMenu(GetParent(hWnd)),↙
                            ↪ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD↙
                            ↪ ,MF_BYCOMMAND|(g_SFT.GetSolveGlitchNear()?↙
                            ↪ MF_CHECKED:MF_UNCHECKED));
125             }
                else if(wParam==IDOK || wParam==IDC_BUTTON1 || wParam==↙
                    ↪ IDC_BUTTON2 || wParam==IDC_BUTTON5){
```

```
                        SetDlgItemText(hWnd,IDC_EDIT3,"Reading...");
                        g_SFT.Stop();
                        g_bAnim=FALSE;
130                     KillTimer(hWnd,1);
                        if(wParam!=IDC_BUTTON2 && g_bExamineDirty){
                                g_SFT.SaveMapB(g_stExamine[g_nExamine]);
                                std::string szFile = replace_path_extension(↙
                                    ↳ g_stExamine[g_nExamine], "jpg");
                                if(FileExists(szFile))
135                                     g_SFT.SaveJpg(szFile,100);
                        }
                        if(wParam==IDC_BUTTON5)
                                g_bExamineDirty=TRUE;
                        else
140                             g_bExamineDirty=FALSE;
                        if(wParam==IDOK){
                                g_nExamine++;
                                if(g_nExamine>=ssize_t(g_stExamine.size()))
                                        g_nExamine=0;
145                     }
                        else if(wParam==IDC_BUTTON1 || wParam==IDC_BUTTON5){
                                g_nExamine--;
                                if(g_nExamine<0){
                                        wParam=IDC_BUTTON1;
150                                     g_nExamine=g_stExamine.size()-1;
                                        g_bExamineDirty=FALSE;
                                }
                        }
                        SetDlgItemInt(hWnd,IDC_EDIT1,g_nExamine,FALSE);
155                     UpdateWindow(GetDlgItem(hWnd,IDC_EDIT1));
                        g_SFT.OpenMapB(g_stExamine[g_nExamine],wParam==↙
                            ↳ IDC_BUTTON5,(double)1/g_nExamineZoom);
                        g_SFT.ApplyColors();
                        if(wParam!=IDC_BUTTON2)
                                SetTimer(hWnd,1,500,NULL);
160                     InvalidateRect(GetParent(hWnd),NULL,FALSE);
                        UpdateWindow(GetParent(hWnd));
//                      SetCapture(hWnd);
                        g_bWaitRead=true;
                }
165             else if(wParam==IDC_BUTTON3){
                        SetDlgItemText(hWnd,IDC_EDIT3,"Reading...");
                        g_SFT.Stop();
                        g_bAnim=FALSE;
                        KillTimer(hWnd,1);
170                     g_nExamine = GetDlgItemInt(hWnd,IDC_EDIT1,NULL,FALSE);
                        if(g_nExamine<0)
                                g_nExamine=0;
                        else if(g_nExamine>=ssize_t(g_stExamine.size()))
                                g_nExamine = g_stExamine.size()-1;
175                     g_SFT.OpenMapB(g_stExamine[g_nExamine],wParam==↙
                            ↳ IDC_BUTTON5,(double)1/g_nExamineZoom);
                        g_SFT.ApplyColors();
                        SetTimer(hWnd,1,500,NULL);
                        InvalidateRect(GetParent(hWnd),NULL,FALSE);
                        UpdateWindow(GetParent(hWnd));
180                     g_bWaitRead=true;
```

```
                }
                else if(wParam==IDC_BUTTON4){
                        if(!g_SFT.GetAutoSolveGlitches())
                        {
185                             g_bAutoSolveGlitch = 1;
                                SetTimer(hWnd,2,10,NULL);
                        }
                        g_SFT.SetAutoSolveGlitches(! g_SFT.GetAutoSolveGlitches↙
                            ↳ ());
                        if(g_SFT.GetAutoSolveGlitches())
190                     {
                                // update additional references limit from GUI
                                g_nAutoSolveGlitchLimit = kGetDlgItemInt(hWnd, ↙
                                    ↳ IDC_EDITMAXREFS);
                                if (g_nAutoSolveGlitchLimit < 1) ↙
                                    ↳ g_nAutoSolveGlitchLimit = 10;
                                kSetDlgItemInt(hWnd, IDC_EDITMAXREFS, ↙
                                    ↳ g_nAutoSolveGlitchLimit);
195                             SetDlgItemText(hWnd,IDC_BUTTON4,"Stop Auto solve↙
                                    ↳  glitch");
                        }
                        else{
                                SetDlgItemText(hWnd,IDC_EDIT4,"");
                                SetDlgItemText(hWnd,IDC_BUTTON4,"Auto solve ↙
                                    ↳ glitch");
200                     }
                }
                else if(wParam==IDC_RADIO1){
                        g_bAddReference=TRUE;
                        g_bAddMainReference=false;
205                     g_bEraser=FALSE;
                        EnableWindow(GetDlgItem(hWnd,IDC_CHECK1),TRUE);
                }
                else if(wParam==IDC_RADIO2){
                        g_bAddReference=FALSE;
210                     g_bAddMainReference=true;
                        g_bEraser=FALSE;
                        EnableWindow(GetDlgItem(hWnd,IDC_CHECK1),FALSE);
                }
                else if(wParam==IDC_RADIO3){
215                     g_bEraser=TRUE;
                        g_bAddReference=FALSE;
                        g_bAddMainReference=false;
                        EnableWindow(GetDlgItem(hWnd,IDC_CHECK1),FALSE);
                }
220             else if(wParam==IDC_CHECK1){
                        g_SFT.SetSolveGlitchNear(! g_SFT.GetSolveGlitchNear());
                        CheckMenuItem(GetMenu(GetParent(hWnd)),↙
                            ↳ ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD↙
                            ↳ ,MF_BYCOMMAND|(g_SFT.GetSolveGlitchNear()?↙
                            ↳ MF_CHECKED:MF_UNCHECKED));
                }
        }
225     else if(uMsg==WM_TIMER && wParam==0)
                SetDlgItemText(hWnd,IDC_EDIT2,g_bExamineDirty?"Changed":"");
        else if(uMsg==WM_TIMER && wParam==1){
                g_bWaitRead=false;
```

```
                    SetDlgItemText(hWnd,IDC_EDIT3,"");
230                 KillTimer(hWnd,1);
                    ReleaseCapture();
                    int nMI = g_SFT.GetIterations();
                    g_SFT.OpenFile(g_szExamine);
                    g_SFT.SetIterations(nMI);
235                 CDecNumber A = CDecNumber(g_SFT.GetZoom())/(CDecNumber(↙
                        ↳ g_nExamineZoom)^(g_stExamine.size()-g_nExamine-1));
                    std::string szRe = g_SFT.GetRe();
                    std::string szIm = g_SFT.GetIm();
                    g_SFT.SetPosition(szRe,szIm,A.ToText());

240                 PostMessage(GetParent(hWnd),WM_USER+199,0,0);
            }
            else if(uMsg==WM_TIMER && wParam==2){
                    KillTimer(hWnd,2);
                    if(!g_bAutoSolveGlitch)
245                         return 0;
    SetDlgItemText(hWnd,IDC_EDIT4,"AutoSolveGlitch");
    UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
                    int rx, ry;
    SetDlgItemText(hWnd,IDC_EDIT4,"Search for glitch");
250 UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
                    while(!g_SFT.FindCenterOfGlitch(rx,ry) || g_bAutoSolveGlitch>=↙
                        ↳ g_nAutoSolveGlitchLimit){
                            if(g_nExamine==0){
                                    g_bAutoSolveGlitch=0;
                                    SetDlgItemText(hWnd,IDC_BUTTON4,"Auto solve ↙
                                        ↳ glitch");
255 SetDlgItemText(hWnd,IDC_EDIT4,"Done");
    UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
                                    return 0;
                            }
                            if(g_bAutoSolveGlitch>1){
260 SetDlgItemText(hWnd,IDC_EDIT4,"No more glitch found - save and previous");
    UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
                                    SendMessage(hWnd,WM_COMMAND,IDC_BUTTON5,0);
                            }
                            else{
265 SetDlgItemText(hWnd,IDC_EDIT4,"No glitch found - previous");
    UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
                                    SendMessage(hWnd,WM_COMMAND,IDC_BUTTON1,0);
                            }
                            KillTimer(hWnd,1);
270                         g_bAutoSolveGlitch=1;
                    }
                    if(g_bAutoSolveGlitch==1){
    SetDlgItemText(hWnd,IDC_EDIT4,"First glitch found - read location");
    UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
275                         int nMI = g_SFT.GetIterations();
                            g_SFT.OpenFile(g_szExamine);
                            g_SFT.SetIterations(nMI);
                            CDecNumber A = CDecNumber(g_SFT.GetZoom())/(CDecNumber(↙
                                ↳ g_nExamineZoom)^(g_stExamine.size()-g_nExamine-1))↙
                                ↳ ;
                            std::string szRe = g_SFT.GetRe();
280                         std::string szIm = g_SFT.GetIm();
```

```
                        g_SFT.SetPosition(szRe,szIm,A.ToText());
                }

        char szAdd[128];
285     wsprintf(szAdd,"Add reference %d",g_bAutoSolveGlitch);
        SetDlgItemText(hWnd,IDC_EDIT4,szAdd);
        UpdateWindow(GetDlgItem(hWnd,IDC_EDIT4));
                g_SFT.AddReference(rx,ry);
                g_bAutoSolveGlitch++;
290             SetTimer(GetParent(hWnd),0,500,NULL);
                g_bExamineDirty=TRUE;
        }
        else if(uMsg==WM_USER+199 && g_bAutoSolveGlitch){
                SendMessage(hWnd,WM_TIMER,2,0);
295     }
        return 0;
}

extern const char *ExamineToolTip(int nID)
300 {
        switch(nID){
        case IDC_RADIO1:
                return "Add Reference\nclick in the view to add additional ↙
                    ↳ references";
        case IDC_RADIO2:
305             return "Set main reference.\nThe whole image will be rendered";
        case IDC_RADIO3:
                return "Erase specific parts of the view by clicking\nAfter ↙
                    ↳ erased add references in the erased areas";
        case IDC_BUTTON1:
                return "Save eventual changes and go to previous Key Frame";
310     case IDOK:
                return "Save eventual changes and go to next Key Frame";
        case IDC_BUTTON5:
                return "Save eventual changes and go to previous Key Frame\nThe ↙
                    ↳ current frame will be applied on the center of the ↙
                    ↳ previous frame";
        case IDC_BUTTON2:
315             return "Undo all changes on current Key Frame";
        case IDC_EDIT1:
                return "Current Key Frame index";
        case IDC_BUTTON3:
                return "Jump to specified Key Frame index";
320     case IDCANCEL:
                return "Close this dialog";
        case IDC_EDIT2:
                return "Show if current Key Frame is changed";
        case IDC_EDIT3:
325             return "Status of reading/saving current Key Frame";
        case IDC_BUTTON4:
                return "Automatically solve glitches in Key Frames\nThe Key ↙
                    ↳ Frames will be browsed backwards\nand stop on the first ↙
                    ↳ frame\nStart this function preferable from the last frame↙
                    ↳ ";
        case IDC_EDIT4:
                return "Shows status of automatically glitch solving";
330     case IDC_EDITMAXREFS:
```

```
                return "Choose maximum number of references to add per frame ↙
                    ↳ when solving glitches";
  default:
    static char tooltip[100];
    snprintf(tooltip, 100, "%d", nID);
335 return tooltip;
  }
}
```

83 fraktal_sft/main_examine.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Affero General Public License for more details.
15
You should have received a copy of the GNU Affero General Public License
along with this program. If not, see <https://www.gnu.org/licenses/>.
*/

20 #ifndef KF_MAIN_EXAMINE_H
#define KF_MAIN_EXAMINE_H 1

#include <windows.h>

25 extern bool Examine(HWND hWnd);
extern int WINAPI ExamineProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
extern const char *ExamineToolTip(int nID);

#endif
```

84 fraktal_sft/main.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

```
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_MAIN_H
#define KF_MAIN_H 1

#include <string>

#include <windows.h>

extern int SaveImage(const std::string &szFileName, HBITMAP bmBmp, int nQuality,↙
    ↳  const std::string &comment);
extern int SaveImage(const std::string &szFileName, const BYTE *lpBits, int ↙
    ↳ biWidth, int biHeight, int nQuality, const std::string &comment);
extern char *GetToolText(int nID,LPARAM lParam);
extern double GetDlgItemFloat(HWND hWnd,int nID);
extern void SetDlgItemFloat(HWND hWnd,int nID,double val);
extern int FileExists(const std::string &szFind);

extern std::string replace_path_filename(const std::string &path, const std::↙
    ↳ string &file);
extern std::string replace_path_extension(const std::string &path, const std::↙
    ↳ string &ext);
extern std::string get_filename_extension(const std::string &file);
extern std::string get_filename_zoom_string(const std::string &file);

extern std::string g_szFile;
extern bool g_bExamineDirty;
extern bool g_bAnim;
extern bool g_bAddReference;
extern bool g_bAddMainReference;
extern bool g_bEraser;
extern bool g_bWaitRead;

extern HICON g_hIcon;

extern HWND g_hwStatus;
extern HWND g_hwExamine;

#endif
```

85 fraktal_sft/main_iterations.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
```

```
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

    #include "main.h"
    #include "main_iterations.h"
    #include "fraktal_sft.h"
    #include "resource.h"
    #include "../formula/formula.h"
    #include "../common/tooltip.h"

    static int g_nPrevCalc = -1;

    static void UpdateIterations(HWND hWnd)
    {
            int64_t i = g_SFT.GetIterations();
            char s[100];
            snprintf(s, 100, "%lld", i);
            SetDlgItemText(hWnd,IDC_EDIT1,s);
    }

    static void RefreshIterations(HWND hWnd)
    {
            char sz[256];
            GetDlgItemText(hWnd,IDC_EDIT1,sz,sizeof(sz));
            int64_t i = atoll(sz);
            g_SFT.SetIterations(i);
    }

    static void UpdateMinIterations(HWND hWnd, int64_t i)
    {
            char s[100];
            snprintf(s, 100, "%lld", i);
            SetDlgItemText(hWnd,IDC_EDIT2,s);
    }

    static void UpdateMaxIterations(HWND hWnd, int64_t i)
    {
            char s[100];
            snprintf(s, 100, "%lld", i);
            SetDlgItemText(hWnd,IDC_EDIT5,s);
    }

    static void UpdateApproxIterations(HWND hWnd)
    {

            int64_t i = g_SFT.GetMaxApproximation();
            char s[100];
            snprintf(s, 100, "%lld", i);
            SetDlgItemText(hWnd,IDC_EDIT7,s);
    }

    static void UpdateSmoothMethod(HWND hWnd)
```

```
{
        int i = g_SFT.GetSmoothMethod();
        SendDlgItemMessage(hWnd,IDC_COMBO2,CB_SETCURSEL,i,0);
}

static void RefreshSmoothMethod(HWND hWnd)
{
        int i = SendDlgItemMessage(hWnd,IDC_COMBO2,CB_GETCURSEL,0,0);
        g_SFT.SetSmoothMethod(i);
}

static void UpdatePower(HWND hWnd)
{
        int i = g_SFT.GetPower();
        SetDlgItemInt(hWnd,IDC_COMBO3,i,FALSE);
}

static int RefreshPower(HWND hWnd, bool refresh = true)
{
        char szPower[256];
        GetDlgItemText(hWnd,IDC_COMBO3,szPower,sizeof(szPower));
        int i = atoi(szPower);
        if (refresh)
        {
                i = validate_power_for_fractal_type(g_SFT.GetFractalType(), i);
                g_SFT.SetPower(i);
        }
        return i;
}

static void UpdateFractalType(HWND hWnd, int i = -1, int p = -1)
{
        if (i < 0) i = g_SFT.GetFractalType();
        if (p < 0) p = g_SFT.GetPower();
        SendDlgItemMessage(hWnd,IDC_COMBO5,CB_SETCURSEL,i,0);
        i = SendDlgItemMessage(hWnd,IDC_COMBO5,CB_GETCURSEL,0,0);
        p = validate_power_for_fractal_type(i, p);
        update_power_dropdown_for_fractal_type(hWnd, IDC_COMBO3, i, p);
        g_SFT.SetFractalType(i);
        g_SFT.SetPower(p);
}

static int RefreshFractalType(HWND hWnd, bool refresh = true)
{
        int i = SendDlgItemMessage(hWnd,IDC_COMBO5,CB_GETCURSEL,0,0);
        if (refresh) g_SFT.SetFractalType(i);
        return i;
}

static void UpdateMaxReferences(HWND hWnd)
{
        int i = g_SFT.GetMaxReferences();
        SetDlgItemInt(hWnd,IDC_EDIT3,i,FALSE);
}

static void RefreshMaxReferences(HWND hWnd)
{
```

```
        int i = GetDlgItemInt(hWnd,IDC_EDIT3,NULL,FALSE);
        g_SFT.SetMaxReferences(i);
}

static void UpdateGlitchLowTolerance(HWND hWnd)
{
        bool b = g_SFT.GetGlitchLowTolerance();
        SendDlgItemMessage(hWnd,IDC_GLITCHLOWTOLERANCE,BM_SETCHECK,b,0);
}

static void RefreshGlitchLowTolerance(HWND hWnd)
{
        bool b = SendDlgItemMessage(hWnd,IDC_GLITCHLOWTOLERANCE,BM_GETCHECK,0,0)
            ↪ ;
        g_SFT.SetGlitchLowTolerance(b);
}

static void UpdateApproxLowTolerance(HWND hWnd)
{
        bool b = g_SFT.GetApproxLowTolerance();
        SendDlgItemMessage(hWnd,IDC_CHECK1,BM_SETCHECK,b,0);
}

static void RefreshApproxLowTolerance(HWND hWnd)
{
        bool b = SendDlgItemMessage(hWnd,IDC_CHECK1,BM_GETCHECK,0,0);
        g_SFT.SetApproxLowTolerance(b);
}

static void UpdateAutoApproxTerms(HWND hWnd)
{
        bool b = g_SFT.GetAutoApproxTerms();
        SendDlgItemMessage(hWnd,IDC_CHECK2,BM_SETCHECK,b,0);
}

static void RefreshAutoApproxTerms(HWND hWnd)
{
        bool b = SendDlgItemMessage(hWnd,IDC_CHECK2,BM_GETCHECK,0,0);
        g_SFT.SetAutoApproxTerms(b);
}

static void UpdateApproxTerms(HWND hWnd)
{
        int i = g_SFT.GetApproxTerms();
        SetDlgItemInt(hWnd,IDC_COMBO6,i,FALSE);
}

static void RefreshApproxTerms(HWND hWnd)
{
        char szTerms[256];
        GetDlgItemText(hWnd,IDC_COMBO6,szTerms,sizeof(szTerms));
        int i = atoi(szTerms);
        g_SFT.SetApproxTerms(i);
}

static void UpdateCalculations(HWND hWnd, int nCalc, int nType)
{
```

```
  char szCalc[128];
  wsprintf(szCalc,"%d",nCalc);
  int k=strlen(szCalc);
  while(k>3){
    int n=strlen(szCalc);
    while(n>k-4){
      szCalc[n+1]=szCalc[n];
      n--;
    }
    szCalc[k-3]=' ';
    k-=3;
  }
  if(nType)
    strcat(szCalc," 000 000");
  if(g_nPrevCalc!=-1){
    int nC = nCalc-g_nPrevCalc;
    if(!nType)
      nC/=1000000;
    wsprintf(szCalc+strlen(szCalc),", %d M/s",nC);
  }
  g_nPrevCalc=nCalc;
  SetDlgItemText(hWnd,IDC_EDIT8,szCalc);
}

static void UpdateJitterSeed(HWND hWnd)
{
        int i = g_SFT.GetJitterSeed();
        SetDlgItemInt(hWnd,IDC_JITTERSEED,i,FALSE);
}

static void RefreshJitterSeed(HWND hWnd)
{
        char sz[256];
        GetDlgItemText(hWnd,IDC_JITTERSEED,sz,sizeof(sz));
        int i = atoi(sz);
        g_SFT.SetJitterSeed(i);
}

static void UpdateJitterScale(HWND hWnd)
{
        double f = g_SFT.GetJitterScale();
        char szTmp[40];
        snprintf(szTmp,40,"%g",f);
        SetDlgItemText(hWnd,IDC_JITTERSCALE,szTmp);
}

static void RefreshJitterScale(HWND hWnd)
{
        char szTmp[40];
        GetDlgItemText(hWnd,IDC_JITTERSCALE,szTmp,sizeof(szTmp));
        double f = atof(szTmp);
        g_SFT.SetJitterScale(f > 0 ? f : 1);
}

static void UpdateJitterShape(HWND hWnd)
{
        int i = g_SFT.GetJitterShape();
```

```
        SendDlgItemMessage(hWnd,IDC_JITTERSHAPE,BM_SETCHECK, i != 0,0);
}

static void RefreshJitterShape(HWND hWnd)
{
        g_SFT.SetJitterShape(SendDlgItemMessage(hWnd,IDC_JITTERSHAPE,BM_GETCHECK↙
            ↳ ,0,0) ? 1 : 0);
}

static void UpdateDerivatives(HWND hWnd)
{
        int i = g_SFT.GetDerivatives();
        SendDlgItemMessage(hWnd,IDC_DERIVATIVES,BM_SETCHECK, i != 0,0);
}

static void RefreshDerivatives(HWND hWnd)
{
        g_SFT.SetDerivatives(SendDlgItemMessage(hWnd,IDC_DERIVATIVES,BM_GETCHECK↙
            ↳ ,0,0) ? 1 : 0);
}

static void UpdateReal(HWND hWnd)
{
        bool b = g_real != 0;
        SendDlgItemMessage(hWnd,IDC_CHECK3,BM_SETCHECK,b,0);
}

static void RefreshReal(HWND hWnd)
{
        g_real = SendDlgItemMessage(hWnd,IDC_CHECK3,BM_GETCHECK,0,0);
}

static void UpdateImag(HWND hWnd)
{
        bool b = g_imag != 0;
        SendDlgItemMessage(hWnd,IDC_CHECK5,BM_SETCHECK,b,0);
}

static void RefreshImag(HWND hWnd)
{
        g_imag = SendDlgItemMessage(hWnd,IDC_CHECK5,BM_GETCHECK,0,0);
}

static void UpdateSeedR(HWND hWnd)
{
        char szTmp[40];
        snprintf(szTmp,40,"%g",g_SeedR);
        SetDlgItemText(hWnd,IDC_EDIT4,szTmp);
}

static void RefreshSeedR(HWND hWnd)
{
        char szTmp[40];
        GetDlgItemText(hWnd,IDC_EDIT4,szTmp,sizeof(szTmp));
        g_SeedR = atof(szTmp);
}
```

```
static void UpdateSeedI(HWND hWnd)
{
        char szTmp[40];
        snprintf(szTmp,40,"%g",g_SeedI);
        SetDlgItemText(hWnd,IDC_EDIT9,szTmp);
}

static void RefreshSeedI(HWND hWnd)
{
        char szTmp[40];
        GetDlgItemText(hWnd,IDC_EDIT9,szTmp,sizeof(szTmp));
        g_SeedI = atof(szTmp);
}

static void UpdateFactorAR(HWND hWnd)
{
        char szTmp[40];
        snprintf(szTmp,40,"%g",g_FactorAR);
        SetDlgItemText(hWnd,IDC_EDIT28,szTmp);
}

static void RefreshFactorAR(HWND hWnd)
{
        char szTmp[40];
        GetDlgItemText(hWnd,IDC_EDIT28,szTmp,sizeof(szTmp));
        g_FactorAR = atof(szTmp);
}

static void UpdateFactorAI(HWND hWnd)
{
        char szTmp[40];
        snprintf(szTmp,40,"%g",g_FactorAI);
        SetDlgItemText(hWnd,IDC_EDIT10,szTmp);
}

static void RefreshFactorAI(HWND hWnd)
{
        char szTmp[40];
        GetDlgItemText(hWnd,IDC_EDIT10,szTmp,sizeof(szTmp));
        g_FactorAI = atof(szTmp);
}

extern INT_PTR WINAPI IterationProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM ↙
    ↳ lParam)
{
        (void) lParam;
        if(uMsg==WM_INITDIALOG || uMsg==WM_TIMER){
                if(uMsg==WM_INITDIALOG){
                        SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon↙
                            ↳ ));
                        SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon))↙
                            ↳ ;
                        InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,1);
                        UpdateIterations(hWnd);
                        SetTimer(hWnd,0,1000,NULL);

                        SendDlgItemMessage(hWnd,IDC_COMBO2,CB_ADDSTRING,0,(↙
```

```
                    ↪ LPARAM)"High bailout");
                SendDlgItemMessage(hWnd,IDC_COMBO2,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"Bailout=2");
                SendDlgItemMessage(hWnd,IDC_COMBO2,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"Low bailout");
                UpdateSmoothMethod(hWnd);

                combo5_addstrings(hWnd, IDC_COMBO5);
                SIZE sc;
                HDC hDC = GetDC(NULL);
                SelectObject(hDC,(HFONT)GetStockObject(ANSI_VAR_FONT));
                int i, nMaxWidth=0;
                for(i=0;i<SendDlgItemMessage(hWnd,IDC_COMBO5,CB_GETCOUNT↙
                    ↪ ,0,0);i++){
                    int n = SendDlgItemMessage(hWnd,IDC_COMBO5,↙
                        ↪ CB_GETLBTEXTLEN,i,0);
                    char *szT = new char[n+1];
                    SendDlgItemMessage(hWnd,IDC_COMBO5,CB_GETLBTEXT,↙
                        ↪ i,(LPARAM)szT);
                    GetTextExtentPoint32A(hDC,szT,strlen(szT),&sc);
                    if(sc.cx>nMaxWidth)
                        nMaxWidth = sc.cx;
                    delete[] szT;
                }
                SendDlgItemMessage(hWnd,IDC_COMBO5,CB_SETDROPPEDWIDTH,↙
                    ↪ nMaxWidth+8+GetSystemMetrics(SM_CXHTHUMB),0);
                UpdateFractalType(hWnd);
        int nType = RefreshFractalType(hWnd, false);

                UpdateGlitchLowTolerance(hWnd);
                UpdateApproxLowTolerance(hWnd);
                UpdateAutoApproxTerms(hWnd);
                UpdatePower(hWnd);
                UpdateMaxReferences(hWnd);

                SendDlgItemMessage(hWnd,IDC_COMBO6,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"5");
                SendDlgItemMessage(hWnd,IDC_COMBO6,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"10");
                SendDlgItemMessage(hWnd,IDC_COMBO6,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"15");
                SendDlgItemMessage(hWnd,IDC_COMBO6,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"20");
                SendDlgItemMessage(hWnd,IDC_COMBO6,CB_ADDSTRING,0,(↙
                    ↪ LPARAM)"30");
                UpdateApproxTerms(hWnd);

                EnableWindow(GetDlgItem(hWnd,IDC_CHECK2),nType==0 && ↙
                    ↪ g_SFT.GetPower()==2);
                EnableWindow(GetDlgItem(hWnd,IDC_COMBO6),nType==0 && ↙
                    ↪ g_SFT.GetPower()==2);

                UpdateReal(hWnd);
                UpdateImag(hWnd);
                UpdateSeedR(hWnd);
                UpdateSeedI(hWnd);
                UpdateFactorAR(hWnd);
```

```
                        UpdateFactorAI(hWnd);
                        UpdateJitterSeed(hWnd);
                        UpdateJitterScale(hWnd);
                        UpdateJitterShape(hWnd);
                        UpdateDerivatives(hWnd);
                }
                int64_t nMin, nMax;
                int nCalc=0,nType=0;

                g_SFT.GetIterations(nMin,nMax,&nCalc,&nType);
        UpdateMinIterations(hWnd, nMin);
        UpdateMaxIterations(hWnd, nMax);
                UpdateApproxIterations(hWnd);
                if(uMsg==WM_TIMER){
          UpdateCalculations(hWnd,nCalc,nType);
                        if(SendDlgItemMessage(hWnd,IDC_CHECK2,BM_GETCHECK,0,0))
                                UpdateApproxTerms(hWnd);
                }
                else
                        g_nPrevCalc=-1;
                return 1;
          }
          else if(uMsg==WM_COMMAND){
                if(wParam==IDOK){
                        g_SFT.UndoStore();
                        g_bExamineDirty=TRUE;
                        RefreshDerivatives(hWnd);
                        RefreshJitterSeed(hWnd);
                        RefreshJitterShape(hWnd);
                        RefreshJitterScale(hWnd);
                        RefreshReal(hWnd);
                        RefreshImag(hWnd);
                        RefreshSeedR(hWnd);
                        RefreshSeedI(hWnd);
                        RefreshFactorAR(hWnd);
                        RefreshFactorAI(hWnd);
                        RefreshSmoothMethod(hWnd);
                        RefreshApproxTerms(hWnd);
                        RefreshGlitchLowTolerance(hWnd);
                        RefreshApproxLowTolerance(hWnd);
                        RefreshAutoApproxTerms(hWnd);
                        RefreshFractalType(hWnd);
                        RefreshPower(hWnd);
                        RefreshMaxReferences(hWnd);
          RefreshIterations(hWnd);
                        ExitToolTip(hWnd);
                        EndDialog(hWnd, 1);
                }
                else if(wParam==IDC_CHECK3){
                        if(!SendDlgItemMessage(hWnd,IDC_CHECK3,BM_GETCHECK,0,0) ↙
                            ↳ && !SendDlgItemMessage(hWnd,IDC_CHECK5,BM_GETCHECK↙
                            ↳ ,0,0))
                                SendDlgItemMessage(hWnd,IDC_CHECK5,BM_SETCHECK↙
                                    ↳ ,1,0);
                }
                else if(wParam==IDC_CHECK5){
                        if(!SendDlgItemMessage(hWnd,IDC_CHECK3,BM_GETCHECK,0,0) ↙
```

```
                                    && !SendDlgItemMessage(hWnd,IDC_CHECK5,BM_GETCHECK
                                    ,0,0))
                                        SendDlgItemMessage(hWnd,IDC_CHECK3,BM_SETCHECK
                                            ,1,0);
                    }
                    else if(wParam==IDCANCEL){
                            ExitToolTip(hWnd);
                            EndDialog(hWnd,0);
                    }
                    else if(HIWORD(wParam)==CBN_SELCHANGE && LOWORD(wParam)==
                        IDC_COMBO5){
                            int nType = RefreshFractalType(hWnd, false);
                            int nPow = RefreshPower(hWnd, false);
                            UpdateFractalType(hWnd, nType, nPow);
#ifndef _DEBUG
                            EnableWindow(GetDlgItem(hWnd,IDC_COMBO3),nType<=4);
#endif
                    }
                    int nType = SendDlgItemMessage(hWnd,IDC_COMBO5,CB_GETCURSEL,0,0)
                        ;
                    EnableWindow(GetDlgItem(hWnd,IDC_CHECK2),nType==0 &&
                        GetDlgItemInt(hWnd,IDC_COMBO3,NULL,FALSE)==2);
                    EnableWindow(GetDlgItem(hWnd,IDC_COMBO6),nType==0 &&
                        GetDlgItemInt(hWnd,IDC_COMBO3,NULL,FALSE)==2);
                    EnableWindow(GetDlgItem(hWnd,IDC_COMBO6),!SendDlgItemMessage(
                        hWnd,IDC_CHECK2,BM_GETCHECK,0,0));
            }
            return 0;
}

extern const char *IterationToolTip(int nID)
{
  switch(nID){
  case IDC_EDIT1:
    return "Maximum number of iterations";
  case IDC_EDIT2:
    return "Minimum number of iteration in current view";
  case IDC_EDIT5:
    return "Maximum number of iteration in current view";
  case IDC_EDIT7:
    return "Iterations skipped by Series Approximation";
  case IDC_COMBO2:
    return "Bailout value for iterations";
  case IDC_COMBO3:
    return "Power of Mandelbrot function";
  case IDC_EDIT3:
    return "Maximum of extra references for glitch correction";
  case IDC_GLITCHLOWTOLERANCE:
    return "Checked for low tolerance of Glitch Detection\nComplex images may
        need this to be checked to be rendered correctly\nThe render time may
        be faster if this checkbox is not checked";
  case IDC_CHECK1:
    return "Checked for low tolerance of Series Approximation\nComplex images
        may need this to be checked to be rendered correctly\nThe render time
        may be faster if this checkbox is not checked";
  case IDC_CHECK2:
    return "Terms for Series approximation is adjusted\nbased on the number of
```

```
        ↳ pixels to be rendered.";
490   case IDC_COMBO5:
        return "List of type of Mandelbrot based Fractals\nSome of them have ↙
            ↳ additional Power options";
      case IDC_COMBO6:
        return "Terms for Series approximation.\nMore terms usually yield more ↙
            ↳ skipped iterations and faster rendering,\nhowever is more time ↙
            ↳ consuming to be processed";
      case IDC_EDIT8:
495     return "Display number of calculations performed";
      case IDC_CHECK3:
        return "Include real part when checking bailout.\nUncheck for variation";
      case IDC_CHECK5:
        return "Include imaginary part when checking bailout.\nUncheck for variation↙
            ↳ ";
500   case IDC_JITTERSEED:
        return "Pseudo-random number generator seed for pixel jitter\nSet to 0 to ↙
            ↳ disable jitter";
      case IDC_JITTERSCALE:
        return "Pixel jitter amount\nDefault 1.0";
      case IDC_JITTERSHAPE:
505     return "Select checkbox to use Gaussian jitter\nUncheck for uniform";
      case IDC_DERIVATIVES:
        return "Select checkbox to compute derivatives\nRequired for analytic DE";
      case 1002:
        return "Real seed value (0 is standard)";
510   case 1003:
        return "Imaginary seed value (0 is standard)";
      case IDOK:
        return "Apply and close";
      case IDCANCEL:
515     return "Close and undo";
      default:
        static char tooltip[100];
        snprintf(tooltip, 100, "%d", nID);
        return tooltip;
520   }
}
```

86 fraktal_sft/main_iterations.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen
5

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15

You should have received a copy of the GNU Affero General Public License
```

```
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_MAIN_ITERATIONS_H
#define KF_MAIN_ITERATIONS_H 1

#include <windows.h>

extern INT_PTR WINAPI IterationProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
extern const char *IterationToolTip(int nID);

#endif
```

87 fraktal_sft/main_position.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "main.h"
#include "main_position.h"
#include "fraktal_sft.h"
#include "resource.h"
#include "newton.h"
#include "../common/tooltip.h"

static void FixNumber(char *sz)
{
        int a, b, p, e, m;
        for(a=0;sz[a]==' ' || sz[a]=='\t' || sz[a]=='\r' || sz[a]=='\n';a++);
        for(b=p=e=m=0;sz[a];a++){
                if(isdigit(sz[a]) || (!e && (sz[a]=='e' || sz[a]=='E')) || (!p && sz[a]=='.') || (!m && sz[a]=='-'))
                        sz[b++]=sz[a];
                if(!e && (sz[a]=='e' || sz[a]=='E'))
                        e=1;
                if(!p && sz[a]=='.')
                        p=1;
                if(!m && sz[a]=='-')
                        m=1;
        }
```

```
        sz[b]=0;
}

extern int WINAPI PositionProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
        (void) lParam;
        if(uMsg==WM_INITDIALOG){
                SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
                InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,2);
                SendDlgItemMessage(hWnd,IDC_EDIT1,EM_SETLIMITTEXT,0,0);
                SendDlgItemMessage(hWnd,IDC_EDIT3,EM_SETLIMITTEXT,0,0);
    std::string re = g_SFT.GetRe();
    std::string im = g_SFT.GetIm();
    std::string z = g_SFT.GetZoom();
                SetDlgItemText(hWnd,IDC_EDIT1,re.c_str());
                SetDlgItemText(hWnd,IDC_EDIT3,im.c_str());
                SetDlgItemText(hWnd,IDC_EDIT4,z.c_str());
                int64_t nMin, nMax;
                g_SFT.GetIterations(nMin,nMax);
                char s[100];
                snprintf(s, 100, "%lld", nMin);
                SetDlgItemText(hWnd,IDC_EDIT2,s);
                snprintf(s, 100, "%lld", nMax);
                SetDlgItemText(hWnd,IDC_EDIT5,s);
                snprintf(s, 100, "%lld", g_period);
                SetDlgItemText(hWnd,IDC_LOCATION_PERIOD,s);
                return 1;
        }
        else if(uMsg==WM_COMMAND){
                if(wParam==IDOK){
                        g_SFT.UndoStore();
                        int n = GetWindowTextLength(GetDlgItem(hWnd,IDC_EDIT1));
                        char *szR = new char[n+1];
                        GetDlgItemText(hWnd,IDC_EDIT1,szR,n+1);
                        FixNumber(szR);
                        n = GetWindowTextLength(GetDlgItem(hWnd,IDC_EDIT3));
                        char *szI = new char[n+1];
                        GetDlgItemText(hWnd,IDC_EDIT3,szI,n+1);
                        FixNumber(szI);
                        n = GetWindowTextLength(GetDlgItem(hWnd,IDC_EDIT4));
                        char *szZ = new char[n+1];
                        GetDlgItemText(hWnd,IDC_EDIT4,szZ,n+1);
                        FixNumber(szZ);
                        n = GetWindowTextLength(GetDlgItem(hWnd,IDC_EDIT4));
                        char *szP = new char[n+1];
                        GetDlgItemText(hWnd,IDC_LOCATION_PERIOD,szP,n+1);
                        g_period = atoll(szP);
                        g_SFT.SetPosition(szR,szI,szZ);
                        delete [] szR;
                        delete [] szI;
                        delete [] szZ;
                        delete [] szP;
                        ExitToolTip(hWnd);
                        EndDialog(hWnd,1);
                }
                else if(wParam==IDCANCEL){
```

```
                        ExitToolTip(hWnd);
                        EndDialog(hWnd,0);
                }
        }
        return 0;
}

extern const char *PositionToolTip(int nID)
{
  switch(nID){
  case IDC_EDIT1:
    return "Real value";
  case IDC_EDIT3:
    return "Imaginary value";
  case IDC_EDIT4:
    return "Zoom value";
  case IDC_EDIT2:
    return "Display minimum iteration value of current location";
  case IDC_EDIT5:
    return "Display maximum iteration value of current location";
  case IDC_LOCATION_PERIOD:
    return "Display period of last Newton zoom, sets limit for NanoMB2";
  case IDOK:
    return "Apply and close";
  case IDCANCEL:
    return "Close and undo";
  default:
    static char tooltip[100];
    snprintf(tooltip, 100, "%d", nID);
    return tooltip;
  }
}
```

88 fraktal_sft/main_position.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_MAIN_POSITION_H
#define KF_MAIN_POSITION_H 1
```

```
#include <windows.h>

extern int WINAPI PositionProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
extern const char *PositionToolTip(int nID);

#endif
```

89 fraktal_sft/nanomb1.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef NANOMB1_H
#define NANOMB1_H 1

#include "CDecNumber.h"
#include "complex.h"
#include "floatexp.h"

struct NanoMB1_Reference;
NanoMB1_Reference *NanoMB1_Reference_Calculate(const complex<decNumber> &c, int ↙
    ↪ bm, int bn, int64_t period, int64_t maxiters, floatexp r0, double er2, ↙
    ↪ bool glitchLowTol, volatile bool &stop, int &m_nRDone);
void NanoMB1_Pixel(const NanoMB1_Reference *ctx, complex<floatexp> dc, floatexp ↙
    ↪ pixel_spacing, int64_t maxiters, bool &bGlitch, int64_t &antal, double &↙
    ↪ test1, double &test2, double &de, bool interior_checking);

#endif
```

90 fraktal_sft/nanomb1.inc

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
```

```
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  /*
    Largely based on knighty's public domain nanomb1/nanomb2 code from:
    <https://fractalforums.org/f/28/t/277/msg8132#msg8132>
    <https://fractalforums.org/f/28/t/277/msg7983#msg7983>
    */
25
    #include "nanomb1.h"
    #include "Settings.h"
    #define re m_r
    #define im m_i
30  // type aliases
    typedef int64_t N;
    typedef floatexp R_lo;
    typedef decNumber R_hi;
    typedef complex<R_lo> C_lo;
35  typedef complex<R_hi> C_hi;

    R_lo r_lo(const R_hi &x)
    {
      return mpfr_get_fe(x.backend().data());
40  }

    C_lo c_lo(const C_hi &z)
    {
      return C_lo(r_lo(z.re), r_lo(z.im));
45  }

    #define TABSIZ (MAX_APPROX_TERMS + 1)

    // m.cpp 2018 Knighty
50  // based on some code by Claude Heiland-Allen
    // LICENSE: public domain, cc0, whatever
    // acceleration method for rendering Mandelbrot set
    // based on approximation of n iterations around minibrot of period n using ↙
        ↪ bivariate polynomial
    //-------------------------------------------------------------------------------------------------
        ↪
55  // Knighty (2018-07-10): Edited the code in order to make it possible to render ↙
        ↪ locations where the
    //                       center is not exactly on the nucleus:
    //                       - added the tmpPolyClass: used to find the position if ↙
        ↪ the nucleus.
    //                       - added code to compute the reference orbit wrt the ↙
        ↪ nucleus.
    //                       - added iteratePtDE2() which is almost exactly the same↙
        ↪  as iteratePtDE() but
60  //                         designed to take into account the new reference orbit↙
```

```
    ↪ .
//-------------------------------------------------------------------------------------------------
    ↪
// compile:
//   g++ -std=c++11 -Ofast -mfpmath=387 -march=native mb.cpp -lmpfr -lgmp -Wall -↙
    ↪ Wextra -pedantic -fopenmp
//-------------------------------------------------------------------------------------------------
    ↪
// run:
//   ./a --help
// view results:
//   display out.ppm

#include <vector>

//-------------------------------------------------------------------------------------------------
    ↪
class refElem{
public:
        C_lo z;
        R_lo g;
        refElem(const C_lo &z, const R_lo &t)
        : z(z)
        , g(t * norm(z))
        { };
};
class refClass{
        R_lo m_tolerance;
        N m_n;
        std::vector<refElem> m_v;
public:
        refClass(R_lo tolerance): m_tolerance(tolerance), m_n(0) {}
        R_lo getTolerance() { return m_tolerance; }
        void adde(C_lo c){ m_v.push_back(refElem(c, m_tolerance)); m_n++;}
        const refElem & operator[](N i) const { return m_v[i % m_n];}
};

class perturbationClassD{
        C_lo m_d0;
        C_lo m_d;
        C_lo m_dd;
        N    m_n0;
        R_lo m_col;
        bool m_escaped;
        bool m_glitched;
        int64_t m_iters;
        R_lo m_test1;
        R_lo m_test2;
        R_lo m_de;
public:
        perturbationClassD(C_lo d0, C_lo d, C_lo dd, N n0): m_d0(d0), m_d(d), ↙
            ↪ m_dd(dd), m_n0(n0), m_col(0), m_escaped(false), m_glitched(false),↙
            ↪  m_iters(0), m_test1(0), m_test2(0), m_de(0) {}
        void run(const refClass &ref, N maxiter, R_lo er2, R_lo pixel_spacing){
                for(N i = m_n0 ; i < maxiter; i++){
                        C_lo zn(ref[i].z);
                        R_lo r(norm(zn + m_d));
```

```
                        m_test2 = m_test1;
                        m_test1 = r;
                        if (r > er2){
                                m_escaped = true;
                                m_iters = i;
                                m_de = sqrt(r) * log(r) / abs(m_dd * ↙
                                    ↳ pixel_spacing);
                                return;
                        }
                        if (r < ref[i].g)
                        {
                                m_glitched = true;
                                m_iters = i;
                                return;
                        }
                        m_dd = R_lo(2) * m_dd * (m_d + zn) + R_lo(1);
                        m_d = m_d * (R_lo(2) * zn + m_d) + m_d0;
                }
        }
        bool haveEscaped(){return m_escaped;}
        bool haveGlitched(){return m_glitched;}
        int64_t getIters(){return m_iters;}
        R_lo getTest1(){return m_test1;}
        R_lo getTest2(){return m_test2;}
        R_lo getDE(){return m_de;}
};

class uniPolyClass;
class tmpPolyClass;
class biPolyClass {
        friend class uniPolyClass;
        friend class tmpPolyClass;
        N m_m, m_n;
        C_lo tab[TABSIZ][TABSIZ];
        C_lo ttab[TABSIZ][TABSIZ];
        void mcopy(){
                for(N l=0; l <= m_m; l++)
                        for(N c=0; c<= m_n; c++)
                                ttab[l][c] = tab[l][c];
        }
        C_lo csqrc(N k, N l){
                C_lo v(0);
                for(N i=0; i <= k; i++)
                        for(N j=0; j <= l; j++)
                                v += ttab[i][j] * ttab[k-i][l-j];
                return v;
        }
public:
        biPolyClass(N m, N n): m_m(m), m_n(n) {
                for(N l=0; l <= m_m; l++)
                        for(N c=0; c<= m_n; c++)
                                tab[l][c] = C_lo(0);
                tab[1][0] = C_lo(1);
        }

        void sqr(){
                mcopy();
```

```
                for(N i=0; i <= m_m; i++)
                        for(N j=0; j <= m_n; j++)
                                tab[i][j] = csqrc(i,j);
170         }

            void cstep(C_lo z){
                sqr();
                tab[0][0]  = z;
175             tab[0][1] += C_lo(1);
            }

            C_lo eval(C_lo u, C_lo v){
                C_lo r(0);
180             C_lo ui(1);
                for(N i=0; i <= m_m; i+=2){
                        C_lo vj(ui);
                        for(N j=0; j <= m_n; j++){
                                r += tab[i][j] * vj;
185                             vj *= v;
                        }
                        ui *= u*u;
                }
                return r;
190         }

            C_lo eval_dc(C_lo u, C_lo v){
                C_lo r(0);
                C_lo ui(1);
195             for(N i=0; i <= m_m; i+=2){
                        C_lo vj(ui);
                        for(N j=1; j <= m_n; j++){
                                r += R_lo(j) * tab[i][j] * vj;
                                vj *= v;
200                     }
                        ui *= u*u;
                }
                return r;
            }
205
            C_lo eval_dz(C_lo u, C_lo v){
                C_lo r(0);
                C_lo ui(u);
                for(N i=2; i <= m_m; i+=2){
210                     C_lo vj(C_lo(i) * ui);
                        for(N j=0; j <= m_n; j++){
                                r += tab[i][j] * vj;
                                vj *= v;
                        }
215                     ui *= u*u;
                }
                return r;
            }

220         R_lo getRadius(){
                //return abs(tab[0][1])/abs(tab[0][2]);
                R_lo r(0);
                for(int i = 0; i < 10; i++){
```

```
                        C_lo den(0);
                        R_lo rr(1);
                        for(int j = 2; j <=m_n; j++){
                                den += tab[0][j] * rr;
                                rr  *= r;
                        }
                        r = abs(tab[0][1])/abs(den);
                }
                return R_lo(0.5) * r;
        }
};

//temporary poly class for solving for nucleus relative position. It is initialized with the part of the SSA that depends only on c.
class tmpPolyClass {
        N m_m;
        C_lo b[TABSIZ];
public:
        tmpPolyClass(const biPolyClass &p): m_m(p.m_n) {
                for(N i = 0; i <= m_m; i++)
                        b[i] = p.tab[0][i];
        }

        //evaluation function. It would be nice to add an ()-operator... or not. :)
        C_lo eval(C_lo u){
                C_lo r(0);
                C_lo ui(1);
                for(N i=0; i <= m_m; i++){
                        r += b[i] * ui;
                        ui *= u;
                }
                return r;
        }

        //evaluate derivative.
        C_lo evalD(C_lo u){
                C_lo r(0);
                C_lo ui(1);
                for(N i=1; i <= m_m; i++){
                        r += R_lo(i) * b[i] * ui;
                        ui *= u;
                }
                return r;
        }

        //Gives the nearest root to the 0. To use if and when applicable (that is the reference is near 0... atom domain thing!)
        //Newton should do the job (otherwise IA-Newton ?).
        C_lo getRoot(){
                C_lo rt(0);
                //R_lo t = abs(eval(rt));
                for(N i=0; i<30; i++){
                        C_lo num = eval(rt);
                        C_lo den = evalD(rt);
                        C_lo delta = num / den;
                        num = rt;
```

```
                        rt -= delta;
                        if( rt.re == num.re && rt.im == num.im) break;
280                 }
                    return rt;
            }
    };

285 // https://fractalforums.org/f/28/t/277/msg7952#msg7952
    class uniPolyClass {
            N m_m;
            C_lo b[TABSIZ];
            C_lo dbdc[TABSIZ];
290 public:
            uniPolyClass(const biPolyClass &p, C_lo c): m_m(p.m_m) {
                    for(N i = 0; i <= m_m; i += 2)
                    {
                      C_lo s(0), ds(0), cj(1), cj1(0);
295                   for (N j = 0; j <= p.m_n; ++j)
                      {
                        s += p.tab[i][j] * cj;
                        ds += C_lo(j) * p.tab[i][j] * cj1;
                        cj *= c;
300                     cj1 *= c;
                        if (j == 0) cj1 = C_lo(1);
                      }
                      b[i] = s;
                      dbdc[i] = ds;
305                 }
            }

            void eval(C_lo &z) const {
              C_lo zs(0), zi(1);
310           for (N i = 0; i <= m_m; i += 2)
              {
                zs += b[i] * zi;
                zi *= z * z;
              }
315           z = zs;
            }

            void eval(C_lo &z, C_lo &dc) const {
              C_lo zs(0), dcs(0), zi(1), zi1(0);
320           for (N i = 0; i <= m_m; i += 2)
              {
                dcs += C_lo(i) * b[i] * zi1 * dc + dbdc[i] * zi;
                zs += b[i] * zi;
                zi *= z * z;
325             zi1 *= z * z;
                if (i == 0) zi1 = z;
              }
              z = zs;
              dc = dcs;
330         }

            void eval_dz(C_lo &z, C_lo &dz) const {
              C_lo zs(0), dzs(0), zi(1), zi1(0);
              for (N i = 0; i <= m_m; i += 2)
```

```
335             {
                  dzs += C_lo(i) * b[i] * zi1 * dz;
                  zs += b[i] * zi;
                  zi *= z * z;
                  zi1 *= z * z;
340               if (i == 0) zi1 = z;
                }
                z = zs;
                dz = dzs;
              }
345
              void eval(C_lo &z, C_lo &dz, C_lo &dc, C_lo &dzdz, C_lo &dcdz) const {
                C_lo zs(0), dzs(0), dcs(0), dzdzs(0), dcdzs(0), zi(1), zi1(0), zi2(0);
                for (N i = 0; i <= m_m; i += 2)
                {
350               dcdzs += C_lo(i) * C_lo(i - 1) * b[i] * zi2 * dz * dc + C_lo(i) * b[↙
                      ↪ i] * zi1 * dcdz + C_lo(i) * dbdc[i] * zi1 * dz;
                  dzdzs += C_lo(i) * C_lo(i - 1) * b[i] * zi2 * dz * dz + C_lo(i) * b[↙
                      ↪ i] * zi1 * dzdz;
                  dcs += C_lo(i) * b[i] * zi1 * dc + dbdc[i] * zi;
                  dzs += C_lo(i) * b[i] * zi1 * dz;
                  zs += b[i] * zi;
355               zi *= z * z;
                  zi1 *= z * z;
                  zi2 *= z * z;
                  if (i == 0) zi1 = z;
                  if (i == 0) zi2 = C_lo(1);
360             }
                z = zs;
                dz = dzs;
                dc = dcs;
                dzdz = dzdzs;
365             dcdz = dcdzs;
              }
};

void iteratePtDE2(const C_lo &d0, const biPolyClass &fp, const refClass &ref, ↙
    ↪ const C_lo &nucleusPos, N period, const R_lo &Bout, const R_lo &er2, const↙
    ↪  R_lo &pixel_spacing, N maxiter, bool &bGlitch, int64_t &antal, double &↙
    ↪ test1, double &test2, double &de, bool interior_checking)
370 {
        C_lo d(0);
        C_lo dd(0);
        N i(0);
        N si(0);
375     R_lo d_norm(1.0/0.0);
        if(abs(d0)<Bout){
          uniPolyClass up(fp, d0);
              while(i<maxiter && norm(d)<Bout){
                    up.eval(d, dd);
380                 i+=period;
                    si++;
                    if (interior_checking)
                    {
                    if (norm(d) < d_norm)
385                 {
                      d_norm = norm(d);
```

```
                                // check interior: newton iterations for w0 = f^(si*p) (w0, d0)
                                C_lo w0(d);
                                C_lo dw;
390                             bool converged = false;//------------------
                                for (int step = 0; step < 16; ++step)
                                {
                                  C_lo w(w0);
                                  dw = C_lo(1);
395                               for (int64_t n = 0; n < si; ++n)
                                  {
                                    up.eval_dz(w, dw);
                                  }
                                  C_lo w0_next = w0 - (w - w0) / (dw - C_lo(1));
400                               //R_lo delta = norm(w0_next - w0);
                                  C_lo delta = (w0_next - w0);//-----------------
                                  w0 = w0_next;
                                  //R_lo epsilon2(0); // FIXME
                                  //if (delta <= epsilon2) break; // converged
405                               if (abs(delta.re) < abs(w0.re) * R_lo(1e-6) && abs(delta.im) < abs(w0.im) * R_lo(1e-6)) { converged = true; break;}//-----------------------
                                }
                                if (converged && norm(dw) < R_lo(1))
                                {
                                  bGlitch = 0;
410                               antal = maxiter;
                                  test1 = 0;
                                  test2 = 0;
                                  de = 0;
                                  return;
415 #if 0
                                  // is interior, calculate interior DE
                                  C_lo z(w0), dz(1), dc(0), dzdz(0), dcdz(0);
                                  for (int n = 0; n < si; ++n)
                                    up.eval(z, dz, dc, dzdz, dcdz);
420                               R_lo de = (R_lo(1) - norm(dz)) / abs(dcdz + (dzdz * dc) / (C_lo(1) - dz));
                                  iters = si * period;
                                  trans = 0.0f;
                                  return -de;
    #endif
425                             }
                              }
                              }
                    }
            } //else dd = C_lo(1);//knighty: That was a mistake. We begin iterating at 0 so derivative doesn't have to be changed.
430         if(i>=maxiter)
            {
              bGlitch = 0;
              antal = maxiter;
              test1 = 0;
435           test2 = 0;
              de = 0;
              return;
```

```
        }
        //d0 and d have to be transformed into ref frame.
        C_lo d0_(d0 - nucleusPos); // FIXME unused?
        //Use perturbation to finish the job.
        //ToInvestigate: Usually when arriving here (same thing with SA) delta_c
            ↘ (d0 here) is too small wrt d.
        //              One can use perturbation without adding delta_c. Some
            ↘ other optimizations may become possible. (from comment by
            ↘ Pauldelbrot @fractalforums.com)
        d -= ref[i].z; //put d in reference orbit's local frame
        perturbationClassD p(d0,d,dd,i);//We use original d0 because the
            ↘ reference orbit was computed there.
        p.run(ref, maxiter, er2, pixel_spacing);
        if (p.haveGlitched())
        {
          bGlitch = 1;
          antal = p.getIters();
          test1 = p.getTest1().todouble();
          test2 = p.getTest2().todouble();
          de = p.getDE().todouble();
          return;
        }
        else if(p.haveEscaped())
        {
          bGlitch = 0;
          antal = p.getIters();
          test1 = p.getTest1().todouble();
          test2 = p.getTest2().todouble();
          de = p.getDE().todouble();
          return;
        }
        else
        {
          antal = maxiter;
          test1 = 0;
          test2 = 0;
          de = 0;
          return;
        }
}

#if 0
  floatexp tmax(radius);
  precision = N(-tmax.exp) + 64;
  if (force_type == "double")
    main1(c, bm, bn, period, maxiters, width, height, filename, kfb_filename,
        ↘ double(tmax));
  else
  if (force_type == "long double")
    main1(c, bm, bn, period, maxiters, width, height, filename, kfb_filename, (
        ↘ long double)(tmax));
  else
  if (force_type == "floatexp")
    main1(c, bm, bn, period, maxiters, width, height, filename, kfb_filename,
        ↘ floatexp(tmax));
  else
  if ((precision - 64) * 3 * (bm+2*bn) < 500 * 20 ){ // * (bm+bn) because
```

```
      ↪ scaling is not yet implemented
    cerr << "using double ..." << endl;
        main1(c, bm, bn, period, maxiters, width, height, filename, kfb_filename↙
            ↪ , double(tmax));
  }
  else
  if ((precision - 64) * 3 * (bm+2*bn) < 8000  * 20 ){ // * (bm+bn) because ↙
      ↪ scaling is not yet implemented
    cerr << "using long double ..." << endl;
        main1(c, bm, bn, period, maxiters, width, height, filename, kfb_filename↙
            ↪ , (long double)(tmax));
  }
  else
  if (precision < INT_MAX){
        cerr << "using long double ..." << endl;
    main1(c, bm, bn, period, maxiters, width, height, filename, kfb_filename, ↙
        ↪ floatexp(tmax));
  }
  else
  {
    cerr << "bad precision" << endl;
    abort();
  }
#endif


//-------------------------------------------------------------------------------------------------
    ↪
// entry point

struct NanoMB1_Reference
{
  biPolyClass SSA;
  refClass ref;
  C_lo nucleusPos;
  N period;
  R_lo Bout;
  R_lo er2;
  NanoMB1_Reference(const biPolyClass &SSA, const refClass &ref, const C_lo &↙
      ↪ nucleusPos, N period, const R_lo &Bout, const R_lo &er2)
  : SSA(SSA)
  , ref(ref)
  , nucleusPos(nucleusPos)
  , period(period)
  , Bout(Bout)
  , er2(er2)
  { };
};

NanoMB1_Reference *NanoMB1_Reference_Calculate(const C_hi &c, N bm, N bn, N ↙
    ↪ period, N maxiters, R_lo r0, double er2, bool lowGlitchTol, volatile bool ↙
    ↪ &stop, int64_t &m_nRDone)
{
  double glitch_threshold = 0.0000001;
  if (lowGlitchTol)
    glitch_threshold = sqrt(glitch_threshold);
  //Compute bivariate polynomial approximating period iterations↙
```

```
    ↪ -------------------------
  biPolyClass fp(bm,bn);
  refClass ref(glitch_threshold);
  C_hi z(0);
  C_lo zlo(0);
  for(N i = 0; i < period; i++){
        ref.adde(zlo);
        z = z*z + c;
        zlo = c_lo(z);
        fp.cstep(zlo);
        m_nRDone++;
  }
  R_lo Bout = fp.getRadius();
  //-------------------------------------------------------------------------------------↙
    ↪
  //In case the location is not "exactly" at a nucleus, we need to "correct" the↙
    ↪  perturbation reference.
  //This is because the reference orbit is computed for only one period. If the ↙
    ↪ reference c is outside the mbset, it will eventually escape or be likely↙
    ↪  off 0.
  //Find the nucleus of the minibrot.
  tmpPolyClass tp(fp);
  C_lo nucleusPos = tp.getRoot();
  //Rebase the reference orbit to the nucleus
  refClass ref1(glitch_threshold);
  C_lo zlo1(0);
  for(N i = 0; i < period; i++){
        zlo = ref[i].z;
        ref1.adde(zlo+zlo1);
        zlo1 = zlo1 * (zlo1 + R_lo(2) * zlo) + nucleusPos;
  }
  //At this point zlo+zlo1 should be very close to 0
  //zlo = c_lo(z, tmax);
  //cout << "----------------" << endl << "at iteration: " << period << ", zlo1 ↙
    ↪ = " << zlo1 << endl;
  //cout << "zlo1+zlo = " << zlo1+zlo << endl;
  //-------------------------------------------------------------------------------------↙
    ↪
  if (stop)
    return nullptr;
  else
    return new NanoMB1_Reference(fp, ref, nucleusPos, period, Bout, er2);
}

void NanoMB1_Pixel(const NanoMB1_Reference *ctx, C_lo dc, R_lo pixel_spacing, N ↙
    ↪ maxiters, bool &bGlitch, int64_t &antal, double &test1, double &test2, ↙
    ↪ double &de, bool interior_checking)
{
  iteratePtDE2(dc, ctx->SSA, ctx->ref, ctx->nucleusPos, ctx->period, ctx->Bout, ↙
      ↪ ctx->er2, pixel_spacing, maxiters, bGlitch, antal, test1, test2, de, ↙
      ↪ interior_checking);
}
```

91 fraktal_sft/nanomb1_perturbation.cpp

```
/*
Kalles Fraktaler 2
```

```
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "nanomb1.h"

    void CFraktalSFT::MandelCalcNANOMB1()
    {
25          bool interior_checking = GetInteriorChecking();
            m_bIterChanged = TRUE;
            int x, y, w, h;
            while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
                    int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
30                  if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
                            SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, ↙
                                ↳ w, h);
                            if (m_bMirrored)
                                    Mirror(x, y);
                            continue;
35                  }
                    if (GuessPixel(x, y, w, h))
                            continue;

                    floatexp D0r = 0;
40                  floatexp D0i = 0;
                    floatexp daa = 1;
                    floatexp dab = 0;
                    floatexp dba = 0;
                    floatexp dbb = 1;
45                  GetPixelCoordinates(x, y, D0r, D0i, daa, dab, dba, dbb);

                    complex<floatexp> dc(D0r, D0i);
                    bool bGlitch = 0;
                    int64_t antal = 0;
50                  double test1 = 0, test2 = 0, de = 0;

                    if (m_NanoMB1Ref)
                            NanoMB1_Pixel(m_NanoMB1Ref, dc, m_fPixelSpacing, ↙
                                ↳ m_nMaxIter, bGlitch, antal, test1, test2, de, ↙
                                ↳ interior_checking);
                    if (antal > m_nMaxIter) antal = m_nMaxIter;
55
                    OutputIterationData(x, y, w, h, bGlitch, antal, test1, test2, de↙
```

```
                    ↳ );
                InterlockedIncrement((LPLONG)&m_nDone);
                OutputPixelData(x, y, w, h, bGlitch);
        }
}
```

92 fraktal_sft/nanomb1_reference.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "fraktal_sft.h"
#include "nanomb1.h"
#include "nanomb1.inc"
extern int64_t g_period;

void CFraktalSFT::CalculateReferenceNANOMB1()
{
        if (! (m_nFractalType == 0 && m_nPower == 2))
                return;

        Precision prec(m_rref.m_f.precision());
        complex<decNumber> c(m_rref.m_f, m_iref.m_f);
        int m = GetOrderM();
        int n = GetOrderN() / 2;
        int64_t period = g_period ? g_period : 1;
        double er2 = m_nBailout * m_nBailout;
        floatexp r0(m_fPixelSpacing * hypot(m_nX, m_nY));

        if (m_NanoMB1Ref)
        {
                delete m_NanoMB1Ref;
                m_NanoMB1Ref = nullptr;
        }
        m_NanoMB1Ref = NanoMB1_Reference_Calculate(c, m, n, period, m_nMaxIter, ↙
            ↳ r0, er2, GetGlitchLowTolerance(), m_bStop, m_nRDone);
}
```

93 fraktal_sft/nanomb2.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef NANOMB2_H
#define NANOMB2_H 1

#include "CDecNumber.h"
#include "complex.h"
#include "floatexp.h"

struct NanoMB2_Reference;

NanoMB2_Reference *NanoMB2_Reference_Calculate
  ( const complex<decNumber> &c
  , int64_t bm
  , int64_t bn
  , int64_t maxperiod
  , floatexp r0
  , volatile bool &stop
  , int64_t &m_nRDone
  , floatexp radius_scale
  );

void NanoMB2_Pixel
  ( const NanoMB2_Reference *ctx
  , complex<floatexp> dc
  , floatexp pixel_spacing
  , int64_t maxsi
  , int64_t maxiters
  , bool &bGlitch
  , int64_t &antal
  , double &test1
  , double &test2
  , double &de
  , bool interior_checking
  );

#endif
```

94 fraktal_sft/nanomb2.inc

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rumm
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

/*
Largely based on knighty's public domain nanomb1/nanomb2 code from:
<https://fractalforums.org/f/28/t/277/msg8132#msg8132>
<https://fractalforums.org/f/28/t/277/msg7983#msg7983>
*/

#include "nanomb2.h"
#include "Settings.h"
#define re m_r
#define im m_i
// type aliases
typedef int64_t N;
typedef floatexp R_lo;
typedef decNumber R_hi;
typedef complex<R_lo> C_lo;
typedef complex<R_hi> C_hi;

#define TABSIZ (MAX_APPROX_TERMS + 1)

//------------------------------------------------------------------------------------------------
    ↪
// Conversion routines
inline R_lo r_lo(const R_hi &z)
{
  return mpfr_get_fe(z.backend().data());
}

inline C_lo c_lo(const C_hi &z)
{
  return C_lo(r_lo(z.m_r), r_lo(z.m_i));
}

//------------------------------------------------------------------------------------------------
    ↪
class refClass{
        N m_n;
        std::vector<C_lo> m_v;
public:
```

```
        refClass(): m_n(0), m_v(0) {}
        void adde(C_lo c){ m_v.push_back(c); m_n++;}
        C_lo & operator[](N i){ return m_v[i % m_n];}
        const C_lo & operator[](N i) const { return m_v[i % m_n];}
};

class perturbationClassD{
        C_lo m_d0;
        C_lo m_d;
        C_lo m_dd;
        N    m_si;
        N    m_n0;
        bool m_escaped;
        int64_t m_iters;
  R_lo m_test1;
  R_lo m_test2;
        R_lo m_de;
public:

        perturbationClassD(C_lo d0, C_lo d, C_lo dd, N si, N n0): m_d0(d0), m_d(d), m_dd(dd), m_si(si), m_n0(n0), m_escaped(false), m_iters(0), m_test1(0), m_test2(0), m_de(0) {}

        //Experimental: Usually, after series approximation delta_0 (m_d0 here) have no effect.
        //              Here we do a first loop where we add delta_0 then, when delta_0 is too small,
        //             a second loop is performed where delta_0 is not used anymore. (idea from Pauldelbrot)
        // It works! ToDo: ...
        void run1(const refClass &ref, N maxsi, N maxiter, R_lo pixel_spacing){
                N si = m_si, i = m_n0;
                for( ; si < maxsi && i < maxiter; si++, i++){
                        C_lo zn(ref[i]);
                        if(abs(m_d0.re)+abs(m_d0.im) < 1e-20 * (abs(zn.re)+abs(zn.im))) break;
                        m_dd = R_lo(2) * m_dd * (m_d + zn) + R_lo(1);
                        m_d = m_d * (R_lo(2) * zn + m_d) + m_d0;
                        R_lo r(norm(zn + m_d));
        m_test2 = m_test1;
        m_test1 = r;
                        if (r > R_lo(1e10)){
                                m_escaped = true;
                                m_iters = i;
                                m_de = sqrt(r) * log(r) / abs(m_dd * pixel_spacing);
                                return;
                        }
                }
                for( ; si < maxsi && i < maxiter; si++, i++){
                        C_lo zn(ref[i]);
                        m_dd = R_lo(2) * m_dd * (m_d + zn);// + R_lo(1);
                        m_d = m_d * (R_lo(2) * zn + m_d);// + m_d0;
                        R_lo r(norm(zn + m_d));
        m_test2 = m_test1;
        m_test1 = r;
                        if (r > R_lo(1e10)){
```

```
                                m_escaped = true;
                                m_iters = i;
                                m_de = sqrt(r) * log(r) / abs(m_dd * ↙
                                    ↳ pixel_spacing);
                                return;
                        }
                }
        }
        void run(const refClass &ref, N maxsi, N maxiter, R_lo pixel_spacing){
                for(N si = m_si, i = m_n0 ; si < maxsi && i < maxiter; si++, i↙
                    ↳ ++){
                        C_lo zn(ref[i]);
                        m_dd = R_lo(2) * m_dd * (m_d + zn) + R_lo(1);
                        m_d = m_d * (R_lo(2) * zn + m_d) + m_d0;
                        R_lo r(norm(zn + m_d));
        m_test2 = m_test1;
        m_test1 = r;
                        if (r > R_lo(1e10)){
                                m_escaped = true;
                                m_iters = i;
                                m_de = sqrt(r) * log(r) / abs(m_dd * ↙
                                    ↳ pixel_spacing);
                                return;
                        }
                }
        }
        bool haveEscaped(){return m_escaped;}
        int64_t getIters(){return m_iters;}
        R_lo getTest1(){return m_test1;}
        R_lo getTest2(){return m_test2;}
        R_lo getDE(){return m_de;}
};

//------------------------------------------------------------------------------------------↙
    ↳
//Polynomial class
//------------------------------------------------------------------------------------------↙
    ↳
class polyClass{
        N m_m; //highest exponent
        C_lo tab[TABSIZ];//to store the coefficients. It is more than big ↙
            ↳ enought.
public:
        //Constructor
        polyClass(N m): m_m(m) {
                for(N l=0; l <= m_m; l++)
                        tab[l] = C_lo(0);
        }
        //Copy constructor
        polyClass(const polyClass& p): m_m(p.m_m) {
                for(N l=0; l <= m_m; l++)
                        tab[l] = p.tab[l];
        }

        //evaluation function. It would be nice to add an ()-operator.
        C_lo eval(C_lo u) const{
                C_lo r(0);
```

```
                C_lo ui(1);
                for(N i=0; i <= m_m; i++){
                        r += tab[i] * ui;
                        ui *= u;
                }
                return r;
        }

        //evaluat derivative.
        C_lo evalD(C_lo u) const{
                C_lo r(0);
                C_lo ui(1);
                for(N i=1; i <= m_m; i++){
                        r += R_lo(i) * tab[i] * ui;
                        ui *= u;
                }
                return r;
        }

        //Gives the nearest root to the 0. To use if and when applicable (that ↙
            ↳ is the reference is near 0... atom domain thing!)
        //Newton should do the job (otherwise IA-Newton ?).
        C_lo getRoot() const{
                C_lo rt(0);
                //R_lo t = abs(eval(rt));
                for(N i=0; i<30; i++){
                        C_lo num = eval(rt);
                        C_lo den = evalD(rt);
                        C_lo delta = num / den;
                        num = rt;
                        rt -= delta;
                        if( rt.re == num.re && rt.im == num.im) break;
                }
                return rt;
        }

        //Shift the polynomial to the given (relative) origin point
        //given a polynomial p(z) this function returns the polynomial q(z)==p(z↙
            ↳ +shift)
        polyClass shift(C_lo shift) const{
                polyClass res(m_m);
                R_lo bino[TABSIZ];//vector<R_lo> bino(m_m + 2);//bino could be ↙
                    ↳ precomuted.
                for(int i=0;i<=m_m;i++)
                        bino[i]=1;
                for(int i=0; i<=m_m; i++){
                        C_lo v(0);
                        for(int j=m_m-i; j>=0; j--)
                                v = tab[j+i] * bino[j] + shift * v;
                        for(int j=1; j<=m_m; j++)
                                bino[j] += bino[j-1];
                        res.tab[i] = v;
                }
                return res;
        }

        friend class biPolyClass;
```

```
};

//------------------------------------------------------------------------------------------------↙
    ↳
//Bivariate polynomial class. Used for SSA.
//------------------------------------------------------------------------------------------------↙
    ↳
class biPolyClass {
        N    m_m, m_n;
        C_lo tab[TABSIZ][TABSIZ];
        C_lo ttab[TABSIZ][TABSIZ];
        C_lo m_shift;
        N    m_period;
        R_lo m_escRadius;

        //copy tab into ttab. used for the squaring.
        void mcopy(){
                for(N l=0; l <= m_m; l++)
                        for(N c=0; c<= m_n; c++)
                                ttab[l][c] = tab[l][c];
        }

        //for the squaring operation: ttab will be the square of tab
        C_lo csqrc(N k, N l) const{
                C_lo v(0);
                for(N i=0; i <= k; i++)
                        for(N j=0; j <= l; j++)
                                v += ttab[i][j] * ttab[k-i][l-j];
                return v;
        }
public:
        //Constructor
        biPolyClass(N m, N n): m_m(m), m_n(n), m_shift(0), m_period(0), ↙
            ↳ m_escRadius(0) {
                for(N l=0; l <= m_m; l++)
                        for(N c=0; c<= m_n; c++)
                                tab[l][c] = C_lo(0);
                tab[1][0] = C_lo(1);
        }

        C_lo getShift()const{ return m_shift;}
        N    getPeriod()const{ return m_period;}
        void setPeriod(N period){ m_period = period;}
        void setEscRadius(R_lo multiplier = R_lo(.1)){
                m_escRadius = multiplier * getRadius();
        }
        R_lo getEscRadius()const{ return m_escRadius;}

        //squaring
        void sqr(){
                mcopy();
                for(N i=0; i <= m_m; i++)
                        for(N j=0; j <= m_n; j++)
                                tab[i][j] = csqrc(i,j);
        }

        //Apply one Mandelbrot iteration
```

```
265         void cstep(C_lo z){
                sqr();
                tab[0][0]  = z;
                tab[0][1] += C_lo(1);
            }
270
            //Evaluate the SSA polynomial
            C_lo eval(C_lo u, C_lo v)const{
                //v -= m_shift;//Take into account the shift
                C_lo r(0);
275             C_lo ui(1);
                for(N i=0; i <= m_m; i+=2){
                        C_lo vj(ui);
                        for(N j=0; j <= m_n; j++){
                                r += tab[i][j] * vj;
280                             vj *= v;
                        }
                        ui *= u*u;
                }
                return r;
285         }

            //Evaluate the derivative wrt c
            C_lo eval_dc(C_lo u, C_lo v)const{
                //v -= m_shift;
290             C_lo r(0);
                C_lo ui(1);
                for(N i=0; i <= m_m; i+=2){
                        C_lo vj(ui);
                        for(N j=1; j <= m_n; j++){
295                             r += R_lo(j) * tab[i][j] * vj;
                                vj *= v;
                        }
                        ui *= u*u;
                }
300             return r;
            }

            //Evaluate the derivative wrt z
            C_lo eval_dz(C_lo u, C_lo v)const{
305             //v -= m_shift;
                C_lo r(0);
                C_lo ui(u);
                for(N i=2; i <= m_m; i+=2){
                        C_lo vj(C_lo(i) * ui);
310                     for(N j=0; j <= m_n; j++){
                                r += tab[i][j] * vj;
                                vj *= v;
                        }
                        ui *= u*u;
315             }
                return r;
            }

            //gives the escape radius for this SSA.
320         //based on polynomial roots properties: Rouché theorem.
            //ToDo: Document the method by which we find the radius.
```

```
        R_lo getRadius()const{
                if(abs(tab[0][2]) < R_lo(1.0, -1100100100)) return 1e30;
#if 0
                return abs(tab[0][1])/abs(tab[0][2]);//Is actually accurate ↙
                    ↳ enough
#else
                R_lo r(0);
                for(int i = 0; i < m_n * m_n; i++){
                        C_lo den(0);
                        R_lo rr(1);
                        for(int j = 2; j <=m_n; j++){
                                den += tab[0][j] * rr;
                                rr  *= r;
                        }
                        r = abs(tab[0][1])/abs(den);
                }
std::cerr << "period " << m_period << " radius " << r.toString() << std::endl;
                return r;
#endif
        }
#if 0
        //version that gives an estimate of the bail out radius in dynamic plane↙
            ↳  depending on v.
        R_lo getRadius(C_lo v)const{
                C_lo t2(0), t4(0), vi(1);
                for(N i = 0; i <= m_n; i++){
                        t2 += tab[2][i] * vi;
                        t4 += tab[4][i] * vi;
                        vi *= v;
                }
                return abs(t2)/abs(R_lo(2)*t4);//Is actually accurate enough...
        }
#endif
        //Just give the part that depends on v only. It can be used as the ↙
            ↳ classical series approximation.
        polyClass getSAPoly()const{
                polyClass SApol(m_n);
                for(N i=0; i<=m_n; i++)
                        SApol.tab[i] = tab[0][i];
                return SApol;
        }

        //Partially evaluate the biPoly for the given v value.
        polyClass getEvalPoly(C_lo v)const{
                polyClass SApol(m_m);
                //v -= m_shift;
                for(N i=0; i<=m_m; i++){
                        C_lo vj(1);
                        C_lo r(0);
                        for(N j=0; j <= m_n; j++){
                                r  += tab[i][j] * vj;
                                vj *= v;
                        }
                        SApol.tab[i] = r;
                }
                return SApol;
        }
```

```
        //Partially evaluate the biPoly for the given v value. This is for ↙
            ↳ derivative evaluation.
        polyClass getEvalPoly_dz(C_lo v)const{
                polyClass SApol = getEvalPoly(v);
380             //v -= m_shift;
                for(N i=0; i<m_m; i++){
                        SApol.tab[i] = SApol.tab[i+1] * R_lo(i+1);
                }
                SApol.m_m -= 1;
385             return SApol;
        }

        //Partially evaluate the biPoly for the given v value. This is for ↙
            ↳ derivative evaluation.
        //To implement
390     polyClass getEvalPoly_dc(C_lo v)const{
                polyClass SApol(m_m);
                //v -= m_shift;
                for(N i=0; i<=m_m; i++){
                        C_lo vi(1);
395                     C_lo r(0);
                        for(N j=1; j<=m_n; j++){
                                r  += R_lo(j) * vi * tab[i][j];
                                vi *= v;
                        }
400                     SApol.tab[i] = r;
                }
                return SApol;
        }

405     //gives a shifted biPoly.We shift only wrt v param therefore we will use↙
            ↳  the shift function of polyClass.
        biPolyClass getShiftedBP(C_lo sv)const{
                biPolyClass SBP(m_m,m_n);
                for(N i=0; i<=m_m; i++){
                        polyClass pol(m_n);
410                     for(N j=0; j<=m_n; j++)
                                pol.tab[j] = tab[i][j];
                        polyClass popol = pol.shift(sv);
                        for(N j=0; j<=m_n; j++)
                                SBP.tab[i][j] = popol.tab[j];
415             }
                SBP.m_shift = m_shift + sv;//ToDo: verify if itis necessary to ↙
                    ↳ do add m_shift . This function is meant to be called only ↙
                    ↳ for the main SSA where shift==0. Just in case...
                return SBP;
        }
};
420
//-------------------------------------------------------------------------------------------
    ↳
// iterate using the given list of SSAs.
// d0 is relative to the center of the screen. The SSAs transparently transform ↙
    ↳ the point into their own reference frame when called thanks to their ↙
    ↳ m_shift member variable.
//-------------------------------------------------------------------------------------------
```

```
    ↪
425
static void iteratePtDE1( C_lo d0, const std::vector<biPolyClass> &SSAs, const ↙
    ↪ refClass &ref, N maxsi, N maxiter, R_lo pixel_spacing, int64_t &antal, ↙
    ↪ double &test1, double &test2, double &de, bool interior_checking){
        //try the SSAs from the latest (deepest) to the first. if not possible ↙
            ↪ use perturbation until we get near to 0 again then repeat.
        //ToDo: edit refClass to add another run(inside radius) function that ↙
            ↪ verify if the iterated point returns near 0
        //ToDo2: we need to test for escapes AFTER the iteration --> edit ↙
            ↪ biPolyClass to compute the "other" bailout.
430
        C_lo d(0);
        C_lo dd(0);
        N i(0);
        N si(0);
435     N nbrSSAs(SSAs.size());
        N curSSA(-1);

        //first iteration with SSA. It uses bailout radius in parameter space.
        for(N k = nbrSSAs - 1; k>=0; k--){
440             C_lo d00 = d0 - SSAs[k].getShift();//delta_c coordinates have de↙
                    ↪ be wrt current SSA
                if(abs(d00)<SSAs[k].getEscRadius()){
                        dd = dd * SSAs[k].eval_dz(d,d00) + SSAs[k].eval_dc(d,d00↙
                            ↪ );
                        d = SSAs[k].eval(d,d00);
                        i+= SSAs[k].getPeriod();
445                     si++;
                        curSSA = k;
                        break;
                }
        }
450     N firstSSA = curSSA;
        R_lo d_norm(1.0/0.0);
        //Use SSAs from lower to upper to iterate
        for(; curSSA>=0 && SSAs[curSSA].getPeriod() >= 1; curSSA--){
                C_lo d00 = d0 - SSAs[curSSA].getShift();//delta_c coordinates ↙
                    ↪ have de be wrt current SSA
455             while(si < maxsi && i<maxiter && norm(d)<SSAs[curSSA].↙
                    ↪ getEscRadius()) {// norm(d)<SSAs[curSSA].getEscRadius() ↙
                    ↪ <=> abs(d)<sqrt(SSAs[curSSA].getEscRadius())
                if (interior_checking)
                {
                        if (curSSA == firstSSA && norm(d) < d_norm)
                        {
460                       d_norm = norm(d);
                          // check interior: newton iterations for w0 = f^(si*p)↙
                              ↪ (w0, d0)
                          C_lo w0(d);
                          C_lo dw;
                          bool converged = false;//------------------
465                       for (int step = 0; step < 16; ++step)
                          {
                            C_lo w(w0);
                            dw = C_lo(1);
                            for (int64_t n = 0; n < si; ++n)
```

```
                                {
                                  dw = dw * SSAs[curSSA].eval_dz(w, d00);
                                  w = SSAs[curSSA].eval(w, d00);
                                }
                                C_lo w0_next = w0 - (w - w0) / (dw - C_lo(1));
                                R_lo delta = norm(w0_next - w0);
                                //C_lo delta = (w0_next - w0);//-----------------
                                w0 = w0_next;
                                R_lo epsilon2(0); // FIXME
                                if (delta <= epsilon2) { converged = true;break; }; // converged
                                //if (abs(delta.re) < abs(w0.re) * R_lo(1e-6) && abs(delta.im) < abs(w0.im) * R_lo(1e-6)) { converged = true; break;}//-----------------------
                              }
                              if (converged && norm(dw) < R_lo(1))
                              {
            antal = maxiter;
            test1 = 0;
            test2 = 0;
            de = 0;
            return;
    #if 0
                                // is interior, calculate interior DE
                                C_lo z(w0), dz(1), dc(0), dzdz(0), dcdz(0);
                                for (int64_t n = 0; n < si; ++n)
                                  up.eval(z, dz, dc, dzdz, dcdz);
                                R_lo de = (R_lo(1) - norm(dz)) / abs(dcdz + (dzdz * dc) / (C_lo(1) - dz));
                                iters = si * period;
                                trans = 0.0f;
                                return -de;
    #endif
                              }
                            }
                  }
                            dd = dd * SSAs[curSSA].eval_dz(d,d00) + SSAs[curSSA].eval_dc(d,d00);
                            d  = SSAs[curSSA].eval(d,d00);
                            i += SSAs[curSSA].getPeriod();
                            si++;
                  }
                  if(si >= maxsi || i >= maxiter) break;
          }

          //We are out of SSAs or maxiter attained
          //R_lo f(1);
          //if(i == 0) {dd = C_lo(1); f = R_lo(.5);}
          if(si >= maxsi || i>=maxiter)
          {
            antal = maxiter;
            test1 = 0;
      test2 = 0;
      de = 0;
            return;
          }
```

```
        //Use perturbation to finish the job.
        //ToInvestigate: Usually when arriving here (same thing with SA) delta_c
            ↳  (d0 here) is too small wrt d.
        //                One can use perturbation without adding delta_c. Some
            ↳ other optimizations may become possible. (from comment by
            ↳ Pauldelbrot @fractalforums.com)
525     d -= ref[i]; //put d in reference orbit's local frame
        perturbationClassD p(d0,d,dd,si,i);//We use original d0 because the
            ↳ reference orbit was computed there.
        p.run(ref,maxsi,maxiter, pixel_spacing);

        if(p.haveEscaped())
530     {
          antal = p.getIters();
          test1 = p.getTest1().todouble();
          test2 = p.getTest1().todouble();
      de = p.getDE().todouble();
535   return;
        }
        else
        {
          antal = maxiter;
540       test1 = 0;
      test2 = 0;
      de = 0;
      return;
        }
545 }

//-------------------------------------------------------------------------------------------
    ↳
// entry point

550 struct NanoMB2_Reference
{
  std::vector<biPolyClass> SSAs;
  refClass ref;
  NanoMB2_Reference(std::vector<biPolyClass> SSAs, refClass ref)
555   : SSAs(SSAs)
  , ref(ref)
  { };
};

560 NanoMB2_Reference *NanoMB2_Reference_Calculate(const C_hi &c, N bm, N bn, N
    ↳ maxperiod, R_lo r0, volatile bool &stop, int64_t &m_nRDone, R_lo
    ↳ radius_scale)
{
  R_lo tmax = r0;

  //Compute bivariate polynomial approximating period iterations
      ↳ -------------------------
565   //ToDo: compute also the bivariate polynomial for the "upper level" hyperbolic
      ↳  components centres (HCC):
  //   - Use the running bivariate polynomial... actually the part that doesn't
      ↳ depend on "u" (the SA part)
  //   - Find the upper HCCs by solving the zroes of the SA poly when applicable
```

```
    ↪ .
//   - We will not suppose that "c" is at the center of a minibrot anymore.
//   - The last (deepest) HC is the one which center is inside or near the ↙
    ↪ rendered area
//   This procedure may fail when the SA part of tha bipoly is no longer ↙
    ↪ accurate while we haven't found
//  an HCC inside the rendered area and that area is too small to be handled ↙
    ↪ by the previous HCC
//  (insufficient accuracy). Maybe a kind of "super" perturbation?
//    For now we will suppose we have a good deepest HCC.
//ToDo: what about the case where no deepest minibrot is found... We still ↙
    ↪ need good reference points... maybe subdivide the rendered area?
biPolyClass fp(bm,bn);
std::vector<biPolyClass> SSAs;//to store the set of the relevent hyperbolic ↙
    ↪ components' SSAs from lowest to highest period.
R_lo smallest_ref_rad_so_far(1000);//for atom domain
refClass ref;
C_hi z(0);
C_lo zlo(0);
bool bNotFinished = true;
bool stopRecSSAs = false;

for(N i = 0; i < maxperiod && bNotFinished && !stop; i++){
      ref.adde(zlo);
      z = z*z + c;
      zlo = c_lo(z);
      fp.cstep(zlo);
      //See if we have a new atom domain... Atom domain does work but not ver ↙
          ↪ well
      //R_lo eRad = fp.getRadius() * R_lo(1000.);//scale down in order to not ↙
          ↪ choose too far HC center. ToVerify... worse!
      //doing "if(zmag < /*eRad*/ R_lo(.5)*smallest_ref_rad_so_far)" is better↙
          ↪  (with the .5 factor which removes a lot of SSAs)
      R_lo zmag = abs(zlo);

      if(zmag < /*eRad*/ R_lo(.5)*smallest_ref_rad_so_far /*&& i>7*/ && !↙
          ↪ stopRecSSAs){
            //compute the SSA for current hyperbolic component and store it ↙
                ↪ in SSAs:

            //get the SA
            polyClass SA = fp.getSAPoly();

            //compute the corresponding HC center
            C_lo HCcentre = SA.getRoot();//HCcentre is relative to c.

            //get shifted SSA
            biPolyClass SSA = fp.getShiftedBP(HCcentre);
            SSA.setPeriod(i+1);
            SSA.setEscRadius(radius_scale);

            if(SSA.getEscRadius() > R_lo(0.1) * tmax){
                  //Store the shifted SSA into SSAs
                  //We don't need HC with too low period. They may be ↙
                      ↪ slower than using perturbation.
                  if(i>7)//-----------------------------------------
                        SSAs.push_back(SSA);
```

```
                        //if HCcentre is inside or near the rendered area we ↙
                            ↳ should stop!
615                     //--> We will consider that HCcenter is near when it's ↙
                            ↳ magnitude is less than say 1000th the rendered ↙
                            ↳ area width? to study!!!
                        //if(abs(HCcentre) < R_lo(100) * tmax) bNotFinished = ↙
                            ↳ false;

                    } else stopRecSSAs = true;
                    //
620                 smallest_ref_rad_so_far = zmag;
            }
      m_nRDone++;
      }
      if (stop)
625     return nullptr;
      else
        return new NanoMB2_Reference(SSAs, ref);
    }

630 void NanoMB2_Pixel(const NanoMB2_Reference *ctx, C_lo dc, R_lo pixel_spacing, N ↙
      ↳ maxsi, N maxiters, bool &bGlitch, int64_t &antal, double &test1, double &↙
      ↳ test2, double &de, bool interior_checking)
    {
      iteratePtDE1( dc, ctx->SSAs, ctx->ref, maxsi, maxiters, pixel_spacing, antal, ↙
          ↳ test1, test2, de, interior_checking);
      bGlitch = false;
    }
```

95 fraktal_sft/nanomb2_perturbation.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen
 5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "nanomb2.h"

    void CFraktalSFT::MandelCalcNANOMB2()
    {
25          bool interior_checking = GetInteriorChecking();
```

```
	m_bIterChanged = TRUE;
	int x, y, w, h;
	while (!m_bStop && m_P.GetPixel(x, y, w, h, m_bMirrored)){
		int nIndex = x * 3 + (m_bmi->biHeight - 1 - y)*m_row;
		if (m_nPixels[x][y] != PIXEL_UNEVALUATED){
			SetColor(nIndex, m_nPixels[x][y], m_nTrans[x][y], x, y, w, h);
			if (m_bMirrored)
				Mirror(x, y);
			continue;
		}
		if (GuessPixel(x, y, w, h))
			continue;

		floatexp D0r = 0;
		floatexp D0i = 0;
		floatexp daa = 1;
		floatexp dab = 0;
		floatexp dba = 0;
		floatexp dbb = 1;
		GetPixelCoordinates(x, y, D0r, D0i, daa, dab, dba, dbb);

		complex<floatexp> dc(D0r, D0i);
		bool bGlitch = false;
		int64_t antal = 0;
		double test1 = 0, test2 = 0, de = 0;

		int64_t maxsi = m_nMaxIter; // FIXME
		if (m_NanoMB2Ref)
			NanoMB2_Pixel(m_NanoMB2Ref, dc, m_fPixelSpacing, maxsi, m_nMaxIter, bGlitch, antal, test1, test2, de, interior_checking);
		if (antal > m_nMaxIter) antal = m_nMaxIter;

		OutputIterationData(x, y, w, h, bGlitch, antal, test1, test2, de);
		InterlockedIncrement((LPLONG)&m_nDone);
		OutputPixelData(x, y, w, h, bGlitch);
	}
}
```

96 fraktal_sft/nanomb2_reference.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
```

```
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "nanomb2.h"
    #include "nanomb2.inc"
    extern int64_t g_period;

25  void CFraktalSFT::CalculateReferenceNANOMB2()
    {
            if (!(m_nFractalType == 0 && m_nPower == 2))
                    return;

30          Precision prec(m_rref.m_f.precision());
            complex<decNumber> c(m_rref.m_f, m_iref.m_f);
            int m = GetOrderM();
            int n = GetOrderN();
            int64_t maxperiod = g_period ? g_period : m_nMaxIter;
35          floatexp r0(m_fPixelSpacing * hypot(m_nX, m_nY));
            floatexp radius_scale(GetRadiusScale());

            if (m_NanoMB2Ref)
            {
40                  delete m_NanoMB2Ref;
                    m_NanoMB2Ref = nullptr;
            }
            m_NanoMB2Ref = NanoMB2_Reference_Calculate(c, m, n, maxperiod, r0, ↙
                ↳ m_bStop, m_nRDone, radius_scale);
    }
```

97 fraktal_sft/newton.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen
 5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include <windows.h>
    #include <stdio.h>
    #include <math.h>
    #include <malloc.h>
```

```
#include <time.h>
#include <float.h>
#include "resource.h"
#include "complex.h"
#include "fraktal_sft.h"
#include "CDecNumber.h"
#include "main.h"
#include "../common/barrier.h"
#include "../common/StringVector.h"
#include "newton.h"
#include <iostream>
#include <string>

#define KF_MAIN 1
#include "../formula/generated/formula.h"
#undef abs
#undef sgn

static const struct formula *get_formula(int type, int power)
{
  std::string name = "formula_" + std::to_string(type) + "_" + std::to_string(power);
  const struct formula *f = (const struct formula *) GetProcAddress(GetModuleHandle(nullptr), name.c_str());
  if (f) return f;
  name = "formula_" + std::to_string(type);
  f = (const struct formula *) GetProcAddress(GetModuleHandle(nullptr), name.c_str());
  return f;
}

extern CFraktalSFT g_SFT;
extern HICON g_hIcon;

static volatile int running = 0;
bool g_bNewtonRunning = false;
bool g_bNewtonStop = false;
static bool g_bNewtonExit = false;
bool g_bJustDidNewton = false;
bool g_bUseBallPeriod = false;

static floatexp g_fNewtonDelta2[2];
static int g_nNewtonETA = 0;

static int newtonETA(const floatexp &delta0, const floatexp &delta1, const floatexp &epsilon)
{
  floatexp l0 = log(delta0);
  floatexp l1 = log(delta1);
  floatexp e  = log(epsilon);
  // (e - l0) = 1 (l1 - l0) + 2 (l1 - l0) + 4 (l1 - l0) + ... 2^N (l1 - l0)
  // (e - l0) / (l1 - l0) = sum_0^N 2^i
  floatexp N = log2((e - l0) / (l1 - l0));
  return double(N);
}

static std::string g_szRe;
```

```
static std::string g_szIm;
static std::string g_szZoom;
static char g_szProgress[128];
static int g_nMinibrotPos=0;
static double g_nMinibrotFactor=1.0;
static std::string g_sMinibrotSourceZoom;

#define flyttyp CDecNumber

const complex<flyttyp> _2(2,0);
const complex<flyttyp> _1(1,0);

static inline int sgn(const flyttyp &z) {
  if (z > 0) { return  1; }
  if (z < 0) { return -1; }
  return 0;
}

static inline bool odd(int a) {
  return a & 1;
}

static inline floatexp cabs2(const complex<floatexp> &z) {
  return z.m_r * z.m_r + z.m_i * z.m_i;
}
static inline flyttyp cabs2(const complex<flyttyp> &z) {
  return z.m_r * z.m_r + z.m_i * z.m_i;
}
static inline bool isfinite(const flyttyp &a){
        (void) a;
        return true;
/*      if (a <= DBL_MAX && a >= -DBL_MAX)
                return true;
        return false;*/
}
static inline bool cisfinite(const complex<flyttyp> &z) {
  return isfinite(z.m_r) && isfinite(z.m_i);
}

struct BallPeriodCommon
{
  HWND hWnd;
  barrier *barrier;
  volatile bool *stop;
  bool *haveperiod;
  int64_t *period;
  int64_t maxperiod;
  mpfr_t cr, ci, zr, zi, zr2, zi2, zri, t;
  floatexp zradius;
};

struct BallPeriod
{
  int threadid;
  HANDLE hDone;
  BallPeriodCommon *c;
};
```

```
135 static DWORD WINAPI ThBallPeriod(BallPeriod *b)
    {
      SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
      SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
      int t = b->threadid;
140   barrier *barrier = b->c->barrier;
      volatile bool *stop = b->c->stop;
      bool *haveperiod = b->c->haveperiod;
      int64_t *period = b->c->period;
      char szStatus[300];
145   floatexp r = b->c->zradius;
      complex<floatexp> z(0.0, 0.0);
      complex<floatexp> dz(0.0, 0.0);
      floatexp rz = 0.0;
      floatexp rdz = 0.0;
150   floatexp Ei = 0.0;
      floatexp rr = 0.0;
      uint32_t last = 0;
      if (t == 0)
      {
155     wsprintf(szStatus,"Finding period, 0...");
        SetDlgItemText(b->c->hWnd,IDC_EDIT1,szStatus);
        last = GetTickCount();
      }
      for (int64_t i = 1; i < b->c->maxperiod; ++i)
160   {
        if (t == 0)
        {
          if(i%100==0){
            uint32_t now = GetTickCount();
165         if (now - last > 250){
              snprintf(szStatus,200,"Finding period, %lld...",i);
              SetDlgItemText(b->c->hWnd,IDC_EDIT1,szStatus);
              last = now;
            }
170       }
          Ei = rdz * rdz + (2 * rz + r * (2 * rdz + r * Ei)) * Ei;
          dz = 2 * z * dz + complex<floatexp>(1.0, 0.0);
          z = complex<floatexp>(mpfr_get_fe(b->c->zr), mpfr_get_fe(b->c->zi));
          rdz = abs(dz);
175       rz = abs(z);
          rr = r * (rdz + r * Ei);
          if (rz - rr > 2) // outside everything
          {
            *haveperiod = true;
180         *period = 0;
          }
          if (rz - rr <= 0) // surrounds 0
          {
            *haveperiod = true;
185         *period = i;
          }
        }
        switch (t)
        {
190       case 0:
```

```
        {
          mpfr_sqr(b->c->zr2, b->c->zr, MPFR_RNDN);
          mpfr_sqr(b->c->zi2, b->c->zi, MPFR_RNDN);
          break;
        }
        case 1:
        {
          mpfr_mul(b->c->zri, b->c->zr, b->c->zi, MPFR_RNDN);
          break;
        }
      }
      if (barrier->wait(stop)) break;
      if (*haveperiod) break;
      switch (t)
      {
        case 0:
          mpfr_sub(b->c->t, b->c->zr2, b->c->zi2, MPFR_RNDN);
          mpfr_add(b->c->zr, b->c->t, b->c->cr, MPFR_RNDN);
          break;
        case 1:
          mpfr_mul_2ui(b->c->zri, b->c->zri, 1, MPFR_RNDN);
          mpfr_add(b->c->zi, b->c->zri, b->c->ci, MPFR_RNDN);
          break;
      }
      if (barrier->wait(stop)) break;
    }
    if (t == 0)
    {
      if (*period == 0)
      {
        *haveperiod = false;
      }
    }
    SetEvent(b->hDone);
    mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
    return 0;
  }

  static int64_t ball_period_do(const complex<flyttyp> &center, flyttyp radius, ↙
      ↳ int64_t maxperiod,int &steps,HWND hWnd) {
    radius = flyttyp(4)/radius;
    mp_bitcnt_t bits = mpfr_get_prec(center.m_r.m_dec.backend().data());
    barrier bar(2);
    bool haveperiod = false;
    int64_t period = 0;
    HANDLE hDone[2];
    // prepare threads
    BallPeriod ball[2];
    BallPeriodCommon c;
    c.maxperiod = maxperiod;
    c.hWnd = hWnd;
    c.barrier = &bar;
    c.stop = &g_bNewtonStop;
    c.haveperiod = &haveperiod;
    c.period = &period;
    c.maxperiod = maxperiod;
    mpfr_init2(c.cr, bits); mpfr_set(c.cr, center.m_r.m_dec.backend().data(), ↙
```

```
      ↪ MPFR_RNDN);
    mpfr_init2(c.ci, bits); mpfr_set(c.ci, center.m_i.m_dec.backend().data(), ↙
      ↪ MPFR_RNDN);
    mpfr_init2(c.zr, bits); mpfr_set(c.zr, center.m_r.m_dec.backend().data(), ↙
      ↪ MPFR_RNDN);
    mpfr_init2(c.zi, bits); mpfr_set(c.zi, center.m_i.m_dec.backend().data(), ↙
      ↪ MPFR_RNDN);
    mpfr_init2(c.zr2, bits);
    mpfr_init2(c.zi2, bits);
    mpfr_init2(c.zri, bits);
    mpfr_init2(c.t, bits);
    c.zradius = mpfr_get_fe(radius.m_dec.backend().data());
    for (int t = 0; t < 2; ++t)
    {
      ball[t].threadid = t;
      ball[t].hDone = hDone[t] = CreateEvent(NULL, 0, 0, NULL);
      ball[t].c = &c;
    }
    // spawn threads
    for (int i = 0; i < 2; i++)
    {
      DWORD dw;
      HANDLE hThread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ThBallPeriod, ↙
        ↪ (LPVOID)&ball[i], 0, &dw);
      CloseHandle(hThread);
    }
    // wait for threads to complete
    WaitForMultipleObjects(2, hDone, TRUE, INFINITE);
    for (int i = 0; i < 2; i++)
    {
      CloseHandle(hDone[i]);
    }
    mpfr_clear(c.cr);
    mpfr_clear(c.ci);
    mpfr_clear(c.zr);
    mpfr_clear(c.zi);
    mpfr_clear(c.zr2);
    mpfr_clear(c.zi2);
    mpfr_clear(c.zri);
    mpfr_clear(c.t);
    steps = period;
    return haveperiod ? period : 0;
  }

  #if 0
  struct BoxPeriod
  {
    int threadid;
    barrier *barrier;
    volatile bool *stop;
    int maxperiod;
    mpfr_t zr, zi, zr2, zi2, zri, cr, ci, t, *z2r, *z2i;
    int *crossing[4];
    int *haveperiod;
    int *period;
    HANDLE hDone;
    HWND hWnd;
```

```
};
static DWORD WINAPI ThBoxPeriod(BoxPeriod *b)
{
  SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
  SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
  int t = b->threadid;
  barrier *barrier = b->barrier;
  volatile bool *stop = b->stop;
  int *haveperiod = b->haveperiod;
  int *period = b->period;
  char szStatus[300];
  uint32_t last = 0;
  if (t == 0)
  {
    wsprintf(szStatus,"Finding period, 0...");
    SetDlgItemText(b->hWnd,IDC_EDIT1,szStatus);
    last = GetTickCount();
  }
  for (int i = 1; i < b->maxperiod; ++i)
  {
    // *b->crossing[t] = crosses_positive_real_axis(&b->zr, &b->zi, b->z2r, b->z2i);
    bool crossing = false;
    if (mpfr_sgn(b->zi) != mpfr_sgn(*b->z2i))
    {
      // d = b - a;
      mpfr_sub(b->zr2, *b->z2r, b->zr, MPFR_RNDN);
      mpfr_sub(b->zi2, *b->z2i, b->zi, MPFR_RNDN);
      int fs = mpfr_sgn(b->zi2);
      // t = cross(d, a);
      mpfr_mul(b->zri, b->zi2, b->zr, MPFR_RNDN);
      mpfr_mul(b->zi2, b->zr2, b->zi, MPFR_RNDN);
      mpfr_sub(b->t, b->zri, b->zi2, MPFR_RNDN);
      int ft = mpfr_sgn(b->t);
      crossing = fs == ft;
    }
    *b->crossing[t] = crossing;
    if (barrier->wait(stop)) break;
    if (t == 0)
    {
      if(i%100==0){
        uint32_t now = GetTickCount();
        if (now - last > 250){
          wsprintf(szStatus,"Finding period, %d...",i);
          SetDlgItemText(b->hWnd,IDC_EDIT1,szStatus);
          last = now;
        }
      }
      int surround = 0;
      for (int s = 0; s < 4; ++s)
      {
        surround += *b->crossing[s];
      }
      if (surround & 1)
      {
        *haveperiod = true;
        *period = i;
```

```
355         }
        }
        // z = z * z + c
        mpfr_sqr(b->zr2, b->zr, MPFR_RNDN);
        mpfr_sqr(b->zi2, b->zi, MPFR_RNDN);
360     mpfr_mul(b->zri, b->zr, b->zi, MPFR_RNDN);
        mpfr_sub(b->t, b->zr2, b->zi2, MPFR_RNDN);
        mpfr_add(b->zr, b->t, b->cr, MPFR_RNDN);
        mpfr_mul_2ui(b->zri, b->zri, 1, MPFR_RNDN);
        mpfr_add(b->zi, b->zri, b->ci, MPFR_RNDN);
365     if (barrier->wait(stop)) break;
        if (*haveperiod) break;
      }
      SetEvent(b->hDone);
      mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
370   return 0;
    }

    static int m_d_box_period_do(const complex<flyttyp> &center, flyttyp radius, int↙
        ↪ maxperiod,int &steps,HWND hWnd) {
          radius = flyttyp(4)/radius;
375
      mp_bitcnt_t bits = mpfr_get_prec(center.m_r.m_dec.backend().data());
      barrier bar(4);
      complex<flyttyp> c[4];
      c[0] = center + complex<flyttyp>(-radius, -radius);
380   c[1] = center + complex<flyttyp>(radius, -radius);
      c[2] = center + complex<flyttyp>(radius, radius);
      c[3] = center + complex<flyttyp>(-radius, radius);
      int crossing[4] = { 0, 0, 0, 0 };
      int haveperiod = false;
385   int period = 0;
      HANDLE hDone[4];
      // prepare threads
      BoxPeriod box[4];
      for (int t = 0; t < 4; ++t)
390   {
        box[t].threadid = t;
        box[t].barrier = &bar;
        box[t].maxperiod = maxperiod;
        mpfr_init2(box[t].cr, bits); mpfr_set(box[t].cr, c[t].m_r.m_dec.backend().↙
            ↪ data(), MPFR_RNDN);
395     mpfr_init2(box[t].ci, bits); mpfr_set(box[t].ci, c[t].m_i.m_dec.backend().↙
            ↪ data(), MPFR_RNDN);
        mpfr_init2(box[t].zr, bits); mpfr_set(box[t].zr, c[t].m_r.m_dec.backend().↙
            ↪ data(), MPFR_RNDN);
        mpfr_init2(box[t].zi, bits); mpfr_set(box[t].zi, c[t].m_i.m_dec.backend().↙
            ↪ data(), MPFR_RNDN);
        mpfr_init2(box[t].zr2, bits);
        mpfr_init2(box[t].zi2, bits);
400     mpfr_init2(box[t].zri, bits);
        mpfr_init2(box[t].t, bits);
        box[t].z2r = &box[(t + 1) % 4].zr;
        box[t].z2i = &box[(t + 1) % 4].zi;
        box[t].stop = &g_bNewtonStop;
405     box[t].crossing[0] = &crossing[0];
        box[t].crossing[1] = &crossing[1];
```

```
      box[t].crossing[2] = &crossing[2];
      box[t].crossing[3] = &crossing[3];
      box[t].haveperiod = &haveperiod;
      box[t].period = &period;
      box[t].hDone = hDone[t] = CreateEvent(NULL, 0, 0, NULL);
      box[t].hWnd = hWnd;
    }
    // spawn threads
    for (int i = 0; i < 4; i++)
    {
      DWORD dw;
      HANDLE hThread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ThBoxPeriod, ↙
          ↳ (LPVOID)&box[i], 0, &dw);
      CloseHandle(hThread);
    }
    // wait for threads to complete
    WaitForMultipleObjects(4, hDone, TRUE, INFINITE);
    for (int i = 0; i < 4; i++)
    {
      CloseHandle(hDone[i]);
      mpfr_clear(box[i].cr);
      mpfr_clear(box[i].ci);
      mpfr_clear(box[i].zr);
      mpfr_clear(box[i].zi);
      mpfr_clear(box[i].zr2);
      mpfr_clear(box[i].zi2);
      mpfr_clear(box[i].zri);
      mpfr_clear(box[i].t);
    }
    steps = period;
    return haveperiod ? period : 0;

  }
  #endif

  struct STEP_STRUCT_COMMON
  {
          HWND hWnd;
          barrier *barrier;
          volatile bool *stop;
          int newtonStep;
          int64_t period;
          mpfr_t zr, zi, zrn, zin, zr2, zi2, cr, ci, dcr, dci, dcrn, dcin, dcrzr, ↙
              ↳ dcrzi, dcizr, dcizi;
  };
  struct STEP_STRUCT
  {
          int nType;
          HANDLE hDone;
          STEP_STRUCT_COMMON *common;
  };
  static DWORD WINAPI ThStep(STEP_STRUCT *t0)
  {
    SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
    SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
    struct STEP_STRUCT_COMMON *t = t0->common;
    char szStatus[256];
```

```
    uint32_t last = GetTickCount();
    switch (t0->nType)
    {
465   case 0: // zr
        for (int64_t i = 0; i < t->period; ++i)
        {
          mpfr_sqr(t->zr2, t->zr, MPFR_RNDN);
          mpfr_sqr(t->zi2, t->zi, MPFR_RNDN);
470       mpfr_sub(t->zrn, t->zr2, t->zi2, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
          mpfr_add(t->zr, t->zrn, t->cr, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
        }
475     break;
      case 1: // zi
        for (int64_t i = 0; i < t->period; ++i)
        {
          if(i%100==0){
480               uint32_t now = GetTickCount();
                  if (now - last > 250)
                  {
                          wsprintf(szStatus,"NR %d/%d (%d%%) %s",t->newtonStep + ↙
                              ↳ 1,g_nNewtonETA ? t->newtonStep + g_nNewtonETA : 0,↙
                              ↳ i*100/t->period,g_szProgress);
                          SetDlgItemText(t->hWnd,IDC_EDIT1,szStatus);
485                       last = now;
                  }
          }
          mpfr_mul(t->zin, t->zr, t->zi, MPFR_RNDN);
          mpfr_mul_2ui(t->zin, t->zin, 1, MPFR_RNDN);
490       if (t->barrier->wait(t->stop)) break;
          mpfr_add(t->zi, t->zin, t->ci, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
        }
        break;
495   case 2: // dcr
        for (int64_t i = 0; i < t->period; ++i)
        {
          mpfr_mul(t->dcrzr, t->dcr, t->zr, MPFR_RNDN);
          mpfr_mul(t->dcizi, t->dci, t->zi, MPFR_RNDN);
500       mpfr_sub(t->dcrn, t->dcrzr, t->dcizi, MPFR_RNDN);
          mpfr_mul_2ui(t->dcrn, t->dcrn, 1, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
          mpfr_add_ui(t->dcr, t->dcrn, 1, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
505     }
        break;
      case 3: // dci
        for (int64_t i = 0; i < t->period; ++i)
        {
510       mpfr_mul(t->dcrzi, t->dcr, t->zi, MPFR_RNDN);
          mpfr_mul(t->dcizr, t->dci, t->zr, MPFR_RNDN);
          mpfr_add(t->dcin, t->dcrzi, t->dcizr, MPFR_RNDN);
          mpfr_mul_2ui(t->dcin, t->dcin, 1, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
515       mpfr_set(t->dci, t->dcin, MPFR_RNDN);
          if (t->barrier->wait(t->stop)) break;
```

```
        }
        break;
    }
    SetEvent(t0->hDone);
    mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
    return 0;
}

static int m_d_nucleus_step(complex<flyttyp> *c_out, const complex<flyttyp> &c_guess, const int64_t period,const flyttyp &epsilon2,HWND hWnd,int newtonStep, const flyttyp &radius2) {
    complex<flyttyp> z(0,0);
    complex<flyttyp> zr(0,0);
    complex<flyttyp> dc(0,0);
    int i;

        int threads = 4;
        mp_bitcnt_t bits = mpfr_get_prec(c_guess.m_r.m_dec.backend().data());
        barrier bar(4);
        STEP_STRUCT_COMMON m;
        m.hWnd = hWnd;
        m.barrier = &bar;
        m.stop = &g_bNewtonStop;
        m.newtonStep = newtonStep;
        m.period = period;
        mpfr_init2(m.zr, bits); mpfr_set_ui(m.zr, 0, MPFR_RNDN);
        mpfr_init2(m.zi, bits); mpfr_set_ui(m.zi, 0, MPFR_RNDN);
        mpfr_init2(m.zrn, bits);
        mpfr_init2(m.zin, bits);
        mpfr_init2(m.zr2, bits);
        mpfr_init2(m.zi2, bits);
        mpfr_init2(m.cr, bits); mpfr_set(m.cr, c_guess.m_r.m_dec.backend().data(), MPFR_RNDN);
        mpfr_init2(m.ci, bits); mpfr_set(m.ci, c_guess.m_i.m_dec.backend().data(), MPFR_RNDN);
        mpfr_init2(m.dcr, bits); mpfr_set_ui(m.dcr, 0, MPFR_RNDN);
        mpfr_init2(m.dci, bits); mpfr_set_ui(m.dci, 0, MPFR_RNDN);
        mpfr_init2(m.dcrn, bits);
        mpfr_init2(m.dcin, bits);
        mpfr_init2(m.dcrzr, bits);
        mpfr_init2(m.dcrzi, bits);
        mpfr_init2(m.dcizr, bits);
        mpfr_init2(m.dcizi, bits);
        STEP_STRUCT mc[4];
        HANDLE hDone[4];
        for (i = 0; i<4; i++){
                mc[i].nType =i;
                hDone[i] = mc[i].hDone = CreateEvent(NULL, 0, 0, NULL);
                mc[i].common = &m;
        }

        HANDLE hThread;
        DWORD dw;
        for (i = 0; i<threads; i++){
                hThread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ThStep, (LPVOID)&mc[i], 0, &dw);
                CloseHandle(hThread);
```

```
        }

        WaitForMultipleObjects(threads, hDone, TRUE, INFINITE);
        for (i = 0; i<threads; i++){
                CloseHandle(hDone[i]);
        }

        mpfr_set(z.m_r.m_dec.backend().data(), m.zr, MPFR_RNDN);
        mpfr_set(z.m_i.m_dec.backend().data(), m.zi, MPFR_RNDN);
        mpfr_set(dc.m_r.m_dec.backend().data(), m.dcr, MPFR_RNDN);
        mpfr_set(dc.m_i.m_dec.backend().data(), m.dci, MPFR_RNDN);
        mpfr_clear(m.zr);
        mpfr_clear(m.zi);
        mpfr_clear(m.zrn);
        mpfr_clear(m.zin);
        mpfr_clear(m.zr2);
        mpfr_clear(m.zi2);
        mpfr_clear(m.cr);
        mpfr_clear(m.ci);
        mpfr_clear(m.dcr);
        mpfr_clear(m.dci);
        mpfr_clear(m.dcrn);
        mpfr_clear(m.dcin);
        mpfr_clear(m.dcrzr);
        mpfr_clear(m.dcrzi);
        mpfr_clear(m.dcizr);
        mpfr_clear(m.dcizi);

  SetDlgItemText(hWnd,IDC_EDIT4,"");
  flyttyp ad;
#if 0
  = 1/cabs2(dc);
  if (ad < epsilon2) {
    *c_out = c_guess;
    return 0;
  }
#endif
  if(dc.m_r==0 && dc.m_i==0)
          return -1;
  complex<flyttyp> c_new = c_guess - z / dc;
  complex<flyttyp> d = c_new - c_guess;
  ad = cabs2(d);

  // progress reporting
  floatexp delta = floatexp(CFixedFloat(ad.m_dec));
  floatexp epsilon = floatexp(CFixedFloat(epsilon2.m_dec));
  std::string elast = sqrt(floatexp(4.0)/delta).toString(1);
  std::string etarget = sqrt(floatexp(4.0)/epsilon).toString(1);
  snprintf(g_szProgress, sizeof(g_szProgress) - 1, "%s %s %s", elast.c_str(), ad<
      ↳ < epsilon2 ? ">" : "<", etarget.c_str());
  g_fNewtonDelta2[0] = g_fNewtonDelta2[1];
  g_fNewtonDelta2[1] = delta;
  if (g_fNewtonDelta2[0] > 0)
    g_nNewtonETA = newtonETA(g_fNewtonDelta2[0], g_fNewtonDelta2[1], epsilon);

  if (ad < epsilon2) {
    *c_out = c_new;
```

```
625       return 0;
        }
        if (cisfinite(d) && cabs2(c_new) < 16 && ad < radius2) {
          *c_out = c_new;
          return 1;
630     } else {
          *c_out = c_guess;
          return -1;
        }
      }
635
      bool SaveNewtonBackup(const std::string &szFile, const std::string &re, const ↙
          ↪ std::string &im, const std::string &zoom, int64_t period)
      {
              bool overwrite = false; // FIXME
              CStringTable stSave;
640           stSave.AddRow();
              stSave.AddString(stSave.GetCount() - 1, "Re");
              stSave.AddString(stSave.GetCount() - 1, re);
              stSave.AddRow();
              stSave.AddString(stSave.GetCount() - 1, "Im");
645           stSave.AddString(stSave.GetCount() - 1, im);
              stSave.AddRow();
              stSave.AddString(stSave.GetCount() - 1, "Zoom");
              stSave.AddString(stSave.GetCount() - 1, zoom);
              stSave.AddRow();
650           stSave.AddString(stSave.GetCount() - 1, "Period");
              stSave.AddInt   (stSave.GetCount() - 1, period);
              char *szData = stSave.ToText(": ", "\r\n");
              HANDLE hFile = CreateFile(szFile.c_str(), GENERIC_WRITE, 0, NULL, ↙
                  ↪ overwrite ? CREATE_ALWAYS : CREATE_NEW, 0, NULL);
              if (hFile == INVALID_HANDLE_VALUE)
655           {
                      stSave.DeleteToText(szData);
                      return false;
              }
              DWORD dw;
660           WriteFile(hFile, szData, strlen(szData), &dw, NULL);
              CloseHandle(hFile);
              stSave.DeleteToText(szData);
              return true;
      }
665
      bool SaveNewtonBackup(const complex<flyttyp> &c_new, const complex<flyttyp> &↙
          ↪ c_old, int64_t period, int step)
      {
        complex<flyttyp> delta = c_new - c_old;
        complex<floatexp> delta_lo = delta;
670     std::string re = c_new.m_r.ToText();
        std::string im = c_new.m_i.ToText();
        std::string zoom = sqrt(floatexp(4.0) / cabs2(delta_lo)).toString();
        char extension[100];
        snprintf(extension, sizeof(extension) - 1, "newton-%04d.kfr", step);
675     return SaveNewtonBackup(replace_path_extension(g_szFile, extension), re, im, ↙
            ↪ zoom, period);
      }
```

```
static int m_d_nucleus(complex<flyttyp> *c_out, const complex<flyttyp> &c_guess,↙
    ↪ int64_t period, int maxsteps,int &steps,const flyttyp &radius,HWND hWnd) ↙
    ↪ {
  int result = -1, i;
  complex<flyttyp> c = c_guess;
  complex<flyttyp> c_new;

  flyttyp epsilon2 = flyttyp(1)/(radius*radius*radius);
  flyttyp radius2 = radius * radius;
  for (i = 0; i < maxsteps && !g_bNewtonStop && !g_bNewtonExit; ++i) {
    result = m_d_nucleus_step(&c_new, c, period,epsilon2,hWnd,i,radius2);
    SaveNewtonBackup(c_new, c, period, i + 1);
    c = c_new;
    if (result != 1)
      break;
  }
  steps = i;
  *c_out = c;
  if(g_bNewtonExit)
        result=0;
  return result;
}

static inline complex<floatexp> fec(const complex<flyttyp> &z)
{
  return complex<floatexp>(mpfr_get_fe(z.m_r.m_dec.backend().data()), ↙
      ↪ mpfr_get_fe(z.m_i.m_dec.backend().data()));
}

static complex<floatexp> m_d_size(const complex<flyttyp> &nucleus, int64_t ↙
    ↪ period,HWND hWnd)
{
  complex<floatexp> fec1(1,0);
  complex<floatexp> fec2(2,0);
  complex<floatexp> l(1,0);
  complex<floatexp> b(1,0);
  complex<flyttyp> z(0,0);
  char szStatus[256];
  uint32_t last = GetTickCount();
  for (int64_t i = 1; i < period && !g_bNewtonStop; ++i) {
        if(i%100==0){
              uint32_t now = GetTickCount();
              if (now - last > 250)
              {
                wsprintf(szStatus,"Determine size %d%%...",100*i/period);
                SetDlgItemText(hWnd,IDC_EDIT1,szStatus);
                last = now;
              }
        }
    z = z * z + nucleus;
    complex<floatexp> zlo = fec(z);
    l = fec2 * zlo * l;
    b = b + fec1 / l;
  }
  return fec1 / (b * l * l);
}
```

```
static int g_useDZ = 1;
static double g_skew[4];
int64_t g_period = 0;

static int WINAPI ThSkew(HWND hWnd)
{
  SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
  SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
  const int type = g_SFT.GetFractalType();
  const int power = g_SFT.GetPower();
  const struct formula *f = get_formula(type, power);

  const int64_t iters = g_SFT.GetIterations();
  g_szRe = g_SFT.GetRe();
  g_szIm = g_SFT.GetIm();
  g_szZoom = g_SFT.GetZoom();
  flyttyp radius = g_szZoom;
  radius*=g_SFT.GetZoomSize();
  const char *e = strstr(g_szZoom.c_str(),"E");
  if(!e) e = strstr(g_szZoom.c_str(),"e");
  int exp = (e?atoi(e+1):0);
  unsigned uprec = exp + 6;
  Precision prec(uprec);
  complex<flyttyp> center(g_szRe,g_szIm);

  g_skew[0] = 1;
  g_skew[1] = 0;
  g_skew[2] = 0;
  g_skew[3] = 1;
  if (f && f->skew && f->skew(iters, g_FactorAR, g_FactorAI, center.m_r.m_dec.↙
      ↳ backend().data(), center.m_i.m_dec.backend().data(), g_useDZ, &g_skew↙
      ↳ [0], &running))
  {
    PostMessage(hWnd,WM_USER+2,0,2);
  }
  else
  {
    PostMessage(hWnd,WM_USER+2,0,-2);
  }
  mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
  return 0;
}


static int WINAPI ThNewton(HWND hWnd)
{
  SetThreadPriority(GetCurrentThread(), THREAD_MODE_BACKGROUND_BEGIN);
  SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
  const int type = g_SFT.GetFractalType();
  const int power = g_SFT.GetPower();
  const struct formula *f = get_formula(type, power);
  g_skew[0] = 1;
  g_skew[1] = 0;
  g_skew[2] = 0;
  g_skew[3] = 1;

        char szStatus[300];
```

```
        flyttyp radius = g_szZoom;
        radius*=g_SFT.GetZoomSize();
        const char *e = strstr(g_szZoom.c_str(),"E");
790     if(!e)
                e = strstr(g_szZoom.c_str(),"e");
        int exp = (e?atoi(e+1):0);
        unsigned uprec = exp + 6;
        Precision prec(uprec);
795     complex<flyttyp> center(g_szRe,g_szIm);

        char szVal[25];
        int i;
        for(i=0;i<24 && g_szZoom[i] && g_szZoom[i]!='e' && g_szZoom[i]!='E' && ↙
            ↳ g_szZoom[i]!='+' && g_szZoom[i]!='-';i++)
800             szVal[i]=g_szZoom[i];
        szVal[i]=0;
        e = strstr(g_szZoom.c_str(),"E");
        if(!e)
                e = strstr(g_szZoom.c_str(),"e");
805     int startZooms = (e?atof(e+1)/0.30103:0) + log10(atof(szVal));

        int steps = 0;
        if(SendDlgItemMessage(hWnd,IDC_CHECK1,BM_GETCHECK,0,0)){
                g_period = GetDlgItemInt(hWnd,IDC_EDIT3,NULL,0);
810             uprec *= 3;
        }
        else{
          if (type == 0 && power == 2)
          {
815             int64_t maxperiod = INT_MAX; // FIXME
                g_period= ball_period_do(center,radius,maxperiod,steps,hWnd);
          }
          else
          {
820             if (f)
                {
                  flyttyp r = flyttyp(4) / radius;
                  if (g_bUseBallPeriod)
                  {
825                 g_period = f->period_jsk(INT_MAX, 1e50, g_FactorAR, ↙
                        ↳ g_FactorAI, center.m_r.m_dec.backend().data(), center.↙
                        ↳ m_i.m_dec.backend().data(), r.m_dec.backend().data(), ↙
                        ↳ &g_skew[0], &running);
                  }
                  else
                  {
                    g_period = f->period_tri(INT_MAX, 1e50, g_FactorAR, ↙
                        ↳ g_FactorAI, center.m_r.m_dec.backend().data(), center.↙
                        ↳ m_i.m_dec.backend().data(), r.m_dec.backend().data(), ↙
                        ↳ &running);
830                 }
                  if (g_period < 0) g_period = 0;
                }
                else
                {
835               g_period = 0;
```

```
            }
          }
          uprec *= 2;
        }
840     // std::cerr << "period " << g_period << std::endl;
        Precision prec2(uprec);

        SetDlgItemInt(hWnd,IDC_EDIT3,g_period,0);
        BOOL bOK=-1;
845     if(g_period){
            sprintf(szStatus,"period=%lld (steps:%d)\n",g_period,steps);
            SetDlgItemText(hWnd,IDC_EDIT1,szStatus);
            complex<flyttyp> c;
            int test = 1;
850         if (type == 0 && power == 2)
            {
              int maxsteps = INT_MAX; // FIXME
              test = m_d_nucleus(&c,center,g_period,maxsteps,steps,radius,↙
                  ↳ hWnd);
            }
855         else
            {
              if (f)
              {
                // c = center; // seems to copy precision too, so do it low-↙
                    ↳ level...
860             mpfr_set(c.m_r.m_dec.backend().data(), center.m_r.m_dec.↙
                    ↳ backend().data(), MPFR_RNDN);
                mpfr_set(c.m_i.m_dec.backend().data(), center.m_i.m_dec.↙
                    ↳ backend().data(), MPFR_RNDN);
                test = f->newton(100, g_period, g_FactorAR, g_FactorAI, c.↙
                    ↳ m_r.m_dec.backend().data(), c.m_i.m_dec.backend().data↙
                    ↳ (), &running) ? 0 : 1;
                flyttyp r = flyttyp(4) / radius;
                if (! (cabs2(c - center) < r * r))
865               test = 1;
                steps = 1;
              }
            }
            // std::cerr << "newton " << test << " " << steps << std::endl;
870
            if(test==0 && steps){
                g_szRe = c.m_r.ToText();
                g_szIm = c.m_i.ToText();

875             Precision prec3(exp + 6);
                flyttyp msize = 0;
                if (type == 0 && power == 2)
                {
                  complex<floatexp> size = m_d_size(c,g_period,hWnd);
880               floatexp msizefe = floatexp(.25)/sqrt(cabs2(size));
                  mpfr_set_fe(msize.m_dec.backend().data(), msizefe);
                }
                else
                {
885               if (f)
                  {
```

```
                                f->size(g_period, g_FactorAR, g_FactorAI, c.m_r.↙
                                    ↳ m_dec.backend().data(), c.m_i.m_dec.backend().↙
                                    ↳ data(), msize.m_dec.backend().data(), &g_skew↙
                                    ↳ [0], &running);
                                msize = flyttyp(.25) / msize;
                            }
890                         }
                            // std::cerr << "size " << msize.m_dec << std::endl;

                            double zooms1;
                            {
895                           std::ostringstream oss;
                              oss << std::scientific << msize.m_dec;
                              std::string sszSize = oss.str();
                              char *szSize0 = strdup(sszSize.c_str());
                              char *szSize = szSize0;
900                           char szTmpSize[200];
                              if(!strstr(szSize,"e") && !strstr(szSize,"E")){
                                      char *szP = strstr(szSize, ".");
                                      if (szP)
                                              *szP = 0;
905                                   int exp = strlen(szSize) - 1;
                                      if (exp>2){
                                              int end = 12;
                                              if (end>exp)
                                                      end = exp;
910                                           szSize[end + 1] = 0;
                                              while (end>1){
                                                      szSize[end] = szSize[end - 1];
                                                      end--;
                                              }
915                                           szSize[end] = '.';
                                              strcpy(szTmpSize,szSize);
                                              char szNum[20];
                                              sprintf(szNum, "E%d", exp);
                                              strcat(szTmpSize, szNum);
920                                           szSize = szTmpSize;
                                      }
                              }
                              if(strstr(szSize,"e") || strstr(szSize,"E")){
                                      int i;
925                                   for(i=0;i<24 && szSize[i] && szSize[i]!='e' &&↙
                                          ↳  szSize[i]!='E' && szSize[i]!='+' && ↙
                                          ↳ szSize[i]!='-';i++)
                                              szVal[i]=szSize[i];
                                      szVal[i]=0;
                                      e = strstr(szSize,"E");
                                      if(!e)
930                                           e = strstr(szSize,"e");
    #define LOG_10_2 0.30102999566398114
                                      zooms1 = (e?atof(e+1)/LOG_10_2:0) + log10(atof↙
                                          ↳ (szVal));
                              }
                              else
935                                   zooms1 = log10(atof(szSize))/LOG_10_2;
    #undef LOG_10_2
                              g_szZoom = szSize;
```

```
                        free(szSize0);
                    }

                    double zooms0;
                    {
                        char *szSize0 = strdup(g_sMinibrotSourceZoom.c_str());
                        char *szSize = szSize0;
                        char szTmpSize[200];
                        if(!strstr(szSize,"e") && !strstr(szSize,"E")){
                                char *szP = strstr(szSize, ".");
                                if (szP)
                                        *szP = 0;
                                int exp = strlen(szSize) - 1;
                                if (exp>2){
                                        int end = 12;
                                        if (end>exp)
                                                end = exp;
                                        szSize[end + 1] = 0;
                                        while (end>1){
                                                szSize[end] = szSize[end - 1];
                                                end--;
                                        }
                                        szSize[end] = '.';
                                        strcpy(szTmpSize,szSize);
                                        char szNum[20];
                                        sprintf(szNum, "E%d", exp);
                                        strcat(szTmpSize, szNum);
                                        szSize = szTmpSize;
                                }
                        }
                        if(strstr(szSize,"e") || strstr(szSize,"E")){
                                int i;
                                for(i=0;i<24 && szSize[i] && szSize[i]!='e' &&↙
                                    ↳  szSize[i]!='E' && szSize[i]!='+' && ↙
                                    ↳ szSize[i]!='-';i++)
                                        szVal[i]=szSize[i];
                                szVal[i]=0;
                                e = strstr(szSize,"E");
                                if(!e)
                                        e = strstr(szSize,"e");
#define LOG_10_2 0.30102999566398114
                                zooms0 = (e?atof(e+1)/LOG_10_2:0) + log10(atof↙
                                    ↳ (szVal));
                        }
                        else
                                zooms0 = log10(atof(szSize))/LOG_10_2;
#undef LOG_10_2
                        free(szSize0);
                    }

                    double zooms = zooms1;
                    if(g_nMinibrotPos){
                            if(g_nMinibrotPos==1)
                                    zooms = zooms0 * 0.5 + 0.5 * zooms1;
                            else if(g_nMinibrotPos==2)
                                    zooms = zooms0 * 0.25 + 0.75 * zooms1;
                            else if(g_nMinibrotPos==3)
```

```
                                        zooms = zooms0 * 0.125 + 0.875 * zooms1;
                                else if(g_nMinibrotPos==4)
                                {
995                                     double f = 1 - (1 - g_nMinibrotFactor) *↙
                                            ↳ 2;
                                        zooms = zooms0 * (1 - f) + f * zooms1;
                                }
                        }
                        if(4 * g_period > zooms && zooms>startZooms)
1000                    {
                                radius = flyttyp(2)^zooms;
                                g_szZoom = radius.ToText();
                                bOK=1;
                        }
1005            }
        }
        g_bNewtonRunning=FALSE;
        PostMessage(hWnd,WM_USER+2,0,bOK);
        mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
1010    return 0;
}
static __int64 t1, t2;
static SYSTEMTIME st1, st2;
extern int WINAPI NewtonProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
1015 {
        static int g_AutoSkew = 0;
        if(uMsg==WM_INITDIALOG){
                SendMessage(hWnd, WM_SETICON, ICON_SMALL, LPARAM(g_hIcon));
                SendMessage(hWnd, WM_SETICON, ICON_BIG, LPARAM(g_hIcon));
1020 //             InitToolTip(hWnd,GetModuleHandle(NULL),GetToolText,1);
                if(g_nMinibrotPos==1)
                        SendDlgItemMessage(hWnd,IDC_RADIO2,BM_SETCHECK,1,0);
                else if(g_nMinibrotPos==2)
                        SendDlgItemMessage(hWnd,IDC_RADIO3,BM_SETCHECK,1,0);
1025            else if(g_nMinibrotPos==3)
                        SendDlgItemMessage(hWnd,IDC_RADIO4,BM_SETCHECK,1,0);
                else if(g_nMinibrotPos==4)
                        SendDlgItemMessage(hWnd,IDC_RADIO5,BM_SETCHECK,1,0);
                else
1030                    SendDlgItemMessage(hWnd,IDC_RADIO1,BM_SETCHECK,1,0);
                if (g_AutoSkew)
                        SendDlgItemMessage(hWnd,IDC_AUTOSKEW,BM_SETCHECK,1,0);
                std::string z = g_SFT.GetZoom();
                SetDlgItemText(hWnd, IDC_EDIT4, z.c_str());
1035            std::ostringstream s;
                s << g_nMinibrotFactor;
                SetDlgItemText(hWnd, IDC_EDIT2, s.str().c_str());
                return 1;
        }
1040    if(uMsg==WM_COMMAND && wParam==IDCANCEL){
                if(g_bNewtonRunning){
                        running = 0;
                        g_bNewtonStop=TRUE;
                        while(g_bNewtonRunning){
1045                            MSG msg;
                                while(PeekMessage(&msg,NULL,0,0,PM_REMOVE)){
                                        TranslateMessage(&msg);
```

```
                                        DispatchMessage(&msg);
                                    }
                                    Sleep(1);
                                }
                                SetDlgItemText(hWnd,IDCANCEL,"Close");
                            }
                            else{
                                HWND hMain = GetParent(hWnd);
                                PostMessage(hMain,WM_COMMAND,ID_SPECIAL_NEWTON,0);
                                DestroyWindow(hWnd);
                            }
                        }
                        if(uMsg==WM_COMMAND && wParam==IDCANCEL2){
                            if(g_bNewtonRunning){
                                g_bNewtonExit=TRUE;
                                EnableWindow(GetDlgItem(hWnd,IDCANCEL2),FALSE);
                            }
                            else
                                MessageBeep((UINT)-1);
                        }
                        if(uMsg==WM_COMMAND && wParam==IDC_BUTTON2){
                            std::string z = g_SFT.GetZoom();
                            SetDlgItemText(hWnd, IDC_EDIT4, z.c_str());
                        }
                        if(uMsg==WM_COMMAND && wParam==IDC_AUTOSKEW_BUTTON){
                            if(!g_bNewtonRunning){
                                DWORD dw;
                                g_bNewtonStop=FALSE;
                                g_bNewtonExit=FALSE;
                                *g_szProgress=0;
                                g_nNewtonETA = 0;
                                g_fNewtonDelta2[0] = 0;
                                g_fNewtonDelta2[1] = 0;
                                running = 1;
                                g_useDZ = SendDlgItemMessage(hWnd,IDC_AUTOSKEW_USEDZ,↙
                                    ↳ BM_GETCHECK,0,0);
                                HANDLE hThread = CreateThread(NULL,0,(↙
                                    ↳ LPTHREAD_START_ROUTINE)ThSkew,hWnd,0,&dw);
                                CloseHandle(hThread);
                                g_bNewtonRunning=TRUE;
                                SetDlgItemText(hWnd,IDCANCEL,"Stop");
                            }
                        }
                        if(uMsg==WM_USER+1){
                            if(!g_bNewtonRunning){
                                mat2 m = g_SFT.GetTransformMatrix();
                                g_skew[0] = m[0][0];
                                g_skew[1] = m[0][1];
                                g_skew[2] = m[1][0];
                                g_skew[3] = m[1][1];
                                RECT r = *(RECT*)lParam;
                                g_szRe = g_SFT.GetRe(r.left,r.top,r.right,r.bottom);
                                g_szIm = g_SFT.GetIm(r.left,r.top,r.right,r.bottom);
                                g_szZoom = g_SFT.GetZoom();
                                g_nMinibrotPos=0;
                                if(SendDlgItemMessage(hWnd,IDC_RADIO2,BM_GETCHECK,0,0))
                                    g_nMinibrotPos=1;
```

```
                        else if(SendDlgItemMessage(hWnd,IDC_RADIO3,BM_GETCHECK↙
                            ↳ ,0,0))
                                g_nMinibrotPos=2;
1105                    else if(SendDlgItemMessage(hWnd,IDC_RADIO4,BM_GETCHECK↙
                            ↳ ,0,0))
                                g_nMinibrotPos=3;
                        else if(SendDlgItemMessage(hWnd,IDC_RADIO5,BM_GETCHECK↙
                            ↳ ,0,0))
                                g_nMinibrotPos=4;
                        g_bUseBallPeriod = SendDlgItemMessage(hWnd,↙
                            ↳ IDC_BALL_PERIOD,BM_GETCHECK,0,0);
1110                    DWORD dw;
                        g_bNewtonStop=FALSE;
                        g_bNewtonExit=FALSE;
                        char szText[256];
                        GetDlgItemText(hWnd,IDC_EDIT2,szText,sizeof(szText));
1115                    g_nMinibrotFactor = atof(szText);
                        GetDlgItemText(hWnd,IDC_EDIT4,szText,sizeof(szText));
                        g_sMinibrotSourceZoom = std::string(szText);
                        *g_szProgress=0;
                        g_nNewtonETA = 0;
1120                    g_fNewtonDelta2[0] = 0;
                        g_fNewtonDelta2[1] = 0;
                        GetLocalTime(&st1);
                        running = 1;
                        HANDLE hThread = CreateThread(NULL,0,(↙
                            ↳ LPTHREAD_START_ROUTINE)ThNewton,hWnd,0,&dw);
1125                    CloseHandle(hThread);
                        g_bNewtonRunning=TRUE;
                        SetDlgItemText(hWnd,IDCANCEL,"Stop");
                }
        }
1130    if(uMsg==WM_USER+2){
                SetDlgItemText(hWnd,IDCANCEL,"Close");
                g_bNewtonRunning=FALSE;
                if(lParam == 1){
                        // newton success
1135                    g_SFT.UndoStore();
                        if((g_AutoSkew = SendDlgItemMessage(hWnd,IDC_AUTOSKEW,↙
                            ↳ BM_GETCHECK,0,0)))
                                g_SFT.SetTransformMatrix(mat2(g_skew[0], g_skew↙
                                    ↳ [1], g_skew[2], g_skew[3]));
                        g_SFT.SetPosition(g_szRe,g_szIm,g_szZoom);
                        g_SFT.SetIterations(3*g_SFT.GetIterations()/2);
1140                    g_bJustDidNewton = true;
                        PostMessage(GetParent(hWnd),WM_KEYDOWN,VK_F5,0);
                        if(SendDlgItemMessage(hWnd,IDC_CHECK9,BM_GETCHECK,0,0)){
                                GetLocalTime(&st2);
                                SystemTimeToFileTime(&st1,(LPFILETIME)&t1);
1145                            SystemTimeToFileTime(&st2,(LPFILETIME)&t2);
                                __int64 nT2 = t2-t1;
                                FileTimeToSystemTime((LPFILETIME)&nT2,&st1);
                                char szResult[256];
                                wsprintf(szResult,"Time: %02d:%02d:%02d.%03d.%05↙
                                    ↳ d\nPeriod: %d",st1.wHour,st1.wMinute,st1.↙
                                    ↳ wSecond,st1.wMilliseconds,(int)(nT2%10000)↙
                                    ↳ ,g_period);
```

```
1150                                    MessageBox(hWnd, szResult ," Result" ,MB_OK);
                                }
                        }
                        if(lParam == 2){
                                // auto skew success
1155                            g_SFT.UndoStore();
                                g_SFT.SetTransformMatrix(mat2(g_skew[0], g_skew[1], ↙
                                    ↳ g_skew[2], g_skew[3]));
                                PostMessage(GetParent(hWnd),WM_KEYDOWN,VK_F5,0);
                        }
                        PostMessage(GetParent(hWnd),WM_COMMAND,ID_SPECIAL_NEWTON,0);
1160                    if(lParam == -1 && !g_bNewtonStop)
                                MessageBox(GetParent(hWnd),"Could not apply Newton-↙
                                    ↳ Raphson\nYou may zoom in a little and try again","↙
                                    ↳ Error",MB_OK|MB_ICONSTOP);
                        if(lParam == -2 && !g_bNewtonStop)
                                MessageBox(GetParent(hWnd),"Could not auto find skew\n↙
                                    ↳ ","Error",MB_OK|MB_ICONSTOP);
                        if((lParam == 0 || lParam < -2 || lParam > 2))
1165                            MessageBox(GetParent(hWnd),"Unexpected stop message ↙
                                    ↳ parameter (internal error)\n","Error",MB_OK|↙
                                    ↳ MB_ICONSTOP);
                }
                return 0;
        }
```

98 fraktal_sft/newton.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen
5
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15
You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20  #ifndef KF_NEWTON_H
    #define KF_NEWTON_H 1

    extern bool g_bNewtonRunning;
    extern bool g_bNewtonStop;
25  extern int64_t g_period;
    extern bool g_bJustDidNewton;

    extern int WINAPI NewtonProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
```

```
30  #endif
```

99 fraktal_sft/Parameter.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "cmdline.h"
    #include "tiff.h"
    #include "png.h"
    #include "jpeg.h"
25  #include "exr.h"
    #include "main.h"
    #include "newton.h"
    #include "../common/StringVector.h"

30  #include <iostream>

    BOOL CFraktalSFT::OpenFile(const std::string &szFile, BOOL bNoLocation)
    {
            std::string data;
35    std::string extension = get_filename_extension(szFile);
            if (extension == "tif" || extension == "tiff")
            {
                    data = ReadTIFFComment(szFile);
                    if (data == "")
40                    return FALSE;
            }
            else if (extension == "png")
            {
                    data = ReadPNGComment(szFile);
45                  if (data == "")
                      return FALSE;
            }
            else if (extension == "jpg" || extension == "jpeg")
            {
50                  data = ReadJPEGComment(szFile);
                    if (data == "")
                      return FALSE;
            }
```

```
        else if (extension == "exr")
        {
                data = ReadEXRComment(szFile);
                if (data == "")
                  return FALSE;
        }
        else // anything else, probably .kfr
        {
                DWORD dw;
                HANDLE hFile = CreateFile(szFile.c_str(), GENERIC_READ, 0, NULL,↙
                    ↳ OPEN_EXISTING, 0, NULL);
                if (hFile == INVALID_HANDLE_VALUE)
                        return FALSE;
                int nData = GetFileSize(hFile, NULL);
                char *szData = new char[nData + 1];
                ReadFile(hFile, szData, nData, &dw, NULL);
                CloseHandle(hFile);
                szData[nData] = 0;
    data = szData;
    delete[] szData;
        }
  return OpenString(data, bNoLocation);
}

BOOL CFraktalSFT::OpenString(const std::string &data, BOOL bNoLocation)
{
        CStringTable stParams(data.c_str(), ": ", "\r\n");
        {
                int nv = stParams.FindString(0, "Version");
                if (nv != -1)
                {
                        int str_version_number = atoi(stParams[nv][1]);
                        if (str_version_number > kfr_version_number)
                        {
                                fprintf(stderr, "WARNING: file format is newer ↙
                                    ↳ than this EXE version\n");
                        }
                }
        }
        int nR = -1, nI = -1, nZ = -1, nIterations = -1;
        if (! bNoLocation)
        {
                nR = stParams.FindString(0, "Re");
                nI = stParams.FindString(0, "Im");
                nZ = stParams.FindString(0, "Zoom");
                nIterations = stParams.FindString(0, "Iterations");
        }

        int nID = stParams.FindString(0, "IterDiv");
        if (nID == -1)
                m_nIterDiv = 1;
        else
                m_nIterDiv = atof(stParams[nID][1]);

        if (! bNoLocation)
        {
        nID = stParams.FindString(0, "Rotate");
```

```
        if (nID == -1)
110             g_Degree = 0;
        else
                g_Degree = atof(stParams[nID][1]);
        }

115     if (! bNoLocation)
        {
        nID = stParams.FindString(0, "Ratio");
        if (nID == -1)
                m_scRatio.cy = 360;
120     else
                m_scRatio.cy = atof(stParams[nID][1]);
        SIZE size;
        size.cx = GetImageWidth();
        size.cy = GetImageHeight();
125     double xRatio = 640.0/size.cx;
        size.cx = 640;
        size.cy = size.cy*xRatio;
        xRatio = (double)size.cy/(double)360;
        m_scRatio.cy*=xRatio;
130     }

        nID = stParams.FindString(0, "ColorMethod");
        if (nID != -1){
                int m = atoi(stParams[nID][1]);
135             if (m<0 || m>10)
                        m = 0;
                m_nColorMethod = ColorMethod(m);
        }
        else
140             m_nColorMethod = ColorMethod_Standard;
        nID = stParams.FindString(0, "Differences");
        if (nID != -1)
        {
                int m = atoi(stParams[nID][1]);
145             if (m < 0 || m > 7)
                        m = 0;
                m_nDifferences = Differences(m);
        }
        else
150             m_nDifferences = Differences_Traditional;
        nID = stParams.FindString(0, "ColorOffset");
        if (nID != -1){
                m_nColorOffset = atoi(stParams[nID][1]);
                if (m_nColorOffset<0 || m_nColorOffset>1023)
155                     m_nColorOffset = 0;
        }
        else
                m_nColorOffset = 0;

160     if (! bNoLocation)
        {
        nID = stParams.FindString(0, "Power");
        if (nID != -1){
                m_nPower = atoi(stParams[nID][1]);
165             if (m_nPower<2)
```

```
                m_nPower = 2;
        }
        else
                m_nPower = 2;
        nID = stParams.FindString(0, "FractalType");
        if (nID != -1)
                m_nFractalType = atoi(stParams[nID][1]);
        else
                m_nFractalType = 0;
        }

        int nT = stParams.FindString(0, "Smooth");
        if (nT != -1)
                m_bTrans = atoi(stParams[nT][1]);
        if (! bNoLocation)
        {
        nID = stParams.FindString(0, "SmoothMethod"); // must come after "Power"
        if (nID != -1){
                int m = atoi(stParams[nID][1]);
                if (m<0 || m>2)
                        m = 0;
                m_nSmoothMethod = SmoothMethod(m);
        }
        m_nBailout = m_nSmoothMethod == SmoothMethod_SqrtLow ? pow(2.0, 1.0 / (↙
            ↳ m_nPower - 1))
                : m_nSmoothMethod == SmoothMethod_Sqrt ? 2 : SMOOTH_BAILOUT;
        m_nBailout2 = m_nBailout*m_nBailout;
        }

        nID = stParams.FindString(0, "Slopes");
        if (nID != -1)
                m_bSlopes = atoi(stParams[nID][1]);
        else
                m_bSlopes = FALSE;
        nID = stParams.FindString(0, "SlopePower");
        if (nID != -1){
                m_nSlopePower = atoi(stParams[nID][1]);
                if (m_nSlopePower<1)
                        m_nSlopePower = 1;
        }
        else
                m_nSlopePower = 100;
        nID = stParams.FindString(0, "SlopeRatio");
        if (nID != -1){
                m_nSlopeRatio = atoi(stParams[nID][1]);
                if (m_nSlopeRatio<0)
                        m_nSlopeRatio = 0;
        }
        else
                m_nSlopeRatio = 50;
        nID = stParams.FindString(0, "SlopeAngle");
        if (nID != -1)
                m_nSlopeAngle = atoi(stParams[nID][1]);
        else
                m_nSlopeAngle = 45;

        SetSlopes(m_bSlopes, m_nSlopePower, m_nSlopeRatio, m_nSlopeAngle);
```

```
	if (! bNoLocation)
	{
	nID = stParams.FindString(0, "real");
	if (nID != -1)
		g_real = atoi(stParams[nID][1]);
	else
		g_real = 1;
	nID = stParams.FindString(0, "imag");
	if (nID != -1)
		g_imag = atoi(stParams[nID][1]);
	else
		g_imag = 1;

	nID = stParams.FindString(0, "SeedR");
	if (nID != -1)
		g_SeedR = atof(stParams[nID][1]);
	else
		g_SeedR = 0;
	nID = stParams.FindString(0, "SeedI");
	if (nID != -1)
		g_SeedI = atof(stParams[nID][1]);
	else
		g_SeedI = 0;
	nID = stParams.FindString(0, "FactorAR");
	if (nID != -1)
		g_FactorAR = atof(stParams[nID][1]);
	else
		g_FactorAR = 1;
	nID = stParams.FindString(0, "FactorAI");
	if (nID != -1)
		g_FactorAI = atof(stParams[nID][1]);
	else
		g_FactorAI = 0;

	nID = stParams.FindString(0, "Period");
	if (nID != -1)
		g_period = atoll(stParams[nID][1]);
	else
		g_period = 0;
	}

	if (! bNoLocation)
	{
	bool ld = GetLongDoubleAlways();
	bool fe = GetFloatExpAlways();
	if (scaling_supported(m_nFractalType, m_nPower, GetDerivatives()))
	{
		if (m_nPower == 2)
		{
			g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_2;
			g_nEXP = FLOATEXP_THRESHOLD_POWER_2;
		}
		if (m_nPower == 3)
		{
			g_nLDBL = LONG_DOUBLE_THRESHOLD_POWER_3;
			g_nEXP = FLOATEXP_THRESHOLD_POWER_3;
```

```
                }
        }
        else
        {
                g_nLDBL = LONG_DOUBLE_THRESHOLD_DEFAULT;
                g_nEXP = FLOATEXP_THRESHOLD_DEFAULT;
        }
        if (ld)
        {
                g_nLDBL = 3;
        }
        if (fe)
        {
                g_nEXP = 2;
        }
        }

        int nC = stParams.FindString(0, "Colors");
    if (nC != -1)
    {
                CStringTable stColors(stParams[nC][1], "", ",");
                m_nParts = stColors.GetCount() / 3;
                int i;
                for (i = 0; i<m_nParts; i++){
                        m_cKeys[i].r = atoi(stColors[i * 3][0]);
                        m_cKeys[i].g = atoi(stColors[i * 3 + 1][0]);
                        m_cKeys[i].b = atoi(stColors[i * 3 + 2][0]);
                }
    }
        int i = stParams.FindString(0, "MultiColor");
        if (i != -1){
                m_bMW = atoi(stParams[i][1]);
                m_nMW = 0;
                i = stParams.FindString(0, "MultiColors");
                if (i != -1){
                        CStringTable stColors(stParams[i][1], "\t", ",");
                        for (i = 0; i<stColors.GetCount() && m_nMW<MULTIWAVE_MAX
                            ↪ ; i++){
                                m_MW[m_nMW].nPeriod = atoi(stColors[i][0]);
                                m_MW[m_nMW].nStart = atoi(stColors[i][1]);
                                m_MW[m_nMW].nType = atoi(stColors[i][2]);
                                m_nMW++;
                        }
                }
        }
        else
                m_bMW = 0;
        i = stParams.FindString(0, "BlendMC");
        if (i != -1)
                m_bBlend = atoi(stParams[i][1]);

        i = stParams.FindString(0, "InteriorColor");
        if (i != -1)
        {
                CStringTable stColor(stParams[i][1], "", ",");
                if (stColor.GetCount() == 3)
                {
```

```
335                             m_cInterior.r = atoi(stColor[0][0]);
                                m_cInterior.g = atoi(stColor[1][0]);
                                m_cInterior.b = atoi(stColor[2][0]);
                        }
                }
340
                if (! bNoLocation)
                {
                        if (nIterations != -1) m_nMaxIter = atoll(stParams[nIterations↙
                            ↳ ][1]);
                        std::string re = GetRe();
345                     std::string im = GetIm();
                        std::string zm = GetZoom();
                        if (nR != -1) re = stParams[nR][1];
                        if (nI != -1) im = stParams[nI][1];
                        if (nZ != -1) zm = stParams[nZ][1];
350                     SetPosition(re, im, zm);
                }
                ApplyColors();
                if (m_hWnd) InvalidateRect(m_hWnd, NULL, FALSE);
                return TRUE;
355     }

        std::string CFraktalSFT::ToText()
        {
                CStringTable stSave;
360             stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "Re");
                stSave.AddString(stSave.GetCount() - 1, GetRe());
                stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "Im");
365             stSave.AddString(stSave.GetCount() - 1, GetIm());
                stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "Zoom");
                stSave.AddString(stSave.GetCount() - 1, GetZoom());
                stSave.AddRow();
370             stSave.AddString(stSave.GetCount() - 1, "Iterations");
                stSave.AddInt(stSave.GetCount() - 1, m_nMaxIter);
                stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "IterDiv");
                char szDiv[256];
375             sprintf(szDiv, "%f", m_nIterDiv);
                stSave.AddString(stSave.GetCount() - 1, szDiv);
                stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "SmoothMethod");
                stSave.AddInt(stSave.GetCount() - 1, m_nSmoothMethod);
380             stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "ColorMethod");
                stSave.AddInt(stSave.GetCount() - 1, m_nColorMethod);
                stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "Differences");
385             stSave.AddInt(stSave.GetCount() - 1, m_nDifferences);
                stSave.AddRow();
                stSave.AddString(stSave.GetCount() - 1, "ColorOffset");
                stSave.AddInt(stSave.GetCount() - 1, m_nColorOffset);

390             stSave.AddRow();
```

```
	stSave.AddString(stSave.GetCount() - 1, "Rotate");
	sprintf(szDiv, "%f", g_Degree);
	stSave.AddString(stSave.GetCount() - 1, szDiv);

	stSave.AddRow();
	stSave.AddString(stSave.GetCount() - 1, "Ratio");
	double nRatio = m_scRatio.cy;
	SIZE size;
	size.cx = m_nX;
	size.cy = m_nY;
	double xRatio = 640.0/size.cx;
	size.cx = 640;
	size.cy = size.cy*xRatio;
	xRatio = (double)360/(double)size.cy;
	nRatio*=xRatio;
	sprintf(szDiv, "%f", nRatio);
	stSave.AddString(stSave.GetCount() - 1, szDiv);

	CStringTable stColors;
	int i;
	for (i = 0; i<m_nParts; i++){
		stColors.AddRow();
		stColors.AddInt(stColors.GetCount() - 1, m_cKeys[i].r);
		stColors.AddRow();
		stColors.AddInt(stColors.GetCount() - 1, m_cKeys[i].g);
		stColors.AddRow();
		stColors.AddInt(stColors.GetCount() - 1, m_cKeys[i].b);
	}
	char *szColors = stColors.ToText("", ",");
	stSave.AddRow();
	stSave.AddString(stSave.GetCount() - 1, "Colors");
	stSave.AddString(stSave.GetCount() - 1, szColors);
	stColors.DeleteToText(szColors);

	{
		CStringTable stColor;
		stColor.AddRow();
		stColor.AddInt(stColor.GetCount() - 1, m_cInterior.r);
		stColor.AddRow();
		stColor.AddInt(stColor.GetCount() - 1, m_cInterior.g);
		stColor.AddRow();
		stColor.AddInt(stColor.GetCount() - 1, m_cInterior.b);
		char *szColor = stColor.ToText("", ",");
		stSave.AddRow();
		stSave.AddString(stSave.GetCount() - 1, "InteriorColor");
		stSave.AddString(stSave.GetCount() - 1, szColor);
		stColor.DeleteToText(szColor);
	}

	stSave.AddRow();
	stSave.AddString(stSave.GetCount() - 1, "Smooth");
	stSave.AddInt(stSave.GetCount() - 1, m_bTrans);

	stSave.AddRow();
	stSave.AddString(stSave.GetCount() - 1, "MultiColor");
	stSave.AddInt(stSave.GetCount() - 1, mbMW);
	stSave.AddRow();
```

```
        stSave.AddString(stSave.GetCount() - 1, "BlendMC");
        stSave.AddInt(stSave.GetCount() - 1, m_bBlend);
        stColors.Reset();
        for (i = 0; i<mnMW; i++){
                stColors.AddRow();
                stColors.AddInt(stColors.GetCount() - 1, mMW[i].nPeriod);
                stColors.AddInt(stColors.GetCount() - 1, mMW[i].nStart);
                stColors.AddInt(stColors.GetCount() - 1, mMW[i].nType);
        }
        szColors = stColors.ToText("\t", ",");
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "MultiColors");
        stSave.AddString(stSave.GetCount() - 1, szColors);
        stColors.DeleteToText(szColors);

        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "Power");
        stSave.AddInt(stSave.GetCount() - 1, m_nPower);

        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "FractalType");
        stSave.AddInt(stSave.GetCount() - 1, m_nFractalType);

        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "Slopes");
        stSave.AddInt(stSave.GetCount() - 1, m_bSlopes);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "SlopePower");
        stSave.AddInt(stSave.GetCount() - 1, m_nSlopePower);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "SlopeRatio");
        stSave.AddInt(stSave.GetCount() - 1, m_nSlopeRatio);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "SlopeAngle");
        stSave.AddInt(stSave.GetCount() - 1, m_nSlopeAngle);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "imag");
        stSave.AddInt(stSave.GetCount() - 1, g_imag);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "real");
        stSave.AddInt(stSave.GetCount() - 1, g_real);
        stSave.AddRow();
        char szTmp[50];
        sprintf(szTmp,"%g",g_SeedR);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "SeedR");
        stSave.AddString(stSave.GetCount() - 1, szTmp);
        sprintf(szTmp,"%g",g_SeedI);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "SeedI");
        stSave.AddString(stSave.GetCount() - 1, szTmp);
        sprintf(szTmp,"%g",g_FactorAR);
        stSave.AddRow();
        stSave.AddString(stSave.GetCount() - 1, "FactorAR");
        stSave.AddString(stSave.GetCount() - 1, szTmp);
        sprintf(szTmp,"%g",g_FactorAI);
        stSave.AddRow();
```

```
505         stSave.AddString(stSave.GetCount() - 1, "FactorAI");
            stSave.AddString(stSave.GetCount() - 1, szTmp);
            stSave.AddRow();
            stSave.AddString(stSave.GetCount() - 1, "Period");
            stSave.AddInt(stSave.GetCount() - 1, g_period);
510         stSave.AddRow();
            stSave.AddString(stSave.GetCount() - 1, "Version");
            stSave.AddInt(stSave.GetCount() - 1, kfr_version_number);
            char *szData = stSave.ToText(": ", "\r\n");
            std::string ret(szData);
515         stSave.DeleteToText(szData);
            return ret;
    }

    BOOL CFraktalSFT::SaveFile(const std::string &szFile, bool overwrite)
520 {
            std::string szText(ToText());
            const char *szData = szText.c_str();
            DWORD dw;
            HANDLE hFile = CreateFile(szFile.c_str(), GENERIC_WRITE, 0, NULL, ↙
                ↳ overwrite ? CREATE_ALWAYS : CREATE_NEW, 0, NULL);
525         if (hFile == INVALID_HANDLE_VALUE)
                return FALSE;
            WriteFile(hFile, szData, strlen(szData), &dw, NULL);
            CloseHandle(hFile);
            return TRUE;
530 }
```

100 fraktal_sft/png.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2018 Claude Heiland-Allen
 5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include <stdio.h>
    #include <string.h>
    #include <time.h>
    #include <string>
    #include <png.h>
25  #include <zlib.h>

    #include "png.h"
```

```
static void kf_png_error_handler(png_structp png, png_const_charp msg)
{
        // FIXME make this display in the GUI or something
        fprintf(stderr, "PNG ERROR: %s\n", msg);
        longjmp(*static_cast<jmp_buf *>(png_get_error_ptr(png)), 1);
}

static void kf_png_warning_handler(png_structp png, png_const_charp msg)
{
        (void) png;
        // FIXME make this display in the GUI or something
        fprintf(stderr, "PNG WARNING: %s\n", msg);
}

static bool skip_png_image(png_structp png, png_infop info);

extern int SavePNG(const std::string &szFileName, char *Data, int nHeight, int nWidth, int nColors, const std::string &comment)
{
        jmp_buf jmpbuf;
        if (nColors != 3)
                return 0;
        FILE *file = fopen(szFileName.c_str(), "wb");
        if (! file)
                return 0;
        png_structp png = png_create_write_struct(PNG_LIBPNG_VER_STRING, &jmpbuf, kf_png_error_handler, kf_png_warning_handler);
        if (! png)
                return 0;
        png_infop info = png_create_info_struct(png);
        if (! info)
        {
                png_destroy_write_struct(&png, 0);
                fclose(file);
                return 0;
        }
        if (setjmp(jmpbuf))
        {
                png_destroy_write_struct(&png, &info);
                fclose(file);
                return 0;
        }
        png_init_io(png, file);
        png_set_compression_level(png, Z_BEST_COMPRESSION);
        png_set_IHDR(png, info, nWidth, nHeight, 8, PNG_COLOR_TYPE_RGB, PNG_INTERLACE_ADAM7, PNG_COMPRESSION_TYPE_DEFAULT, PNG_FILTER_TYPE_DEFAULT);
        png_time mtime;
        png_convert_from_time_t(&mtime, time(0));
        png_set_tIME(png, info, &mtime);
        png_text text;
        text.compression = PNG_TEXT_COMPRESSION_NONE;
        const std::string &key = "Comment";
        text.key = const_cast<char *>(key.c_str());
        text.text = const_cast<char *>(comment.c_str());
        png_set_text(png, info, &text, 1);
```

```
        png_write_info(png, info);
        png_bytepp row = new png_bytep[nHeight];
        for (int y = 0; y < nHeight; ++y)
                row[y] = (png_bytep)(Data + nWidth * nColors * y);
        png_write_image(png, row);
        png_write_end(png, 0);
        delete [] row;
        fclose(file);
        return 1;
}

extern std::string ReadPNGComment(const std::string &filename)
{
        jmp_buf jmpbuf;
        FILE *file = fopen(filename.c_str(), "rb");
        if (! file)
                return "";
        png_structp png = png_create_read_struct(PNG_LIBPNG_VER_STRING, &jmpbuf,↙
            ↳ kf_png_error_handler, kf_png_warning_handler);
        if (! png)
        {
                fclose(file);
                return "";
        }
        png_infop info = png_create_info_struct(png);
        if (! info)
        {
                png_destroy_read_struct(&png, 0, 0);
                fclose(file);
                return "";
        }
        png_infop enfo = png_create_info_struct(png);
        if (! enfo)
        {
                png_destroy_read_struct(&png, &info, 0);
                fclose(file);
                return "";
        }
        if (setjmp(jmpbuf))
        {
                png_destroy_read_struct(&png, &info, 0);
                fclose(file);
                return "";
        }
        png_init_io(png, file);
        png_read_info(png, info);
        png_textp text;
        int count = 0;
        std::string comment = "";
        if (png_get_text(png, info, &text, &count) > 0)
                for (int t = 0; t < count; t++)
                        // we save as capitalized, but processing with ↙
                            ↳ ImageMagick downcases
                        if (0 == stricmp("Comment", text[t].key))
                                comment = text[t].text; // copy
        if (comment == "")
        {
```

```
                if (skip_png_image(png, info))
                {
                        png_read_end(png, enfo);
                        png_textp etext;
                        int ecount = 0;
                        if (png_get_text(png, enfo, &etext, &ecount) > 0)
                                for (int t = 0; t < ecount; t++)
                                        // we save as capitalized, but ↙
                                            ↳ processing with ImageMagick ↙
                                            ↳ downcases
                                        if (0 == stricmp("Comment", etext[t].key↙
                                            ↳ ))
                                                comment = etext[t].text; // copy
                }
        }
        png_destroy_read_struct(&png, &info, &enfo);
        fclose(file);
        return comment;
}

static bool skip_png_image(png_structp png, png_infop info)
{
        // this doesn't really skip, it decodes the image
        // hack: use one single row of memory for each row pointer
        // reduces memory usage to O(W + H) from O(W * H)
        bool ok = false;
        png_uint_32 width, height = 0;
        int bit_depth, color_type;
        if (png_get_IHDR(png, info, &width, &height, &bit_depth, &color_type, 0,↙
            ↳ 0, 0))
        {
                png_read_update_info(png, info);
                png_uint_32 bytes = png_get_rowbytes(png, info);
                png_bytep row;
                if ((row = (png_bytep) malloc(bytes)))
                {
                        png_bytepp rows;
                        if ((rows = (png_bytepp) calloc(1, height * sizeof(↙
                            ↳ png_bytep))))
                        {
                                for (png_uint_32 i = 0; i < height; ++i) rows[i]↙
                                    ↳ = row;
                                png_read_image(png, rows);
                                ok = true;
                                free(rows);
                        }
                        free(row);
                }
        }
        return ok;
}
```

101 fraktal_sft/png.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
```

```
    Copyright (C) 2017-2018 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #ifndef KF_PNG_H
    #define KF_PNG_H 1

    #include <string>

25  extern int SavePNG(const std::string &szFileName, char *Data, int nHeight, int ↙
        ↳ nWidth, int nColors, const std::string &comment);
    extern std::string ReadPNGComment(const std::string &filename);

    #endif
```

102 fraktal_sft/render.cpp

```
    /*
    Kalles Fraktaler 2
    Copyright (C) 2013-2017 Karl Rummo
    Copyright (C) 2017-2019 Claude Heiland-Allen
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include "fraktal_sft.h"
    #include "newton.h"
    #include "../common/StringVector.h"
    #include "../common/bitmap.h"
    #include "../common/parallell.h"
25
    #include <cstring>

    static int WINAPI ThRenderFractal(CFraktalSFT *p)
```

```
    {
30  #ifndef _DEBUG
            try{
    #endif
                    p->RenderFractal();
    #ifndef _DEBUG
35          }
            catch (...) {
                    p->m_bRunning=FALSE;
    //MessageBox(GetActiveWindow(),"Krash - 2","Krash",MB_OK);
            }
40  #endif
            mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
            return 0;
    }

45  static int ThMandelCalc(TH_PARAMS *pMan)
    {
    #ifndef _DEBUG
            try{
    #endif
50                  pMan->p->MandelCalc();
    #ifndef _DEBUG
            }
            catch (...) {
    //              pMan->p->m_bRunning=FALSE;
55  MessageBox(GetActiveWindow(),"Krash - 1","Krash",MB_OK);
            }
    #endif
            mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
            return 0;
60  }

    static int ThMandelCalcEXP(TH_PARAMS *pMan)
    {
            pMan->p->MandelCalcEXP();
65          mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
            return 0;
    }

    static int ThMandelCalcLDBL(TH_PARAMS *pMan)
70  {
            pMan->p->MandelCalcLDBL();
            mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
            return 0;
    }
75
    static int ThMandelCalcNANOMB1(TH_PARAMS *pMan)
    {
            pMan->p->MandelCalcNANOMB1();
            mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
80          return 0;
    }

    static int ThMandelCalcNANOMB2(TH_PARAMS *pMan)
    {
85          pMan->p->MandelCalcNANOMB2();
```

```
        mpfr_free_cache2(MPFR_FREE_LOCAL_CACHE);
        return 0;
}

void CFraktalSFT::RenderFractal(int nX, int nY, int64_t nMaxIter, HWND hWnd, BOOL bNoThread, BOOL bResetOldGlitch)
{
        m_bStop = TRUE;
        double counter = 0;
        while(m_bRunning)
        {
                Sleep(4);
                counter += 4;
        }
#ifdef KF_DEBUG_SLEEP
        if (counter > 0)
                std::cerr << "RenderFractal() slept for " << counter << "ms" << std::endl;
#endif
        m_bStop = FALSE;
        if (hWnd)
                m_hWnd = hWnd;
        m_nX = nX;
        m_nY = nY;
        SetImageWidth(nX);
        SetImageHeight(nY);
        m_nMaxIter = nMaxIter;
        m_nRDone = m_nDone = m_nGuessed = 0;
        if (bResetOldGlitch)
                memset(m_pOldGlitch, -1, sizeof(m_pOldGlitch));
        if (m_nPower>10 && m_nPrevPower != m_nPower){
                m_nPrevPower = m_nPower;
                if (m_pnExpConsts){
                        delete[] m_pnExpConsts;
                        m_pnExpConsts = NULL;
                }
                CStringTable stVal("4,6", "", ",");
                int i, k;
                char szTmp[20];
                for (i = 5; i <= m_nPower; i++){
                        stVal[0].InsertString(0, itoa(i, szTmp, 10));
                        for (k = 1; k<stVal.GetCount(); k++)
                                stVal[k].InsertString(0, itoa(atoi(stVal[k - 1][1]) + atoi(stVal[k][0]), szTmp, 10));
                        if (i % 2 == 0){
                                stVal.AddRow();
                                stVal.AddInt(stVal.GetCount() - 1, atoi(stVal[stVal.GetCount() - 2][1]) * 2);
                        }
                }
                m_pnExpConsts = new int[m_nPower - 1];
                k = 0;
                for (i = 0; i<stVal.GetCount(); i++)
                        m_pnExpConsts[k++] = atoi(stVal[i][0]);
                if (m_nPower % 2 == 0)
                        i--;
                for (i--; i >= 0; i--)
```

```
                        m_pnExpConsts[k++] = atoi(stVal[i][0]);
        }
        else if (m_nPower <= 10 && m_pnExpConsts){
                delete[] m_pnExpConsts;
                m_pnExpConsts = NULL;
        }

        WaitForSingleObject(m_hMutex, INFINITE);
        bool resize = m_nXPrev != m_nX || m_nYPrev != m_nY;
        if (resize){
                SetImageSize(m_nX, m_nY);
        }
        if (m_bResized || resize)
        {
                DeleteObject(m_bmBmp);
                m_bmBmp = NULL;
        }
        HDC hDC = GetDC(NULL);
        if (!m_bmBmp)
                m_bmBmp = create_bitmap(hDC, m_nX, m_nY);

        if (!m_bAddReference){
                if (m_bmi)
                        free(m_bmi);
                m_bmi = (BITMAPINFOHEADER *)malloc(sizeof(BITMAPINFOHEADER)+↙
                    ↳ sizeof(RGBQUAD)* 256);
                memset(m_bmi, 0, sizeof(BITMAPINFOHEADER)+sizeof(RGBQUAD)* 256);
                m_bmi->biSize = sizeof(BITMAPINFOHEADER);
                if (!GetDIBits(hDC, m_bmBmp, 0, 0, NULL, (LPBITMAPINFO)m_bmi, ↙
                    ↳ DIB_RGB_COLORS))
                        { /*Beep(1000,10)*/ }
                m_bmi->biCompression = m_bmi->biClrUsed = m_bmi->biClrImportant ↙
                    ↳ = 0;
                m_bmi->biBitCount = 24;
                if (m_bmi->biBitCount != 24)
                        m_bmi->biClrUsed = 1 << m_bmi->biBitCount;
                m_row = ((((m_bmi->biWidth*(DWORD)m_bmi->biBitCount) + 31)&~31) ↙
                    ↳ >> 3);
                m_bmi->biSizeImage = m_row*m_bmi->biHeight;
                if (!m_lpBits || (int)m_bmi->biSizeImage != m_nSizeImage){
                        m_nSizeImage = m_bmi->biSizeImage;
                        if (m_lpBits)
                                delete[] m_lpBits;
                        m_lpBits = new BYTE[m_bmi->biSizeImage];
                        if (m_imageHalf)
                                delete[] m_imageHalf;
                        m_imageHalf = nullptr;
                        SetHalfColour(GetHalfColour()); // reallocate if ↙
                            ↳ necessary
                        if (!GetDIBits(hDC, m_bmBmp, 0, m_bmi->biHeight, ↙
                            ↳ m_lpBits,
                                (LPBITMAPINFO)m_bmi, DIB_RGB_COLORS))
                                { /*Beep(1000,10)*/ }
                }
        }
        ReleaseDC(NULL, hDC);
        ReleaseMutex(m_hMutex);
```

```
190
        CFixedFloat pixel_spacing = (m_istop - m_istart) / m_nY;
        m_fPixelSpacing = pixel_spacing;
        m_dPixelSpacing = m_fPixelSpacing.todouble();
        m_lPixelSpacing = m_fPixelSpacing.toLongDouble();
195
        if (bNoThread){
                if (m_hWnd)
                        SetTimer(m_hWnd, 0, 100, NULL);
                ThRenderFractal(this);
200     }
        else{
                m_bRunning = TRUE;
                DWORD dw;
                HANDLE hThread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)↙
                    ↳ ThRenderFractal, (LPVOID)this, 0, &dw);
205             CloseHandle(hThread);
        }
}

void CFraktalSFT::RenderFractal()
210 {
        m_C = cos(g_Degree);
        m_S = sin(g_Degree);

        CFixedFloat div = m_istop - m_istart;
215
        int nZeroes = 0;
        std::string sszZoom = m_istop.ToText();
        const char *szZoom = sszZoom.c_str();
        if (*szZoom == '-')
220             szZoom++;
        if (*szZoom == '0'){
                szZoom++;
                if (*szZoom == '.'){
                        szZoom++;
225                     while (*szZoom == '0'){
                                nZeroes++;
                                szZoom++;
                        }
                }
230     }

        {
                Precision q(LOW_PRECISION);
                FixedFloat f(div.m_f);
235             f.precision(LOW_PRECISION);
                ToZoom(CDecNumber(FixedFloat(4 / f)), m_nZoom);
        }
        if (m_bAddReference){
                int x, y;
240             m_nTotal = 0;
                for (x = 0; x<m_nX; x++)
                for (y = 0; y<m_nY; y++)
                if (m_nPixels[x][y] == PIXEL_UNEVALUATED)
                        m_nTotal++;
245     }
```

```
        else
                m_nTotal = m_nX*m_nY;
        if (GetUseNanoMB1() && GetFractalType() == 0 && GetPower() == 2 && ! ↙
            ↳ m_bAddReference)
        {
                RenderFractalNANOMB1();
                return;
        }
        else if (GetUseNanoMB2() && GetFractalType() == 0 && GetPower() == 2 && ↙
            ↳ ! m_bAddReference)
        {
                RenderFractalNANOMB2();
                return;
        }
        else if (m_nZoom>=g_nLDBL && g_LDBL && m_nZoom <= g_nEXP && m_nPower<8)↙
            ↳ {// && !(m_nFractalType==1 && m_nPower==3)){
                if (m_db_dxr){
                        delete[] m_db_dxr;
                        m_db_dxr = NULL;
                }
                if (m_db_dxi){
                        delete[] m_db_dxi;
                        m_db_dxi = NULL;
                }
                if (m_dxr){
                        delete[] m_dxr;
                        m_dxr = NULL;
                }
                if (m_dxi){
                        delete[] m_dxi;
                        m_dxi = NULL;
                }
#ifdef KF_OPENCL
                if (cl)
                {
                        RenderFractalEXP();
                }
                else
#endif
                {
                        RenderFractalLDBL();
                }
                return;
        }
        else if (m_nZoom>=g_nLDBL){
                if (m_db_dxr){
                        delete[] m_db_dxr;
                        m_db_dxr = NULL;
                }
                if (m_db_dxi){
                        delete[] m_db_dxi;
                        m_db_dxi = NULL;
                }
                if (m_ldxr){
                        delete[] m_ldxr;
                        m_ldxr = NULL;
                }
```

```
300                     if (m_ldxi){
                                delete[] m_ldxi;
                                m_ldxi = NULL;
                        }
                        RenderFractalEXP();
305                     return;
                }
                if (m_ldxr){
                        delete[] m_ldxr;
                        m_ldxr = NULL;
310             }
                if (m_ldxi){
                        delete[] m_ldxi;
                        m_ldxi = NULL;
                }
315             if (m_dxr){
                        delete[] m_dxr;
                        m_dxr = NULL;
                }
                if (m_dxi){
320                     delete[] m_dxi;
                        m_dxi = NULL;
                }
                m_P.Init(m_nX, m_nY, m_bInteractive);
                int i;
325             if (!GetReuseReference() || !m_db_dxr || m_nZoom<g_nRefZero){
                        if (m_bAddReference != 1 || m_nZoom<g_nRefZero){
                                if (m_nZoom >= g_nRefZero){
                                        m_rref = (m_rstop + m_rstart)*.5;
                                        m_iref = (m_istop + m_istart)*.5;
330                                     g_nAddRefX = -1;
                                        g_nAddRefY = -1;
                                }
                                else{
                                        m_rref = 0;
335                                     m_iref = 0;
                                        g_nAddRefX = -1;
                                        g_nAddRefY = -1;
                                }
                        }
340                     m_nScalingOffset = 0;
                        m_nScaling = 1;
                        for (i = SCALED_DOUBLE_THRESHOLD; i<m_nZoom; i++){
                                m_nScalingOffset++;
                                m_nScaling = m_nScaling*.1;
345                     }
                        CalculateReference();
                }

                CFixedFloat step = (m_rstop - m_rstart)*(1 / (double)m_nX);
350             m_pixel_step_x = step;
                m_pixel_center_x = m_rstart + (m_nX/2) * step - m_rref;
                step = (m_istop - m_istart)*(1 / (double)m_nY);
                m_pixel_step_y = step;
                m_pixel_center_y = m_istart + (m_nY/2) * step - m_iref;
355
                m_rApprox.left = 0;
```

```
        m_rApprox.top = 0;
        m_rApprox.right = m_nX;
        m_rApprox.bottom = m_nY;
/*
        if(m_bAddReference){
                m_rApprox.left = m_nX/2;
                m_rApprox.top = m_nY/2;
                m_rApprox.right = m_nX/2;
                m_rApprox.bottom = m_nY/2;
                for (y = 0; y<m_nY; y++){
                        for (x = 0; x<m_nX; x++){
                                if(m_nPixels[x][y]==-1){
                                        if(m_rApprox.left>x)
                                                m_rApprox.left=x;
                                        if(m_rApprox.top>y)
                                                m_rApprox.top=y;
                                        if(m_rApprox.right<x)
                                                m_rApprox.right=x;
                                        if(m_rApprox.bottom<y)
                                                m_rApprox.bottom=y;
                                }
                        }
                }
        }
        else{
                m_rApprox.left = 0;
                m_rApprox.top = 0;
                m_rApprox.right = m_nX;
                m_rApprox.bottom = m_nY;
        }
*/
        CalculateApproximation(0);

        if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
                m_bNoGlitchDetection = FALSE;
        else
                m_bNoGlitchDetection = TRUE;

#ifdef KF_OPENCL
        if (cl)
        {
                RenderFractalOpenCL();
        }
        else
#endif
        {
                CalcStart();

                SYSTEM_INFO sysinfo;
                GetSystemInfo(&sysinfo);
                int nParallel = GetThreadsPerCore() * sysinfo.↙
                    ↳ dwNumberOfProcessors - GetThreadsReserveCore();
                if (nParallel < 1) nParallel = 1;
                int nStep = m_nX / nParallel;
                int nXStart = 0;

                CParallell P(
```

```
#ifdef _DEBUG
                        1
#else
                        nParallel
#endif
                        );
                TH_PARAMS *pMan = new TH_PARAMS[nParallel];
                nXStart = 0;
                for (i = 0; i<nParallel; i++){
                        pMan[i].p = this;
                        pMan[i].nXStart = nXStart;
                        nXStart += nStep;
                        if (nXStart>m_nX)
                                nXStart = m_nX;
                        if (i == nParallel - 1)
                                pMan[i].nXStop = m_nX;
                        else
                                pMan[i].nXStop = nXStart;
                        P.AddFunction((LPEXECUTE)ThMandelCalc, &pMan[i]);
                        if (pMan[i].nXStop == m_nX && pMan[i].nXStop - pMan[i].↙
                            ↳ nXStart>1 && i<nParallel - 1){
                                pMan[i].nXStop--;
                                nXStart = pMan[i].nXStop;
                        }
                        if (pMan[i].nXStop == m_nX){
                                break;
                        }
                }
                P.Execute();
                P.Reset();
                delete[] pMan;
        }
        m_bAddReference = FALSE;
        if (!m_bNoPostWhenDone)
                if (m_hWnd)
                        PostMessage(m_hWnd, WM_USER + 199, m_bStop, 0);
        m_bNoPostWhenDone = FALSE;
        m_bRunning = FALSE;
}

void CFraktalSFT::RenderFractalLDBL()
{
        m_P.Init(m_nX, m_nY, m_bInteractive);
        int i;
        if (!GetReuseReference() || !m_ldxr){
                if (m_bAddReference != 1 || m_nZoom<g_nRefZero){
                        if (m_nZoom >= g_nRefZero){
                                m_rref = (m_rstop + m_rstart)*.5;
                                m_iref = (m_istop + m_istart)*.5;
                                g_nAddRefX = -1;
                                g_nAddRefY = -1;
                        }
                        else{
                                m_rref = 0;
                                m_iref = 0;
                                g_nAddRefX = -1;
                                g_nAddRefY = -1;
```

```
                        }
                }
                m_nScalingOffsetL = 0;
                m_nScalingL = 1;
                for (i = 4900; i<m_nZoom; i++){
                        m_nScalingOffsetL++;
                        m_nScalingL = m_nScalingL*.1L;
                }
                CalculateReferenceLDBL();
        }

        CFixedFloat step = (m_rstop - m_rstart)*(1 / (double)m_nX);
        m_pixel_step_x = step;
        m_pixel_center_x = m_rstart + (m_nX/2) * step - m_rref;
        step = (m_istop - m_istart)*(1 / (double)m_nY);
        m_pixel_step_y = step;
        m_pixel_center_y = m_istart + (m_nY/2) * step - m_iref;

        m_rApprox.left = 0;
        m_rApprox.top = 0;
        m_rApprox.right = m_nX;
        m_rApprox.bottom = m_nY;
        CalculateApproximation(1);

        CalcStart();

        SYSTEM_INFO sysinfo;
        GetSystemInfo(&sysinfo);
        int nParallel = GetThreadsPerCore() * sysinfo.dwNumberOfProcessors - ↙
            ↳ GetThreadsReserveCore();
        if (nParallel < 1) nParallel = 1;

        CParallell P(
#ifdef _DEBUG
                1
#else
                nParallel
#endif
                );
        TH_PARAMS *pMan = new TH_PARAMS[nParallel];
        int nStep = m_nX / nParallel;
        if (nStep<2)
                nStep = 2;
        else
                nStep++;
        int nXStart = 0;
        for (i = 0; i<nParallel; i++){
                pMan[i].p = this;
                pMan[i].nXStart = nXStart;
                nXStart += nStep;
                if (nXStart>m_nX)
                        nXStart = m_nX;
                if (i == nParallel - 1)
                        pMan[i].nXStop = m_nX;
                else
                        pMan[i].nXStop = nXStart;
                P.AddFunction((LPEXECUTE)ThMandelCalcLDBL, &pMan[i]);
```

```
525                     if (pMan[i].nXStop == m_nX && pMan[i].nXStop - pMan[i].nXStart>1↙
                            ↳ && i<nParallel - 1){
                                pMan[i].nXStop--;
                                nXStart = pMan[i].nXStop;
                        }
                        if (pMan[i].nXStop == m_nX){
530                             break;
                        }
                }
                P.Execute();
                P.Reset();
535             delete[] pMan;
                m_bAddReference = FALSE;
                if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
                        m_bNoGlitchDetection = FALSE;
                else
540                     m_bNoGlitchDetection = TRUE;
                if (!m_bNoPostWhenDone)
                        PostMessage(m_hWnd, WM_USER + 199, m_bStop, 0);
                m_bNoPostWhenDone = FALSE;
                m_bRunning = FALSE;
545     }

        void CFraktalSFT::RenderFractalEXP()
        {
                m_P.Init(m_nX, m_nY, m_bInteractive);
550             if (!GetReuseReference() || !m_dxr){
                        if (m_bAddReference != 1 || m_nZoom<g_nRefZero){
                                if (m_nZoom >= g_nRefZero){
                                        m_rref = (m_rstop + m_rstart)*.5;
                                        m_iref = (m_istop + m_istart)*.5;
555                                     g_nAddRefX = -1;
                                        g_nAddRefY = -1;
                                }
                                else{
                                        m_rref = 0;
560                                     m_iref = 0;
                                        g_nAddRefX = -1;
                                        g_nAddRefY = -1;
                                }
                        }
565                     CalculateReferenceEXP();
                }
                int i;

                CFixedFloat step = (m_rstop - m_rstart)*(1 / (double)m_nX);
570             m_pixel_step_x = step;
                m_pixel_center_x = m_rstart + (m_nX/2) * step - m_rref;
                step = (m_istop - m_istart)*(1 / (double)m_nY);
                m_pixel_step_y = step;
                m_pixel_center_y = m_istart + (m_nY/2) * step - m_iref;
575
                m_rApprox.left = 0;
                m_rApprox.top = 0;
                m_rApprox.right = m_nX;
                m_rApprox.bottom = m_nY;
580             CalculateApproximation(2);
```

```
        CalcStart();

#ifdef KF_OPENCL
  if (cl)
  {

    RenderFractalOpenCLEXP();

        }
        else
#endif
        {

                SYSTEM_INFO sysinfo;
                GetSystemInfo(&sysinfo);
                int nParallel = GetThreadsPerCore() * sysinfo.↙
                    ↳ dwNumberOfProcessors - GetThreadsReserveCore();
                if (nParallel < 1) nParallel = 1;

                CParallell P(
#ifdef _DEBUG
                        1
#else
                        nParallel
#endif
                        );
                TH_PARAMS *pMan = new TH_PARAMS[nParallel];
                int nStep = m_nX / nParallel;
                if (nStep<2)
                        nStep = 2;
                else
                        nStep++;
                int nXStart = 0;
                for (i = 0; i<nParallel; i++){
                        pMan[i].p = this;
                        pMan[i].nXStart = nXStart;
                        nXStart += nStep;
                        if (nXStart>m_nX)
                                nXStart = m_nX;
                        if (i == nParallel - 1)
                                pMan[i].nXStop = m_nX;
                        else
                                pMan[i].nXStop = nXStart;
                        P.AddFunction((LPEXECUTE)ThMandelCalcEXP, &pMan[i]);
                        if (pMan[i].nXStop == m_nX && pMan[i].nXStop - pMan[i].↙
                            ↳ nXStart>1 && i<nParallel - 1){
                                pMan[i].nXStop--;
                                nXStart = pMan[i].nXStop;
                        }
                        if (pMan[i].nXStop == m_nX){
                                break;
                        }
                }
                P.Execute();
                P.Reset();
                delete[] pMan;
```

```
		}

		m_bAddReference = FALSE;
		if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
			m_bNoGlitchDetection = FALSE;
		else
			m_bNoGlitchDetection = TRUE;
		if (!m_bNoPostWhenDone)
			PostMessage(m_hWnd, WM_USER + 199, m_bStop, 0);
		m_bNoPostWhenDone = FALSE;
		m_bRunning = FALSE;
}

void CFraktalSFT::RenderFractalNANOMB1()
{
		m_P.Init(m_nX, m_nY, m_bInteractive);
		if (! GetReuseReference() || ! m_NanoMB1Ref || g_bJustDidNewton)
		{
			m_rref = (m_rstop + m_rstart)*.5;
			m_iref = (m_istop + m_istart)*.5;
			g_nAddRefX = -1;
			g_nAddRefY = -1;
			g_bJustDidNewton = false;
			CalculateReferenceNANOMB1();
		}
		int i;
		CFixedFloat step = (m_rstop - m_rstart)*(1 / (double)m_nX);
		m_pixel_step_x = step;
		m_pixel_center_x = m_rstart + (m_nX/2) * step - m_rref;
		step = (m_istop - m_istart)*(1 / (double)m_nY);
		m_pixel_step_y = step;
		m_pixel_center_y = m_istart + (m_nY/2) * step - m_iref;
		m_rApprox.left = 0;
		m_rApprox.top = 0;
		m_rApprox.right = m_nX;
		m_rApprox.bottom = m_nY;
		//CalculateApproximation(2);
		CalcStart();
		SYSTEM_INFO sysinfo;
		GetSystemInfo(&sysinfo);
		int nParallel = GetThreadsPerCore() * sysinfo.dwNumberOfProcessors - ↙
		    ↳ GetThreadsReserveCore();
		if (nParallel < 1) nParallel = 1;
		CParallell P(
#ifdef _DEBUG
			1
#else
			nParallel
#endif
			);
		TH_PARAMS *pMan = new TH_PARAMS[nParallel];
		int nStep = m_nX / nParallel;
		if (nStep<2)
			nStep = 2;
		else
			nStep++;
```

```
        int nXStart = 0;
        for (i = 0; i<nParallel; i++){
                pMan[i].p = this;
                pMan[i].nXStart = nXStart;
                nXStart += nStep;
                if (nXStart>m_nX)
                        nXStart = m_nX;
                if (i == nParallel - 1)
                        pMan[i].nXStop = m_nX;
                else
                        pMan[i].nXStop = nXStart;
                P.AddFunction((LPEXECUTE)ThMandelCalcNANOMB1, &pMan[i]);
                if (pMan[i].nXStop == m_nX && pMan[i].nXStop - pMan[i].nXStart>1
                    && i<nParallel - 1){
                        pMan[i].nXStop--;
                        nXStart = pMan[i].nXStop;
                }
                if (pMan[i].nXStop == m_nX){
                        break;
                }
        }
        P.Execute();
        P.Reset();
        delete[] pMan;
        m_bAddReference = FALSE;
        if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
                m_bNoGlitchDetection = FALSE;
        else
                m_bNoGlitchDetection = TRUE;
        if (!m_bNoPostWhenDone)
                PostMessage(m_hWnd, WM_USER + 199, m_bStop, 0);
        m_bNoPostWhenDone = FALSE;
        m_bRunning = FALSE;
}

void CFraktalSFT::RenderFractalNANOMB2()
{
        m_P.Init(m_nX, m_nY, m_bInteractive);
        if (! GetReuseReference() || ! m_NanoMB2Ref || g_bJustDidNewton)
        {
                m_rref = (m_rstop + m_rstart)*.5;
                m_iref = (m_istop + m_istart)*.5;
                g_nAddRefX = -1;
                g_nAddRefY = -1;
                g_bJustDidNewton = false;
                CalculateReferenceNANOMB2();
        }
        int i;
        CFixedFloat step = (m_rstop - m_rstart)*(1 / (double)m_nX);
        m_pixel_step_x = step;
        m_pixel_center_x = m_rstart + (m_nX/2) * step - m_rref;
        step = (m_istop - m_istart)*(1 / (double)m_nY);
        m_pixel_step_y = step;
        m_pixel_center_y = m_istart + (m_nY/2) * step - m_iref;
        m_rApprox.left = 0;
        m_rApprox.top = 0;
        m_rApprox.right = m_nX;
```

```
        m_rApprox.bottom = m_nY;
        //CalculateApproximation(2);
        CalcStart();
        SYSTEM_INFO sysinfo;
        GetSystemInfo(&sysinfo);
        int nParallel = GetThreadsPerCore() * sysinfo.dwNumberOfProcessors - GetThreadsReserveCore();
        if (nParallel < 1) nParallel = 1;
        CParallell P(
#ifdef _DEBUG
                1
#else
                nParallel
#endif
                );
        TH_PARAMS *pMan = new TH_PARAMS[nParallel];
        int nStep = m_nX / nParallel;
        if (nStep<2)
                nStep = 2;
        else
                nStep++;
        int nXStart = 0;
        for (i = 0; i<nParallel; i++){
                pMan[i].p = this;
                pMan[i].nXStart = nXStart;
                nXStart += nStep;
                if (nXStart>m_nX)
                        nXStart = m_nX;
                if (i == nParallel - 1)
                        pMan[i].nXStop = m_nX;
                else
                        pMan[i].nXStop = nXStart;
                P.AddFunction((LPEXECUTE)ThMandelCalcNANOMB2, &pMan[i]);
                if (pMan[i].nXStop == m_nX && pMan[i].nXStop - pMan[i].nXStart>1 && i<nParallel - 1){
                        pMan[i].nXStop--;
                        nXStart = pMan[i].nXStop;
                }
                if (pMan[i].nXStop == m_nX){
                        break;
                }
        }
        P.Execute();
        P.Reset();
        delete[] pMan;
        m_bAddReference = FALSE;
        if (m_nMaxOldGlitches && m_pOldGlitch[m_nMaxOldGlitches-1].x == -1)
                m_bNoGlitchDetection = FALSE;
        else
                m_bNoGlitchDetection = TRUE;
        if (!m_bNoPostWhenDone)
                PostMessage(m_hWnd, WM_USER + 199, m_bStop, 0);
        m_bNoPostWhenDone = FALSE;
        m_bRunning = FALSE;
}

void CFraktalSFT::CalcStart(int x0, int x1, int y0, int y1)
```

```
{
        for (int x = x0; x < x1; x++)
        {
                for (int y = y0; y < y1; ++y)
                        m_nPixels[x][y] = PIXEL_UNEVALUATED;
                memset(&m_nTrans[x][y0], 0, sizeof(m_nTrans[x][y0]) * (y1 - y0));
        }
}

static int ThCalcStart(TH_PARAMS *pMan)
{
        pMan->p->CalcStart(pMan->nXStart, pMan->nXStop, 0, pMan->p->GetHeight());
        return 0;
}

void CFraktalSFT::CalcStart()
{
        if (!m_bAddReference){
                SYSTEM_INFO sysinfo;
                GetSystemInfo(&sysinfo);
                int nParallel = GetThreadsPerCore() * sysinfo.dwNumberOfProcessors - GetThreadsReserveCore();
                if (nParallel < 1) nParallel = 1;
                CParallell P(
#ifdef _DEBUG
                        1
#else
                        nParallel
#endif
                        );
                TH_PARAMS *pMan = new TH_PARAMS[nParallel];
                int nXStart = 0;
                int nXStep = (m_nX + nParallel - 1) / nParallel;
                for (int i = 0; i < nParallel; i++)
                {
                        pMan[i].p = this;
                        pMan[i].nXStart = nXStart;
                        nXStart += nXStep;
                        if (nXStart > m_nX) nXStart = m_nX;
                        pMan[i].nXStop = nXStart;
                        P.AddFunction((LPEXECUTE)ThCalcStart, &pMan[i]);
                }
                P.Execute();
                P.Reset();
                delete[] pMan;
        }
}
```

103 fraktal_sft/resource.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen
```

```
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  //{{NO_DEPENDENCIES}}
    // Microsoft Visual C++ generated include file.
    // Used by fraktal_sft.rc
    //
    #define IDCANCEL2                       3
25  #define IDD_DIALOG1                     101
    #define IDR_MENU1                       102
    #define IDD_DIALOG2                     103
    #define IDI_ICON1                       104
    #define IDD_DIALOG3                     105
30  #define IDD_DIALOG4                     106
    #define IDD_DIALOG5                     107
    #define IDD_DIALOG6                     108
    #define IDD_DIALOG7                     109
    #define IDD_DIALOG8                     110
35  #define IDD_DIALOG9                     112
    #define IDC_CURSOR1                     113
    #define IDD_DIALOG10                    114
    #define IDD_DIALOG11                    115
    #define IDD_DIALOG12                    116
40  #define IDD_STOREZOOM                   117

    #define IDC_EDIT1                       1000
    #define IDC_EDIT3                       1001
    #define IDC_SPIN1                       1002
45  #define IDC_EDIT4                       1002
    #define IDC_EDIT9                       1003
    #define IDC_LIST1                       1004
    #define IDC_EDIT28                      1004
    #define IDC_PREV                        1005
50  #define IDC_EDIT10                      1005
    #define IDC_EDIT20                      1006
    #define IDC_EDIT12                      1007
    #define IDC_SPIN6                       1008
    #define IDC_EDIT2                       1009
55  #define IDC_BUTTON2                     1010
    #define IDC_EDIT5                       1010
    #define IDC_SHOW                        1011
    #define IDC_EDIT7                       1011
    #define IDC_BUTTON1                     1012
60  #define IDC_EDIT8                       1012
    #define IDC_EDIT6                       1013
    #define IDC_BUTTON5                     1013
```

```
#define IDC_BUTTON3 1014
#define IDC_BUTTON6 1014
#define IDC_BUTTON4 1015
#define IDC_BUTTON7 1015
#define IDC_CHECK1 1016
#define IDC_SPIN2 1017
#define IDC_CHECK9 1017
#define IDC_BUTTON8 1018
#define IDC_BUTTON9 1019
#define IDC_QUALITY_LABEL 1019
#define IDC_SPIN3 1019
#define IDC_BUTTON10 1020
#define IDC_BUTTON11 1021
#define IDC_SPIN4 1021
#define IDC_BUTTON12 1022
#define IDC_BUTTON13 1023
#define IDC_BUTTON14 1024
#define IDC_RADIO1 1025
#define IDC_SPIN5 1025
#define IDC_RADIO2 1026
#define IDC_CHECK2 1026
#define IDC_CHECK3 1027
#define IDC_RADIO3 1027
#define IDC_COMBO2 1028
#define IDC_BUTTON22 1028
#define IDC_COMBO3 1029
#define IDC_EDIT11 1029
#define IDC_BUTTON15 1030
#define IDC_EDIT13 1030
#define IDC_COMBO5 1030
#define IDC_BUTTON16 1031
#define IDC_COMBO6 1031
#define IDC_BUTTON17 1032
#define IDC_CHECK5 1032
#define IDC_BUTTON18 1033
#define IDC_COMBO1 1034
#define IDC_RADIO4 1035
#define IDC_EDIT14 1036
#define IDC_EDIT15 1037
#define IDC_BUTTON19 1038
#define IDC_EDIT16 1039
#define IDC_EDIT17 1040
#define IDC_BUTTON20 1041
#define IDC_EDIT18 1042
#define IDC_EDIT19 1043
#define IDC_BUTTON21 1044
#define IDC_RADIO5 1045
#define IDC_EDIT21 1046
#define IDC_CHECK4 1047
#define IDC_SPIN7 1048
#define IDC_EDIT22 1049
#define IDC_SPIN8 1050
#define IDC_LABEL 1050
#define IDC_CHECK8 1051
#define IDC_EDIT26 1052
#define IDC_EDIT27 1053
#define IDC_BUTTON23 1054
```

```
120 #define IDC_EDIT29                      1055
    #define IDC_LIST6                       1068
    #define IDC_FRAME6                      1069
    #define IDC_COMBO4                      1070
    #define Colors2                         1071
125 #define IDC_EDIT23                      1072
    #define IDC_BUTTON26                    1073
    #define IDC_BUTTON27                    1074
    #define IDC_BUTTON28                    1075
    #define IDC_CHECK6                      1076
130 #define IDC_EDIT24                      1077
    #define IDC_BUTTON29                    1078
    #define Colors3                         1079
    #define IDC_EDIT25                      1080
    #define IDC_CHECK7                      1081
135 #define IDC_BUTTON30                    1082
    #define IDC_DIFFERENCES                 1083
    #define IDC_EDITMAXREFS                 1084
    #define IDC_GLITCHLOWTOLERANCE          1085
    #define IDC_STOREZOOM_KFB               1086
140 #define IDC_STOREZOOM_JPG               1087
    #define IDC_STOREZOOM_PNG               1088
    #define IDC_STOREZOOM_COUNT             1089
    #define IDC_STOREZOOM_COUNTAUTO         1090
    #define IDC_JITTERSEED                  1091
145 #define IDC_JITTERSCALE                 1092
    #define IDC_JITTERSHAPE                 1093
    #define IDC_AUTOCOLOUR                  1094
    #define IDC_DERIVATIVES                 1095
    #define IDC_INTERIORCOLOR               1096
150 #define IDC_STOREZOOM_TIF               1097
    #define IDC_AUTOSKEW                    1098
    #define IDC_AUTOSKEW_BUTTON             1099
    #define IDC_AUTOSKEW_USEDZ              1100
    #define IDC_LOCATION_PERIOD             1101
155 #define IDC_BALL_PERIOD                 1102
    #define IDC_STOREZOOM_EXR               1103

    #define ID_ACTIONS_SETTINGS             40001
    #define ID_ACTIONS_EXIT                 40002
160 #define ID_ACTIONS_POSITION             40003
    #define ID_ACTIONS_ITERATIONS           40004
    #define ID_ACTIONS_SETCOLORS            40005
    #define ID_ACTIONS_ZOOMSIZE_2           40007
    #define ID_ACTIONS_ZOOMSIZE_4           40008
165 #define ID_ACTIONS_ZOOMSIZE_8           40009
    #define ID_ACTIONS_ZOOMSIZE_16          40010
    #define ID_ACTIONS_ZOOMSIZE_32          40011
    #define ID_ACTIONS_ZOOMSIZE_64          40012
    #define ID_ACTIONS_REUSEREFERENCE       40015
170 #define ID_FILE_SAVEAS_                 40018
    #define ID_FILE_OPEN_                   40020
    #define ID_FILE_SAVE_                   40021
    #define ID_ACTIONS_CREATEZOOMSEQUENCE   40022
    #define ID_ACTIONS_RESET                40023
175 #define ID_ACTIONS_FINDMINIBROT          40024
    #define ID_MENU_ABOUT                   40025
```

```
#define ID_ACTIONS_CENTERCURSOR 40026
#define ID_FILE_SAVEASJPEG 40027
#define ID_ACTIONS_ZOOMSIZE_128 40028
#define ID_ACTIONS_FINDHIGHESTITERATION 40029
#define ID_ACTIONS_ADDREFERENCE 40030
#define ID_ACTIONS_FINDCENTEROFGLITCH 40031
#define ID_ACTIONS_AUTOSOLVEGLITCHES 40032
#define ID_Menu 40033
#define ID_ACTIONSSHOWITERATIONS 40034
#define ID_ACTIONS_SPECIAL 40035
#define ID_ACTIONS_REFRESH 40036
#define ID_ACTIONS_CANCELRENDERING 40037
#define ID_FILE_STOREZOOMOUTIMAGES 40038
#define ID_ACTIONS_SPECIAL_NOAPPROXIMATION 40039
#define ID_ACTIONS_SETWINDOWSIZE 40040
#define ID_SPECIAL_NONEXACTFINDMINIBROT 40041
#define ID_FILE_SAVEMAP 40042
#define ID_ACTIONS_SETIMAGESIZE 40044
#define ID_ACTIONS_SPECIAL_SHOWERRORS 40048
#define ID_ACTIONS_SPECIAL_ADDREFERENCEERRORS 40049
#define ID_ACTIONS_SPECIAL_FINDCENTEROFGLITCHERRORS 40050
#define ID_FILE_EXAMINEZOOMSEQUENCE 40051
#define ID_FILE_RESUMEZOOMSEQUENCE 40052
#define ID_ACTIONS_SPECIAL_SETMAINREFERENCE 40053
#define ID_ACTIONS_SPECIAL_USEERRORDETECTION 40054
#define ID_ACTIONS_SPECIAL_SHOWSMOOTHTRANSITIONCOLORS 40055
#define ID_ACTIONS_SPECIAL_USELONGDOUBLEFROMSTART 40057
#define ID_ACTIONS_SPECIAL_AUTOITERATION 40058
#define ID_ACTIONS_ZOOMSIZE_1 40059
#define ID_ACTIONS_SPECIAL_SOLVEGLITCHWITHNEARPIXELSMETHOD 40060
#define ID_SPECIAL_USEFLOATEXPALWAYS 40061
#define ID_ACTIONS_SPECIAL_USEFLOATEXPALWAYS 40062
#define ID_ACTIONS_SPECIAL_SPECIAL_MIRROR1 40064
#define ID_ACTIONS_ROTATE 40066
#define ID_ACTIONS_RESETROTATION 40067
#define ID_ACTIONS_SHOWINFLECTION 40068
#define ID_ACTIONS_SPECIAL_SPECIAL_SETRATIO 40069
#define ID_ACTIONS_SPECIAL_SPECIAL_RESETRATIO 40070
#define ID_ACTIONS_SKEW 40071
#define ID_ACTIONS_ANIMATEZOOM 40072
#define ID_SPECIAL_ARBITRARYSIZE 40073
#define ID_SPECIAL_SKEWANIMATION 40074
#define ID_SPECIAL_SHOWGLITCHES 40075
#define ID_ZOOMSIZE_CUSTOM 40076
#define ID_SPECIAL_NEWTON 40077
#define ID_FILE_SAVEASPNG 40078
#define ID_FILE_OPENSETTINGS 40079
#define ID_FILE_SAVESETTINGS 40080
#define ID_ACTIONS_SPECIAL_NOREUSECENTER 40081
#define ID_ACTIONS_SPECIAL_GUESSING 40082
#define ID_ACTIONS_THREADS_1 40083
#define ID_ACTIONS_THREADS_2 40084
#define ID_ACTIONS_THREADS_4 40085
#define ID_ACTIONS_ISOLATED_0 40086
#define ID_ACTIONS_ISOLATED_4 40087
#define ID_ACTIONS_ISOLATED_8 40088
#define ID_FILE_OPENMAP 40089
```

```
#define ID_FILE_SAVEASTIF                       40090
#define ID_SPECIAL_SHOWCROSSHAIR                40091
#define ID_SPECIAL_PRESET_FAST                  40092
#define ID_SPECIAL_PRESET_BEST                  40093
#define ID_FILE_CHECKFORUPDATE                  40094
#define ID_ACTIONS_THREADS_1_4                  40095
#define ID_ACTIONS_THREADS_1_2                  40096
#define ID_ACTIONS_UNDO                         40097
#define ID_ACTIONS_REDO                         40098
#define ID_ACTIONS_COPY                         40099
#define ID_ACTIONS_PASTE                        40100
#define ID_ACTIONS_SPECIAL_USENANOMB1           40101
#define ID_ACTIONS_SPECIAL_USENANOMB2           40102
#define ID_ACTIONS_SPECIAL_INTERIORCHECKING             40103
#define ID_IMAGE_SHRINK_FAST                    40104
#define ID_IMAGE_SHRINK_DEFAULT                 40105
#define ID_IMAGE_SHRINK_BEST                    40106
#define ID_ACTIONS_THREADS_RESERVE_CORE         40107
#define ID_FILE_SAVEASEXR                       40108
#define ID_SPECIAL_HALFCOLOUR                   40109
#define ID_SPECIAL_SAVE_OVERWRITES              40110
#define ID_SPECIAL_THREADED_REFERENCE           40111
#define ID_SIMD_CHUNK_SIZE_1        40112
#define ID_SIMD_CHUNK_SIZE_2        40113
#define ID_SIMD_CHUNK_SIZE_4        40114
#define ID_SIMD_CHUNK_SIZE_8        40115
#define ID_SIMD_CHUNK_SIZE_16       40116
#define ID_SIMD_CHUNK_SIZE_32       40117
#define ID_SIMD_CHUNK_SIZE_64       40118
#define ID_SIMD_CHUNK_SIZE_128      40119
#define ID_SIMD_CHUNK_SIZE_256      40120
#define ID_SIMD_CHUNK_SIZE_512      40121
#define ID_SIMD_CHUNK_SIZE_1024     40122
#define ID_SIMD_VECTOR_SIZE_1       40123
#define ID_SIMD_VECTOR_SIZE_2       40124
#define ID_SIMD_VECTOR_SIZE_4       40125
#define ID_SIMD_VECTOR_SIZE_8       40126
#define ID_SIMD_VECTOR_SIZE_16      40127


// OpenCL
#ifdef KF_OPENCL
#define IDD_DIALOG_OPENCL             120
#define IDC_COMBO_OPENCL_DEVICE      1104
#define ID_SPECIAL_OPENCL           40128
#endif

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        121
#define _APS_NEXT_COMMAND_VALUE         40105
#define _APS_NEXT_CONTROL_VALUE         1104
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

104 fraktal_sft/scale_bitmap.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.


This file based in part on 'pixman-0.38.4/demos/scale.c':
*/
/*
 * Copyright 2012, Red Hat, Inc.
 * Copyright 2012, Soren Sandmann
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice (including the next
 * paragraph) shall be included in all copies or substantial portions of the
 * Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * Author: Soren Sandmann <soren.sandmann@gmail.com>
 */

#include <math.h>
#include <pixman.h>
#include <stdlib.h>

// based on: https://source.winehq.org/git/wine.git/blob/39935↙
    ↳ fe5ad889d537d828cc82771bdb969cdb2d4:/dlls/gdi32/gdi_private.h#l1561
static inline long long get_bitmap_stride(long long width, long long bpp)
{
```

```
  return ((width * bpp + 15LL) >> 3LL) & ~1LL;
}

extern bool scale_bitmap_rgb8
  ( unsigned char *dp, int dw, int dh
  , const unsigned char *sp, int sw, int sh
  )
{
  bool retval = false;

  int sstride = get_bitmap_stride(sw, 24);
  pixman_image_t *src = pixman_image_create_bits(PIXMAN_r8g8b8, sw, sh, (↙
      ↪ uint32_t *) sp, sstride);
  int dstride = get_bitmap_stride(dw, 24);
  pixman_image_t *dst = pixman_image_create_bits(PIXMAN_r8g8b8, dw, dh, (↙
      ↪ uint32_t *) dp, dstride);

  double fscale_x = dw / (double) sw;
  double fscale_y = dh / (double) sh;
  pixman_f_transform_t ftransform;
  pixman_f_transform_init_identity(&ftransform);
  pixman_f_transform_scale(&ftransform, nullptr, fscale_x, fscale_y);
  pixman_f_transform_invert(&ftransform, &ftransform);
  pixman_transform_t transform;
  pixman_transform_from_pixman_f_transform(&transform, &ftransform);
  pixman_image_set_transform(src, &transform);

  double sx = hypot(ftransform.m[0][0], ftransform.m[0][1]) / ftransform.m↙
      ↪ [2][2];
  double sy = hypot(ftransform.m[1][0], ftransform.m[1][1]) / ftransform.m↙
      ↪ [2][2];
  int n_params = 0;
  pixman_kernel_t k = PIXMAN_KERNEL_LANCZOS3;
  pixman_fixed_t *params = pixman_filter_create_separable_convolution
    ( &n_params
    , sx * 65536.0 + 0.5
    , sy * 65536.0 + 0.5
    , k, k // reconstruct x,y
    , k, k // sample x,y
          , 4, 4 // subsample x,y
    );
  if (! params) goto cleanup;
  if (! dst) goto cleanup;
  if (! src) goto cleanup;
  pixman_image_set_filter(src, PIXMAN_FILTER_SEPARABLE_CONVOLUTION, params, ↙
      ↪ n_params);
  pixman_image_set_repeat(src, PIXMAN_REPEAT_PAD);

  pixman_image_composite
    ( PIXMAN_OP_SRC
    , src, nullptr, dst
    , 0, 0,  0, 0,  0, 0
    , dw, dh
    );

  retval = true;
cleanup:
```

```
  if (dst) pixman_image_unref(dst);
  if (src) pixman_image_unref(src);
  if (params) free(params);
  return retval;
}
```

105 fraktal_sft/scale_bitmap.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_SCALE_BITMAP_H
#define KF_SCALE_BITMAP_H 1

extern bool scale_bitmap_rgb8
  ( unsigned char *dp, int dw, int dh
  , const unsigned char *sp, int sw, int sh
  );

#endif
```

106 fraktal_sft/Settings.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

```
*/

#include "Settings.h"

#include <cstring>
#include <cstdlib>

#include <windows.h>
#include "../common/StringVector.h"
#include "cmdline.h"
#include "jpeg.h"
#include "png.h"
#include "tiff.h"
#include "exr.h"

bool Settings::FromText(const std::string &text)
{
  char *data = strdup(text.c_str());
  CStringTable s(data, ": ", "\r\n");
  {
    int nv = s.FindString(0, "SettingsVersion");
    if (nv != -1)
    {
      int str_version_number = atoi(s[nv][1]);
      if (str_version_number > kfs_version_number)
      {
        fprintf(stderr, "WARNING: file format is newer than this EXE version\n");
      }
    }
  }
#define DOUBLE(KEY) { int n = s.FindString(0, #KEY); if (n != -1) { m_##KEY = atof(s[n][1]); } }
#define INT(KEY)    { int n = s.FindString(0, #KEY); if (n != -1) { m_##KEY = atoll(s[n][1]); } }
#define BOOL(KEY)   { int n = s.FindString(0, #KEY); if (n != -1) { m_##KEY = atoll(s[n][1]); } }
  DOUBLE(ZoomSize)
  INT(MaxReferences)
  BOOL(GlitchLowTolerance)
  BOOL(ApproxLowTolerance)
  BOOL(AutoApproxTerms)
  INT(ApproxTerms)
  INT(WindowWidth)
  INT(WindowHeight)
  INT(WindowTop)
  INT(WindowLeft)
  INT(WindowBottom)
  INT(WindowRight)
  INT(ImageWidth)
  INT(ImageHeight)
  DOUBLE(ThreadsPerCore)
  INT(ThreadsReserveCore)
  BOOL(AnimateZoom)
  BOOL(ArbitrarySize)
  BOOL(ReuseReference)
  BOOL(AutoSolveGlitches)
```

```
  BOOL(Guessing)
  BOOL(SolveGlitchNear)
  BOOL(NoApprox)
  BOOL(Mirror)
  BOOL(LongDoubleAlways)
  BOOL(FloatExpAlways)
  BOOL(AutoIterations)
  BOOL(ShowGlitches)
  BOOL(NoReuseCenter)
  INT(IsolatedGlitchNeighbourhood)
  INT(JitterSeed)
  INT(JitterShape)
  DOUBLE(JitterScale)
  BOOL(Derivatives)
  BOOL(ShowCrossHair)
  BOOL(UseNanoMB1)
  BOOL(UseNanoMB2)
  INT(OrderM)
  INT(OrderN)
  BOOL(InteriorChecking)
  DOUBLE(RadiusScale)
  INT(Shrink)
  BOOL(HalfColour)
  BOOL(SaveOverwrites)
  BOOL(ThreadedReference)
  INT(SIMDVectorSize)
  INT(SIMDChunkSize)
#undef DOUBLE
#undef INT
#undef BOOL
  std::free(data);
  return true;
}

std::string Settings::ToText() const
{
  CStringTable s;
#define DOUBLE(KEY) { char d[100]; snprintf(d, 100, "%.18g", Get##KEY()); s.↙
    ↪ AddRow(); s.AddString(s.GetCount() - 1, #KEY); s.AddString(s.GetCount() - ↙
    ↪ 1, d); }
#define INT(KEY)    { s.AddRow(); s.AddString(s.GetCount() - 1, #KEY); s.AddInt(↙
    ↪ s.GetCount() - 1, Get##KEY()); }
#define BOOL(KEY)   { s.AddRow(); s.AddString(s.GetCount() - 1, #KEY); s.AddInt(↙
    ↪ s.GetCount() - 1, Get##KEY()); }
  DOUBLE(ZoomSize)
  INT(MaxReferences)
  BOOL(GlitchLowTolerance)
  BOOL(ApproxLowTolerance)
  BOOL(AutoApproxTerms)
  INT(ApproxTerms)
  INT(WindowWidth)
  INT(WindowHeight)
  INT(WindowTop)
  INT(WindowLeft)
  INT(WindowBottom)
  INT(WindowRight)
  INT(ImageWidth)
```

```
    INT(ImageHeight)
125 DOUBLE(ThreadsPerCore)
    INT(ThreadsReserveCore)
    BOOL(AnimateZoom)
    BOOL(ArbitrarySize)
    BOOL(ReuseReference)
130 BOOL(AutoSolveGlitches)
    BOOL(Guessing)
    BOOL(SolveGlitchNear)
    BOOL(NoApprox)
    BOOL(Mirror)
135 BOOL(LongDoubleAlways)
    BOOL(FloatExpAlways)
    BOOL(AutoIterations)
    BOOL(ShowGlitches)
    BOOL(NoReuseCenter)
140 INT(IsolatedGlitchNeighbourhood)
    INT(JitterSeed)
    INT(JitterShape)
    DOUBLE(JitterScale)
    BOOL(Derivatives)
145 BOOL(ShowCrossHair)
    BOOL(UseNanoMB1)
    BOOL(UseNanoMB2)
    INT(OrderM)
    INT(OrderN)
150 BOOL(InteriorChecking)
    DOUBLE(RadiusScale)
    INT(Shrink)
    BOOL(HalfColour)
    BOOL(SaveOverwrites)
155 BOOL(ThreadedReference)
    INT(SIMDVectorSize)
    INT(SIMDChunkSize)
#undef DOUBLE
#undef INT
160 #undef BOOL
    { s.AddRow(); s.AddString(s.GetCount() - 1, "SettingsVersion"); s.AddInt(s.↙
        ↳ GetCount() - 1, kfs_version_number); }
    char *data = s.ToText(": ", "\r\n");
    std::string r(data);
    s.DeleteToText(data);
165 return r;
}

bool Settings::OpenFile(const std::string &filename)
{
170 const char *szFile = filename.c_str();
      char *szData = 0;
      char *extension = strrchr(szFile, '.');
      if (extension && 0 == strcmp(".png", extension))
      {
175           std::string filename = szFile;
              std::string comment = ReadPNGComment(filename);
              if (comment == "")
                return false;
              size_t n = comment.length() + 1;
```

```
180                 szData = new char[n];
                    strncpy(szData, comment.c_str(), n);
                    szData[n-1] = 0;
            }
            else if (extension && (0 == strcmp(".tif", extension) || 0 == strcmp(".↙
                ↳ tiff", extension)))
185         {
                    std::string filename = szFile;
                    std::string comment = ReadTIFFComment(filename);
                    if (comment == "")
                      return false;
190                 size_t n = comment.length() + 1;
                    szData = new char[n];
                    strncpy(szData, comment.c_str(), n);
                    szData[n-1] = 0;
            }
195         else if (extension && (0 == strcmp(".jpg", extension) || 0 == strcmp(".↙
                ↳ jpeg", extension)))
            {
                    std::string filename = szFile;
                    std::string comment = ReadJPEGComment(filename);
                    if (comment == "")
200                   return false;
                    size_t n = comment.length() + 1;
                    szData = new char[n];
                    strncpy(szData, comment.c_str(), n);
                    szData[n-1] = 0;
205         }
            else if (extension && (0 == strcmp(".exr", extension)))
            {
                    std::string filename = szFile;
                    std::string comment = ReadEXRComment(filename);
210                 if (comment == "")
                      return false;
                    size_t n = comment.length() + 1;
                    szData = new char[n];
                    strncpy(szData, comment.c_str(), n);
215                 szData[n-1] = 0;
            }
            else // anything else, probably .kfr/.kfs
            {
                    DWORD dw;
220                 HANDLE hFile = CreateFile(szFile, GENERIC_READ, 0, NULL, ↙
                        ↳ OPEN_EXISTING, 0, NULL);
                    if (hFile == INVALID_HANDLE_VALUE)
                            return false;
                    int nData = GetFileSize(hFile, NULL);
                    szData = new char[nData + 1];
225                 ReadFile(hFile, szData, nData, &dw, NULL);
                    CloseHandle(hFile);
                    szData[nData] = 0;
            }
      std::string szText(szData);
230   delete[] szData;
      return FromText(szText);
    }
```

```
bool Settings::SaveFile(const std::string &filename, bool overwrite) const
{
        std::string szText(ToText());
        const char *szData = szText.c_str();
  const char *szFile = filename.c_str();
       DWORD dw;
       HANDLE hFile = CreateFile(szFile, GENERIC_WRITE, 0, NULL, overwrite ? ↙
           ↳ CREATE_ALWAYS : CREATE_NEW, 0, NULL);
       if (hFile == INVALID_HANDLE_VALUE)
            return false;
       WriteFile(hFile, szData, strlen(szData), &dw, NULL);
       CloseHandle(hFile);
       return true;
}
```

107 fraktal_sft/Settings.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#ifndef KF_SETTINGS_H
#define KF_SETTINGS_H 1

#include <string>

// this sets the maximum number of references per image
#define OLD_GLITCH 10000

// this sets the range of approximation terms
// storage is O(terms^2) for R2 fractals, O(terms) for C fractals
#define MIN_APPROX_TERMS 3
#define MAX_APPROX_TERMS 63

class Settings
{

private:

  double m_ZoomSize;
  int64_t m_MaxReferences;
  bool m_GlitchLowTolerance;
```

```
    bool m_ApproxLowTolerance;
    bool m_AutoApproxTerms;
    int64_t m_ApproxTerms;
    int64_t m_WindowWidth;
    int64_t m_WindowHeight;
    int64_t m_WindowTop;
    int64_t m_WindowLeft;
    int64_t m_WindowBottom;
    int64_t m_WindowRight;
    int64_t m_ImageWidth;
    int64_t m_ImageHeight;
    double m_ThreadsPerCore;
    int64_t m_ThreadsReserveCore;
    bool m_AnimateZoom;
    bool m_ArbitrarySize;
    bool m_ReuseReference;
    bool m_AutoSolveGlitches;
    bool m_Guessing;
    bool m_SolveGlitchNear;
    bool m_NoApprox;
    bool m_Mirror;
    bool m_LongDoubleAlways;
    bool m_FloatExpAlways;
    bool m_AutoIterations;
    bool m_ShowGlitches;
    bool m_NoReuseCenter;
    int64_t m_IsolatedGlitchNeighbourhood;
    int64_t m_JitterSeed;
    int64_t m_JitterShape;
    double m_JitterScale;
    bool m_Derivatives;
    bool m_ShowCrossHair;
    bool m_UseNanoMB1;
    bool m_UseNanoMB2;
    int64_t m_OrderM;
    int64_t m_OrderN;
    bool m_InteriorChecking;
    double m_RadiusScale;
    int64_t m_Shrink;
    bool m_HalfColour;
    bool m_SaveOverwrites;
    bool m_ThreadedReference;
    int64_t m_SIMDVectorSize;
    int64_t m_SIMDChunkSize;

public:

    Settings()
    : m_ZoomSize(2.0)
    , m_MaxReferences(10000)
    , m_GlitchLowTolerance(false)
    , m_ApproxLowTolerance(false)
    , m_AutoApproxTerms(true)
    , m_ApproxTerms(10)
    , m_WindowWidth(640)
    , m_WindowHeight(360)
```

```
    , m_WindowTop(-1)
    , m_WindowLeft(-1)
    , m_WindowBottom(-1)
    , m_WindowRight(-1)
    , m_ImageWidth(640)
    , m_ImageHeight(360)
    , m_ThreadsPerCore(1)
    , m_ThreadsReserveCore(0)
    , m_AnimateZoom(true)
    , m_ArbitrarySize(true)
    , m_ReuseReference(false)
    , m_AutoSolveGlitches(true)
    , m_Guessing(true)
    , m_SolveGlitchNear(false)
    , m_NoApprox(false)
    , m_Mirror(false)
    , m_LongDoubleAlways(false)
    , m_FloatExpAlways(false)
    , m_AutoIterations(true)
    , m_ShowGlitches(true)
    , m_NoReuseCenter(true)
    , m_IsolatedGlitchNeighbourhood(4)
    , m_JitterSeed(0)
    , m_JitterShape(0)
    , m_JitterScale(1)
    , m_Derivatives(false)
    , m_ShowCrossHair(false)
    , m_UseNanoMB1(false)
    , m_UseNanoMB2(false)
    , m_OrderM(16)
    , m_OrderN(16)
    , m_InteriorChecking(false)
    , m_RadiusScale(0.1)
    , m_Shrink(1)
    , m_HalfColour(false)
    , m_SaveOverwrites(false)
    , m_ThreadedReference(true)
    , m_SIMDVectorSize(2)
    , m_SIMDChunkSize(64)
  { };

  bool FromText(const std::string &text);
  std::string ToText() const;

  bool OpenFile(const std::string &filename);
  bool SaveFile(const std::string &filename, bool overwrite) const;

  inline double GetZoomSize() const { return m_ZoomSize; };
  inline void   SetZoomSize(double z) { m_ZoomSize = z; };

  inline int64_t    GetMaxReferences() const { return m_MaxReferences; };
  inline void   SetMaxReferences(int64_t r) {
    if (r < 0) r = 0;
    if (r > OLD_GLITCH) r = OLD_GLITCH;
    m_MaxReferences = r;
  };
```

```
155     inline bool   GetGlitchLowTolerance() const { return m_GlitchLowTolerance; };
        inline void   SetGlitchLowTolerance(bool b) { m_GlitchLowTolerance = b; };

        inline bool   GetApproxLowTolerance() const { return m_ApproxLowTolerance; };
        inline void   SetApproxLowTolerance(bool b) { m_ApproxLowTolerance = b; }
160
        inline bool   GetAutoApproxTerms() const { return m_AutoApproxTerms; };
        inline void   SetAutoApproxTerms(bool b) { m_AutoApproxTerms = b; };

        inline int64_t    GetApproxTerms() const { return m_ApproxTerms; };
165     inline void   SetApproxTerms(int64_t t) {
          if (t < MIN_APPROX_TERMS) t = MIN_APPROX_TERMS;
          if (t > MAX_APPROX_TERMS) t = MAX_APPROX_TERMS;
          m_ApproxTerms = t;
        };
170
        inline int64_t    GetWindowWidth() const { return m_WindowWidth; };
        inline void   SetWindowWidth(int64_t w) { m_WindowWidth = w; };

        inline int64_t    GetWindowHeight() const { return m_WindowHeight; };
175     inline void   SetWindowHeight(int64_t h) { m_WindowHeight = h; };

        inline int64_t    GetWindowTop() const { return m_WindowTop; };
        inline void   SetWindowTop(int64_t w) { m_WindowTop = w; };

180     inline int64_t    GetWindowLeft() const { return m_WindowLeft; };
        inline void   SetWindowLeft(int64_t w) { m_WindowLeft = w; };

        inline int64_t    GetWindowBottom() const { return m_WindowBottom; };
        inline void   SetWindowBottom(int64_t w) { m_WindowBottom = w; };
185
        inline int64_t    GetWindowRight() const { return m_WindowRight; };
        inline void   SetWindowRight(int64_t w) { m_WindowRight = w; };

        inline int64_t    GetImageWidth() const { return m_ImageWidth; };
190     inline void   SetImageWidth(int64_t w) { m_ImageWidth = w; };

        inline int64_t    GetImageHeight() const { return m_ImageHeight; };
        inline void   SetImageHeight(int64_t h) { m_ImageHeight = h; };

195     inline double GetThreadsPerCore() const { return m_ThreadsPerCore; };
        inline void   SetThreadsPerCore(double t) { m_ThreadsPerCore = t; };

        inline int64_t    GetThreadsReserveCore() const { return m_ThreadsReserveCore;↙
            ↳ };
        inline void   SetThreadsReserveCore(int64_t t) { m_ThreadsReserveCore = t; };
200
        inline bool   GetAnimateZoom() const { return m_AnimateZoom; };
        inline void   SetAnimateZoom(bool b) { m_AnimateZoom = b; };

        inline bool   GetArbitrarySize() const { return m_ArbitrarySize; };
205     inline void   SetArbitrarySize(bool b) { m_ArbitrarySize = b; };

        inline bool   GetReuseReference() const { return m_ReuseReference; };
        inline void   SetReuseReference(bool b) { m_ReuseReference = b; };

210     inline bool   GetAutoSolveGlitches() const { return m_AutoSolveGlitches; };
```

```
    inline void   SetAutoSolveGlitches(bool b) { m_AutoSolveGlitches = b; };

    inline bool   GetGuessing() const { return m_Guessing; };
    inline void   SetGuessing(bool b) { m_Guessing = b; };

    inline bool   GetSolveGlitchNear() const { return m_SolveGlitchNear; };
    inline void   SetSolveGlitchNear(bool b) { m_SolveGlitchNear = b; };

    inline bool   GetNoApprox() const { return m_NoApprox; };
    inline void   SetNoApprox(bool b) { m_NoApprox = b; };

    inline bool   GetMirror() const { return m_Mirror; };
    inline void   SetMirror(bool b) { m_Mirror = b; };

    inline bool   GetLongDoubleAlways() const { return m_LongDoubleAlways; };
    inline void   SetLongDoubleAlways(bool b) { m_LongDoubleAlways = b; };

    inline bool   GetFloatExpAlways() const { return m_FloatExpAlways; };
    inline void   SetFloatExpAlways(bool b) { m_FloatExpAlways = b; };

    inline bool   GetAutoIterations() const { return m_AutoIterations; };
    inline void   SetAutoIterations(bool b) { m_AutoIterations = b; };

    inline bool   GetShowGlitches() const { return m_ShowGlitches; };
    inline void   SetShowGlitches(bool b) { m_ShowGlitches = b; };

    inline bool   GetNoReuseCenter() const { return m_NoReuseCenter; };
    inline void   SetNoReuseCenter(bool b) { m_NoReuseCenter = b; };

    inline int64_t    GetIsolatedGlitchNeighbourhood() const { return ↙
        ↳ m_IsolatedGlitchNeighbourhood; };
    inline void   SetIsolatedGlitchNeighbourhood(int64_t n) { ↙
        ↳ m_IsolatedGlitchNeighbourhood = n; };

    inline int64_t    GetJitterSeed() const { return m_JitterSeed; };
    inline void   SetJitterSeed(int64_t n) { m_JitterSeed = n; };

    inline int64_t    GetJitterShape() const { return m_JitterShape; };
    inline void   SetJitterShape(int64_t n) { m_JitterShape = 0 <= n && n <= 1 ? n↙
        ↳ : 0; };

    inline double GetJitterScale() const { return m_JitterScale; };
    inline void   SetJitterScale(double n) { m_JitterScale = n; };

    inline bool   GetDerivatives() const { return m_Derivatives; };
    inline void   SetDerivatives(bool b) { m_Derivatives = b; };

    inline bool   GetShowCrossHair() const { return m_ShowCrossHair; };
    inline void   SetShowCrossHair(bool b) { m_ShowCrossHair = b; };

    inline bool   GetUseNanoMB1() const { return m_UseNanoMB1; };
    inline void   SetUseNanoMB1(bool b) { m_UseNanoMB1 = b; };

    inline bool   GetUseNanoMB2() const { return m_UseNanoMB2; };
    inline void   SetUseNanoMB2(bool b) { m_UseNanoMB2 = b; };

    inline int64_t    GetOrderM() const { return m_OrderM; };
```

```
265     inline void   SetOrderM(int64_t t)
        {
          if (t < 1) t = 1;
          if (t > MAX_APPROX_TERMS) t = MAX_APPROX_TERMS;
          m_OrderM = t;
270     };

        inline int64_t    GetOrderN() const { return m_OrderN; };
        inline void   SetOrderN(int64_t t)
        {
275       if (t < 1) t = 1;
          if (t > MAX_APPROX_TERMS) t = MAX_APPROX_TERMS;
          m_OrderN = t;
        };

280     inline bool   GetInteriorChecking() const { return m_InteriorChecking; };
        inline void   SetInteriorChecking(bool b) { m_InteriorChecking = b; };

        inline double GetRadiusScale() const { return m_RadiusScale; };
        inline void   SetRadiusScale(double b) { m_RadiusScale = b; };
285
        inline int64_t    GetShrink() const { return m_Shrink; };
        inline void   SetShrink(int64_t n) { m_Shrink = 0 <= n && n <= 2 ? n : 1; };

        inline bool   GetHalfColour() const { return m_HalfColour; };
290     inline void   SetHalfColour(bool b) { m_HalfColour = b; };

        inline bool   GetSaveOverwrites() const { return m_SaveOverwrites; };
        inline void   SetSaveOverwrites(bool b) { m_SaveOverwrites = b; };

295     inline bool   GetThreadedReference() const { return m_ThreadedReference; };
        inline void   SetThreadedReference(bool b) { m_ThreadedReference = b; };

        inline int64_t    GetSIMDChunkSize() const { return m_SIMDChunkSize; };
        inline void   SetSIMDChunkSize(int64_t n) { m_SIMDChunkSize = 0 < n ? n : 1; ↙
            ↳ };
300
        inline int64_t    GetSIMDVectorSize() const { return m_SIMDVectorSize; };
        inline void   SetSIMDVectorSize(int64_t n) {
          if (n > 1 << KF_SIMD) n = 1 << KF_SIMD;
          switch (n)
305       {
            case 2: case 4: case 8: case 16: m_SIMDVectorSize = n; break;
            default: m_SIMDVectorSize = 1; break;
          }
        }
310
    };

    #endif
```

108 fraktal_sft/tiff.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen
```

```
5
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include <stdio.h>
    #include <string.h>
    #include <time.h>
    #include <string>
    #include <tiffio.h>
25
    #include "tiff.h"

    extern int SaveTIFF(const std::string &szFileName, char *Data, int nHeight, int ↙
        ↪ nWidth, int nColors, const std::string &comment)
    {
30          bool ok = true;
            TIFF *tif = TIFFOpen(szFileName.c_str(), "w");
            if (!tif) return 0;
            ok &= 1 == TIFFSetField(tif, TIFFTAG_IMAGEDESCRIPTION, comment.c_str());
            ok &= 1 == TIFFSetField(tif, TIFFTAG_IMAGEWIDTH, nWidth);
35          ok &= 1 == TIFFSetField(tif, TIFFTAG_IMAGELENGTH, nHeight);
            ok &= 1 == TIFFSetField(tif, TIFFTAG_SAMPLESPERPIXEL, nColors);
            ok &= 1 == TIFFSetField(tif, TIFFTAG_BITSPERSAMPLE, 8);
            ok &= 1 == TIFFSetField(tif, TIFFTAG_ORIENTATION, ORIENTATION_TOPLEFT);
            ok &= 1 == TIFFSetField(tif, TIFFTAG_PLANARCONFIG, PLANARCONFIG_CONTIG);
40          ok &= 1 == TIFFSetField(tif, TIFFTAG_PHOTOMETRIC, PHOTOMETRIC_RGB);
            ok &= 1 == TIFFSetField(tif, TIFFTAG_ROWSPERSTRIP, TIFFDefaultStripSize(↙
                ↪ tif, nWidth * nColors));
            for (int y = 0; y < nHeight; ++y)
            {
                    ok &= 1 == TIFFWriteScanline(tif, Data + y * nWidth * nColors, y↙
                        ↪ , 0);
45          }
            TIFFClose(tif);
            return ok ? 1 : 0;
    }

50  extern std::string ReadTIFFComment(const std::string &filename)
    {
            auto e = TIFFSetErrorHandler(0);
            auto w = TIFFSetWarningHandler(0);
            bool ok = true;
55          TIFF *tif = TIFFOpen(filename.c_str(), "r");
            if (!tif)
            {
                    TIFFSetErrorHandler(e);
```

```
                TIFFSetWarningHandler(w);
60              return "";
        }
        void *data = nullptr;
        ok &= 1 == TIFFGetField(tif, TIFFTAG_IMAGEDESCRIPTION, &data);
        if (ok)
65      {
                std::string comment((const char *) data);
                TIFFClose(tif);
                TIFFSetErrorHandler(e);
                TIFFSetWarningHandler(w);
70              return comment;
        }
        TIFFClose(tif);
        TIFFSetErrorHandler(e);
        TIFFSetWarningHandler(w);
75      return "";
}
```

109 fraktal_sft/tiff.h

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2018 Claude Heiland-Allen
5

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
10

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.
15

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

20 #ifndef KF_TIFF_H
#define KF_TIFF_H 1

#include <string>

25 extern int SaveTIFF(const std::string &szFileName, char *Data, int nHeight, int ↙
    ↳ nWidth, int nColors, const std::string &comment);
extern std::string ReadTIFFComment(const std::string &filename);

#endif
```

110 .gitignore

```
kf.exe
kf.32.exe
kf.64.exe
```

```
kf-tile.exe
kf-tile.32.exe
kf-tile.64.exe
*.o
*.d
*.hi
*.jpg
*.kfr
*.kfs
*.kfb
*.f32
*.pdf
*.pgm
*.ppm
*.png
*.log
*.mdz
kf.rec
preprocessor
formula/formula.cpp
cl/formula.cpp
cl/*_cl.c
vm/
-
examples/examples.html
```

111 LICENSE.md

```
### GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007

Copyright (C) 2007 Free Software Foundation, Inc.
<https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

### Preamble

The GNU Affero General Public License is a free, copyleft license for
software and other kinds of works, specifically designed to ensure
cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed
to take away your freedom to share and change the works. By contrast,
our General Public Licenses are intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains
free software for all its users.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.
```

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a

computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same

work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the

written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

315

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or
320 installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

325 Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

330

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions.
335 Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by
340 this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own
345 removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you
350 add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
355 - b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in
360 reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
365 - f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

370

All other non-permissive additional terms are considered "further

restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means,

then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the

Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,

INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.
>
> You should have received a copy of the GNU Affero General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

```
You should also get your employer (if you work as a programmer) or
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. For more information on this, and how to apply and follow
the GNU AGPL, see <https://www.gnu.org/licenses/>.
```

112 Makefile

```
# Kalles Fraktaler 2
# Copyright (C) 2013-2017 Karl Rummo
# Copyright (C) 2017-2019 Claude Heiland-Allen
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
# published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.

# make SYSTEM=native for 64bit (most optimized, non-portable)
# make SYSTEM=64+    for 64bit with recent SIMD extensions
# make SYSTEM=64     for 64bit
# make SYSTEM=32     for 32bit (experimental)
SYSTEM ?= native
include $(SYSTEM).mk

CLEWPREFIX := $(WINPREFIX)/src/clew

FLAGS := -Wall -Wextra -Wno-missing-field-initializers -Wno-unused-parameter -↙
    ↪ Wno-unused-function -Wno-cast-function-type -Wno-deprecated-copy -pipe -↙
    ↪ MMD -g -O3 -fno-var-tracking-assignments -I$(WINPREFIXPLUS)/include -I$(↙
    ↪ WINPREFIX)/include -I$(WINPREFIX)/include/pixman-1 -I$(WINPREFIX)/include/↙
    ↪ OpenEXR -D_FILE_OFFSET_BITS=64 -DKF_SIMD=$(SIMD) -I$(CLEWPREFIX)/include
LINK_FLAGS := -static-libgcc -static-libstdc++ -Wl,--stack,67108864 -Wl,-↙
    ↪ subsystem,windows -L$(WINPREFIXPLUS)/lib -L$(WINPREFIX)/lib
LIBS := -lgdi32 -lcomdlg32 -lole32 -loleaut32 -lcomctl32 -lwininet -lurlmon -↙
    ↪ luuid -lmpfr -lgmp -ljpeg -ltiff -lpixman-1 $(WINPREFIX)/lib/libpng16.a -↙
    ↪ lz -lgsl -lgslcblas -lIlmImf-2_4 -lImath-2_4 -lHalf-2_4 -lIex-2_4 -↙
    ↪ lIexMath-2_4 -lIlmThread-2_4 -lz

FRAKTAL_SOURCES_CPP = \
fraktal_sft/CDecNumber.cpp \
fraktal_sft/CFixedFloat.cpp \
fraktal_sft/check_for_update.cpp \
fraktal_sft/cmdline.cpp \
fraktal_sft/double_perturbation.cpp \
fraktal_sft/double_reference.cpp \
fraktal_sft/exr.cpp \
fraktal_sft/floatexp_approximation.cpp \
fraktal_sft/floatexp_perturbation.cpp \
fraktal_sft/floatexp_reference.cpp \
fraktal_sft/fraktal_sft.cpp \
```

```
    fraktal_sft/gradient.cpp \
    fraktal_sft/jpeg.cpp \
45  fraktal_sft/listbox.cpp \
    fraktal_sft/long_double_perturbation.cpp \
    fraktal_sft/long_double_reference.cpp \
    fraktal_sft/main.cpp \
    fraktal_sft/main_color.cpp \
50  fraktal_sft/main_examine.cpp \
    fraktal_sft/main_iterations.cpp \
    fraktal_sft/main_position.cpp \
    fraktal_sft/nanomb1_perturbation.cpp \
    fraktal_sft/nanomb1_reference.cpp \
55  fraktal_sft/nanomb2_perturbation.cpp \
    fraktal_sft/nanomb2_reference.cpp \
    fraktal_sft/newton.cpp \
    fraktal_sft/Parameter.cpp \
    fraktal_sft/png.cpp \
60  fraktal_sft/render.cpp \
    fraktal_sft/scale_bitmap.cpp \
    fraktal_sft/Settings.cpp \
    fraktal_sft/tiff.cpp

65  FRAKTAL_SOURCES_H = \
    fraktal_sft/CDecNumber.h \
    fraktal_sft/CFixedFloat.h \
    fraktal_sft/check_for_update.h \
    fraktal_sft/cmdline.h \
70  fraktal_sft/colour.h \
    fraktal_sft/complex.h \
    fraktal_sft/exr.h \
    fraktal_sft/floatexp.h \
    fraktal_sft/fraktal_sft.h \
75  fraktal_sft/gradient.h \
    fraktal_sft/itercount_array.h \
    fraktal_sft/jpeg.h \
    fraktal_sft/listbox.h \
    fraktal_sft/main.h \
80  fraktal_sft/main_color.h \
    fraktal_sft/main_examine.h \
    fraktal_sft/main_iterations.h \
    fraktal_sft/main_position.h \
    fraktal_sft/nanomb1.h \
85  fraktal_sft/nanomb1.inc \
    fraktal_sft/nanomb2.h \
    fraktal_sft/nanomb2.inc \
    fraktal_sft/newton.h \
    fraktal_sft/png.h \
90  fraktal_sft/resource.h \
    fraktal_sft/scale_bitmap.h \
    fraktal_sft/Settings.h \
    fraktal_sft/tiff.h

95  COMMON_SOURCES_CPP = \
    common/FolderBrowser.cpp \
    common/getimage.cpp \
    common/parallell.cpp \
    common/StringVector.cpp \
```

```
100 common/tooltip.cpp \
    common/bitmap.cpp \
    common/matrix.cpp

    COMMON_SOURCES_H = \
105 common/FolderBrowser.h \
    common/getimage.h \
    common/parallell.h \
    common/StringVector.h \
    common/tooltip.h \
110 common/bitmap.h \
    common/matrix.h

    FORMULA_SOURCES_CPP = formula/formula.cpp

115 UTILS_SOURCES_CPP = utils/kf-tile.cpp

    OPENCL_SOURCES_CPP = cl/opencl.cpp cl/formula.cpp
    OPENCL_SOURCES_C = $(CLEWPREFIX)/src/clew.c
    OPENCL_SOURCES_H = cl/opencl.h $(CLEWPREFIX)/include/clew.h
120
    SOURCES_CPP = $(FRAKTAL_SOURCES_CPP) $(COMMON_SOURCES_CPP)
    SOURCES_C = $(wildcard formula/generated/*.c)
    SOURCES_H = $(FRAKTAL_SOURCES_H) $(COMMON_SOURCES_H)

125 FLAGS_WINDRES = -DKF_SIMD=$(SIMD)

    ifeq ($(OPENCL), 1)
    FLAGS += -DKF_OPENCL
    SOURCES_CPP += $(OPENCL_SOURCES_CPP)
130 SOURCES_C += $(OPENCL_SOURCES_C)
    SOURCES_H += $(OPENCL_SOURCES_H)
    FLAGS_WINDRES += -DKF_OPENCL=1
    endif

135 COMPILE_FLAGS := -xc++ -std=c++11 $(FLAGS)

    SOURCES = $(SOURCES_CPP) $(SOURCES_C) $(SOURCES_H)

    OBJECTS_CPP := $(patsubst %.cpp,%.o,$(SOURCES_CPP)) formula/formula.1.o formula/↙
        ↪ formula.2.o formula/formula.3.o formula/formula.4.o formula/formula.5.o ↙
        ↪ formula/formula.6.o formula/formula.7.o formula/formula.8.o formula/↙
        ↪ formula.9.o formula/formula.a.o
140 OBJECTS_C := $(patsubst %.c,%.o,$(SOURCES_C))
    OBJECTS := $(OBJECTS_CPP) $(OBJECTS_C) res.o

    DEPENDS := $(patsubst %.o,%.d,$(OBJECTS))

145 all: kf.exe kf-tile.exe

    clean:
            rm -f $(OBJECTS) $(DEPENDS) $(FORMULA_SOURCES_CPP) $(patsubst %.cpp,%.o,↙
                ↪ $(UTILS_SOURCES_CPP))
            rm -f cl/formula.cpp cl/common_cl.c cl/double_pre_cl.c cl/double_post_cl↙
                ↪ .c cl/double_post_rc_cl.c cl/double_post_m_cl.c cl/floatexp_pre_cl↙
                ↪ .c cl/floatexp_post_cl.c cl/floatexp_post_rc_cl.c cl/↙
                ↪ floatexp_post_m_cl.c
```

```
150         rm -f preprocessor preprocessor.hi preprocessor.o

    kf.exe: $(OBJECTS)
        $(LINK) -o kf.exe $(OBJECTS) $(LINK_FLAGS) $(LIBS)

155 kf-tile.exe: utils/kf-tile.o
        $(LINK) -o kf-tile.exe utils/kf-tile.o -static-libgcc -static-libstdc++ ↙
            ↳ -L$(WINPREFIX)/lib -lmpfr -lgmp

    res.o: fraktal_sft/fraktal_sft.rc fraktal_sft/resource.h
        $(WINDRES) -i fraktal_sft/fraktal_sft.rc -o res.o $(FLAGS_WINDRES)
160
    %.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -o $@ -c $<

    %.o: %.c
165         $(COMPILE) $(COMPILE_FLAGS) -o $@ -c $<

    %.1.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS1 -o $@ -c $<

170 %.2.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS2 -o $@ -c $<

    %.3.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS3 -o $@ -c $<
175
    %.4.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS4 -o $@ -c $<

    %.5.o: %.cpp
180         $(COMPILE) $(COMPILE_FLAGS) -DPASS5 -o $@ -c $<

    %.6.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS6 -o $@ -c $<

185 %.7.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS7 -o $@ -c $<

    %.8.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS8 -o $@ -c $<
190
    %.9.o: %.cpp
        $(COMPILE) $(COMPILE_FLAGS) -DPASS9 -o $@ -c $<

    %.a.o: %.cpp
195         $(COMPILE) $(COMPILE_FLAGS) -DPASSA -o $@ -c $<

    formula/formula.cpp: formula/common.cpp formula/formula.xsl formula/formula.xml ↙
        ↳ preprocessor
        ( echo "/* this file is autogenerated, do not edit directly */" ; cat ↙
            ↳ formula/common.cpp ; $(XSLTPROC) formula/formula.xsl formula/↙
            ↳ formula.xml | ./preprocessor ) > $@

200 cl/%_cl.c: cl/%.cl cl/s2c.sh
        ./cl/s2c.sh perturbation_opencl_$* < $< > $@
```

```
cl/formula.cpp: cl/formula.xsl formula/formula.xml preprocessor
	( $(XSLTPROC) cl/formula.xsl formula/formula.xml | ./preprocessor ) > $@

preprocessor: preprocessor.hs
	ghc -package parsec -package mtl -O preprocessor.hs

cl/opencl.o: cl/opencl.cpp cl/opencl.h

cl/formula.o: cl/formula.cpp cl/common_cl.c cl/softfloat_cl.c cl/double_pre_cl.c cl/double_post_cl.c cl/double_post_rc_cl.c cl/double_post_m_cl.c cl/floatexp_pre_cl.c cl/floatexp_post_cl.c cl/floatexp_post_rc_cl.c cl/floatexp_post_m_cl.c cl/softfloat_pre_cl.c cl/softfloat_post_cl.c cl/softfloat_post_rc_cl.c cl/softfloat_post_m_cl.c

%.pdf: %.md
	pandoc -f markdown -t latex -V "papersize=a4" -V "geometry=margin=1in" < $< -o $@

-include $(DEPENDS)
```

113 native.mk

```
WINPREFIXPLUS ?= $(HOME)/win64+
WINPREFIX ?= $(HOME)/win64
SIMD ?= 4
COMPILE ?= x86_64-w64-mingw32-g++ -march=native
LINK ?= x86_64-w64-mingw32-g++
WINDRES ?= x86_64-w64-mingw32-windres
AR ?= x86_64-w64-mingw32-ar
AR2 ?= x86_64-w64-mingw32-ranlib
XSLTPROC ?= xsltproc
RM ?= rm -f
GCC ?= x86_64-w64-mingw32-gcc
```

114 openexr-2.4.0.patch

```
diff -ruw openexr-2.4.0.old/IlmBase/IlmThread/IlmThread.h openexr-2.4.0.new/IlmBase/IlmThread/IlmThread.h
--- openexr-2.4.0.old/IlmBase/IlmThread/IlmThread.h	2019-09-18 02:02:06.000000000 +0100
+++ openexr-2.4.0.new/IlmBase/IlmThread/IlmThread.h	2019-09-23 13:00:01.447911162 +0100
@@ -106,7 +106,7 @@
 #	include <pthread.h>
 #   endif
 #else
-#   include <thread>
+#   include <mingw-std-threads/mingw.thread.h>
 #endif

 ILMTHREAD_INTERNAL_NAMESPACE_HEADER_ENTER
diff -ruw openexr-2.4.0.old/IlmBase/IlmThread/IlmThreadMutex.h openexr-2.4.0.new/IlmBase/IlmThread/IlmThreadMutex.h
--- openexr-2.4.0.old/IlmBase/IlmThread/IlmThreadMutex.h	2019-09-18 02:02:06.000000000 +0100
+++ openexr-2.4.0.new/IlmBase/IlmThread/IlmThreadMutex.h	2019-09-23 13:01:04.368243384 +0100
```

```
@@ -81,7 +81,7 @@
 #       include <pthread.h>
 #   endif
 #else
-#   include <mutex>
+#   include <mingw-std-threads/mingw.mutex.h>
 #endif

 ILMTHREAD_INTERNAL_NAMESPACE_HEADER_ENTER
diff -ruw openexr-2.4.0.old/IlmBase/IlmThread/IlmThreadPool.cpp openexr-2.4.0.↙
    ↪ new/IlmBase/IlmThread/IlmThreadPool.cpp
--- openexr-2.4.0.old/IlmBase/IlmThread/IlmThreadPool.cpp        2019-09-18 ↙
    ↪ 02:02:06.000000000 +0100
+++ openexr-2.4.0.new/IlmBase/IlmThread/IlmThreadPool.cpp        2019-09-23 ↙
    ↪ 12:59:44.051818967 +0100
@@ -47,7 +47,7 @@
 #ifndef ILMBASE_FORCE_CXX03
 # include <memory>
 # include <atomic>
-# include <thread>
+# include <mingw-std-threads/mingw.thread.h>
 #else
 # ifndef _WIN32
 #  include <unistd.h>
diff -ruw openexr-2.4.0.old/IlmBase/IlmThread/IlmThreadSemaphore.h openexr↙
    ↪ -2.4.0.new/IlmBase/IlmThread/IlmThreadSemaphore.h
--- openexr-2.4.0.old/IlmBase/IlmThread/IlmThreadSemaphore.h     2019-09-18 ↙
    ↪ 02:02:06.000000000 +0100
+++ openexr-2.4.0.new/IlmBase/IlmThread/IlmThreadSemaphore.h     2019-09-23 ↙
    ↪ 13:00:45.276142774 +0100
@@ -62,8 +62,8 @@
 #         include <pthread.h>
 #       endif
 #   else
-#       include <mutex>
-#       include <condition_variable>
+#       include <mingw-std-threads/mingw.thread.h>
+#       include <mingw-std-threads/mingw.condition_variable.h>
 #   endif
 #endif

diff -ruw openexr-2.4.0.old/OpenEXR/IlmImf/dwaLookups.cpp openexr-2.4.0.new/↙
    ↪ OpenEXR/IlmImf/dwaLookups.cpp
--- openexr-2.4.0.old/OpenEXR/IlmImf/dwaLookups.cpp      2019-09-18 ↙
    ↪ 02:02:06.000000000 +0100
+++ openexr-2.4.0.new/OpenEXR/IlmImf/dwaLookups.cpp      2019-09-23 ↙
    ↪ 12:59:31.075750097 +0100
@@ -46,7 +46,7 @@
 #include <OpenEXRConfigInternal.h>

 #if __cplusplus >= 201103L
-#include <thread>
+#include <mingw-std-threads/mingw.thread.h>
 #endif

 #ifdef OPENEXR_IMF_HAVE_SYSCONF_NPROCESSORS_ONLN
```

115 prepare.sh

```
#!/bin/sh
# Kalles Fraktaler 2
# Copyright (C) 2013-2017 Karl Rummo
# Copyright (C) 2017-2019 Claude Heiland-Allen
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
# published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
set -e
NCPUS="$(( $(nproc) * 2 ))"
export CPPFLAGS=-D__USE_MINGW_ANSI_STDIO
export LDFLAGS="-static-libgcc -static-libstdc++"
if [ "x$1" = "xdl" ]
then
mkdir -p ~/win64/src
mkdir -p ~/win32/src
cp -avft ~/win64/src *.patch
cp -avft ~/win32/src *.patch
# download
cd ~/win64/src
wget -c https://dl.bintray.com/boostorg/release/1.71.0/source/boost_1_71_0.7z
wget -c https://gmplib.org/download/gmp/gmp-6.1.2.tar.lz
wget -c https://www.mpfr.org/mpfr-current/mpfr-4.0.2.tar.xz
wget -c https://www.mpfr.org/mpfr-current/allpatches
wget -c https://zlib.net/zlib-1.2.11.tar.xz
wget -c https://jpegclub.org/support/files/jpegsrc.v6b2.tar.gz
wget -c https://download.sourceforge.net/libpng/libpng-1.6.37.tar.xz
wget -c https://download.osgeo.org/libtiff/tiff-4.0.10.tar.gz
wget -c https://ftp.gnu.org/gnu/gsl/gsl-2.6.tar.gz
wget -c https://www.cairographics.org/releases/pixman-0.38.4.tar.gz
wget -c https://github.com/g-truc/glm/releases/download/0.9.9.5/glm-0.9.9.5.7z
wget -c https://github.com/openexr/openexr/archive/v2.4.0.tar.gz -O openexr
    ↪ -2.4.0.tar.gz
git clone https://github.com/meganz/mingw-std-threads.git || ( cd mingw-std-
    ↪ threads && git pull )
git clone https://github.com/martijnberger/clew.git || ( cd clew && git pull )
cp -avft ~/win32/src *z allpatches mingw-std-threads clew
elif [ "x$1" = "x64" ]
then
# gmp 64
cd ~/win64/src
tar xf gmp-*.tar.lz
cd gmp-*/
CC_FOR_BUILD="gcc" CPP_FOR_BUILD="gcc -E" ./configure --build=x86_64-pc-linux-
    ↪ gnu --host=x86_64-w64-mingw32 --prefix=$HOME/win64
make -j $NCPUS
```

```
make install
make check
# mpfr 64
cd ~/win64/src
tar xf mpfr-*.tar.xz
cd mpfr-*/
patch -N -Z -p1 < ../allpatches
./configure --host=x86_64-w64-mingw32 --prefix=$HOME/win64 --with-gmp-build=../gmp-6.1.2 --enable-static --disable-shared
make -j $NCPUS
make install
make check
# zlib 64
cd ~/win64/src
tar xf zlib-*.tar.xz
cd zlib-*/
CC=x86_64-w64-mingw32-gcc ./configure --static --prefix=$HOME/win64
CC=x86_64-w64-mingw32-gcc make -j $NCPUS
CC=x86_64-w64-mingw32-gcc make install
# png 64
cd ~/win64/src
tar xf libpng-*.tar.xz
cd libpng-*/
./configure --disable-shared --host=x86_64-w64-mingw32 CPPFLAGS="$CPPFLAGS -I$HOME/win64/include" LDFLAGS="$LDFLAGS -L$HOME/win64/lib" --prefix=$HOME/win64
make -j $NCPUS
make install
# jpeg 64
cd ~/win64/src
tar xf jpegsrc.v6b2.tar.gz
cd jpeg-6b2
./configure --disable-shared --host=x86_64-w64-mingw32 --prefix=$HOME/win64
make -j $NCPUS
make install
# tiff 64
cd ~/win64/src
tar xf tiff-*.tar.gz
cd tiff-*/
./configure --disable-shared --host=x86_64-w64-mingw32 --prefix=$HOME/win64
make -j $NCPUS
make install
# gsl 64
cd ~/win64/src
tar xf gsl-*.tar.gz
cd gsl-*/
./configure --disable-shared --host=x86_64-w64-mingw32 --prefix=$HOME/win64
make -j $NCPUS
make install
ln -s gsl-histogram.exe gsl-histogram # hack for test suite
make check
# pixman 64
cd ~/win64/src
tar xf pixman-*.tar.gz
cd pixman-*/
CC=x86_64-w64-mingw32-gcc LDFLAGS=-L$HOME/win64/lib ./configure --disable-shared --disable-openmp --prefix=$HOME/win64
```

```
    make -j $NCPUS
    make install
    make check || echo "expected 1 FAIL (thread-test)"
    # boost 64
110 cd ~/win64/src
    7zr x boost*.7z
    cd ~/win64/include
    rm -f boost
    ln -s ../src/boost*/boost/
115 # glm 64
    cd ~/win64/src
    7zr x glm*.7z
    cd ~/win64/include
    rm -f glm
120 ln -s ../src/glm*/glm/
    # mingw-std-threads 64
    cd ~/win64/include
    rm -f mingw-std-threads
    ln -s ../src/mingw-std-threads
125 # openexr 64
    cd ~/win64/src
    tar xf openexr-*.tar.gz
    cd openexr-*/
    patch -p1 < $(ls ../openexr-*.patch)
130 mkdir -p build
    cd build
    cmake -DBUILD_SHARED_LIBS=OFF -DCMAKE_TOOLCHAIN_FILE=../cmake/Toolchain-mingw.↙
        ↪ cmake -DCMAKE_CXX_FLAGS=-I$HOME/win64/include -DZLIB_INCLUDE_DIR=$HOME/↙
        ↪ win64/include -DZLIB_LIBRARY=$HOME/win64/lib/libz.a -DCMAKE_INSTALL_PREFIX↙
        ↪ =$HOME/win64 ..
    make -j $NCPUS
    make install
135 # clew 64
    #nop
    elif [ "x$1" = "x32" ]
    then
    # gmp 32
140 cd ~/win32/src
    tar xf gmp-*.tar.lz
    cd gmp-*/
    CC_FOR_BUILD="gcc" CPP_FOR_BUILD="gcc -E" ./configure --build=x86_64-pc-linux-↙
        ↪ gnu --host=i686-w64-mingw32 --prefix=$HOME/win32
    make -j $NCPUS
145 make install
    make check
    # mpfr 32
    cd ~/win32/src
    tar xf mpfr-*.tar.xz
150 cd mpfr-*/
    patch -N -Z -p1 < ../allpatches
    ./configure --host=i686-w64-mingw32 --prefix=$HOME/win32 --with-gmp-build=../gmp↙
        ↪ -6.1.2 --enable-static --disable-shared
    make -j $NCPUS
    make install
155 make check
    # zlib 32
    cd ~/win32/src
```

```
tar xf zlib-*.tar.xz
cd zlib-*/
CC=i686-w64-mingw32-gcc ./configure --static --prefix=$HOME/win32
CC=i686-w64-mingw32-gcc make -j $NCPUS
CC=i686-w64-mingw32-gcc make install
# png 32
cd ~/win32/src
tar xf libpng-*.tar.xz
cd libpng-*/
./configure --disable-shared --host=i686-w64-mingw32 CPPFLAGS="$CPPFLAGS -I$HOME/win32/include" LDFLAGS="$LDFLAGS -L$HOME/win32/lib" --prefix=$HOME/win32
make -j $NCPUS
make install
# jpeg 32
cd ~/win32/src
tar xf jpegsrc.v6b2.tar.gz
cd jpeg-6b2
./configure --disable-shared --host=i686-w64-mingw32 --prefix=$HOME/win32
make -j $NCPUS
make install
# tiff 32
cd ~/win32/src
tar xf tiff-*.tar.gz
cd tiff-*/
./configure --disable-shared --host=i686-w64-mingw32 --prefix=$HOME/win32
make -j $NCPUS
make install
# gsl 32
cd ~/win32/src
tar xf gsl-*.tar.gz
cd gsl-*/
./configure --disable-shared --host=i686-w64-mingw32 --prefix=$HOME/win32
make -j $NCPUS
make install
ln -s gsl-histogram.exe gsl-histogram # hack for test suite
make -k check || echo "expected 2 FAILs (multifit_nlinear, multilarge_nlinear)"
# pixman 32
cd ~/win32/src
tar xf pixman-*.tar.gz
cd pixman-*/
CC=i686-w64-mingw32-gcc LDFLAGS=-L$HOME/win32/lib ./configure --disable-shared --disable-openmp --prefix=$HOME/win32
make -j $NCPUS
make install
make check || echo "expected 1 FAIL (thread-test)"
# boost 32
cd ~/win32/src
7zr x boost*.7z
cd ~/win32/include
rm -f boost
ln -s ../src/boost*/boost/
# glm 32
cd ~/win32/src
7zr x glm*.7z
cd ~/win32/include
rm -f glm
ln -s ../src/glm*/glm/
```

```
# mingw-std-threads 32
cd ~/win32/include
rm -f mingw-std-threads
ln -s ../src/mingw-std-threads
# openexr 32
cd ~/win32/src
tar xf openexr-*.tar.gz
cd openexr-*/
patch -p1 < $(ls ../openexr-*.patch)
mkdir -p build
cd build
sed -i "s/x86_64/i686/g" ../cmake/Toolchain-mingw.cmake
cmake -DBUILD_SHARED_LIBS=OFF -DCMAKE_TOOLCHAIN_FILE=../cmake/Toolchain-mingw.↙
    ↪ cmake -DCMAKE_CXX_FLAGS=-I$HOME/win32/include -DZLIB_INCLUDE_DIR=$HOME/↙
    ↪ win32/include -DZLIB_LIBRARY=$HOME/win32/lib/libz.a -DCMAKE_INSTALL_PREFIX↙
    ↪ =$HOME/win32 ..
make -j $NCPUS
make install
# clew 32
#nop
else
  echo "usage:"
  echo "  $0 dl    # download sources"
  echo "  $0 64    # build 64bit libs"
  echo "  $0 32    # build 32bit libs"
  exit 1
fi
```

116 preprocessor.hs

```
{-
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
-}
import Text.Parsec
import Text.Parsec.Token
import Text.Parsec.Expr
import Text.Parsec.Language

import Control.Monad.Trans.RWS
import Control.Monad.Identity (Identity)

import Data.Char (isSpace)
```

```
import Data.List (nub)

data Expr
  = EInt Int
  | EVar String
  | EPow Expr Expr
  | EMul Expr Expr
  | EAdd Expr Expr
  | ESub Expr Expr
  | EGt Expr Expr
  | ELt Expr Expr
  | ENeg Expr
  | ESgn Expr
  | EAbs Expr
  | ESqr Expr
  | EDiffAbs Expr Expr
  | EAssign Expr Expr
  | EIf Expr Expr
  | ETE Expr Expr
  deriving Show

data Instruction
  = ISet String String
  | IAbs String String
  | INeg String String
  | ISgn String String
  | ISub String String String
  | IAdd String String String
  | IMul String String String
  | IMulI String String Int
  | ISqr String String
  deriving Show

temporaries = [ "t" ++ show i | i <- [0..] ]

isTemporary ('t':_) = True
isTemporary _ = False

allocate = do
  ~(t:ts) <- get
  put ts
  return t

deallocate t = do
  if isTemporary t
    then do
      ts <- get
      put (t:ts)
    else do
      return ()

instruction (ISet a b) = tell (["mpfr_set(", a, ",", b, ",MPFR_RNDN);\n"], [a, b↙
    ↳ ])
instruction (IAbs a b) = tell (["mpfr_abs(", a, ",", b, ",MPFR_RNDN);\n"], [a, b↙
    ↳ ])
instruction (INeg a b) = tell (["mpfr_neg(", a, ",", b, ",MPFR_RNDN);\n"], [a, b↙
    ↳ ])
```

```
instruction (ISub a b c) = tell (["mpfr_sub(", a, ",", b, ",", c, ",MPFR_RNDN);\n"], [a, b, c])
instruction (IAdd a b c) = tell (["mpfr_add(", a, ",", b, ",", c, ",MPFR_RNDN);\n"], [a, b, c])
instruction (IMulI a b c) = tell (["mpfr_mul_ui(", a, ",", b, ",", show (abs c), ",MPFR_RNDN);\n"] ++ if c < 0 then ["mpfr_neg(", a, ",", a, ",MPFR_RNDN);"] else [], [a, b])
instruction (IMul a b c) | b == c = tell (["mpfr_sqr(", a, ",", b, ",MPFR_RNDN);\n"], [a, b, c])
                         | otherwise = tell (["mpfr_mul(", a, ",", b, ",", c, ",MPFR_RNDN);\n"], [a, b, c])
instruction (ISqr a b) = tell (["mpfr_sqr(", a, ",", b, ",MPFR_RNDN);\n"], [a, b])

compile (EAssign (EVar v) a) = do
  u <- compile a
  instruction (ISet v u)
  deallocate u
  return v

compile (EVar v) = do
  return v

compile (EAbs a) = do
  u <- compile a
  v <- allocate
  instruction (IAbs v u)
  deallocate u
  return v

compile (ENeg a) = do
  u <- compile a
  v <- allocate
  instruction (INeg v u)
  deallocate u
  return v

compile (ESgn a) = do
  u <- compile a
  v <- allocate
  instruction (ISgn v u)
  deallocate u
  return v

compile (ESub a b) = do
  u <- compile a
  v <- compile b
  w <- allocate
  instruction (ISub w u v)
  deallocate u
  deallocate v
  return w

compile (EAdd a b) = do
  u <- compile a
  v <- compile b
  w <- allocate
```

```
    instruction (IAdd w u v)
    deallocate u
    deallocate v
    return w

compile (EMul (EInt a) b) = do
    v <- compile b
    w <- allocate
    instruction (IMulI w v a)
    deallocate v
    return w

compile (EMul a (EInt b)) = do
    u <- compile a
    w <- allocate
    instruction (IMulI w u b)
    deallocate u
    return w

compile (EMul (ENeg (EInt a)) b) = do
    v <- compile b
    w <- allocate
    instruction (IMulI w v (negate a))
    deallocate v
    return w

compile (EMul a (ENeg (EInt b))) = do
    u <- compile a
    w <- allocate
    instruction (IMulI w u (negate b))
    deallocate u
    return w

compile (EMul a b) = do
    u <- compile a
    v <- compile b
    w <- allocate
    instruction (IMul w u v)
    deallocate u
    deallocate v
    return w

compile (ESqr a) = do
    u <- compile a
    v <- allocate
    instruction (ISqr v u)
    deallocate u
    return v

compile x = error (show x)

init2 = concatMap (\t -> "mpfr_t " ++ t ++ "; mpfr_init2(" ++ t ++ ", bits);\n")
clear = concatMap (\t -> "mpfr_clear(" ++ t ++ ");\n")

runCompile es = case evalRWS (mapM_ compile es) () temporaries of
  ((), (is, vs)) ->
    let ts = filter isTemporary (nub vs)
```

```
    in  init2 ts ++ loopStart ++ concat is ++ loopEnd ++ clear ts

loopStart = "for (i = 0; i < nMaxIter && !m_bStop; i++) { DLOOP\n"
loopEnd   = "LOOP }\n"

vars (EInt _) = []
vars (EVar a) = [a]
vars (EPow a b) = vars a ++ vars b
vars (EMul a b) = vars a ++ vars b
vars (EAdd a b) = vars a ++ vars b
vars (ESub a b) = vars a ++ vars b
vars (EGt a b) = vars a ++ vars b
vars (ELt a b) = vars a ++ vars b
vars (ENeg a) = vars a
vars (ESgn a) = vars a
vars (EAbs a) = vars a
vars (ESqr a) = vars a
vars (EDiffAbs a b) = vars a ++ vars b
vars (EAssign a b) = vars a ++ vars b
vars (EIf a b) = vars a ++ vars b
vars (ETE a b) = vars a ++ vars b

interpret _ (EInt n) = show n
interpret _ (EVar v) = v

interpret t@"" (EPow a b) = "(" ++ interpret t a ++ "^" ++ interpret t b ++ ")"
interpret t@"" (EMul a@(ENeg (EInt _)) b) = "(" ++ interpret t a ++ "*" ++ ↙
    ↳ interpret t b ++ ")"
interpret t@"" (EMul a b@(ENeg (EInt _))) = "(" ++ interpret t a ++ "*" ++ ↙
    ↳ interpret t b ++ ")"
interpret t@"" (EMul a@(EInt _) b) = "(" ++ interpret t a ++ "*" ++ interpret t ↙
    ↳ b ++ ")"
interpret t@"" (EMul a b@(EInt _)) = "(" ++ interpret t a ++ "*" ++ interpret t ↙
    ↳ b ++ ")"
interpret t@"" (EMul a b) = "(" ++ interpret t a ++ "*" ++ interpret t b ++ ")"
interpret t@"" (EAdd a b@(ENeg (EInt _))) = "(" ++ interpret t a ++ "+" ++ ↙
    ↳ interpret t b ++ ")"
interpret t@"" (EAdd a b@(EInt _)) = "(" ++ interpret t a ++ "+" ++ interpret t ↙
    ↳ b ++ ")"
interpret t@"" (EAdd a@(ENeg (EInt _)) b) = "(" ++ interpret t a ++ "+" ++ ↙
    ↳ interpret t b ++ ")"
interpret t@"" (EAdd a@(EInt _) b) = "(" ++ interpret t a ++ "+" ++ interpret t ↙
    ↳ b ++ ")"
interpret t@"" (EAdd a b) = "(" ++ interpret t a ++ "+" ++ interpret t b ++ ")"
interpret t@"" (ESub a b@(ENeg (EInt _))) = "(" ++ interpret t a ++ "-" ++ ↙
    ↳ interpret t b ++ ")"
interpret t@"" (ESub a b@(EInt _)) = "(" ++ interpret t a ++ "-" ++ interpret t ↙
    ↳ b ++ ")"
interpret t@"" (ESub a@(ENeg (EInt _)) b) = "(" ++ interpret t a ++ "-" ++ ↙
    ↳ interpret t b ++ ")"
interpret t@"" (ESub a@(EInt _) b) = "(" ++ interpret t a ++ "-" ++ interpret t ↙
    ↳ b ++ ")"
interpret t@"" (ESub a b) = "(" ++ interpret t a ++ "-" ++ interpret t b ++ ")"
interpret t@"" (ELt a b) = "(" ++ interpret t a ++ "<" ++ interpret t b ++ ")"
interpret t@"" (EGt a b) = "(" ++ interpret t a ++ ">" ++ interpret t b ++ ")"
interpret t@"" (ENeg (EInt v)) = interpret t (EInt (negate v))
interpret t@"" (ENeg a) = "-(" ++ interpret t a ++ ")"
```

```
interpret t@"" (ESgn a) = "sgn(" ++ interpret t a ++ ")"
interpret t@"" (EAbs a) = "abs(" ++ interpret t a ++ ")"
interpret t@"" (ESqr a) = "sqr(" ++ interpret t a ++ ")"
interpret t@"" (EDiffAbs a b) = "diffabs(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t@"" (EAssign (EVar v) a) = v ++ "=" ++ interpret t a ++ ";"
interpret t@"" (EIf a (ETE b c)) = "(" ++ interpret t a ++ "?" ++ interpret t b ++ ":" ++ interpret t c ++ ")"

interpret t (EPow a b) = t ++ "powi(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EMul a@(ENeg (EInt _)) b) = t ++ "imul(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EMul a b@(ENeg (EInt _))) = t ++ "muli(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EMul a@(EInt _) b) = t ++ "imul(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EMul a b@(EInt _)) = t ++ "muli(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EMul a b) = t ++ "mul(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EAdd a b@(ENeg (EInt _))) = t ++ "addi(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EAdd a b@(EInt _)) = t ++ "addi(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EAdd a@(ENeg (EInt _)) b) = t ++ "iadd(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EAdd a@(EInt _) b) = t ++ "iadd(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EAdd a b) = t ++ "add(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ESub a b@(ENeg (EInt _))) = t ++ "subi(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ESub a b@(EInt _)) = t ++ "subi(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ESub a@(ENeg (EInt _)) b) = t ++ "isub(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ESub a@(EInt _) b) = t ++ "isub(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ESub a b) = t ++ "sub(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ELt a b) = t ++ "lt(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EGt a b) = t ++ "gt(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (ENeg (EInt v)) = interpret t (EInt (negate v))
interpret t (ENeg a) = t ++ "neg(" ++ interpret t a ++ ")"
interpret t (ESgn a) = t ++ "sgn(" ++ interpret t a ++ ")"
interpret t (EAbs a) = t ++ "abs(" ++ interpret t a ++ ")"
interpret t (ESqr a) = t ++ "sqr(" ++ interpret t a ++ ")"
interpret t (EDiffAbs a b) = t ++ "diffabs(" ++ interpret t a ++ "," ++ interpret t b ++ ")"
interpret t (EAssign (EVar v) a) = v ++ "=" ++ interpret t a ++ ";"
interpret t (EIf a (ETE b c)) = "(" ++ interpret t a ++ "?" ++ interpret t b ++ ":" ++ interpret t c ++ ")"

prepare "d" vs = unlines . concat $
```

```
    [ [ "const T Xr2 = Xr * Xr;" | "Xr2" `elem` vs ]
  , [ "const T Xi2 = Xi * Xi;" | "Xi2" `elem` vs ]
  , [ "const V xr2 = xr * xr;" | "xr2" `elem` vs ]
  , [ "const V xi2 = xi * xi;" | "xi2" `elem` vs ]
  ]
prepare "dc" vs = unlines . concat $
  [ [ "const complex<T> X2 = X * X;" | "X2" `elem` vs ]
  , [ "const complex<V> x2 = x * x;" | "x2" `elem` vs ]
  ]
prepare "cld" vs = unlines . concat $
  [ [ "const double Xr2 = d_mul(Xr, Xr);" | "Xr2" `elem` vs ]
  , [ "const double Xi2 = d_mul(Xi, Xi);" | "Xi2" `elem` vs ]
  , [ "const double xr2 = d_mul(xr, xr);" | "xr2" `elem` vs ]
  , [ "const double xi2 = d_mul(xi, xi);" | "xi2" `elem` vs ]
  ]
prepare "cldc" vs = unlines . concat $
  [ [ "const dcomplex X2 = dc_sqr(X);" | "X2" `elem` vs ]
  , [ "const dcomplex x2 = dc_sqr(x);" | "x2" `elem` vs ]
  ]
prepare "clfe" vs = unlines . concat $
  [ [ "const floatexp Xr2 = fe_mul(Xr, Xr);" | "Xr2" `elem` vs ]
  , [ "const floatexp Xi2 = fe_mul(Xi, Xi);" | "Xi2" `elem` vs ]
  , [ "const floatexp xr2 = fe_mul(xr, xr);" | "xr2" `elem` vs ]
  , [ "const floatexp xi2 = fe_mul(xi, xi);" | "xi2" `elem` vs ]
  ]
prepare "clfec" vs = unlines . concat $
  [ [ "const fecomplex X2 = fec_sqr(X);" | "X2" `elem` vs ]
  , [ "const fecomplex x2 = fec_sqr(x);" | "x2" `elem` vs ]
  ]
prepare "clsf" vs = unlines . concat $
  [ [ "const softfloat Xr2 = sf_sqr(Xr);" | "Xr2" `elem` vs ]
  , [ "const softfloat Xi2 = sf_sqr(Xi);" | "Xi2" `elem` vs ]
  , [ "const softfloat xr2 = sf_sqr(xr);" | "xr2" `elem` vs ]
  , [ "const softfloat xi2 = sf_sqr(xi);" | "xi2" `elem` vs ]
  ]
prepare "clsfc" vs = unlines . concat $
  [ [ "const sfcomplex X2 = sfc_sqr(X);" | "X2" `elem` vs ]
  , [ "const sfcomplex x2 = sfc_sqr(x);" | "x2" `elem` vs ]
  ]

def :: GenLanguageDef String () Identity
def = emptyDef{ identStart = letter
              , identLetter = alphaNum <|> char '_'
              , opStart = oneOf ops
              , opLetter = oneOf ops
              , reservedOpNames = map (:[]) ops
              , reservedNames = ["sqr", "sgn", "abs", "diffabs"]
              }
  where ops = "=?:<>+-*^"

TokenParser{ parens = m_parens
           , identifier = m_identifier
           , reservedOp = op
           , reserved = m_reserved
           , integer = m_integer } = makeTokenParser def

exprparser = buildExpressionParser table term <?> "expression"
```

```
table = [ [Prefix (op "-" >> return ENeg)]
        , [Infix (op "^" >> return EPow) AssocLeft]
        , [Infix (op "*" >> return EMul) AssocLeft]
        , [Infix (op "+" >> return EAdd) AssocLeft
          ,Infix (op "-" >> return ESub) AssocLeft]
        , [Infix (op ">" >> return EGt) AssocLeft
          ,Infix (op "<" >> return ELt) AssocLeft]
        , [Infix (op ":" >> return ETE) AssocLeft]
        , [Infix (op "?" >> return EIf) AssocLeft]
        , [Infix (op "=" >> return EAssign) AssocLeft]
        ]
term = m_parens exprparser
       <|> (EVar <$> m_identifier)
       <|> (EInt . fromIntegral <$> m_integer)
       <|> (char '-' >> EInt . negate . fromIntegral <$> m_integer)
       <|> (m_reserved "sqr" >> ESqr <$ string "(" <*> exprparser <* string ")")
       <|> (m_reserved "sgn" >> ESgn <$ string "(" <*> exprparser <* string ")")
       <|> (m_reserved "abs" >> EAbs <$ string "(" <*> exprparser <* string ")")
       <|> (m_reserved "diffabs" >> EDiffAbs <$ string "(" <*> exprparser <* ↙
           ↳ string "," <*> exprparser <* string ")")

data CL
  = Context String
  | Block String String

context = Context <$> many1 (noneOf "@")
block :: ParsecT String () Identity CL
block = do
  char '@'
  t <- many (noneOf " ")
  many (char ' ')
  char '{'
  s <- many (noneOf "}")
  char '}'
  pure (Block t s)

blockp = many (exprparser <* string ";") <* eof

parseCL s = case parse (many (block <|> context) <* eof) "" s of
  Right cl -> concatMap interpretCL cl
  Left e -> error (show e)

interpretCL (Context s) = s
interpretCL (Block "rd" s) = unlines . (++ [""]) . map (++ " \\") . ("#define ↙
    ↳ DLOOP" :) . lines $ s
interpretCL (Block "rr" s) = case parse blockp "" $ filter (not . isSpace) s of
  Right es -> runCompile es ++ "#undef DLOOP\n"
  Left e -> error (show e ++ " : " ++ show s)
interpretCL (Block "rc" s) = "assert(! \"implemented yet\");\n"
interpretCL (Block t s) = case parse blockp "" $ filter (not . isSpace) s of
  Right es -> "{" ++ prepare t (nub $ concatMap vars es) ++ unlines (map (↙
      ↳ interpret t2) es) ++ "}"
  Left e -> error (show e ++ " : " ++ show s)
  where
    t2 | t == "d" = ""
       | t == "dc" = ""
       | otherwise = case t of 'c':'l':ts -> ts ++ "_"
```

```
main = interact parseCL
```

117 README.md

```
# Kalles Fraktaler 2 +

As the orginal upstream author Karl Runmo says:

> Want to create DEEP Mandelbrot fractals 100 times faster than the commercial
> programs, for FREE? One hour or one minute? Three months or one day?
> Try Kalles Fraktaler!

I (Claude Heiland-Allen) forked the code and swapped out the custom arbitrary
precision floating point code for highly optimized libraries, making it
even faster. Cross-compiled to Windows from Linux MINGW64. Now with many other
enhancements.

Original upstream version:

- <http://www.chillheimer.de/kallesfraktaler/>

This version:

- <https://mathr.co.uk/kf/kf.html>

Feedback:

- <https://fractalforums.org/kalles-fraktaler/15> current forum
- <http://www.fractalforums.com/kalles-fraktaler/> legacy forum (read only)
- <mailto:claude@mathr.co.uk?subject=Kalles%20Fraktaler%202> personal mail


## Quick Start

- Download the latest archive from the website:
  <https://mathr.co.uk/kf/kf.html#download>

- Unzip it wherever you want, no installation required.
  You need `7-zip` to unzip `.7z` archives, available from
  <https://www.7-zip.org/>.

- Launch `kf.exe` for 64-bit (32-bit binaries are no longer provided).

- Start exploring!

  - Use the mouse scroll wheel to zoom.

  - Choose different fractal formulas in the iterations dialog (Ctrl-I).

  - Choose different colours in the colouring dialog (Ctrl-C).

  - Zoom deeper more quickly with Newton-Raphson zooming (Ctrl-D).


## Known Bugs
```

```
- Windows Defender sometimes reports that KF contains malware
  (false positive, I hope - I check with virustotal.com and all seems fine)
  (possibly due to screenshots for crosshair window, some code also moves
  the mouse cursor)
- out of memory conditions cause crashes (for example, if bitmap creation
  fails - also need to check huge sizes) (reported by gerrit)
- resizing window during examine zoom sequence auto solve glitches leads to
  corruption of the zoom sequence data
- "stop autosolve" during examine zoom sequence fails and corrupts zoom sequence
- minimizing window during zoom sequence rendering corrupts image (saves blank
  image or repeated frame) (reported by gerrit and CFJH)
- zoom size 1.2 sequence JPEGs have "boxes" while KFB is ok (reported by saka)
- resizing window by dragging frame corner in WINE on Linux sometimes crashes
- with "reuse reference", corrupt image at transition between number types
  (eg e600) (reported by CFJH) - workaround is to render in segments or force
  the number type higher ("use long double always", "use floatexp always")
- crash when zooming too quickly near interior black regions (reported by
  Foxxie) "usually near the elephant valley area or seahorse valley area of
  minibrots, happens worse the faster you zoom, usually if you try to zoom at
  the skinniest part very quickly"
- crash when zooming out by a huge custom zoom factor (2^25) (reported by CFJH)
  (divide by zero in `StretchBlt()`)
- on special locations kf renders endless references and comes to no end
  (reported by CFJH)
- scaled long double rendering broken with some locations (reported by CFJH)
  (blank image, "always use floatexp" gives correct render)
- analytic DE broken with some power 3 Mandelbrot locations (reported by gerrit)
  (workaround is to disable series approximation)
- "resume zoom sequence" re-uses last set zoom count limit
- "examine zoom sequence" doesn't save corrected PNG images during glitch solve
- speckles when rendering zoom out sequence
- black regions when rendering zoom out sequence (maximum iterations are reduced
  too much before spirals appear in next frame) (reported by gerrit)
- there is still a race conditions in guessing (doesn't wait for previous
  progressive interlacing pass to be 100% done before the next one starts)
- newton-raphson zooming to minibrot doesn't increase maxiters enough sometimes
- newton-raphson zoom preset depths are bad for formulas with power other than 2
- burning ship series approximation stops at first fold (typically 1 period of
  a central miniship), could potentially subdivide the region (and shift the
  series, for the folded part) and carry on?
- burning ship series approximation probe points might miss some folds near
  the edges of the image (need to increase probe point density?)
- bad combinations of skew, distance estimation, and series approximation
- scaled (long) double yr,yi can underflow to 0, eventually causing derivatives
  to be too small and de overflows to infinity -> blank screen: workaround is to
  force long double or floatexp as appropriate
- auto skew (escape) button doesn't work well with some formulas (eg SimonBrot)
- navigation with scroll wheel and -/+ keys is hardcoded to factor of 2 instead
  of using the zoom size set in the View menu
- NR zoom doesn't work well in skewed locations
- nanomb1/2 OrderM, OrderN can only be changed by hand-editing .kfs Settings ↙
  ↳ files
- nanomb2 RadiusScale can only be changed by hand-editing .kfs Settings files
- nanomb1/2 number type fixed to floatexp (long double or double would be faster↙
  ↳ )
- nanomb1/2 number type is not rescaled (only matters for long double / double)
- nanomb2 currently disables glitch detection and correction
```

```
- nanomb1/2 reference calculations are not multithreaded (single core only)
- nanomb1/2 reference calculations are using slow Boost C++ wrapper for MPFR
- kf-tile.exe doesn't support skew yet
- status bar reference count doesn't reset when zooming before it is "Done"
- aborting rendering should not recolour incomplete image (reported by CFJH)
- cannot edit coordinates when zoom is deeper than e7000 or so (reported by CFJH↵
  ↳ )
- help button in file browser does nothing
- may be difficult to build the source at the moment
  (out of date instructions for Windows, dependency on 'et', ...)


## Differences From Upstream 2.11.1

### Incompatible Changes

- **In versions `2.14.8` and above**, the iteration count limit is raised
  above 2 giga-iterations. Parameters with such high iterations counts will
  load incorrectly in earlier versions.

- **In versions `2.14.8` and above**, directional DE is saved in EXR files
  using `DEX` and `DEY` channels, instead of distance using `DE` channel.
  Old EXR files' `DE` can not be re-loaded in newer versions, and
  new EXR files' `DEX` and `DEY` can not be re-loaded in older versions.

- **In versions `2.14.7` and above**, the new "LogLog" and "ATan" iteration
  count transfer functions (ColorMethod 9 and 10) are not available in
  earlier versions.

- **In versions `2.14.5` and above**, the new "Low bailout" (SmoothMethod: 2)
  is not available in earlier versions.

- **In versions `2.14.5` and above**, the two new General Abs Quadratic formulas
  are not available in earlier versions.

- **In versions `2.14.4` and above**, the new "Mothbrot" formulas are not
  available in earlier versions.

- **In versions `2.14.3` and above**, colouring of zoomed out views is fixed,
  and may be different from earlier versions.

- **In versions `2.14.1` and above**, there are two new formulas which are
  not available in earlier versions.

- **In versions `2.13.1` through `2.13.9`**, derivatives are calculated by
  default. In versions `2.14.1` and above, you need to enable the derivatives
  calculations (in the Iterations dialog, or via a settings file) if you
  want to use analytic distance estimation colouring. Derivatives are not
  available in earlier versions.

- **In versions `2.13.1` through `2.13.4` inclusive** the interior is
  white. In `2.13.5` and above it is user-settable, defaulting to black,
  Parameter files made with earlier `2.13` versions should be modified to
  explicitly set a white interior colour. In earlier versions (including
  upstream `2.11.1`) the interior is black with no way of changing it.

- **In version `kf-2.12.10` only** the jitter is "Gaussian" with no way of
```

```
  changing the shape. In `2.12.11` and above, the shape can be changed, and
  the default is now "uniform". Earlier versions (including upstream `2.11.1`)
  do not support jitter at all. To get the same results as `2.12.10` you must
  enable the Gaussian jitter checkbox and set the jitter scale box to 1.

- **In version `kf-2.12.1` and above**, DE colouring method #5 is once again
  backwards compatible with upstream `2.11.1`. Parameter files made with
  `2.11.1+gmp.DATE` versions should be modified to use Distance (Square Root)
  colouring method #8.

- **In version `kf-2.11.1+gmp.20170822` only**, DE colouring method #5 used log
  instead of sqrt for a more perceptually linear effect. In later versions,
  this log scaling is achieved with a new colouring method #7, while the DE
  colouring method #5 reverts to sqrt as before. The new colouring method ID
  allows old `2.11.1+gmp.DATE`
  parameter files to be loaded into current versions and display as
  intended. Any parameter files saved with the new Distance (Logarithm)
  colouring method will not display as intended in older versions. Parameter
  files using Distance colouring method saved with this particular version
  should be modified to use Distance (Logarithm) in the latest version.

- multiple finite difference methods for distance colouring
  (only the default `Traditional` is available in `2.11.1`).

- analytic DE colouring with derivatives (not available in `2.12.x` or earlier).


### Other Changes

- Makefile build system using MINGW to cross-compile to Windows from Linux
- uses GMP+MPFR for arbitrary precision floating point instead of custom code
- uses Boost wrapper around MPFR floats for higher-level coding
- use installed JPEG library, instead of bundled sources
- long double support built into EXE (no separate DLL needed)
- virtually unlimited precision (memory needed for precise numbers is an issue)
- threaded calculations reimplemented with barriers to avoid WINE slowdown
- workaround for WINE issue artificially limiting image size (up to 2GiB now)
- bugfix: inflection performance issue (was converting number types needlessly)
- bugfix: cross-hair resource issue (reported and fixed by Kalles Fraktaler)
- miscellaneous code cleanups (-fpermissive fixes, const fixes, delete[] fixes,
  64bit compatibility paranoia)
- formula inner loops generated at compile time from high level specification
  XML using XSLT and a preprocessor implemented in Haskell
- optimized some reference calculations by floating temporaries out of loops
- XML preprocessor optimizes more reference calculations in the same way
- optimized Newton-Raphson zooming by using lower-level GMP calls
- save images to PNG, TIFF and EXR format as well as JPEG
- colouring uses floating point internally (fewer quantisation steps)
- dithering at end of colouring to improve perceptual quality (reduced banding)
- "glitch low tolerance" checkbox that can be enabled to detect glitches more
  aggressively (disabled by default, enable it if you get undetected glitches)
- updated program icon with transparent background and large version
- parameter data is saved as comment in image files (all formats)
- preferences (rendering settings not related to location) save and load
  (.kfs files and image comments too)
- command line arguments to load settings and/or location
- command line arguments to render and save images / maps before quiting
```

```
- see change log below for more...


## Change Log

- **kf-2.14.9** (????-??-??)

    - OpenCL support for GPU rendering (enabled at build time with
      'make clean && make OPENCL=1')
    - don't set zoom size when resuming zoom out sequence (now you need to
      set it manually before resuming) (fixes issue where it was clamped to
      integer; reported by david)
    - clamp half-precision colour output in EXR (fixes issue with NaNs)
    - save file format version number with parameters and settings

- **kf-2.14.8** (2019-09-23)

    - feature: SIMD support.  Adjustable tuning parameters SIMD vector size
      (default 2) and chunk size (default 64) settings in the Advanced menu.
      Should speed up perturbation calculations in double precision (up to
      zoom depth 1e300 or so).  SIMD is also supported in scaled double for
      Mandelbrot power 2 (zoom depth 1e600 or so) and power 3 (zoom depth
      1e400 or so).  SIMD is not yet implemented for scaled double with
      derivatives.  Compile KF yourself with optimisations for your own
      specific CPU for a bigger speed boost.
    - feature: Newton-Raphson zooming completion estimate (number of steps)
    - feature: Newton-Raphson zooming progress backups (after each step)
    - feature: fewer required fields when loading location parameters (allows
      loading the NR-zoom backups)
    - feature: directional DE stored in EXR as 'DEX' and 'DEY' channels,
      and used for slopes colouring with 'Analytic' differences (requires
      derivatives, fixes noisy texture when jitter is enabled)
    - feature: 64bit iteration count support (limit is now 9 exa-iterations
      up from 2 giga-iterations), use EXR for export.  This feature is very
      experimental, I haven't had the patience to test it properly.  Please
      report back success or failure.
    - note: periods are still 32bit and limited to 2 giga-iterations
    - bugfix: command line --save-exr now sets non-interactive mode flag.
    - note: built for 64bit only; 32bit hardware is obsolete by now.
    - note: distribution package compressed with 7-zip for smaller size.
    - upgrade to openexr 2.4.0
    - upgrade to mingw64 g++ 9.2.1-8+22~exp1 (Debian experimental)
      and rebuild everything

- **kf-2.14.7.1** (2019-08-30)

    - bugfix: iteration data channels are no longer incorrectly vertically
      flipped when saving and loading EXR files; they now should match the
      RGB orientation
    - feature: optionally disable threaded reference calculations
    - bugfix: the 20% performance regression vs kf-2.12 has been fixed
    - bugfix: "save overwrites" now saved in .kfs settings files

- **kf-2.14.7** (2019-08-29)

    - feature: save EXR (combined image with iteration map data and metadata)
      and open EXR (as settings, location or iteration map data).  For
```

```
      motivation and specifications see <https://fractalforums.org/f/11/t/3017>.
    - feature: LogLog and ATan color methods (iteration count transfer
      functions) (ported from Mandel Machine, suggested by Dinkydau)
    - feature: option to control overwriting existing files with Save/Ctrl-S.
      With overwrite disabled by default, KF will save files with a
      timestamp before the .kfr extension, preventing accidental data loss
      (suggested by fractal). The previous behaviour can be restored in
      the Advanced menu.
    - feature: option to reduce thread count by 1 (suggested by saka)
    - bugfix: set thread background mode for improved responsiveness
      (reported by saka)
    - bugfix: remove annoying warning dialogs from store zoom out sequence
      (reported by saka)
    - bugfix: size status bar sections proportionally to window width
      (fixes accessibility issues with non-standard system font size)
      (reported by fractal)
    - bugfix: make Octave/Matlab scripts into function files for broader
      compatibility (usage example: `stratify("tile",4)`) (bug reported
      by Chronicler1701, fix suggested by gerrit).
    - new dependency: mingw-std-threads git
    - new dependency: ilmbase 2.3.0
    - new dependency: openexr 2.3.0
    - upgrade to boost 1.71.0
    - upgrade to gsl 2.6

- **kf-2.14.6.1** (2019-07-19)

    - bugfix: load palette after map in command line rendering so that the
      colours from the palette are actually used (reported by blob)

- **kf-2.14.6** (2019-07-18)

    - new feature: high quality image downscaling with anti-aliasing using
      the Pixman library
    - new feature: ball method for period detection in Newton-Raphson zoom
    - new feature: command line KFB map colouring
    - bugfix: image is refreshed from first calculated pixels, instead of 1%
      (reported by Fluoroantimonic_Acid)
    - bugfix: don't read Iteration Divisor from KFB map files (for historical
      reasons this is an integer in the file format, instead of the floating
      point value that it can take now in the rest of the program)
    - upgrade to boost 1.70.0
    - upgrade to mpfr 4.0.2p1
    - upgrade to png 1.6.37
    - upgrade to glm 0.9.9.5
    - upgrade to mingw64 g++ 8.3-win32 20190406 and rebuild everything

- **kf-2.14.5** (2019-03-20)

    - new feature: command line zoom sequence rendering with `-z NFRAMES` flag
    - new feature: low bailout (SmoothMethod: 2) (suggested by Dinkydau)
      <https://fractalforums.org/f/28/t/2594/msg12801#msg12801>
    - new feature: General Abs Quadratic formulas (inspired by Alef and gerrit)
      <https://fractalforums.org/f/28/t/2653/msg13242#msg13242>
    - bugfix: thin black line in exterior between iteration bands (reported by
      ↳ CFJH)
    - bugfix: malformed location no longer crashes (reported by Foxxie)
```

```
    - upgrade to mpfr 4.0.2
    - upgrade to glm 0.9.9.4

- **kf-2.14.4** (2019-01-18)

    - new feature: nanomb1 algorithm for power 2 Mandelbrot (experimental)
      (originally by knighty) (not quite as experimental as nanomb2, suggested
      by gerrit, requires exact period to be set by Newton or location dialog)
    - new feature: make interior checking for nanomb optional (new menu item)
    - new feature: RadiusScale setting for nanomb2 (only settable in .kfs so
      far, default value 0.1 may be safe but slow)
    - new feature: Mothbrot family of formulas (suggested by simon.snake)
      (not optimized yet, preliminary version for testing)
    - bugfix: incremental drawing looks much better with slopes and/or nde
      (there is a crosshair visible while rendering, which disappears when
      the image is finished, not sure why this happens)
    - upgrade to libpng 1.6.36
    - upgrade to libtiff 4.0.10
    - upgrade to libmpfr 4.0.1-p14
    - upgrade to boost 1.69.0

- **kf-2.14.3** (2018-10-29)

    - new feature: nanomb2 algorithm for power 2 Mandelbrot (experimental)
      (originally by knighty)
    - new feature: display Newton period (and set limit for nanomb2 algorithm)
      in Location dialog
    - new feature: Ctrl-Shift-W shortcut to set image size (suggested by gerrit)
    - bugfix: distinguish dialog titles for set window size and set image size
    - bugfix: fix corrupt images when zooming out from the default view
    - new feature: dialogs to en/disable derivatives when opening file with/out
      analytic DE; automatically enabling derivatives if needed in CLI mode
      (reported by gerrit)
    - default zoom size changed from 4 to 2 (press Ctrl-4 to reset to 4, or load
      a settings file with your preference) (suggested by gerrit, to avoid the
      dreaded "zoom size is not 2" warning when saving zoom out sequence).
    - merged changes from 2.13.11

- **kf-2.13.11** (2018-10-29)

    - bugfix: make preprocessor compatible with ghc-8.6 (MonadFail)
    - bugfix: fix translation with reuse reference enabled
      (reported by Dinkydau)
    - bugfix: don't add additional references if autosolve is disabled
    - upgrade to libpng 1.6.35

- **kf-2.14.2** (2018-10-23)

    - bugfix: initial window size was off by a few pixels
    - new feature: undo/redo (menu and Ctrl-Z, Ctrl-Y) (suggested by TwinDragon)
    - new feature: added copy/paste to menu (Ctrl-X, Ctrl-V worked already)
    - new feature: Ctrl-Shift-K shortcut to reset skew
    - new feature: Ctrl-Shift-T shortcut to reset rotation
    - bugfix: minor improvements to documentation
    - bugfix: fix series approximation derivative initialization
    - internal: refactor series approximation for Mandelbrot power > 2
    - merged changes from 2.13.10
```

```
- **kf-2.13.10** (2018-10-23)

    - bugfix: corrupt image at transition between number types (eg near e600)
      (reported by CFJH)
    - bugfix: changing "threads per CPU" during rendering crashes
      (reported by CFJH) (the menu is now disabled during rendering)
    - bugfix: set approx terms to 3 for other than power 2 Mandelbrot
    - bugfix: fix floatexp toString for negative values and 0
    - internal: update to et-kf API version 5

- **kf-2.14.1** (2018-09-27)

    - new feature: series approximation for Burning Ship power 2, using
      case analysis to know how it gets reflected at each iteration
    - two new formulas: General Quadratic Minus, General Quadratic Plus;
      "RedShiftRider factor A" sets coefficients (suggested by gerrit)
    - new feature: "Fast" preset activated by default
    - new feature: GUI menus reorganized to make more sense and updated
      the manual to reflect this
    - new feature: "check for updates" option in the file menu (requires
      internet access)
    - new feature: 1/4 and 1/2 threads per CPU settings (minimum 1 thread)
    - new feature: optional use of d/dZ in autoskew escape (suggested by
      gerrit)

- **kf-2.13.9** (2018-09-06)

    - new feature: auto skew without miniset: new button in Newton-Raphson
      zooming dialog, "Auto Skew (Escape)", that skews at the center pixels
      (algorithm suggested by gerrit)
    - bugfix: set window title on parameter drag and drop (reported by gerrit)
    - bugfix: reading PNG metadata works even if it is moved after the image
      data chunks and has a miscapitalized "Comment" keyword
    - bugfix: fix Mandelbar derivative calculations for ADE (reported by gerrit)
    - bugfix: fix 4th False Quasi formulas Newton-Raphson zooming

- **kf-2.13.8** (2018-08-28)

    - new feature: auto skew (via Newton-Raphson zooming dialog)
    - new feature: show/hide crosshair window (suggested by gerrit and others)
    - new feature: quality presets ("fast" but inaccurate, "best" but slow)
    - new feature: page up / page down keyboard shortcuts to zoom in / out
    - new feature: drag-and-drop parameter files from the file manager to
      the main window to open them
    - new feature: drag-and-drop palette files from the file manager to
      the color dialog window to open them
    - new feature: CLI prints total remaining pixels (suggested by gerrit)
    - bugfix: suppress error dialogs when loading metadata from TIFF
    - bugfix: remove obsolete .ini stuff that was overriding default settings
    - bugfix: enabled "no reuse center" by default
      (without it zoom out sequence sometimes glitches)
    - major documentation improvements
    - new dependency: GLM 0.9.9.0

- **kf-2.13.7** (2018-08-14)
```

```
    - export to uncompressed TIFF images (faster than PNG, but larger)
      (suggested by gerrit)
    - nicer default palette
    - upgrade to Boost 1.68.0
    - upgrade to MPFR 4.0.1-p13

- **kf-2.13.6** (2018-08-02)

    - fix for "a single dot appears in the middle of a mini"
      (reported by gerrit)
    - fix for "zooming out saves Zoom 0 in KFR parameter file"
      (reported by CFJH)
    - "open map" function (file menu)
      workflow: open KFR, ESC to abort, open KFB, adjust colours, save image
      (ignoring the warning about a KFB map existing with the same name)

- **kf-2.13.5** (2018-06-26)

    - colouring is parallelized for speedup when multiple cores are available
    - only sort pixel indices in interactive mode (speeds up batch mode)
    - palette loading fixed to load all colouring data
    - interior colour can be set now, defaults to black
    - upgrade to GSL 2.5

- **kf-2.13.4** (2018-06-21)

    - derivative computations can be switched off in the iterations
      dialog (increases rendering speed, breaks analytic DE colouring)

- **kf-2.12.13.1** (2018-06-04)

    - fix severe performance bug introduced in previous version
    - upgrade to Boost 1.67.0
    - upgrade to MPFR 4.0.1-p6

- **kf-2.12.13** (2018-05-15)

    - fix crasher bug relating to series approximation memory allocation
      (reported by gerrit)

- **kf-2.13.3** (2018-05-08)

    - Newton-Raphson zooming precision loss bug fix (reported by gerrit)
    - Newton-Raphson zooming for Mandelbrot powers 6 7 8 9 10
      (reported by gerrit)
    - Newton-Raphson zooming size estimate fixed for more formulas (now
      correctly uses smallest power instead of largest, for renormalization
      behaviour near zero)
    - Mandelbrot power 6 7 8 9 10 analytic distance estimation bug fixes
      (reported by gerrit)

- **kf-2.13.2** (2018-05-01)

    - Newton-Raphson zooming for all formulas (using code generated by 'et')
      (no progress reporting yet, nor convergence detection for early exit)
    - R^2 matrix derivatives for all formulas (fixes analytic DE with skew)
    - fix image corruption when enabling guessing with analytic DE
```

```
505       - fix assertion failed popup (race condition between parameter changes and
            colouring during rendering) (reported by gerrit)

    - **kf-2.13.1** (2018-03-30)

510       - added derivative calculations
          - known bug: calculated even if not needed for colouring
          - known bug: some derivatives are C approximations instead of R^2 matrix
          - analytic DE colouring mode
          - known bug: DE data not saved to KFB map files
515       - interior is white instead of black
          - known bug: interior colour should be customizable
          - series approximation uses Horner's rule to evaluate polynomials
            (minor optimisation)
          - default settings are best-quality by default
520
    - **kf-2.12.12** (2018-03-30)

          - Free Software under GNU Affero General Public License, version 3 or ↙
            ↳ greater
          - least squares numerical differences (experimental; subject to change)
525       - Laplacian numerical differences (suggested by gerrit)
          - (possibly stratified) tiled settings/parameters generator program
          - stratified tiling reassembler GNU Octave script

    - **kf-2.12.11** (2018-03-12)
530
          - uniform jitter by default (suggested by gerrit)
          - Gaussian jitter available as an option
          - jitter scale factor setting (suggested by gerrit)
          - log verbosity flag for command line rendering
535       - upgrade to libjpeg6b2 <https://jpegclub.org/support/>
          - fix bug where Ctrl-S would overwrite the wrong file (reported by gerrit)
          - fix bug where lines would appear aligned with references (reported by
            gerrit and CFJH)
          - fix "color method and infinite waves are immediately applied" even when
540         colouring is disabled (reported by gerrit)

    - **kf-2.12.10** (2018-02-22)

          - upgrade to MPFR 4.0.1
545       - removed some build instructions from README (see the prepare.sh script)
            (it was too annoying to have to keep updating everything in two places)
          - command line non-interactive mode works without opening a window
            (suggested by gerrit)
          - pixel jitter (dithering of position): set non-zero seed in iterations
550         dialog to enable it, different seeds give slightly different images which
            can be stacked later for anti-aliasing as an alternative to rendering
            large and downscaling later; helps reduce Moiré grid artifacts at the
            cost of noise
          - "Enable" toggle in colouring dialog: colouring takes long for large
555         images, disable the toggle if you want to make many changes at once, then
            click "Apply" or re-enable to see the results (discussed with gerrit)

    - **kf-2.12.9.1** (2018-01-24)

560       - fixed broken "approx low tolerance" checkbox (thanks to Dinkydau)
```

```
    - fixed NR-zoom dialog to remember custom zoom factor between reopenings
      (reported by gerrit)
    - fixed initialization of some menu item state on load

- **kf-2.12.9** (2018-01-24)

    - improved Newton-Raphson zooming dialog, now zooms to a factor between
      current zoom and minibrot size (suggested by Dinkydau)
    - implemented "scaled long double" iterations for Mandelbrot power 2 and 3,
      which might speed up some zooms between e4900 and e9800
    - fixed "the maxiterations suddenly resets to some very low value"
      (reported by gerrit)
    - fixed "Newton iterates 100 times before failing", now fails fast if
      the C goes outside the target (reported by gerrit and Dinkydau, fix
      suggested by knighty)
    - fixed "Newton zooms to blank image", reported by gerrit, fix was to
      remove a check on the derivative being huge that was falsely reporting
      success
    - fixed "manual ApproxTerms gets reset to AutoApproxTerms value"
      (reported by gerrit)
    - barrier implementation yields if there are fewer CPUs than threads
      (fixes very slow Newton-Raphson zooming and reference calculations
      on dual-core laptop) (this is the behaviour of 2.12.5 and earlier, but
      conditional on number of cores)
    - upgrade to libpng 1.6.34
    - added shell script containing some of the build instructions from
      the documentation

- **kf-2.12.8** (2017-12-22)

    - fixed the fix for hang in normalisation (was generating bad images)
      (reported by gerrit with bug7.png)
    - restore ignoring isolated glitches
    - fixed DE spots bug (somehow the reference was being reset to the center
      of the screen sometimes without its pixel position being updated)
      (reported by gerrit)
    - fixed bugs with guessing and glitch status (only interpolate when the
      glitch status of both neighbours is the same) (caused lines when dragging
      the view, probably other badness too)
    - added logic to prevent too-huge image size being entered via the GUI
      (settings files are still not checked...)
    - clarified store zoom count user interface (reported by CFJH)
    - fixed store zoom out for deep zooms (bug reported by gerrit) (the issue
      was exceptions thrown by std::stod(), remember to catch them)
    - refactor auto iterations (issue reported by gerrit remains unresolved)
    - fixed hardcoded count in glitch correction
    - use Taylor intervals in ball-period method (code copied from knighty)

- **kf-2.12.7** (2017-12-07)

    - renabled guessing conditional on menu option (was disabled in 2.12.4 as
      enabling it made some random speckles, possibly due to a race condition)
      (requested by Kalles Fraktaler and Fractal universe)
    - added "threads per cpu core" setting
    - compiles clean with -Wwrite-strings
    - even lower resolution preview for more intensive locations (suggested by
      Foxxie) (implemented with Adam7-style interlacing with circular sorting)
```

```
    - copy (Ctrl-X) and paste (Ctrl-V) parameters from the system clipboard
    - fixed hang crash bug when normalizing smooth iteration values
    - use interval arithmetic ball-period method instead of box-period (speeds
      up Newton-Raphson zooming a bit) (idea from knighty and gerrit)

- **kf-2.12.6** (2017-11-24)

    - fix central differences (reported by gerrit)
    - fix insufficient precision in Zoom: saved in .kfr (reported by CFJH)
    - option to render zoom out sequence without saving KFB maps (suggested by
      CFJH)
    - option to stop rendering zoom out sequence after a certain number of
      frames (suggested by CFJH)
    - resume zoom sequence works without KFB maps saved every frame (still needs
      a "last.kfb", this is saved automatically when needed)
    - barrier no longer yields (fixes priority inversion on heavily loaded
      systems) (reported by gerrit)
    - switch from GMP mpf_t to MPFR (fixes some blank images on load, also some
      Newton-Raphson zoom failures - bug involved incorrect normalization)
      (reported by Kalles Fraktaler and gerrit)

- **kf-2.12.5** (2017-11-02)

    - preferences (rendering settings not related to location) save and load
      (.kfs files and PNG/JPEG comments too)
    - command line arguments to load settings and/or location
    - command line arguments to render and save PNG/JPEG/KFB Map before quiting
    - auto-added new references recalculate only all glitched pixels (in earlier
      versions it would recalculate all pixels with same integer iteration
      count, which may or may not have been glitched, and may have missed some
      glitches)
    - glitch correction now uses glitch flag instead of just iteration count
      (this ensures the reference is added in a really glitched pixel, so at
      least one pixel will be fixed by each reference, ensuring termination
      with a finite number of references)
    - fix bugs with references when calculating their own pixels
    - fix off-by-one when references fix their own pixels (reported by gerrit)
    - single pixel glitches are no longer fixed by copying neighbour
    - fixed glitch at image boundary correction
    - fixed memory leak in glitch correction
    - fixed horizontal line corruption in examine zoom sequence glitch
      correction (reported by Fractal universe)
    - fixed "Mandelbar Celtic" formula (reported by Kalles Fraktaler)
    - fixed "Mandelbar" formula (reported by Foxxie)
    - fixed "Burning Ship Power 4" formula (reported by Foxxie)
    - fixed complex formulas reference precision problems (reported by Foxxie)
    - fix for auto-iterations (now respects GUI) (reported by Foxxie)
    - fix for crash selecting invalid power (reported by Foxxie)
    - fix for gigantic zoom value bug in Newton zooming (reported by Foxxie and
      gerrit)
    - added "no reuse center" option to prevent rectangle pasting (suggested by
      quaz0r)
    - fix for signed integer overflow reports negative percentage in status bar
      when the number of iterations is large (reported by Foxxie and another)
    - fix for confusing PNG save options dialog called "JPEG properties"
    - major code refactoring into multiple files for ease of maintenance
    - delete no-longer-used single-threaded Newton-Raphson zooming code
```

```
675
- **kf-2.12.4** (2017-10-06)

    - "glitch low tolerance" checkbox that can be enabled to detect glitches
      more aggressively (enabling it allows "Olbaid-ST-023.kfr" to render
680   correctly, but taking 16x longer than with it disabled) (incorrect
      render reported by Kalles Fraktaler)
    - updated program icon with transparent background and large version,
      and use it for child windows too
    - long double off-by-one bug fixed (incorrect render reported by CFJH)
685 - floatexp implementation bug fixes (0.0 was implemented incorrectly)
      (caused a series approximation underskipping failure that was reported
      by Kalles Fraktaler)
    - fix distance colouring grid artifacts with small zoom size by disabling
      "reuse center" when zoom size is not an integer (reported by gerrit)
690 - fix some iteration band edge artifacts (with external postprocessing of
      kfb map files) by increasing the smooth bailout radius from 100 to 10000
      (reported by gerrit)
    - ensure added references fix their corresponding pixel (suggested by
      Pauldelbrot) (prevents loop in auto solve glitches whereby a reference
695   was repeatedly being added at the same location without progress being
      made)
    - disabled "guessing" (was causing occasional randomly bright single pixels
      at low zoom levels, possibly a race condition?)
    - build against an installed libjpeg instead of each time after clean
700 - fix broken complex formula reference calculations
    - fix compilation warnings (now almost clean with -Wall -Wextra)
    - compile as C++17 (should also still work as C++11 and C++14)
    - delete bitrotten code paths for SetEvent()-based multithreading
    - delete unused thread affinity setting code
705 - parameter data is saved as comment in image files (both PNG and JPEG)
    - can load parameters from image file comments (both PNG and JPEG)
    - delete obsolete VS build system
    - fix "infinite waves" colouring (reported by gerrit)

710 - **kf-2.12.3** (2017-09-25)

    - multiple finite difference methods for distance colouring
      (suggested by gerrit)
    - fix bug in examine zoom sequence auto solve glitches (first frame only)
715   (reported by Dinkydau and Fractal universe)
    - raise limit for maximum number of references from 199 to 10000
      (default still 69, you can change it in iterations dialog)
    - settable number of references per pass for examine zoom sequence auto
      solve glitches (default still 10)
720
- **kf-2.12.2** (2017-09-20)

    - PNG image saving support using libpng and zlib;
    - JPEG default quality to 100 (was 99);
725 - colouring uses floating point internally to reduce quantisation steps;
    - dithering at end of colouring to improve perceptual quality;
    - `formula.cpp` included in source zip so GHC is not needed unless changing
      formula code;
    - optimized `diffabs()` code: one test Burning Ship location is 7.5% faster;
730 - preprocessor optimizes reference calculations by floating temporary
      variable (re)allocations out of the inner loops: one test Burning Ship
```

```
      location is 30% faster;

- **kf-2.12.1** (2017-09-19)

    - simplified version numbering;
    - built for 64bit (as before) and 32bit (new);
    - documentation improvements;
    - fix division by zero assertion failure in File -> Examine zoom sequence;
    - fix crash in File -> Examine zoom sequence with only 1 image file;
    - adjust distance colour modes for backwards compatibility;

- **kf-2.11.1+gmp.20170913**

    - revert incompatible de log vs sqrt colouring change, instead add a new
      Distance (Logarithm) colouring method #7;
    - documentation improvements;
    - limit maximum series approximation terms to 60 to try to fix overskipping
      with large images

- **kf-2.11.1+gmp.20170822**

    - bugfix preprocessor for abs() formulas
    - de colouring with log instead of sqrt

- **kf-2.11.1+gmp.20170820**

    - bugfix preprocessor for diffabs() formulas

- **kf-2.11.1+gmp.20170714**

    - disabled OpenCL (be more compatible)

- **kf-2.11.1+gmp.20170713**

    - optimized Newton-Raphson zooming (3x faster in one test)

- **kf-2.11.1+gmp.20170711**

    - workaround for WINE issue artificially limiting image size (now bitmaps
      up to 2GiB can be created on all platforms)

- **kf-2.11.1+gmp.20170710**

    - optimized formulas (reference calculation for quadratic Mandlebrot is much
      faster due to lower-level calls to gmp)
    - very experimental opencl support (mostly broken)
    - bugfixes (fix hang loading deep zoom locations, fix newton size in new
      view radius calculation, more complete library credits in documentation)
    - prune dead code (incomplete jpeg library deleted from source, complete
      version downloaded at build time as needed, delete rudimentary openmp
      support, delete non-performant barrier variant, delete slower-than-gmp
      mpfr support, delete custom floating point support)

- **kf-2.11.1+gmp.20170703**

    - formulas now generated at compile time from formula definition XML using
      XSL stylesheet
```

```
    - used fixed format floats instead of scientific
    - try to hide command prompt window on Windows

- **kf-2.11.1+gmp.20170508**

    - restored threaded reference calculations (reimplemented with barrier()
      semantics to avoid single-threaded WINE SetEvent() rendezvous)

- **kf-2.11.1+gmp.20170504**

    - removed threaded reference calculations (too much overhead)
    - miscellaneous code cleanups (no need for -fpermissive, const fixes,
      delete[] fixes, 64bit compatibility paranoia)

- **kf-2.11.1+gmp.20170406**

    - fixed precision bugs (easy deep zoom, interactive failure)
    - fixed performance bug with inflections
    - fixed cross-hair resource bug
    - added WINDRES argument to build system
    - added more info to about dialog
    - include source code with release

- **kf-2.11.1+gmp.20170330.1**

    - fixes a crasher bug in the previous version

- **kf-2.11.1+gmp.20170330**

    - unlimited precision
    - separate compilation

- **kf-2.11.1+gmp.20170313**

    - long double compiled into exe (no dll)

- **kf-2.11.1+gmp.20170307**

    - kf-2.11.1 + gmp

- **kf-2.9.3+gmp.20170307**

    - kf-2.9.3 + gmp


## TODO

### User Interface

- crosshair cursor with more contrast (suggested by CFJH)
- show box-region for nr-zoom before clicking (suggested by
  Foxxie)
- adjust the size of the box via slider or like shift_scroll
  wheel or something like that? (suggested by Foxxie for nr-zoom, could also
  be useful for ctrl-left-click zoom)
- undo history for calculation data (suggested by TwinDragon)
- online help within program (suggested by TwinDragon)
```

```
- save image now function (without waiting for calculations)
- command line: print total runtime (suggested by gerrit)
- log window for diagnostics/debugging

### Calculations

- increase ref count limit without restarting from scratch
- increase maxiters limit without restarting from scratch
- optimize series approximation and probe point stuff
- calculate series approximation in parallel with reference
- only store reference orbit after series approximation
- refine minibrot using interior distance estimates
- refine minibrot using boundary shrinking (calculate edges only)
- use minimum |z| pixels for new references (suggested by quaz0r)

### Newton-Raphson Zooming

- add zoom depth offset in addition to zoom depth factor to (eg) zoom
  closer to minibrot or embedded Julia
- zoom to Misiurewicz points (custom zoom factor, manual preperiod
  selection) (suggested by gerrit)
- properly debug huge zoom values from size estimate
- make it work better in hard-skewed locations (need to skew the box
  period coordinates?)
- improve stopping criterion for resuming (don't over-refine)

### NanoMB

- automatic number type selection
- automatic number type promotion on overflow
- re-scaled number types
- SSA escape radius factor control (around 0.1 to 1.0)
- automatic period detection

### Preprocessor

- flatten complex numbers to separate real and imaginary parts
- common subexpression elimination (share results, might be
  especially useful for large powers of complex numbers)
- automatically parallelize reference iterations

### Colouring

- assume sRGB display and gamma-correct downscaling
  (pixman currently supports 32bit sRGB+A, but no 24bit sRGB without alpha)
- load/save palette to/from image
- rework entirely (now: 1024 colours with mandatory interpolation)
- implement Pauldelbrot's multiwave colouring
- colour cycling (suggested by blob)
- show .kfp filename in colour dialog (suggested by gerrit)
- more flexible colouring with lighting/layers/etc (suggested by Fraktalist)
- import gradients from various formats (.ugr/.gradient, .map, .xml (flam3))
  inspired by Mandel Meute + padleywood's gradient convert tool:
  <https://fractalforums.org/f/11/t/2934>

## Getting The Code
```

```
I distribute EXEs bundled together with the corresponding source code.

The latest source code is available from my git repository:

    git clone https://code.mathr.co.uk/kalles-fraktaler-2.git
    cd kalles-fraktaler-2
    git checkout master       # for Karl's original upstream
    git checkout claude       # for MINGW build system and bug fixes
    git checkout kf-2.13      # old stable (bugfixes only)
    git checkout kf-2.14      # for current development
    git tag -l                # list available release tags

You also need `et` to generate the formula code for Newton-Raphson zooming:

    git clone https://code.mathr.co.uk/et.git

This is outside the scope of this document at the moment, easiest is to
download the release bundle and copy the `formula/generated` folder from
the included src zip.  Get in touch for help if you want to regenerate
these sources yourself.

## Building On Linux

Compiling KF for your own CPU is recommended for optimal performance.  The
performance boost can be significant, as the release EXE is compiled for
generic x86_64 but newer CPUs have additional instructions available.

Note: there is an upstream bug in the GCC compiler.  On Debian the compiler
is patched in recent versions.  Patching it yourself is not hard, but it does
take a long time and need about 30GB of disk space.  Without a patched compiler,
KF may crash in SIMD code due to 32-byte aligned moves with a 16-byte aligned
stack.  See <https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=939559>.
Contact me if you need help to patch your compiler.

Build instructions for cross-compiling from GNU/Linux require about 10GB of
disk space and good internet download speed (or patience).  About 750MB of
downloads including the chroot debootstrap step.  If you have recent Debian
you can skip the chroot step and install natively.

0. Setup Debian Buster chroot:

        mkdir ./vm
        sudo debootstrap buster ./vm/
        sudo mount proc ./vm/proc -t proc
        sudo mount sysfs ./vm/sys -t sysfs
        sudo cp /etc/hosts ./vm/etc/hosts
        sudo chroot ./vm /bin/bash
        cd

1. Install dependencies (inside the chroot if you made one):

        dpkg --add-architecture i386
        echo >> /etc/apt/source.list "deb http://ftp.uk.debian.org/debian/ ↙
            ↳ experimental main contrib non-free"
        apt update
```

```
        apt install \
          build-essential \
          cabal-install \
          ghc \
          git \
          libghc-parsec3-dev \
          libtool \
          lzip \
          m4 \
          p7zip \
          pkg-config \
          wget \
          wine32 \
          wine64 \
          wine-binfmt \
          xsltproc \
          zip
        apt install -t experimental \
          mingw-w64
        apt install \
          pandoc \
          texlive-fonts-recommended \
          texlive-latex-recommended   # for PDF manual
        mount binfmt_misc /proc/sys/fs/binfmt_misc -t binfmt_misc
        update-binfmts --import /usr/share/binfmts/wine
```

   For Ubuntu replace "wine32 wine64 wine-binfmt" with "wine" (but see note about build failures with some versions).

2. Prepare non-root build user:

```
        adduser build
        # enter and confirm password
        su - build
        mkdir -p ~/win64/src
```

3. Download Kalles Fraktaler 2 + sources:

```
        cd ~/win64/src
        git clone https://code.mathr.co.uk/kalles-fraktaler-2.git
        cd kalles-fraktaler-2
        git checkout kf-2.14
```

4. Download and build and install 3rd party library sources (inspect the script if you want to be sure it isn't doing anything dodgy, or to copy/paste parts if necessary), the script can build both 64bit and 32bit variants if necessary:

```
        cd ~/win64/src/kalles-fraktaler-2
        bash ./prepare.sh dl
        bash ./prepare.sh 64
```

5. Download the previous version and copy the 'et'-generated formulas from it:

```
        cd ~/win64/src/kalles-fraktaler-2
        wget -c "https://mathr.co.uk/kf/kf-$(wget -q -O- https://mathr.co.uk/kf/↙
            ↳ VERSION.txt).7z"
        7zr x kf-*.7z
```

```
        cd kf-*/
        unzip kf-*src.zip
        cd kf-*-src/
        cp -avit ../../formula/generated formula/generated/*.c
```

6. To build Kalles Fraktaler 2 + optimized for your own CPU:

```
        cd ~/win64/src/kalles-fraktaler-2
        make clean
        make SYSTEM=native -j $(nproc)
        ./kf.exe
```

7. To build Kalles Fraktaler 2 + release:

```
        cd ~/win64/src/kalles-fraktaler-2
        ./release.sh $(git describe)
```

Note: build fails on Ubuntu 16.04.3 LTS (xenial):

```
    $ make
    x86_64-w64-mingw32-g++ -mfpmath=sse -xc++ -Wno-write-strings -pipe -MMD -g -↙
        ↳ O3 -ffast-math -I/home/claude/win64/include -↙
        ↳ DKF_THREADED_REFERENCE_BARRIER -o fraktal_sft/CDecNumber.o -c ↙
        ↳ fraktal_sft/CDecNumber.cpp
    In file included from fraktal_sft/CDecNumber.cpp:1:0:
    fraktal_sft/CDecNumber.h:5:76: error: 'decNumber' was not declared in this ↙
        ↳ scope
     typedef boost::multiprecision::number<boost::multiprecision::gmp_float<0>> ↙
         ↳ decNumber;
                                                                            ^↙
                                                                              ↳
    fraktal_sft/CDecNumber.h:5:76: error: template argument 1 is invalid
    fraktal_sft/CDecNumber.h:5:62: error: template argument 1 is invalid
     typedef boost::multiprecision::number<boost::multiprecision::gmp_float<0>> ↙
         ↳ decNumber;
                                                          ^
    fraktal_sft/CDecNumber.h:5:62: error: template argument 2 is invalid
    fraktal_sft/CDecNumber.h:5:32: warning: 'typedef' was ignored in this ↙
        ↳ declaration
     typedef boost::multiprecision::number<boost::multiprecision::gmp_float<0>> ↙
         ↳ decNumber;
                                 ^

    ...
    $ x86_64-w64-mingw32-g++ --version
    x86_64-w64-mingw32-g++ (GCC) 5.3.1 20160211
    Copyright (C) 2015 Free Software Foundation, Inc.
    This is free software; see the source for copying conditions.  There is NO
    warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The working Debian Stretch has:

```
    $ x86_64-w64-mingw32-g++ --version
    x86_64-w64-mingw32-g++ (GCC) 6.3.0 20170516
    Copyright (C) 2016 Free Software Foundation, Inc.
    This is free software; see the source for copying conditions.  There is NO
    warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
The working Debian Buster has:

    $ x86_64-w64-mingw32-g++ --version
    x86_64-w64-mingw32-g++ (GCC) 8.3-win32 20190406
    Copyright (C) 2018 Free Software Foundation, Inc.
    This is free software; see the source for copying conditions.  There is NO
    warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

The working Debian Buster with compiler from Experimental has:

    $ x86_64-w64-mingw32-g++ --version
    x86_64-w64-mingw32-g++ (GCC) 9.2-win32 20190909
    Copyright (C) 2019 Free Software Foundation, Inc.
    This is free software; see the source for copying conditions.  There is NO
    warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


## Building on Windows

(note: these instructions are out of date)

Build instructions for compiling on Windows (thanks to knighty!):

0. Remove any old msys2.

1. Downloaded latest version of msys2 (msys2-x86_64-20161025.exe).
   This is the 64 bit version. msys2-i686-20161025.exe is the 32 bit version.

2. After running it, it installs msys2. At the end the msys2 shell is launched.

3. In the msys2 shell, invoke pacman:

        pacman -Syuu

    This have to be done until is says there is nothing to do anymore.

4. Close the msys2 shell:

        exit

5. Reopen msys2 shell (from startup menu).

6. Install mingw/gcc 64 bit:

        pacman -S mingw-w64-x86_64-toolchain

    one can also install 32 bit version by:

        pacman -S mingw-w64-i686-toolchain

7. Install Boost

        pacman -S mingw-w64-x86_64-boost

    from msys shell
```

8. Close msys2 shell then open "msys2 mingw 64 bit" shell (in order to have all the environment variables properly set)

9. Change directory to the kalles fraktaler sources (where `Makefile` resides).

10. Compile

```
mingw32-make WINDRES=windres
```

(if this doesn't work edit the Makefile to replace the line

```
WINDRES ?= x86_64-w64-mingw32-windres
```

to

```
WINDRES ?= windres
```

and run `mingw32-make` without arguments)

11. Execute it this way from (msys2 mingw 64 bit) command line:

```
./fraktal_sft64    # for the claude branch
./kf.exe           # for the claude-gmp branch
```

because it is linked dynamically to some libraries. In order to execute it from the explorer one needs to copy `libgmp-10.dll` and `libwinpthread-1.dll` from `msys64/mingw64/bin` next to the generated executable.

Legal

Kalles Fraktaler 2 +
Copyright (C) 2013-2017 Karl Rummo
Copyright (C) 2017-2019 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

- this software is based in part on the work of the Independent JPEG Group
  <https://jpegclub.org/reference/libjpeg-license/>
- the TIFF library is used under the libtiff license:
  <https://gitlab.com/libtiff/libtiff/blob/master/COPYRIGHT>
- the PNG library is used under the libpng license
  <https://libpng.org/pub/png/src/libpng-LICENSE.txt>
- the ZLIB library is used under the zlib license
  <https://zlib.net/zlib_license.html>

```
- the GMP library is used under the conditions of the GNU Lesser General Public
  License version 3 and the GNU General Public License version 2
  <https://www.gnu.org/licenses/lgpl-3.0.en.html>
  <https://www.gnu.org/licenses/gpl-2.0.html>
- the MPFR library is used under the conditions of the GNU Lesser General Public
  License version 3
  <https://www.gnu.org/licenses/lgpl-3.0.en.html>
- the GSL library is used under the conditions of the GNU General Public License
  <https://www.gnu.org/licenses/gpl.html>
- the PIXMAN library is used under the conditions of the MIT License
  <https://cgit.freedesktop.org/pixman/tree/COPYING>
- the MINGW-STD-THREADS library is used under the conditions of the Simplified
  BSD License
  <https://github.com/meganz/mingw-std-threads/blob/master/LICENSE>
- the ILMBASE library is used under the conditions of the Modified BSD License
  <https://www.openexr.com/license.html>
- the OPENEXR library is used under the conditions of the Modified BSD License
  <https://www.openexr.com/license.html>
- the GLM library is used under the conditions of the MIT License
  <https://glm.g-truc.net/copying.txt>
- the BOOST library is used under the Boost Software License Version 1.0
  <https://www.boost.org/LICENSE_1_0.txt>
- the CLEW library is used under the Boost Software License Version 1.0
  <https://www.boost.org/LICENSE_1_0.txt>

**NOTE**:
If you redistribute the binaries or provide access to the binaries as a
service, you must also be prepared to distribute the source corresponding
to those binaries. To make this easier for you, the more recent zips
include the source too (though you'll also need to get the third party
library sources).


## Acknowledgements

Thanks to:

- K.I.Martin for applying Perturbation and Series Approximation on the
  Mandelbrot set and generously sharing the theory and Java source code!
- Pauldelbrot for finding the reliable glitch detection method
- Botond Kósa and knighty for the extensions of Series Approximation
- laser blaster for the Burning ship formula
- stardust4ever for other fractal types
- claude for the Newton-Raphson method
- gerrit for the distance colouring differencing variations
- Dinkydau, Fractal universe, CFJH, Foxxie and others for reporting bugs
- Chillheimer for hosting my program

Claude also thanks Karl for releasing the source code and assigning a Free
Software license.


# User Manual

## File

  - **Open**
```

```
1235    Opens the current location from a parameter file (*.kfr)
        You can also load metadata from images saved by KF.
        You can also drag-and-drop files from the file manager on to the
        main window to open them as parameters.

1240  - **Save**

        Saves the current location in the current parameter file (*.kfr)

      - **Save as**
1245
        - **KFR** Saves the current location in a new parameter file (*.kfr)

        - **PNG**
          Saves the current location in a PNG image file (*.png).
1250      The location and settings are saved in the file metadata.

        - **JPEG**
          Saves the current location in a JPEG image file (*.jpg).
          The location and settings are saved in the file metadata.
1255
        - **TIFF**
          Saves the current location in a TIFF image file (*.tif).
          The location and settings are saved in the file metadata.

1260    - **EXR**
          Saves the current location in a EXR image file (*.exr).
          The iteration data is also saved in the file.
          The location and settings are saved in the file metadata.

1265    - **KFB**
          Saves the iteration data in a map file (*.kfb). This file can be used by
          the KeyFramMovie program.  **The location and settings are not saved.**

        Saved image dimensions can be smaller than the calculated image size set in
1270    the settings.  The image will be downscaled according to the shrink quality
        setting.  If the saved image dimensions are larger, the image will be
        recalculated at the new size.

        Note that for EXR, the saved image dimensions are only for the "preview ↙
            ↳ image",
1275    the full calculated image size is used for the main image and the iteration
        data.

      - **Open map**

1280    Load the iteration data from map file (*.kfb).

        Note: you must set the aspect ratio of the window to match the KFB data
        before opening the map file.  If you have also saved images, you can do
        that by loading the image as a settings file.
1285
        Note: if you want to continue zooming from the location, you must load
        the KFR file before opening the map file.  You can also load a saved image
        as a location file.  Location information is *not* stored in KFB files.
```

Note: there is a historical accident whereby the iteration divider is saved in the KFB as an integer, losing any fractional part and sometimes resetting to 1 on load. A workaround is to load the palette from a KFP file (which is just a .kfr renamed to .kfp) after opening the map file.

- **Store zoom-out images**

  Zoom out automatically with the selected Zoom size and store JPEG/PNG/TIFF images and map file (*.kfb) for each zoom out. The zoom out stops when the depth is lower than 1. The resulting files can be used by the KeyFramMovie program to create a zoom-in animation.

- **Examine Zoom sequence**

  Make sure you store the end location as a kfr file in the same directory as you store the zoom sequence frames. This function allows you to examine the frames one by one and add references to remove eventual visible glitch blobs, or choose another pixel as the main reference.

- **Resume Zoom sequence**

  Make sure you store the end location as a kfr file in the same directory as you store the zoom sequence frames. This function allows you to resume and continue the zoom out sequnce, if it got interrupted.

- **Open Settings**

  Opens rendering settings from a settings file (*.kfs)
  You can also load metadata from images saved by KF.

- **Save Settings**

  Saves the current rendering settings to a settings file (*.kfs)

- **Check for update**

  Retrieves information from KF's homepage on the internet, to let you know if a newer version is available.

- **Exit**

  Exit this program

Fraktal

- **Presets**

  Set groups of settings to suggested preset values.

  - **Fast** accuracy may be compromised but it's fast for browsing. Sets ignore isolated glitch neighbourhood to 4, enables guessing, disables low tolerance for glitches and approximation, and disables derivatives computation unless analytic DE colouring is currently in use.

  - **Best** highest quality settings for important images, but slow. Sets ignore isolated glitch neighbourhood to 0 (disabled), disables guessing, enables low tolerance for glitches and approximation, and enables jitter

```
      with a default seed of 1.

  - **Location...**

    Displays the Location dialog where the coordinates for this location is
    displayed and can be edited.

  - **Iterations...**

    Displays the Iterations dialog where the maximum iteration number for this
    location is displayed and can be edited.

    The smooth color transition method is also set here, and the power on the
    Mandelbrot function.

    The fractal types is also set here - Mandelbrot, Burning Ship, Buffalo or
    Celtic.

    This dialog also displays

    - **Min:** The minimum iteration count for a pixel in this location

    - **Max:** The maximum iteration count for a pixel in this location

    - **Appr:** The number of iterations given by Series approximation

    - **Calculations:** The number of calculations performed and also the number↙
      ↳ of
      calculations per second is shown if this dialog is displayed while the
      image is rendered

  - **Set colors...**

    Displays the Number of colors dialog where the colors can be edited.

  - **Refresh**

    Render the current location

  - **Cancel rendering**

    Cancel the current rendering

  - **Reset**

    Set the location to the start point

  - **Undo**

    Go back to previous location(s).

  - **Redo**

    Go forward in the undo history.

  - **Copy**
```

```
    Save the current location to the system clipboard.

- **Paste**

    Set the current location from the system clipboard.

## View

- **Zoom size**

    Set the level of zoom, left mouse click to zoom in, right to zoom out

- **Animate soom**

    Turns animation on or off when zooming

- **Set window size**

    Set the size of the display window.

- **Set image size**

    Set the size of the internal image size. If this is larger than the window
    size, an anti-alias effect is achieved

- **Arbitrary size**

    (I don't know what this does?)

- **Rotate**

    Activate rotation, drag to rotate the image

- **Reset rotation**

    Clear any rotation

- **Skew**

    Opens the Skew dialog which allows to "un-skew" locations that are skewed.

    Automatic unskew is available via the Newton-Raphson zooming dialog.

- **Set Ratio**

    Enables changing the ratio between height and width of the background image
    in order to enable stretching locations. Combinated with rotation, an
    almost infinite skewing ability is enabled, useful when exploring the
    hidden treasures of the new Fractals! (This is another name for skew.)

- **Reset Ratio**

    Reset skew ratio to default

- **Skew animation**
```

```
1460    Activates or deactivas skew animation. If activated, a popup allows you to
        specify end skew parameters and number of frames. The fractal will be
        rendered frame by frame, and can be combined with frame by frame rendering
        in KeyFrameMovieMaker or MMY3D

1465  - **Shrink quality**

        Set quality of image scaling. "Fast" and "Default" are better suited
        for exploring, but "Best" looks much better at the cost of slower colouring
        (it does properly filtered anti-aliasing).
1470
        You shouldn't need to adjust this unless you set the image size larger than
        the window size, or intend saving image files at smaller resolutions than ↙
            ↳ the
        image size.

1475    Note: this feature is still a bit experimental and subject to change in
        future versions.

    ## Navigation

1480  - **Newton-Raphson zooming**

        When activated, a dialog will be displayed, which allows you to select if
        the zoom should jump directly to the minibrot, or to 3/4 zooms to the
        minibrot, where the current pattern is doubled, etc.
1485
        The zoom level of the current pattern is set when opening the dialog, and
        can be changed with the capture button (which gets the current zoom level
        from the image view).

1490    Click on the fractal to specify the start point of the search of the
        minibrot. The current zoom size is used to set the boundaries of search
        around the selected point.  It is recommended to set the zoom size to 128
        and use the crosshair window for pixel-accurate selection.

1495    Notice that it can take a long time to calculate the position of
        deep minibrots. However, that should be still much faster than
        zooming to the minibrot manually by selecting the center of the pattern
        in the view, or with the automatic search of minibrot that is also using
        the pattern center.
1500
        When "auto skew (newton)" is enabled before activating, the view will be
        skewed to make features near the minibrot approximately circular.

        When "auto skew (escape)" is activated, the view will be skewed to make
1505    features in the current view approximately circular (without zooming).

        "Auto skew escape" has an option to take into account d/dz derivatives as
        well as d/dc, some fractals work better with it enabled, some without.
        Experiment and report back!
1510
  - **Find Minibrot**

        Starts an automatic zoom-in in the image's pattern center, until a Minibrot
        is found or if it fails to find the center.
1515
```

```
    It's probably better to use Newton-Raphson zooming if possible.

  - **Non exact find Minibrot**

    Makes the Find Minibrot function fail every 20 zoom-in, in order to gain
    depth automatically without ending up in a Minibrot

    Newton-Raphson zooming may be a more useful option.

  - **Center cursor**

    Center the cursor to image's pattern center

  - **Find center of glitch (Color)**

    Centers the mouse pointer over the glitch blob found, if any

  - **Find highest iteration**

    Centers the mouse pointer over the pixel with the highest iteration

  - **Show Inflection**

    Activate or deactivate display of Inflection

  - **Show crosshair window**

    Display a small window that magnifies the area around the mouse cursor.
    Perfect for precisely picking particular pixels for zooming etc.

## Advanced

  - **Auto solve glitches**

    Turns the Auto solve glitches function on or off

  - **Solve glitch with near pixel method**

    Instead of re-render all pixels with the same iteration count value(color)
    only the connected pixels are re-rendered. On some locations other areas in
    the same view have the exact same iteration count values. These pixels may
    be correctly rendered and may be incorrect if re-rendered with another
    reference

  - **Ignore isolated small glitches**

    When enabled, ignores single-pixel glitches by interpolating their value
    from neighbouring pixels. If the image size is very large, there may be
    a very large number of these tiny glitches, whose incorrect rendering may
    be invisible to the eye, and whose correct rendering may take forever.

  - **Set main reference**

    Let you click the image and select the main reference for the whole image.
    This can be useful when glitches appears on top of minibrots when the
    reference is outside this minibrot. The glitch pattern disappears from the
    minibrot if the main reference is selected inside the minibrot.
```

```
- **Add reference (Color)**

  Add a reference and re-calculates the pixels with the same iteration count
  as the reference. This is useful if the Auto solve glitches function fails
  to find and solve glitches in the image

- **Reuse reference**

  Do not re-calculate the reference for further zooming. This can be useful
  when during automatic zoom-out and to test different reference points, but
  must not be used together with the Auto solve glitches function active.

  Note: reuse reference cannot be used for zoom sequences in which the number
  type used for calculations changes (which happens near 1e9864 1e4932 1e616
  1e308 for power 2 formulas). To avoid corrupt zoom out images when reuse
  reference is enabled, also enable "Use floatexp always" (deeper than 1e4932)
  or "Use long double always" (deeper than 1e308). This may slow down
  calculations. Alternatively, render the zoom out sequence in several
  segments, one for each number type (floatexp, scaled long double, long
  double, scaled double, double).

- **No reuse center**

  Don't paste the previous image in the middle when zooming out. Disabling
  this (ie, do reuse center) can be faster but can also lead to bad images.

- **Show iterations**

  Displays the image black-and-white with the pixels with the highest
  iteration as white and the pixels with the lowest iteration as black

- **Show smooth transition colors**

  Displays the image black-and-white representing the smoothing coefficient

- **Show glitches**

  When activated, glitches are displayed with a solid color

- **Mirror**

  mirrors the image around the x-axis. Can be used on the deeper half of a
  zoom sequence to a minibrot - but not too close to the minibrot and too
  close to the half...

- **No series approximation**

  Turns the Series approximation function on or off.

- **Use long double always**

  Use always the 80-bit long double hardware data type. This can solve some
  type of glitches

- **Use floatexp always**
```

```
1630      Use always the double mantissa/integer exponent data type. This probably
          only make the render slower

      - **Multi-threaded reference calcs**

1635      Use multiple threads when calculating reference orbits
          (available for long double and floatexp only).
          Speed and CPU efficiency may differ when toggling this flag;
          evaluate the optimal choice for your location and system.

1640  - **Use NanoMB1 (experimental)**

          For power 2 Mandelbrot only.

          Use knighty's experimental NanoMB1 algorithm for bivariate super-
1645      series-approximation. Calculations are done with floatexp always.
          A regular series approximation plus perturbation pass follows for
          glitch correction.

          It is required to set the period in the Location dialog before
1650      enabling NanoMB1. Using Newton zoom sets the period automatically.

          It is recommended to enable Reuse Reference after recalculating with
          NanoMB1. A new reference is calculated automatically after each Newton
          zooming, whatever the setting of Reuse Reference.
1655
          Whether NanoMB1 is faster or not depends heavily on the location: views
          close to minis should be significantly faster than the regular 'fast'
          preset.

1660  - **Use NanoMB2 (experimental)**

          For power 2 Mandelbrot only.

          Use knighty's experimental NanoMB2 algorithm for bivariate super-
1665      series-approximation. Calculations are done with floatexp always,
          glitch detection and correction is disabled (and may be unnecessary?).

          It is recommended to set the period limit in the Location dialog before
          enabling NanoMB2, otherwise it uses the maximum iteration count which
1670      may take significantly longer. Using Newton zoom sets the period limit
          automatically.

          It is recommended to enable Reuse Reference after recalculating with
          NanoMB2. A new reference is calculated automatically after each Newton
1675      zooming, whatever the setting of Reuse Reference.

          Whether NanoMB2 is faster or not depends heavily on the location: views
          close to minis should be significantly faster than the regular 'fast'
          preset.
1680
          The RadiusScale field in the .kfs settings file controls the scaling of
          the escape radius calculated for each minibrot in the chain. Increasing
          it may be faster but lead to visible distortion, decreasing it may help
          the distortion but slow things down. No GUI for this control yet.
1685
      - **Interior checking (NanoMB only)**
```

```
    For power 2 Mandelbrot rendered with NanoMB1 or NanoMB2 only.

    Use an interior checking algorithm, which may or may not speed up
    per-pixel calculations in locations with large interior regions
    visible.

    Experimental. Correctness is to be evaluated. Subject to change.

- **Use auto iterations**

    Turns automatic iteration control on or off. This is on per default.

- **Use guessing**

    Enable interpolation of neighbouring pixel data when the iteration count
    is the same. This speeds up rendering of interior regions, but some
    colouring can lead to visible artifacts in the exterior.

- **Half-float image buffer**

    Pre-allocate 16bit image buffer for EXR export. Otherwise it is only
    allocated when needed. May speed up EXR export if this is checked.
    If not using EXR, leave this unchecked.

- **'Save' overwrites existing file**

    When this is checked, Save (Ctrl-S) overwrites the current file without
    asking. When this is unchecked, Save will add a timestamp to file names
    to prevent accidental data loss.

- **SIMD vector size**

    Number of pixels to calculate simultaneously per core. Setting it too
    small or too large will lead to slower performance. The sweet spot will
    depend on your particular CPU model, cache sizes, how KF is compiled (e.g.
    64 vs native), etc.

    Setting it to 1 disables the explicitly vectorized code path; some SIMD
    instructions may still be used depending on the compiler.

- **SIMD chunk size**

    Number of iterations to calculate between escape/glitch checkes. Setting
    it too small or too large will lead to slower performance, because checks
    will be done too often (small chunk size) or too many extra iterations will
    be done per pixel (large chunk size). The sweet spot will depend on your
    CPU and the location (if the number of iterations is high, probably a higher
    chunk size would be beneficial).

    How it works: the last chunk (when any of the SIMD vector escape to infinity
    or glitch detected) is repeated for each pixel in the vector individually,
    after rolling back to before the last chunk. This means some iterations are
    repeated, but the speedup from doing escape checks less frequently may
    overcome this issue.

- **Threads per CPU**
```

```
1745    The default of 1 is sensible if KF is the only thing you're running.
        Increase the count if you want KF to peg your system, decrease it if
        you want to do other things at the same time.

        The last menu item reduces the calculated number of threads by 1,
1750    which can help improve system responsiveness in some configurations.

        There will always be at least one thread.

        These options only affect perturbation rendering and image colouring,
1755    not Newton-Raphson zooming or reference calculations.

        Note: changing this during rendering could lead to crashes, so the menu
        is disabled during rendering to prevent that.

1760 ## About

     At the very top right:

       - **?**
1765
         Open about dialog, with version information and credits.

         This also functions as a lock mechanism, preventing accidental zooming
         while a long render is taking place.
1770
     ## Iterations dialog

       - Number of iterations. Increase this if the interior is "blobby".

1775   - Minimum iteration count achieved in the image (display only).

       - Maximum iteration count achieved in the image (display only).

       - Series approximation iteration count (display only).
1780
       - Smooth method

           - High bailout: large escape radius gives a smoother appearance.

1785       - Bailout=2: small escape radius can help finding features.

       - Value of the power `p` in the fractal formula (not used in every
         formula)

1790   - Fractal type (formula name). See below for details.

       - Maximum number of secondary reference points for automatic glitch
         correction. There is a hard limit of 10000, which is also the
         default.
1795
       - Glitch low tolerance. When checked, glitches are more likely to be
         detected. Disabling it can lead to bad images, but is faster.

       - Series approximation low tolerance. When checked, series approximation
1800     is stricter. Disabling it can lead to bad images, but is faster.
```

- Automatic approximation terms based on number of pixels.

- Approximation terms: number of terms used for series approximation.

- Calculations per second (display only).

- Real and Imag checkboxes: use these parts of ‘z‘ when considering bailout past the escape radius.

- Seed R and I number boxes: start iterating ‘z‘ from these coordinates (default ‘0 + 0 i‘, for best semantics it should be a critical point of the iteration formula, where it’s ‘d/dz‘ derivative is zero).

- Factor ‘a‘ R and I number boxes: set the complex number ‘a‘ (denoted ‘f = d + e i‘ in the formula list below) for TheRedshiftRider formulas.

- Jitter seed: non-zero enables jitter with a pseudo-random-number generator seed value.

- Jitter scale in units of a pixel (1 pixel is sensible in most cases).

- Gaussian jitter is probably best left disabled (uniform jitter looks better).

- Derivatives calculation can be enabled (if needed for analytic DE colouring) or disabled (speeds up rendering).

Formulas

Notation:

```
i            -- imaginary unit (square root of -1)
p            -- integer power between 2 and 5 (10 for Mandelbrot)
c = a + i b  -- pixel coordinates (parameter plane)
z = x + i y  -- iteration variable
w = u + i v  -- temporary variable for two-stage formulas
f = d + i e  -- constant 'a' for TheRedshiftRider formulas
l m ...      -- juxtaposition is multiplication
^            -- raise an expression to a positive integer power
|.|          -- surrounding a real-valued expression: absolute value
```

Formulas:

- Mandelbrot

```
z := z^p + c
```

- Burning Ship

```
z := (|x| + i |y|)^p + c
```

- Buffalo

```
w := z^p
z := (|u| + i |v|) + c
```

- Celtic

```
w := z^p
z := (|u| + i v) + c
```

- Mandelbar

```
z := (x - i y)^p + c
```

- Mandelbar Celtic

```
w := (x - i y)^2
z := (|u| + i v) + c
```

- Perpendicular Mandelbrot

```
z := (|x| - i y)^2 + c
```

- Perpendicular Burning Ship

```
z := (x - i |y|)^2 + c
```

- Perpendicular Celtic

```
w := (|x| - i y)^2
z := (|u| + i v) + c
```

- Perpendicular Buffalo

```
w := (x - i |y|)^2
z := (|u| + i v) + c
```

- Cubic Quasi Burning Ship

```
z := (|x| (x^2 - 3 y^2) - i |y (3 x^2 - y^2)|) + c
```

- Cubic Partial BS Real

```
z := (|x| (x^2 - 3 y^2) + i y (3 x^2 - y^2)) + c
```

- Cubic Partial BS Imag

```
z := (x (x^2 - 3 y^2) + i |y| (3 x^2 - y^2)) + c
```

- Cubic Flying Squirrel (Buffalo Imag)

```
z := (x (x^2 - 3 y^2) + i |y (3 x^2 - y^2)|) + c
```

- Cubic Quasi Perpendicular

```
z := (|x| (x^2 - 3 y^2) - i y |3 x^2 - y^2|) + c
```

- 4th Burning Ship Partial Imag

```
z := (x + i |y|)^4 + c
```

- 4th Burning Ship Partial Real

```
            z := (|x| + i y)^4 + c
```

- 4th Burning Ship Partial Real Mbar

```
            z := (|x| - i y)^4 + c
```

- 4th Celtic Burning Ship Partial Imag

```
            w := (x + i |y|)^4
            z := (|u| + i v) + c
```

- 4th Celtic Burning Ship Partial Real

```
            w := (|x| + i y)^4
            z := (|u| + i v) + c
```

- 4th Celtic Burning Ship Partial Real Mbar

```
            w := (|x| - i |y|)^4
            z := (|u| + i v) + c
```

- 4th Buffalo Partial Imag

```
            w := z^4
            z := (u + i |v|) + c
```

- 4th Celtic Mbar

```
            w := (x - i y)^4
            z := (|u| + i v) + c
```

- 4th False Quasi Perpendicular

```
            z := ((x^4 + y^4 - 6 x^2 y^2) - i 4 x y |x^2 - y^2|) + c
```

- 4th False Quasi Heart

```
            z := ((x^4 + y^4 - 6 x^2 y^2) + i 4 x y |x^2 - y^2|) + c
```

- 4th Celtic False Quasi Perpendicular

```
            z := (|x^4 + y^4 - 6 x^2 y^2| - i 4 x y |x^2 - y^2|) + c
```

- 4th Celtic False Quasi Heart

```
            z := (|x^4 + y^4 - 6 x^2 y^2| + i 4 x y |x^2 - y^2|) + c
```

- 5th Burning Ship Partial

```
            z := (|x| + i y)^5 + c
```

- 5th Burning Ship Partial Mbar

```
            z := (|x| - i y)^5 + c
```

- 5th Celtic Mbar

```
        w := (x - i y)^5
        z := (|u| + i v) + c
```

- 5th Quasi Burning Ship (BS/Buffalo Hybrid)

```
        w := (|x| + i y)^5
        z := (u - i |v|) + c
```

- 5th Quasi Perpendicular

```
        z := (|x| (x^4 + 5 y^4 - 10 x^2 y^2) - i y (|5 x^4 + y^4 - 10 x^2 y^
            ↳ 2|)) + c
```

- 5th Quasi Heart

```
        z := (|x| (x^4 + 5 y^4 - 10 x^2 y^2) + i y (|5 x^4 + y^4 - 10 x^2 y^
            ↳ 2|)) + c
```

- SimonBrot 4th

```
        z := z^2 (|x| + i |y|)^2 + c
```

- 4th Imag Quasi Perpendicular / Heart

```
        z := ((x^4 + y^4 - 6 x^2 y^2) + i 4 x |y (x^2 - y^2)|) + c
```

- 4th Real Quasi Perpendicular

```
        z := ((x^4 + y^4 - 6 x^2 y^2) - i 4 y |x (x^2 - y^2)|) + c
```

- 4th Real Quasi Heart

```
        z := ((x^4 + y^4 - 6 x^2 y^2) + i 4 y |x (x^2 - y^2)|) + c
```

- 4th Celtic Imag Quasi Perpendicular / Heart

```
        z := (|x^4 + y^4 - 6 x^2 y^2| + i 4 x |y (x^2 - y^2)|) + c
```

- 4th Celtic Real Quasi Perpendicular

```
        z := (|x^4 + y^4 - 6 x^2 y^2| - i 4 y |x (x^2 - y^2)|) + c
```

- 4th Celtic Real Quasi Heart

```
        z := (|x^4 + y^4 - 6 x^2 y^2| + i 4 y |x (x^2 - y^2)|) + c
```

- SimonBrot 6th

```
        z := z^3 (|x| + i |y|)^3 + c
```

- HPDZ Buffalo

```
        z := (((x^2 - y^2) - |x|) + i (|2xy| - |y|)) + c
```

- TheRedshiftRider 1: `a*z^2+z^3+c`

```
        z := (f z^2 + z^3) + c
```

- TheRedshiftRider 2: `a*z^2-z^3+c`

```
        z := (f z^2 - z^3) + c
```

- TheRedshiftRider 3: `2*z^2-z^3+c`

```
        z := (2 z^2 - z^3) + c
```

- TheRedshiftRider 4: `a*z^2+z^4+c`

```
        z := (f z^2 + z^4) + c
```

- TheRedshiftRider 5: `a*z^2-z^4+c`

```
        z := (f z^2 - z^4) + c
```

- TheRedshiftRider 6: `a*z^2+z^5+c`

```
        z := (f z^2 + z^5) + c
```

- TheRedshiftRider 7: `a*z^2-z^5+c`

```
        z := (f z^2 - z^5) + c
```

- TheRedshiftRider 8: `a*z^2+z^6+c`

```
        z := (f z^2 + z^6) + c
```

- TheRedshiftRider 9: `a*z^2-z^6+c`

```
        z := (f z^2 - z^6) + c
```

- SimonBrot2 4th

```
        w := z^2
        z := w (|u| + i |v|) + c
```

- General Quadratic Minus

```
        z := ((x^2 - y^2) + i (2 d x y + e x^2)) + c
```

- General Quadratic Plus

```
        z := ((x^2 + y^2) + i (2 d x y + e x^2)) + c
```

- Mothbrot 2nd 1x1

```
        z := z^1 (|x| + i |y|)^1 + c
```

- Mothbrot 3rd 1x2

```
        z := z^1 (|x| + i |y|)^2 + c
```

- Mothbrot 3rd 2x1

```
        z := z^2 (|x| + i |y|)^1 + c
```

- Mothbrot 4th 1x3

```
        z := z^1 (|x| + i |y|)^3 + c
```

- Mothbrot 4th 2x2 (aka SimonBrot 4th)

```
        z := z^2 (|x| + i |y|)^2 + c
```

- Mothbrot 4th 3x1

```
        z := z^3 (|x| + i |y|)^1 + c
```

- Mothbrot 5th 1x4

```
        z := z^1 (|x| + i |y|)^4 + c
```

- Mothbrot 5th 2x3

```
        z := z^2 (|x| + i |y|)^3 + c
```

- Mothbrot 5th 3x2

```
        z := z^3 (|x| + i |y|)^2 + c
```

- Mothbrot 5th 4x1

```
        z := z^4 (|x| + i |y|)^1 + c
```

- Mothbrot 6th 1x5

```
        z := z^1 (|x| + i |y|)^5 + c
```

- Mothbrot 6th 2x4 (Simon's Mothbrot)

```
        z := z^2 (|x| + i |y|)^4 + c
```

- Mothbrot 6th 3x3 (aka SimonBrot 6th)

```
        z := z^3 (|x| + i |y|)^3 + c
```

- Mothbrot 6th 4x2

```
        z := z^4 (|x| + i |y|)^2 + c
```

- Mothbrot 6th 5x1

```
        z := z^5 (|x| + i |y|)^1 + c
```

- General Abs Quadratic Minus

```
        z := ((x^2 - y^2) + i (2 d |x y| + e x^2)) + c
```

- General Abs Quadratic Plus

```
        z := ((x^2 + y^2) + i (2 d |x y| + e x^2)) + c
```

```
A machine-readable version of this formula list is found in the 'et' repository:
<https://code.mathr.co.uk/et/blob/kf:/kf/formulas.et> This is used by 'et' when
generating formula code (for Newton-Raphson zooming, etc).

## Number of colors dialog

  - **Number of key colors**

    Set the number of key colors between 1 and 1024.

  - **Divide iteration**

    Divide each iteration number with this value, for dense images this value
    can be greater than 1.  For DE, values less than 1 can be useful.

  - **Color offset**

    Offset the colors in the palette

  - **Random**

    Fill the palette with random colors made from the Seed value. The Seed
    button select a seed value randomly.

  - **More contrast**

    Move RGB values closer to max or min

  - **Less contrast**

    Move RGB values closer to the middle

  - **Show slopes**

    Enable slope encoding for 3D effect.

    First value is the magnification of the slopes. The start value of 100 is
    suitable for the unzoomed view. Deep views requires a couple of magnitudes
    higher value.

    The second value is the percentage with which the slope encoding is applied
    on the coloring. 100 is max, however flat areas will still have the palette
    color visible.

  - **Save palette**

    Save the current palette in KFP (*.kfp) file

  - **Open palette**

    Load palette from a KFP (*.kfp) file

  - **Expand double**

    Double the number of key colors without changing the palette. This allows
```

```
    finer control of individual colors without changing the palette for other
    colors

- **Expand all**

    Increase the number of key color to maximum 1024 without changing the
    palette

- **Double**

    Double the key colors by repeating them

- **Merge Colors**

    Allows a selected color to be merged to every specied key color

- **Show index**

    Capture the mouse, hover the mouse over the fractal image and the
    corresponding color in the list will be highlighted. Click and the color
    selection dialog will be displayed for the active color

- **Smooth color transition**

    Makes the transitions of colors smooth

- **Inverse smooth color transition**

    Inverse the smooth color transition which makes edges more visible

- **Unnamed dropdown box**

    Specifies handling of the iteration count values prior to coloring

- **Palette waves**

    The palette can be filled from sine waves applied on Red, Green, Blue and
    Black-and-white. Each input box specifies the number of periods applied on
    the number of key colors in the palette. If the input box is left empty, no
    wave of this color is applied. At right of each input box the "P"-button
    makes the number you entered prime, since different prime numbers probably
    give more variation. The last input box specifies the waves offset.

    The button "Generate" applies the waves on the palette, the "Seed" button
    fills the fields with random values

- **Infinite waves**

    Waves can be applied on Hue, Saturation and Brightness rather than RGB
    values. The Period value specifies the length of the period (not the
    number of periods as for the Palette waves). Periods with prime numbers
    should be able to produce an infinite number unique colors

    A negative value on Hue, Saturation or Brightness makes a flat percentage
    value to be applied on all iterations.
```

Command Line Usage

```
kf.exe [options]
    -o, --load-map       [FILE.kfb]  load map file
    -c, --load-palette   [FILE.kfp]  load palette file
    -l, --load-location  [FILE.kfr]  load location file
    -s, --load-settings  [FILE.kfs]  load settings file
    -x, --save-exr       [FILE.exr]  save EXR
    -t, --save-tif       [FILE.tif]  save TIFF
    -p, --save-png       [FILE.png]  save PNG
    -j, --save-jpg       [FILE.jpg]  save JPEG
    -m, --save-map       [FILE.kfb]  save KFB
    -z, --zoom-out       [NFRAMES]   zoom sequence
    --log (debug|status|info|warn|error)
                                     logging verbosity
    -v, -V, --version                show version
    -h, -H, -?, --help               show this help
```

Locations and settings can also be image files with embedded comments.

If any of the save options are give, KF switches to a non-interactive mode - it will render the image and save to all specified types before quitting. No GUI.

A typical workflow would be to start KF without arguments, set up the window size (eg 640x360), image size (eg 3840x2160), glitch low tolerance flag, etc, then save the settings to a .kfs file, before quitting.

Then launch KF from the command line telling it to load the settings file you just saved, plus the location file you want to render, and where to save the output images to. Then wait patiently. You can write a script that renders multiple locations in succession, either in batch .BAT on Windows, or in Shell .sh on *nix with WINE.

Note that you might have to double up backslashes within quoted paths (if they contain spaces). Maybe forward slashes work instead, but you do need quotes (either single `''` or double `""`, in a matching pair around the whole argument if there are spaces. Your shell might also do funky stuff with slashes and quotes, so you might need to double up or quadruple up the backslashes and quotes. Easiest to avoid spaces and keep your files in the current working directory... Example:

```
kf.exe -s settings.kfs -l location.kfr -p out.png -j out.jpg -m out.kfb
```

Use `--log info` to disable the status updates, use `--log warn` to output only important messages. The default is `--log status`.

With `-z` you can specify how many frames to render, or `-1` to zoom all the way out. In zoom out mode the PNG, JPEG, and KFB filenames should contain a printf flag for an integer, for example `image-%08d.png` will have 8 decimal digits padded with leading 0. This is filled by the frame number, which always starts from 0. Zooming is by the zoom size in the settings file.

New in 2.14.6 is standalone KFB map colouring support with the `-o`/`--load-map` flag:

```
kf.exe -o map.kfb -c palette.kfp -p out.png
```

```
### Tiled Rendering

Tiled rendering is useful for large images, including images larger than the
2GB pixel data limit of Windows bitmaps.  The currently supported way of
rendering tiled images is via EXR files, though there are some Octave/Matlab
scripts still available that can work with more common image files.

Example: target final size 64000x36000, tile factor 10x10.  Configure image
size to 6400x3600, adjust other settings, save `input.kfs` and `input.kfr`.
Then run (syntax for Bash shell):

    kf-tile.exe input.kfs input.kfr 10
    for tile in input-*.kfs
    do
      kf.exe -s ${tile} -l ${tile%.kfs}.kfr --save-exr ${tile%.kfs}.exr
    done
    exrtactile.exe input 10 0 output.exr  # for side-by-side tiles
    exrtactile.exe input 10 1 output.exr  # for stratified tiles

The EXR tile assembler is available at <https://mathr.co.uk/exrtact>
including Windows program binary.

## Third-party Software

### exrtact

<https://mathr.co.uk/exrtact>

exrtact is a suite of small command-line programs for manipulating EXR
files.  Of particular note is `exrtactile` for tile assembly (see above).

### kf-extras

<https://code.mathr.co.uk/kf-extras/blob/HEAD:/README>

    git clone https://code.mathr.co.uk/kf-extras.git

kf-extras is a suite of small command-line programs for manipulating KFB
files.  Programs include colouring algorithms (curvature, histogram,
pseudo-de, rainbow, stretch) and other tools (de-histogram, expmap, resize,
statistics, to-exr, to-mmit) as well as a bash script to generate a zoom
video (which needs the mightymandel zoom assembler below).  The programs
use a small library to abstract the KFB and PPM reading and writing, which
can be used as a basis for your own programs.

### zoom-tools zoom-interpolator

<https://mathr.co.uk/zoom>

zoom-tools is a suite of small command-line programs for turning EXR zoom
sequences into zoom videos.  It has filters for input keyframes and output
video frames, allowing video files to be encoded without needing so much
temporary disk space.

There is also an experimental OpenGL version that allows colouring with a
fragment shader, which should be significantly faster and more flexible (if
```

you don't mind a bit of coding).

After Effects zoom tools

<https://fractalforums.org/f/15/t/2965>

saka on fractalforums.org is working on a plugin for Adobe After Effects:

> I'm working on a plug-in for Adobe After Effects. It renders from the
> .kfb file data generated by Kalles Fraktaler. I will release the
> plug-in as open-source, but of course Adobe products are not free.

> It's still very much a work in progress, but I thought I'd share in
> case anyone is interested in trying it out at this early stage.
> It might have some bugs.

> Why After Effects? It provides a pretty solid animation, layering and
> rendering engine. All the settings can be animated, and it is easy to
> build layered compositions using KF data. Plus having 4k and 8k export
> in various codecs is appealing.

> Requirements: After Effects 2019, Windows only.

See fractalforums.org for downloads and tutorials.

Key Frames Movie Maker

<http://www.chillheimer.de/kallesfraktaler>

The original Movie Maker by Karl Runmo. Has controls for changing speed, rotation, colour cycling, and more, at specific frames. Also has an option for zoom depth text overlay. Works from KFB files and generates multiple video file segments.

Yann Le Bihan's Movie Maker 3D

<http://www.chillheimer.de/kallesfraktaler/mmy/>

Works from KFB files and generates video segments with a 3D effect:

> This Manual is formed as a tutorial describing the steps to make a
> movie with 3D effect, which includes all settings.

Dinkydau's Mandel Machine Convertor

Mandel Machine is a fast Mandelbrot renderer for Windows written in Java and assembly.

Dinkydau on fractalforums.org has written:

> something to convert the MMI file format to KFB

so that Mandel Machine glitches can be corrected in KF or KF renders can be coloured in Mandel Machine. But Dinkydau has not released it. Maybe if you ask nicely you will be granted access. More information at:

<https://fractalforums.org/f/66/t/769/msg3960#msg3960>

```
2425
    ### SeryZone's Map Visualizer and Fractal Video Maker

    Dinkydau on fractalforums.org has written:

2430 > My favorite zoom video maker (it only creates PNG sequences) is
    > SeryZone's kfb map visualiser.  It's hard to use because of the very
    > not-intuitive interface though.

    <http://www.fractalforums.com/announcements-and-news/sft-map-visualizer>
2435
    > With this tool you can create PNG images from KFB maps and use them as
    > keyframes with the other tool fractal video maker.  Use fractal video
    > maker to interpolate extra png files, for example 60 between each
    > keyframe for a 60 fps zoom video at a speed of 1 zoom per second.
2440 > (where by "zoom" I mean an increase in  magnification by a factor 2,
    > which is the recommended zoom size in kalles fraktaler)

    <https://fractalforums.org/f/15/t/1364/msg7023#msg7023>

2445 Later in that thread are download links for the tools.

    ### mightymandel zoom interpolator

    mightymandel is a Mandelbrot set renderer using OpenGL.
2450
    <https://mightymandel.mathr.co.uk/>

    It has a zoom assembler in its `extra` subfolder.

2455 <https://mightymandel.mathr.co.uk/current/usage.html#usagezoom>

        usage: zoom iwidth iheight iframes olength [ofps [oshutter]] \
          < stream.ppm > stream.y4m

2460 It is a bit picky about the format of the input PPM stream, and input frames
    must be 2x zoomed at each frame from zoomed out to zoomed in.  Arguments:

    iwidth: input frame width

2465 iheight: input frame height

    iframes: input frame count

    olength: output video length in seconds
2470
    ofps: output frame rate (25fps default)

    oshutter: motion blur (0 no blur, 1 full blur, 0.5 default)

2475 You might need to downscale the input PPM stream to something your GPU can
    manage, noting that the internal code needs a texture 2x bigger than the
    input/output frames.  Up to 4096x4096 input should be ok on most modern
    cards.  The output dimensions are the same as the input.  For best quality
    downscale after zoom interpolation, not before, if possible.
2480
    ### et zoom interpolator
```

et is an escape time fractal project implemented in Haskell.

<https://mathr.co.uk/et>

It has a forked version of the mightymandel zoom interpolator optimized for distance estimation rendering. Usage is the same as the mightymandel zoom interpolator except that the input is raw headerless floating point data (single channel containing distance estimate, normalized to 1 for the neighbouring pixel).

The difference to the mightymandel zoom interpolator is that it interpolates the DE data before colouring instead of after, which gives a much smoother image (no strobing filaments). Output size is hardcoded to 1920x1080, you can edit this near the top of the `main()` function in `zoom.c` before compiling. Input frame size should be as large as possible for best quality.

The program has two more arguments compared to the mightymandel zoom interpolator: whether to output a header (set to 1 unless you know what you are doing) and an "inverse video" flag that makes it white on black instead of black on white. The output Y4M stream is mono, you might have to tell ffmpeg to encode `-pix_fmt yuv420p` for some codecs.

```
usage: zoom iwidth iheight iframes olength \
  [ofps [oshutter [header [inverse]]]] < stream.raw > stream.y4m
```

"book" zoom interpolator

The mightymandel zoom interpolator is better in every way (it is based on the same code, just enhanced). And the book repository is 60MB because it has images. But if you want to study it for historical reference:

```
wget -O Makefile https://code.mathr.co.uk/book/blob_plain/HEAD:/code/Makefile
wget -O zoom.c https://code.mathr.co.uk/book/blob_plain/HEAD:/code/zoom.c
make zoom
```

Or using git (note: the repository is large):

```
git clone https://code.mathr.co.uk/book.git
cd book/code
make zoom
```

Usage:

```
./zoom iwidth iheight iframes olength < stream.ppm > stream.y4m
```

File Formats

EXR Channels and Metadata

Discussion about a common format for escape time fractal data is taking place here:

<https://fractalforums.org/f/11/t/3017>

Here are the channels and metadata that KF currently supports:

```
- RGB half float colours

- normalized iteration count

    `uint32 N` integer iteration count

    - `0xFFFFFFFF` is non-escaped before header metadata field `int Iterations`
      (or `string Iterations`, as it can exceed the range of `int`)

    - 0x00000000 is uncalculated/glitch/no-data-available.

    - If actual iteration values can be zero or negative, add a bias
      constant to each count and store it in the header metadata field
      `int IterationsBias` (or `string IterationsBias`, it can exceed the
      range of `int`). The bias could be negative, this might allow you
      to store high iteration counts without necessarily needing two channels
      if the actual min/max range is small enough)

    For images with biased iteration counts above `0xFFFFFFFE`, split into
    two channels:

    `uint32 N0` least significant 32 bits

    `uint32 N1` most significant 32 bits

    `(0xFFFFFFFF, 0xFFFFFFFF)` is interpreted as non-escaped

    For future supercomputers, this can be extended with `N2` etc...

    `float NF` fractional iteration count, expected to be in [0.0 .. 1.0)

    The continuous iteration count (when escaped) is `N+NF-IterationsBias`.
    This is stored separately to avoid losing precision at high iteration counts

- directional DE (when derivatives have been calculated)

    `float DEX`, `float DEY` directional distance estimate in cartesian
    form, normalized such that distance to a neighbouring boundary pixel
    `sqrt(DEX^2 + DEY^2)` is approximately `1.0`.

    If some pixels have no directional DE the missing data can be written
    as `(0.0, 0.0)`, but readers should also handle `NaN`s in this case.
    The vector points away from the fractal boundary.
```

118 release.sh

```
#!/bin/sh
# Kalles Fraktaler 2
# Copyright (C) 2013-2017 Karl Rummo
# Copyright (C) 2017-2019 Claude Heiland-Allen
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
# published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
```

```
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
set -e

VERSION="${1}"
if [ -z "${VERSION}" ]
then
        echo "usage: ${0} version"
        exit 1
fi

make clean

SRC="${VERSION}-src"
mkdir "${SRC}"
make formula/formula.cpp
cp -avit "${SRC}/" fraktal_sft formula cl common utils preprocessor.hs Makefile ↙
    ↳ 32.mk 64.mk 64+.mk native.mk README.md LICENSE.md prepare.sh "${0}"
zip -0 -r "${SRC}.zip" "${SRC}/"
BIN="${VERSION}"
mkdir "${BIN}"
cp -avit "${BIN}/" "${SRC}.zip" utils/stratify.m utils/resizeKFB.m

make -j 32 SYSTEM=64
strip kf.exe
strip kf-tile.exe
cp -avi kf.exe "${BIN}/kf.exe"
cp -avi kf-tile.exe "${BIN}/kf-tile.exe"

make README.pdf
make LICENSE.pdf

cp -avi README.md "${BIN}/kf.txt"
cp -avi README.pdf "${BIN}/kf.pdf"
cp -avi LICENSE.md "${BIN}/LICENSE.txt"
cp -avi LICENSE.pdf "${BIN}/LICENSE.pdf"
7zr a -t7z -m0=lzma -mx=9 -mfb=64 -md=32m -ms=on "${BIN}.7z" "${BIN}/"
gpg -b "${BIN}.7z"

ls -1sh "${BIN}.7z"*
```

119 utils/kf-tile.cpp

```
/*
Kalles Fraktaler 2
Copyright (C) 2013-2017 Karl Runmo
Copyright (C) 2017-2018 Claude Heiland-Allen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.
```

```
10
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.
15
    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */

20  #include <algorithm>
    #include <cmath>
    #include <cstdlib>
    #include <iomanip>
    #include <iostream>
25  #include <fstream>
    #include <sstream>
    #include <string>
    #include <utility> // pair
    #include <vector>
30
    #include <boost/multiprecision/mpfr.hpp>
    typedef boost::multiprecision::number<boost::multiprecision::mpfr_float_backend
        ↪ <53>> mpfr53;
    typedef boost::multiprecision::number<boost::multiprecision::mpfr_float_backend
        ↪ <0>> mpfr;
    #define LOG_10_2 0.30102999566398114
35
    std::string fixed(const mpfr &f)
    {
      std::ostringstream os;
      os << std::fixed << std::setprecision(f.precision() + 3) << f;
40    std::string s = os.str();
      std::string::size_type e = s.find('e');
      if (e != std::string::npos)
        s[e] = 'E';
      return s;
45  }

    std::string scientific(const mpfr53 &f)
    {
      std::ostringstream os;
50    os << std::scientific << std::setprecision(int(LOG_10_2 * 53 + 3)) << f;
      std::string s = os.str();
      std::string::size_type e = s.find('e');
      if (e != std::string::npos)
        s[e] = 'E';
55    e = s.find('E');
      if (e != std::string::npos)
        if (s[e + 1] == '+')
          s.erase(e + 1, 1);
      return s;
60  }

    void usage(const char *progname)
    {
      std::cerr << "usage: " << progname << " settings.kfs parameter.kfr factor" << ↙
```

```
      ↪ std::endl;
}

typedef std::pair<std::string, std::string> KV;
typedef std::vector<KV> KVV;

KVV read(const std::string &filename)
{
  KVV kvv;
  std::ifstream input(filename);
  std::string line;
  while (std::getline(input, line))
  {
    std::string::size_type ix = line.find(':');
    if (ix != std::string::npos)
    {
      std::string key = line.substr(0, ix);
      std::string val = line.substr(ix + 2, std::string::npos);
      KV kv(key, val);
      kvv.push_back(kv);
    }
  }
  return kvv;
}

void write(const std::string &filename, const KVV &kvv)
{
  std::ofstream output(filename);
  for (auto kv : kvv)
    output << kv.first << ": " << kv.second << std::endl;
}

std::string lookup(const KVV &kvv, const std::string &key)
{
  for (auto kv : kvv)
    if (kv.first == key)
      return kv.second;
  return "";
}

KVV &update(KVV &kvv, const std::string &key, const std::string &val)
{
  for (size_t i = 0; i < kvv.size(); ++i)
    if (kvv[i].first == key)
    {
      kvv[i].second = val;
      return kvv;
    }
  // not found, insert it
  KV kv(key, val);
  kvv.push_back(kv);
  return kvv;
}

extern int main(int argc, char **argv)
{
  if (argc != 4)
```

```
  {
    usage(argv[0]);
    return 1;
  }
  const std::string kfsname(argv[1]);
  const std::string kfrname(argv[2]);
  const int factor = std::atoi(argv[3]);
  if (factor <= 0)
  {
    std::cerr << argv[0] << ": factor must be > 0 (got `" << factor << "')" << ↙
        ↪ std::endl;
    return 1;
  }
  // parse settings
  KVV kfs = read(kfsname);
  const int width = std::atoi(lookup(kfs, "ImageWidth").c_str());
  const int height = std::atoi(lookup(kfs, "ImageHeight").c_str());
  const int seed = std::atoi(lookup(kfs, "JitterSeed").c_str());
  const double jscale = std::atof(lookup(kfs, "JitterScale").c_str());
  // parse parameter
  KVV kfr = read(kfrname);
  mpfr53 zoom(lookup(kfr, "Zoom"));
  int bits = std::max(53, int(53 + mpfr_get_exp(zoom.backend().data()) + std::↙
      ↪ ceil(std::log2(double(factor)))));
  int precision = std::ceil(LOG_10_2 * bits);
  mpfr::default_precision(precision);
  mpfr re(lookup(kfr, "Re"));
  mpfr im(lookup(kfr, "Im"));
  int colormethod = std::atoi(lookup(kfr, "ColorMethod").c_str());
  int differences = std::atoi(lookup(kfr, "Differences").c_str());
  int slopes = std::atoi(lookup(kfr, "Slopes").c_str());
#if 0
  double coloroffset = std::atof(lookup(kfr, "ColorOffset").c_str());
#endif
  double iterdiv = std::atof(lookup(kfr, "IterDiv").c_str());
  if (iterdiv == 0.0) iterdiv = 1.0;
  double rotate = std::atof(lookup(kfr, "Rotate").c_str());
  double co = cos(rotate);
  double si = sin(rotate);
  double ratio = std::atof(lookup(kfr, "Ratio").c_str());
  if (ratio != 0 && ratio != 360)
  {
    std::cerr << argv[0] << ": skew ratio must be 360 (got `" << ratio << "')" ↙
        ↪ << std::endl;
    return 1;
  }
  double aspect = double(width) / (double) height;
  // tile
  std::string stem;
  std::string::size_type ix = kfrname.rfind('.');
  if (ix == std::string::npos)
    stem = kfrname;
  else
    stem = kfrname.substr(0, ix);
  bool stratify = (colormethod < 5 || differences == 7) && slopes == 0;
  if (stratify)
  {
```

```
175       std::cerr << "using stratified tiling (best quality)" << std::endl
                    << "after rendering, assemble with Octave/Matlab:" << std::endl
                    << "octave kf-stratify.m " << stem << " " << factor << std::endl;
          std::ostringstream sjscale;
          sjscale << (jscale / factor);
180       update(kfs, "JitterScale", sjscale.str());
          for (int j = 0; j < factor; ++j)
            for (int i = 0; i < factor; ++i)
            {
              double x = ((i + 0.5) / factor - 0.5) * 4 / height;
185           double y = ((j + 0.5) / factor - 0.5) * 4 / height;
              mpfr tre(re + mpfr(mpfr53(  co * x + si * y) / zoom));
              mpfr tim(im + mpfr(mpfr53(- si * x + co * y) / zoom));
              // output
              std::ostringstream sjseed;
190           sjseed << (seed ? seed * factor * factor + j * factor + i : 0);
              std::ostringstream s;
              s << stem << "-" << std::setfill('0') << std::setw(4) << j << "-" << std↙
                  ↳ ::setfill('0') << std::setw(4) << i << ".kfs";
              write(s.str(), update(kfs, "JitterSeed", sjseed.str()));
              std::ostringstream r;
195           r << stem << "-" << std::setfill('0') << std::setw(4) << j << "-" << std↙
                  ↳ ::setfill('0') << std::setw(4) << i << ".kfr";
              write(r.str(), update(update(kfr, "Re", fixed(tre)), "Im", fixed(tim)));
            }
        }
        else
200     {
          std::cerr << "using adjacent tiling (seams may be visible)" << std::endl
                    << (slopes ? "because of: slopes\n" : "")
                    << (colormethod >= 5 && differences != 7 ? "because of: numerical ↙
                        ↳ differencing (try analytic instead)\n" : "")
                    << "after rendering, assemble with ImageMagick:" << std::endl
205                 << "montage " << stem << "-*-*.png -mode Concatenate -tile " << ↙
                        ↳ factor << "x" << factor << " " << stem << ".png" << std::↙
                        ↳ endl;
      #if 0
          // makes tiled image colours more similar to one-shot huge image colours
          // not necessarily desireable...
          if (colormethod == 7)
210       {
            std::ostringstream tcoloroffset;
            tcoloroffset << (coloroffset + log(factor) / iterdiv);
            update(kfr, "ColorOffset", tcoloroffset.str());
          }
215   #endif
          mpfr53 tzoom(zoom * factor);
          update(kfr, "Zoom", scientific(tzoom));
          for (int j = 0; j < factor; ++j)
            for (int i = 0; i < factor; ++i)
220         {
              double x = ((i + 0.5) / factor - 0.5) * 4 * aspect;
              double y = ((j + 0.5) / factor - 0.5) * 4;
              mpfr tre(re + mpfr(mpfr53(  co * x + si * y) / zoom));
              mpfr tim(im + mpfr(mpfr53(- si * x + co * y) / zoom));
225           // output
              std::ostringstream sjseed;
```

```
            sjseed << (seed ? seed * factor * factor + j * factor + i : 0);
            std::ostringstream s;
            s << stem << "-" << std::setfill('0') << std::setw(4) << j << "-" << std↙
                ↳ ::setfill('0') << std::setw(4) << i << ".kfs";
230         write(s.str(), update(kfs, "JitterSeed", sjseed.str()));
            std::ostringstream r;
            r << stem << "-" << std::setfill('0') << std::setw(4) << j << "-" << std↙
                ↳ ::setfill('0') << std::setw(4) << i << ".kfr";
            write(r.str(), update(update(kfr, "Re", fixed(tre)), "Im", fixed(tim)));
          }
235     }
      return 0;
    }
```

120 utils/resizeKFB.m

```
function resizeKFB(fnin,fnout,scale)

meth = 'bicubic';

5 % read kfb file and scale by factor 1/scale

info = dir(fnin);
fid = fopen(fnin,'r');
d = fread(fid, [1, info.bytes], '*uint8');
10 fclose(fid);
fid = fopen(fnout,'w');
%d(1:3) == 'KFB' test skipped
fwrite(fid,d(1:3),'uint8');
k = 4;
15 nBytes = 4;
width = typecast(uint8(d(k:k+nBytes-1)),'uint32');
width2 = width/scale;
width2 = typecast(uint32(width2),'uint8');
fwrite(fid,width2,'uint8');
20
k = k + nBytes;
height = typecast(uint8(d(k:k+nBytes-1)),'uint32');
height2 = height/scale;
height2 = typecast(uint32(height2),'uint8');
25 fwrite(fid,height2,'uint8');

k = k + nBytes;
nCounts = double(width)*double(height); %32 bit ints
nBytes = 4 * nCounts;
30 counts = typecast(uint8(d(k:k+nBytes-1)),'uint32');
counts = reshape(counts,height,width);
counts2 = exp(imresize(log(single(counts)),1/scale,meth));
clear counts;
counts2 = typecast(uint32(counts2(:)),'uint8');
35 fwrite(fid,counts2,'uint8');
clear counts2;

k = k + nBytes;
nBytes = 4;
40 iterdiv = typecast(uint8(d(k:k+nBytes-1)),'uint32');
fwrite(fid,d(k:k+nBytes-1),'uint8');
```

```
    k = k + nBytes;
    parts = typecast(uint8(d(k:k+nBytes-1)),'uint32');
    fwrite(fid,d(k:k+nBytes-1),'uint8');
    parts = double(parts);

    k = k + nBytes;
    nBytes = parts * 3;
    % r,g,b,... parts triples
    keys = typecast(uint8(d(k:k+nBytes-1)),'uint8');
    fwrite(fid,d(k:k+nBytes-1),'uint8');
    keys = double(keys);

    k = k + nBytes;
    nBytes = 4;
    maxiterInt32 = typecast(uint8(d(k:k+nBytes-1)),'uint32');
    fwrite(fid,d(k:k+nBytes-1),'uint8');
    maxiter = double(maxiterInt32);

    k = k + nBytes;
    nBytes = 4 * nCounts;
    trans = typecast(uint8(d(k:k+nBytes-1)),'single');
    trans = reshape(trans,height,width);
    trans2 = exp(imresize(log(trans),1/scale,meth));
    clear trans;
    trans2 = typecast(trans2(:),'uint8');
    fwrite(fid,trans2,'uint8');
    clear trans2;

    % DE data
    k = k + nBytes;
    if(k>info.bytes)
        fprintf('No ADE data found\n');
    else
        nBytes = 4 * nCounts;
        de = typecast(uint8(d(k:k+nBytes-1)),'single');
        de = reshape(de,height,width);
        de2 = exp(imresize(log(de),1/scale,meth));
        clear de;
        de2 = typecast(de2(:),'uint8');
        fwrite(fid,de2,'uint8');
    endif

    fclose(fid);

    endfunction
```

121 utils/stratify.m

```
    % Kalles Fraktaler 2
    % Copyright (C) 2013-2017 Karl Rummo
    % Copyright (C) 2017-2019 Claude Heiland-Allen
    %
    % This program is free software: you can redistribute it and/or modify
    % it under the terms of the GNU Affero General Public License as
    % published by the Free Software Foundation, either version 3 of the
    % License, or (at your option) any later version.
```

```
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU Affero General Public License for more details.
%
% You should have received a copy of the GNU Affero General Public License
% along with this program.  If not, see <https://www.gnu.org/licenses/>.

function stratify(stem, factor)

pattern = sprintf('%s-%s-%s.png', stem, '%04d', '%04d');
output = sprintf('%s.png', stem);

for y=1:factor
  for x=1:factor
    filename = sprintf(pattern, y - 1, x - 1);
    image = imread(filename);
    if (x == 1)
      row = image;
    else
      row = [ row , image ];
    end
  end
  if (y == 1)
    tiled = row;
  else
    tiled = [ tiled ; row ];
  end
end

[height, width, channels] = size(image);
tiled = reshape(tiled, [ height, factor, width, factor, channels ]);
tiled = permute(tiled, [ 2, 1, 4, 3, 5 ]);
tiled = reshape(tiled, [ factor * height, factor * width, channels ]);

imwrite(tiled, output);

endfunction
```