graphgrow

Claude Heiland-Allen

2012–2017

Contents

1 c/graphgrow-filter.c . . . 4
2 c/graphgrow_filter_frag.glsl . . . 6
3 c/graphgrow-filter.pd . . . 8
4 c/graphgrow_filter_vert.glsl . . . 17
5 c/Makefile . . . 17
6 c/s2c.sh . . . 18
7 c/shader.c . . . 18
8 c/shader.h . . . 19
9 extra/mk.hs . . . 20
10 Fractal/GraphGrow/Analysis/Statistics.hs . . . 23
11 Fractal/GraphGrow/Editor/Graph.hs . . . 23
12 Fractal/GraphGrow/Engine/Geometry.hs . . . 28
13 Fractal/GraphGrow/Engine/Graph.hs . . . 30
14 Fractal/GraphGrow/Engine/Grow.hs . . . 32
15 Fractal/GraphGrow/Engine/Zoomer.hs . . . 33
16 Fractal/GraphGrow/GUI/DrawPoints.hs . . . 34
17 Fractal/GraphGrow/GUI/FrameSync.hs . . . 37
18 Fractal/GraphGrow/Main.hs . . . 38
19 Fractal/GraphGrow/Text/Compile.hs . . . 42
20 Fractal/GraphGrow/Text/Glyphs.hs . . . 44
21 Fractal/GraphGrow/Text/Parse.hs . . . 45
22 Fractal/GraphGrow/Utils.hs . . . 45
23 .gitignore . . . 46
24 graphgrow3/audio/.gitignore . . . 46
25 graphgrow3/audio/graphgrow-audio.cc . . . 47
26 graphgrow3/audio/graphgrow-audio-test.cc . . . 59
27 graphgrow3/Audio.hs . . . 60
28 graphgrow3/audio/Makefile . . . 61
29 graphgrow3/audio/README . . . 61
30 graphgrow3/colour.frag . . . 62
31 graphgrow3/colour.vert . . . 62
32 graphgrow3/compress~.pd . . . 62
33 graphgrow3/Dimension.hs . . . 63
34 graphgrow3/edge~.pd . . . 65
35 graphgrow3/graphgrow3.cabal . . . 67
36 graphgrow3/graphgrow3.pd . . . 68
37 graphgrow3/iface/deselected.png . . . 71
38 graphgrow3/iface/.gitignore . . . 72
39 graphgrow3/iface/graphgrow-iface.cc . . . 72
40 graphgrow3/iface/help.png . . . 96
41 graphgrow3/iface/iface.xcf . . . 97
41.1 Layer 0 . . . 97

41.2 Layer 1 . . . 97
41.3 Layer 2 . . . 97
41.4 Layer 3 . . . 97
41.5 Layer 4 . . . 97
41.6 Layer 5 . . . 97
41.7 Layer 6 . . . 98
41.8 Layer 7 . . . 98
41.9 Layer 8 . . . 98
41.10 Layer 9 . . . 98
41.11 Layer 10 . . . 98
41.12 Layer 11 . . . 98
41.13 Layer 12 . . . 99
41.14 Layer 13 . . . 99
41.15 Layer 14 . . . 99
41.16 Layer 15 . . . 99
41.17 Layer 16 . . . 99
41.18 Layer 17 . . . 100
42 graphgrow3/iface/Makefile . . . 100
43 graphgrow3/iface/README . . . 100
44 graphgrow3/iface/selected.png . . . 101
45 graphgrow3/initial.frag . . . 101
46 graphgrow3/initial.vert . . . 101
47 graphgrow3/LICENSE . . . 102
48 graphgrow3/Main.hs . . . 114
49 graphgrow3/pitchshift~.pd . . . 120
50 graphgrow3/README.md . . . 121
51 graphgrow3/rule~.pd . . . 122
52 graphgrow3/Setup.hs . . . 123
53 graphgrow3/step.frag . . . 123
54 graphgrow3/step.vert . . . 124
55 graphgrow3/Types.hs . . . 125
56 graphgrow3/video/.gitignore . . . 125
57 graphgrow3/video/graphgrow-video.cc . . . 125
58 graphgrow3/video/graphgrow-video-test.cc . . . 138
59 graphgrow3/Video.hs . . . 140
60 graphgrow3/video/Makefile . . . 144
61 graphgrow3/video/README . . . 145
62 graphgrow.cabal . . . 145
63 LICENSE . . . 146
64 README . . . 158
65 Setup.hs . . . 158
66 the-sky-cracked-open/ChangeLog.md . . . 158
67 the-sky-cracked-open/colour.frag . . . 158
68 the-sky-cracked-open/initial.frag . . . 159
69 the-sky-cracked-open/LICENSE . . . 159
70 the-sky-cracked-open/main.hs . . . 171
71 the-sky-cracked-open/Setup.hs . . . 176
72 the-sky-cracked-open/Shader.hs . . . 176
73 the-sky-cracked-open/Snapshot.hs . . . 176
74 the-sky-cracked-open/step.frag . . . 177
75 the-sky-cracked-open/the-sky-cracked-open.cabal . . . 178

1 c/graphgrow-filter.c

```
#include <complex.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#include <sndfile.h>
#include <GL/glew.h>
#include <GLFW/glfw3.h>

#include "graphgrow_filter_vert.glsl.c"
#include "graphgrow_filter_frag.glsl.c"
#include "shader.h"

struct filter {
  struct shader shader;
  struct { GLint u_first_pass, u_spectrum, u_bins, u_length, u_speed, u_gain, u_scaleA, u_scaleB; } uniform;
  struct { GLint u_first_pass, u_spectrum; GLfloat u_bins, u_length, u_speed, u_gain, u_scaleA, u_scaleB; } value;
  GLuint vao;
  GLuint tex[2];
  GLuint fbo[2];
  int which;
  complex float spectrum[4096];
};

void filter_init(struct filter *f) {
  shader_init(&f->shader, graphgrow_filter_vert, graphgrow_filter_frag);
  glUseProgram(f->shader.program);
  shader_uniform(f, u_first_pass);
  shader_uniform(f, u_spectrum);
  shader_uniform(f, u_bins);
  shader_uniform(f, u_length);
  shader_uniform(f, u_speed);
  shader_uniform(f, u_gain);
  shader_uniform(f, u_scaleA);
  shader_uniform(f, u_scaleB);
  f->value.u_bins = 4096;
  f->value.u_length = 1;
  f->value.u_speed = 100;
  f->value.u_gain = 0.99;
  shader_updatef(f, u_bins);
  shader_updatef(f, u_length);
  shader_updatef(f, u_speed);
  shader_updatef(f, u_gain);
  glGenVertexArrays(1, &f->vao);
  glBindVertexArray(f->vao);
  glGenTextures(2, f->tex);
  glActiveTexture(GL_TEXTURE0 + 0);
  glBindTexture(GL_TEXTURE_1D, f->tex[0]);
  glTexImage1D(GL_TEXTURE_1D, 0, GL_RGB32F, f->value.u_bins / 2, 0, GL_RGB, GL_FLOAT, 0);
  glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
  glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

```
  glActiveTexture(GL_TEXTURE0 + 1);
  glBindTexture(GL_TEXTURE_1D, f->tex[1]);
  glTexImage1D(GL_TEXTURE_1D, 0, GL_RGB32F, f->value.u_bins / 2, 0, GL_RGB, ↙
      ↳ GL_FLOAT, 0);
  glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)↙
      ↳ ;
  glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glGenFramebuffers(2, f->fbo);
  GLenum bufs = GL_COLOR_ATTACHMENT0;
  glBindFramebuffer(GL_FRAMEBUFFER, f->fbo[0]);
  glFramebufferTexture1D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_1D, f↙
      ↳ ->tex[0], 0);
  glDrawBuffers(1, &bufs);
  glBindFramebuffer(GL_FRAMEBUFFER, f->fbo[1]);
  glFramebufferTexture1D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_1D, f↙
      ↳ ->tex[1], 0);
  glDrawBuffers(1, &bufs);
  glViewport(0, 0, f->value.u_bins / 2, 1);
}

void filter_kernel(struct filter *f, double t) {
  f->value.u_scaleA = 1.0/4.0 + t * (1.0/2.0 - 1.0/4.0);
  f->value.u_scaleB = sqrt(2) * (1.0/2.0 - f->value.u_scaleA);
  shader_updatef(f, u_scaleA);
  shader_updatef(f, u_scaleB);
  glClearColor(0, 0, 0, 0);
  glBindFramebuffer(GL_FRAMEBUFFER, f->fbo[0]);
  glClear(GL_COLOR_BUFFER_BIT);
  glBindFramebuffer(GL_FRAMEBUFFER, f->fbo[1]);
  glClear(GL_COLOR_BUFFER_BIT);
  int which = 0;
  for (int pass = 0; pass < 20; ++pass) {
    f->value.u_first_pass = pass == 0;
    shader_updatei(f, u_first_pass);
    f->value.u_spectrum = which;
    shader_updatei(f, u_spectrum);
    glActiveTexture(GL_TEXTURE0 + which);
    glGenerateMipmap(GL_TEXTURE_2D);
    glBindFramebuffer(GL_FRAMEBUFFER, f->fbo[1 - which]);
    glDrawArrays(GL_LINES, 0, 2);
    which = 1 - which;
  }
  glReadPixels(0, 0, f->value.u_bins, 1, GL_RG, GL_FLOAT, f->spectrum);
  f->spectrum[2048] = 0;
  for (int k = 1; k < 2048; ++k) {
    f->spectrum[4096 - k] = conjf(f->spectrum[k]);
  }
}

const double pi = 3.141592653589793;

int main(int argc, char **argv) {
  (void) argc;
  (void) argv;

  glfwInit();
  glfwWindowHint(GLFW_CLIENT_API, GLFW_OPENGL_API);
```

```
105   glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
      glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
      glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
      glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
      GLFWwindow *window = glfwCreateWindow(512, 16, "graphgrow-filter", 0, 0);
110   glfwMakeContextCurrent(window);
      glewExperimental = GL_TRUE;
      glewInit();
      glGetError(); // discard common error from glew

115   struct filter filter;
      filter_init(&filter);

      SF_INFO sf_info = { 0, 48000, 2, SF_FORMAT_WAV | SF_FORMAT_FLOAT, 0, 0 };
      SNDFILE *sf_file = sf_open("graphgrow-filter.wav", SFM_WRITE, &sf_info);
120
      for (int frame = 0; frame < 750; ++frame) {
        filter_kernel(&filter, 0.5*(1 - cos(pi * frame/750.0)));
        sf_writef_float(sf_file, (float *) &filter.spectrum[0], filter.value.u_bins)↙
            ↳ ;
      }
125
      sf_close(sf_file);
      return 0;
    }
```

2 c/graphgrow_filter_frag.glsl

```
    #version 330 core

    const float pi = 3.141592653589793;
    const float twopi = 2.0 * pi;
  5 const float SR = 48000.0;

    uniform bool u_first_pass;
    uniform sampler1D u_spectrum;
    uniform float u_bins;
 10 uniform float u_length;
    uniform float u_speed;
    uniform float u_gain;
    uniform float u_scaleA;
    uniform float u_scaleB;
 15 uniform float u_offsetA;
    uniform float u_offsetB;

    in float i_bin;

 20 out vec3 o_response;

    vec2 cis(float t) {
      return vec2(cos(t), sin(t));
    }
 25
    vec2 cmul(vec2 a, vec2 b) {
      return vec2(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x);
    }
```

```
vec2 cpow(vec2 a, float b) {
  return pow(length(a), b) * cis(atan(a.y, a.x) * b);
}

void main() {
  float rms = sqrt(texelFetch(u_spectrum, 0, 11).z);
  float freq = SR * i_bin / u_bins;
  float phase = freq * u_length / u_speed;
  if (u_first_pass) {
    vec2 OUTPUT = pow(u_gain, u_length) * cis(twopi * phase);
    o_response = vec3(OUTPUT, dot(OUTPUT, OUTPUT));
    return;
  }
  vec2 u = pow(u_gain, u_scaleA * u_length) * cis(twopi * u_scaleA * phase);
  vec2 v = pow(u_gain, u_scaleB * u_length) * cis(twopi * u_scaleB * phase);
  {
    float bin2 = i_bin * u_scaleA;
    vec2 u2 = cpow(texture(u_spectrum, bin2/u_bins).xy/rms, u_scaleA);
    if (! ((u2.x == 0 && u2.y == 0) || isnan(u2.x) || isnan(u2.y) || isinf(u2.x)
        ↳ || isinf(u2.y))) { u = u2; }
  }
  {
    float bin2 = i_bin * u_scaleB;
    vec2 v2 = cpow(texture(u_spectrum, bin2/u_bins).xy/rms, u_scaleB);
    if (! ((v2.x == 0 && v2.y == 0) || isnan(v2.x) || isnan(v2.y) || isinf(v2.x)
        ↳ || isinf(v2.y))) { v = v2; }
  }
  vec2 INPUT = vec2(1.0, 0.0);
  // edge losses
  vec2 a = u, b = v, c = v, d = v, e = v, f = u, g = u, h = u;
  // node values
  vec2 z = vec2(0.0, 0.0);
  vec2 A = z, B = z, C = z, D = z, E = z, F = z, G = z, H = z;
  // fixed point iteration
  for (int k = 0; k < 512; ++k) {
    // scale by degree
    A /= 1.0;
    B /= 3.0;
    C /= 3.0;
    D /= 3.0;
    E /= 3.0;
    F /= 1.0;
    G /= 1.0;
    H /= 1.0;
    // accumulate neighbours
    vec2 An = cmul(a, B) + INPUT;
    vec2 Bn = cmul(a, A) + cmul(b, C) + cmul(e, E);
    vec2 Cn = cmul(b, B) + cmul(c, D) + cmul(f, F);
    vec2 Dn = cmul(c, C) + cmul(d, E) + cmul(h, H);
    vec2 En = cmul(e, B) + cmul(d, D) + cmul(g, G);
    vec2 Fn = cmul(f, C);
    vec2 Gn = cmul(g, E);
    vec2 Hn = cmul(h, D);
    // step
    A = An;
    B = Bn;
    C = Cn;
```

```
      D = Dn;
      E = En;
      F = Fn;
      G = Gn;
      H = Hn;
    }
    // output
    vec2 OUTPUT = H;
    o_response = vec3(OUTPUT, dot(OUTPUT, OUTPUT));
}
```

3 c/graphgrow-filter.pd

```
#N canvas 1984 133 1920 1022 10;
#X obj 101 100 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 101 122 openpanel;
#X obj 100 168 soundfiler;
#N canvas 1055 276 454 660 fft 0;
#X obj 19 17 block~ 4096 4 1;
#X obj 28 50 inlet~;
#X obj 28 72 *~;
#X obj 31 334 rfft~;
#X obj 32 368 *~;
#X obj 32 396 -~;
#X obj 62 372 *~;
#X obj 95 371 *~;
#X obj 127 371 *~;
#X obj 40 305 tabreceive~ window;
#X obj 95 399 +~;
#X obj 31 465 rifft~;
#X obj 31 487 *~;
#X obj 31 509 outlet~;
#X obj 343 23 inlet;
#X obj 343 45 / 4096;
#X obj 343 67 int;
#X obj 234 188 * 4096;
#X obj 65 171 tabreceive~ ramp;
#X floatatom 374 102 5 0 0 0 - - -, f 5;
#X obj 97 430 /~ 1;
#X obj 30 422 /~ 1;
#X obj 226 339 +~;
#X obj 185 337 +~;
#X obj 313 155 int;
#X obj 288 131 t f f;
#X obj 300 198 -;
#X obj 268 261 t b f;
#X obj 268 283 1;
#X obj 268 305 -;
#X obj 185 301 *~ 0;
#X obj 223 300 *~ 0;
#X obj 330 308 *~ 0;
#X obj 368 307 *~ 0;
#X obj 382 201 * 4096;
#X obj 249 49 inlet~;
#X obj 249 71 snapshot~;
#X obj 199 50 bang~;
```

```
45  #X obj 249 93 *;
    #X obj 316 42 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
    -1;
    #X obj 378 124 + 1;
    #X obj 378 146 int;
50  #X obj 381 174 min 0;
    #X obj 419 100 - 1;
    #X obj 234 163 min 0;
    #X obj 174 213 tabread4~ re;
    #X obj 198 235 tabread4~ im;
55  #X obj 340 226 tabread4~ re;
    #X obj 350 246 tabread4~ im;
    #X connect 1 0 2 0;
    #X connect 2 0 3 0;
    #X connect 3 0 4 0;
60  #X connect 3 0 7 0;
    #X connect 3 1 6 0;
    #X connect 3 1 8 0;
    #X connect 4 0 5 0;
    #X connect 5 0 21 0;
65  #X connect 6 0 5 1;
    #X connect 7 0 10 0;
    #X connect 8 0 10 1;
    #X connect 9 0 2 1;
    #X connect 9 0 12 1;
70  #X connect 10 0 20 0;
    #X connect 11 0 12 0;
    #X connect 12 0 13 0;
    #X connect 14 0 15 0;
    #X connect 15 0 16 0;
75  #X connect 16 0 19 0;
    #X connect 16 0 38 1;
    #X connect 16 0 43 0;
    #X connect 17 0 45 1;
    #X connect 17 0 46 1;
80  #X connect 18 0 45 0;
    #X connect 18 0 46 0;
    #X connect 18 0 47 0;
    #X connect 18 0 48 0;
    #X connect 20 0 11 1;
85  #X connect 21 0 11 0;
    #X connect 22 0 7 1;
    #X connect 22 0 6 1;
    #X connect 23 0 8 1;
    #X connect 23 0 4 1;
90  #X connect 24 0 26 1;
    #X connect 24 0 44 0;
    #X connect 25 0 26 0;
    #X connect 25 1 24 0;
    #X connect 26 0 27 0;
95  #X connect 26 0 32 1;
    #X connect 26 0 33 1;
    #X connect 27 0 28 0;
    #X connect 27 1 29 1;
    #X connect 28 0 29 0;
100 #X connect 29 0 31 1;
    #X connect 29 0 30 1;
```

```
#X connect 30 0 23 0;
#X connect 31 0 22 0;
#X connect 32 0 23 1;
#X connect 33 0 22 1;
#X connect 34 0 47 1;
#X connect 34 0 48 1;
#X connect 35 0 36 0;
#X connect 36 0 38 0;
#X connect 37 0 36 0;
#X connect 38 0 25 0;
#X connect 38 0 40 0;
#X connect 39 0 16 0;
#X connect 40 0 41 0;
#X connect 41 0 42 0;
#X connect 42 0 34 0;
#X connect 43 0 42 1;
#X connect 43 0 44 1;
#X connect 44 0 17 0;
#X connect 45 0 30 0;
#X connect 46 0 31 0;
#X connect 47 0 32 0;
#X connect 48 0 33 0;
#X restore 98 232 pd fft;
#X obj 365 136 table re;
#X obj 118 350 dac~;
#X obj 267 186 table window 4096;
#X obj 267 217 loadbang;
#X msg 267 239 4096;
#X obj 267 261 until;
#X obj 267 283 f 0;
#X obj 313 284 + 1;
#X obj 313 306 mod 4096;
#X obj 269 331 expr 0.5-0.5*cos(2.0*3.141592653*$f1/4096);
#X obj 269 309 t f f;
#X obj 271 392 tabwrite window;
#X obj 96 255 *~ 0;
#X floatatom 174 227 5 0 0 0 - - -, f 5;
#X floatatom 152 398 5 0 0 0 - - -, f 5;
#X obj 172 248 / 200000;
#X obj 405 189 table ramp 4096;
#X obj 281 365 tabwrite ramp;
#X obj 365 158 table im;
#X msg 216 319 0;
#X obj 1014 41 bang~;
#X obj 1014 63 snapshot~;
#X obj 167 281 env~ 16384;
#X obj 196 377 t f f;
#X floatatom 196 399 5 0 0 0 - - -, f 5;
#X obj 197 357 max;
#X msg 697 68 0 \, destroy;
#X obj 649 164 gemhead;
#X msg 670 327 snap;
#X obj 670 300 t b a;
#X obj 639 491 separator;
#X obj 1009 492 separator;
#X obj 819 494 separator;
#X obj 737 391 scale 4;
```

```
#X obj 753 331 square 4;
#X obj 655 71 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X obj 670 254 t a a;
#X obj 639 600 scale;
#X obj 819 600 scale;
#X obj 1009 600 scale;
#X obj 1242 494 separator;
#X obj 1241 582 scale;
#X obj 639 663 square 1;
#X obj 825 664 square 1;
#X obj 1013 668 square 1;
#X obj 1238 658 square 1;
#X floatatom 1014 85 5 0 0 0 - - -, f 5;
#X obj 1016 108 expr 1.0/4.0+0.5*(1-cos(3.141592653*$f1))*(1.0/2.0-1.0/4.0)
;
#X obj 1012 160 expr sqrt(2.0)*(1.0/2.0-$f1);
#X floatatom 1058 133 5 0 0 0 - - -, f 5;
#X floatatom 1045 183 5 0 0 0 - - -, f 5;
#X obj 1103 131 s scaleA;
#X obj 1160 572 r scaleA;
#X obj 1006 234 s scaleB;
#X obj 1257 204 -;
#X obj 1257 160 t b f;
#X obj 1258 251 s offsetA;
#X obj 639 551 translateXYZ -0.5 0 0;
#X obj 1121 506 r offsetA;
#X obj 902 521 * -1;
#X obj 820 552 translateXYZ 0 -0.5 0;
#X obj 1011 550 translateXYZ -0.5 0 0;
#X obj 1244 552 translateXYZ 0 0.5 0;
#X obj 1257 182 1;
#X obj 1013 134 t f f;
#X obj 735 461 t a a a a a;
#X obj 631 699 separator;
#X obj 1001 700 separator;
#X obj 811 702 separator;
#X obj 634 799 scale;
#X obj 814 799 scale;
#X obj 1004 798 scale;
#X obj 1234 702 separator;
#X obj 1237 801 scale;
#X obj 631 871 square 1;
#X obj 817 872 square 1;
#X obj 1005 876 square 1;
#X obj 1230 866 square 1;
#X obj 631 759 translateXYZ -0.5 0 0;
#X obj 907 717 * -1;
#X obj 812 760 translateXYZ 0 -0.5 0;
#X obj 1003 758 translateXYZ -0.5 0 0;
#X obj 1236 760 translateXYZ 0 0.5 0;
#X obj 733 672 t a a a a a;
#X obj 1149 711 r offsetB;
#X obj 1008 187 t f f;
#X obj 1091 249 s offsetB;
#X obj 1152 780 r scaleB;
#X obj 749 284 separator;
```

```
#X obj 737 414 rotateXYZ 0 0 45;
#X obj 639 622 rotateXYZ 0 0 45;
#X obj 820 620 rotateXYZ 0 0 45;
#X obj 1010 630 rotateXYZ 0 0 45;
#X obj 1239 602 rotateXYZ 0 0 45;
#X obj 1091 227 expr $f1;
#X obj 1257 226 expr $f1*sqrt(2.0);
#X obj 737 372 pix_texture;
#X msg 629 388 quality 2;
#X obj 630 363 loadbang;
#X obj 752 309 colorRGB 1 1 1;
#X obj 15 112 *~ 2;
#X obj 15 134 -~ 1;
#X obj 15 156 abs~;
#X obj 642 119 gemwin 25;
#X obj 13 208 noise~;
#N canvas 163 366 454 660 fft 0;
#X obj 19 17 block~ 4096 4 1;
#X obj 28 50 inlet~;
#X obj 28 72 *~;
#X obj 31 334 rfft~;
#X obj 32 368 *~;
#X obj 32 396 -~;
#X obj 62 372 *~;
#X obj 95 371 *~;
#X obj 127 371 *~;
#X obj 40 305 tabreceive~ window;
#X obj 95 399 +~;
#X obj 31 465 rifft~;
#X obj 31 487 *~;
#X obj 31 509 outlet~;
#X obj 343 23 inlet;
#X obj 343 45 / 4096;
#X obj 343 67 int;
#X obj 234 188 * 4096;
#X obj 65 171 tabreceive~ ramp;
#X floatatom 374 102 5 0 0 0 - - -, f 5;
#X obj 97 430 /~ 1;
#X obj 30 422 /~ 1;
#X obj 226 339 +~;
#X obj 185 337 +~;
#X obj 313 155 int;
#X obj 288 131 t f f;
#X obj 300 198 -;
#X obj 268 261 t b f;
#X obj 268 283 1;
#X obj 268 305 -;
#X obj 185 301 *~ 0;
#X obj 223 300 *~ 0;
#X obj 330 308 *~ 0;
#X obj 368 307 *~ 0;
#X obj 382 201 * 4096;
#X obj 249 49 inlet~;
#X obj 249 71 snapshot~;
#X obj 199 50 bang~;
#X obj 249 93 *;
#X obj 316 42 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
```

```
-1;
#X obj 378 124 + 1;
#X obj 378 146 int;
#X obj 381 174 min 0;
#X obj 419 100 - 1;
#X obj 234 163 min 0;
#X obj 174 213 tabread4~ re;
#X obj 198 235 tabread4~ im;
#X obj 340 226 tabread4~ re;
#X obj 350 246 tabread4~ im;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 3 0 7 0;
#X connect 3 1 6 0;
#X connect 3 1 8 0;
#X connect 4 0 5 0;
#X connect 5 0 21 0;
#X connect 6 0 5 1;
#X connect 7 0 10 0;
#X connect 8 0 10 1;
#X connect 9 0 2 1;
#X connect 9 0 12 1;
#X connect 10 0 20 0;
#X connect 11 0 12 0;
#X connect 12 0 13 0;
#X connect 14 0 15 0;
#X connect 15 0 16 0;
#X connect 16 0 19 0;
#X connect 16 0 38 1;
#X connect 16 0 43 0;
#X connect 17 0 45 1;
#X connect 17 0 46 1;
#X connect 18 0 45 0;
#X connect 18 0 46 0;
#X connect 18 0 47 0;
#X connect 18 0 48 0;
#X connect 20 0 11 1;
#X connect 21 0 11 0;
#X connect 22 0 7 1;
#X connect 22 0 6 1;
#X connect 23 0 8 1;
#X connect 23 0 4 1;
#X connect 24 0 26 1;
#X connect 24 0 44 0;
#X connect 25 0 26 0;
#X connect 25 1 24 0;
#X connect 26 0 27 0;
#X connect 26 0 32 1;
#X connect 26 0 33 1;
#X connect 27 0 28 0;
#X connect 27 1 29 1;
#X connect 28 0 29 0;
#X connect 29 0 31 1;
#X connect 29 0 30 1;
#X connect 30 0 23 0;
#X connect 31 0 22 0;
```

```
#X connect 32 0 23 1;
#X connect 33 0 22 1;
#X connect 34 0 47 1;
#X connect 34 0 48 1;
#X connect 35 0 36 0;
#X connect 36 0 38 0;
#X connect 37 0 36 0;
#X connect 38 0 25 0;
#X connect 38 0 40 0;
#X connect 39 0 16 0;
#X connect 40 0 41 0;
#X connect 41 0 42 0;
#X connect 42 0 34 0;
#X connect 43 0 42 1;
#X connect 43 0 44 1;
#X connect 44 0 17 0;
#X connect 45 0 30 0;
#X connect 46 0 31 0;
#X connect 47 0 32 0;
#X connect 48 0 33 0;
#X restore 99 288 pd fft;
#X obj 97 311 *~ 0;
#X obj 14 264 noise~;
#X msg 176 207 20;
#X obj 734 437 colorRGB 0.7 0.7 0.7;
#X obj 649 193 t a a;
#X obj 434 625 loadbang;
#X msg 434 654 file out 0 \, auto 1;
#X obj 407 590 spigot;
#X obj 3 34 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 13 90 vline~;
#X obj 61 475 writesf~ 2;
#X msg 8 412 stop;
#X msg 57 428 open -bytes 4 out.wav \, start;
#X msg 453 539 1;
#X msg 441 563 0;
#X obj 186 185 loadbang;
#X msg 101 144 read -maxsize 1e+09 -resize \$1 re im;
#X msg 13 66 0 \, 1 600000, f 13;
#X obj 8 390 delay 600000;
#X msg 642 44 dimen 512 512 \, create \, 1;
#X obj 737 350 pix_snap2tex 0 0 512 512;
#X obj 415 692 pix_write 0 0 512 512;
#X connect 0 0 1 0;
#X connect 1 0 126 0;
#X connect 2 0 3 2;
#X connect 2 0 109 2;
#X connect 3 0 16 0;
#X connect 7 0 8 0;
#X connect 8 0 9 0;
#X connect 9 0 10 0;
#X connect 10 0 11 0;
#X connect 10 0 14 0;
#X connect 11 0 12 0;
#X connect 12 0 10 1;
#X connect 13 0 15 0;
```

```
#X connect 14 0 13 0;
#X connect 14 0 21 0;
#X connect 14 1 15 1;
#X connect 14 1 21 1;
#X connect 16 0 26 0;
#X connect 16 0 5 0;
#X connect 16 0 120 0;
#X connect 17 0 19 0;
#X connect 19 0 16 1;
#X connect 19 0 110 1;
#X connect 23 0 29 1;
#X connect 24 0 25 0;
#X connect 25 0 50 0;
#X connect 26 0 18 0;
#X connect 26 0 29 0;
#X connect 27 0 28 0;
#X connect 27 1 29 1;
#X connect 29 0 27 0;
#X connect 30 0 107 0;
#X connect 31 0 114 0;
#X connect 32 0 130 0;
#X connect 33 0 32 0;
#X connect 33 1 130 0;
#X connect 34 0 61 0;
#X connect 35 0 65 0;
#X connect 36 0 64 0;
#X connect 37 0 93 0;
#X connect 39 0 107 0;
#X connect 40 0 33 0;
#X connect 40 1 92 0;
#X connect 41 0 94 0;
#X connect 42 0 95 0;
#X connect 43 0 96 0;
#X connect 44 0 66 0;
#X connect 45 0 97 0;
#X connect 50 0 51 0;
#X connect 51 0 53 0;
#X connect 51 0 55 0;
#X connect 51 0 68 0;
#X connect 52 0 54 0;
#X connect 52 0 89 0;
#X connect 56 0 43 1;
#X connect 56 0 42 1;
#X connect 56 0 41 1;
#X connect 56 0 45 1;
#X connect 58 0 99 0;
#X connect 59 0 67 0;
#X connect 59 1 58 1;
#X connect 61 0 41 0;
#X connect 62 0 63 0;
#X connect 62 0 65 1;
#X connect 62 0 66 2;
#X connect 63 0 64 2;
#X connect 63 0 61 1;
#X connect 64 0 42 0;
#X connect 65 0 43 0;
#X connect 66 0 45 0;
```

```
#X connect 67 0 58 0;
#X connect 68 0 52 0;
#X connect 68 1 59 0;
#X connect 69 0 34 0;
#X connect 69 1 36 0;
#X connect 69 2 35 0;
#X connect 69 3 44 0;
#X connect 69 4 87 0;
#X connect 70 0 82 0;
#X connect 71 0 85 0;
#X connect 72 0 84 0;
#X connect 73 0 78 0;
#X connect 74 0 79 0;
#X connect 75 0 80 0;
#X connect 76 0 86 0;
#X connect 77 0 81 0;
#X connect 82 0 73 0;
#X connect 83 0 84 2;
#X connect 83 0 82 1;
#X connect 83 0 85 2;
#X connect 83 0 84 1;
#X connect 84 0 74 0;
#X connect 85 0 75 0;
#X connect 86 0 77 0;
#X connect 87 0 70 0;
#X connect 87 1 72 0;
#X connect 87 2 71 0;
#X connect 87 3 76 0;
#X connect 88 0 83 0;
#X connect 88 0 85 1;
#X connect 88 0 86 2;
#X connect 88 0 86 1;
#X connect 88 0 82 2;
#X connect 89 0 57 0;
#X connect 89 1 98 0;
#X connect 91 0 75 1;
#X connect 91 0 74 1;
#X connect 91 0 73 1;
#X connect 91 0 77 1;
#X connect 92 0 103 0;
#X connect 93 0 113 0;
#X connect 94 0 46 0;
#X connect 95 0 47 0;
#X connect 96 0 48 0;
#X connect 97 0 49 0;
#X connect 98 0 90 0;
#X connect 99 0 60 0;
#X connect 100 0 37 0;
#X connect 101 0 100 0;
#X connect 102 0 101 0;
#X connect 103 0 38 0;
#X connect 104 0 105 0;
#X connect 105 0 106 0;
#X connect 106 0 3 1;
#X connect 106 0 25 0;
#X connect 106 0 109 1;
#X connect 108 0 3 0;
```

```
#X connect 109 0 110 0;
#X connect 110 0 5 1;
#X connect 110 0 120 1;
#X connect 111 0 109 0;
#X connect 112 0 17 0;
#X connect 113 0 69 0;
#X connect 114 0 117 0;
#X connect 114 1 40 0;
#X connect 115 0 116 0;
#X connect 116 0 131 0;
#X connect 117 0 131 0;
#X connect 118 0 127 0;
#X connect 118 0 122 0;
#X connect 118 0 123 0;
#X connect 118 0 128 0;
#X connect 119 0 104 0;
#X connect 121 0 120 0;
#X connect 122 0 120 0;
#X connect 123 0 117 1;
#X connect 124 0 117 1;
#X connect 125 0 112 0;
#X connect 126 0 2 0;
#X connect 127 0 119 0;
#X connect 128 0 121 0;
#X connect 128 0 124 0;
#X connect 129 0 107 0;
#X connect 130 0 100 0;
#X connect 130 1 100 1;
```

4 c/graphgrow_filter_vert.glsl

```
#version 330 core

uniform float u_bins;

out float i_bin;

void main() {
  switch (gl_VertexID) {
    default:
    case 0:
      gl_Position = vec4(-1.0, 0.0, 0.0, 1.0);
      i_bin = 0.0;
      break;
    case 1:
      gl_Position = vec4( 1.0, 0.0, 0.0, 1.0);
      i_bin = u_bins / 2.0;
      break;
  }
}
```

5 c/Makefile

```
graphgrow-filter: graphgrow-filter.c graphgrow_filter_frag.glsl.c ↙
    ↳ graphgrow_filter_vert.glsl.c shader.c shader.h
        gcc -std=c99 -Wall -Wextra -pedantic -O3 -fopenmp -o graphgrow-filter ↙
            ↳ graphgrow-filter.c shader.c -lm -lsndfile -lGL -lGLEW -lglfw
```

```
graphgrow_filter_frag.glsl.c: graphgrow_filter_frag.glsl s2c.sh
	./s2c.sh graphgrow_filter_frag < graphgrow_filter_frag.glsl > ↙
	    ↳ graphgrow_filter_frag.glsl.c

graphgrow_filter_vert.glsl.c: graphgrow_filter_vert.glsl s2c.sh
	./s2c.sh graphgrow_filter_vert < graphgrow_filter_vert.glsl > ↙
	    ↳ graphgrow_filter_vert.glsl.c

.SUFFIXES:
```

6 c/s2c.sh

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

7 c/shader.c

```
/* ============================================================================
pool-party -- water simulation
Copyright (C) 2016  Claude Heiland-Allen <claude@mathr.co.uk>
----------------------------------------------------------------------------
based on:
rdex -- reaction-diffusion explorer
Copyright (C) 2008  Claude Heiland-Allen <claude@mathr.co.uk>
----------------------------------------------------------------------------
Generic Shader
============================================================================ */

#include <stdio.h>
#include <stdlib.h>
#include "shader.h"

//============================================================================
// print a shader object's debug log
void shader_debug(GLuint obj) {
//  return; // FIXME: only dump logs when shader compile/link failed
  int infologLength = 0;
  int maxLength;
  if (glIsShader(obj)) {
    glGetShaderiv(obj, GL_INFO_LOG_LENGTH, &maxLength);
  } else {
    glGetProgramiv(obj, GL_INFO_LOG_LENGTH, &maxLength);
  }
  char *infoLog = malloc(maxLength);
  if (!infoLog) {
    return;
  }
  if (glIsShader(obj)) {
    glGetShaderInfoLog(obj, maxLength, &infologLength, infoLog);
  } else {
```

```
    glGetProgramInfoLog(obj, maxLength, &infologLength, infoLog);
  }
  if (infologLength > 0) {
    fprintf(stderr, "%s\n", infoLog);
  }
  free(infoLog);
}

//============================================================
// generic shader initialization
struct shader *shader_init(
  struct shader *shader, const char *vert, const char *frag
) {
  if (! shader) { return 0; }
  shader->linkStatus     = 0;
  shader->vertexSource   = vert;
  shader->fragmentSource = frag;
  if (shader->vertexSource || shader->fragmentSource) {
    shader->program = glCreateProgram();
    if (shader->vertexSource) {
      shader->vertex =
        glCreateShader(GL_VERTEX_SHADER);
      glShaderSource(shader->vertex,
        1, &shader->vertexSource, 0
      );
      glCompileShader(shader->vertex);
      shader_debug(shader->vertex);
      glAttachShader(shader->program, shader->vertex);
    }
    if (shader->fragmentSource) {
      shader->fragment =
        glCreateShader(GL_FRAGMENT_SHADER);
      glShaderSource(shader->fragment,
        1, &shader->fragmentSource, 0
      );
      glCompileShader(shader->fragment);
      shader_debug(shader->fragment);
      glAttachShader(shader->program, shader->fragment);
    }
    glLinkProgram(shader->program);
    shader_debug(shader->program);
    glGetProgramiv(shader->program,
      GL_LINK_STATUS, &shader->linkStatus
    );
    if (! shader->linkStatus) { return 0; }
  } else { return 0; }
  return shader;
}

// EOF
```

8 c/shader.h

```
/* ============================================================
pool-party -- water simulation
Copyright (C) 2016  Claude Heiland-Allen <claude@mathr.co.uk>
------------------------------------------------------------
```

```
based on:
rdex -- reaction-diffusion explorer
Copyright (C) 2008  Claude Heiland-Allen <claude@mathr.co.uk>
-------------------------------------------------------------------------
Generic Shader
=========================================================================*/

#ifndef SHADER_H
#define SHADER_H 1

#include <GL/glew.h>

//=========================================================================
// generic shader data
struct shader {
  GLint linkStatus;
  GLuint program;
  GLuint fragment;
  GLuint vertex;
  const char *fragmentSource;
  const char *vertexSource;
};

//=========================================================================
// generic shader uniform location access macro
#define shader_uniform(self,name) \
  (self)->uniform.name = \
    glGetUniformLocation((self)->shader.program, #name)

//=========================================================================
// generic shader uniform update access macro (integer)
#define shader_updatei(self,name) \
  glUniform1i((self)->uniform.name, (self)->value.name)

//=========================================================================
// generic shader uniform update access macro (integer vector)
#define shader_update2i(self,name) \
  glUniform2iv((self)->uniform.name, 1, (self)->value.name)

//=========================================================================
// generic shader uniform update access macro (float)
#define shader_updatef(self,name) \
  glUniform1f((self)->uniform.name, (self)->value.name)

//=========================================================================
// generic shader initialization
struct shader *shader_init(
  struct shader *shader, const char *vert, const char *frag
);

#endif
```

9 extra/mk.hs

```
import Control.Monad (forM_)
import Control.Monad.ST (ST, runST)
import Data.STRef
```

```
import Data.Array.Unboxed as A
import Data.Array.ST (STUArray, newArray, readArray, writeArray)
import Data.Array.Unsafe (unsafeFreeze)

import Data.ByteString as Strict
import Data.ByteString.Lazy as Lazy

import Data.Char as C hiding (isAlpha, toUpper)
import Data.List as L
import Data.Map as M
import Data.Ord as O
import Data.Set as S
import Data.Word as W

import System.Environment (getArgs)

equating f a b = f a == f b

chain :: Ord a => Set a -> [a] -> UArray (Int, Int) Word64
chain wordSet input@(hinput:tinput) = runST $ do
  let n = S.size wordSet
      bs = ((0,0),((n-1),(n-1)))
      ix w = S.size (fst (w `S.split` wordSet))
      inc x | x == maxBound = maxBound
            | otherwise = x + 1
  arr <- newArray bs 0 :: ST s (STUArray s (Int, Int) Word64)
  context <- newSTRef (ix hinput)
  forM_ (tinput ++ [hinput]) $ \word -> do
    i <- readSTRef context
    let j = ix word
    writeArray arr (i, j) . inc =<< readArray arr (i, j)
    writeSTRef context j
  unsafeFreeze arr

prettyChain :: Int -> Set Strict.ByteString -> UArray (Int, Int) Word64 -> Lazy.↙
  ↳ ByteString
prettyChain maxLinks wordSet chain = Lazy.fromChunks . L.concat $
  [ L.intersperse spc ([context, col] ++ (L.map fst . L.take maxLinks . L.sortBy↙
    ↳ (flip (comparing snd))) [ (word, count) | (word, j) <- S.toList wordSet↙
    ↳ `L.zip` [i+0-i..], let count = chain A.! (i, j), count > 0 ]) ++ [eol] ↙
    ↳ | (context, i) <- S.toList wordSet `L.zip` [0..] ]
  where
    eol = Strict.pack [fromIntegral $ ord '\n']
    spc = Strict.pack [fromIntegral $ ord ' ']
    col = Strict.pack [fromIntegral $ ord ':']

histogram :: Ord a => [a] -> Map a Int
histogram = L.foldl' inc M.empty
  where
    inc m w = case M.lookup w m of
      Nothing -> M.insert w 1 m
      Just k -> let k' = k + 1 in k' `seq` M.insert w k' m

commonest :: Ord a => Map a Int -> [a]
commonest = L.map fst . L.sortBy (flip (comparing snd)) . M.toList
```

```
toWords :: String -> Lazy.ByteString -> [Strict.ByteString]
toWords tag = L.concatMap (lineWords tag) . L.map toStrict . llines

lineWords :: String -> Strict.ByteString -> [Strict.ByteString]
lineWords "irc" = \l -> case Strict.uncons (Strict.drop 9 l) of
  Just (c, l)
    | c == fromIntegral (ord '-') -> []
    | otherwise -> (L.drop 1 . L.filter isWord . Strict.groupBy (equating ↙
      ↳ isAlpha) . Strict.map toUpper) l
  Nothing -> []
lineWords "mail" = L.drop 1 . L.filter isWord . Strict.groupBy (equating ↙
  ↳ isAlpha) . Strict.map toUpper

isWord :: Strict.ByteString -> Bool
isWord s = 4 <= Strict.length s && Strict.all isAlpha s

isAlpha :: Word8 -> Bool
isAlpha = \c -> a <= c && c <= z
  where
    a = fromIntegral (ord 'A')
    z = fromIntegral (ord 'Z')

toUpper :: Word8 -> Word8
toUpper = \c -> if a <= c && c <= z then c - a + bigA else c
  where
    a = fromIntegral (ord 'a')
    z = fromIntegral (ord 'z')
    bigA = fromIntegral (ord 'A')

toStrict :: Lazy.ByteString -> Strict.ByteString
toStrict = Strict.concat . Lazy.toChunks

llines :: Lazy.ByteString -> [Lazy.ByteString]
llines s = if Lazy.null s then [] else l : llines (Lazy.drop 1 ls)
  where
    (l, ls) = Lazy.span neol s

neol = \c -> c /= eol
  where
    eol = fromIntegral (ord '\n')

slines :: Strict.ByteString -> [Strict.ByteString]
slines s = if Strict.null s then [] else l : slines (Strict.drop 1 ls)
  where
    (l, ls) = Strict.span neol s

nl = Strict.pack [fromIntegral $ ord '\n']

main :: IO ()
main = do
  args <- getArgs
  case args of
    [tag, "words", wordCount] -> Lazy.interact (Lazy.fromChunks . L.concatMap (\↙
      ↳ x->[x,nl]) . L.take (read wordCount) . commonest . histogram . toWords↙
      ↳  tag)
    [tag, "chain", wordFile] -> do
      wordList <- (S.fromList . slines) `fmap` Strict.readFile wordFile
```

```
110         Lazy.interact (prettyChain 32 wordList . chain wordList . L.filter (`S.member` wordList) . toWords tag)
```

10 Fractal/GraphGrow/Analysis/Statistics.hs

```
module Fractal.GraphGrow.Analysis.Statistics where

import Data.List (foldl')
import Data.Monoid (Monoid(..))

data Stat s = Stat{ s0, s1, s2 :: !s }
  deriving (Eq, Ord, Read, Show)

instance Num s => Monoid (Stat s) where
  mempty = Stat{ s0 = 0, s1 = 0, s2 = 0 }
  a `mappend` b = Stat{ s0 = s0 a + s0 b, s1 = s1 a + s1 b, s2 = s2 a + s2 b }
  mconcat = foldl' mappend mempty

stat :: Num s => s -> Stat s
stat s = Stat{ s0 = 1, s1 = s, s2 = s * s }

getCount :: Stat s -> s
getCount = s0

getSum :: Stat s -> s
getSum = s1

getMean :: Fractional s => Stat s -> s
getMean s = s1 s / s0 s

getVariance :: Fractional s => Stat s -> s
getVariance s = (s0 s * s2 s - s1 s * s1 s) / (s0 s * s0 s)

getStdDev :: Floating s => Stat s -> s
getStdDev = sqrt . getVariance

type StatOrd s = (Stat s, MinMax s)

statOrd :: Num s => s -> StatOrd s
statOrd s = (stat s, minMax s)

newtype MinMax s = MinMax{ getMinMax :: Maybe (s, s) }
  deriving (Eq, Ord, Read, Show)

minMax :: s -> MinMax s
minMax s = MinMax $! Just $! (((,) $! s) $! s)

instance Ord s => Monoid (MinMax s) where
  mempty = MinMax Nothing
  MinMax Nothing `mappend` b = b
  a `mappend` MinMax Nothing = a
  MinMax (Just (ain, aax)) `mappend` MinMax (Just (bin, bax)) = MinMax $! Just $! (((,) $! ain `min` bin) $! aax `max` bax)
  mconcat = foldl' mappend mempty
```

11 Fractal/GraphGrow/Editor/Graph.hs

```
{-# LANGUAGE MultiParamTypeClasses, TypeFamilies #-}
module Fractal.GraphGrow.Editor.Graph (GraphEditor, newGraphEditor) where

import Graphics.UI.Gtk.Toy.Prelude hiding (D, bg)

import Data.Colour.RGBSpace (uncurryRGB)
import Data.Colour.RGBSpace.HSV (hsv)
import Data.Colour.SRGB (sRGB, toSRGBBounded)
import Data.Fixed (mod')

import Data.Default (Default(def))
import Data.Label (get)
import qualified Data.Map as M
import Data.Maybe (maybeToList)

import qualified Graphics.UI.Gtk as G
import Control.Monad (forM_)
import Data.IORef (IORef, newIORef, readIORef, writeIORef)

import Fractal.GraphGrow.Engine.Geometry (Transform, scale, rotate, translate)
import Fractal.GraphGrow.Engine.Graph (Node(Node))

type GraphEditor = Toy Graph

newGraphEditor :: Node -> ([(Node, Transform)] -> IO ()) -> IO GraphEditor
newGraphEditor node putUpdates = newToy (graph node putUpdates)

setGraphEditorNodes :: GraphEditor -> [Node] -> IO ()
setGraphEditorNodes tb nodes = do
  g <- snd `fmap` readIORef (toyState tb)


phi :: Double
phi = (sqrt 5 + 1) / 2

nodeColour (Node n) = uncurryRGB sRGB $ hsv (360 * ((phi * fromIntegral n) `mod'` 1)) 1 1

type D = Draggable Cairo CairoDiagram -- every thing needs be draggable

-- knob and link ids
type KID = Int
type LID = (KID, KID)

-- things
data Thing = Background {-nextKnobID-}KID D | Knob {-isFixedPosition-}Bool KID D D | Link LID D D Node

-- visual representation
bg n = mkDraggable (r2 (300, 300)) $ rect 600 600 # fc (blend (1/16) (nodeColour n) white)
knob n fixed p =
    ( mkDraggable (r2 p) $ shape  5 # lc black # fc col
    , mkDraggable (r2 p) $ shape 12 # fcA (col `withOpacity` 0.5)
    )
  where
    col = nodeColour n
```

```
55      shape = if fixed then square . ((pi/2) *) else circle
    link n p@(x0,y0) q@(x1,y1) =
      ( mkDraggable (r2((x0+x1)/2,(y0+y1)/2)) $ circle 5 # lc black # fc col
      , mkDraggable (r2(0,0)) $ p2 p ~~ p2 q # stroke # lc col
      )
60    where
        col = nodeColour n

    -- extract diagrams
    tDiagram (Background _ d)   = [d]
65  tDiagram (Knob _ _ d1 d2)   = [d1, d2]  -- d1 is inner core, d2 is outer ring
    tDiagram (Link _   d1 d2 _) = [d1, d2]  -- d1 is inner core, d2 is line

    -- what did a mouse event do
    data Act
70    = Create KID
      | Delete KID
      | MoveStart KID
      | Moving    KID
      | MoveStop  KID
75    | LinkStart        KID  (Double, Double)
      | Linking              (Double, Double)
      | LinkStop (Maybe (KID, (Double, Double)))

    -- what mode are we in
80  data Mode
      = MMove KID                                        -- moving a knob
      | MLink KID (Double, Double) (Double, Double) -- creating a link

    data Graph = Graph
85    { gNode      :: Node
      , gGetNodes :: IO [Node]
      , gPutEdges :: [(Node, Transform)] -> IO ()
      , gMode      :: Maybe Mode
      , gThings    :: [Thing]
90    }

    -- update links when knobs are moved or deleted
    resort g@Graph{ gThings = ts } = g{ gThings = ts' }
      where
95      knobs = [ n | n@(Knob      {}) <- ts ]
        links = [ l | l@(Link      {}) <- ts ]
        backg = [ b | b@(Background{}) <- ts ]
        positions = M.fromList [ (i, unr2 $ get dragOffset d) | Knob _ i d _ <- ↙
            ↳ knobs ]
        links' =
100       [ Link ij d1 d2 n
          | Link ij@(i, j) _ _ n <- links
          , (d1, d2) <- maybeToList $
              liftA2 (link n) (M.lookup i positions) (M.lookup j positions)
          ]
105     ts' = knobs ++ links' ++ backg

    -- mouse handling
    -- linking knobs
    tMouse _ (Just(MLink j q _))(Just(False, 0)) p t@(Background _ _    ) ↙
        ↳                                    = (Just $ LinkStop Nothing      , [t↙
```

```
    ↳ ]) -- note: Background is the last Thing in the list
tMouse _ (Just(MLink j q _))(Just(False, 0)) p t@(Knob fixed i d1 d2) | j /= i ↙
    ↳ && clickInside d2 (p2 p)    = (Just $ LinkStop (Just (i, p)), [t])
tMouse _ (Just(MLink j q _))(Just(False, 0)) p t ↙
    ↳                                                    = (Nothing ↙
    ↳                        , [t])
tMouse _ (Just(MLink j q _))_                p t ↙
    ↳                                                    = (Just $ ↙
    ↳ Linking     p          , [t])
-- moving knobs
tMouse _ Nothing          m@(Just (True,  0)) p t@(Knob fixed i d1 d2) | ↙
    ↳ clickInside d1 (p2 p) && not fixed = (Just $ MoveStart i   , [Knob False i ↙
    ↳ (mouseDrag m p d1) (mouseDrag m p d2)])
                                                                         | ↙
                                                                           ↳ clickInside↙
                                                                           ↳  d2 ↙
                                                                           ↳ (p2 ↙
                                                                           ↳ p) ↙
                                                                           ↳                ↙
                                                                           ↳ = (↙
                                                                           ↳ Just↙
                                                                           ↳  $ ↙
                                                                           ↳ LinkStart↙
                                                                           ↳  i p↙
                                                                           ↳  ↙
                                                                           ↳          ↙
                                                                           ↳ , [t↙
                                                                           ↳ ])
tMouse _ (Just(MMove j)) m@(Just (False, 0)) p   (Knob False i d1 d2) | j == i ↙
    ↳                          = (Just $ MoveStop  i   , [Knob False i (↙
    ↳ mouseDrag m p d1) (mouseDrag m p d2)])
tMouse _ (Just(MMove j)) m                   p   (Knob False i d1 d2) | j == i ↙
    ↳                          = (Just $ Moving    i   , [Knob False i (↙
    ↳ mouseDrag m p d1) (mouseDrag m p d2)])
-- creating knobs
tMouse c Nothing          (Just (True,  0)) p   (Background i d) ↙
    ↳                                    = (Just $ Create    i   , [uncurry↙
    ↳  (Knob False i) (knob c False p), Background (i + 1) d])
tMouse _ Nothing          (Just (True,  1)) p   (Knob False i d1 d2) | ↙
    ↳ clickInside d1 (p2 p)          = (Just $ Delete    i   , [])
tMouse _ _currentMode     _mouseEvent     _ t@_thingToTest ↙
    ↳                                      = (Nothing           , [t])

-- concatMap until a thing signals something was done, then just copy the rest
tMouses _ _ _ _ a@(Just _) ws = (a, ws)
tMouses _ _ _ _ Nothing [] = (Nothing, [])
tMouses c d m p Nothing (w:ws) =
  let (a, ws') = tMouse c d m p w
  in  (ws' ++) `fmap` tMouses c d m p a ws

type instance V Graph = R2

graph node getNodes putUpdates = Graph node getNodes putUpdates Nothing
  [ uncurry (Knob True 1) (knob node True (500, 300))
  , uncurry (Knob True 0) (knob node True (100, 300))
  , Background 2 (bg node)
  ]
```

```
instance Interactive Graph where
  mouse = simpleMouse $ \m p g@(Graph{ gNode = c, gMode = d, gThings = ts}) -> ↙
      ↳ resort $ case (d, tMouses c d m p Nothing ts) of
    -- moving knobs
    (Nothing           , (Just (MoveStart i           ), ts'))          -> g{ ↙
        ↳ gThings = ts', gMode = Just (MMove i) }
    (Just (MMove j)    , (Just (Moving    i           ), ts')) | j == i -> g{ ↙
        ↳ gThings = ts', gMode = Just (MMove i) }
    (Just (MMove j)    , (Just (MoveStop  i           ), ts')) | j == i -> g{ ↙
        ↳ gThings = ts', gMode = Nothing }
    -- linking knobs
    (Nothing,            (Just (LinkStart i p         ), ts'))          -> g{ ↙
        ↳ gThings = ts', gMode = Just (MLink i p p) }
    (Just (MLink j q _), (Just (Linking     p         ), ts'))          -> g{ ↙
        ↳ gThings = ts', gMode = Just (MLink j q p) }
    (Just (MLink j q _), (Just (LinkStop (Just (i, p))), ts')) | j /= i -> g{ ↙
        ↳ gThings = uncurry (Link (j, i)) (link c q p) c: ts', gMode = Nothing }
    (Just (MLink j q _), (Just (LinkStop Nothing      ), ts'))          -> g{ ↙
        ↳ gThings = ts', gMode = Nothing }
    -- everything else
    (_, (_, ts')) -> g{ gThings = ts', gMode = Nothing }

instance Diagrammable Cairo Graph where
  diagram (Graph{ gMode = m, gThings = ts}) = mconcat . map diagram $ ↙
      ↳ modeDiagram ++ concatMap tDiagram ts
    where
      modeDiagram = case m of
        Just (MLink _ p q) -> [mkDraggable (r2(0,0)) $ p2 p ~~ p2 q # stroke # ↙
            ↳ lc black]
        _ -> []

instance GtkInteractive Graph where
  display = displayDiagram diagram

main :: IO ()
main = do
  G.initGUI
  graphsRef <- newIORef M.empty
  window <- G.windowNew
  G.windowSetDefaultSize window 600 600
  box <- G.hButtonBoxNew

  button <- G.buttonNewWithLabel "+"
  button `G.on` G.buttonActivated $ do
    graphs <- readIORef graphsRef
    let n = maybe (Node 0) ((\(Node m) -> Node (m+1)) . fst . fst) (M.↙
        ↳ maxViewWithKey graphs)
        gcol = uncurryRGB G.Color $ toSRGBBounded (nodeColour n)
    g <- newGraphEditor n (M.keys `fmap` readIORef graphsRef) (\_ -> return ())
    w <- G.windowNew
    G.windowSetGeometryHints w (Nothing `asTypeOf` Just w) (Just (600, 600)) (↙
        ↳ Just (600, 600)) Nothing Nothing Nothing
    G.set w $ [G.containerChild G.:= toyWindow g]
    writeIORef graphsRef (M.insert n (w, g) graphs)
    b <- G.buttonNew
    b `G.on` G.buttonActivated $ G.windowPresent{-WithTime-} w
```

```
      forM_ [G.StateNormal, G.StateActive, G.StatePrelight, G.StateSelected, G.↙
          ↳ StateInsensitive] $ \s ->
        G.widgetModifyBg b s gcol
      G.containerAdd box b
      G.widgetShowAll window
      G.widgetShowAll w

  G.containerAdd box button
  G.containerAdd window box
  G.widgetShowAll window
  G.mainGUI
```

12 Fractal/GraphGrow/Engine/Geometry.hs

```
module Fractal.GraphGrow.Engine.Geometry
  ( -- * Affine transforms.
    Transform(Transform)
  , identity
  , o
  , compose
  , scale
  , rotate
  , translate
    -- * Vectors.
  , Point(Point)
  , scalePoint
  , addPoint
  , distanceSquared
    -- * Inspecting transforms.
  , center
  , sizeSquared
    -- * Bounding boxes.
  , Box(Box)
  , makeBox
  , scaleBox
  , boxSize
  , clampBox
  , outsideBox
  , ) where

import Data.List (foldl')
import Foreign (Storable(..), castPtr)

import Fractal.GraphGrow.Utils (sqr, clamp)

data Transform = Transform !Double !Double !Double !Double !Double !Double
  deriving (Read, Show)
instance Storable Transform where
  alignment _ = alignment (0 :: Double)
  sizeOf _ = 6 * sizeOf (0 :: Double)
  poke p (Transform a b c d e f) = do
    let q = castPtr p
    pokeElemOff q 0 a
    pokeElemOff q 1 b
    pokeElemOff q 2 c
    pokeElemOff q 3 d
    pokeElemOff q 4 e
```

```
      pokeElemOff q 5 f
    peek p = do
      let q = castPtr p
      a <- peekElemOff q 0
      b <- peekElemOff q 1
      c <- peekElemOff q 2
      d <- peekElemOff q 3
      e <- peekElemOff q 4
      f <- peekElemOff q 5
      return (Transform a b c d e f)

identity :: Transform
identity = Transform 1 0 0 1 0 0

o :: Transform -> Transform -> Transform
Transform a b c d e f `o` Transform u v w x y z = Transform
  (a * u + b * w) (a * v + b * x)
  (c * u + d * w) (c * v + d * x)
  (a * y + b * z + e) (c * y + d * z + f)

compose :: [Transform] -> Transform
compose = foldl' o identity

scale :: Double -> Transform
scale factor = Transform factor 0 0 factor 0 0

rotate :: Double -> Transform
rotate turns = Transform c (-s) s c 0 0
  where
    a = 2 * pi * turns
    c = cos a
    s = sin a

translate :: Double -> Double -> Transform
translate x y = Transform 1 0 0 1 x y

data Point = Point !Double !Double
  deriving (Read, Show)

scalePoint :: Double -> Point -> Point
scalePoint factor (Point x y) = Point (factor * x) (factor * y)

addPoint :: Point -> Point -> Point
addPoint (Point x y) (Point u v) = Point (x + u) (y + v)

distanceSquared :: Point -> Point -> Double
distanceSquared (Point x y) (Point u v) = sqr (x - u) + sqr (y - v)

center :: Transform -> Point
center (Transform _ _ _ _ x y) = Point x y

sizeSquared :: Transform -> Double
sizeSquared (Transform a b _ _ _ _) = sqr a + sqr b

data Box = Box !Double !Double !Double !Double
  deriving (Read, Show)

```

```
makeBox :: Double -> Double -> Box
makeBox width height = Box (-width) (-height) width height

scaleBox :: Double -> Box -> Box
scaleBox factor (Box lx ly hx hy) = Box (cx - dx) (cy - dy) (cx + dx) (cy + dy)
  where
    cx = (hx + lx) / 2
    cy = (hy + ly) / 2
    dx = (hx - lx) / 2 * factor
    dy = (hy - ly) / 2 * factor

boxSize :: Box -> Double
boxSize (Box lx ly hx hy) = sqrt (sqr (hx - lx) + sqr (hy - ly))

clampBox :: Box -> Point -> Point
clampBox (Box lx ly hx hy) (Point x y) = Point (clamp x lx hx) (clamp y ly hy)

outsideBox :: Box -> Point -> Double -> Bool
outsideBox (Box lx ly hx hy) (Point ox oy) radiusSquared =
  let r = sqrt radiusSquared
  in  ox + r < lx || hx < ox - r || oy + r < ly || hy < oy - r
```

13 Fractal/GraphGrow/Engine/Graph.hs

```
{-# LANGUAGE GeneralizedNewtypeDeriving #-}
module Fractal.GraphGrow.Engine.Graph where

import Data.Ix (Ix(..))
import Data.Map (Map)
import qualified Data.Map as M
import Data.Maybe (catMaybes, fromMaybe)
import Data.Vector.Storable (Vector)
import qualified Data.Vector.Storable as V
import Data.Word (Word32)
import Foreign.Storable (Storable(..))

import Fractal.GraphGrow.Engine.Geometry
import Fractal.GraphGrow.Utils (SThree(..))

-- Word32 is probably enough, as each edge needs ~56 bytes...
newtype Node = Node Word32 deriving (Read, Show, Ix, Eq, Ord, Storable)
newtype Edge = Edge Word32 deriving (Read, Show, Ix, Eq, Ord, Storable)

class Graph g where
  edgesFrom :: g -> Node -> [SThree Edge Node Transform]
  edgeTransform :: g -> Edge -> Transform

data MGraph = MGraph
  { mGraph :: Map Node [(Node, Transform)]
  }

annotate :: MGraph -> MMGraph
annotate g = MMGraph
    { mNodes = M.fromListWith (++) [ (source, [edge]) | (edge, (source, _)) <- edges ]
    , mEdges = M.fromList [ (edge, tt) | (edge, (_, tt)) <- edges ]
    }
```

```
  where
    edges = map Edge [0..] `zip` [ (source, (target, transform)) | (source, tts)↙
        ↳ <- M.toAscList (mGraph g), (target, transform) <- tts ]

data MMGraph = MMGraph
  { mNodes :: Map Node [Edge]
  , mEdges :: Map Edge (Node, Transform)
  }

instance Graph MMGraph where
  edgesFrom g n = catMaybes
    [ (\(m,t) -> SThree e m t) `fmap` M.lookup e (mEdges g)
    | e <- fromMaybe [] (M.lookup n (mNodes g))
    ]
  edgeTransform g e = snd $ mEdges g M.! e

vectorize :: MMGraph -> Maybe VGraph
vectorize g
  | dense = Just VGraph
      { vNodeOffset = V.fromList . map (Offset . fromIntegral) . init . scanl ↙
          ↳ (+) 0 . map (length . snd) . M.toAscList . mNodes $ g
      , vNodeEdges  = V.fromList . concatMap snd . M.toAscList . mNodes $ g
      , vEdgeTargets = V.fromList . map (fst . snd) . M.toAscList . mEdges $ g
      , vEdgeTransforms = V.fromList . map (snd . snd) . M.toAscList . mEdges $ ↙
          ↳ g
      }
  | otherwise = Nothing
  where
    dense = fmap (fst . fst) (M.minViewWithKey (mNodes g)) == Just (Node 0)
        && fmap (fst . fst) (M.maxViewWithKey (mNodes g)) == Just (Node (↙
            ↳ fromIntegral (M.size (mNodes g) - 1)))
        && fmap (fst . fst) (M.minViewWithKey (mEdges g)) == Just (Edge 0)
        && fmap (fst . fst) (M.maxViewWithKey (mEdges g)) == Just (Edge (↙
            ↳ fromIntegral (M.size (mEdges g) - 1)))

newtype Offset = Offset Word32 deriving (Read, Show, Ix, Eq, Ord, Storable)

data VGraph = VGraph
  { vNodeOffset     :: Vector Offset     -- indexed by Node
  , vNodeEdges      :: Vector Edge       -- indexed by Offset
  , vEdgeTargets    :: Vector Node       -- indexed by Edge
  , vEdgeTransforms :: Vector Transform  -- indexed by Edge
  }

instance Graph VGraph where
  edgesFrom g (Node n) = fromMaybe [] $ do
    Offset start <- vNodeOffset g V.!? fromIntegral n
    let Offset end = fromMaybe (Offset . fromIntegral . V.length . vNodeEdges $ ↙
        ↳ g) $ vNodeOffset g V.!? (fromIntegral n + 1)
        go off =
          let edge@(Edge e) = vNodeEdges g V.! off
              e' = fromIntegral e
              target = vEdgeTargets g V.! e'
              transform = vEdgeTransforms g V.! e'
          in  SThree edge target transform
    return $ map go [fromIntegral start .. fromIntegral end - 1]
  edgeTransform g (Edge e) = vEdgeTransforms g V.! fromIntegral e
```

14 Fractal/GraphGrow/Engine/Grow.hs

```
module Fractal.GraphGrow.Engine.Grow where

import Data.List (partition)

import Fractal.GraphGrow.Engine.Geometry
import Fractal.GraphGrow.Engine.Graph
import Fractal.GraphGrow.Utils (sqr, SThree(..))

data Context = Context
  { ctxNode :: !Node
  , ctxPath :: [Edge]
  , ctxZoom :: !Double
  , ctxTransform :: !Transform
  }
  deriving (Read, Show)

type ContextPlus = SThree Context Point Double

plus :: Context -> ContextPlus
plus ctx = SThree ctx (center (ctxTransform ctx)) (sizeSquared (ctxTransform ctx↙
    ↳ ))

unplus :: ContextPlus -> Context
unplus (SThree ctx _ _) = ctx

renormalize :: Graph g => g -> Box -> Context -> [Context]
renormalize graph box ctx
  | ctxZoom ctx > 1e16 = []
  | otherwise = case map unplus $ grow graph box (16 * boxSize box) False ctx of
      ls@(_:_) ->
        [ let g = compose . map (edgeTransform graph) . reverse $ ctxPath c
          in  Context
                { ctxNode = ctxNode c
                , ctxPath = []
                , ctxZoom = ctxZoom c * sqrt (sizeSquared g)
                , ctxTransform = ctxTransform ctx `o` g
                }
        | c <- ls ]
      _ -> [ctx]

visible :: Box -> ContextPlus -> Bool
visible box (SThree _ c s) = not (outsideBox box c s)

grow :: Graph g => g -> Box -> Double -> Bool -> Context -> [ContextPlus]
grow graph box threshold keepAll = \ctx -> go [plus ctx]
  where
    thresholdSq = sqr threshold
    small (SThree _ _ sizeSq) = sizeSq < thresholdSq
    next = concatMap (map plus . split graph . unplus)
    go | keepAll = goKeepAll
       | otherwise = goDiscard
    goDiscard [] = []
    goDiscard ss =
      let (smalls, bigs) = (partition small . filter (visible box)) ss
      in  smalls ++ goDiscard (next bigs)
```

```
      goKeepAll [] = []
      goKeepAll ss =
        let bigs = (filter (visible box) . filter (not . small)) ss
        in  ss ++ goKeepAll (next bigs)

split :: Graph g => g -> Context -> [Context]
split g ctx =
  [ Context
      { ctxNode = m
      , ctxPath = e : ctxPath ctx
      , ctxZoom = ctxZoom ctx
      , ctxTransform = ctxTransform ctx `o` s
      }
  | SThree e m s <- edgesFrom g (ctxNode ctx)
  ]

zoom :: Transform -> Double -> Context -> Context
zoom z w ctx =
  ctx{ ctxTransform = z `o` ctxTransform ctx, ctxZoom = w * ctxZoom ctx }
```

15 Fractal/GraphGrow/Engine/Zoomer.hs

```
module Fractal.GraphGrow.Engine.Zoomer
  ( ZoomerOptions(..)
  , defaultZoomerOptions
  , Zoomer
  , zoomer
  , Zoomed(..)
  ) where

import Control.Parallel (par)
import Data.List (minimumBy)
import Data.Ord (comparing)
import System.Random (RandomGen, randomR)

import Fractal.GraphGrow.Engine.Geometry
import Fractal.GraphGrow.Engine.Graph
import Fractal.GraphGrow.Engine.Grow
import Fractal.GraphGrow.Utils

type Zoomer = Double -> Zoomed
data Zoomed = Zoomed{ zoomedPoints :: [STwo Point Double], zoomedZoomer :: ↙
    ↳ Zoomer }

data ZoomerOptions g = ZoomerOptions
  { zoomerOuterBox    :: Box
  , zoomerInnerBox    :: Box
  , zoomerPerturb     :: Double
  , zoomerReCenter    :: Double
  , zoomerZoomFactor  :: Double
  , zoomerGraph       :: g
  , zoomerSeed        :: [Context]
  , zoomerCenter      :: Point
  }

zoomer :: (RandomGen r, Graph g) => ZoomerOptions g -> r -> Zoomer
zoomer (ZoomerOptions a b c d e f g h) = zoomer' a b c d e f g h
```

```
35    where
        zoomer' box cbox perturb recenter zoomFactor graph = go
          where
            go seed centerP rg0 threshold =
              let points = concatMap (map deplus . grow graph box threshold True) ↙
                  ↳ seed
40                deplus (SThree _ p s) = STwo p s
                  (dx, rg1) = randomR (-1, 1) rg0
                  (dy, rg2) = randomR (-1, 1) rg1
                  dp = Point dx dy
                  target = clampBox cbox . addPoint centerP . scalePoint perturb $ ↙
                      ↳ dp
45                central = comparing (distanceSquared target)
                  minimumBy' _ [] = Point 0 0
                  minimumBy' cmp xs = minimumBy cmp xs
                  small = map fstTwo . filter ((<= threshold) . sndTwo)
                  centerP'@(Point x y) = minimumBy' central (small points)
50                z = translate (recenter * x) (recenter * y) `o`
                      scale zoomFactor `o`
                      translate (negate x) (negate y)
                  seed' = concatMap (renormalize graph box . zoom z zoomFactor) seed
                  next = go seed' centerP' rg2
55            in  centerP' `par` Zoomed points next

    defaultZoomerOptions :: g -> Node -> ZoomerOptions g
    defaultZoomerOptions graph node = ZoomerOptions
      { zoomerOuterBox    = box
60    , zoomerInnerBox    = scaleBox 0.5 box
      , zoomerPerturb     = 0.002 * w
      , zoomerReCenter    = 0.995
      , zoomerZoomFactor = 10 ** (1 / 60)
      , zoomerGraph       = graph
65    , zoomerSeed        = [Context node [] 1 identity]
      , zoomerCenter      = Point 0 0
      }
      where
        box = Box (-w) (-h) w h
70      w = aspect * h
        h = 16
        aspect = 16/9
```

16 Fractal/GraphGrow/GUI/DrawPoints.hs

```
    module Fractal.GraphGrow.GUI.DrawPoints
      ( withPointDraw
      , flattenPoints
      ) where
5
    import Control.Monad (forM_)
    import Foreign (Ptr, allocaArray, castPtr, pokeElemOff, with, withArray, peek, ↙
        ↳ nullPtr, sizeOf)
    import Foreign.C (withCString, peekCStringLen)
    import Foreign.ForeignPtr.Safe (ForeignPtr, mallocForeignPtrBytes, ↙
        ↳ withForeignPtr)
10  import Graphics.GL
    import Unsafe.Coerce (unsafeCoerce)
```

```
import Fractal.GraphGrow.Engine.Geometry (Point(..), Box(..))
import Fractal.GraphGrow.Utils (STwo(..), sum')

floatToGLfloat :: Float -> GLfloat
floatToGLfloat = unsafeCoerce

flatten :: [STwo Point Double] -> [GLfloat]
flatten fs = map (floatToGLfloat . realToFrac) $ concat [[x, y, z, 2 * pi * (t + 0.5) / 4] | STwo (Point x y) z <- fs, t <- [0,1,2,3]]

flattenPoints :: [STwo Point Double] -> IO [(ForeignPtr GLfloat, Int)]
flattenPoints [] = return []
flattenPoints ps = do
  let (now, later) = splitAt (chunkBytes `div` (16 * sizeOf (0::GLfloat))) ps
  fptr <- mallocForeignPtrBytes chunkBytes
  withForeignPtr fptr $ \ptr -> do
    count <- pokeArrayCount ptr (flatten now)
    rest <- flattenPoints later
    return $ (fptr, count `div` 16) : rest

chunkBytes :: Int
chunkBytes = 16 * 1024 * 4 * 4 * 4

withPointDraw :: Box -> (String -> IO a) -> (([(ForeignPtr GLfloat, Int)] -> IO Int) -> IO a) -> IO a
withPointDraw (Box lx ly hx hy) cbFail cbOk = do
  withProgram (Just circleVertSource) Nothing (Just circleFragSource) cbFail $ \circle -> do
    glUseProgram circle
    mvp' <- withCString "mvp" $ glGetUniformLocation circle . castPtr
    withArray [ realToFrac $ 2 / (hx - lx), 0, 0, realToFrac $ 2 / (hy - ly) ] $ glUniformMatrix2fv mvp' 1 (fromIntegral GL_FALSE)
    glUseProgram 0
    cbOk $ \ps -> if null ps then return 0 else do
      glClearColor 0 0 0 1
      glClear (fromIntegral GL_COLOR_BUFFER_BIT)
      glEnable GL_BLEND
      glBlendFunc GL_SRC_ALPHA GL_ONE
      glUseProgram circle
      glEnableVertexAttribArray 0
      forM_ ps $ \(fptr, count) -> withForeignPtr fptr $ \ptr -> do
        glVertexAttribPointer 0 4 GL_FLOAT (fromIntegral GL_FALSE) 0 ptr
        glDrawArrays GL_QUADS 0 (4 * fromIntegral count)
      glDisableVertexAttribArray 0
      glUseProgram 0
      return (sum' $ map snd ps)

pokeArrayCount :: Ptr GLfloat -> [GLfloat] -> IO Int
pokeArrayCount ptr vals0 = go vals0 0
  where
    go [] n       = return n
    go (val:vals) n = do pokeElemOff ptr n val; go vals (n + 1)

withProgram :: Maybe String -> Maybe String -> Maybe String -> (String -> IO a) -> (GLuint -> IO a) -> IO a
withProgram mv mg mf cbFail cbOk = do
  prog <- glCreateProgram
```

```
65    let compile _ Nothing = return ()
          compile t (Just src) = withCString src $ \s -> with s $ \p -> do
            sh <- glCreateShader t
            glShaderSource sh 1 (castPtr p) nullPtr
            glCompileShader sh
70          glAttachShader prog sh
            glDeleteShader sh
      compile GL_VERTEX_SHADER   mv
      compile GL_GEOMETRY_SHADER mg
      compile GL_FRAGMENT_SHADER mf
75    glLinkProgram prog
      result <- with 0 $ \p -> glGetProgramiv prog GL_LINK_STATUS p >> peek p
      logLen <- with 0 $ \p -> glGetProgramiv prog GL_INFO_LOG_LENGTH p >> peek p
      logTxt <- if logLen <= 1 then return "" else allocaArray (1 + fromIntegral ↙
          ↳ logLen) $ \p -> do
        glGetProgramInfoLog prog (fromIntegral logLen) nullPtr p
80      peekCStringLen (castPtr p, fromIntegral logLen - 1)
      r <- if result == fromIntegral GL_TRUE then cbOk prog else cbFail logTxt
      --glDeleteProgram prog
      return r

85  circleVertSource :: String
    circleVertSource = unlines
      [ "#version 330"
      , "uniform mat2 mvp;"
      , "layout (location = 0) in vec4 position;"
90    , "smooth out vec3 texCoord;"
      , "void main() {"
      , "  const float minSize = 1.0;"
      , "  const float maxSize = 512.0;"
      , "  const float brightness = 1.0 / 256.0;"
95    , "  const float gamma = 1.0;"
      , "  float scaling = length(mvp * vec2(1.0)) / length(vec2(1.0));"
      , "  float q = sqrt(1.0/3.0) * sqrt(position.z) / scaling;"
      , "  vec2 r = mvp * vec2(q);"
      , "  float size = length(r) / scaling;"
100   , "  float alpha = pow(brightness, pow(size / maxSize, gamma));"
      , "  float s = max(minSize * scaling, length(r));"
      , "  vec2 delta = sqrt(2.0) * vec2(cos(position.w), sin(position.w));"
      , "  gl_Position = vec4(mvp * (position.xy + s * delta), 0.0, 1.0);"
      , "  texCoord = vec3(delta, alpha);"
105   , "}"
      ]

    circleFragSource :: String
    circleFragSource = unlines
110   [ "#version 330"
      , "in vec3 texCoord;"
      , "out vec4 colour;"
      , "void main() {"
      , "  const float lineWidth = 1.0;"
115   , "  const vec3 blue   = vec3(0.0, 0.0, 1.0);"
      , "  const vec3 orange = vec3(1.0, 0.7, 0.0);"
      , "  vec2 p = texCoord.xy;"
      , "  vec2 dp = dFdx(p);"
      , "  float r = length(p);"
120   , "  float t = texCoord.z;"
```

```
    , "  if (r >= 1.0) {"
    , "    discard;"
    , "  } else if (r >= 1.0 - lineWidth * length(dp)) {"
    , "    colour = vec4(vec3(mix(0.0, 1.0, t)), t);"
125 , "  } else {"
    , "    discard;"
    , "  }"
    , "}"
    ]
```

17 Fractal/GraphGrow/GUI/FrameSync.hs

```
module Fractal.GraphGrow.GUI.FrameSync
  ( FrameSyncOptions(..)
  , defaultFrameSyncOptions
  , newFrameSync
5 , FrameSync
  , Deadline(..)
  , Statistics(..)
  ) where

10 import Prelude hiding ((++))

import Control.Monad (replicateM_)
import Data.IORef (newIORef, readIORef, writeIORef)
import Data.List (foldl')
15 import Data.Monoid (Monoid(..))

import Fractal.GraphGrow.Analysis.Statistics hiding (s0)

data Statistics = Statistics
20  { statFrameTime :: !(StatOrd Double)
  , statSpareTime :: !(StatOrd Double)
  , statOk        :: !(StatOrd Double)
  , statMissed    :: !Int
  , statCount     :: !Int
25  }

instance Monoid Statistics where
  mempty = let e = mempty in Statistics e e e 0 0
  Statistics a b c d e `mappend` Statistics s t u v w =
30    let (++) = mappend in Statistics (a++s)(b++t)(c++u)(d+v)(e+w)
  mconcat = foldl' mappend mempty

data Deadline = Missed | Ok !Double
  deriving (Read, Show, Eq, Ord)
35
type FrameSync = IO (Deadline, Statistics)

data FrameSyncOptions = FrameSyncOptions
  { frameSyncMeasure  :: Int
40  , frameSyncMultiple :: Int
  , frameSyncMissed   :: Double
  , frameSyncTarget   :: Double
  }

45 defaultFrameSyncOptions :: FrameSyncOptions
```

```
defaultFrameSyncOptions = FrameSyncOptions
  { frameSyncMeasure  = 60
  , frameSyncMultiple = 1
  , frameSyncMissed   = 1.5
  , frameSyncTarget   = 0.75
  }

newFrameSync :: IO Double -> IO () -> FrameSyncOptions -> IO (Double, FrameSync)
newFrameSync getTime swapBuffers opts = do
  frameInterval <- measureFrameInterval getTime swapBuffers (frameSyncMeasure ↙
      ↳ opts)
  statRef <- newIORef mempty
  frameTimeRef <- newIORef =<< getTime
  let interval = fromIntegral (frameSyncMultiple opts) * frameInterval
      frameSync = do
        lastTime <- readIORef frameTimeRef
        preTime <- getTime
        swapBuffers
        frameTime <- getTime
        writeIORef frameTimeRef frameTime
        s0 <- readIORef statRef
        let s = s0{ statFrameTime = statFrameTime s0 `mappend` statOrd (↙
            ↳ frameTime - lastTime)
                  , statSpareTime = statSpareTime s0 `mappend` statOrd (↙
                      ↳ frameTime - preTime )
                  , statCount     = statCount s0 + 1
                  }
            missed = frameTime - lastTime > interval * frameSyncMissed opts
            delta = (frameTime - preTime) / (interval * (1 - frameSyncTarget ↙
                ↳ opts))
            result@(_, s')
              | missed    = (Missed,   s{ statMissed = statMissed s + 1 })
              | otherwise = (Ok delta, s{ statOk = statOk s `mappend` statOrd ↙
                  ↳ delta })
        writeIORef statRef s'
        return result
  return (recip interval, frameSync)

measureFrameInterval :: IO Double -> IO () -> Int -> IO Double
measureFrameInterval getTime swapBuffers count = do
  swapBuffers
  startTime <- getTime
  replicateM_ count swapBuffers
  endTime <- getTime
  return $ (endTime - startTime) / fromIntegral count
```

18 Fractal/GraphGrow/Main.hs

```
module Main (main) where

import Control.Concurrent (forkIO, killThread)
import Control.Concurrent.BoundedChan (newBoundedChan, readChan, writeChan)
import Control.Monad (join, replicateM_, when)
import Data.IORef (newIORef, readIORef, writeIORef)
import qualified Data.Map as M
import Data.Maybe (fromMaybe)
import System.Environment (getArgs)
```

```
10  import System.Exit (exitFailure)
    import System.IO (hPutStrLn, stderr)
    import System.Random (newStdGen)

    import qualified Graphics.UI.Gtk        as G
15  import qualified Graphics.UI.Gtk.OpenGL as G
    import qualified System.Glib.GDateTime  as G

    import Fractal.GraphGrow.Engine.Geometry
    import Fractal.GraphGrow.Engine.Graph
20  import Fractal.GraphGrow.Engine.Zoomer
    import Fractal.GraphGrow.Text.Parse
    import Fractal.GraphGrow.Text.Compile
    import Fractal.GraphGrow.GUI.DrawPoints
    import Fractal.GraphGrow.GUI.FrameSync
25  import Fractal.GraphGrow.Utils
    import Fractal.GraphGrow.Analysis.Statistics

    readGraph :: Maybe FilePath -> Maybe String -> IO (Maybe (VGraph, Node))
    readGraph mf mw = do
30    s <- case mf of
        Nothing -> getContents
        Just f -> readFile f
      return $ do
        (ws, g) <- compileText `fmap` parseText s
35      v <- vectorize (annotate g)
        let n = fromMaybe (Node 0) ((`M.lookup` ws) =<< mw)
        return (v, n)

    errLn :: String -> IO ()
40  errLn = hPutStrLn stderr

    usageMessage :: IO ()
    usageMessage = errLn "usage: FIXME"

45  inputMessage :: IO ()
    inputMessage = errLn "bad input"

    vsyncMessage :: Double -> IO ()
    vsyncMessage fps = do
50    errLn$"insane fps: " ++ show fps
      errLn "enable OpenGL sync-to-vblank and retry"

    statsMessage :: Statistics -> IO ()
    statsMessage s = do
55    let frames = statCount s
          totalTime = (s1 . fst . statFrameTime) s
          fps = fromIntegral frames / totalTime
          pretty ss = "\t" ++ show mi ++ "\t" ++ show mean ++ "\t" ++ show ma ++ "\t↙
              ↳ " ++ show dev
            where
60            (mi, ma) = fromMaybe (0/0, 0/0) . getMinMax . snd $ ss
              mean = getMean (fst ss)
              dev = getStdDev (fst ss)
      errLn "[ STATISTICS"
      errLn $ "FPS\t" ++ show fps ++ "\t( " ++ show frames ++ " f / " ++ show ↙
          ↳ totalTime ++ " s ) [ " ++ show (statMissed s) ++ " MISSED ]"
```

```
    errLn $ "FRAME" ++ pretty (statFrameTime s)
    errLn $ "SPARE" ++ pretty (statSpareTime s)
    errLn $ "DELTA" ++ pretty (statOk s)
    errLn "]"

insaneFPS :: Double -> Bool
insaneFPS fps = fps < 20 || 200 < fps

parseArgs :: [String] -> Maybe Config
parseArgs [file] = Just defaultConfig{ configGraph = if file == "-" then Nothing else Just file }
parseArgs [file, node] = Just defaultConfig{ configGraph = if file == "-" then Nothing else Just file, configNode = Just node }
parseArgs _ = Nothing

data Config = Config
  { configGraph :: Maybe FilePath
  , configNode :: Maybe String
  , configWidth :: Int
  , configHeight :: Int
  , configMinQuality :: Double
  , configMaxQuality :: Double
  , configVerbose :: Bool
  }

defaultConfig :: Config
defaultConfig = Config
  { configGraph = Nothing
  , configNode = Nothing
  , configWidth = 1024
  , configHeight = 576
  , configMinQuality = recip 16
  , configMaxQuality = 16
  , configVerbose = False
  }

configSize :: Config -> Double
configSize config = sqrt (fromIntegral $ sqr (configWidth config) + sqr (configHeight config))

newtype Continue = Continue{ runContinue :: IO Continue }

skip :: Continue
skip = Continue (return skip)

abort :: IO Continue
abort = exitFailure >> return skip

main :: IO ()
main = do
  args <- getArgs
  case parseArgs args of
    Nothing -> usageMessage
    Just config -> do
      mgraph <- readGraph (configGraph config) (configNode config)
      case mgraph of
        Nothing -> inputMessage
```

```
        Just (graph, node) -> do
          let initial = defaultZoomerOptions graph node
          withGL "GraphGrow" (configWidth config) (configHeight config) $ \↙
              ↳ getTime swapBuffers -> do
            (fps, frameSync) <- newFrameSync getTime swapBuffers ↙
                ↳ defaultFrameSyncOptions
            if insaneFPS fps
              then vsyncMessage fps >> abort
              else withPointDraw (zoomerOuterBox initial) (\s -> errLn s >> ↙
                  ↳ abort) $ \draw -> do
                worker <- newIORef Nothing
                let queueSize = 2
                    restart = do
                      maybe (return ()) killThread =<< readIORef worker
                      qualityChan <- newBoundedChan queueSize
                      pointsChan <- newBoundedChan queueSize
                      writeIORef worker . Just =<< forkIO . workerThread (↙
                          ↳ readChan qualityChan) (writeChan pointsChan) . ↙
                          ↳ zoomer initial =<< newStdGen
                      replicateM_ queueSize $ writeChan qualityChan (↙
                          ↳ configMaxQuality config)
                      loop (readChan pointsChan) (writeChan qualityChan)
                    workerThread getQuality putPoints = go
                      where
                        go zoom = do
                          quality <- getQuality
                          let pixelSize = quality * boxSize (zoomerOuterBox ↙
                              ↳ initial) / configSize config
                              Zoomed points zoom' = zoom pixelSize
                              sanePoints = filter saneSize points
                              saneSize (STwo _ sizeSq) = sizeSq < sqr (boxSize (↙
                                  ↳ zoomerOuterBox initial)) / 4
                          if null sanePoints then putPoints [] else do
                            () <- putPoints =<< flattenPoints sanePoints
                            go zoom'
                    loop getPoints putQuality = go (configMaxQuality config)
                      where
                        go quality = do
                          points <- getPoints
                          if null points then restart else do
                            _drawnCount <- draw points
                            (deadline, stats) <- frameSync
                            when (configVerbose config && statCount stats `mod` ↙
                                ↳ 256 == 0) $
                              statsMessage stats
                            let e = negate . recip . fromIntegral $ 16 * ↙
                                ↳ queueSize
                                slew = case deadline of
                                  Missed -> sqrt 2
                                  Ok k
                                    | k > 1     -> k ** e
                                    | otherwise -> k ** e
                                slew' = clamp slew (sqrt 0.5) (sqrt 2)
                                quality' = clamp (quality * slew') (↙
                                    ↳ configMinQuality config) (configMaxQuality↙
                                    ↳  config)
                            putQuality quality'
```

```
                                return $ Continue (go quality')
                restart

withGL :: String -> Int -> Int -> (IO Double -> IO () -> IO Continue) -> IO ()
withGL title w h go = do
  _ <- G.initGUI
  window <- G.windowNew
  _ <- G.onDestroy window G.mainQuit
  G.windowSetDefaultSize window w h
  G.windowSetGeometryHints window (Nothing `asTypeOf` Just window) (Just (w, h))↙
      ↳ (Just (w, h)) Nothing Nothing Nothing
  canvas <- G.glDrawingAreaNew =<< G.glConfigNew [G.GLModeRGBA, G.GLModeDouble]
  G.widgetSetSizeRequest canvas w h
  continue <- newIORef skip
  swapBuffersR <- newIORef (return ())
  let getTime = do
        G.GTimeVal sec usec <- G.gGetCurrentTime
        return $ fromIntegral sec + fromIntegral usec / 1000000
      swapBuffers = join (readIORef swapBuffersR)
      tick = do
        _ <- G.withGLDrawingArea canvas $ \gl -> do
          writeIORef swapBuffersR (G.glDrawableSwapBuffers gl)
          writeIORef continue =<< runContinue =<< readIORef continue
          writeIORef swapBuffersR (return ())
        return True
  _ <- G.onRealize canvas $ do
    _ <- G.withGLDrawingArea canvas $ \gl -> do
      writeIORef swapBuffersR (G.glDrawableSwapBuffers gl)
      writeIORef continue =<< go getTime swapBuffers
      writeIORef swapBuffersR (return ())
    G.timeoutAddFull tick G.priorityDefaultIdle 1
  return ()
  event <- G.eventBoxNew
  _ <- G.onKeyPress event $ \_ -> G.mainQuit >> return True
  G.set event  [G.containerBorderWidth G.:= 0, G.containerChild G.:= canvas]
  G.set window [G.windowTitle G.:= title, G.containerChild G.:= event]
  G.widgetShowAll window
  G.mainGUI
```

19 Fractal/GraphGrow/Text/Compile.hs

```
module Fractal.GraphGrow.Text.Compile (compileText) where

import Control.Monad (forM_)
import Control.Monad.Identity (Identity, runIdentity)
import Control.Monad.Supply (SupplyT, evalSupplyT, supply)
import Control.Monad.State (StateT, execStateT, get, put)
import Control.Monad.Writer (WriterT, runWriterT, tell)
import Control.Monad.Trans (lift)
import Data.Map (Map)
import qualified Data.Map as M

import Fractal.GraphGrow.Engine.Geometry (Transform)
import Fractal.GraphGrow.Engine.Graph (MGraph(..), Node(..))
import Fractal.GraphGrow.Text.Glyphs (string, char)

-- the input
```

```
type Specification = [(String, [String])]

-- the output
type Environment = Map String Node

-- intermediate data
type PreGraph = [(Node, [(Node, Transform)])]

-- the main compiler monad stack
type Compile = StateT Environment (WriterT PreGraph (SupplyT Node Identity))

runCompile :: Compile () -> (Environment, MGraph)
runCompile = runIdentity . supplyingNodes . writingGraph . trackingEnvironment
  where
    supplyingNodes m = evalSupplyT m $ map Node [0..]
    writingGraph m = fmap (MGraph . M.fromListWith (++)) `fmap` runWriterT m
    trackingEnvironment m = execStateT m M.empty

-- allocate stable nodes for strings
namedNode :: String -> Compile Node
namedNode s = do
  m <- get
  case M.lookup s m of
    Nothing -> do
      n <- supply
      put $ M.insert s n m
      return n
    Just n -> do
      return n

-- allocate unique nodes for characters
node :: Compile Node
node = supply

-- add an edge to the directed graph
edge :: Node -> Node -> Transform -> Compile ()
edge from to transform = tell [(from, [(to, transform)])]

-- the main compiler program
-- maps each stroke of each character in each "source" string..
-- .. to one of that source's "target" strings
compileText :: Specification -> (Environment, MGraph)
compileText graph = runCompile $ do
  forM_ graph $ \(source, targets) -> do
    sourceNode <- namedNode source
    flip evalSupplyT (cycle targets) $ do
      -- re-using supply makes sense, but either
      --   a) there are a lot of 'lift's (as in this code)
      --   b) needs orphan instances to automatigally lift
      forM_ (source `zip` string (length source)) $ \(sourceChar, charTransform)↙
          ↳ -> do
        charNode <- lift node
        lift $ edge sourceNode charNode charTransform
        forM_ (char sourceChar) $ \strokeTransform -> do
          target <- supply
          targetNode <- lift $ namedNode target
          lift $ edge charNode targetNode strokeTransform
```

20 Fractal/GraphGrow/Text/Glyphs.hs

```
module Fractal.GraphGrow.Text.Glyphs
  ( -- * The characters within a string.
    string
    -- * The strokes within a character.
  , char
  ) where

import Data.Char (toUpper)
import Data.Maybe (fromMaybe)
import Data.Map as M

import Fractal.GraphGrow.Engine.Geometry (Transform, o, scale, rotate, translate)
import Fractal.GraphGrow.Utils (sqr)

string :: Int -> [Transform]
string n =
  [ translate (2 * (fromIntegral i / n' - 0.5)) 0 `o` scale (2 / n')
  | i <- [1 .. n]
  ]
  where
    n' = fromIntegral n + 1

char :: Char -> [Transform]
char c = fromMaybe [] $ M.lookup (toUpper c) chars

chars :: Map Char [Transform]
chars = M.fromList
  [ ( c
    , [ let x = (x1 + x0) / 2
            y = (y1 + y0) / 2
            u = (x1 - x0) / 2
            v = (y1 - y0) / 2
            a = atan2 v u / (2 * pi) - strokeTilt
            s = sqrt $ sqr u + sqr v
        in  scale characterSize `o` translate x y `o` scale s `o` rotate a
      | ((x0,y0),(x1,y1)) <- strokes
      ]
    )
  | (c, strokes) <- font
  ]

characterSize :: Double
characterSize = 1 / sqrt 7

strokeTilt :: Double
strokeTilt = phi / 100

phi :: Double
phi = (sqrt 5 + 1) / 2

type Coord = (Double, Double)

type Stroke = (Coord, Coord)
```

```
font :: [(Char, [Stroke])]
font =
  [('A',[((-1,-1),(-1,0)),((-1,0),(0,1)),((0,1),(1,0)),((1,0),(1,-1)),((-1,0)↙
      ↳ ,(1,0))])
  ,('B',[((-1,1),(-1,0)),((-1,0),(-1,-1)),((-1,0),(1,0)),((1,0),(1,-1)),((1,-1)↙
      ↳ ,(-1,-1))])
  ,('C',[((1,1),(-1,1)),((-1,1),(-1,-1)),((-1,-1),(1,-1))])
  ,('D',[((1,1),(1,0)),((1,0),(1,-1)),((1,0),(-1,0)),((-1,0),(-1,-1)),((-1,-1)↙
      ↳ ,(1,-1))])
  ,('E',[((-1,1),(-1,0)),((-1,0),(-1,-1)),((-1,1),(1,1)),((-1,0),(0,0)),((-1,-1)↙
      ↳ ,(1,-1))])
  ,('F',[((-1,1),(-1,0)),((-1,0),(-1,-1)),((-1,1),(1,1)),((-1,0),(0,0))])
  ,('G',[((1,1),(-1,1)),((-1,1),(-1,-1)),((-1,-1),(1,-1)),((1,-1),(1,0)),((1,0)↙
      ↳ ,(0,0))])
  ,('H',[((-1,-1),(-1,1)),((-1,0),(1,0)),((1,-1),(1,1))])
  ,('I',[((0,1),(0,-1))])
  ,('J',[((0,1),(0,-1)),((0,-1),(-1,-1))])
  ,('K',[((-1,-1),(-1,1)),((-1,0),(1,1)),((-1,0),(1,-1))])
  ,('L',[((-1,1),(-1,-1)),((-1,-1),(1,-1))])
  ,('M',[((-1,-1),(-1,1)),((-1,1),(0,0)),((0,0),(1,1)),((1,1),(1,-1))])
  ,('N',[((-1,-1),(-1,1)),((-1,1),(1,-1)),((1,-1),(1,1))])
  ,('O',[((1,1),(-1,1)),((-1,1),(-1,-1)),((-1,-1),(1,-1)),((1,-1),(1,1))])
  ,('P',[((-1,-1),(-1,0)),((-1,0),(-1,1)),((-1,0),(1,0)),((1,0),(1,1)),((1,1)↙
      ↳ ,(-1,1))])
  ,('Q',[((1,-1),(1,0)),((1,0),(1,1)),((1,0),(-1,0)),((-1,0),(-1,1)),((-1,1)↙
      ↳ ,(1,1))])
  ,('R',[((-1,0),(-1,-1)),((-1,0),(1,0))])
  ,('S',[((1,1),(-1,1)),((-1,1),(-1,0)),((-1,0),(1,0)),((1,0),(1,-1)),((1,-1)↙
      ↳ ,(-1,-1))])
  ,('T',[((-1,1),(1,1)),((0,1),(0,-1))])
  ,('U',[((-1,1),(-1,-1)),((-1,-1),(1,-1)),((1,-1),(1,1))])
  ,('V',[((-1,1),(0,-1)),((0,-1),(1,1))])
  ,('W',[((-1,1),(-1,-1)),((-1,-1),(0,0)),((0,0),(1,-1)),((1,-1),(1,1))])
  ,('X',[((-1,-1),(0,0)),((0,0),(1,1)),((-1,1),(0,0)),((0,0),(1,-1))])
  ,('Y',[((-1,1),(0,0)),((1,1),(0,0)),((0,0),(0,-1))])
  ,('Z',[((-1,1),(1,1)),((1,1),(-1,-1)),((-1,-1),(1,-1))])
  ]
```

21 Fractal/GraphGrow/Text/Parse.hs

```
module Fractal.GraphGrow.Text.Parse (parseText) where

parseText :: String -> Maybe [(String, [String])]
parseText = mapM (uncons . words) . lines
  where
    uncons (w:_:ws) = Just (w, ws)
    uncons _ = Nothing
```

22 Fractal/GraphGrow/Utils.hs

```
module Fractal.GraphGrow.Utils
  ( -- * Miscellaneous maths.
    sum'
  , sqr
  , clamp
  , roundUpTwo
```

```
    -- * Strict tuples.
  , STwo(..)
  , SThree(..)
    -- * Ordered lists.
  , mergeSortedBy
  , mergeSortedsBy
  ) where

import Data.List (foldl')

sum' :: Num a => [a] -> a
sum' = foldl' (+) 0

sqr :: Num a => a -> a
sqr a = a * a

clamp :: Ord a => a -> a -> a -> a
clamp x mi ma = mi `max` x `min` ma

roundUpTwo :: (Num a, Ord a) => a -> a
roundUpTwo n = head . dropWhile (< n) . iterate (2 *) $ 1

data STwo a b = STwo{ fstTwo :: !a, sndTwo :: !b }
  deriving (Read, Show, Eq, Ord)

data SThree a b c = SThree{ fstThree :: !a, sndThree :: !b, thdThree :: !c }
  deriving (Read, Show, Eq, Ord)

mergeSortedBy :: (a -> a -> Ordering) -> [a] -> [a] -> [a]
mergeSortedBy cmp = go
  where
    go [] ys = ys
    go xs [] = xs
    go xxs@(x:xs) yys@(y:ys) = case x `cmp` y of
      LT -> x : go xs yys
      EQ -> x : y : go xs ys
      GT -> y : go xxs ys

mergeSortedsBy :: (a -> a -> Ordering) -> [[a]] -> [a]
mergeSortedsBy cmp = foldr (mergeSortedBy cmp) []
```

23 .gitignore

```
.cabal-sandbox
cabal.sandbox.config
dist
-
*.ogv
*.wav
*.tif
c/graphgrow-filter
*.glsl.c
```

24 graphgrow3/audio/.gitignore

```
graphgrow-audio
graphgrow-audio-test
```

25 graphgrow3/audio/graphgrow-audio.cc

```
#include <assert.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <jack/jack.h>
#include <lo/lo.h>

#define PI 3.141592653589793
#define SR 48000
#define BLOCKSIZE 64
// DELAYSIZE must be a multiple of BLOCKSIZE
#define DELAYSIZE 2048

#define GG_HIP 25.0f
#define GG_LOP 5000.0f

typedef float sample;

// 256 is BLOCKSIZE * sizeof(sample)
typedef float    signal __attribute__ ((vector_size (256)));
typedef int32_t isignal __attribute__ ((vector_size (256)));

// 8192 is DELAYSIZE * sizeof(sample)
typedef sample delaybuffer __attribute__ ((vector_size (8192)));

#define   likely(x) __builtin_expect((x),1)
#define unlikely(x) __builtin_expect((x),0)

// postcondition: 0 <= phase <= 1
static inline float cos_reduce(float phase) {
  float p = fabsf(phase);
  // p >= 0
  if (likely(p < (1 << 24))) {
    int q = p;
    return p - q;
  } else {
    if (unlikely(isnanf(p) || isinff(p))) {
      // return NaN
      return p - p;
    } else {
      // int could overflow, and it will be integral anyway
      return 0.0f;
    }
  }
}

signal cos_reduce(const signal &phase) {
  signal p = phase >= 0.0f ? phase : -phase;
  isignal i = isignal(p);
  signal q = signal(i);
  return p < sample(1 << 24) ? p - q : 0.0f;
}
```

```
// precondition: 0 <= phase <= 1
static inline float cos_unsafe(float phase) {
  float p = fabsf(4.0f * phase - 2.0f) - 1.0f;
  // p in -1 .. 1
  float s
    = 1.5707963267948965580e+00f * p
    - 6.4596271553942852250e-01f * p * p * p
    + 7.9685048314861006702e-02f * p * p * p * p * p
    - 4.6672571910271187789e-03f * p * p * p * p * p * p * p
    + 1.4859762069630022552e-04f * p * p * p * p * p * p * p * p * p;
  // compiler figures out optimal simd multiplications
  return s;
}

signal cos_unsafe(const signal &phase) {
  signal p = 4.0f * phase - 2.0f;
  p = p >= 0.0f ? p : -p;
  p -= 1.0f;
  // p in -1 .. 1
  signal p2 = p * p;
  signal p3 = p2 * p;
  signal p5 = p2 * p3;
  signal p7 = p2 * p5;
  signal p9 = p2 * p7;
  signal s
    = 1.5707963267948965580e+00f * p
    - 6.4596271553942852250e-01f * p3
    + 7.9685048314861006702e-02f * p5
    - 4.6672571910271187789e-03f * p7
    + 1.4859762069630022552e-04f * p9;
  return s;
}

inline sample cos_poly(sample phase) {
  return cos_unsafe(cos_reduce(phase));
}

inline sample sin_poly(sample phase) {
  return cos_poly(phase - 0.25f);
}

signal cos(const signal &phase) {
  return cos_unsafe(cos_reduce(phase));
}

signal sin(const signal &phase) {
  return cos(phase - 0.25f);
}

inline sample dbtorms(sample f) {
  return expf(0.5f * 0.23025850929940458f * (fminf(f, 870.0f) - 100.0f));
}

inline sample rmstodb(sample f) {
  return 100.0f + 2.0f * 4.3429448190325175f * logf(f);
}
```

```
inline sample wrap(sample a) {
  return a - floorf(a);
}

signal wrap(const signal &a) {
  signal c;
  for (int i = 0; i < BLOCKSIZE; ++i)
    c[i] = wrap(a[i]);
  return c;
#if 0
  isignal i = isignal(a) + (a < 0.0f); // vector compare produces ~0
  signal q = signal(i);
  return a - q;
#endif
}

signal exp(const signal &a)
{
  signal c;
  for (int i = 0; i < BLOCKSIZE; ++i)
    c[i] = expf(a[i]);
  return c;
}

signal log(const signal &a)
{
  signal c;
  for (int i = 0; i < BLOCKSIZE; ++i)
    c[i] = logf(a[i]);
  return c;
}

signal sqrt(const signal &a)
{
  signal c;
  for (int i = 0; i < BLOCKSIZE; ++i)
    c[i] = sqrtf(a[i]);
  return c;
}

signal tanh(const signal &a)
{
  signal c;
  for (int i = 0; i < BLOCKSIZE; ++i)
    c[i] = tanhf(a[i]);
  return c;
}

signal sqr(const signal &a)
{
  return a * a;
}

signal min(const signal &a, const signal &b) {
  return a < b ? a : b;
}
```

```
170
    signal min(const signal &a, sample b) {
      return a < b ? a : b;
    }

175 signal min(sample a, const signal &b) {
      return a < b ? a : b;
    }

    sample min(sample a, sample b) {
180   return a < b ? a : b;
    }

    signal dbtorms(const signal &f)
    {
185   return exp(0.5f * 0.23025850929940458f * (min(f, 870.0f) - 100.0f));
    }

    signal rmstodb(const signal &f)
    {
190   return 100.0f + 2.0f * 4.3429448190325175f * log(f);
    }

    struct lop
    {
195   // http://www.arpchord.com/pdf/coeffs_first_order_filters_0p1.pdf
      sample k, k12, x1, y1;
      lop(sample hz) : x1(0.0f), y1(0.0f)
      {
        double a = 2 * PI * hz / double(SR);
200     k = (1 - sin(a)) / cos(a);
        k12 = (1 - k) / 2;
      };
      signal operator()(const signal &x)
      {
205     signal y;
        sample last_x = x1;
        sample last_y = y1;
        for (int i = 0; i < BLOCKSIZE; ++i)
        {
210       sample now_x = x[i];
          last_y = y[i] = (now_x + last_x) * k12 + k * last_y;
          last_x = now_x;
        }
        x1 = last_x;
215     y1 = last_y;
        return y;
      };
    };

220 struct hip
    {
      // http://www.arpchord.com/pdf/coeffs_first_order_filters_0p1.pdf
      sample k, k12, x1, y1;
      hip(sample hz) : x1(0.0f), y1(0.0f)
225   {
        double a = 2 * PI * hz / double(SR);
```

```
    k = (1 - sin(a)) / cos(a);
    k12 = (1 + k) / 2;
  };
  signal operator()(const signal &x)
  {
    signal y;
    sample last_x = x1;
    sample last_y = y1;
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      sample now_x = x[i];
      last_y = y[i] = (now_x - last_x) * k12 + k * last_y;
      last_x = now_x;
    }
    x1 = last_x;
    y1 = last_y;
    return y;
  };
};

struct vsig
{
  sample x1;
  vsig(sample x) : x1(x)
  {
  };
  signal operator()(sample x)
  {
    signal y;
    sample last_x = x1;
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      sample t = (sample(i) + 0.5f) / sample(BLOCKSIZE);
      y[i] = last_x + (x - last_x) * t;
    }
    x1 = x;
    return y;
  };
};

struct phasor
{
  sample p;
  phasor() : p(0.0f)
  {
  };
  signal operator()(const signal &hz)
  {
    signal y;
    sample last_p = p;
    for (int i = 0; i < BLOCKSIZE; ++i)
      last_p = y[i] = wrap(last_p + hz[i] / sample(SR));
    p = last_p;
    return y;
  };
};
```

```
struct noise
{
  // pd-0.47-0/src/d_osc.c
  static int32_t seed;
  int32_t val;
  noise()
  {
    val = (seed *= 1319);
  };
  signal operator()()
  {
    signal y;
    int32_t v = val;
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      y[i] = sample((v & 0x7fffffff) - 0x40000000) * sample(1.0 / 0x40000000);
      v = v * 435898247 + 382842987;
    }
    val = v;
    return y;
  };
};
int32_t noise::seed = 307;

struct delay
{
  delaybuffer v;
  int w1;
  delay() : w1(0)
  {
    for (int i = 0; i < DELAYSIZE; ++i)
      v[i] = 0.0f;
  };
  void write(const signal &x)
  {
    int w = w1;
    assert(w + BLOCKSIZE <= DELAYSIZE);
    for (int i = 0; i < BLOCKSIZE; ++i)
      v[w + i] = x[i];
    w += BLOCKSIZE;
    if (w == DELAYSIZE)
      w = 0;
    w1 = w;
  };
  signal operator()(sample ms)
  {
    signal y;
    int w = w1;
    int r = roundf(ms * sample(SR / 1000.0f));
    assert(0 <= r);
    assert(r < DELAYSIZE - BLOCKSIZE);
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      int k = w - r + i;
      k = k < 0 ? k + DELAYSIZE : k;
      assert(0 <= k);
      assert(k < DELAYSIZE);
```

```
      y[i] = v[k];
    }
    return y;
  };
  signal operator()(const signal &ms)
  {
    // https://en.wikipedia.org/wiki/Cubic_Hermite_spline#↙
       ↪ Interpolation_on_the_unit_interval_without_exact_derivatives
    signal y;
    int w = w1;
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      sample d = ms[i] * sample(SR / 1000.0f);
      int d1 = floorf(d);
      int d0 = d1 - 1;
      int d2 = d1 + 1;
      int d3 = d1 + 2;
      sample t = d - d1;
      assert(0 <= d0);
      assert(d3 < DELAYSIZE - BLOCKSIZE);
      assert(0 <= t);
      assert(t < 1);
      int r0 = w - d0 + i;
      int r1 = w - d1 + i;
      int r2 = w - d2 + i;
      int r3 = w - d3 + i;
      r0 = r0 < 0 ? r0 + DELAYSIZE : r0;
      r1 = r1 < 0 ? r1 + DELAYSIZE : r1;
      r2 = r2 < 0 ? r2 + DELAYSIZE : r2;
      r3 = r3 < 0 ? r3 + DELAYSIZE : r3;
      assert(0 <= r0);
      assert(0 <= r1);
      assert(0 <= r2);
      assert(0 <= r3);
      assert(r0 < DELAYSIZE);
      assert(r1 < DELAYSIZE);
      assert(r2 < DELAYSIZE);
      assert(r3 < DELAYSIZE);
      sample y0 = v[r0];
      sample y1 = v[r1];
      sample y2 = v[r2];
      sample y3 = v[r3];
      sample a0 = -t*t*t + 2.0f*t*t - t;
      sample a1 = 3.0f*t*t*t - 5.0f*t*t + 2.0f;
      sample a2 = -3.0f*t*t*t + 4.0f*t*t + t;
      sample a3 = t*t*t - t*t;
      y[i] = 0.5f * (a0 * y0 + a1 * y1 + a2 * y2 + a3 * y3);
    }
    return y;
  };
};

struct pitchshift
{
  delay del;
  phasor p;
  signal operator()(const signal &audio, const signal &transpose)
```

```
  {
    const sample d = 20.0f;
    del.write(audio);
    signal hz = (1.0f - exp(transpose * 0.05776f)) * (1000.0f / d);
    signal phase1 = p(hz);
    signal phase2 = wrap(phase1 + 0.5f);
    return cos((phase1 - 0.5f) * 0.5f) * del(d * phase1 + 2.0f)
        + cos((phase2 - 0.5f) * 0.5f) * del(d * phase2 + 2.0f);
  };
};

struct compress
{
  sample threshold;
  sample factor;
  hip hi;
  lop lo1, lo2;
  compress(sample db)
  : threshold(db), factor(0.25f / dbtorms((100.0f - db) * 0.125f + db)), hi(5.0f↙
    ↳ ), lo1(10.0f), lo2(25.0f)
  {
  };
  signal operator()(const signal &audio)
  {
    signal rms = lo2(0.01f + sqrt(lo1(sqr(hi(audio)))));
    signal db = rmstodb(rms);
    db = db > threshold ? threshold + (db - threshold) * 0.125f : threshold;
    signal gain = factor * dbtorms(db);
    return tanh(audio / rms * gain);
  };
};

struct edge
{
  signal *inl, *inr;
  signal *outl, *outr;
  pitchshift psl, psr;
  vsig scale, pan, level;
  edge(signal *from_l, signal *from_r, signal *to_l, signal *to_r)
  : inl(from_l), inr(from_r), outl(to_l), outr(to_r), scale(0.5), pan(0.5), ↙
    ↳ level(0)
  {
  };
  edge() : edge(0, 0, 0, 0) { };
  void operator()(sample iscale, sample ipan, sample ilevel)
  {
    signal vscale = scale(iscale);
    signal vpan = pan(0.25f * ipan);
    signal vlevel = level(ilevel) * vscale;
    signal transpose = log(vscale * 2.0f) * sample(-12.0f / logf(2.0f));
    *outl += vlevel * cos(vpan) * psl(*inl, transpose);
    *outr += vlevel * sin(vpan) * psr(*inr, transpose);
  };
};

struct rule
{
```

```
  signal send_l, send_r, return_l, return_r;
  signal *out_l, *out_r;
  delay delayl, delayr;
  noise noisel, noiser;
  hip hipl, hipr;
  lop lopl, lopr;
  compress compressl, compressr;
  sample delaytime;
  rule(signal *outl, signal *outr, sample delaytime)
  : out_l(outl), out_r(outr), hipl(GG_HIP), hipr(GG_HIP), lopl(GG_LOP), lopr(
    ↘ GG_LOP), compressl(48.0f), compressr(48.0f), delaytime(delaytime)
  {
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      send_l[i] = 0.0f;
      send_r[i] = 0.0f;
      return_l[i] = 0.0f;
      return_r[i] = 0.0f;
    }
  };
  rule() : rule(0, 0, 2) { };
  void operator()()
  {
    send_l = noisel() * 1.0e-6f + delayl(delaytime);
    send_r = noiser() * 1.0e-6f + delayr(delaytime);
    signal l = compressl(lopl(hipl(return_l)));
    signal r = compressr(lopr(hipr(return_r)));
    // one block delay
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      return_l[i] = 0.0f;
      return_r[i] = 0.0f;
    }
    delayl.write(l);
    delayr.write(r);
    *out_l += l;
    *out_r += r;
  };
};

struct graphgrow
{
  signal out_l, out_r;
  hip hil, hir;
  compress compressl, compressr;
  rule    rules[4];
  edge    edges[4][4][8];
  /* { osc 1152 bytes { */
  sample  scale[4][4][8];
  sample  pan  [4][4][8];
  uint8_t level[4][4][8];
  /* } osc } */
  uint8_t last_level[4][4][8];
  graphgrow()
  : hil(GG_HIP), hir(GG_HIP), compressl(48.0f), compressr(48.0f)
  {
    for (int i = 0; i < BLOCKSIZE; ++i)
```

```
    {
      out_l[i] = 0.0f;
      out_r[i] = 0.0f;
    }
    for (int i = 0; i < 4; ++i)
      rules[i] = rule(&out_l, &out_r, 10.0f + sample(i));
    for (int i = 0; i < 4; ++i)
      for (int j = 0; j < 4; ++j)
        for (int k = 0; k < 8; ++k)
        {
          edges[i][j][k] = edge(&rules[i].send_l, &rules[i].send_r, &rules[j].↙
              ↪ return_l, &rules[j].return_r);
          scale[i][j][k] = 0.5f;
          pan  [i][j][k] = 0.5f;
          level[i][j][k] = 0;
          last_level[i][j][k] = 0;
        }
  };
  void operator()()
  {
    for (int i = 0; i < BLOCKSIZE; ++i)
    {
      out_l[i] = 0.0f;
      out_r[i] = 0.0f;
    }
    for (int i = 0; i < 4; ++i)
      rules[i]();
    for (int i = 0; i < 4; ++i)
      for (int j = 0; j < 4; ++j)
        for (int k = 0; k < 8; ++k)
        {
          if (level[i][j][k] || last_level[i][j][k])
            edges[i][j][k](scale[i][j][k], pan[i][j][k], level[i][j][k]);
          last_level[i][j][k] = level[i][j][k];
        }
    out_l = compressl(hil(out_l));
    out_r = compressr(hir(out_r));
  };
};

struct control
{
  sample  scale[4][4][8];
  sample  pan  [4][4][8];
  uint8_t level[4][4][8];
};


struct
{
  graphgrow G;
  int count;
  jack_client_t *client;
  jack_port_t *out_port[2];
  volatile int in_processcb;
  volatile int running;
  control C;
```

```
} S;

int processcb(jack_nframes_t nframes, void *arg)
{
  S.in_processcb = 1;
  jack_default_audio_sample_t *out_l = (jack_default_audio_sample_t *) ↙
      ↳ jack_port_get_buffer(S.out_port[0], nframes);
  jack_default_audio_sample_t *out_r = (jack_default_audio_sample_t *) ↙
      ↳ jack_port_get_buffer(S.out_port[1], nframes);

  for (int i = 0; i < 4; ++i)
    for (int j = 0; j < 4; ++j)
      for (int k = 0; k < 8; ++k)
      {
        S.G.scale[i][j][k] = S.C.scale[i][j][k];
        S.G.pan  [i][j][k] = S.C.pan  [i][j][k];
        S.G.level[i][j][k] = S.C.level[i][j][k];
      }

  int count = S.count;
  for (jack_nframes_t i = 0; i < nframes; ++i)
  {
    if (count == 0)
      S.G();
    out_l[i] = S.G.out_l[count];
    out_r[i] = S.G.out_r[count];
    count = (count + 1) % BLOCKSIZE;
  }
  S.count = count;

  S.in_processcb = 0;
  return 0;
  (void) arg;
}

void errorcb(int num, const char *msg, const char *path)
{
  fprintf(stderr, "liblo server error %d in path %s: %s\n", num, path, msg);
}

int audiocb(const char *path, const char *types, lo_arg **argv, int argc, void *↙
    ↳ message, void *user_data)
{
  const int bytes = sizeof(S.C);
  int    size =  argv[0]->blob.size;
  void *data = &argv[0]->blob.data;
  if (size == bytes)
  {
    while (S.in_processcb)
      ;
    memcpy(&S.C, data, bytes);
  }
  return 0;
  (void) path;
  (void) types;
  (void) argc;
  (void) message;
```

```
  (void) user_data;
}

int quitcb(const char *path, const char *types, lo_arg **argv, int argc, void *↙
    ↳ message, void *user_data)
{
  S.running = 0;
  return 0;
  (void) path;
  (void) types;
  (void) argv;
  (void) argc;
  (void) message;
  (void) user_data;
}

int main()
{

   if (! (S.client = jack_client_open("graphgrow", JackNoStartServer, 0))) {
    fprintf(stderr, "jack server not running?\n");
    return 1;
  }
  jack_set_process_callback(S.client, processcb, 0);
  S.out_port[0] = jack_port_register(S.client, "output_1", ↙
      ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0);
  S.out_port[1] = jack_port_register(S.client, "output_2", ↙
      ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0);
  if (jack_activate(S.client)) {
    fprintf (stderr, "cannot activate JACK client");
    return 1;
  }
  if (jack_connect(S.client, "graphgrow:output_1", "system:playback_1")) {
    fprintf(stderr, "cannot connect output port\n");
  }
  if (jack_connect(S.client, "graphgrow:output_2", "system:playback_2")) {
    fprintf(stderr, "cannot connect output port\n");
  }

  S.running = 1;

  lo_server_thread lo = lo_server_thread_new("6060", errorcb);
  lo_server_thread_add_method(lo, "/audio", "b", audiocb, 0);
  lo_server_thread_add_method(lo, "/quit", "", quitcb, 0);
  lo_server_thread_start(lo);

  while (S.running)
    sleep(1);

  lo_server_thread_free(lo);

  jack_client_close(S.client);

  return 0;
}
```

26 graphgrow3/audio/graphgrow-audio-test.cc

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <lo/lo.h>

struct control
{
  float   scale[4][4][8];
  float   pan  [4][4][8];
  uint8_t level[4][4][8];
};

int main()
{
  control C;
  memset(&C, 0, sizeof(C));

  lo_address t = lo_address_new("255.255.255.255", "6060");

  while (1)
  {
    sleep(1);

    int i = rand() % 4;
    int j = rand() % 4;
    int k = rand() % 8;
    float   scale =  rand() / (double) RAND_MAX; scale *= scale; scale *= scale;
    float   pan   = (rand() / (double) RAND_MAX - 0.5) * 0.5 + 0.5;
    uint8_t level =  rand() % 2;
    C.scale[i][j][k] = scale;
    C.pan  [i][j][k] = pan;
    C.level[i][j][k] = level;

    int count = 0;
    for (i = 0; i < 4; ++i)
      for (j = 0; j < 4; ++j)
        for (k = 0; k < 8; ++k)
          count += C.level[i][j][k];
    fprintf(stderr, "  %d\r", count);

    lo_blob blob = lo_blob_new(sizeof(C), &C);
    if (blob)
    {
      if (lo_send(t, "/audio", "b", blob) == -1)
        fprintf(stderr, "OSC error %d: %s\n", lo_address_errno(t), ↙
            ↳ lo_address_errstr(t));
      lo_blob_free(blob);
    }
    else
      fprintf(stderr, "OSC error: couldn't lo_blob_new\n");

  }
  return 0;
}
```

27 graphgrow3/Audio.hs

```
module Audio (audioNew, Audio) where

import System.IO (Handle, hPutStr, hFlush)
import Data.IORef (IORef, newIORef, atomicModifyIORef)

import Control.Concurrent (threadDelay)

import Data.Map (Map)
import qualified Data.Map.Strict as M
import qualified Data.Map.Strict.Merge as M

import System.Process (createProcess, proc, close_fds, std_in, terminateProcess,↙
    ↪ StdStream(CreatePipe))

import Paths_graphgrow3 (getDataFileName)
import Types (NID, RID)

type Audio = (Int, [(NID, NID, RID, Double, Double)])

data AudioEngine = AudioEngine
  { aOld :: IORef (Map (RID, NID, NID) (RID, Double, Double))
  , aPull :: IO (Map RID Audio)
  , aPdSend :: Handle
  }

audioNew :: IO (Map RID Audio) -> IO (IO (), IO ())
audioNew pull = do
  aOld' <- newIORef M.empty
  patch <- getDataFileName "graphgrow3.pd"
  (_, _, _, pd) <- createProcess (proc "pd" ["-nogui", patch]){ close_fds = True↙
      ↪ }
  threadDelay (1000 * 1000)
  (Just h, _, _, pdsend) <- createProcess (proc "pdsend"  ["6060"]){ close_fds =↙
      ↪ True, std_in = CreatePipe }
  return (audio AudioEngine
    { aOld = aOld'
    , aPull = pull
    , aPdSend = h
    }, terminateProcess pdsend >> terminateProcess pd)

audio :: AudioEngine -> IO ()
audio a = do
  m <- aPull a
  let m' = M.fromList
             [ ((r0, min n0 n1, max n0 n1), (r1, s, x))
             | (r0, (_, es)) <- M.toList m
             , (n0, n1, r1, s, x) <- es
             ]
  m0' <- atomicModifyIORef (aOld a) $ \m0' -> (m', m0')
  let toDelete = m0' `M.difference` m'
      toAdd    = m' `M.difference` m0'
      toUpdate = M.merge
                   (M.mapMissing $ \_ (_, s1, x1) -> (Nothing, Just s1, Just x1))
                   M.dropMissing
                   (M.zipWithMatched $ \_ (r1, s1, x1) (r0, s0, x0) ->
```

```
                    ( if r1 == r0 then Nothing else Just r1
                    , if s1 == s0 then Nothing else Just s1
                    , if x1 == x0 then Nothing else Just x1
                    )
                  )
                  m' m0'
      deletes = unlines
        [ "delete " ++ (unwords $ map show [r0, n0, n1]) ++ ";"
        | (r0, n0, n1) <- M.keys toDelete
        ]
      adds = unlines
        [ "add " ++ (unwords $ map show [r0, n0, n1, r1]) ++ ";"
        | ((r0, n0, n1), (r1, _, _)) <- M.toList toAdd
        ]
      updateTargets = unlines
        [ "set " ++ (unwords $ map show [r0, n0, n1]) ++ " target " ++ show r1 ++ ";"
        | ((r0, n0, n1), (Just r1, _, _)) <- M.toList toUpdate
        ]
      updateScales = unlines
        [ "set " ++ (unwords $ map show [r0, n0, n1]) ++ " scale " ++ show s ++ ";"
        | ((r0, n0, n1), (_, Just s, _)) <- M.toList toUpdate
        ]
      updatePans = unlines
        [ "set " ++ (unwords $ map show [r0, n0, n1]) ++ " pan " ++ show x ++ ";"
        | ((r0, n0, n1), (_, _, Just x)) <- M.toList toUpdate
        ]
      packet = deletes ++ adds ++ updateTargets ++ updatePans ++ updateScales
  hPutStr (aPdSend a) packet
  hFlush (aPdSend a)
```

28 graphgrow3/audio/Makefile

```
all: graphgrow-audio graphgrow-audio-test

clean:
	-rm graphgrow-audio graphgrow-audio-test

graphgrow-audio: graphgrow-audio.cc
	g++ -std=c++11 -Wall -Wextra -pedantic -Wno-variadic-macros -O3 -march=native -ffast-math -I$(HOME)/opt/include -L$(HOME)/opt/lib -o graphgrow-audio graphgrow-audio.cc -ljack -llo

graphgrow-audio-test: graphgrow-audio-test.cc
	g++ -std=c++11 -Wall -Wextra -pedantic -Wno-variadic-macros -O3 -march=native -ffast-math -I$(HOME)/opt/include -L$(HOME)/opt/lib -o graphgrow-audio-test graphgrow-audio-test.cc -llo
```

29 graphgrow3/audio/README

```
Needs liblo from https://github.com/radarsat1/liblo
Tested with HEAD at 276dc11db2038b44f04fe67cdb425929427a5337
```

30 graphgrow3/colour.frag

```
#version 130
uniform sampler2DArray src;
uniform float speed;

in vec2 texCoord;

out vec4 colour;

void main() {
  vec2 p = texCoord;
  vec2 z = texture(src, vec3(p, 0.0)).xy;
  float n = z.x * speed;
  vec3 yuv;
  if (n < 0.0) {
    yuv = vec3(0.0);
  } else {
    yuv = vec3(clamp(log(1.0 + n / 4.0), 0.0, 1.0), 0.125 * sin(n), -0.125 * cos(n));
  }
  vec3 rgb = yuv * mat3(1.0, 0.0, 1.4, 1.0, -0.395, -0.581, 1.0, 2.03, 0.0);
  colour = vec4(rgb, 1.0);
}
```

31 graphgrow3/colour.vert

```
#version 130

out vec2 texCoord;

void main() {
  float t = 0.5 * 500.0 / 2048.0;
  switch (gl_VertexID) {
    default:
    case 0:
      gl_Position = vec4(-1.0, -1.0, 0.0, 1.0);
      texCoord = vec2(0.5 - t, 0.5 + t);
      break;
    case 1:
      gl_Position = vec4( 1.0, -1.0, 0.0, 1.0);
      texCoord = vec2(0.5 + t, 0.5 + t);
      break;
    case 2:
      gl_Position = vec4(-1.0,  1.0, 0.0, 1.0);
      texCoord = vec2(0.5 - t, 0.5 - t);
      break;
    case 3:
      gl_Position = vec4( 1.0,  1.0, 0.0, 1.0);
      texCoord = vec2(0.5 + t, 0.5 - t);
      break;
  }
}
```

32 graphgrow3/compress~.pd

```
#N canvas 275 94 452 587 10;
#X obj 28 28 inlet~;
#X obj 27 543 outlet~;
#X obj 58 82 hip~ 5;
#X obj 58 108 *~;
#X obj 281 25 inlet;
#X obj 87 281 rmstodb~;
#X obj 86 353 dbtorms~;
#X obj 27 474 *~;
#X obj 57 135 lop~ 10;
#X obj 87 221 lop~ 25;
#X obj 25 272 /~;
#X obj 201 390 dbtorms;
#X obj 86 417 /~ 0;
#X obj 87 179 sqrt~;
#X obj 86 442 *~ 0.25;
#X obj 310 55 loadbang;
#X obj 310 79 f \$1;
#X obj 27 514 expr~ tanh($v1);
#X obj 86 314 expr~ if($v1>$f2 \, $f2+($v1-$f2)/8 \, $f2);
#X obj 201 366 expr (100-$f1)/8+$f1;
#X obj 88 202 +~ 0.01;
#X connect 0 0 2 0;
#X connect 0 0 10 0;
#X connect 2 0 3 0;
#X connect 2 0 3 1;
#X connect 3 0 8 0;
#X connect 4 0 18 1;
#X connect 4 0 19 0;
#X connect 5 0 18 0;
#X connect 6 0 12 0;
#X connect 7 0 17 0;
#X connect 8 0 13 0;
#X connect 9 0 10 1;
#X connect 9 0 5 0;
#X connect 10 0 7 0;
#X connect 11 0 12 1;
#X connect 12 0 14 0;
#X connect 13 0 20 0;
#X connect 14 0 7 1;
#X connect 15 0 16 0;
#X connect 16 0 18 1;
#X connect 16 0 19 0;
#X connect 17 0 1 0;
#X connect 18 0 6 0;
#X connect 19 0 11 0;
#X connect 20 0 9 0;
```

33 graphgrow3/Dimension.hs

```
-- |
-- Compute Hausdorff dimension of a graph directed iterated function system of
-- similarities specified by adjacency matrix, assuming the open set condition
-- is satisfied.  For example:
--
-- > -- Koch snowflake
-- > > dimH (listArray ((0,0),(0,0)) [[1/3,1/3,1/3,1/3]])
```

```
-- > Just 1.2618595361709595
--
-- > -- Sierpinski triangle
-- > > dimH (listArray ((0,0),(0,0)) [[1/2,1/2,1/2]])
-- > Just 1.5849626064300537
--
-- > -- Koch + Sierpinski hybrid
-- > > dimH (listArray ((0,0),(1,1)) ↙
    ↳ [[1/3,1/3,1/3,1/3],[1/2],[1/3],[1/2,1/2,1/2]])
-- > Just 1.7465347051620483

module Dimension (dimH) where

import Data.Array (Array, (!), (//), bounds, indices, listArray)

import Data.Graph.Inductive.PatriciaTree (Gr)
import Data.Graph.Inductive.Query.DFS (scc)
import Data.Graph.Inductive.Graph (mkGraph, subgraph, noNodes, labNodes, ↙
    ↳ labEdges)

-- compute dimension as maximum over components
dimH :: Array (Int, Int) [Double] -> Maybe Double
dimH = maximum . (Nothing:) . map dimSCC . components

-- precondition: m is square
components :: Array (Int, Int) [Double] -> [Array (Int, Int) [Double]]
components m = map fromGraph $ map (`subgraph` g) (scc g)
  where
    g = toGraph m

toGraph :: Array (Int, Int) [Double] -> Gr Int [Double]
toGraph m = mkGraph
    [(i, i) | i <- [0..n]]
    [ (i, j, m ! (i, j)) | (i, j) <- indices m, not (null (m ! (i, j))) ]
  where
    ((0, 0), (n, _n)) = bounds m

fromGraph :: Gr Int [Double] -> Array (Int, Int) [Double]
fromGraph g = listArray ((0, 0), (n, n)) (replicate ((n + 1) * (n + 1)) []) //
    [ ((i',j'), l)
    | (i, j, l) <- es, Just i' <- [lookup i ns], Just j' <- [lookup j ns]
    ]
  where
    n = noNodes g - 1
    ns = map fst (labNodes g) `zip` [0..]
    es = labEdges g

-- precondition: g corresponds to a strongly connected component
dimSCC :: Array (Int, Int) [Double] -> Maybe Double
dimSCC g = findZero f 0 10
  where
   f s = fmap (subtract 1) $ spectralRadius (fmap (sum . map (** s)) g)

-- precondition: m is square
spectralRadius :: Array (Int, Int) Double -> Maybe Double
spectralRadius m = go 0 1 0 ones
  where
```

```
    ((0, 0), (m1,_m1)) = bounds m
    ones = listArray (0, m1) (replicate (m1 + 1) 1)
    go :: Int -> Double -> Double -> Array Int Double -> Maybe Double
    go n h r v
      | n > 1000 = Nothing
      | abs h < 0.00001 = Just r
      | otherwise = go (n + 1) h' r' v'
      where
        h' = r' - r
        r' = dot v' v / dot v v
        v' = mul m v

-- precondition: a and b have the same bounds
dot :: Array Int Double -> Array Int Double -> Double
dot a b = sum [ a ! i * b ! i | i <- indices a ]

-- precondition: m is square matching the dimensions of v
mul :: Array (Int, Int) Double -> Array Int Double -> Array Int Double
mul m v = listArray (bounds v)
  [ sum [ m ! (i, j) * v ! j | j <- indices v ] | i <- indices v ]

-- precondition: f is continuous and monotonic with a zero between x1 and x2
findZero :: (Double -> Maybe Double) -> Double -> Double -> Maybe Double
findZero f x1' x2' = do
  y1 <- f x1'
  y2 <- f x2'
  case () of
    _ | zero y1 -> Just x1'
      | zero y2 -> Just x2'
      | y1 > 0 && y2 > 0 -> Nothing
      | y1 < 0 && y2 < 0 -> Nothing
      | otherwise -> go x1' y1 x2' y2
  where
    zero x = abs x < 0.0000001
    go x1 y1 x2 y2 = do
      let x = (x1 + x2) / 2
      y <- f x
      case () of
        _ | zero y -> Just x
          | (y2 > 0 && y > 0) || (y2 < 0 && y < 0) -> go x1 y1 x y
          | (y2 > 0 && y < 0) || (y2 < 0 && y > 0) -> go x y x2 y2
          | otherwise -> Nothing
```

34 graphgrow3/edge~.pd

```
#N canvas 3 93 450 358 10;
#X obj 40 198 pitchshift~;
#X obj 178 44 r EDGE;
#X obj 178 66 route \$1;
#X obj 178 88 route \$2;
#X obj 178 110 route \$3;
#X obj 113 174 vline~;
#X obj 114 129 f;
#X obj 228 179 f;
#X obj 40 243 *~;
#X obj 72 244 vline~;
#X obj 114 154 expr -12*log(\$f1/0.5)/log(2);
```

```
#X obj 39 314 throw~ RULE-\$4-l;
#X obj 163 314 throw~ RULE-\$4-r;
#X obj 38 287 *~;
#X obj 164 291 *~;
#X obj 251 251 expr 3.141592653/2*$f1;
#X obj 251 274 expr cos($f1) \; sin($f1);
#X obj 66 287 vline~;
#X obj 197 290 vline~;
#X obj 41 43 r~ RULE-\$1-l;
#X obj 52 71 r~ RULE-\$1-r;
#X obj 50 218 pitchshift~;
#X obj 52 264 *~;
#X msg 229 200 set RULE-\$1-l;
#X msg 238 220 set RULE-\$1-r;
#X msg 320 154 \; \$1 bang;
#X obj 178 132 route scale target pan echo active;
#X obj 360 201 switch~;
#X obj 382 166 f;
#X obj 371 81 loadbang;
#X msg 371 103 1;
#X connect 0 0 8 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 26 0;
#X connect 5 0 0 1;
#X connect 5 0 21 1;
#X connect 6 0 9 0;
#X connect 6 0 10 0;
#X connect 7 0 23 0;
#X connect 7 0 24 0;
#X connect 8 0 13 0;
#X connect 9 0 8 1;
#X connect 9 0 22 1;
#X connect 10 0 5 0;
#X connect 13 0 11 0;
#X connect 14 0 12 0;
#X connect 15 0 16 0;
#X connect 16 0 17 0;
#X connect 16 1 18 0;
#X connect 17 0 13 1;
#X connect 18 0 14 1;
#X connect 19 0 0 0;
#X connect 20 0 21 0;
#X connect 21 0 22 0;
#X connect 22 0 14 0;
#X connect 23 0 11 0;
#X connect 24 0 12 0;
#X connect 26 0 6 0;
#X connect 26 1 7 0;
#X connect 26 2 15 0;
#X connect 26 3 25 0;
#X connect 26 4 28 0;
#X connect 28 0 27 0;
#X connect 29 0 30 0;
#X connect 30 0 28 0;
```

35 graphgrow3/graphgrow3.cabal

```
name:                graphgrow3
version:             3
synopsis:            graph-directed iterated function system fractals
description:
  graphgrow3 for graph-directed iterated function system fractals.
  .
  GraphGrow3 has a graphical node editor, visualization with
  OpenGL and sonification via messages to Pure-data.
  .
  > pd graphgrow3.pd &
  > graphgrow3 | pdsend 6060

homepage:            https://mathr.co.uk/blog/graphgrow.html
license:             GPL-3
license-file:        LICENSE
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
category:            Graphics
build-type:          Simple
cabal-version:       >=1.8
extra-doc-files:     README.md
data-files:
  initial.vert
  initial.frag
  colour.vert
  colour.frag
  step.vert
  step.frag
  graphgrow3.pd
  rule~.pd
  edge~.pd
  compress~.pd
  pitchshift~.pd

executable graphgrow3
  main-is:  Main.hs
  other-modules:
    Types
    Audio
    Video
    Dimension
  build-depends:
    base >= 4.9 && < 4.10,
    array >= 0.5 && < 0.6,
    colour >= 2.3 && < 2.4,
    containers >= 0.5.8 && < 0.6,
    fgl >= 5.5 && < 5.6,
    glib >= 0.13 && < 0.14,
    gtk >= 0.14 && < 0.15,
    gtkglext >= 0.13 && < 0.14,
    lens >= 4.15 && < 4.16,
    linear >= 1.20 && < 1.21,
    mtl >= 2.2 && < 2.3,
    OpenGLRaw >= 3.2 && < 3.3,
    process >= 1.4.2 && < 1.5,
```

```
    toy-gtk-diagrams >= 0.4 && < 0.5
  ghc-options: -Wall -threaded

Source-repository head
  type:                git
  location:            https://code.mathr.co.uk/graphgrow.git
  subdir:              graphgrow3

Source-repository this
  type:                git
  location:            https://code.mathr.co.uk/graphgrow.git
  subdir:              graphgrow3
  tag:                 v3
```

36 graphgrow3/graphgrow3.pd

```
#N canvas 24 92 520 592 10;
#N canvas 3 143 450 300 RULES 0;
#X obj 41 35 rule~ 0;
#X obj 41 57 rule~ 1;
#X obj 41 79 rule~ 2;
#X obj 41 101 rule~ 3;
#X obj 41 123 rule~ 4;
#X obj 41 145 rule~ 5;
#X obj 41 167 rule~ 6;
#X obj 41 189 rule~ 7;
#X restore 32 139 pd RULES;
#X obj 61 283 dac~;
#N canvas 3 143 450 300 EDGES 0;
#X restore 32 112 pd EDGES;
#X obj 215 51 netreceive 6060;
#X obj 278 401 list prepend obj 10 10 edge~;
#X obj 278 423 list trim;
#X obj 298 530 s pd-EDGES;
#X obj 278 445 t b b a b;
#X msg 295 467 loadbang;
#X msg 338 491 \; pd dsp 0;
#X msg 268 489 \; pd dsp 1;
#X msg 420 466 clear;
#X obj 33 205 hip~ 5;
#X obj 33 165 catch~ OUT-l;
#X obj 129 187 hip~ 5;
#X obj 129 165 catch~ OUT-r;
#X obj 41 318 env~ 16384;
#X floatatom 41 340 5 0 0 0 - - -, f 5;
#X obj 120 318 env~ 16384;
#X floatatom 120 340 5 0 0 0 - - -, f 5;
#X obj 259 267 s EDGE;
#X obj 311 269 r ECHO;
#X msg 312 295 0;
#X obj 216 303 list prepend;
#X obj 216 202 t a a b;
#X msg 296 221 1;
#X obj 216 352 route 0 1;
#X obj 216 325 list trim;
#X msg 216 374 \$1 \$2 \$3 active 1 \, \$1 \$2 \$3 target \$4;
#X obj 216 396 s EDGE;
```

```
#X obj 215 123 route add delete set;
#X text 219 143 ffff;
#X text 274 142 fff;
#X msg 277 174 \$1 \$2 \$3 active 0;
#X obj 277 196 s EDGE;
#X text 321 143 fffsf;
#X obj 336 196 s EDGE;
#X msg 259 245 \$1 \$2 \$3 echo symbol ECHO;
#X obj 35 487 writesf~ 2;
#X msg 54 453 stop;
#X obj 35 24 loadbang;
#X msg 35 70 \; pd dsp 1;
#X msg 18 429 open -bytes 4 audio.wav \, start;
#X obj 33 231 compress~ 48;
#X obj 128 230 compress~ 48;
#X obj 116 257 *~ 1;
#X obj 34 254 *~ 1;
#N canvas 529 88 593 502 RANDOMIZE 0;
#X obj 87 11 inlet;
#X obj 96 73 random 8;
#X obj 96 95 + 1;
#X msg 164 72 clear;
#X obj 164 94 s pd-EDGES;
#X msg 245 71 \; pd dsp 0;
#X obj 96 175 * 3;
#X obj 96 197 until;
#X obj 84 289 random;
#X obj 98 379 pack f f f f f f;
#X obj 237 285 random;
#X obj 176 285 f 0;
#X obj 204 285 + 1;
#X msg 140 290 0;
#X obj 96 219 t b b b b b b;
#X obj 297 300 t f f;
#X obj 297 322 *;
#X obj 446 301 - 0.5;
#X obj 446 323 * 0.5;
#X obj 446 345 + 0.5;
#X obj 96 114 t f b f;
#X obj 114 157 + 2;
#X obj 114 135 random 3;
#X msg 159 187 0;
#X msg 94 406 add \$1 \$2 \$3 \$4 \, set \$1 \$2 \$3 scale \$5 \, set
\$1 \$2 \$3 pan \$6;
#X obj 91 437 outlet;
#X msg 30 77 \; pd dsp 1;
#X obj 298 256 random 1e+06;
#X obj 297 278 / 1e+06;
#X obj 448 257 random 1e+06;
#X obj 446 279 / 1e+06;
#X obj 87 33 t b b b b b;
#X msg 425 65 ----------;
#X connect 0 0 31 0;
#X connect 1 0 2 0;
#X connect 2 0 20 0;
#X connect 3 0 4 0;
#X connect 6 0 7 0;
```

```
#X connect 7 0 14 0;
#X connect 8 0 9 0;
#X connect 9 0 24 0;
#X connect 10 0 9 3;
#X connect 11 0 12 0;
#X connect 11 0 9 2;
#X connect 12 0 11 1;
#X connect 13 0 9 1;
#X connect 14 0 8 0;
#X connect 14 1 13 0;
#X connect 14 2 11 0;
#X connect 14 3 10 0;
#X connect 14 4 27 0;
#X connect 14 5 29 0;
#X connect 15 0 16 0;
#X connect 15 1 16 1;
#X connect 16 0 9 4;
#X connect 17 0 18 0;
#X connect 18 0 19 0;
#X connect 19 0 9 5;
#X connect 20 0 6 0;
#X connect 20 1 22 0;
#X connect 20 1 23 0;
#X connect 20 2 8 1;
#X connect 20 2 10 1;
#X connect 21 0 6 1;
#X connect 22 0 21 0;
#X connect 23 0 11 1;
#X connect 24 0 25 0;
#X connect 27 0 28 0;
#X connect 28 0 15 0;
#X connect 29 0 30 0;
#X connect 30 0 17 0;
#X connect 31 0 26 0;
#X connect 31 1 1 0;
#X connect 31 2 3 0;
#X connect 31 3 5 0;
#X connect 31 4 32 0;
#X connect 32 0 25 0;
#X restore 380 93 pd RANDOMIZE;
#X obj 382 61 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 399 122 print -n;
#X obj 35 46 delay 1;
#X connect 3 0 30 0;
#X connect 4 0 5 0;
#X connect 5 0 7 0;
#X connect 7 0 10 0;
#X connect 7 1 8 0;
#X connect 7 2 6 0;
#X connect 7 3 9 0;
#X connect 8 0 6 0;
#X connect 11 0 6 0;
#X connect 12 0 43 0;
#X connect 13 0 12 0;
#X connect 14 0 44 0;
#X connect 15 0 14 0;
```

```
#X connect 16 0 17 0;
#X connect 18 0 19 0;
#X connect 21 0 22 0;
#X connect 22 0 23 1;
#X connect 23 0 27 0;
#X connect 24 0 23 0;
#X connect 24 1 37 0;
#X connect 24 2 25 0;
#X connect 25 0 23 1;
#X connect 26 0 28 0;
#X connect 26 1 4 0;
#X connect 27 0 26 0;
#X connect 28 0 29 0;
#X connect 30 0 24 0;
#X connect 30 1 33 0;
#X connect 30 2 36 0;
#X connect 33 0 34 0;
#X connect 37 0 20 0;
#X connect 39 0 38 0;
#X connect 40 0 50 0;
#X connect 42 0 38 0;
#X connect 43 0 46 0;
#X connect 44 0 45 0;
#X connect 45 0 18 0;
#X connect 45 0 1 1;
#X connect 45 0 38 1;
#X connect 46 0 1 0;
#X connect 46 0 16 0;
#X connect 46 0 38 0;
#X connect 47 0 30 0;
#X connect 47 0 49 0;
#X connect 48 0 47 0;
#X connect 50 0 41 0;
```

37 graphgrow3/iface/deselected.png

38 graphgrow3/iface/.gitignore

```
graphgrow-iface
*.raw
```

39 graphgrow3/iface/graphgrow-iface.cc

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <vector>
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include <lo/lo.h>

// ------------------------------ declarations -------------------------

struct video_control;
struct audio_control;

enum graph_mode
{
  mode_create_node,
  mode_delete_node,
  mode_move_node,
  mode_create_link,
  mode_delete_link,
  mode_flip_link,
  mode_target_link,
  mode_help
};

struct graph_node;
struct graph_link;
struct graph_arrow;
struct graph_cord;
struct graph_designer;

typedef bool (*node_mouse_t)(graph_node *, double, double);
typedef bool (*link_mouse_t)(graph_link *, double, double);
typedef bool (*arrow_mouse_t)(graph_arrow *, double, double);
typedef bool (*cord_mouse_t)(graph_cord *, double, double);
typedef bool (*designer_mouse_t)(graph_designer *, double, double);

graph_link *link_new(graph_designer *designer, graph_node *from, graph_node *to)↙
    ↳ ;
void link_delete(graph_link *link);
void link_leave_mode(graph_link *link);
void link_enter_mode(graph_link *link, graph_mode mode);
void link_update(graph_link *link);
graph_arrow *arrow_new();
void arrow_delete(graph_arrow *arrow);
void arrow_leave_mode(graph_arrow *arrow);
void arrow_enter_mode(graph_arrow *arrow, graph_mode mode);
graph_cord *cord_new(double x1, double y1, double x2, double y2);
```

```
    void cord_delete(graph_cord *cord);
50  graph_node *node_new(graph_designer *designer, double x, double y, bool fixed);
    void node_delete(graph_node *node);
    bool nodes_linked(graph_node *a, graph_node *b);
    void node_leave_mode(graph_node *node);
    bool node_mouse_hit(graph_node *node, double x, double y);
55  bool node_mouse_down_new(graph_node *node, double x, double y);
    bool node_mouse_down_delete(graph_node *node, double x, double y);
    bool node_mouse_down_move(graph_node *node, double x, double y);
    bool node_mouse_down_link(graph_node *node, double x, double y);
    bool node_mouse_up_corded(graph_node *node, double x, double y);
60  void node_enter_mode(graph_node *node, graph_mode mode);
    graph_designer *designer_new(int id);
    void designer_leave_mode(graph_designer *designer);
    void designer_enter_mode(graph_designer *designer, graph_mode mode);
    void node_move_by(graph_node *node, double dx, double dy);
65  bool designer_mouse_down(graph_designer *designer, double x, double y);
    bool designer_mouse_down_node_new(graph_designer *designer, double x, double y);
    bool designer_mouse_move_dragging(graph_designer *designer, double x, double y);
    bool designer_mouse_move_dragged(graph_designer *designer, double x, double y);
    bool designer_mouse_move_cording(graph_designer *designer, double x, double y);
70  bool designer_mouse_move_corded(graph_designer *designer, double x, double y);

    // ----------------------------- constants -----------------------------

    const glm::vec3 white = glm::vec3(1.0f);
75  const glm::vec3 black = glm::vec3(0.0f);
    const glm::vec3 colours[4] =
      { glm::vec3(1.0f, 0.5f, 0.0f)
      , glm::vec3(0.5f, 1.0f, 0.0f)
      , glm::vec3(0.0f, 0.5f, 1.0f)
80    , glm::vec3(0.5f, 0.0f, 1.0f)
      };

    const double minimum_x = 50;
    const double minimum_y = 50;
85  const double maximum_x = 750;
    const double maximum_y = 750;

    // ----------------------------- controls -----------------------------

90  struct video_control
    {
      int32_t active;
      int32_t count    [4];
      int32_t source   [4][8];
95    float   scale    [4][8];
      float   transform[4][8][3][3];
    };

    struct audio_control
100 {
      float   scale[4][4][8];
      float   pan  [4][4][8];
      uint8_t level[4][4][8];
    };
105
```

```
// ------------------------------ structs -------------------------------

struct graph_link
{
  graph_designer *designer;
  graph_node *from, *to;
  double x1, y1, x2, y2;
  int target;
  bool flipped;
  graph_arrow *arrow;
  link_mouse_t on_mouse_down, on_mouse_move, on_mouse_up;
};

struct graph_arrow
{
  graph_link *link;
  double x1, y1, x2, y2, x3, y3;
  arrow_mouse_t on_mouse_down, on_mouse_move, on_mouse_up;
};

struct graph_cord
{
  double x1, y1, x2, y2;
};

struct graph_node
{
  graph_designer *designer;
  std::vector< graph_link * > link1, link2;
  bool fixed;
  double x;
  double y;
  double dragx;
  double dragy;
  node_mouse_t on_mouse_down, on_mouse_move, on_mouse_up;
};

struct graph_designer
{
  int id;
  graph_node *start_node;
  graph_node *end_node;
  graph_node *dragging_node;
  graph_node *cording_node;
  graph_cord *cord;
  std::vector< graph_link  * >  links;
  std::vector< graph_arrow * > arrows;
  std::vector< graph_node  * >  nodes;
  designer_mouse_t on_mouse_down, on_mouse_move, on_mouse_up;
};

struct graph_draw
{
  int components;
  float tri[4096];
  GLuint vbo_tri;
  GLuint vao_tri;
```

```
  GLuint vao_quad;
  GLuint prog_circle;
  GLuint prog_arrow;
  GLuint prog_line;
  GLuint prog_buttons;
  GLuint prog_help;
  GLuint tex[3];
  GLint u_mode;
};

// ----------------------------- link ---------------------------------

graph_link *link_new(graph_designer *designer, graph_node *from, graph_node *to)
{
  graph_link *link = new graph_link();
  link->designer = designer;
  link->from = from;
  link->to = to;
  link->x1 = from->x;
  link->y1 = from->y;
  link->x2 = to->x;
  link->y2 = to->y;
  link->target = designer->id;
  link->arrow = new graph_arrow();
  link->arrow->link = link;
  from->link1.push_back(link);
  to->link2.push_back(link);
  designer->links.push_back(link);
  designer->arrows.push_back(link->arrow);
  link_update(link);
  return link;
}

void link_delete(graph_link *link)
{
  for (int i = 0; i < link->from->link1.size(); )
    if (link == link->from->link1[i])
      link->from->link1.erase(link->from->link1.begin() + i);
    else
      ++i;
  for (int i = 0; i < link->to->link2.size(); )
    if (link == link->to->link2[i])
      link->to->link2.erase(link->to->link2.begin() + i);
    else
      ++i;
  for (int i = 0; i < link->designer->links.size(); )
    if (link == link->designer->links[i])
      link->designer->links.erase(link->designer->links.begin() + i);
    else
      ++i;
  for (int i = 0; i < link->designer->arrows.size(); )
    if (link->arrow == link->designer->arrows[i])
      link->designer->arrows.erase(link->designer->arrows.begin() + i);
    else
      ++i;
  arrow_delete(link->arrow);
  link->arrow = 0;
```

```
220     link->from = 0;
        link->to = 0;
        link->designer = 0;
        delete link;
    }
225
    void link_leave_mode(graph_link *link)
    {
      (void) link;
    }
230
    void link_enter_mode(graph_link *link, graph_mode mode)
    {
      (void) link;
      (void) mode;
235 }

    void link_update(graph_link *link)
    {
      const double w = 30;
240   const double c = w * -0.5;
      const double s = w * 0.8660254037844387;
      double x1 = link->x1;
      double y1 = link->y1;
      double x2 = link->x2;
245   double y2 = link->y2;
      double dx = x2 - x1;
      double dy = y2 - y1;
      if (link->flipped)
      {
250     dx = -dx;
        dy = -dy;
      }
      double r = sqrt(dx * dx + dy * dy);
      dx /= r;
255   dy /= r;
      double ox = (x1 + x2) / 2;
      double oy = (y1 + y2) / 2;
      link->arrow->x1 = ox + w * dx;
      link->arrow->y1 = oy + w * dy;
260   link->arrow->x2 = ox + c * dx + s * dy;
      link->arrow->y2 = oy - s * dx + c * dy;
      link->arrow->x3 = ox + c * dx - s * dy;
      link->arrow->y3 = oy + s * dx + c * dy;
    }
265
    // ---------------------------- arrow --------------------------------

    graph_arrow *arrow_new()
    {
270   graph_arrow *arrow = new graph_arrow();
      arrow->on_mouse_down = 0;
      arrow->on_mouse_move = 0;
      arrow->on_mouse_up   = 0;
      return arrow;
275 }
```

```
void arrow_delete(graph_arrow *arrow)
{
  delete arrow;
}

void arrow_leave_mode(graph_arrow *arrow)
{
  arrow->on_mouse_down = 0;
  arrow->on_mouse_move = 0;
  arrow->on_mouse_up   = 0;
}

bool arrow_hit(graph_arrow *arrow, double x, double y)
{
  double x0 = (arrow->x1 + arrow->x2 + arrow->x3) / 3;
  double y0 = (arrow->y1 + arrow->y2 + arrow->y3) / 3;
  double dx = x - x0;
  double dy = y - y0;
  double r2 = dx * dx + dy * dy;
  return r2 < 500;
}

bool arrow_mouse_down_delete(graph_arrow *arrow, double x, double y)
{
  if (arrow_hit(arrow, x, y))
  {
    link_delete(arrow->link);
    // arrow is deleted
    return true;
  }
  return false;
}

bool arrow_mouse_down_flip(graph_arrow *arrow, double x, double y)
{
  if (arrow_hit(arrow, x, y))
  {
    arrow->link->flipped = !(arrow->link->flipped);
    link_update(arrow->link);
    return true;
  }
  return false;
}

bool arrow_mouse_down_target(graph_arrow *arrow, double x, double y)
{
  if (arrow_hit(arrow, x, y))
  {
    arrow->link->target = (arrow->link->target + 1) % 4;
    link_update(arrow->link);
    return true;
  }
  return false;
}

void arrow_enter_mode(graph_arrow *arrow, graph_mode mode)
{
```

```
  switch (mode)
  {
    case mode_delete_link:
      arrow->on_mouse_down = arrow_mouse_down_delete;
      break;
    case mode_flip_link:
      arrow->on_mouse_down = arrow_mouse_down_flip;
      break;
    case mode_target_link:
      arrow->on_mouse_down = arrow_mouse_down_target;
      break;
    case mode_create_link:
    case mode_create_node:
    case mode_delete_node:
    case mode_move_node:
    case mode_help:
      break;
  }
}

// ------------------------------ cord ---------------------------------

graph_cord *cord_new(double x1, double y1, double x2, double y2)
{
  graph_cord *cord = new graph_cord();
  cord->x1 = x1;
  cord->y1 = y1;
  cord->x2 = x2;
  cord->y2 = y2;
  return cord;
}

void cord_delete(graph_cord *cord)
{
  delete cord;
}

// ------------------------------ node ---------------------------------

graph_node *node_new(graph_designer *designer, double x, double y, bool fixed)
{
  graph_node *node = new graph_node();
  node->fixed = fixed;
  node->x = fmin(fmax(x, minimum_x), maximum_x);
  node->y = fmin(fmax(y, minimum_y), maximum_y);
  node->dragx = node->x;
  node->dragy = node->y;
  node->designer = designer;
  designer->nodes.push_back(node);
  return node;
}

void node_delete(graph_node *node)
{
  graph_designer *designer = node->designer;
  for (int i = 0; i < designer->nodes.size(); )
    if (node == designer->nodes[i])
```

```
        designer->nodes.erase(designer->nodes.begin() + i);
      else
        ++i;
  while (node->link1.size())
    link_delete(node->link1[0]);
  while (node->link2.size())
    link_delete(node->link2[0]);
  node->designer = 0;
  delete node;
}

bool nodes_linked(graph_node *a, graph_node *b)
{
  for (int i = 0; i < a->link1.size(); ++i)
    if (a->link1[i]->to == b)
      return true;
  for (int i = 0; i < b->link1.size(); ++i)
    if (b->link1[i]->to == a)
      return true;
  return false;
}

void node_leave_mode(graph_node *node)
{
  node->on_mouse_down = 0;
  node->on_mouse_move = 0;
  node->on_mouse_up   = 0;
}

bool node_mouse_hit(graph_node *node, double x, double y)
{
  double dx = x - node->x;
  double dy = y - node->y;
  double r2 = dx * dx + dy * dy;
  return r2 < 500;
}

bool node_mouse_down_new(graph_node *node, double x, double y)
{
  return node_mouse_hit(node, x, y);
}

bool node_mouse_down_delete(graph_node *node, double x, double y)
{
  if (node_mouse_hit(node, x, y))
  {
    node_delete(node);
    return true;
  }
  return false;
}

bool node_mouse_down_move(graph_node *node, double x, double y)
{
  if (node_mouse_hit(node, x, y))
  {
    node->designer->dragging_node = node;
```

```
      node->designer->on_mouse_move = designer_mouse_move_dragging;
      node->designer->on_mouse_up   = designer_mouse_move_dragged;
      node->dragx = fmin(fmax(x, minimum_x), maximum_x);
      node->dragy = fmin(fmax(y, minimum_y), maximum_y);
      return true;
    }
    return false;
}

bool node_mouse_down_link(graph_node *node, double x, double y)
{
  if (node_mouse_hit(node, x, y) && node->designer->links.size() < 8)
  {
    node->designer->cording_node = node;
    node->designer->cord = cord_new(node->x, node->y, x, y);
    for (int i = 0; i < node->designer->nodes.size(); ++i)
      if (node != node->designer->nodes[i] && ! nodes_linked(node, node->designer->nodes[i]))
        node->designer->nodes[i]->on_mouse_up = node_mouse_up_corded;
    node->designer->on_mouse_move = designer_mouse_move_cording;
    node->designer->on_mouse_up   = designer_mouse_move_corded;
    return true;
  }
  return false;
}

bool node_mouse_up_corded(graph_node *node, double x, double y)
{
  if (node_mouse_hit(node, x, y))
  {
    link_new(node->designer, node->designer->cording_node, node);
    node->designer->cording_node = 0;
    delete node->designer->cord;
    node->designer->cord = 0;
    for (int i = 0; i < node->designer->nodes.size(); ++i)
      node->designer->nodes[i]->on_mouse_up = 0;
    node->designer->on_mouse_move = 0;
    node->designer->on_mouse_up   = 0;
    return true;
  }
  return false;
}

void node_enter_mode(graph_node *node, graph_mode mode)
{
  switch (mode)
  {
    case mode_create_node:
      node->on_mouse_down = node_mouse_down_new;
      break;
    case mode_delete_node:
      if (! node->fixed)
        node->on_mouse_down = node_mouse_down_delete;
      break;
    case mode_move_node:
      if (! node->fixed)
        node->on_mouse_down = node_mouse_down_move;
```

```
        break;
      case mode_create_link:
        node->on_mouse_down = node_mouse_down_link;
        break;
      case mode_delete_link:
      case mode_flip_link:
      case mode_target_link:
      case mode_help:
        break;
    }
  }

// ------------------------------ designer ------------------------------

graph_designer *designer_new(int id)
{
  graph_designer *designer = new graph_designer();
  designer->id = id;
  designer->start_node = node_new(designer, 100, 400, true);
  designer->end_node   = node_new(designer, 700, 400, true);
  designer->dragging_node = 0;
  designer->cording_node = 0;
  designer->on_mouse_down = 0;
  designer->on_mouse_move = 0;
  designer->on_mouse_up   = 0;
  return designer;
}

void designer_leave_mode(graph_designer *designer)
{
  for (int i = 0; i < designer->links.size(); ++i)
    link_leave_mode(designer->links[i]);
  for (int i = 0; i < designer->arrows.size(); ++i)
    arrow_leave_mode(designer->arrows[i]);
  for (int i = 0; i < designer->nodes.size(); ++i)
    node_leave_mode(designer->nodes[i]);
  designer->on_mouse_down = 0;
  designer->on_mouse_move = 0;
  designer->on_mouse_up   = 0;
}

void designer_enter_mode(graph_designer *designer, graph_mode mode)
{
  for (int i = 0; i < designer->links.size(); ++i)
    link_enter_mode(designer->links[i], mode);
  for (int i = 0; i < designer->arrows.size(); ++i)
    arrow_enter_mode(designer->arrows[i], mode);
  for (int i = 0; i < designer->nodes.size(); ++i)
    node_enter_mode(designer->nodes[i], mode);
  switch (mode)
  {
    case mode_create_node:
      designer->on_mouse_down = designer_mouse_down_node_new;
      break;
    case mode_delete_node:
    case mode_move_node:
    case mode_create_link:
```

```
      case mode_delete_link:
      case mode_flip_link:
      case mode_target_link:
      case mode_help:
        break;
    }
}

void node_move_by(graph_node *node, double dx, double dy)
{
  for (int i = 0; i < node->link2.size(); ++i)
  {
    node->link2[i]->x2 = fmin(fmax(node->link2[i]->x2 + dx, minimum_x), ↙
        ↳ maximum_x);
    node->link2[i]->y2 = fmin(fmax(node->link2[i]->y2 + dy, minimum_y), ↙
        ↳ maximum_y);
    link_update(node->link2[i]);
  }
  for (int i = 0; i < node->link1.size(); ++i)
  {
    node->link1[i]->x1 = fmin(fmax(node->link1[i]->x1 + dx, minimum_x), ↙
        ↳ maximum_x);
    node->link1[i]->y1 = fmin(fmax(node->link1[i]->y1 + dy, minimum_y), ↙
        ↳ maximum_y);
    link_update(node->link1[i]);
  }
  node->x = fmin(fmax(node->x + dx, minimum_x), maximum_x);
  node->y = fmin(fmax(node->y + dy, minimum_y), maximum_y);
}

bool designer_mouse_down(graph_designer *designer, double x, double y)
{
  for (int i = 0; i < designer->nodes.size(); ++i)
    if (designer->nodes[i]->on_mouse_down)
      if (designer->nodes[i]->on_mouse_down(designer->nodes[i], x, y))
        return true;
  for (int i = 0; i < designer->arrows.size(); ++i)
    if (designer->arrows[i]->on_mouse_down)
      if (designer->arrows[i]->on_mouse_down(designer->arrows[i], x, y))
        return true;
  if (designer->on_mouse_down)
    return designer->on_mouse_down(designer, x, y);
  return false;
}

bool designer_mouse_move(graph_designer *designer, double x, double y)
{
  for (int i = 0; i < designer->nodes.size(); ++i)
    if (designer->nodes[i]->on_mouse_move)
      if (designer->nodes[i]->on_mouse_move(designer->nodes[i], x, y))
        return true;
  if (designer->on_mouse_move)
    return designer->on_mouse_move(designer, x, y);
  return false;
}

bool designer_mouse_up(graph_designer *designer, double x, double y)
```

```
{
  for (int i = 0; i < designer->nodes.size(); ++i)
    if (designer->nodes[i]->on_mouse_up)
      if (designer->nodes[i]->on_mouse_up(designer->nodes[i], x, y))
        return true;
  if (designer->on_mouse_up)
    return designer->on_mouse_up(designer, x, y);
  return false;
}

bool designer_mouse_down_node_new(graph_designer *designer, double x, double y)
{
  if (minimum_x <= x && x <= maximum_x && minimum_y <= y && y <= maximum_y && ↙
      ↳ designer->nodes.size() < 18)
  {
    graph_node *node = node_new(designer, x, y, false);
    node_enter_mode(node, mode_create_node);
  }
  return true;
}

bool designer_mouse_move_dragging(graph_designer *designer, double x, double y)
{
  double dx = fmin(fmax(x, minimum_x), maximum_x) - designer->dragging_node->↙
      ↳ dragx;
  double dy = fmin(fmax(y, minimum_y), maximum_y) - designer->dragging_node->↙
      ↳ dragy;
  designer->dragging_node->dragx = fmin(fmax(x, minimum_x), maximum_x);
  designer->dragging_node->dragy = fmin(fmax(y, minimum_y), maximum_y);
  node_move_by(designer->dragging_node, dx, dy);
  return true;
}

bool designer_mouse_move_dragged(graph_designer *designer, double x, double y)
{
  designer_mouse_move_dragging(designer, x, y);
  designer->on_mouse_move = 0;
  designer->on_mouse_up = 0;
  designer->dragging_node = 0;
  return true;
}

bool designer_mouse_move_cording(graph_designer *designer, double x, double y)
{
  designer->cord->x2 = fmin(fmax(x, minimum_x), maximum_x);
  designer->cord->y2 = fmin(fmax(y, minimum_y), maximum_y);
  return true;
}

bool designer_mouse_move_corded(graph_designer *designer, double x, double y)
{
  for (int i = 0; i < designer->nodes.size(); ++i)
    designer->nodes[i]->on_mouse_up = 0;
  designer->on_mouse_move = 0;
  designer->on_mouse_up = 0;
  designer->cording_node = 0;
  delete designer->cord;
```

```
  designer->cord = 0;
  return true;
  (void) x;
  (void) y;
}

// ----------------------------- drawing -----------------------------

int draw_enqueue_triangle(graph_draw *draw, int k, double x1, double y1, double x2, double y2, double x3, double y3, glm::vec3 colour)
{
  draw->tri[k++] = x1;
  draw->tri[k++] = y1;
  draw->tri[k++] = 1;
  draw->tri[k++] = 0;
  draw->tri[k++] = 0;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x2;
  draw->tri[k++] = y2;
  draw->tri[k++] = 0;
  draw->tri[k++] = 1;
  draw->tri[k++] = 0;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x3;
  draw->tri[k++] = y3;
  draw->tri[k++] = 0;
  draw->tri[k++] = 0;
  draw->tri[k++] = 1;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  return k;
}

int draw_enqueue_quad(graph_draw *draw, int k, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4, glm::vec3 colour)
{
  draw->tri[k++] = x1;
  draw->tri[k++] = y1;
  draw->tri[k++] = 0;
  draw->tri[k++] = 0;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x2;
  draw->tri[k++] = y2;
  draw->tri[k++] = 0;
  draw->tri[k++] = 1;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x3;
```

```
  draw->tri[k++] = y3;
  draw->tri[k++] = 1;
  draw->tri[k++] = 0;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x3;
  draw->tri[k++] = y3;
  draw->tri[k++] = 1;
  draw->tri[k++] = 0;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x2;
  draw->tri[k++] = y2;
  draw->tri[k++] = 0;
  draw->tri[k++] = 1;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  draw->tri[k++] = x4;
  draw->tri[k++] = y4;
  draw->tri[k++] = 1;
  draw->tri[k++] = 1;
  draw->tri[k++] = colour[0];
  draw->tri[k++] = colour[1];
  draw->tri[k++] = colour[2];
  return k;
}


int draw_enqueue_line(graph_draw *draw, int k, double x1, double y1, double x2, ↙
    ↳ double y2, double w, glm::vec3 colour)
{
  double dx = x2 - x1;
  double dy = y2 - y1;
  double r = sqrt(dx * dx + dy * dy);
  dx /= r;
  dy /= r;
  k = draw_enqueue_quad(draw, k, x1 - w * dy, y1 + w * dx, x2 - w * dy, y2 + w *↙
      ↳  dx, x1 + w * dy, y1 - w * dx, x2 + w * dy, y2 - w * dx, colour);
  return k;
}

int draw_enqueue_circle(graph_draw *draw, int k, double x, double y, double w, ↙
    ↳ glm::vec3 colour)
{
  k = draw_enqueue_quad(draw, k, x - w, y - w, x + w, y - w, x - w, y + w, x + w↙
      ↳ , y + w, colour);
  return k;
}

void draw_designer(graph_draw *draw, graph_designer *designer, bool is_active, ↙
    ↳ bool big)
{
  glm::vec3 bg = glm::mix(colours[designer->id], black, 0.75f);
  if (! is_active)
```

```
775       bg = glm::mix(bg, black, 0.5f);
        glClearColor(bg[0], bg[1], bg[2], 0);
        glClear(GL_COLOR_BUFFER_BIT);
        int k = 0;
        for (int i = 0; i < designer->links.size() && k <= draw->components - 7 * 6; ↙
            ↳ ++i)
780     {
          double w = big ? 8 : 16;
          double x1 = designer->links[i]->x1;
          double y1 = designer->links[i]->y1;
          double x2 = designer->links[i]->x2;
785       double y2 = designer->links[i]->y2;
          glm::vec3 colour = colours[designer->links[i]->target];
          k = draw_enqueue_line(draw, k, x1, y1, x2, y2, w, colour);
        }
        if (designer->cord)
790     {
          double w = big ? 8 : 16;
          double x1 = designer->cord->x1;
          double y1 = designer->cord->y1;
          double x2 = designer->cord->x2;
795       double y2 = designer->cord->y2;
          k = draw_enqueue_line(draw, k, x1, y1, x2, y2, w, colours[designer->id]);
        }
        glBufferSubData(GL_ARRAY_BUFFER, 0, k * sizeof(float), &draw->tri[0]);
        glBindVertexArray(draw->vao_quad);
800     glUseProgram(draw->prog_line);
        glDrawArrays(GL_TRIANGLES, 0, k / 7);
        k = 0;
        for (int i = 0; i < designer->arrows.size() && k < draw->components - 8 * 3; ↙
            ↳ ++i)
        {
805       double x1 = designer->arrows[i]->x1;
          double y1 = designer->arrows[i]->y1;
          double x2 = designer->arrows[i]->x2;
          double y2 = designer->arrows[i]->y2;
          double x3 = designer->arrows[i]->x3;
810       double y3 = designer->arrows[i]->y3;
          glm::vec3 colour = colours[designer->arrows[i]->link->target];
          k = draw_enqueue_triangle(draw, k, x1, y1, x2, y2, x3, y3, colour);
        }
        glBufferSubData(GL_ARRAY_BUFFER, 0, k * sizeof(float), &draw->tri[0]);
815     glBindVertexArray(draw->vao_tri);
        glUseProgram(draw->prog_arrow);
        glDrawArrays(GL_TRIANGLES, 0, k / 8);
        k = 0;
        for (int i = 0; i < designer->nodes.size() && k <= draw->components - 7 * 6; ↙
            ↳ ++i)
820     {
          const double w = 20;
          double x = designer->nodes[i]->x;
          double y = designer->nodes[i]->y;
          glm::vec3 colour = colours[designer->nodes[i]->designer->id];
825       if (designer->nodes[i]->fixed)
            colour = white;
          k = draw_enqueue_circle(draw, k, x, y, w, colour);
        }
```

```
  glBufferSubData(GL_ARRAY_BUFFER, 0, k * sizeof(float), &draw->tri[0]);
  glBindVertexArray(draw->vao_quad);
  glUseProgram(draw->prog_circle);
  glDrawArrays(GL_TRIANGLES, 0, k / 7);
}

void draw_buttons(graph_draw *draw, glm::vec3 colour, graph_mode mode)
{
  glm::vec3 bg = glm::mix(colour, black, 0.75f);
  glClearColor(bg[0], bg[1], bg[2], 0);
  glClear(GL_COLOR_BUFFER_BIT);
  int k = 0;
  k = draw_enqueue_quad(draw, k, -1, -1, 1, -1, -1, 1, 1, 1, colour);
  glBufferSubData(GL_ARRAY_BUFFER, 0, k * sizeof(float), &draw->tri[0]);
  glBindVertexArray(draw->vao_quad);
  glUseProgram(draw->prog_buttons);
  glUniform1i(draw->u_mode, mode);
  glDrawArrays(GL_TRIANGLES, 0, k / 7);
}

void draw_help(graph_draw *draw, glm::vec3 colour)
{
  glm::vec3 bg = glm::mix(colour, black, 0.75f);
  glClearColor(bg[0], bg[1], bg[2], 0);
  glClear(GL_COLOR_BUFFER_BIT);
  int k = 0;
  k = draw_enqueue_quad(draw, k, -1, -1, 1, -1, -1, 1, 1, 1, colour);
  glBufferSubData(GL_ARRAY_BUFFER, 0, k * sizeof(float), &draw->tri[0]);
  glBindVertexArray(draw->vao_quad);
  glUseProgram(draw->prog_help);
  glDrawArrays(GL_TRIANGLES, 0, k / 7);
}

void debug_program(GLuint program) {
  GLint status = 0;
  glGetProgramiv(program, GL_LINK_STATUS, &status);
  GLint length = 0;
  glGetProgramiv(program, GL_INFO_LOG_LENGTH, &length);
  char *info = 0;
  if (length) {
    info = (char *)malloc(length + 1);
    info[0] = 0;
    glGetProgramInfoLog(program, length, 0, info);
    info[length] = 0;
  }
  if ((info && info[0]) || ! status) {
    fprintf(stderr, "program link info:\n%s", info ? info : "(no info log)");
  }
  if (info) {
    free(info);
  }
}

void debug_shader(GLuint shader, GLenum type, const char *source) {
  GLint status = 0;
  glGetShaderiv(shader, GL_COMPILE_STATUS, &status);
  GLint length = 0;
```

```
  glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &length);
  char *info = 0;
  if (length) {
    info = (char *)malloc(length + 1);
    info[0] = 0;
    glGetShaderInfoLog(shader, length, 0, info);
    info[length] = 0;
  }
  if ((info && info[0]) || ! status) {
    const char *type_str = "unknown";
    switch (type) {
      case GL_VERTEX_SHADER: type_str = "vertex"; break;
      case GL_FRAGMENT_SHADER: type_str = "fragment"; break;
      case GL_COMPUTE_SHADER: type_str = "compute"; break;
    }
    fprintf(stderr, "%s shader compile info:\n%s\nshader source:\n%s", type_str,↙
        ↪ info ? info : "(no info log)", source ? source : "(no source)");
  }
  if (info) {
    free(info);
  }
}

GLuint vertex_fragment_shader(const char *vert, const char *frag) {
  GLuint program = glCreateProgram();
  {
    GLuint shader = glCreateShader(GL_VERTEX_SHADER);
    glShaderSource(shader, 1, &vert, 0);
    glCompileShader(shader);
    debug_shader(shader, GL_VERTEX_SHADER, vert);
    glAttachShader(program, shader);
    glDeleteShader(shader);
  }
  {
    GLuint shader = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(shader, 1, &frag, 0);
    glCompileShader(shader);
    debug_shader(shader, GL_FRAGMENT_SHADER, frag);
    glAttachShader(program, shader);
    glDeleteShader(shader);
  }
  glLinkProgram(program);
  debug_program(program);
  return program;
}

const char *draw_circle_vert =
"#version 330 core\n"
"layout (location = 0) in vec2 pos;\n"
"layout (location = 1) in vec2 tcd;\n"
"layout (location = 2) in vec3 clr;\n"
"out vec2 v_texcoord;\n"
"out vec3 v_colour;\n"
"void main() {\n"
"  gl_Position = vec4(2.0/800.0 * pos - vec2(1.0,1.0), 0.0, 1.0);\n"
"  v_texcoord = tcd * 2.0 - vec2(1.0, 1.0);\n"
"  v_colour = clr;\n"
```

```
    "}\n"
    ;

945 const char *draw_circle_frag =
    "#version 330 core\n"
    "in vec2 v_texcoord;\n"
    "in vec3 v_colour;\n"
    "out vec4 colour;\n"
950 "void main() {\n"
    "  float r = length(v_texcoord);\n"
    "  float d = mix(length(dFdx(v_texcoord)), length(dFdy(v_texcoord)), 0.5);\n"
    "  colour = vec4(v_colour * (smoothstep(0.2, 0.2 + d, r) - smoothstep(0.8 - d, ↙
        ↳ 0.8, r)), 1.0 - smoothstep(1.0 - d, 1.0, r));\n"
    "}\n"
955 ;

    const char *draw_arrow_vert =
    "#version 330 core\n"
    "layout (location = 0) in vec2 pos;\n"
960 "layout (location = 1) in vec3 tcd;\n"
    "layout (location = 2) in vec3 clr;\n"
    "out vec3 v_texcoord;\n"
    "out vec3 v_colour;\n"
    "void main() {\n"
965 "  gl_Position = vec4(2.0/800.0 * pos - vec2(1.0,1.0), 0.0, 1.0);\n"
    "  v_texcoord = tcd;\n"
    "  v_colour = clr;\n"
    "}\n"
    ;
970
    const char *draw_arrow_frag =
    "#version 330 core\n"
    "in vec3 v_texcoord;\n"
    "in vec3 v_colour;\n"
975 "out vec4 colour;\n"
    "void main() {\n"
    "  vec3 w = fwidth(v_texcoord);\n"
    "  vec3 a = smoothstep(vec3(0.1), vec3(0.1) + w, v_texcoord);\n"
    "  vec3 b = smoothstep(vec3(0.0), vec3(0.0) + w, v_texcoord);\n"
980 "  float e1 = max(min(min(a.x, a.y), a.z), 1.0 - smoothstep(0.05, 0.05 + length(↙
        ↳ w), abs(v_texcoord.y - v_texcoord.z)));\n"
    "  float e2 = min(min(b.x, b.y), b.z);\n"
    "  colour = vec4(v_colour * e1, e2);\n"
    "}\n"
    ;
985
    const char *draw_line_vert =
    "#version 330 core\n"
    "layout (location = 0) in vec2 pos;\n"
    "layout (location = 1) in vec2 tcd;\n"
990 "layout (location = 2) in vec3 clr;\n"
    "out vec2 v_texcoord;\n"
    "out vec3 v_colour;\n"
    "void main() {\n"
    "  gl_Position = vec4(2.0/800.0 * pos - vec2(1.0,1.0), 0.0, 1.0);\n"
995 "  v_texcoord = tcd * 2.0 - vec2(1.0, 1.0);\n"
    "  v_colour = clr;\n"
```

```
    "}\n"
    ;

1000 const char *draw_line_frag =
    "#version 330 core\n"
    "in vec2 v_texcoord;\n"
    "in vec3 v_colour;\n"
    "out vec4 colour;\n"
1005 "void main() {\n"
    "  float r = abs(v_texcoord.x);\n"
    "  float d = abs(dFdx(v_texcoord.x)) + abs(dFdy(v_texcoord.x));\n"
    "  colour = vec4(v_colour * (1.0 - smoothstep(0.333 - d, 0.333, r)), 1.0 - ↙
        ↳ smoothstep(1.0 - d, 1.0, r));\n"
    "}\n"
1010 ;

    const char *draw_buttons_vert =
    "#version 330 core\n"
    "layout (location = 0) in vec2 pos;\n"
1015 "layout (location = 1) in vec2 tcd;\n"
    "layout (location = 2) in vec3 clr;\n"
    "out vec2 v_texcoord;\n"
    "out vec3 v_colour;\n"
    "void main() {\n"
1020 "  gl_Position = vec4(pos, 0.0, 1.0);\n"
    "  v_texcoord = vec2(tcd.y, 1.0 - tcd.x);\n"
    "  v_colour = clr;\n"
    "}\n"
    ;
1025
    const char *draw_buttons_frag =
    "#version 330 core\n"
    "uniform sampler2D selected;\n"
    "uniform sampler2D deselected;\n"
1030 "uniform int mode;\n"
    "in vec2 v_texcoord;\n"
    "in vec3 v_colour;\n"
    "out vec4 colour;\n"
    "void main() {\n"
1035 "  int me = int(floor(v_texcoord.y * 8.0));\n"
    "  float a = me == mode ? texture(selected, v_texcoord).r : texture(deselected, ↙
        ↳ v_texcoord).r;\n"
    "  colour = vec4(v_colour, a);\n"
    "}\n"
    ;
1040
    const char *draw_help_vert =
    "#version 330 core\n"
    "layout (location = 0) in vec2 pos;\n"
    "layout (location = 1) in vec2 tcd;\n"
1045 "layout (location = 2) in vec3 clr;\n"
    "out vec2 v_texcoord;\n"
    "out vec3 v_colour;\n"
    "void main() {\n"
    "  gl_Position = vec4(pos, 0.0, 1.0);\n"
1050 "  v_texcoord = vec2(tcd.y, 1.0 - tcd.x);\n"
    "  v_colour = clr;\n"
```

```
"}\n"
;

const char *draw_help_frag =
"#version 330 core\n"
"uniform sampler2D help;\n"
"in vec2 v_texcoord;\n"
"in vec3 v_colour;\n"
"out vec4 colour;\n"
"void main() {\n"
"  float a = texture(help, v_texcoord).r;\n"
"  colour = vec4(v_colour, a);\n"
"}\n"
;

unsigned char *read_file(const char *name) {
  FILE *f = fopen(name, "rb");
  fseek(f, 0, SEEK_END);
  long len = ftell(f);
  fseek(f, 0, SEEK_SET);
  unsigned char *p = (unsigned char *) malloc(len);
  fread(p, len, 1, f);
  fclose(f);
  return p;
}

graph_draw *draw_new()
{
  graph_draw *draw = new graph_draw();
  draw->components = 4096;
  glGenBuffers(1, &draw->vbo_tri);
  glBindBuffer(GL_ARRAY_BUFFER, draw->vbo_tri);
  glBufferData(GL_ARRAY_BUFFER, draw->components * sizeof(GLfloat), 0, ↙
      ↳ GL_DYNAMIC_DRAW);
  glGenVertexArrays(1, &draw->vao_tri);
  glBindVertexArray(draw->vao_tri);
  glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 8 * sizeof(GLfloat), 0);
  glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(GLfloat), ((char *)↙
      ↳ 0) + 2 * sizeof(GLfloat));
  glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(GLfloat), ((char *)↙
      ↳ 0) + 5 * sizeof(GLfloat));
  glEnableVertexAttribArray(0);
  glEnableVertexAttribArray(1);
  glEnableVertexAttribArray(2);
  glGenVertexArrays(1, &draw->vao_quad);
  glBindVertexArray(draw->vao_quad);
  glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 7 * sizeof(GLfloat), 0);
  glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 7 * sizeof(GLfloat), ((char *)↙
      ↳ 0) + 2 * sizeof(GLfloat));
  glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 7 * sizeof(GLfloat), ((char *)↙
      ↳ 0) + 4 * sizeof(GLfloat));
  glEnableVertexAttribArray(0);
  glEnableVertexAttribArray(1);
  glEnableVertexAttribArray(2);
  glGenTextures(3, &draw->tex[0]);
  unsigned char *g;
  glActiveTexture(GL_TEXTURE0 + 1);
```

```
  glBindTexture(GL_TEXTURE_2D, draw->tex[0]);
  g = read_file("deselected.raw");
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, 240, 800, 0, GL_RED, GL_UNSIGNED_BYTE, g);
  free(g);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glActiveTexture(GL_TEXTURE0 + 2);
  glBindTexture(GL_TEXTURE_2D, draw->tex[1]);
  g = read_file("selected.raw");
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, 240, 800, 0, GL_RED, GL_UNSIGNED_BYTE, g);
  free(g);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glActiveTexture(GL_TEXTURE0 + 0);
  glBindTexture(GL_TEXTURE_2D, draw->tex[2]);
  g = read_file("help.raw");
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, 800, 800, 0, GL_RED, GL_UNSIGNED_BYTE, g);
  free(g);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  draw->prog_circle  = vertex_fragment_shader(draw_circle_vert,  draw_circle_frag);
  draw->prog_arrow   = vertex_fragment_shader(draw_arrow_vert,   draw_arrow_frag);
  draw->prog_line    = vertex_fragment_shader(draw_line_vert,    draw_line_frag);
  draw->prog_buttons = vertex_fragment_shader(draw_buttons_vert, draw_buttons_frag);
  draw->prog_help    = vertex_fragment_shader(draw_help_vert,    draw_help_frag);
  glUseProgram(draw->prog_buttons);
  glUniform1i(glGetUniformLocation(draw->prog_buttons, "deselected"), 1);
  glUniform1i(glGetUniformLocation(draw->prog_buttons, "selected"), 2);
  draw->u_mode = glGetUniformLocation(draw->prog_buttons, "mode");
  glUseProgram(0);
  glDisable(GL_DEPTH_TEST);
  glEnable(GL_SCISSOR_TEST);
  glEnable(GL_BLEND);
  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  return draw;
}

struct {
  graph_mode current_mode;
  int active;
  graph_designer *designer[4];
  graph_draw *draw;
  double x, y;
} S;

void keycb(GLFWwindow *window, int key, int scancode, int action, int mods)
{
  glfwSetWindowShouldClose(window, GL_TRUE);
  (void) key;
```

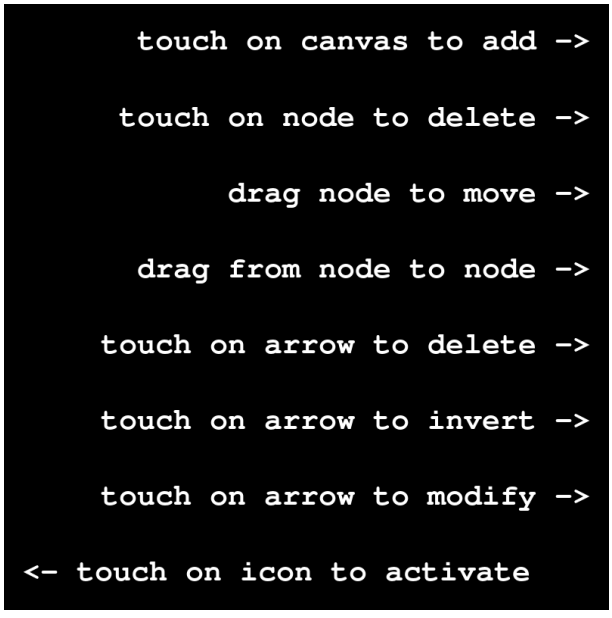
```
  (void) scancode;
  (void) action;
  (void) mods;
}

void motioncb(GLFWwindow *window, double x, double y)
{
  y = 800 - 1 - y;
  designer_mouse_move(S.designer[S.active], x - 240, y);
  S.x = x;
  S.y = y;
  (void) window;
}

void buttoncb(GLFWwindow *window, int button, int action, int mods)
{
  if (button == GLFW_MOUSE_BUTTON_LEFT)
  {
    if (S.x < 240)
    {
      int active = 4 - S.y / 200;
      if (active < 0)
        active = 0;
      if (active > 3)
        active = 3;
      designer_leave_mode(S.designer[S.active]);
      S.active = active;
      designer_enter_mode(S.designer[S.active], S.current_mode);
    }
    else if (S.x > 1040)
    {
      int mode = 8 - S.y / 100;
      if (mode < 0)
        mode = 0;
      if (mode > 7)
        mode = 7;
      designer_leave_mode(S.designer[S.active]);
      designer_enter_mode(S.designer[S.active], S.current_mode = graph_mode(mode
          ↪ ));
    }
    else
    {
      double x = S.x - 240;
      if (action == GLFW_PRESS)
        designer_mouse_down(S.designer[S.active], x, S.y);
      if (action == GLFW_RELEASE)
        designer_mouse_up(S.designer[S.active], x, S.y);
    }
  }
  (void) window;
  (void) mods;
}

int main()
{
  glfwInit();
  glfwWindowHint(GLFW_CLIENT_API, GLFW_OPENGL_API);
```

```
  glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
  glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
  glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
  glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
  glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);
  GLFWwindow *window = glfwCreateWindow(1280, 800, "graphgrow", 0, 0);
  glfwMakeContextCurrent(window);
  glewExperimental = GL_TRUE;
  glewInit();
  glGetError(); // discard common error from glew
  glfwSetKeyCallback(window, keycb);
  glfwSetCursorPosCallback(window, motioncb);
  glfwSetMouseButtonCallback(window, buttoncb);

  S.designer[0] = designer_new(0);
  S.designer[1] = designer_new(1);
  S.designer[2] = designer_new(2);
  S.designer[3] = designer_new(3);
  designer_enter_mode(S.designer[S.active = 0], S.current_mode = mode_help);
  S.draw = draw_new();

  lo_address at = lo_address_new("255.255.255.255", "6060");
  lo_address vt = lo_address_new("255.255.255.255", "6061");

  int frame = 0;
  while (1)
  {
    glfwPollEvents();
    if (glfwWindowShouldClose(window)) { break; }

    glScissor(0, 0, 240, 200);
    glViewport(20, 0, 200, 200);
    draw_designer(S.draw, S.designer[3], 3 == S.active, false);

    glScissor(0, 200, 240, 200);
    glViewport(20, 200, 200, 200);
    draw_designer(S.draw, S.designer[2], 2 == S.active, false);

    glScissor(0, 400, 240, 200);
    glViewport(20, 400, 200, 200);
    draw_designer(S.draw, S.designer[1], 1 == S.active, false);

    glScissor(0, 600, 240, 200);
    glViewport(20, 600, 200, 200);
    draw_designer(S.draw, S.designer[0], 0 == S.active, false);

    glScissor(240, 0, 800, 800);
    glViewport(240, 0, 800, 800);
    draw_designer(S.draw, S.designer[S.active], true, true);
    if (S.current_mode == mode_help)
      draw_help(S.draw, colours[S.active]);

    glScissor(1040, 0, 240, 800);
    glViewport(1040, 0, 240, 800);
    draw_buttons(S.draw, colours[S.active], S.current_mode);

    glfwSwapBuffers(window);
```

```
    GLint e = glGetError();
    if (e)
      fprintf(stderr, "OpenGL ERROR %d\n", e);

    audio_control a;
    memset(&a, 0, sizeof(a));
    for (int i = 0; i < 4; ++i)
      for (int k = 0; k < 8 && k < S.designer[i]->links.size(); ++k)
      {
        graph_link *l = S.designer[i]->links[k];
        int j = l->target;
        double dx = l->x2 - l->x1;
        double dy = l->y2 - l->y1;
        double ox = l->x2 + l->x1;
        a.scale[i][j][k] = sqrt(dx * dx + dy * dy) / 600;
        a.pan  [i][j][k] = (0.5 * ox - 400) / 400 * 0.5 + 0.5;
        a.level[i][j][k] = 1;
      }
    lo_blob ablob = lo_blob_new(sizeof(a), &a);
    if (ablob)
    {
      if (lo_send(at, "/audio", "b", ablob) == -1)
        fprintf(stderr, "OSC error %d: %s\n", lo_address_errno(at), ↙
            ↳ lo_address_errstr(at));
      lo_blob_free(ablob);
    }
    else
      fprintf(stderr, "OSC error: couldn't lo_blob_new\n");

    video_control v;
    memset(&v, 0, sizeof(v));
    v.active = S.active;
    for (int i = 0; i < 4; ++i)
    {
      v.count[i] = S.designer[i]->links.size();
      for (int j = 0; j < S.designer[i]->links.size() && j < 8; ++j)
      {
        graph_link *l = S.designer[i]->links[j];
        double dx = l->x2 - l->x1;
        double dy = l->y2 - l->y1;
        if (l->flipped)
        {
          dx = -dx;
          dy = -dy;
        }
        double ox = (l->x2 + l->x1) / 2;
        double oy = (l->y2 + l->y1) / 2;
        double s  = sqrt(dx * dx + dy * dy);
        dx /= 600;
        dy /= 600;
        s  /= 600;
        ox -= 400;
        oy -= 400;
        ox /= 600;
        oy /= 600;
        oy = -oy;
```

```
          v.source[i][j] = l->target;
          v.scale [i][j] = s;
          glm::mat3 transform = glm::mat3(float(dx), float(dy), float(ox), float(-↙
              ↳ dy), float(dx), float(oy), 0.0f, 0.0f, 1.0f);
1325      transform = glm::inverse(transform);
          for (int p = 0; p < 3; ++p)
            for (int q = 0; q < 3; ++q)
              v.transform[i][j][p][q] = transform[p][q];
        }
1330    }
      lo_blob vblob = lo_blob_new(sizeof(v), &v);
      if (vblob)
      {
        if (lo_send(vt, "/video", "b", vblob) == -1)
1335      fprintf(stderr, "OSC error %d: %s\n", lo_address_errno(vt), ↙
              ↳ lo_address_errstr(vt));
        lo_blob_free(vblob);
      }
      else
        fprintf(stderr, "OSC error: couldn't lo_blob_new\n");
1340
      frame++;
    }

    glfwDestroyWindow(window);
1345  glfwTerminate();
    return 0;
}
```

40 graphgrow3/iface/help.png



41 graphgrow3/iface/iface.xcf

41.1 Layer 0

41.2 Layer 1

41.3 Layer 2

41.4 Layer 3

41.5 Layer 4

41.6 Layer 5

41.7 Layer 6

41.8 Layer 7

41.9 Layer 8

41.10 Layer 9

41.11 Layer 10

41.12 Layer 11

41.13 Layer 12

41.14 Layer 13

41.15 Layer 14

41.16 Layer 15

41.17 Layer 16

41.18 Layer 17

42 graphgrow3/iface/Makefile

```
all: graphgrow-iface selected.raw deselected.raw help.raw

clean:
	-rm graphgrow-iface selected.raw deselected.raw help.raw

graphgrow-iface: graphgrow-iface.cc
	g++ -std=c++11 -Wall -Wextra -pedantic -Wno-variadic-macros -Wno-sign-compare -O3 -march=native -ffast-math -I$(HOME)/opt/include -L$(HOME)/opt/lib -o graphgrow-iface graphgrow-iface.cc -lGL -lGLEW -lglfw -llo

selected.raw: selected.png
	pngtopnm < selected.png | tail -c 192000 > selected.raw

deselected.raw: deselected.png
	pngtopnm < deselected.png | tail -c 192000 > deselected.raw

help.raw: help.png
	pngtopnm < help.png | tail -c 640000 > help.raw
```

43 graphgrow3/iface/README

```
Needs liblo from https://github.com/radarsat1/liblo
Tested with HEAD at 276dc11db2038b44f04fe67cdb425929427a5337
```

44 graphgrow3/iface/selected.png

45 graphgrow3/initial.frag

```
#version 130
uniform float er;
uniform float rho;

in vec2 texCoord;

out vec4 colour;

void main() {
  vec2 p = er * atanh(texCoord * 2.0 - vec2(1.0));
  float l = length(p);
  float n = (log(clamp(l, er * rho, er)) - log(er)) / log(rho);
  colour = vec4(n, 1.0, 0.0, 0.0);
}
```

46 graphgrow3/initial.vert

```
#version 130

out vec2 texCoord;

void main() {
  switch (gl_VertexID) {
    default:
    case 0:
      gl_Position = vec4(-1.0, -1.0, 0.0, 1.0);
      texCoord = vec2(0.0, 0.0);
      break;
    case 1:
```

```
        gl_Position = vec4( 1.0, -1.0, 0.0, 1.0);
        texCoord = vec2(1.0, 0.0);
        break;
      case 2:
        gl_Position = vec4(-1.0,  1.0, 0.0, 1.0);
        texCoord = vec2(0.0, 1.0);
        break;
      case 3:
        gl_Position = vec4( 1.0,  1.0, 0.0, 1.0);
        texCoord = vec2(1.0, 1.0);
        break;
    }
}
```

47 graphgrow3/LICENSE

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is 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. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

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.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License

giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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 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).

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 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.

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.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. 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 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 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 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

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 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

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.

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. Use with the GNU Affero General Public License.

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 Affero 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 special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU 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 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 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 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 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 General Public License for more details.

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

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

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 GPL, see <http://www.gnu.org/licenses/>.

48 graphgrow3/Main.hs

```
{-# LANGUAGE MultiParamTypeClasses, TypeFamilies, FlexibleContexts, ↙
    ↳ ForeignFunctionInterface #-}
module Main (main) where

import Graphics.UI.Toy.Gtk.Prelude hiding (D, bg, clamp, arrow, size, with)

import qualified Graphics.UI.Gtk        as G
import qualified Graphics.UI.Gtk.OpenGL as G


import Control.Monad (when, replicateM_, forM_)
import Control.Monad.IO.Class (liftIO)
import Data.IORef (IORef, newIORef, readIORef, writeIORef, atomicModifyIORef)

import Data.Colour.RGBSpace (uncurryRGB)
import Data.Colour.RGBSpace.HSV (hsv)
import Data.Colour.SRGB (sRGB, toSRGBBounded)
import Data.Fixed (mod')

import Linear (V2, V3, M33, (!*!), norm, (^-^), inv33)
import Data.Foldable (toList)

import Control.Lens (view)
import Data.Map (Map)
import qualified Data.Map.Strict as M
import qualified Data.Map.Strict.Merge as M
import Data.Maybe (maybeToList)

import System.IO (hFlush, stdout)

import Types
import Audio
import Video

type D = Draggable CairoDiagram -- every thing needs be draggable

mkD :: V2 Double -> CairoDiagram -> D
mkD = mkDraggable

-- things
data Thing = Background {-nextNodeID-}NID D | Node {-isFixedPosition-}Bool NID D↙
    ↳ D | Link LID D D D RID

-- visual representation
bg :: Int -> D
bg r = mkDraggable (r2 (250, 250)) $ (rect 500 500 # fc (blend 0.8 white (↙
    ↳ ruleColour r)))
```

```
node :: Int -> Bool -> V2 Double -> (D, D)
node r fixed p =
    ( mkDraggable p $ shape  5 # lc black # fc col
    , mkDraggable p $ shape 12 # fcA (col `withOpacity` 0.5)
    )
  where col = ruleColour r ; shape = if fixed then square . ((pi/2) *) else circle

link :: Int -> V2 Double -> V2 Double -> (D, D, D)
link r p@(V2 x0 y0) q@(V2 x1 y1) =
    ( mkD pc $ circle 5 # lc black # fc col
    , mkD pa $ arrow # scale 3 # strokeLocTrail # lc black # fc col # rotate a
    , mkD (V2 0 0) $ P p ~~ P q # strokeP # lc col
    )
  where
    col = ruleColour r
    pc@(V2 cx cy) = V2 ((x0+x1)/2) ((y0+y1)/2)
    pa = V2 (cx + 16 * dx) (cy + 16 * dy)
    ux = x1 - x0
    uy = y1 - y0
    a = (realToFrac $ atan2 uy ux) @@ rad
    l = case sqrt (ux * ux + uy * uy) of
      x | x == 0 -> 1
        | otherwise -> x
    dx =   uy / l
    dy = - ux / l
    arrow = (`at` p2(0,0)) . closeTrail . fromVertices . map p2 $ [(-3,-1),(1,-1),(1,-2),(3,0),(1,2),(1,1),(-3,1)]

-- extract diagrams
tDiagram :: Thing -> [D]
tDiagram (Background _ d)   = [d]
tDiagram (Node _ _ d1 d2)   = [d1, d2]  -- d1 is inner core, d2 is outer ring
tDiagram (Link _   d1 d2 d3 _) = [d1, d2, d3]  -- d1 is inner core, d3 is line, d2 is arrow

-- what did a mouse event do
data Act
  = Create NID
  | Delete NID
  | MoveStart NID
  | Moving    NID
  | MoveStop  NID
  | LinkStart        NID  (V2 Double)
  | Linking               (V2 Double)
  | LinkStop (Maybe (NID, (V2 Double)))
  | Adjust LID
  | ReverseLink LID
  | DeleteLink LID

-- what mode are we in
data Mode
  = MMove NID                          -- moving a node
  | MLink NID (V2 Double) (V2 Double) -- creating a link

data State = State (IORef (Map RID (Maybe G.Window, Toy State), IORef (Map RID Video, Map RID Audio))) RID (Maybe Mode) [Thing] Video Audio
```

```
100 -- update links when nodes are moved or deleted
    resort :: [RID] -> State -> State
    resort rs (State gs r m ts _ _) = State gs r m ts' trs adjs
      where
        adjs = (length nodes,
105       [ (i, j, lr, s, x)
          | Link (i, j) _ _ _ lr <- links'
          , let Just p0 = M.lookup i positions
          , let Just p1 = M.lookup j positions
          , let scale = norm (node0Position ^-^ node1Position)
110       , let s = norm (p0 ^-^ p1) / scale
          , let x = (view _x (lerp 0.5 p0 p1 ^-^ node0Position) / scale - 0.5) * 0.5↙
              ↳ + 0.5
          ])
        trs =
          [ (c, scaleFactor, transformMatrix)
115       | Link ij@(i, j) _ _ _ lr <- links'
          , let c = if lr `elem` rs then lr else 0
          , let Just p0 = M.lookup i positions
          , let Just p1 = M.lookup j positions
          , let (scaleFactor, transformMatrix) = linkTransformation p0 p1
120       ]
        nodes = [ n | n@(Node      {}) <- ts ]
        -- strip multiple links between the same pair of nodes (FIXME is this the ↙
            ↳ best way?)
        links = (M.elems . M.fromList) [ (sortPair ij, l) | l@(Link ij _ _ _ _) <- ↙
            ↳ ts ]
        backg = [ b | b@(Background{}) <- ts ]
125     positions = M.fromList [ (i, view dragOffset d) | Node _ i d _ <- nodes ]
        links' =
          [ Link ij d1 d2 d3 c
          | Link ij@(i, j) _ _ _ lr <- links
          , let c = if lr `elem` rs then lr else 0
130       , (d1, d2, d3) <- maybeToList $
              liftA2 (link c) (M.lookup i positions) (M.lookup j positions)
          ]
        ts' = nodes ++ links' ++ backg

135 linkTransformation :: V2 Double -> V2 Double -> (Double, M33 Double)
    linkTransformation p0@(V2 x0 y0) p1@(V2 x1 y1) = (scaleFactor, inv33 matrix)
      where
        scale = norm (node0Position ^-^ node1Position)
        scaleFactor = norm (p0 ^-^ p1) / scale
140     angle = -atan2 (y1 - y0) (x1 - x0)
        translation = (((p0 + p1)^/2) ^-^ ((node0Position + node1Position)^/2)) ^/ ↙
            ↳ scale
        matrix = transformRST angle scaleFactor translation

    transformRST :: Double -> Double -> V2 Double -> M33 Double
145 transformRST a l (V2 x y) = V3 (V3 c s x) (V3 (-s) c y) (V3 0 0 1)
      where
        c = l * cos a
        s = l * sin a

150

    -- mouse handling
```

```
tMouse :: Int -> Maybe Mode -> Maybe (Bool, Int) -> V2 Double -> Thing -> (Maybe Act, [Thing])
-- linking nodes
tMouse _ (Just(MLink j q _))(Just(False, 0)) p t@(Background _ _    ) = (Just $ LinkStop Nothing      , [t]) -- note: Background is the last Thing in the list
tMouse _ (Just(MLink j q _))(Just(False, 0)) p t@(Node fixed i d1 d2) | j /= i && clickInside d2 (P p)     = (Just $ LinkStop (Just (i, p)), [t])
tMouse _ (Just(MLink j q _))(Just(False, 0)) p t                                = (Nothing                        , [t])
tMouse _ (Just(MLink j q _))_                p t                                = (Just $ Linking     p         , [t])
-- adjusting links
tMouse _ Nothing           (Just (True,  0)) p   (Link i d1 d2 d3 r)  | clickInside d1 (P p)              = (Just $ Adjust      i, [Link i d1 d2 d3 (r + 1)])
                                                                     | clickInside d2 (P p)              = (Just $ ReverseLink i, [Link i d2 d1 d3 r])
tMouse _ Nothing           (Just (True,  1)) p   (Link i d1 d2 d3 _)  | clickInside d1 (P p)              = (Just $ DeleteLink  i, [])
-- moving nodes
tMouse _ Nothing         m@(Just (True,  0)) p t@(Node fixed i d1 d2) | clickInside d1 (P p) && not fixed  = (Just $ MoveStart i  , [Node False i (mouseDrag m q d1) (mouseDrag m q d2)])
                                                                     | clickInside d2 (P p)              = (Just $ LinkStart   i (view _2 q), [])
```

```
                                                                    ↳ dragOffset↙
                                                                    ↳  d1)↙
                                                                    ↳ , [t↙
                                                                    ↳ ]) ↙
                                                                    ↳
                                                                    ↳ where↙
                                                                    ↳  q =↙
                                                                    ↳  ↙
                                                                    ↳ clamp↙
                                                                    ↳ ' p
tMouse _ (Just(MMove j)) m@(Just (False, 0)) p   (Node False i d1 d2) | j == i ↙
    ↳                              = (Just $ MoveStop  i   , [Node False i (↙
    ↳ mouseDrag m q d1) (mouseDrag m q d2)]) where q = clamp' p
tMouse _ (Just(MMove j)) m                    p   (Node False i d1 d2) | j == i ↙
    ↳                              = (Just $ Moving    i   , [Node False i (↙
    ↳ mouseDrag m q d1) (mouseDrag m q d2)]) where q = clamp' p
-- creating nodes
tMouse c Nothing           (Just (True,  0)) p   (Background i d) ↙
    ↳                                    = (Just $ Create    i   , [uncurry↙
    ↳  (Node False i) (node c False p), Background (i + 1) d])
tMouse _ Nothing           (Just (True,  1)) p   (Node False i d1 d2) | ↙
    ↳ clickInside d1 (P p)               = (Just $ Delete    i   , [])
tMouse _ _currentMode      _mouseEvent       _ t@_thingToTest ↙
    ↳                                          = (Nothing             , [t])

clamp' :: V2 Double -> V2 Double
clamp' (V2 x y) = V2 (c x) (c y) where c z = 5 `max` z `min` 495

-- concatMap until a thing signals something was done, then just copy the rest
tMouses :: Int -> Maybe Mode -> Maybe (Bool, Int) -> V2 Double -> Maybe Act -> [↙
    ↳ Thing] -> (Maybe Act, [Thing])
tMouses _ _ _ _ a@(Just _) ws = (a, ws)
tMouses _ _ _ _ Nothing [] = (Nothing, [])
tMouses c d m p Nothing (w:ws) =
  let (a, ws') = tMouse c d m p w
  in  (ws' ++) `fmap` tMouses c d m p a ws

type instance N State = Double
type instance V State = V2

graph :: IORef (Map RID (Maybe G.Window, Toy State), IORef (Map RID Video, Map ↙
    ↳ RID Audio)) -> RID -> State
graph gs r = State gs r Nothing
  [ uncurry (Node True 0) (node r True node0Position)
  , uncurry (Node True 1) (node r True node1Position)
  , Background 2 (bg r)
  ] [] (2, [])

node0Position :: V2 Double
node0Position = V2 100 250

node1Position :: V2 Double
node1Position = V2 400 250

instance Interactive Gtk State where
  mouse _m _p s@(State g0 c0 _ _ _ _) = do
    rs <- (M.keys . fst) `fmap` readIORef g0
```

```
    s' <- simpleMouse ( \m p (State g c d ts trs adjs) -> resort rs $ case (d, ↙
        ↳ tMouses c d m p Nothing ts) of
        -- moving nodes
        (Nothing           , (Just (MoveStart i          ), ts'))         -> ↙
            ↳ State g c (Just (MMove i)) ts' trs adjs
        (Just (MMove j)    , (Just (Moving    i          ), ts')) | j == i -> ↙
            ↳ State g c (Just (MMove i)) ts' trs adjs
        (Just (MMove j)    , (Just (MoveStop  i          ), ts')) | j == i -> ↙
            ↳ State g c Nothing          ts' trs adjs
        -- linking nodes
        (Nothing,            (Just (LinkStart i p        ), ts'))         -> ↙
            ↳ State g c (Just (MLink i p p)) ts' trs adjs
        (Just (MLink j q _), (Just (Linking     p        ), ts'))         -> ↙
            ↳ State g c (Just (MLink j q p)) ts' trs adjs
        (Just (MLink j q _), (Just (LinkStop (Just (i, p))), ts')) | j /= i -> ↙
            ↳ State g c Nothing (uncurry3 (Link (j, i)) (link c q p) c : ts') ↙
            ↳ trs adjs
        (Just (MLink j q _), (Just (LinkStop Nothing      ), ts'))         -> ↙
            ↳ State g c Nothing ts' trs adjs
        -- everything else
        (_, (_, ts')) -> State g c Nothing ts' trs adjs
      ) _m _p s
    case s' of
      State g1 _ _ _ trs adjs -> do
        trsref <- snd `fmap` readIORef g1
        (trsmap, adjmap) <- readIORef trsref
        writeIORef trsref (M.insert c0 trs trsmap, M.insert c0 adjs adjmap)
    return s'

instance Diagrammable Cairo V2 Double Any State where
  diagram (State _ _ m ts _ _) = mconcat . map diagram $ modeDiagram ++ ↙
      ↳ concatMap tDiagram ts
    where
      modeDiagram = case m of
        Just (MLink _ p q) -> [mkD (V2 0 0) $ P p ~~ P q # strokeP # lc black]
        _ -> []

instance GtkDisplay State where
  display = displayDiagram diagram

main :: IO ()
main = do
  G.initGUI
  transforms <- newIORef (M.empty, M.empty)
  graphsRef <- newIORef (M.empty, transforms)
  window <- G.windowNew
  G.onDestroy window G.mainQuit
  G.set window [G.windowTitle G.:= "GG"]
  G.windowSetDefaultSize window 10 10
  G.windowSetTypeHint window G.WindowTypeHintDialog
  box <- G.hButtonBoxNew

  vis <- videoNew (fst <$> readIORef transforms)
  vbutton <- G.buttonNewWithLabel "V"
  vbutton `G.on` G.buttonActivated $ do
    G.windowPresent vis
    return ()
```

```
  G.containerAdd box vbutton

  (audio, audioQuit) <- audioNew (snd <$> readIORef transforms)
  G.timeoutAddFull (audio >> return True) G.priorityDefaultIdle 100

  button <- G.buttonNewWithLabel "+"
  button `G.on` G.buttonActivated $ do
    graphs <- fst `fmap` readIORef graphsRef
    let r = maybe 0 (succ . fst . fst) (M.maxViewWithKey graphs)
        gcol = uncurryRGB G.Color $ toSRGBBounded (ruleColour r)
    g <- newToy (graph graphsRef r)
    writeIORef graphsRef (M.insert r (Nothing, g) graphs, transforms)
    b <- G.buttonNew
    forM_ [G.StateNormal, G.StateActive, G.StatePrelight, G.StateSelected, G.StateInsensitive] $ \s ->
      G.widgetModifyBg b s gcol
    b `G.on` G.buttonActivated $ do
      graphs <- fst `fmap` readIORef graphsRef
      case M.lookup r graphs of
        Just (Just w,  g) -> G.windowPresent{-WithTime-} w
        Just (Nothing, g) -> do
          w <- G.windowNew
          w `G.on` G.deleteEvent $ do
            liftIO $ G.widgetHide w
            return True
          writeIORef graphsRef (M.insert r (Just w, g) graphs, transforms)
          G.windowSetGeometryHints w (Nothing `asTypeOf` Just w) (Just (500, 500)) (Just (500, 500)) Nothing Nothing Nothing
          G.set w [G.containerChild G.:= toyWindow g, G.windowTitle G.:= "GG#" ++ show r]
          G.widgetShowAll w
        _ -> return ()
    G.containerAdd box b
    G.widgetShowAll window

  G.containerAdd box button
  G.containerAdd window box
  G.widgetShowAll window

  G.mainGUI
  audioQuit

phi :: Double
phi = (sqrt 5 + 1) / 2

ruleColour :: Int -> Colour Double
ruleColour r = uncurryRGB sRGB $ hsv (360 * ((phi * fromIntegral r) `mod'` 1)) 1 1
```

49 graphgrow3/pitchshift~.pd

```
#N canvas 77 85 502 515 10;
#X obj 113 278 cos~;
#X obj 113 305 *~;
#X obj 113 334 +~;
#X obj 282 196 wrap~;
#X obj 282 288 cos~;
```

```
#X obj 282 315 *~;
#X obj 282 169 +~ 0.5;
#X obj 113 226 -~ 0.5;
#X obj 113 252 *~ 0.5;
#X obj 282 227 -~ 0.5;
#X obj 282 260 *~ 0.5;
#X obj 114 110 phasor~;
#X obj 115 59 inlet~;
#X obj 345 290 vd~ \$0-del;
#X obj 176 278 vd~ \$0-del;
#X obj 112 361 outlet~;
#X obj 13 7 inlet~;
#X obj 345 261 +~ 2;
#X obj 176 252 +~ 2;
#X obj 345 228 *~ 50;
#X obj 176 226 *~ 50;
#X obj 114 84 expr~ (1-exp($v1*0.05776))*1000.0/$f2;
#X obj 379 35 loadbang;
#X msg 376 61 20;
#X obj 12 34 delwrite~ \$0-del 25;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 15 0;
#X connect 3 0 9 0;
#X connect 3 0 19 0;
#X connect 4 0 5 0;
#X connect 5 0 2 1;
#X connect 6 0 3 0;
#X connect 7 0 8 0;
#X connect 8 0 0 0;
#X connect 9 0 10 0;
#X connect 10 0 4 0;
#X connect 11 0 7 0;
#X connect 11 0 6 0;
#X connect 11 0 20 0;
#X connect 12 0 21 0;
#X connect 13 0 5 1;
#X connect 14 0 1 1;
#X connect 16 0 24 0;
#X connect 17 0 13 0;
#X connect 18 0 14 0;
#X connect 19 0 17 0;
#X connect 20 0 18 0;
#X connect 21 0 11 0;
#X connect 22 0 23 0;
#X connect 23 0 21 1;
#X connect 23 0 19 1;
#X connect 23 0 20 1;
```

50 graphgrow3/README.md

```
# GraphGrow3

graph-directed iterated function system fractals

node editor with audiovisualisation
```

```
## Requirements

At runtime you need Pure-data executables 'pd' and 'pdsend' which should be in
your 'PATH'. Configure your '~/.pdrc' so that the correct audio device and
sample rate is set automatically. 'pd' will listen on network port '6060' to
receive messages from 'graphgrow3'. To build 'graphgrow3' you need 'ghc-8.0.1'
(other 'ghc' versions are untested) and recent 'cabal-install' (supporting
sandboxes). The visualisation uses OpenGL, hardware acceleration essential.

## Quick start

    git clone https://code.mathr.co.uk/toy-interface.git
    git clone https://code.mathr.co.uk/toy-diagrams.git
    git clone https://code.mathr.co.uk/toy-gtk.git
    git clone https://code.mathr.co.uk/toy-gtk-diagrams.git
    git clone https://code.mathr.co.uk/graphgrow.git
    cd graphgrow/graphgrow3
    cabal sandbox init
    cabal sandbox add-source ../../toy-*/
    cabal install
    .cabal-sandbox/bin/graphgrow3

## Usage

On startup the visualisation window "GG#V" is displayed, along with the "GG"
toolbar. The "V" button reshows the visualisation if it is closed. The "+"
button adds a new rule, which are colour coded. Clicking the coloured rule
buttons opens the editor for each rule. All windows apart from the "GG" toolbar
can be closed and re-opened later.

In the rule editor, left-click on empty space creates a new node. Nodes can be
left-mouse dragged by their centers to move them, and nodes are deleted with a
right-click. Left-mouse dragging between the outer rings of two nodes creates
a link. Links are deleted by right-clicking on their center. The direction of
a link can be reversed by left-clicking on the arrow. Left-clicking on the
center of a link changes the target rule for that link, cycling through the
available rules.

The square nodes are fixed, and correspond to the identity transformation (if
there were a link between them, but see bugs below).

## Bugs

Links longer than a bit less than one unit (the length between the two fixed
square nodes) slow down rendering a lot and might cause a denial of service to
your desktop session.

Adding more than 4 rules may break visualisation.

Adding more than 8 rules may break sonification.

Untested on anything apart from GNU/Linux/Debian/Testing/amd64.
```

51 graphgrow3/rule~.pd

```
#N canvas 3 113 450 300 10;
#X obj 44 200 compress~ 48;
```

```
#X obj 76 55 noise~;
#X obj 76 77 *~ 1e-06;
#X obj 43 175 lop~ 5000;
#X obj 43 153 hip~ 25;
#X obj 321 23 loadbang;
#X obj 321 45 f \$1;
#X obj 73 228 throw~ OUT-l;
#X obj 44 97 s~ RULE-\$1-l;
#X obj 43 131 catch~ RULE-\$1-l;
#X obj 167 176 lop~ 5000;
#X obj 167 154 hip~ 25;
#X obj 167 132 catch~ RULE-\$1-r;
#X obj 174 228 throw~ OUT-r;
#X obj 183 68 noise~;
#X obj 183 90 *~ 1e-06;
#X obj 151 110 s~ RULE-\$1-r;
#X obj 164 200 compress~ 48;
#X obj 321 89 + 2;
#X obj 321 67 * 0.1;
#X obj 154 46 delread~ RULE-\$1-r 2;
#X obj 43 24 delread~ RULE-\$1-l 2;
#X obj 128 279 delwrite~ RULE-\$1-r 5;
#X obj 42 255 delwrite~ RULE-\$1-l 5;
#X connect 0 0 7 0;
#X connect 0 0 23 0;
#X connect 1 0 2 0;
#X connect 2 0 8 0;
#X connect 3 0 0 0;
#X connect 4 0 3 0;
#X connect 5 0 6 0;
#X connect 6 0 19 0;
#X connect 9 0 4 0;
#X connect 10 0 17 0;
#X connect 11 0 10 0;
#X connect 12 0 11 0;
#X connect 14 0 15 0;
#X connect 15 0 16 0;
#X connect 17 0 13 0;
#X connect 17 0 22 0;
#X connect 18 0 20 0;
#X connect 18 0 21 0;
#X connect 19 0 18 0;
#X connect 20 0 16 0;
#X connect 21 0 8 0;
```

52 graphgrow3/Setup.hs

```
import Distribution.Simple
main = defaultMain
```

53 graphgrow3/step.frag

```
#version 130
#define MAXCOUNT 16
uniform float er;
uniform int count;
```

```
uniform mat3 transform[MAXCOUNT];
uniform float scaleFactor[MAXCOUNT];
uniform float source[MAXCOUNT];
uniform float layer;

uniform sampler2DArray src;

in vec2 texCoord;

out vec4 colour;

void main() {
  vec2 p0 = er * atanh(texCoord * 2.0 - vec2(1.0));
  float escape = -1000.0;
  float factor = 1.0;
  for (int i = 0; i < count && i < MAXCOUNT; ++i) {
    vec3 p = transform[i] * vec3(p0, 1.0);
    vec2 q = p.xy / p.z;
    float l = length(q);
    if (l < er) {
      vec2 z = texture(src, vec3((tanh(clamp(q / er, -vec2(4.0), vec2(4.0))) + ↙
          ↳ vec2(1.0)) / 2.0, source[i])).xy;
      if (escape < z.x) { escape = z.x; factor = 1.0/scaleFactor[i] * z.y; }
    }
  }
  escape += 1.0;
  vec2 z = texture(src, vec3(texCoord, layer)).xy;
  if (escape > z.x) {
    z.x = escape;
    z.y = factor;
  }
  colour = vec4(z, 0.0, 0.0);
}
```

54 graphgrow3/step.vert

```
#version 130

out vec2 texCoord;

void main() {
  switch (gl_VertexID) {
    default:
    case 0:
      gl_Position = vec4(-1.0, -1.0, 0.0, 1.0);
      texCoord = vec2(0.0, 0.0);
      break;
    case 1:
      gl_Position = vec4( 1.0, -1.0, 0.0, 1.0);
      texCoord = vec2(1.0, 0.0);
      break;
    case 2:
      gl_Position = vec4(-1.0,  1.0, 0.0, 1.0);
      texCoord = vec2(0.0, 1.0);
      break;
    case 3:
      gl_Position = vec4( 1.0,  1.0, 0.0, 1.0);
```

```
      texCoord = vec2(1.0, 1.0);
      break;
  }
}
```

55 graphgrow3/Types.hs

```
module Types where

import Data.Foldable (toList)
import Linear (M33)

-- rule, node and link ids
type RID = Int
type NID = Int
type LID = (NID, NID)

swapPair :: (a, b) -> (b, a)
swapPair (a, b) = (b, a)

sortPair :: Ord a => (a, a) -> (a, a)
sortPair ab@(a, b) = if a <= b then ab else (b, a)

uncurry3 :: (a -> b -> c -> r) -> (a, b, c) -> r
uncurry3 f (a, b, c) = f a b c

toLists :: M33 a -> [[a]]
toLists = map toList . toList
```

56 graphgrow3/video/.gitignore

```
graphgrow-video
graphgrow-video-test
```

57 graphgrow3/video/graphgrow-video.cc

```
#include <assert.h>
#include <complex.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <GL/glew.h>
#include <GLFW/glfw3.h>

#include <lo/lo.h>

// 1428 bytes < 1500 MTU
struct control
{
  int32_t active;
  int32_t count    [4];
  int32_t source   [4][8];
  float   scale    [4][8];
```

```
  float   transform[4][8][3][3];
};

#define FPS 30
#define WIDTH 1920
#define HEIGHT 1080
#define SKIP 1
#define SCALE 1
#define MIPMAP 3
#define QUALITY 10

static struct {
  int running;
  struct control control;
  // graphics
  int which;
  GLuint fbo[11];
  GLuint program[5];
  GLint p0which;
  GLint p0quality;
  GLint p0anim;
  GLint p0source;
  GLint p0count;
  GLint p1which;
  GLint p1quality;
  GLint p1layer;
  GLint p3count;
  GLint p3p;
  GLint p3q;
  GLint p4icount;
  GLint p4hcount;
  GLuint pbo[3];
} S;

void errorcb(int num, const char *msg, const char *path)
{
  fprintf(stderr, "liblo server error %d in path %s: %s\n", num, path, msg);
}

int videocb(const char *path, const char *types, lo_arg **argv, int argc, void *↙
    ↳ message, void *user_data)
{
  const int bytes = sizeof(S.control);
  int   size =  argv[0]->blob.size;
  void *data = &argv[0]->blob.data;
  if (size == bytes)
    memcpy(&S.control, data, bytes);
  return 0;
  (void) path;
  (void) types;
  (void) argc;
  (void) message;
  (void) user_data;
}

int quitcb(const char *path, const char *types, lo_arg **argv, int argc, void *↙
    ↳ message, void *user_data)
```

```
{
  S.running = 0;
  return 0;
  (void) path;
  (void) types;
  (void) argv;
  (void) argc;
  (void) message;
  (void) user_data;
}

static const char *step_vert =
"#version 330 core\n"
"layout(location = 0) in vec2 pos;\n"
"layout(location = 1) in vec2 tc;\n"
"out vec2 coord;\n"
"void main() {\n"
"  gl_Position = vec4(pos, 0.0, 1.0);\n"
"  coord = vec2(tc.x, 1.0 - tc.y);\n"
"}\n"
;

static const char *step_frag =
"#version 330 core\n"
"#define MAXCOUNT 8\n"
"uniform sampler2DArray which;\n"
"uniform int quality;\n"
"uniform int layer;\n"
"uniform int count;\n"
"uniform mat3 transform[MAXCOUNT];\n"
"uniform int source[MAXCOUNT];\n"
"in vec2 coord;\n"
"layout(location = 0) out vec3 colour;\n"
"const vec3[4] shade = vec3[4]\n"
"  ( vec3(1.00, 0.95, 0.90)\n"
"  , vec3(0.95, 1.00, 0.90)\n"
"  , vec3(0.90, 0.95, 1.00)\n"
"  , vec3(0.95, 0.90, 1.00)\n"
"  );\n"
"vec2 fromSquare(vec2 z) {\n"
"  return clamp(atanh(2.0 * z - vec2(1.0)), -4.0, 4.0);\n"
"}"
"vec2 toSquare(vec2 z) {\n"
"  return (tanh(clamp(z, -4.0, 4.0)) + vec2(1.0)) / 2.0;\n"
"}\n"
"float gain(vec2 z, vec2 z0) {\n"
"  vec4 d0 = vec4(dFdx(z), dFdy(z));\n"
"  vec4 d1 = vec4(dFdx(z0), dFdy(z0));\n"
"  return dot(d1,d1) / dot(d0, d0);\n"
"}\n"
"vec3 lookup(vec2 z, vec2 z0, float l) {\n"
"  vec2 w = toSquare(z);\n"
"  return gain(z, z0) * texture(which, vec3(w, l)).rgb;\n"
"}\n"
"void main() {\n"
"  vec3[4] levels = vec3[4]\n"
"    ( texelFetch(which, ivec3(0,0,0), quality).rgb\n"
```

```
"     , texelFetch(which, ivec3(0,0,1), quality).rgb\n"
"     , texelFetch(which, ivec3(0,0,2), quality).rgb\n"
"     , texelFetch(which, ivec3(0,0,3), quality).rgb\n"
"     );\n"
"   float[4] level = float[4]\n"
"     ( max(max(levels[0].r, levels[0].g), levels[0].b)\n"
"     , max(max(levels[1].r, levels[1].g), levels[1].b)\n"
"     , max(max(levels[2].r, levels[2].g), levels[2].b)\n"
"     , max(max(levels[3].r, levels[3].g), levels[3].b)\n"
"     );\n"
"   vec2 z = fromSquare(coord);\n"
"   vec3 total = vec3(0.001);\n"
"   for (int i = 0; i < count && i < MAXCOUNT; ++i) {\n"
"     vec3 w = transform[i] * vec3(z, 1.0);\n"
"     total += shade[source[i]] * lookup(w.xy / w.z, coord, float(source[i])) / level[source[i]];\n"
"   }\n"
"   colour = clamp(total, 0.0, 16777216.0);\n"
"}\n"
;

static const char *show_vert =
"#version 330 core\n"
"uniform int quality;\n"
"uniform float aspect;\n"
"layout(location = 0) in vec2 pos;\n"
"layout(location = 1) in vec2 tc;\n"
"out vec2 coord;\n"
"void main() {\n"
"   gl_Position = vec4(pos, 0.0, 1.0);\n"
"   coord = 2.0 * vec2((tc.x - 0.5), (0.5 - tc.y) / aspect);\n"
"}\n"
;

static const char *show_frag =
"#version 330 core\n"
"uniform sampler2DArray which;\n"
"uniform float level;\n"
"uniform int layer;\n"
"in vec2 coord;\n"
"layout(location = 0) out vec4 colour;\n"
"vec2 toSquare(vec2 z) {\n"
"   return (tanh(clamp(z, -4.0, 4.0)) + vec2(1.0)) / 2.0;\n"
"}\n"
"void main() {\n"
"   vec3 c = texture(which, vec3(toSquare(coord), float(layer))).rgb;\n"
"   colour = vec4(c, dot(vec3(1.0 / 3.0), c));\n"
"}\n"
;

static const char *tone_vert =
"#version 330 core\n"
"layout(location = 0) in vec2 pos;\n"
"layout(location = 1) in vec2 tc;\n"
"out vec2 coord;\n"
"void main() {\n"
"   gl_Position = vec4(pos, 0.0, 1.0);\n"
```

```
    "  coord = vec2(tc.x, tc.y);\n"
    "}\n"
    ;

    static const char *tone_frag =
    "#version 330 core\n"
    "uniform sampler2D source;\n"
    "in vec2 coord;\n"
    "layout(location = 0) out vec4 colour;\n"
    "void main() {\n"
    "  vec4 v = texture(source, coord);\n"
    "  colour = vec4(pow(v.a, 4.0) * v.rgb / max(max(v.r, v.g), v.b), 1.0);\n"
    "}\n"
    ;

    static const char *sort_comp =
    "#version 440 core\n"
    "layout(local_size_x = 1024) in;\n"
    "layout(std430, binding = 0) buffer src {\n"
    "  restrict readonly float srcdata[];\n"
    "};\n"
    "layout(std430, binding = 1) buffer dst {\n"
    "  restrict writeonly float dstdata[];\n"
    "};\n"
    "uniform uint count;\n"
    "uniform uint p;\n"
    "uniform uint q;\n"
    "void main() {\n"
    "  uint i = gl_GlobalInvocationID.x;\n"
    "  uint d = 1u << (p - q);\n"
    "  if ((i & d) == 0 && i < count) {\n"
    "    float a = srcdata[i];\n"
    "    if ((i | d) < count) {\n"
    "      bool up = ((i >> p) & 2) == 0;\n"
    "      float b = srcdata[i | d];\n"
    "      if ((a > b) == up) {\n"
    "        float t = a; a = b; b = t;\n"
    "      }\n"
    "      dstdata[i | d] = b;\n"
    "    }\n"
    "    dstdata[i] = a;\n"
    "  }\n"
    "}\n"
    ;

    static const char *look_comp =
    "#version 440 core\n"
    "layout(local_size_x = 1024) in;\n"
    "layout(std430, binding = 0) buffer img {\n"
    "  restrict readonly float imgdata[];\n"
    "};\n"
    "layout(std430, binding = 1) buffer hst {\n"
    "  restrict readonly float hstdata[];\n"
    "};\n"
    "layout(std430, binding = 2) buffer dst {\n"
    "  restrict writeonly float dstdata[];\n"
    "};\n"
```

```
    "uniform uint icount;\n"
    "uniform uint hcount;\n"
    "void main() {\n"
    "  uint i = gl_GlobalInvocationID.x;\n"
    "  if (i < icount) {\n"
    "    float x = imgdata[4 * i + 3];\n"
    "    uint l = 0;\n"
    "    uint r = hcount;\n"
    "    uint m = (l + r) / 2;\n"
    "    for (uint p = 0; p < 24; ++p) {\n"
    "      if (r < l + 64) {\n"
    "        break;\n"
    "      }\n"
    "      float y = hstdata[m];\n"
    "      if (x < y) {\n"
    "        r = m;\n"
    "        m = (l + r) / 2;\n"
    "        continue;\n"
    "      } else if (x > y) {\n"
    "        l = m;\n"
    "        m = (l + r) / 2;\n"
    "        continue;\n"
    "      } else {\n"
    "       break;\n"
    "      }\n"
    "    }\n"
    "    dstdata[4 * i + 0] = imgdata[4 * i + 0];\n"
    "    dstdata[4 * i + 1] = imgdata[4 * i + 1];\n"
    "    dstdata[4 * i + 2] = imgdata[4 * i + 2];\n"
    "    dstdata[4 * i + 3] = float(m) / float(hcount);\n"
    "  }\n"
    "}\n"
    ;

  static void debug_program(GLuint program, const char *name) {
    if (program) {
      GLint linked = GL_FALSE;
      glGetProgramiv(program, GL_LINK_STATUS, &linked);
      if (linked != GL_TRUE) {
        fprintf(stderr, "%s: OpenGL shader program link failed\n", name);
      }
      GLint length = 0;
      glGetProgramiv(program, GL_INFO_LOG_LENGTH, &length);
      char *buffer = (char *) malloc(length + 1);
      glGetProgramInfoLog(program, length, 0, buffer);
      buffer[length] = 0;
      if (buffer[0]) {
        fprintf(stderr, "%s: OpenGL shader program info log\n", name);
        fprintf(stderr, "%s\n", buffer);
      }
      free(buffer);
      assert(linked == GL_TRUE);
    } else {
      fprintf(stderr, "%s: OpenGL shader program creation failed\n", name);
    }
  }
```

```
static void debug_shader(GLuint shader, GLenum type, const char *name) {
  const char *tname = 0;
  switch (type) {
    case GL_VERTEX_SHADER:   tname = "vertex";   break;
    case GL_FRAGMENT_SHADER: tname = "fragment"; break;
    case GL_COMPUTE_SHADER:  tname = "compute";  break;
    default:                 tname = "unknown";  break;
  }
  if (shader) {
    GLint compiled = GL_FALSE;
    glGetShaderiv(shader, GL_COMPILE_STATUS, &compiled);
    if (compiled != GL_TRUE) {
      fprintf(stderr, "%s: OpenGL %s shader compile failed\n", name, tname);
    }
    GLint length = 0;
    glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &length);
    char *buffer = (char *) malloc(length + 1);
    glGetShaderInfoLog(shader, length, 0, buffer);
    buffer[length] = 0;
    if (buffer[0]) {
      fprintf(stderr, "%s: OpenGL %s shader info log\n", name, tname);
      fprintf(stderr, "%s\n", buffer);
    }
    free(buffer);
    assert(compiled == GL_TRUE);
  } else {
    fprintf(stderr, "%s: OpenGL %s shader creation failed\n", name, tname);
  }
}

static void compile_shader(GLint program, GLenum type, const char *name, const ↙
    ↳ GLchar *source) {
  GLuint shader = glCreateShader(type);
  glShaderSource(shader, 1, &source, 0);
  glCompileShader(shader);
  debug_shader(shader, type, name);
  glAttachShader(program, shader);
  glDeleteShader(shader);
}

static GLint compile_program(const char *name, const GLchar *vert, const GLchar ↙
    ↳ *frag) {
  GLint program = glCreateProgram();
  if (vert) { compile_shader(program, GL_VERTEX_SHADER   , name, vert); }
  if (frag) { compile_shader(program, GL_FRAGMENT_SHADER, name, frag); }
  glLinkProgram(program);
  debug_program(program, name);
  return program;
}

static GLint compile_compute(const char *name, const GLchar *comp) {
  GLint program = glCreateProgram();
  if (comp) { compile_shader(program, GL_COMPUTE_SHADER, name, comp); }
  glLinkProgram(program);
  debug_program(program, name);
  return program;
}
```

```
static void key_press_handler(GLFWwindow *window, int key, int scancode, int ↙
    ↳ action, int mods) {
  (void) key;
  (void) scancode;
  (void) action;
  (void) mods;
  glfwSetWindowShouldClose(window, GL_TRUE);
}

static GLFWwindow *create_window(int major, int minor, int width, int height, ↙
    ↳ const char *title) {
  glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, major);
  glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, minor);
  glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
  glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
  glfwWindowHint(GLFW_OPENGL_DEBUG_CONTEXT, GL_TRUE);
  glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);
  glfwWindowHint(GLFW_DECORATED, GL_FALSE);
  GLFWwindow *window = glfwCreateWindow(width, height, title, 0, 0);
  if (! window) {
    fprintf(stderr, "couldn't create window with OpenGL core %d.%d context\n", ↙
        ↳ major, minor);
  }
  assert(window);
  return window;
}

static int debug_error_count = 0;
static void debug_callback(GLenum source, GLenum type, GLuint id, GLenum ↙
    ↳ severity, GLsizei length, const GLchar *message, const GLvoid *user) {
  (void) user;
  (void) length;
  const char *source_str = "unknown";
  switch (source) {
    case GL_DEBUG_SOURCE_API:             source_str = "OpenGL";          break;
    case GL_DEBUG_SOURCE_WINDOW_SYSTEM:   source_str = "window system";   break;
    case GL_DEBUG_SOURCE_SHADER_COMPILER: source_str = "shader compiler"; break;
    case GL_DEBUG_SOURCE_THIRD_PARTY:     source_str = "third party";     break;
    case GL_DEBUG_SOURCE_APPLICATION:     source_str = "application";     break;
    case GL_DEBUG_SOURCE_OTHER:           source_str = "application";     break;
  }
  const char *type_str = "unknown";
  switch (type) {
    case GL_DEBUG_TYPE_ERROR:       type_str = "error";       break;
    case GL_DEBUG_TYPE_DEPRECATED_BEHAVIOR: type_str = "deprecated behavior"; ↙
        ↳ break;
    case GL_DEBUG_TYPE_UNDEFINED_BEHAVIOR:  type_str = "undefined behavior";  ↙
        ↳ break;
    case GL_DEBUG_TYPE_PORTABILITY: type_str = "portability"; break;
    case GL_DEBUG_TYPE_PERFORMANCE: type_str = "performance"; break;
    case GL_DEBUG_TYPE_MARKER:      type_str = "marker";      break;
    case GL_DEBUG_TYPE_PUSH_GROUP:  type_str = "push group";  break;
    case GL_DEBUG_TYPE_POP_GROUP:   type_str = "pop group";   break;
    case GL_DEBUG_TYPE_OTHER:       type_str = "other";       break;
  }
  const char *severity_str = "unknown";
```

```
  switch (severity) {
    case GL_DEBUG_SEVERITY_HIGH:         severity_str = "high";         break;
    case GL_DEBUG_SEVERITY_MEDIUM:       severity_str = "medium";       break;
    case GL_DEBUG_SEVERITY_LOW:          severity_str = "low";          break;
    case GL_DEBUG_SEVERITY_NOTIFICATION: severity_str = "notification"; break;
  }
  bool should_print = true;
  if (should_print) {
    fprintf(stderr, "graphgrow-video: %s %s %u %s: %s\n", source_str, type_str, ↙
        ↳ id, severity_str, message);
  }
  if (severity == GL_DEBUG_SEVERITY_HIGH && type == GL_DEBUG_TYPE_ERROR) {
    if (++debug_error_count > 10) {
      abort();
    }
  }
}

static GLFWwindow *create_context(int width, int height, const char *title) {
  int glfw_initialized = glfwInit();
  if (! glfw_initialized) {
    fprintf(stderr, "couldn't initialize glfw\n");
    assert(glfw_initialized);
    return 0;
  }
  int major, minor;
  GLFWwindow *window = create_window(major = 4, minor = 4, width, height, title)↙
      ↳ ;
  if (! window) {
    fprintf(stderr, "couldn't create window\n");
    assert(window);
    return 0;
  }
  glfwMakeContextCurrent(window);
  glewExperimental = GL_TRUE;
  glewInit();
  glGetError();
  glDebugMessageCallback(debug_callback, 0);
  return window;
}

static void initialize_gl(int tex_width, int tex_height, int win_width, int ↙
    ↳ win_height) {

  // initialize vertex data for full-screen triangle strip
  GLuint vao;
  glGenVertexArrays(1, &vao);
  glBindVertexArray(vao);
  GLuint vbo;
  glGenBuffers(1, &vbo);
  glBindBuffer(GL_ARRAY_BUFFER, vbo);
  GLfloat vbo_data[] =
    { -1, -1, 0, 1
    , -1,  1, 0, 0
    ,  1, -1, 1, 1
    ,  1,  1, 1, 0
    };
```

```
  glBufferData(GL_ARRAY_BUFFER, 16 * sizeof(GLfloat), vbo_data, GL_STATIC_DRAW);
  glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 4 * sizeof(GLfloat), 0);
465 glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 4 * sizeof(GLfloat), ((char *)↙
      ↳ 0) + 2 * sizeof(GLfloat));
  glEnableVertexAttribArray(0);
  glEnableVertexAttribArray(1);

  // create textures for ping pong square
470 GLuint t_buffer[2];
  glGenTextures(2, t_buffer);
  for (int i = 0; i < 2; ++i) {
    glActiveTexture(GL_TEXTURE0 + i);
    glBindTexture(GL_TEXTURE_2D_ARRAY, t_buffer[i]);
475 glTexImage3D(GL_TEXTURE_2D_ARRAY, 0, GL_RGB32F, tex_width, tex_height, 4, 0,↙
      ↳ GL_RGB, GL_FLOAT, 0);
    glGenerateMipmap(GL_TEXTURE_2D_ARRAY);
    glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MIN_FILTER, ↙
      ↳ GL_LINEAR_MIPMAP_LINEAR);
    glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_S, GL_CLAMP);
480 glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_T, GL_CLAMP);
  }

  // create texture for rectangular view
  GLuint t_screen;
485 glGenTextures(1, &t_screen);
  glActiveTexture(GL_TEXTURE0 + 2);
  glBindTexture(GL_TEXTURE_2D, t_screen);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F, win_width, win_height, 0, GL_RGBA, ↙
      ↳ GL_FLOAT, 0);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
490 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_MIRRORED_REPEAT);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_MIRRORED_REPEAT);

  // create framebuffers for ping pong square
495 glViewport(0, 0, tex_width, tex_height);
  glClearColor(1, 1, 1, 1);
  glGenFramebuffers(9, S.fbo);
  for (int i = 0; i < 8; ++i) {
    glBindFramebuffer(GL_FRAMEBUFFER, S.fbo[i]);
500 glFramebufferTextureLayer(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, t_buffer[1 ↙
      ↳ - i / 4], 0, i % 4);
    GLenum dbuf = GL_COLOR_ATTACHMENT0;
    glDrawBuffers(1, &dbuf);
    GLenum status = glCheckFramebufferStatus(GL_FRAMEBUFFER);
    if (status != GL_FRAMEBUFFER_COMPLETE) {
505   fprintf(stderr, "OpenGL framebuffer not complete\n");
    }
    assert(status == GL_FRAMEBUFFER_COMPLETE);
    glClear(GL_COLOR_BUFFER_BIT);
  }
510 glActiveTexture(GL_TEXTURE0 + 0);
  glGenerateMipmap(GL_TEXTURE_2D_ARRAY);
  glActiveTexture(GL_TEXTURE0 + 1);
  glGenerateMipmap(GL_TEXTURE_2D_ARRAY);
```

```
515   // create framebuffer for rectangular view
      glBindFramebuffer(GL_FRAMEBUFFER, S.fbo[8]);
      glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, ↙
          ↳ t_screen, 0);
      GLenum dbuf = GL_COLOR_ATTACHMENT0;
      glDrawBuffers(1, &dbuf);
520   GLenum status = glCheckFramebufferStatus(GL_FRAMEBUFFER);
      if (status != GL_FRAMEBUFFER_COMPLETE) {
        fprintf(stderr, "OpenGL framebuffer not complete\n");
      }
      assert(status == GL_FRAMEBUFFER_COMPLETE);
525   glClear(GL_COLOR_BUFFER_BIT);

      // compile shaders
      S.program[0] = compile_program("step", step_vert, step_frag);
      S.p0which = glGetUniformLocation(S.program[0], "which");
530   S.p0quality = glGetUniformLocation(S.program[0], "quality");
      S.p0anim  = glGetUniformLocation(S.program[0], "transform");
      S.p0source  = glGetUniformLocation(S.program[0], "source");
      S.p0count = glGetUniformLocation(S.program[0], "count");
      S.program[1] = compile_program("show", show_vert, show_frag);
535   glUseProgram(S.program[1]);
      S.p1which = glGetUniformLocation(S.program[1], "which");
      S.p1quality = glGetUniformLocation(S.program[1], "quality");
      S.p1layer = glGetUniformLocation(S.program[1], "layer");
      GLuint aspect = glGetUniformLocation(S.program[1], "aspect");
540   glUniform1f(aspect, WIDTH / (float) HEIGHT);
      S.which = 0;
      S.program[2] = compile_program("tone", tone_vert, tone_frag);
      glUseProgram(S.program[2]);
      GLint p2source = glGetUniformLocation(S.program[2], "source");
545   glUniform1i(p2source, 2);

      // pixel buffers for ping pong histogram equalisation
      int logsize = ceil(log2(win_width * win_height));
      glGenBuffers(3, &S.pbo[0]);
550   for (int i = 0; i < 3; ++i) {
        glBindBuffer(GL_PIXEL_PACK_BUFFER, S.pbo[i]);
        glBufferData(GL_PIXEL_PACK_BUFFER, 4 * (1 << logsize) * sizeof(float), 0, ↙
            ↳ GL_DYNAMIC_COPY);
      }

555   // compute shaders
      S.program[3] = compile_compute("sort", sort_comp);
      S.p3count = glGetUniformLocation(S.program[3], "count");
      S.p3p = glGetUniformLocation(S.program[3], "p");
      S.p3q = glGetUniformLocation(S.program[3], "q");
560   S.program[4] = compile_compute("look", look_comp);
      S.p4icount = glGetUniformLocation(S.program[4], "icount");
      S.p4hcount = glGetUniformLocation(S.program[4], "hcount");

    }
565
    extern int main(int argc, char **argv) {
      memset(&S, 0, sizeof(S));
      int quality = QUALITY;
      int tex_width = 1 << quality, tex_height = 1 << quality;
```

```
570     int win_width = WIDTH/SCALE, win_height = HEIGHT/SCALE;
        int hist_width = win_width >> MIPMAP;
        int hist_height = win_height >> MIPMAP;
        int logsize = ceil(log2(hist_width * hist_height));
        GLFWwindow *window = create_context(win_width, win_height, "graphgrow");
575     if (! window) {
          assert(window);
          return 1;
        }
        glfwSetKeyCallback(window, key_press_handler);
580     initialize_gl(tex_width, tex_height, win_width, win_height);

        GLuint src = S.pbo[0];
        GLuint dst = S.pbo[1];
        GLuint img = S.pbo[2];
585
        GLuint timers[7];
        glGenQueries(7, timers);

        lo_server_thread lo = lo_server_thread_new("6061", errorcb);
590     lo_server_thread_add_method(lo, "/video", "b", videocb, 0);
        lo_server_thread_add_method(lo, "/quit", "", quitcb, 0);
        lo_server_thread_start(lo);

        int frame = 0;
595     S.running = 1;
        while (S.running && ! glfwWindowShouldClose(window)) {

          if ((frame % 60) == 0 && frame > 0)
          {
600         GLuint64 tifs = 0, tflat = 0, tpbo = 0, tsort = 0, tlup = 0, ttex = 0, ↙
                ↳ tdisp = 0;
            glGetQueryObjectui64v(timers[0], GL_QUERY_RESULT, &tifs);
            glGetQueryObjectui64v(timers[1], GL_QUERY_RESULT, &tflat);
            glGetQueryObjectui64v(timers[2], GL_QUERY_RESULT, &tpbo);
            glGetQueryObjectui64v(timers[3], GL_QUERY_RESULT, &tsort);
605         glGetQueryObjectui64v(timers[4], GL_QUERY_RESULT, &tlup);
            glGetQueryObjectui64v(timers[5], GL_QUERY_RESULT, &ttex);
            glGetQueryObjectui64v(timers[6], GL_QUERY_RESULT, &tdisp);
            double ifs = tifs / 1000.0;
            double flat = tflat / 1000.0;
610         double pbo = tpbo / 1000.0;
            double sort = tsort / 1000.0;
            double lup = tlup / 1000.0;
            double tex = ttex / 1000.0;
            double disp = tdisp / 1000.0;
615         fprintf(stderr, " IFS( %f ) FLAT( %f ) PBO( %f ) SORT( %f ) LUP( %f ) TEX(↙
                ↳ %f ) DISP( %f )\r", ifs, flat, pbo, sort, lup, tex, disp);
            glDeleteQueries(7, timers);
            glGenQueries(7, timers);
          }

620       // iterated function system
          if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[0]);
          glUseProgram(S.program[0]);
          glViewport(0, 0, tex_width, tex_height);
          for (int layer = 0; layer < 4; ++layer)
```

```
625       {
            glUniformMatrix3fv(S.p0anim, 8, GL_TRUE, &S.control.transform[layer↙
                ↳ ][0][0][0]);
            glUniform1iv(S.p0source, 8, &S.control.source[layer][0]);
            glUniform1i(S.p0count, S.control.count[layer]);
            glUniform1i(S.p0which, S.which);
630         glUniform1i(S.p0quality, quality);
            glBindFramebuffer(GL_FRAMEBUFFER, S.fbo[S.which * 4 + layer]);
            glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
          }
          S.which = 1 - S.which;
635       glActiveTexture(GL_TEXTURE0 + S.which);
          glGenerateMipmap(GL_TEXTURE_2D_ARRAY);
          if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

          // flatten from square texture to rectangular view
640       if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[1]);
          glViewport(0, 0, win_width, win_height);
          glBindFramebuffer(GL_FRAMEBUFFER, S.fbo[8]);
          glUseProgram(S.program[1]);
          glUniform1i(S.p1which, S.which);
645       glUniform1i(S.p1quality, quality);
          glUniform1i(S.p1layer, S.control.active);
          glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
          if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

650       if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[2]);
          glActiveTexture(GL_TEXTURE0 + 2);
          if (frame % SKIP == 0)
          {
            // copy to PBO
655         glGenerateMipmap(GL_TEXTURE_2D);
            glBindBuffer(GL_PIXEL_PACK_BUFFER, src);
            float infty = 1.0 / 0.0;
            glClearBufferSubData(GL_PIXEL_PACK_BUFFER, GL_R32F, 0, (1 << logsize) * ↙
                ↳ sizeof(float), GL_RED, GL_FLOAT, &infty);
            glBindBuffer(GL_PIXEL_PACK_BUFFER, src);
660         glGetTexImage(GL_TEXTURE_2D, MIPMAP, GL_ALPHA, GL_FLOAT, 0);
          }
          glBindBuffer(GL_PIXEL_PACK_BUFFER, img);
          glGetTexImage(GL_TEXTURE_2D, 0, GL_RGBA, GL_FLOAT, 0);
          glBindBuffer(GL_PIXEL_PACK_BUFFER, 0);
665       if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

          if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[3]);
          if (frame % SKIP == 0)
          {
670         // ping pong merge sort
            glUseProgram(S.program[3]);
            glUniform1ui(S.p3count, 1 << logsize);
            for (int i = 0; i < logsize; ++i) {
              for (int j = 0; j <= i; ++j) {
675             glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 0, src);
                glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 1, dst);
                glUniform1ui(S.p3p, i);
                glUniform1ui(S.p3q, j);
                glDispatchCompute((1 << logsize) / 1024, 1, 1);
```

```
680             { GLuint t = src; src = dst; dst = t; }
              }
            }
          }
          if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

          // binary search lookup
          if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[4]);
          glUseProgram(S.program[4]);
          glUniform1ui(S.p4icount, win_width * win_height);
          glUniform1ui(S.p4hcount, hist_width * hist_height);
          glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 0, img);
          glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 1, src);
          glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 2, dst);
          glDispatchCompute((win_width * win_height + 1024 - 1) / 1024, 1, 1);
          if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

          // copy to texture
          if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[5]);
          glActiveTexture(GL_TEXTURE0 + 2);
          glBindBuffer(GL_PIXEL_UNPACK_BUFFER, dst);
          glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, win_width, win_height, GL_RGBA, ↙
              ↳ GL_FLOAT, 0);
          glBindBuffer(GL_PIXEL_UNPACK_BUFFER, 0);
          if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

          // display final image
          if ((frame % 60) == 0) glBeginQuery(GL_TIME_ELAPSED, timers[6]);
          glUseProgram(S.program[2]);
          glBindFramebuffer(GL_FRAMEBUFFER, 0);
          glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
          if ((frame % 60) == 0) glEndQuery(GL_TIME_ELAPSED);

          glfwSwapBuffers(window);

          // handle UI and quit conditions
          glfwPollEvents();
          int e = glGetError();
          if (e) { fprintf(stderr, "OpenGL Error %d\n", e); }
          frame++;

        }

        glfwTerminate();
        fprintf(stderr, "\n%d\n", frame);
        return 0;
        (void) argc;
        (void) argv;
      }
```

58 graphgrow3/video/graphgrow-video-test.cc

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```
#include <lo/lo.h>
#include <glm/glm.hpp>

struct control
{
  int32_t active;
  int32_t count    [4];
  int32_t source   [4][8];
  float   scale    [4][8];
  glm::mat3 transform[4][8];
};

int main()
{
  control C;
  memset(&C, 0, sizeof(C));
  for (int i = 0; i < 4; ++i)
    for (int j = 0; j < 8; ++j)
    {
      float s  = rand() / (double) RAND_MAX; s *= s;
      float r =  rand() / (double) RAND_MAX * 2 * 3.141592653589793;
      float tx = rand() / (double) RAND_MAX - 0.5;
      float ty = rand() / (double) RAND_MAX - 0.5;
      float co = cosf(r) * s;
      float si = sinf(r) * s;
      C.source   [i][j] = rand() % 4;
      C.scale    [i][j] = s;
      C.transform[i][j][0][0] = co;
      C.transform[i][j][0][1] = si;
      C.transform[i][j][0][2] = tx;
      C.transform[i][j][1][0] = -si;
      C.transform[i][j][1][1] = co;
      C.transform[i][j][1][2] = ty;
      C.transform[i][j][2][0] = 0;
      C.transform[i][j][2][1] = 0;
      C.transform[i][j][2][2] = 1;
      C.transform[i][j] = glm::inverse(C.transform[i][j]);
    }

  lo_address t = lo_address_new("255.255.255.255", "6061");

  while (1)
  {
    sleep(1);

    C.active = (C.active + 1) % 4;
    int i = rand() % 4;
    C.count[i] = (rand() % 8) + 1;
    int j = rand() % 8;
    float s  = rand() / (double) RAND_MAX; s *= s;
    float r =  rand() / (double) RAND_MAX * 2 * 3.141592653589793;
    float tx = rand() / (double) RAND_MAX - 0.5;
    float ty = rand() / (double) RAND_MAX - 0.5;
    float co = cosf(r) * s;
    float si = sinf(r) * s;
    C.source   [i][j] = rand() % 4;
    C.scale    [i][j] = s;
```

```
      C.transform[i][j][0][0] = co;
      C.transform[i][j][0][1] = si;
      C.transform[i][j][0][2] = tx;
      C.transform[i][j][1][0] = -si;
      C.transform[i][j][1][1] = co;
      C.transform[i][j][1][2] = ty;
      C.transform[i][j][2][0] = 0;
      C.transform[i][j][2][1] = 0;
      C.transform[i][j][2][2] = 1;
      C.transform[i][j] = glm::inverse(C.transform[i][j]);

      lo_blob blob = lo_blob_new(sizeof(C), &C);
      if (blob)
      {
        if (lo_send(t, "/video", "b", blob) == -1)
          fprintf(stderr, "OSC error %d: %s\n", lo_address_errno(t), ↙
              ↳ lo_address_errstr(t));
        lo_blob_free(blob);
      }
      else
        fprintf(stderr, "OSC error: couldn't lo_blob_new\n");

    }
    return 0;
}
```

59 graphgrow3/Video.hs

```
module Video (videoNew, Video) where

import Control.Monad (when, replicateM_, forM_, unless)
import Control.Monad.IO.Class (liftIO)

import System.IO (hPutStrLn, stderr)

import Data.IORef (IORef, newIORef, readIORef, writeIORef)

import Foreign (nullPtr, castPtr, peek, with, withArray, allocaBytes)
import Foreign.C (withCString, peekCStringLen)

import Data.Map (Map)
import qualified Data.Map.Strict as M

import qualified Graphics.UI.Gtk        as G
import qualified Graphics.UI.Gtk.OpenGL as G

import Graphics.GL

import Linear (M33)

import Paths_graphgrow3 (getDataFileName)
import Types

type Video = [(Int, Double, M33 Double)]

videoNew :: IO (Map RID Video) -> IO G.Window
videoNew transforms = do
```

```
    let w = 500
        h = 500
    window <- G.windowNew
    _ <- window `G.on` G.deleteEvent $ do
      liftIO $ G.widgetHide window
      return True
    G.windowSetDefaultSize window w h
    G.windowSetGeometryHints window (Nothing `asTypeOf` Just window) (Just (w, h))↙
        ↳ (Just (w, h)) Nothing Nothing Nothing
    canvas <- G.glDrawingAreaNew =<< G.glConfigNew [G.GLModeRGBA, G.GLModeDouble]
    G.widgetSetSizeRequest canvas w h
    let redraw ref = do
          _ <- G.withGLDrawingArea canvas $ \gl -> do
            trs <- transforms
            st <- readIORef ref
            when (vTrs st /= trs) $ do
              writeIORef ref st{ vTrs = trs }
              visualize ref
            when (M.null trs) $ do
              glClear GL_COLOR_BUFFER_BIT
            G.glDrawableSwapBuffers gl
          return True
    _ <- G.onRealize canvas $ do
      _ <- G.withGLDrawingArea canvas $ \gl -> do
        ref <- newIORef =<< initialize
        visualize ref
        G.glDrawableSwapBuffers gl
        G.timeoutAddFull (redraw ref) G.priorityDefaultIdle 100
      return ()
    G.set window [G.windowTitle G.:= "GG#V", G.containerChild G.:= canvas]
    G.widgetShowAll window
    return window

size :: Int
size = 2048

er :: Double
er = 4

data St = St
  { pInitial :: GLuint, pInitial'er, pInitial'rho :: GLint
  , pStep :: GLuint, pStep'er, pStep'count, pStep'ts, pStep'ws, pStep'ls, pStep'↙
      ↳ layer, pStep'src :: GLint
  , pColour :: GLuint, pColour'src, pColour'speed :: GLint
  , tPing, tPong :: GLuint
  , fBuffer :: GLuint
  , vTrs :: Map RID Video
  }

getUniformLocation :: GLuint -> String -> IO GLint
getUniformLocation program name = withCString name $ glGetUniformLocation ↙
    ↳ program

initialize :: IO St
initialize = do
  pInitial'     <- compileProgram (Just "initial.vert") (Just "initial.frag")
  pInitial'er'  <- getUniformLocation pInitial' "er"
```

```
    pInitial'rho' <- getUniformLocation pInitial' "rho"
    pStep'        <- compileProgram (Just "step.vert") (Just "step.frag")
    pStep'er'     <- getUniformLocation pStep' "er"
    pStep'count'  <- getUniformLocation pStep' "count"
    pStep'ts'     <- getUniformLocation pStep' "transform"
    pStep'ws'     <- getUniformLocation pStep' "source"
    pStep'ls'     <- getUniformLocation pStep' "scaleFactor"
    pStep'layer'  <- getUniformLocation pStep' "layer"
    pStep'src'    <- getUniformLocation pStep' "src"
    pColour'      <- compileProgram (Just "colour.vert") (Just "colour.frag")
    pColour'src'  <- getUniformLocation pColour' "src"
    pColour'speed'<- getUniformLocation pColour' "speed"
    tPing'        <- newTex size
    tPong'        <- newTex size
    fBuffer'      <- newFBO
    glClampColor GL_CLAMP_VERTEX_COLOR   (fromIntegral GL_FALSE)
    glClampColor GL_CLAMP_READ_COLOR     (fromIntegral GL_FALSE)
    glClampColor GL_CLAMP_FRAGMENT_COLOR (fromIntegral GL_FALSE)
    return St
      { pInitial = pInitial', pInitial'er = pInitial'er', pInitial'rho = pInitial'↙
          ↳ rho'
      , pStep = pStep', pStep'er = pStep'er', pStep'count = pStep'count', pStep'ts↙
          ↳  = pStep'ts', pStep'ws = pStep'ws', pStep'ls = pStep'ls', pStep'layer ↙
          ↳ = pStep'layer', pStep'src = pStep'src'
      , pColour = pColour', pColour'src = pColour'src', pColour'speed = pColour'↙
          ↳ speed'
      , tPing = tPing', tPong = tPong', fBuffer = fBuffer'
      , vTrs = M.empty
      }

clamp :: Ord a => a -> a -> a -> a
clamp mi ma x = mi `max` x `min` ma

visualize :: IORef St -> IO ()
visualize sR = do
  s0 <- readIORef sR
  glViewport 0 0 (fromIntegral size) (fromIntegral size)
  glLoadIdentity
  glOrtho 0 1 0 1 (-1) 1
  let nlayers = M.size (vTrs s0)
      layers = [0 .. fromIntegral nlayers - 1]
  forM_ layers $ \layer -> do
    bindFBO (fBuffer s0) (tPing s0) (fromIntegral layer)
    glUseProgram (pInitial s0)
    glUniform1f (pInitial'er s0) (realToFrac er)
    glUniform1f (pInitial'rho s0) (realToFrac (maximum (0:[ rho | Just (_, rho, ↙
        ↳ _) <- sequence $ M.lookup layer (vTrs s0) ])))
    unitQuad
    glUseProgram 0
    unbindFBO
    reportErrors "initialize layer"
  let rho = maximum (0:[ rho' | (_, rho', _) <- concat $ M.elems (vTrs s0) ])
      passes = clamp 16 4096 . round . logBase rho $ 0.001
  replicateM_ passes $ do
    s <- readIORef sR
    glBindTexture GL_TEXTURE_2D_ARRAY (tPing s)
    glUseProgram (pStep s)
```

```
      glUniform1f (pStep'er s) (realToFrac er)
      glUniform1i (pStep'src s) 0
      forM_ layers $ \layer -> case fmap (fromIntegral . length) $ M.lookup layer ↙
          ↳ (vTrs s) of
        Just count -> do
          bindFBO (fBuffer s) (tPong s) (fromIntegral layer)
          glUniform1i (pStep'count s) count
          withArray (concatMap (\(_,_,m) -> map realToFrac . concat . toLists $ m↙
              ↳ ) $ vTrs s M.! layer) $ glUniformMatrix3fv (pStep'ts s) count (↙
              ↳ fromIntegral GL_TRUE)
          withArray (map (\(_,l,_) -> realToFrac l) $ vTrs s M.! layer) $ ↙
              ↳ glUniform1fv (pStep'ls s) count
          withArray (map (\(i,_,_) -> fromIntegral i) $ vTrs s M.! layer) $ ↙
              ↳ glUniform1fv (pStep'ws s) count
          glUniform1f (pStep'layer s) (realToFrac layer)
          unitQuad
          unbindFBO
          reportErrors "step layer"
        Nothing -> return ()
      glUseProgram 0
      glBindTexture GL_TEXTURE_2D_ARRAY 0
      writeIORef sR s{ tPing = tPong s, tPong = tPing s }
      reportErrors "step pass"

    s <- readIORef sR
    glViewport 0 0 500 500
    glBindTexture GL_TEXTURE_2D_ARRAY (tPing s)
    glUseProgram (pColour s)
    glUniform1i (pColour'src s) 0
    glUniform1f (pColour'speed s) (2 * pi / fromIntegral passes)
    fullQuad
    glUseProgram 0
    glBindTexture GL_TEXTURE_2D_ARRAY 0
    reportErrors "draw"

newTex :: Int -> IO GLuint
newTex s = do
  let maxLayers = 4
  t <- with 0 $ \p -> glGenTextures 1 p >> peek p
  glBindTexture GL_TEXTURE_2D_ARRAY t
  glTexImage3D GL_TEXTURE_2D_ARRAY 0 (fromIntegral GL_R32F) (fromIntegral s) (↙
      ↳ fromIntegral s) maxLayers 0 GL_RED GL_UNSIGNED_BYTE nullPtr
  glTexParameteri GL_TEXTURE_2D_ARRAY GL_TEXTURE_MIN_FILTER (fromIntegral ↙
      ↳ GL_LINEAR)
  glTexParameteri GL_TEXTURE_2D_ARRAY GL_TEXTURE_MAG_FILTER (fromIntegral ↙
      ↳ GL_LINEAR)
  glBindTexture GL_TEXTURE_2D_ARRAY 0
  return t

newFBO :: IO GLuint
newFBO = with 0 $ \p -> glGenFramebuffers 1 p >> peek p

bindFBO :: GLuint -> GLuint -> GLint -> IO ()
bindFBO f t layer = do
  glBindFramebuffer GL_FRAMEBUFFER f
  glFramebufferTextureLayer GL_FRAMEBUFFER GL_COLOR_ATTACHMENT0 t 0 layer
  with GL_COLOR_ATTACHMENT0 $ glDrawBuffers 1
```

```
185
unbindFBO :: IO ()
unbindFBO = do
  glFramebufferTextureLayer GL_FRAMEBUFFER GL_COLOR_ATTACHMENT0 0 0 0
  glBindFramebuffer GL_FRAMEBUFFER 0
190
unitQuad :: IO ()
unitQuad = glDrawArrays GL_TRIANGLE_STRIP 0 4

fullQuad :: IO ()
195 fullQuad = glDrawArrays GL_TRIANGLE_STRIP 0 4

reportErrors :: String -> IO ()
reportErrors s = do
  e <- glGetError
200   when (e /= 0) $ do
    hPutStrLn stderr $ s ++ " OpenGL ERROR " ++ show e
    reportErrors s

compileProgram :: Maybe FilePath -> Maybe FilePath -> IO GLuint
205 compileProgram mV mF = do
  p <- glCreateProgram
  case mV of
    Nothing -> return ()
    Just v -> do
210       vert <- glCreateShader GL_VERTEX_SHADER
      source <- readFile =<< getDataFileName v
      shaderSource vert source
      glCompileShader vert
      glAttachShader p vert
215   case mF of
    Nothing -> return ()
    Just f -> do
      frag <- glCreateShader GL_FRAGMENT_SHADER
      source <- readFile =<< getDataFileName f
220       shaderSource frag source
      glCompileShader frag
      glAttachShader p frag
  glLinkProgram p
  n <- with (0 :: GLint) $ \q -> do
225     glGetProgramiv p GL_INFO_LOG_LENGTH q
    peek q
  l <- allocaBytes (fromIntegral n + 1) $ \q -> do
    glGetProgramInfoLog p (fromIntegral n) nullPtr q
    peekCStringLen (castPtr q, fromIntegral n)
230   unless (null l) $ do
    hPutStrLn stderr l
  return p

shaderSource :: GLuint -> String -> IO ()
235 shaderSource shader source = do
  withCString source $ \ptr -> with ptr $ \ptr' ->
    glShaderSource shader 1 (castPtr ptr') nullPtr
```

60 graphgrow3/video/Makefile

```
all: graphgrow-video graphgrow-video-test
```

```
clean:
	-rm graphgrow-video graphgrow-video-test

graphgrow-video: graphgrow-video.cc
	g++ -std=c++11 -Wall -Wextra -pedantic -Wno-variadic-macros -O3 -march=native -ffast-math -I$(HOME)/opt/include -L$(HOME)/opt/lib -o graphgrow-video graphgrow-video.cc -lGL -lGLEW -lglfw -llo

graphgrow-video-test: graphgrow-video-test.cc
	g++ -std=c++11 -Wall -Wextra -pedantic -Wno-variadic-macros -O3 -march=native -ffast-math -I$(HOME)/opt/include -L$(HOME)/opt/lib -o graphgrow-video-test graphgrow-video-test.cc -llo
```

61 graphgrow3/video/README

```
Needs liblo from https://github.com/radarsat1/liblo
Tested with HEAD at 276dc11db2038b44f04fe67cdb425929427a5337
```

62 graphgrow.cabal

```
name:                graphgrow
version:             2.1
synopsis:            graph-directed iterated function system fractals
description:
  graphgrow renders graph-directed iterated function system fractals.
  currently the iterated functions are based on text.
  .
  examples:
  .
  > echo "GRAPHGROW : GRAPHGROW" | graphgrow -
  > echo -e "HELLO : WORLD\nWORLD : HELLO" | graphgrow -
  > echo -e "HELLO : WORLD\nGOODBYE : WORLD\nWORLD : HELLO GOODBYE" | graphgrow -
  > for graph in data/*.gg ; do graphgrow $graph ; done

homepage:            https://code.mathr.co.uk/graphgrow
license:             GPL-3
license-file:        LICENSE
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
category:            Graphics
build-type:          Simple
cabal-version:       >=1.8

executable graphgrow
  main-is:  Fractal/GraphGrow/Main.hs
  other-modules:
      Fractal.GraphGrow.Analysis.Statistics
      Fractal.GraphGrow.Engine.Geometry
      Fractal.GraphGrow.Engine.Graph
      Fractal.GraphGrow.Engine.Grow
      Fractal.GraphGrow.Engine.Zoomer
      Fractal.GraphGrow.GUI.DrawPoints
      Fractal.GraphGrow.GUI.Editor
      Fractal.GraphGrow.GUI.FrameSync
```

```
        Fractal.GraphGrow.Text.Compile
        Fractal.GraphGrow.Text.Glyphs
        Fractal.GraphGrow.Text.Parse
        Fractal.GraphGrow.Utils

    build-depends:
      base >= 4.9 && < 4.10,
      BoundedChan >= 1.0 && < 1.1,
      containers >= 0.5 && < 0.6,
      glib >= 0.13 && < 0.14,
      gtk >= 0.14 && < 0.15,
      gtkglext >= 0.13 && < 0.14,
--      gtk-toy-diagrams >= 0.1,
      monad-supply >= 0.6 && < 0.7,
      mtl >= 2.2 && < 2.3,
      OpenGLRaw >= 3.2 && < 3.3,
      parallel >= 3.2 && < 3.3,
      random >= 1.1 && < 1.2,
      vector >= 0.11 && < 0.12

    ghc-options: -Wall -threaded -funbox-strict-fields -rtsopts -with-rtsopts "-
        ↪ A64M -N"

Source-repository head
  type:                 git
  location:             https://code.mathr.co.uk/graphgrow.git

Source-repository this
  type:                 git
  location:             https://code.mathr.co.uk/graphgrow.git
  tag:                  v2.1
```

63 LICENSE

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is 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.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  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.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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 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).

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 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.

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.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. 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 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 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 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

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 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

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.

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. Use with the GNU Affero General Public License.

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 Affero 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 special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of
the GNU 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 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 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 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 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 General Public License for more details.

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

Also add information on how to contact you by electronic and paper mail.

```
  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  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 GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

64 README

```
data/ has some larger example graphs
  bible-pg10.gg2    http://www.gutenberg.org/ebooks/10
  haskell-irc.gg2   http://tunes.org/~nef/logs/haskell/
  netbehaviour.gg2  http://www.netbehaviour.org/pipermail/netbehaviour/

extra/mk.hs can be used to extract markov chain word graphs from text
```

65 Setup.hs

```
import Distribution.Simple
main = defaultMain
```

66 the-sky-cracked-open/ChangeLog.md

```
# Revision history for the-sky-cracked-open

## 0.1.0.0  -- YYYY-mm-dd

* First version. Released on an unsuspecting world.
```

67 the-sky-cracked-open/colour.frag

```
#version 130
uniform sampler2DArray src;
uniform float er;
uniform float rho;
uniform float pixelSize;
uniform float speed;
```

```
void main() {
  vec2 p = gl_TexCoord[0].xy;
  vec2 z = texture(src, vec3(p, 0.0)).xy;
/*
  float zk = er * pow(1.0/rho, fract(z.x));
  float d = zk * log(zk) / z.y;
  float g = tanh(clamp(d / length(dFdx(p)), 0.0, 4.0));
  gl_FragData[0] = vec4(vec3(g), 1.0);
*/
  float n = z.x * speed;
  vec3 yuv;
  if (n < 0.0) {
    yuv = vec3(0.0);
  } else {
    yuv = vec3(clamp(log(1.0 + n / 4.0), 0.0, 1.0), 0.125 * sin(n), -0.125 * cos↙
        ↳ (n));
  }
  vec3 rgb = yuv * mat3(1.0, 0.0, 1.4, 1.0, -0.395, -0.581, 1.0, 2.03, 0.0);
  gl_FragData[0] = vec4(rgb, 1.0);
}
```

68 the-sky-cracked-open/initial.frag

```
#version 130
uniform float er;
uniform float rho;

void main() {
  vec2 p = er * (gl_TexCoord[0].xy * 2.0 - vec2(1.0));
  float l = length(p);
  float n;
  if (l >= er) {
    n = 0.0;
  } else if (er > l && l >= rho * er) {
    n = (log(er) - log(l)) / -log(rho);
  } else {
    n = -1.0;
  }
  gl_FragData[0] = vec4(n, 1.0, 0.0, 0.0);
}
```

69 the-sky-cracked-open/LICENSE

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.
```

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is 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. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

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.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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 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).

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 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.

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.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. 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 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 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 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

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 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

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.

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. Use with the GNU Affero General Public License.

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 Affero 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 special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU 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 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 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 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 General Public License as published by the Free Software Foundation, either version 3 of the License, or

```
640     (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
645     GNU General Public License for more details.

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

650 Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

655     <program>  Copyright (C) <year>  <name of author>
        This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
        This is free software, and you are welcome to redistribute it
        under certain conditions; type `show c' for details.

660 The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
665 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 GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
670 into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

70 the-sky-cracked-open/main.hs

```
{-# LANGUAGE PatternSynonyms #-}
import Control.Monad (when, replicateM_, forM_)
import System.Exit (exitSuccess)
import Graphics.UI.GLUT hiding (FramebufferObject)
5 import Graphics.GL
  ( glTexImage3D, pattern GL_TEXTURE_2D_ARRAY, pattern GL_RG32F, pattern GL_RG
  , pattern GL_UNSIGNED_BYTE, pattern GL_FALSE, glClampColor
  , pattern GL_CLAMP_VERTEX_COLOR, pattern GL_CLAMP_READ_COLOR, pattern ↙
    ↳ GL_CLAMP_FRAGMENT_COLOR
  , glGenFramebuffers, glBindFramebuffer, glFramebufferTextureLayer, ↙
    ↳ glTexParameteri
10  , pattern GL_FRAMEBUFFER, pattern GL_COLOR_ATTACHMENT0, glUniformMatrix3fv
  , GLuint, GLint, GLfloat, GLdouble, glUniform1fv, glBindTexture
  , pattern GL_TEXTURE_MIN_FILTER, pattern GL_TEXTURE_MAG_FILTER, pattern ↙
    ↳ GL_LINEAR
  )
import Foreign (alloca, peek, nullPtr, withArray)
15 import Data.IORef (IORef, newIORef, readIORef, writeIORef)
import Numeric.GSL.Root
```

```
import qualified Numeric.LinearAlgebra as L
import Numeric.LinearAlgebra hiding (R, dim, size)
import Unsafe.Coerce (unsafeCoerce)

import Shader (shader)
import Snapshot (writeSnapshot)

mXm :: L.Matrix R -> L.Matrix R -> L.Matrix R
mXm = (<>)

size :: Int
size = 2048

er :: R
er = 4

data St = St
  { pInitial :: Program, pInitial'er, pInitial'rho :: UniformLocation
  , pStep :: Program, pStep'er, pStep'count, pStep'ts, pStep'ws, pStep'ls, pStep'layer, pStep'src :: UniformLocation
  , pColour :: Program, pColour'src, pColour'er, pColour'rho, pColour'pixelSize, pColour'speed :: UniformLocation
  , tPing, tPong :: TextureObject
  , fBuffer :: FramebufferObject
  , vTrs :: [[L.Matrix R]], vLs :: [[R]], vT :: Int, vPasses :: Int, vRho :: (R, R)
  }

main :: IO ()
main = do
  _ <- getArgsAndInitialize
  initialWindowSize $= Size 1280 720
  _ <- createWindow "spiral!"
  pInitial'     <- shader Nothing (Just "initial.frag")
  pInitial'er'  <- get $ uniformLocation pInitial' "er"
  pInitial'rho' <- get $ uniformLocation pInitial' "rho"
  pStep'        <- shader Nothing (Just "step.frag")
  pStep'er'     <- get $ uniformLocation pStep' "er"
  pStep'count'  <- get $ uniformLocation pStep' "count"
  pStep'ts'     <- get $ uniformLocation pStep' "transform"
  pStep'ws'     <- get $ uniformLocation pStep' "source"
  pStep'ls'     <- get $ uniformLocation pStep' "scaleFactor"
  pStep'layer'  <- get $ uniformLocation pStep' "layer"
  pStep'src'    <- get $ uniformLocation pStep' "src"
  pColour'      <- shader Nothing (Just "colour.frag")
  pColour'src'  <- get $ uniformLocation pColour' "src"
  pColour'er'   <- get $ uniformLocation pColour' "er"
  pColour'rho'  <- get $ uniformLocation pColour' "rho"
  pColour'pixelSize' <- get $ uniformLocation pColour' "pixelSize"
  pColour'speed'<- get $ uniformLocation pColour' "speed"
  tPing'        <- newTex size
  tPong'        <- newTex size
  fBuffer'      <- newFBO
  glClampColor GL_CLAMP_VERTEX_COLOR   (fromIntegral GL_FALSE)
  glClampColor GL_CLAMP_READ_COLOR     (fromIntegral GL_FALSE)
  glClampColor GL_CLAMP_FRAGMENT_COLOR (fromIntegral GL_FALSE)
  sR <- newIORef St
```

```
      { pInitial = pInitial', pInitial'er = pInitial'er', pInitial'rho = pInitial'↙
        ↘ rho'
      , pStep = pStep', pStep'er = pStep'er', pStep'count = pStep'count', pStep'ts↙
        ↘  = pStep'ts', pStep'ws = pStep'ws', pStep'ls = pStep'ls', pStep'layer ↙
        ↘ = pStep'layer', pStep'src = pStep'src'
      , pColour = pColour', pColour'src = pColour'src', pColour'er = pColour'er', ↙
        ↘ pColour'rho = pColour'rho', pColour'pixelSize = pColour'pixelSize', ↙
        ↘ pColour'speed = pColour'speed'
      , tPing = tPing', tPong = tPong', fBuffer = fBuffer'
75    , vTrs = [[], []], vLs = [[], []], vT = 0, vPasses = 16, vRho = (0, 0)
      }
    reset sR
    displayCallback $= display sR
    addTimerCallback 100 (timer sR)
80  mainLoop

timer :: IORef St -> IO ()
timer sR = do
  addTimerCallback 100 (timer sR)
85  reset sR
  postRedisplay Nothing

reset :: IORef St -> IO ()
reset sR = do
90  s <- readIORef sR
--  when (vT s > 0) $ writeSnapshot (show (vT s) ++ ".ppm") (Position 0 0) (Size↙
    ↘  1280 720)
--  when (vT s >= 375) $ exitSuccess
  let dim1 = 1.001 + 0.75 * abs (sin(pi/2 * fromIntegral (vT s) / 30.0))
      dim2 = 1.001 + 0.75 * abs (sin(pi/2 * fromIntegral (vT s) / 37.5))
95    spiral1 = spiral dim1
      spiral2 = spiral dim2
      ls = [map fst spiral1, map fst spiral2]
      rhos = concat ls
      mrhos = map maximum ls
100   rho = (minimum rhos, maximum rhos)
      trs1 = map snd spiral1
      trs2 = map snd spiral2
      passes = clamp 16 4096 . round . logBase (snd rho) $ 0.001
  writeIORef sR s{ vTrs = [trs1, trs2], vLs = ls, vT = vT s + 1, vPasses = ↙
    ↘ passes, vRho = rho }
105 viewport $= (Position 0 0, Size (fromIntegral size) (fromIntegral size))
  loadIdentity
  ortho2D 0 1 0 1
  forM_ [0,1] $ \layer -> do
    bindFBO (fBuffer s) (tPing s) (fromIntegral layer)
110   currentProgram $= Just (pInitial s)
    uniform (pInitial'er s) $= TexCoord1 (realToFrac er :: GLfloat)
    uniform (pInitial'rho s) $= TexCoord1 (realToFrac (mrhos !! layer) :: ↙
      ↘ GLfloat)
    unitQuad
    currentProgram $= Nothing
115   unbindFBO

clamp :: Ord a => a -> a -> a -> a
clamp mi ma x = mi `max` x `min` ma
```

```
120 display :: IORef St -> IO ()
    display sR = do
      s0 <- readIORef sR
      replicateM_ (vPasses s0) $ do
        s <- readIORef sR
125     viewport $= (Position 0 0, Size (fromIntegral size) (fromIntegral size))
        glBindTexture GL_TEXTURE_2D_ARRAY (case tPing s of TextureObject t -> t)
        currentProgram $= Just (pStep s)
        uniform (pStep'er s) $= TexCoord1 (realToFrac er :: GLfloat)
        uniform (pStep'src s) $= TexCoord1 (0 :: GLint)
130     forM_ [0,1] $ \layer -> do
          bindFBO (fBuffer s) (tPong s) (fromIntegral layer)
          uniform (pStep'count s) $= TexCoord1 (3 :: GLint)
          withArray (map (realToFrac :: R -> GLfloat) . concat . concatMap L.toLists↙
              ↪ . (!! layer) . vTrs $ s) $ glUniformMatrix3fv (unsafeCoerce $ pStep↙
              ↪ 'ts s) 3 1
          withArray (map (realToFrac :: R -> GLfloat) . (!! layer) . vLs $ s) $ ↙
              ↪ glUniform1fv (unsafeCoerce $ pStep'ls s) 3
135       withArray ([[0, 1, 0], [1, 0, 1 :: GLfloat]] !! layer) $ glUniform1fv (↙
              ↪ unsafeCoerce $ pStep'ws s) 3
          uniform (pStep'layer s) $= TexCoord1 (realToFrac layer :: GLfloat)
          unitQuad
          unbindFBO
        currentProgram $= Nothing
140     glBindTexture GL_TEXTURE_2D_ARRAY 0
        writeIORef sR s{ tPing = tPong s, tPong = tPing s }

      s <- readIORef sR
      viewport $= (Position 0 0, Size 1280 720)
145   glBindTexture GL_TEXTURE_2D_ARRAY (case tPong s of TextureObject t -> t)
      currentProgram $= Just (pColour s)
      uniform (pColour'src s) $= TexCoord1 (0 :: GLint)
      uniform (pColour'er s) $= TexCoord1 (realToFrac er :: GLfloat)
      uniform (pColour'rho s) $= TexCoord1 (realToFrac (fst $ vRho s) :: GLfloat)
150   uniform (pColour'pixelSize s) $= TexCoord1 (0.1 :: GLfloat)
      uniform (pColour'speed s) $= TexCoord1 (2 * pi / fromIntegral (vPasses s) :: ↙
          ↪ GLfloat)
      fullQuad
      currentProgram $= Nothing
      glBindTexture GL_TEXTURE_2D_ARRAY 0
155   swapBuffers
      reportErrors

    newTex :: Int -> IO TextureObject
    newTex s = do
160   [TextureObject t] <- genObjectNames 1
      glBindTexture GL_TEXTURE_2D_ARRAY t
      glTexImage3D GL_TEXTURE_2D_ARRAY 0 (fromIntegral GL_RG32F) (fromIntegral s) (↙
          ↪ fromIntegral s) 2 0 GL_RG GL_UNSIGNED_BYTE nullPtr
      glTexParameteri GL_TEXTURE_2D_ARRAY GL_TEXTURE_MIN_FILTER (fromIntegral ↙
          ↪ GL_LINEAR)
      glTexParameteri GL_TEXTURE_2D_ARRAY GL_TEXTURE_MAG_FILTER (fromIntegral ↙
          ↪ GL_LINEAR)
165   glBindTexture GL_TEXTURE_2D_ARRAY 0
      return (TextureObject t)

    newtype FramebufferObject = FramebufferObject GLuint
```

```
newFBO :: IO FramebufferObject
newFBO = fmap FramebufferObject (alloca $ \p -> glGenFramebuffers 1 p >> peek p)

bindFBO :: FramebufferObject -> TextureObject -> GLint -> IO ()
bindFBO (FramebufferObject f) (TextureObject t) layer = do
  glBindFramebuffer GL_FRAMEBUFFER f
  glFramebufferTextureLayer GL_FRAMEBUFFER GL_COLOR_ATTACHMENT0 t 0 layer

unbindFBO :: IO ()
unbindFBO = do
  glFramebufferTextureLayer GL_FRAMEBUFFER GL_COLOR_ATTACHMENT0 0 0 0
  glBindFramebuffer GL_FRAMEBUFFER 0

unitQuad :: IO ()
unitQuad = renderPrimitive Quads $ do
    t 0 1 >> v 0 1
    t 0 0 >> v 0 0
    t 1 0 >> v 1 0
    t 1 1 >> v 1 1
  where
    t, v :: GLdouble -> GLdouble -> IO ()
    t x y = texCoord (TexCoord2 x y)
    v x y = vertex (Vertex2 x y)

fullQuad :: IO ()
fullQuad = renderPrimitive Quads $ do
    t (0.5-tx) (0.5+ty) >> v 0 1
    t (0.5-tx) (0.5-ty) >> v 0 0
    t (0.5+tx) (0.5-ty) >> v 1 0
    t (0.5+tx) (0.5+ty) >> v 1 1
  where
    tx = 0.5 / er
    ty = 0.5 / er * 9 / 16
    t, v :: GLdouble -> GLdouble -> IO ()
    t x y = texCoord (TexCoord2 x y)
    v x y = vertex (Vertex2 x y)

type R = Double

transformRST :: R -> R -> L.Vector R -> L.Matrix R
transformRST a l p = (3><3)[ c, s, x, -s, c, y, 0, 0, 1 ]
  where x:y:_ = toList p
        c = l * cos a
        s = l * sin a

findZero :: (R -> R) -> R -> R
findZero f = head . fst . root Hybrids 1e-12 1000 ((:[]) . f . head) . (:[])

spiral :: R -> [(R, L.Matrix R)]
spiral d = [(l1, inv t1), (l2, inv t2), (l3, inv t3)]
  where
    f l = 2 * sqrt ((1 - l * l) / 4) ** d + l ** d - 1
    l1 = sin (acos l2) / 2
    l2 = findZero f (d - 1)
    l3 = l1
    a1 = pi / 2 - acos l2
```

```
    a2 = -acos l2
    a3 = a1
    x1 = 0
    y1 = 0
    x2 =  l1 * cos a1
    y2 = -l1 * sin a1
    x3 = 1 - x2
    y3 = 0 - y2
    v1 = 2 |> [ x1, y1 ]
    v2 = 2 |> [ x2, y2 ]
    v3 = 2 |> [ x3, y3 ]
    t1 = post `mXm` transformRST a1 l1 v1 `mXm` pre
    t2 = post `mXm` transformRST a2 l2 v2 `mXm` pre
    t3 = post `mXm` transformRST a3 l3 v3 `mXm` pre
    pre  = transformRST 0 1 (2|>[ 0.5,0])
    post = transformRST 0 1 (2|>[-0.5,0])
```

71 the-sky-cracked-open/Setup.hs

```
import Distribution.Simple
main = defaultMain
```

72 the-sky-cracked-open/Shader.hs

```
module Shader (shader) where

import Graphics.Rendering.OpenGL

shader :: Maybe FilePath -> Maybe FilePath -> IO Program
shader mV mF = do
  p <- createProgram
  vs <- case mV of
    Nothing -> return []
    Just v -> do
      vert <- createShader VertexShader
      source <- readFile v
      shaderSource vert $= [ source ]
      compileShader vert
      print =<< get (shaderInfoLog vert)
      return [vert]
  fs <- case mF of
    Nothing -> return []
    Just f -> do
      frag <- createShader FragmentShader
      source <- readFile f
      shaderSource frag $= [ source ]
      compileShader frag
      print =<< get (shaderInfoLog frag)
      return [frag]
  attachedShaders p $= (vs ++ fs)
  linkProgram p
  print =<< get (programInfoLog p)
  return p
```

73 the-sky-cracked-open/Snapshot.hs

```
module Snapshot (hSnapshot, writeSnapshot, snapshotWith) where

import Control.Monad(forM_)
import System.IO(Handle())
import Graphics.Rendering.OpenGL(
  readPixels,
  Position,
  Size(Size),
  PixelData(PixelData),
  PixelFormat(RGB),
  DataType(UnsignedByte))
import Foreign.Marshal.Alloc(allocaBytes)
import Foreign.Ptr(plusPtr)
import qualified Data.ByteString.Internal as BSI
import qualified Data.ByteString as BS

-- save a screenshot as binary PPM
snapshotWith :: (BS.ByteString -> IO b) -> Position -> Size -> IO b
snapshotWith f p0 vp@(Size vw vh) = do
  let fi q = fromIntegral q
      p6 = "P6\n" ++ show vw ++ " " ++ show vh ++ " 255\n"
  allocaBytes (fi (vw*vh*3)) $ \ptr -> do
    readPixels p0 vp $ PixelData RGB UnsignedByte ptr
    px <- BSI.create (fi $ vw * vh * 3) $ \d -> forM_ [0..vh-1] $ \y ->
      BSI.memcpy
        (d `plusPtr` fi(y*vw*3))
        (ptr `plusPtr` fi ((vh-1-y)*vw*3))
        (fi(vw*3))
    f $ BS.pack (map (toEnum . fromEnum) p6) `BS.append` px

hSnapshot :: Handle -> Position -> Size -> IO ()
hSnapshot h = snapshotWith (BS.hPutStr h)

writeSnapshot :: FilePath -> Position -> Size -> IO ()
writeSnapshot f = snapshotWith (BS.writeFile f)
```

74 the-sky-cracked-open/step.frag

```
#version 130
#define MAXCOUNT 16
uniform float er;
uniform int count;
uniform mat3 transform[MAXCOUNT];
uniform float scaleFactor[MAXCOUNT];
uniform float source[MAXCOUNT];
uniform float layer;

uniform sampler2DArray src;

void main() {
  vec2 p0 = er * (gl_TexCoord[0].xy * 2.0 - vec2(1.0));
  float escape = -1.0;
  float factor = 1.0;
  for (int i = 0; i < count && i < MAXCOUNT; ++i) {
    vec3 p = transform[i] * vec3(p0, 1.0);
    vec2 q = p.xy / p.z;
    float l = length(q);
```

```
      if (l < er) {
        vec2 z = texture(src, vec3((q / er + vec2(1.0)) / 2.0, source[i])).xy;
        if (escape < z.x) { escape = z.x; factor = 1.0/scaleFactor[i] * z.y; }
      }
    }
    if (escape >= 0.0) {
      escape += 1.0;
    }
    vec2 z = texture(src, vec3((p0 / er + vec2(1.0)) / 2.0, layer)).xy;
    if (escape > z.x) {
      z.x = escape;
      z.y = factor;
    }
    gl_FragData[0] = vec4(z, 0.0, 0.0);
}
```

75 the-sky-cracked-open/the-sky-cracked-open.cabal

```
name:                the-sky-cracked-open
version:             0.1.0.0
synopsis:            A short fractal video loop rendered with continuous escape time for an iterated function system.
homepage:            https://mathr.co.uk/blog/2011-12-31_the_sky_cracked_open.html
license:             GPL-3
license-file:        LICENSE
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
category:            Demo
build-type:          Simple
extra-source-files:  ChangeLog.md
cabal-version:       >=1.10

executable the-sky-cracked-open
  main-is:             main.hs
  other-modules:       Shader Snapshot
  build-depends:       base >=4.9 && <4.10, bytestring >=0.10 && <0.11, OpenGL >= 3.0 && < 3.1, OpenGLRaw >= 3.2 && < 3.3, GLUT >= 2.7 && < 2.8, hmatrix >= 0.18 && < 0.19, hmatrix-gsl >= 0.18 && < 0.19
  default-language:    Haskell2010
  other-extensions:    PatternSynonyms
```