diary

Claude Heiland-Allen

2018

Contents

1 cover.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

// gcc -std=c99 -Wall -Wextra -pedantic -O3 -o cover cover.c -lgsl -lgslcblas -↙
    ↳ lm
// ./cover W H C N F > cover.ppm

// popen()
#define _POSIX_C_SOURCE 2

#include <assert.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include <gsl/gsl_sf_zeta.h>
```

```
/*
// http://lolengine.net/blog/2013/07/27/rgb-to-hsv-in-glsl\n"
vec3 hsv2rgb(vec3 c) {
  vec4 K = vec4(1.0, 2.0 / 3.0, 1.0 / 3.0, 3.0);
  vec3 p = abs(fract(c.xxx + K.xyz) * 6.0 - K.www);
  return c.z * mix(K.xxx, clamp(p - K.xxx, 0.0, 1.0), c.y);
}
*/
double fract(double x) { return x - floor(x); }
double clamp(double x, double mi, double ma) { return fmax(mi, fmin(x, ma)); }
double mix(double a, double b, double x) { return a * (1 - x) + x * b; }
void hsv2rgb(double *r, double *g, double *b, double h, double s, double v)
{
  double K[4] = { 1.0, 2.0 / 3.0, 1.0 / 3.0, 3.0 };
  double p[3] = { fabs(fract(h + K[0]) * 6 - K[3]), fabs(fract(h + K[1]) * 6 - K[3]), fabs(fract(h + K[2]) * 6 - K[3]) };
  double q[3] = { clamp(p[0] - K[0], 0.0, 1.0), clamp(p[1] - K[0], 0.0, 1.0), clamp(p[2] - K[0], 0.0, 1.0) };
  double c[3] = { v * mix(K[0], q[0], s), v * mix(K[0], q[1], s), v * mix(K[0], q[2], s) };
  *r = c[0];
  *g = c[1];
  *b = c[2];
}

struct histogram
{
  int64_t levels;
  unsigned char **counts;
};

static inline void h_set(struct histogram *h, int64_t l, int64_t y, int64_t x, uint64_t v)
{
  int64_t d = 1LL << l;
  int64_t bytes_per_side = (d + 7) >> 3;
  int64_t bytes = d * bytes_per_side;
  int64_t byte = x >> 3;
  int64_t bit  = x & 7;
  int64_t mask = ~(1 << bit);
  int64_t k0 = y * bytes_per_side + byte;
  int64_t bs = 2 * (h->levels - l) + 1;
  for (int64_t b = 0; b < bs; ++b)
  {
    int64_t k = b * bytes + k0;
    h->counts[l][k] &= mask;
    h->counts[l][k] |= ((v >> b) & 1) << bit;
  }
}

static inline uint64_t h_get(struct histogram *h, int64_t l, int64_t y, int64_t x)
{
  int64_t d = 1LL << l;
  int64_t bytes_per_side = (d + 7) >> 3;
  int64_t bytes = d * bytes_per_side;
```

```
  int64_t byte = x >> 3;
  int64_t bit  = x & 7;
  int64_t mask = (1 << bit);
  int64_t k0 = y * bytes_per_side + byte;
  int64_t bs = 2 * (h->levels - l) + 1;
  uint64_t v = 0;
  for (int64_t b = 0; b < bs; ++b)
  {
    int64_t k = b * bytes + k0;
    v |= ((h->counts[l][k] & mask) >> bit) << b;
  }
  return v;
}

struct histogram *h_new(int64_t width, int64_t height)
{
  int64_t levels = 0;
  int64_t size = 1LL << levels;
  while (size < width || size < height)
  {
    levels += 1;
    size <<= 1;
  }
  struct histogram *h = malloc(sizeof(struct histogram));
  h->levels = levels;
  h->counts = malloc(sizeof(char *) * (levels + 1));
  for (int64_t l = 0; l <= levels; ++l)
  {
    int64_t d = 1LL << l;
    int64_t bytes_per_side = (d + 7) / 8;
    int64_t bytes = d * bytes_per_side;
    int64_t bits = 2 * (levels - l) + 1;
    h->counts[l] = malloc(bits * bytes);
  }
  for (int64_t y = 0; y < size; ++y)
    for (int64_t x = 0; x < size; ++x)
      h_set(h, levels, y, x, y < height && x < width);
  for (int64_t l = levels - 1; l >= 0; --l)
    for (int64_t y = 0; y < 1 << l; ++y)
      for (int64_t x = 0; x < 1 << l; ++x)
        h_set(h, l, y, x,
            h_get(h, l + 1, (y << 1) + 0, (x << 1) + 0)
          + h_get(h, l + 1, (y << 1) + 0, (x << 1) + 1)
          + h_get(h, l + 1, (y << 1) + 1, (x << 1) + 0)
          + h_get(h, l + 1, (y << 1) + 1, (x << 1) + 1)
        );
  assert(h_get(h, 0, 0, 0) == (uint64_t) width * height);
  return h;
}

void h_free(struct histogram *h)
{
  for (int64_t l = 0; l <= h->levels; ++l)
    free(h->counts[l]);
  free(h->counts);
  free(h);
}
```

```
130
    void h_decrement(struct histogram *h, int64_t x, int64_t y)
    {
      for (int64_t l = h->levels; l >= 0; --l)
      {
135     uint64_t v = h_get(h, l, y, x);
        if (v > 0) { v -= 1; }
        else {
          assert(l == h->levels);
          break;
140     }
        h_set(h, l, y, x, v);
        x >>= 1;
        y >>= 1;
      }
145 }

    int h_empty(struct histogram *h)
    {
      return h_get(h, 0, 0, 0) == 0;
150 }

    int h_choose(struct histogram *h, int64_t *x, int64_t *y)
    {
      if (h_empty(h)) return 0;
155   *x = 0;
      *y = 0;
      for (int64_t l = 1; l <= h->levels; ++l)
      {
        *x <<= 1;
160     *y <<= 1;
        int64_t xs[4] = { *x, *x + 1, *x, *x + 1 };
        int64_t ys[4] = { *y, *y, *y + 1, *y + 1 };
        uint64_t ss[4] =
          { h_get(h, l, ys[0], xs[0])
165       , h_get(h, l, ys[1], xs[1])
          , h_get(h, l, ys[2], xs[2])
          , h_get(h, l, ys[3], xs[3])
          };
        uint64_t ts[4] =
170       { ss[0]
          , ss[0] + ss[1]
          , ss[0] + ss[1] + ss[2]
          , ss[0] + ss[1] + ss[2] + ss[3]
          };
175     uint64_t p = rand() % ts[3];
        int i;
        for (i = 0; i < 4; ++i)
          if (p < ts[i]) break;
        *x = xs[i];
180     *y = ys[i];
      }
      return 1;
    }

185 struct image {
      int64_t width;
```

```
  int64_t height;
  int64_t bytes_per_line;
  unsigned char *data;
};

struct image *i_new(int64_t width, int64_t height)
{
  struct image *i = calloc(1, sizeof(struct image));
  i->width = width;
  i->height = height;
  i->bytes_per_line = (width + 3) >> 2;
  i->data = calloc(1, i->bytes_per_line * height);
  return i;
}

void i_set(struct image *i, int64_t y, int64_t x, uint64_t v)
{
  int64_t byte = i->bytes_per_line * y + (x >> 2);
  int64_t bit = (x & 3) << 1;
  int64_t mask = ~(3 << bit);
  i->data[byte] &= mask;
  i->data[byte] |= (v & 3) << bit;
}

uint64_t i_get(const struct image *i, int64_t y, int64_t x)
{
  int64_t byte = i->bytes_per_line * y + (x >> 2);
  int64_t bit = (x & 3) << 1;
  int64_t mask = (3 << bit);
  return (i->data[byte] & mask) >> bit;
}

struct image *image;

struct histogram *initialize(int64_t width, int64_t height)
{
  image = i_new(width, height);
  return h_new(width, height);
}

double rectangle_radius(double ai, double aspect) { double t = atan(aspect); ↙
    ↳ return sqrt(ai / (4 * cos(t) * sin(t))); }

unsigned char gpgm[2][4096][4096];
double gx, gy, gr, ga, gf;

void load_glyph(void)
{
    FILE *f = popen("convert mask.png -background white -alpha remove pgm:-", "r↙
        ↳ ");
    fscanf(f, "P5\n4096 4096\n255");
    assert(fgetc(f) == '\n');
    fread(&gpgm[0][0][0], 4096 * 4096, 1, f);
    pclose(f);
    f = popen("convert shape.png -background white -alpha remove pgm:-", "r");
    fscanf(f, "P5\n4096 4096\n255");
    assert(fgetc(f) == '\n');
```

```
    fread(&gpgm[1][0][0], 4096 * 4096, 1, f);
    pclose(f);
    // compute glyph bounds
    int minx = 4095, miny = 4095, maxx = 0, maxy = 0, w = 0;
    for (int y = 0; y < 4096; ++y)
    for (int x = 0; x < 4096; ++x)
    if (gpgm[0][y][x] == 0)
    {
      minx = x < minx ? x : minx;
      miny = y < miny ? y : miny;
      maxx = x > maxx ? x : maxx;
      maxy = y > maxy ? y : maxy;
      ++w;
    }
    gx = maxx - minx;
    gy = maxy - miny;
    gr = 0.5 * hypot(gx, gy);
    ga = gx / gy;
    gx *= 0.5;
    gy *= 0.5;
    gx += minx;
    gy += miny;
    gf = 3.141592653589793 * gr * gr / w;
}

int draw_glyph(int draw, struct histogram *h, int width, int height, double cx, ↙
    ↳ double cy, double r, double co, double si, int c, int b, int s)
{
  double r2 = r * r;
  int64_t x0 = floor(cx - r);
  int64_t x1 = ceil (cx + r);
  int64_t y0 = floor(cy - r);
  int64_t y1 = ceil (cy + r);
  for (int64_t y = y0; y <= y1; ++y)
  {
    for (int64_t x = x0; x <= x1; ++x)
    {
      double dx = x - cx;
      double dy = y - cy;
      double d2 = dx * dx + dy * dy;
      if (d2 <= r2)
      {
        int64_t ix = gx + (co * dx - si * dy) * gr / r;
        int64_t iy = gy + (si * dx + co * dy) * gr / r;
        if (0 <= ix && ix < 4096 && 0 <= iy && iy < 4096 && gpgm[0][iy][ix] == ↙
            ↳ 0)
        {
          if (draw)
          {
            int64_t u = x, v = y;
            if (0 <= u && u < width && 0 <= v && v < height) { h_decrement(h, u,↙
                ↳  v); i_set(image, v, u, c); }
            if (u < 2 * b - s) u += width - (2 * b - s);
            if (0 <= u && u < width && 0 <= v && v < height) { h_decrement(h, u,↙
                ↳  v); i_set(image, v, u, c); }
            if (u >= width - (2 * b - s)) u -= width - (2 * b - s);
            if (0 <= u && u < width && 0 <= v && v < height) { h_decrement(h, u,↙
```

```
                ↪  v); i_set(image, v, u, c); }
295             if (v < 2 * b - s) { u -= width / 2; u = -u; u += width / 2; v -= b ↙
                ↪ - s / 2; v = -v; v += b - s / 2; }
                if (0 <= u && u < width && 0 <= v && v < height) { h_decrement(h, u,↙
                ↪  v); i_set(image, v, u, c); }
                if (v >= height - (2 * b - s)) { u -= width / 2; u = -u; u += width ↙
                ↪ / 2; v -= height - (b - s / 2); v = -v; v += height - (b - s /↙
                ↪  2); }
                if (0 <= u && u < width && 0 <= v && v < height) { h_decrement(h, u,↙
                ↪  v); i_set(image, v, u, c); }
              }
300           else
              {
                int64_t u = x, v = y;
                if (0 <= u && u < width && 0 <= v && v < height && i_get(image, v, u↙
                ↪ )) return 0;
                if (u < 2 * b - s) u += width - (2 * b - s);
305             if (0 <= u && u < width && 0 <= v && v < height && i_get(image, v, u↙
                ↪ )) return 0;
                if (u >= width - (2 * b - s)) u -= width - (2 * b - s);
                if (0 <= u && u < width && 0 <= v && v < height && i_get(image, v, u↙
                ↪ )) return 0;
                if (v < 2 * b - s) { u -= width / 2; u = -u; u += width / 2; v -= b ↙
                ↪ - s / 2; v = -v; v += b - s / 2; }
                if (0 <= u && u < width && 0 <= v && v < height && i_get(image, v, u↙
                ↪ )) return 0;
310             if (v >= height - (2 * b - s)) { u -= width / 2; u = -u; u += width ↙
                ↪ / 2; v -= height - (b - s / 2); v = -v; v += height - (b - s /↙
                ↪  2); }
                if (0 <= u && u < width && 0 <= v && v < height && i_get(image, v, u↙
                ↪ )) return 0;
              }
            }
          }
315     }
  }
  return 1;
}

320 double area_factor(double A, double c, double N)
{
  return A / gsl_sf_hzeta(c, N);
}

325 double area_i(double Af, double c, double N, int i)
{
  return Af / pow(i + N, c);
}

330 void packing(struct histogram *h, int width, int height, double c, double N, ↙
    ↪ double fill)
{
  int64_t i = 1;
  int border = 170;
  int spine = 290;
335   draw_glyph(1, h, width, height, width - height / 2 / ga - spine, height / 2, ↙
      ↪ height / 2, 0, 1, 1, border, spine);
```

```
  double r1 = spine * 4 * 2./3 / ga;
  int w = r1 * 2 * 5./4;
  for (int y1 = ((height / 2) % w); y1 < height; y1 += w)
    draw_glyph(1, h, width, height, width / 2, y1, r1, 0, -1, 1, border, spine);
  double A = (width - 2 * border) * (height - 2 * border) - (height / 2) * (height / 2) / gf;
  double Af = area_factor(A, c, N);
  double a = 0;
  time_t last = time(0);
  while (a < fill * A && ! h_empty(h))
  {
    double Ai = area_i(Af, c, N, i++);
    a += Ai;
    double r = rectangle_radius(Ai * gf, ga);
    if (r < 1) break;
    uint64_t jj = sqrt(h_get(h, 0, 0, 0));
    for (uint64_t j = 0; j < jj; ++j)
    {
      time_t now = time(0);
      if (now >= last + 1)
      {
        fprintf(stderr, "%16ld\t%f\t%f\t%16lu\r", i, a / (fill * A), r, j);
        last = now;
      }
      int64_t x = -1, y = -1;
      if (h_choose(h, &x, &y))
      {
        double t = rand() / (double) RAND_MAX * 6.283185307179586;
        double co = cos(t);
        double si = sin(t);
        if (draw_glyph(0, h, width, height, x, y, r, co, si, (i % 2) + 2, border, spine))
        {
          draw_glyph(1, h, width, height, x, y, r, co, si, (i % 2) + 2, border, spine);
          break;
        }
      }
      else
      {
        break;
      }
    }
  }
  fprintf(stderr, "\n");
}

void write_ppm(int64_t width, int64_t height, int64_t seed)
{
  double hue = 0.9;
  double sat = 1;
  double val = 1;
  double red, grn, blu;
  hsv2rgb(&red, &grn, &blu, hue, sat, val);
  unsigned char rc = 255 * red;
  unsigned char gc = 255 * grn;
  unsigned char bc = 255 * blu;
```

```
390   unsigned char rgb[4][3] = { { rc, gc, bc }, { rc, gc, bc }, { 255, 255, 255 },↙
        ↳ { 0, 0, 0 } };
      unsigned char *scanline = malloc(3 * width);
      fprintf(stdout, "P6\n%ld %ld\n# %ld\n255\n", width, height, seed);
      for (int64_t y = 0; y < height; ++y)
      {
395     for (int64_t x = 0; x < width; ++x)
        {
          uint64_t c = i_get(image, y, x) & 3;
          scanline[3 * x + 0] = rgb[c][0];
          scanline[3 * x + 1] = rgb[c][1];
400       scanline[3 * x + 2] = rgb[c][2];
        }
        fwrite(scanline, width * 3, 1, stdout);
      }
      fflush(stdout);
405 }

    int main(int argc, char **argv)
    {
      if (argc < 6) return 1;
410   int width = atoi(argv[1]);
      int height = atoi(argv[2]);
      uint64_t npixels = (uint64_t) width * height;
      assert(npixels < 1LU << 31LU);
      double c = atof(argv[3]);
415   double N = atof(argv[4]);
      double fill = atof(argv[5]);
      uint64_t seed = time(0);
      if (argc > 6) seed = atol(argv[6]);
      srand(seed);
420   struct histogram *h = initialize(width, height);
      load_glyph();
      packing(h, width, height, c, N, fill);
      write_ppm(width, height, seed);
      return 0;
425 }
```

2 cover-page.tex

```
% diary -- generate printable diary PDF
% Copyright (C) 2018  Claude Heiland-Allen
% License GPL3+ <http://www.gnu.org/licenses/>

5   \documentclass{article}
    \usepackage[paperwidth=16cm,paperheight=16cm,left=0cm,right=0cm,top=0cm,bottom=0↙
      ↳ cm]{geometry}
    \usepackage[T1]{fontenc}
    \usepackage{lmodern}
    \usepackage{graphicx}
10  \setlength{\parindent}{0mm}
    \renewcommand{\familydefault}{\sfdefault}
    \pagenumbering{gobble}

    \begin{document}
15  \hspace{0pt}
    \vfill
```

```
\resizebox{\textwidth}{!}{\Huge {\bf{}2019}}
\vfill
\hspace{0pt}
\end{document}
```

3 cover-spec.txt

	cm	1200dpi
Spine Width:	0.618cm	290px
Cover Width:	30.88cm	14588px
Cover Height:	21.622575cm	10216px
X Bleed Begins:	0.36cm	170px
X Spine Begins:	15.132275cm	7149px
X Spine Ends	15.749275cm	7439px
X Bleed Ends:	30.52cm	14588px
Y Bleed Begins:	0.36cm	170px
Y Bleed Ends:	21.262575cm	10046px

4 days.txt

```
1 1 New Year's Day
1 6 Epiphany
1 25 Robert Burns' Night
2 14 Valentine's Day
3 1 St David
3 5 Shrove Tuesday
3 6 Ash Wednesday
3 17 St Patrick
3 31 Mother's Day / British Summer Time Begins
4 19 Good Friday
4 21 Easter Sunday
4 22 Easter Monday
4 23 St George
5 6 Early May Bank Holiday
5 27 Spring Bank Holiday
6 16 Father's Day
8 26 Summer Bank Holiday
10 27 British Summer Time Ends
10 31 Halloween
11 10 Remembrance Sunday
11 30 St Andrew
12 25 Christmas Day
12 26 Boxing Day
12 31 New Year's Eve
```

5 diary.sh

```
#!/bin/bash

# diary -- generate printable diary PDF
# Copyright (C) 2018  Claude Heiland-Allen
# License GPL3+ <http://www.gnu.org/licenses/>

(
cat <<EOF
\\documentclass{article}
```

```
\\usepackage[paperwidth=6.08in,paperheight=8.51in,margin=20mm,right=40mm]{geometry}
\\usepackage{tabularx}
\\usepackage{booktabs}
\\usepackage{multirow}
\\usepackage{adjustbox}
\\usepackage{lmodern}
\\renewcommand{\\familydefault}{\\sfdefault}

\\newcolumntype{C}{ >{\\centering\\arraybackslash} m{11mm} }
\\newcolumntype{R}{ >{\\raggedleft\\arraybackslash} X }

\\begin{document}
\\pagestyle{empty}
EOF
(
  echo 2018 12 31 0 365 0
  weekday=1
  month=12
  yearday=1
  for days in 31 28 31 30  31 30 31 31  30 31 30 31
  do
    month=$(((month % 12) + 1))
    for day in $(seq 1 $days)
    do
      echo 2019 $month $day $weekday $yearday $((365 - yearday))
      weekday=$(((weekday + 1) % 7))
      yearday=$((yearday + 1))
    done
  done
  day=1
  month=1
  yearday=1
  while ((weekday != 0))
  do
    echo 2020 $month $day $weekday $yearday $((365 - yearday))
    weekday=$(((weekday + 1) % 7))
    yearday=$((yearday + 1))
    day=$((day + 1))
  done
) |
cat -n |
(
  week=1
  line1=0
  while [ "$line1" != "" ]
  do
    read line1 year1 month1 day1 weekday1 yearday1 endday1
    read line2 year2 month2 day2 weekday2 yearday2 endday2
    read line3 year3 month3 day3 weekday3 yearday3 endday3
    read line4 year4 month4 day4 weekday4 yearday4 endday4
    read line5 year5 month5 day5 weekday5 yearday5 endday5
    read line6 year6 month6 day6 weekday6 yearday6 endday6
    read line7 year7 month7 day7 weekday7 yearday7 endday7
    if [ "$line1" == "" ]
    then
      break
```

```
    fi
    special1=$(grep "^$month1 $day1 " < days.txt | cut -d\  -f 3-)
    special2=$(grep "^$month2 $day2 " < days.txt | cut -d\  -f 3-)
    special3=$(grep "^$month3 $day3 " < days.txt | cut -d\  -f 3-)
    special4=$(grep "^$month4 $day4 " < days.txt | cut -d\  -f 3-)
    special5=$(grep "^$month5 $day5 " < days.txt | cut -d\  -f 3-)
    special6=$(grep "^$month6 $day6 " < days.txt | cut -d\  -f 3-)
    special7=$(grep "^$month7 $day7 " < days.txt | cut -d\  -f 3-)
    case $month1 in
      1) monthname1=January;;
      2) monthname1=February;;
      3) monthname1=March;;
      4) monthname1=April;;
      5) monthname1=May;;
      6) monthname1=June;;
      7) monthname1=July;;
      8) monthname1=August;;
      9) monthname1=September;;
      10) monthname1=October;;
      11) monthname1=November;;
      12) monthname1=December;;
    esac
    case $month7 in
      1) monthname7=January;;
      2) monthname7=February;;
      3) monthname7=March;;
      4) monthname7=April;;
      5) monthname7=May;;
      6) monthname7=June;;
      7) monthname7=July;;
      8) monthname7=August;;
      9) monthname7=September;;
      10) monthname7=October;;
      11) monthname7=November;;
      12) monthname7=December;;
    esac
    if ((month1 != month7))
    then
      extramonthyear=" / $monthname7 $year7"
    else
      extramonthyear="\\phantom{/}"
    fi
    caption=$(cat -n images.txt | grep " $week[[:space:]]" | sed "s/^.*pgm ↙
        ↪ \(.*\)$/\1/g")
    cat << EOF
\\noindent\\begin{tabularx}{\textwidth}[t]{C @{\\extracolsep{\\fill}} R}
\\multicolumn{2}{l}{\\Large $monthname1 $year1$extramonthyear} \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day1} & \\multirow[t]{3}{*}{\\normalsize ↙
    ↪ $special1} \\\\ \\Large Mon & \\\\ \\scriptsize ($yearday1-$endday1) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day2} & \\multirow[t]{3}{*}{\\normalsize ↙
    ↪ $special2} \\\\ \\Large Tue & \\\\ \\scriptsize ($yearday2-$endday2) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day3} & \\multirow[t]{3}{*}{\\normalsize ↙
    ↪ $special3} \\\\ \\Large Wed & \\\\ \\scriptsize ($yearday3-$endday3) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day4} & \\multirow[t]{3}{*}{\\normalsize ↙
```

```
    ↪ $special4} \\\\ \\Large Thu & \\\\ \\scriptsize ($yearday4-$endday4) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day5} & \\multirow[t]{3}{*}{\\normalsize ↙
    ↪ $special5} \\\\ \\Large Fri & \\\\ \\scriptsize ($yearday5-$endday5) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day6} & \\multirow[t]{3}{*}{\\normalsize ↙
    ↪ $special6} \\\\ \\Large Sat & \\\\ \\scriptsize ($yearday6-$endday6) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\Huge {\\bf $day7} & \\multirow[t]{3}{*}{\\normalsize ↙
    ↪ $special7} \\\\ \\Large Sun & \\\\ \\scriptsize ($yearday7-$endday7) & ↙
    ↪ \\\\
\\vfill & \\\\ \\midrule \\normalsize Week \newline $(((week - 1)%52+1)) & \\↙
    ↪ normalsize $caption
\\end{tabularx}
\\newpage
EOF
    week=$((week + 1))
  done
)
cat <<EOF
\\topskip0pt
\\vspace*{\\fill}
\\centering
\\normalsize mathr.co.uk/diary/2019
\\end{document}
EOF
) > diary.tex
pdflatex diary.tex
pdflatex title-page.tex
make
./platonic-solids
./square-log-polar-graph-paper
./triangle-log-polar-graph-paper
./square-conformal-graph-paper
./triangle-conformal-graph-paper
./log-log-graph-paper
./htile
for i in *.pgm
do
  pnmtotiff < $i > $i.tif
  tiff2pdf -x 600 -y 600 -z -o $i.tif.pdf $i.tif
done
images=$(cat images.txt | cut -d\  -f 1 | sed "s/$/.tif.pdf/g")
pdftk title-page.pdf $images cat output images.pdf
pdftk A=images.pdf B=diary.pdf shuffle A B output 2019-print.pdf
gs -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/↙
    ↪ screen -dColorImageDownsampleType=/Bicubic -dGrayImageDownsampleType=/↙
    ↪ Bicubic -dMonoImageDownsampleType=/Bicubic -sOutputFile=2019-web.pdf 2019-↙
    ↪ print.pdf
```

6 .gitignore

```
platonic-solids
square-log-polar-graph-paper
triangle-log-polar-graph-paper
htile
log-log-graph-paper
```

```
square-conformal-graph-paper
triangle-conformal-graph-paper
cover
*.pgm
*.ppm
*.png
*.pdf
*.aux
*.log
*.tif
diary.tex
-
```

7 htile.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

#include <stdio.h>
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) "#version 130\n" #s

static const int width = 3648;
static const int height = 5106;
static const float size = 2.0;

static const char *frag_src = GLSL(

float cosh(float val)
{
        float tmp = exp(val);
        float cosH = (tmp + 1.0 / tmp) / 2.0;
        return cosH;
}

float tanh(float val)
{
        float tmp = exp(val);
        float tanH = (tmp - 1.0 / tmp) / (tmp + 1.0 / tmp);
        return tanH;
}

float sinh(float val)
{
        float tmp = exp(val);
        float sinH = (tmp - 1.0 / tmp) / 2.0;
        return sinH;
}

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 cPower(vec2 z, float n) {
        float r2 = dot(z,z);
        return pow(r2,n/2.0)*vec2(cos(n*atan(z.y, z.x)),sin(n*atan(z.y,z.x)));
}



vec2 cInverse(vec2 a) {
        return  vec2(a.x,-a.y)/dot(a,a);
}

vec2 cDiv(vec2 a, vec2 b) {
        return cMul( a,cInverse(b));
}

vec2 cExp(vec2 z) {
        return vec2(exp(z.x) * cos(z.y), exp(z.x) * sin(z.y));
}

vec2 cLog(vec2 a) {
        float b =  atan(a.y,a.x);
        if (b>0.0) b-=2.0*3.1415;
        return vec2(log(length(a)),b);
}

vec2 cSqr(vec2 z) {
        return vec2(z.x*z.x-z.y*z.y,2.*z.x*z.y);
}

vec2 cSin(vec2 z) {
        return vec2(sin(z.x)*cosh(z.y), cos(z.x)*sinh(z.y));
}

vec2 cCos(vec2 z) {
        return vec2(cos(z.x)*cosh(z.y), -sin(z.x)*sinh(z.y));
}

vec2 cPower2(vec2 z, vec2 a) {
        return cExp(cMul(cLog(z), a));
}

const float pi = 3.141592653;

uniform int p;
uniform int q;

float opp = acosh((cos(pi/float(q)) + cos(pi / float(p)))/ sin(pi/float(p)));
float adj = acosh(cos(pi/float(p)) / sin(pi/float(q)));
float hyp = acosh(cos(pi/float(p)) * cos(pi/float(q)) / (sin(pi/float(p)) * sin(↙
    ↳ pi/float(q))));

vec2 face = vec2(0.0, 0.0);
vec2 edge = vec2(adj, 0.0);
vec2 vertex = vec2(hyp * cos(pi / q), hyp * sin(pi / q));
```

```
float edist(vec2 z, vec2 w) { return length(z - w); }
float hdist(vec2 z, vec2 w) { return acosh(1.0 + 2.0 * dot(z - w, z - w) / ((1.0 - dot(z, z)) * (1.0 - dot(w, w)))); }
float h2e(float z) { return sqrt((cosh(z) - 1.0) / (cosh(z) + 1.0)); }
vec2 h2e(vec2 z) { return normalize(z) * h2e(length(z)); }
float e2h(float z) { return hdist(vec2(z, 0.0), vec2(0.0, 0.0)); }
vec2 e2h(vec2 z) { return normalize(z) * e2h(length(z)); }

vec4 moebius(vec2 z) { return vec4(z, 1.0, 0.0); }
vec2 unmoebius(vec4 z) { return cDiv(z.xy, z.zw); }
float mlength(vec4 z) { return length(z.xy) / length(z.zw); }

mat4 rotation(float a) {
  float c = cos(a);
  float s = sin(a);
  return mat4
    ( c, -s, 0.0, 0.0
    , s,  c, 0.0, 0.0
    , 0.0, 0.0, 1.0, 0.0
    , 0.0, 0.0, 0.0, 1.0
    );
}

mat4 translation(vec2 z) {
  float s = atan(z.y, z.x);
  float l = length(z);
  float e = exp(l);
  float f = e + 1.0;
  float g = e - 1.0;
  mat4 m = mat4
    ( f, 0.0, g, 0.0
    , 0.0, f, 0.0, g
    , g, 0.0, f, 0.0
    , 0.0, g, 0.0, f
    );
  if (l > 0.0) {
    return rotation(-s) * m * rotation(s);
  } else {
    return m;
  }
}

mat4 rotationAbout(vec2 z, float a) { return translation(z) * rotation(a) * translation(-z); }

vec3 circleBetween(vec2 p, vec2 q) {
  float ax = p.x;
  float ay = p.y;
  float a2 = ax * ax + ay * ay;
  float bx = q.x;
  float by = q.y;
  float b2 = bx * bx + by * by;
  float cx = ax / a2;
  float cy = ay / a2;
  float c2 = cx * cx + cy * cy;
  float d = 2.0 * (ax * (by - cy) + bx * (cy - ay) + cx * (ay - by));
```

```
  float ux = (a2*(by - cy) + b2*(cy - ay) + c2*(ay - by)) / d;
  float uy = (a2*(cx - bx) + b2*(ax - cx) + c2*(bx - ax)) / d;
  float dx = ux - q.x;
  float dy = uy - q.y;
  float r = sqrt(dx * dx + dy * dy);
  return vec3(ux, uy, r);
}


// hyperbolic area D(r) = 4 pi sinh^2(r/2)
// angle deficit = - area
// pi = 4 pi sinh^2 (r / 2)
// r = 2 asinh (1/2)
// 2.0 pi / 3 = ... => r = 2 asinh(sqrt(1/6))
void main() {
  vec2 zz = gl_TexCoord[0].yx;
  vec2 z = zz.yx;
  vec2 z0 = vec2(0.0, 0.0);
  vec2 z1 = unmoebius
        ( translation(vertex) * rotation(pi/3.0) * translation(vertex)
        * translation(edge) * translation(edge)
        * moebius(z0)
        );
  float theta = atan(z1.x, z1.y);
  mat4 transformation[32];
  int k = 0;
  for (int f = 0; f < q; ++f) {
    vec2 vert = vec2(hyp * cos(pi * float(2 * f + 1) / float(q)), hyp * sin(pi *↙
        ↪  float(2 * f + 1) / float(q)));
    for (int v = 1; v < p; ++v) {
      transformation[k++] = rotationAbout(vert, 2.0 * pi * float(v) / float(p));
    }
  }
  float v = 1.0;
  if (abs(z.x) < pi/4.0) {
    z = cDiv(cSin(z), cCos(z));
    if (length(z) >= 1.0) { gl_FragColor = vec4(1.0); return; }
    vec4 m = rotation(theta) * moebius(z);
    for (int i = 0; i <12; ++i) {
      if (mlength(m) <= h2e(adj)) { break; }
      vec4 mm = m;
      bool done = true;
      for (int j = 0; j < 32 && j < k; ++j) {
        vec4 mmm = transformation[j] * m;
        if (mlength(mmm) < mlength(mm)) {
          mm = mmm;
          done = false;
        }
      }
      m = normalize((mm));
      if (done) { break; }
    }
    vec2 w = unmoebius(m);
    float a = atan(w.y, w.x) * float(2 * q) / (2.0 * pi);
    float b = atan(w.y, w.x) * float(1 * q) / (2.0 * pi);
    float d = (1.0 + abs(tan(2.0 * zz.y))) * 0.0002 / length(w);
    a -= floor(a);
```

```
    b -= floor(b);
    a = min(a, 1.0 - a);
    b = min(b, 1.0 - b);
    a = min(a, b);
    v = smoothstep(2.0 * d, 4.0 * d, a);
  }
  gl_FragColor = vec4(0.5 + 0.5 * v);
}

);

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 = 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 = 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);
  }
}
```

```
int main(int argc, char **argv) {

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutCreateWindow("graphpaper");
  glewInit();

  GLint success;
  int prog = glCreateProgram();
  int frag = glCreateShader(GL_FRAGMENT_SHADER);
  glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
  glCompileShader(frag);
  glAttachShader(prog, frag);
  glLinkProgram(prog);
  glGetProgramiv(prog, GL_LINK_STATUS, &success);
  if (!success)
  {
    debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
    debug_program(prog);
    exit(1);
  }
  glUseProgram(prog);
  GLuint up = glGetUniformLocation(prog, "p");
  GLuint uq = glGetUniformLocation(prog, "q");

  GLuint tex;
  glGenTextures(1, &tex);
  glBindTexture(GL_TEXTURE_2D, tex);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, ↙
      ↳ GL_UNSIGNED_BYTE, 0);
  glBindTexture(GL_TEXTURE_2D, 0);

  GLuint fbo;
  glGenFramebuffers(1, &fbo);
  glBindFramebuffer(GL_FRAMEBUFFER, fbo);
  glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, ↙
      ↳ tex, 0);

  glViewport(0, 0, width, height);
  glLoadIdentity();
  gluOrtho2D(0, 1, 1, 0);

  unsigned char *buffer = malloc(width * height);
  int pq[9][2] =
    { { 3, 7 }
    , { 3, 8 }
    , { 4, 5 }
    , { 4, 6 }
    , { 5, 4 }
    , { 5, 5 }
    , { 6, 4 }
    , { 7, 3 }
    , { 8, 3 }
    };
  for (int k = 0; k < 9; ++k) {
      glUniform1i(up, pq[k][0]);
```

```
        glUniform1i(uq, pq[k][1]);
        glBegin(GL_QUADS); {
          float u = size / 2.0;
          float w = u * height / width / 0.825223;
          glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
          glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
          glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
          glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
        } glEnd();
        glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
        char fname[200];
        snprintf(fname, 100, "htile-%d-%d.pgm", pq[k][0], pq[k][1]);
        FILE *f = fopen(fname, "wb");
        fprintf(f, "P5\n%d %d\n255\n", width, height);
        fflush(f);
        fwrite(buffer, width * height, 1, f);
        fflush(f);
        fclose(f);
    }

    free(buffer);
    glDeleteFramebuffers(1, &fbo);
    glDeleteTextures(1, &tex);
    glDeleteShader(frag);
    glDeleteProgram(prog);
    glutReportErrors();
    return 0;
}
```

8 images.txt

```
platonic-3-0-0.pgm Edge-Centered Tetrahedra (Stereographic)
square-0-1.pgm Square Polar Graph Paper \((1,0) \times 2\)
htile-3-7.pgm \(\{3,7\}\) Hyperbolic Tiling
triangle-conformal-2-0.pgm Triangular Confocal Ellipses + Hyperbolas \(\times 2\)
log-log-0.pgm Normal Distribution Probability Paper
square-conformal-3-2.pgm Square Self-Confocal Quartic \(\times 3\)
triangle-5-1.pgm Triangular Polar Graph Paper \((1,1) \times 3\)
platonic-3-0-1.pgm Edge-Centered Tetrahedra (Equirectangular)
htile-3-8.pgm \(\{3,8\}\) Hyperbolic Tiling
square-1-1.pgm Square Polar Graph Paper \((1,1) \times 2\)
triangle-conformal-2-1.pgm Triangular Confocal Parabolas \(\times 2\)
platonic-3-1-0.pgm Vertex-Centered Tetrahedra (Stereographic)
square-2-1.pgm Square Polar Graph Paper \((2,1) \times 2\)
htile-4-5.pgm \(\{4,5\}\) Hyperbolic Tiling
triangle-conformal-2-2.pgm Triangular Self-Confocal Cubic \(\times 2\)
platonic-3-1-1.pgm Vertex-Centered Tetrahedra (Equirectangular)
square-3-1.pgm Square Polar Graph Paper \((3,2) \times 2\)
htile-4-6.pgm \(\{4,6\}\) Hyperbolic Tiling
triangle-conformal-3-0.pgm Triangular Confocal Ellipses + Hyperbolas \(\times 3\)
log-log-1.pgm Semi-Log \(y\) Graph Paper
triangle-6-1.pgm Triangular Polar Graph Paper \((1,2) \times 3\)
platonic-4-0-0.pgm Face-Centered Cube (Stereographic)
triangle-0-1.pgm Triangular Polar Graph Paper \((1,0) \times 2\)
square-4-1.pgm Square Polar Graph Paper \((1,0) \times 3\)
```

```
htile-5-4.pgm \(\{5,4\}\) Hyperbolic Tiling
square-5-1.pgm Square Polar Graph Paper \((1,1) \times 3\)
triangle-conformal-3-1.pgm Triangular Confocal Parabolas \(\times 3\)
platonic-4-0-1.pgm Face-Centered Cube (Equirectangular)
triangle-1-1.pgm Triangular Polar Graph Paper \((1,1) \times 2\)
htile-5-5.pgm \(\{5,5\}\) Hyperbolic Tiling
triangle-conformal-3-2.pgm Triangular Self-Confocal Cubic \(\times 3\)
square-6-1.pgm Square Polar Graph Paper \((2,1) \times 3\)
platonic-4-1-0.pgm Edge-Centered Cube (Stereographic)
square-conformal-2-0.pgm Square Confocal Ellipses + Hyperbolas \(\times 2\)
triangle-2-1.pgm Triangular Polar Graph Paper \((1,2) \times 2\)
htile-6-4.pgm \(\{6,4\}\) Hyperbolic Tiling
square-conformal-2-1.pgm Square Confocal Parabolas \(\times 2\)
log-log-2.pgm Semi-Log \(x\) Graph Paper
square-7-1.pgm Square Polar Graph Paper \((3,2) \times 3\)
platonic-4-1-1.pgm Edge-Centered Cube (Equirectangular)
triangle-3-1.pgm Triangular Polar Graph Paper \((2,3) \times 2\)
htile-7-3.pgm \(\{7,3\}\) Hyperbolic Tiling
square-conformal-2-2.pgm Square Self-Confocal Quartic \(\times 2\)
platonic-4-2-0.pgm Vertex-Centered Cube (Stereographic)
triangle-4-1.pgm Triangular Polar Graph Paper \((1,0) \times 3\)
htile-8-3.pgm \(\{8,3\}\) Hyperbolic Tiling
square-conformal-3-0.pgm Square Confocal Ellipses + Hyperbolas \(\times 3\)
platonic-4-2-1.pgm Vertex-Centered Cube (Equirectangular)
square-conformal-3-1.pgm Square Confocal Parabolas \(\times 3\)
platonic-5-0-0.pgm Rhombic Icosahedral (Stereographic)
triangle-7-1.pgm Triangular Polar Graph Paper \((2,3) \times 3\)
log-log-3.pgm Log-Log \(xy\) Graph Paper
platonic-5-0-1.pgm Rhombic Icosahedral (Equirectangular)
```

9 LICENSE.md

```
### GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc.
<https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

### Preamble

The GNU 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 <https://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 <https://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 <https://www.gnu.org/licenses/why-not-lgpl.html>.
```

10 log-log-graph-paper.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

#include <stdio.h>
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) #s

static const int width = 3648;
static const int height = 5106;
static const float size = 5.0;

static const char *frag_src = GLSL(

  const float pi = 3.141592653589793;

  float normal_pdf(float x) { return exp(-x*x/2.0) / sqrt(2.0 * pi); }
  float normal_cdf(float x) { // Zelen & Severo (1964)
    const float b0 = 0.2316419;
    const float b1 = 0.319381530;
    const float b2 = -0.356563782;
    const float b3 = 1.781477937;
    const float b4 = -1.821255978;
    const float b5 = 1.330274429;
```

```
    float t = 1.0 / (1.0 + b0 * x);
    return 1.0 - normal_pdf(x) * (t * (b1 + t * (b2 + t * (b3 + t * (b4 + t * b5↙
        ↳ )))));
  }

  uniform vec2 base;

  void main() {
    vec2 q = gl_TexCoord[0].xy;
    q *= mat2(0.0, -1.0, 1.0, 0.0);
    float d = length(vec4(dFdx(q), dFdy(q)));
    vec2 q0 = q;
    q -= floor(q);
    vec2 b = pow(base, q);
    b = floor(b);
    b = log(b) / log(base);
    if (base.x == 1.0) b.x = floor(8.0 * q.x) / 8.0;
    if (base.x == 1.0 && base.y == 1.0) { q.y = 4.0 * normal_cdf(abs(q0.y)); d =↙
        ↳ length(vec4(dFdx(q), dFdy(q))); }
    if (base.y == 1.0) b.y = floor(8.0 * q.y) / 8.0;
    vec2 e = abs(q - b);
    float f = min(min(e.x, e.y), min(1.0 - e.x, 1.0 - e.y));
    gl_FragColor = vec4(vec3(0.5 + 0.5 * smoothstep(2.0 * d, 4.0 * d, f)), 1.0);
  }

);

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 = 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 = 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);
  }
}

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

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutCreateWindow("graphpaper");
  glewInit();

  GLint success;
  int prog = glCreateProgram();
  int frag = glCreateShader(GL_FRAGMENT_SHADER);
  glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
  glCompileShader(frag);
  glAttachShader(prog, frag);
  glLinkProgram(prog);
  glGetProgramiv(prog, GL_LINK_STATUS, &success);
  if (!success)
  {
    debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
    debug_program(prog);
    exit(1);
  }
  glUseProgram(prog);
  GLuint ubase = glGetUniformLocation(prog, "base");

  GLuint tex;
  glGenTextures(1, &tex);
  glBindTexture(GL_TEXTURE_2D, tex);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, GL_UNSIGNED_BYTE, 0);
  glBindTexture(GL_TEXTURE_2D, 0);

  GLuint fbo;
  glGenFramebuffers(1, &fbo);
  glBindFramebuffer(GL_FRAMEBUFFER, fbo);
  glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, tex, 0);

  glViewport(0, 0, width, height);
  glLoadIdentity();
  gluOrtho2D(0, 1, 1, 0);
```

```
140
  unsigned char *buffer = malloc(width * height);
  float base[4][2] = { { 1, 1 }, { 1, 10 }, { 10, 1 }, { 10, 10 } };
  for (int k = 0; k < 4; ++k) {
        glUniform2fv(ubase, 1, &base[k][0]);
145     glBegin(GL_QUADS); {
          float u = size / 2.0;
          float w = u * height / width / 0.825223;
          glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
          glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
150       glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
          glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
        } glEnd();
        glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
        char fname[200];
155     snprintf(fname, 100, "log-log-%d.pgm", k);
        FILE *f = fopen(fname, "wb");
        fprintf(f, "P5\n%d %d\n255\n", width, height);
        fflush(f);
        fwrite(buffer, width * height, 1, f);
160     fflush(f);
        fclose(f);
  }

  free(buffer);
165 glDeleteFramebuffers(1, &fbo);
  glDeleteTextures(1, &tex);
  glDeleteShader(frag);
  glDeleteProgram(prog);
  glutReportErrors();
170 return 0;
}
```

11 Makefile

```
# diary -- generate printable diary PDF
# Copyright (C) 2018  Claude Heiland-Allen
# License GPL3+ <http://www.gnu.org/licenses/>

5  all: cover platonic-solids triangle-log-polar-graph-paper square-log-polar-graph↙
     ↳ -paper square-conformal-graph-paper triangle-conformal-graph-paper log-log↙
     ↳ -graph-paper htile

cover: cover.c
        gcc -std=c99 -Wall -Wextra -pedantic -O3 -o cover cover.c -lgsl -↙
            ↳ lgslcblas -lm -fopenmp

10 platonic-solids: platonic-solids.c
        gcc -std=c99 -Wall -pedantic -Wextra platonic-solids.c -o platonic-↙
            ↳ solids -lGLEW -lGL -lGLU -lglut -lm

triangle-log-polar-graph-paper: triangle-log-polar-graph-paper.c
        gcc -std=c99 -Wall -pedantic -Wextra triangle-log-polar-graph-paper.c -o↙
            ↳  triangle-log-polar-graph-paper -lGLEW -lGL -lGLU -lglut -lm
15
square-log-polar-graph-paper: square-log-polar-graph-paper.c
        gcc -std=c99 -Wall -pedantic -Wextra square-log-polar-graph-paper.c -o ↙
```

```
            ↳ square-log-polar-graph-paper -lGLEW -lGL -lGLU -lglut

square-conformal-graph-paper: square-conformal-graph-paper.c
        gcc -std=c99 -Wall -pedantic -Wextra square-conformal-graph-paper.c -o ↙
            ↳ square-conformal-graph-paper -lGLEW -lGL -lGLU -lglut

triangle-conformal-graph-paper: triangle-conformal-graph-paper.c
        gcc -std=c99 -Wall -pedantic -Wextra triangle-conformal-graph-paper.c -o↙
            ↳  triangle-conformal-graph-paper -lGLEW -lGL -lGLU -lglut

log-log-graph-paper: log-log-graph-paper.c
        gcc -std=c99 -Wall -pedantic -Wextra log-log-graph-paper.c -o log-log-↙
            ↳ graph-paper -lGLEW -lGL -lGLU -lglut

htile: htile.c
        gcc -std=c99 -Wall -pedantic -Wextra htile.c -o htile -lGLEW -lGL -lGLU ↙
            ↳ -lglut
```

12 platonic-solids.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

#include <math.h>
#include <stdio.h>
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) "#version 120\n" #s

static const int width = 3648;
static const int height = 5106;
static const float size = 5.0;

static const char *frag_src = GLSL(

  const float pi = 3.141592653589793;

  const float phi  = 1.618033988749895;
  const float phi1 = 0.618033988749895;
  vec3 a = -normalize(vec3(-phi,phi-1.0,1.0));
  vec3 b = -normalize(vec3(1.0,-phi,phi+1.0));
  vec3 c = -normalize(vec3(0.0,0.0,-1.0));

  uniform int polyhedron;
  uniform int orientation;
  uniform bool equirectangular;

  vec3 stereo(vec2 x) {
    return vec3(2.0 * x, dot(x, x) - 1.0) / (dot(x, x) + 1.0);
  }
```

```
vec3 equir(vec2 x) {
  return vec3(cos(x.x) * sin(x.y), cos(x.x) * cos(x.y), sin(x.x));
}

void main() {
  vec2 p = gl_TexCoord[0].xy;
  vec3 q;
  if (equirectangular)
  {
    if (abs(p.x) <= pi/2.0 && abs(p.y) <= pi)
    {
      q = equir(p);
      if (orientation < 2 && polyhedron == 4)
      {
        float t = pi / 2.0; q *= mat3(cos(t), -sin(t), 0.0,  sin(t), cos(t), 0.0,  0.0, 0.0, 1.0);
      }
    }
    else
    {
      gl_FragColor = vec4(1.0);
      return;
    }
  }
  else
  {
    q = stereo(1.5 * p);
  }
  if (orientation > 1 && polyhedron == 4)
  {
    float t = pi / 2.0; q *= mat3(cos(t), -sin(t), 0.0,  sin(t), cos(t), 0.0,  0.0, 0.0, 1.0);
    t = atan(1.0, sqrt(2.0)); q *= mat3(1.0, 0.0, 0.0,  0.0, cos(t), -sin(t),  0.0, sin(t), cos(t));
  }
  if (orientation > 0 && polyhedron == 4)
  {
    float t = pi / 4.0; q *= mat3(cos(t), 0.0, -sin(t),  0.0, 1.0, 0.0,  sin(t), 0.0, cos(t));
  }
  if (orientation > 0 && polyhedron == 3)
  {
    float t = pi / 2.0; q *= mat3(cos(t), -sin(t), 0.0,  sin(t), cos(t), 0.0,  0.0, 0.0, 1.0);
    t = atan(1.0, sqrt(2.0)); q *= mat3(1.0, 0.0, 0.0,  0.0, cos(t), -sin(t),  0.0, sin(t), cos(t));
    t = pi / 4.0; q *= mat3(cos(t), 0.0, -sin(t),  0.0, 1.0, 0.0,  sin(t),  0.0, cos(t));
  }
  if (orientation > 1 && polyhedron == 5)
  {
    float t = asin(2.0/3.0)/2.0; q *= mat3(cos(t), -sin(t), 0.0,  sin(t), cos(t),  0.0,  0.0, 0.0, 1.0);
  }
  if (orientation > 0 && polyhedron == 5)
  {
    float t = atan(2.0)/2.0; q *= mat3(1.0, 0.0, 0.0,  0.0, cos(t), -sin(t),
```

```
        ↳ 0.0, sin(t), cos(t));
    }
    float dx = length(dFdx(q));
    float dy = length(dFdy(q));
    float d = min(dx, dy);
    float e = 0.0;
    if (polyhedron == 3 || polyhedron == 4)
    {
      q = abs(q);
      float r = max(q.x, max(q.y, q.z));
      q /= r;
      float mi = min(min(q.x, q.y), q.z);
      float ma = 1.0;
      float md = q.x + q.y + q.z - mi - ma;
      if (polyhedron == 3) e = min(md - mi, ma - md);
      if (polyhedron == 4) e = min(mi, ma - md);
    }
    else if (polyhedron == 5)
    {
      float t;
      q = abs(q);
      t = dot(q, a); if (t < 0.0) q -= 2.0 * t * a;
      t = dot(q, b); if (t < 0.0) q -= 2.0 * t * b;
      t = dot(q, c); if (t < 0.0) q -= 2.0 * t * c;
      t = dot(q, b); if (t < 0.0) q -= 2.0 * t * b;
      q = abs(q);
      t = dot(q, a); if (t < 0.0) q -= 2.0 * t * a;
      t = dot(q, b); if (t < 0.0) q -= 2.0 * t * b;
      t = dot(q, c); if (t < 0.0) q -= 2.0 * t * c;
      t = dot(q, b); if (t < 0.0) q -= 2.0 * t * b;

      vec3 f = vec3(length(q - a), length(q - b), length(q - c));
      float ma = max(max(f.x, f.y), f.z);
      float mi = min(min(f.x, f.y), f.z);
      float md = f.x + f.y + f.z - ma - mi;
      e = ma - md;
    }
    gl_FragColor = vec4(vec3(0.5 + 0.5 * smoothstep(2.0 * d, 4.0 * d, e)), 1.0);
  }

);

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 = 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 = 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);
  }
}

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

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutCreateWindow("graphpaper");
  glewInit();

  GLint success;
  int prog = glCreateProgram();
  int frag = glCreateShader(GL_FRAGMENT_SHADER);
  glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
  glCompileShader(frag);
  glAttachShader(prog, frag);
  glLinkProgram(prog);
  glGetProgramiv(prog, GL_LINK_STATUS, &success);
  if (!success)
  {
    debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
    debug_program(prog);
    exit(1);
  }
  glUseProgram(prog);
  GLuint upolyhedron = glGetUniformLocation(prog, "polyhedron");
  GLuint uorientation = glGetUniformLocation(prog, "orientation");
  GLuint uequirectangular = glGetUniformLocation(prog, "equirectangular");
```

```
200     GLuint tex;
        glGenTextures(1, &tex);
        glBindTexture(GL_TEXTURE_2D, tex);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, ↙
            ↳ GL_UNSIGNED_BYTE, 0);
        glBindTexture(GL_TEXTURE_2D, 0);
205
        GLuint fbo;
        glGenFramebuffers(1, &fbo);
        glBindFramebuffer(GL_FRAMEBUFFER, fbo);
        glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, ↙
            ↳ tex, 0);
210
        glViewport(0, 0, width, height);
        glLoadIdentity();
        gluOrtho2D(0, 1, 1, 0);

215     unsigned char *buffer = malloc(width * height);
        for (int p = 3; p <= 5; ++p) {
          for (int o = 0; o < 3; ++o) {
            if (p == 3 && o == 2)
              continue;
220         if (p == 5 && o > 0)
              continue;
            for (int l = 0; l < 2; ++l) {
              glUniform1i(upolyhedron, p);
              glUniform1i(uorientation, o);
225           glUniform1i(uequirectangular, l);
              glBegin(GL_QUADS); {
                float u, w;
                if (l == 0)
                {
230               u = size / 1.6666666666666666;
                  w = u * height / width;
                  w /= 0.825223;
                }
                else
235             {
                  w = 3.141592653589793 * 851 / 651;
                  u = w * width / height;
                  u *= 0.825223;
                }
240             glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
                glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
                glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
                glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
              } glEnd();
245           glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
              char fname[200];
              snprintf(fname, 100, "platonic-%d-%d-%d.pgm", p, o, l);
              FILE *f = fopen(fname, "wb");
              fprintf(f, "P5\n%d %d\n255\n", width, height);
250           fflush(f);
              fwrite(buffer, width * height, 1, f);
              fflush(f);
              fclose(f);
```

```
        }
      }
    }

    free(buffer);
    glDeleteFramebuffers(1, &fbo);
    glDeleteTextures(1, &tex);
    glDeleteShader(frag);
    glDeleteProgram(prog);
    glutReportErrors();
    return 0;
}
```

13 README.md

```
# diary

generate printable diary PDF

## 2019

currently hardcoded to 2019 week-per-view with images on right hand page

UK special dates

## usage

    ./diary.sh

requres OpenGL and various PDF tools

output in 2019-print.pdf

needs 2.1GB disk space

## cover

needs mask.png, shape.png (black on white/alpha, 4096x4096)
suitable images can be converted by hand from cover-page.tex
note that GIMP imports PDF at 4094x4094 for some unknown reason...

cover is stochastic, run it several times with different seeds and pick the
most aesthetic result.  seed is saved in the ppm header for reproducibility

invocation example for 16 cores / 2.5GB RAM / 6.7GB disk space:

    # these values control appearance
    c=1.04
    N=2
    f=0.99
    for seed in $(seq -w 1 16)
    do
      ./cover 14588 10216 $c $N $f 100$seed > 100$seed.ppm &
    done

## legal
```

```
diary -- generate printable diary PDF

Copyright (C) 2018  Claude Heiland-Allen <claude@mathr.co.uk>

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 <https://www.gnu.org/licenses/>.


--
https://mathr.co.uk/diary
```

14 square-conformal-graph-paper.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

#include <stdio.h>
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) "#define REAL float\n" "#define VEC2 vec2\n" "#define VEC4 vec4\n" #s

static const int width = 3648;
static const int height = 5106;
static const float size = 4.0;


static const char *frag_src = GLSL(

  const float pi = 3.141592653589793;
  uniform float factor;
  uniform int formula;

  vec2 crecip(vec2 x) {
    return vec2(x.x, -x.y) / dot(x, x);
  }

VEC2 cExp(VEC2 z) {
        return exp(z.x) * VEC2(cos(z.y), sin(z.y));
}
```

```
REAL cAbs(VEC2 z) {
        return length(z);
}

VEC2 cAdd( VEC2 a, REAL s ) {
  return VEC2( a.x+s, a.y );
}

VEC2 cAdd( REAL s, VEC2 a ) {
  return VEC2( s+a.x, a.y );
}

VEC2 cAdd( VEC2 a, VEC2 s ) {
  return a + s;
}

VEC2 cSub( VEC2 a, REAL s ) {
  return VEC2( a.x-s, a.y );
}

VEC2 cSub( REAL s, VEC2 a ) {
  return VEC2( s-a.x, -a.y );
}

VEC2 cSub( VEC2 a, VEC2 s ) {
  return a - s;
}

REAL cArg(VEC2 a) {
        return atan(a.y,a.x);
}

VEC2 cSqr(VEC2 z) {
        return VEC2(z.x*z.x-z.y*z.y,2.*z.x*z.y);
}

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 cConj(vec2 a)
{
  return vec2(a.x, -a.y);
}

vec2 cDiv(vec2 a, vec2 b)
{
  return cMul(a, cConj(b)) / dot(b, b);
}

vec4 cExp(vec4 a)
{
  return vec4(cExp(a.xy), cMul(a.zw, cExp(a.xy)));
}

vec4 cSqr(vec4 a)
{
```

```
  return vec4(cSqr(a.xy), 2.0 * cMul(a.xy, a.zw));
}

vec4 cMul(vec4 a, vec4 b)
{
  return vec4(cMul(a.xy, b.xy), cMul(a.xy, b.zw) + cMul(a.zw, b.xy));
}

vec2 cSqrt( vec2 z ) {
  float m = length(z);
  return sqrt( max(vec2(0.0), 0.5*vec2(m+z.x, m-z.x)) ) *
    vec2( 1.0, sign(z.y) );
}

vec4 cSqrt( vec4 z ) {
  return vec4(cSqrt(z.xy), 0.5 * cDiv(z.zw, cSqrt(z.xy)));
}

vec2 cLog(vec2 a)
{
  return vec2(log(cAbs(a.xy)), cArg(a.xy));
}

VEC4 cLog(VEC4 a) {
        return VEC4(cLog(a.xy), cDiv(a.zw, a.xy));
}

VEC4 cAsin(VEC4 z) {
  const VEC4 I = VEC4(0.0, 1.0, 0.0, 0.0);
  return cMul(-I, cLog(cMul(I, z) + cSqrt(vec4(1.0, vec3(0.0)) - cSqr(z))));
}

VEC2 cPow(VEC2 z, VEC2 a) {
        return cExp(cMul(cLog(z), a));
}

VEC4 cPow(VEC4 z, REAL a) {
        return cExp(cLog(z) * a);
}

  void main() {
    vec2 p = gl_TexCoord[0].xy;
    p *= mat2(0.0, -1.0, 1.0, 0.0);
    float d = length(vec4(dFdx(p), dFdy(p)));
    vec4 q4 = vec4(p, d, 0.0);
    if (formula == 0) q4 = cAsin(q4) / pi;
    if (formula == 1) q4 = cSqrt(q4);
    if (formula == 2) q4 = cPow(q4, 1.0/4.0);
    vec2 q = q4.xy;
    d = length(q4.zw);
    float l = ceil(-log(d)/log(factor));
    float f = l + log(d)/log(factor);
    l -= factor < 2.5 ? 9.0 : 5.0;
    float o[2];
    for (int i = 0; i < 2; ++i) {
      l += 1.0;
      vec2 u = q * pow(factor, l);
```

```
      u *= (factor < 2.5 ? 8.0 : 6.0) / pow(factor, 2.0);
      u -= floor(u);
      float r = min
        ( min(length(u), length(u - vec2(1.0, 0.0)))
        , min(length(u - vec2(0.0, 1.0)), length(u - vec2(1.0, 1.0)))
        );
      float c = clamp(pow(2.0, factor)/8.0 * r / (pow(factor, l) * d), 0.0, 1.0)↙
        ↳ ;
      vec2 v = q * pow(factor, l - 1.0);
      v *= (factor < 2.5 ? 8.0 : 6.0) / pow(factor, 2.0);
      v -= floor(v);
      float s = min(min(v.x, v.y), min(1.0 - v.x, 1.0 - v.y));
      float k = clamp(pow(2.0, factor)/8.0 * s / (pow(factor, l - 1.0) * d), ↙
        ↳ 0.0, 1.0);
      o[i] = c * k;
    }
    gl_FragColor = vec4(vec3(0.5 + 0.5 * mix(o[1], o[0], f)), 1.0);
  }

);

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 = 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 = 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);
    }
  }

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

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutCreateWindow("graphpaper");
    glewInit();

    GLint success;
    int prog = glCreateProgram();
    int frag = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
    glCompileShader(frag);
    glAttachShader(prog, frag);
    glLinkProgram(prog);
    glGetProgramiv(prog, GL_LINK_STATUS, &success);
    if (!success)
    {
      debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
      debug_program(prog);
      exit(1);
    }
    glUseProgram(prog);
    GLuint ufactor = glGetUniformLocation(prog, "factor");
    GLuint uformula = glGetUniformLocation(prog, "formula");

    GLuint tex;
    glGenTextures(1, &tex);
    glBindTexture(GL_TEXTURE_2D, tex);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, ↙
        ↳ GL_UNSIGNED_BYTE, 0);
    glBindTexture(GL_TEXTURE_2D, 0);

    GLuint fbo;
    glGenFramebuffers(1, &fbo);
    glBindFramebuffer(GL_FRAMEBUFFER, fbo);
    glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, ↙
        ↳ tex, 0);

    glViewport(0, 0, width, height);
    glLoadIdentity();
    gluOrtho2D(0, 1, 1, 0);

    unsigned char *buffer = malloc(width * height);
    for (int k = 2; k <= 3; ++k) {
      for (int i = 0; i <= 2; ++i) {
          glUniform1f(ufactor, k);
```

```
        glUniform1i(uformula, i);
        glBegin(GL_QUADS); {
          float u = size / 2.0;
          float w = u * height / width / 0.825223;
          glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
          glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
          glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
          glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
        } glEnd();
        glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
        char fname[200];
        snprintf(fname, 100, "square-conformal-%d-%d.pgm", k, i);
        FILE *f = fopen(fname, "wb");
        fprintf(f, "P5\n%d %d\n255\n", width, height);
        fflush(f);
        fwrite(buffer, width * height, 1, f);
        fflush(f);
        fclose(f);

      }
    }

    free(buffer);
    glDeleteFramebuffers(1, &fbo);
    glDeleteTextures(1, &tex);
    glDeleteShader(frag);
    glDeleteProgram(prog);
    glutReportErrors();
    return 0;
}
```

15 square-log-polar-graph-paper.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

#include <stdio.h>
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) #s

static const int width = 3648;
static const int height = 5106;
static const float size = 4.0;

static const char *frag_src = GLSL(

  uniform vec4 twist;
  uniform bool loxodrome;

  vec2 clog(vec2 x) {
```

```
25      return vec2(log(length(x)), atan(x.y, x.x));
    }

vec2 cMul(vec2 a, vec2 b) {
        return vec2( a.x*b.x -  a.y*b.y,a.x*b.y + a.y * b.x);
30  }

vec2 cConj(vec2 a)
{
  return vec2(a.x, -a.y);
35  }

vec2 cDiv(vec2 a, vec2 b)
{
  return cMul(a, cConj(b)) / dot(b, b);
40  }

    vec4 clog(vec4 x)
    {
      return vec4(clog(x.xy), cDiv(x.zw, x.xy)) * 0.15915494309189535;
45    }

    vec3 stereo(vec2 x) {
      return vec3(2.0 * x, dot(x, x) - 1.0) / (dot(x, x) + 1.0);
    }
50
    vec2 unstereo(vec3 x) {
      return x.xy / (1.0 - x.z);
    }

55  vec3 rotate(vec3 x) {
      return x.zyx * vec3(1.0, 1.0, -1.0);
    }

    void main() {
60    float factor = twist.w;
      vec2 p = gl_TexCoord[0].xy;
      p *= mat2(0.0, -1.0, 1.0, 0.0);
      if (loxodrome)
      {
65      p = unstereo(rotate(stereo(p)));
      }
      vec4 q4 = clog(vec4(p, length(vec4(dFdx(p), dFdy(p))), 0.0));
      float a = atan(twist.y, twist.x);
      float h = length(vec2(twist.x, twist.y));
70    q4.xy *= mat2(cos(a), sin(a), -sin(a), cos(a)) * h;
      q4.zw *= mat2(cos(a), sin(a), -sin(a), cos(a)) * h;
      float d = length(q4.zw);
      float l = ceil(-log(d)/log(factor));
      float f = l + log(d)/log(factor);
75    l -= factor < 2.5 ? 9.0 : 5.0;
      float o[2];
      for (int i = 0; i < 2; ++i) {
        l += 1.0;
        vec2 u = q4.xy * pow(factor, l);
80      u *= twist.z / pow(factor, 2.0);
        u -= floor(u);
```

```
      float r = min
        ( min(length(u), length(u - vec2(1.0, 0.0)))
        , min(length(u - vec2(0.0, 1.0)), length(u - vec2(1.0, 1.0)))
        );
      float c = clamp(pow(2.0, factor)/8.0 * r / (pow(factor, l) * d), 0.0, 1.0)↙
          ↳ ;
      vec2 v = q4.xy * pow(factor, l - 1.0);
      v *= twist.z / pow(factor, 2.0);
      v -= floor(v);
      float s = min(min(v.x, v.y), min(1.0 - v.x, 1.0 - v.y));
      float k = clamp(pow(2.0, factor)/8.0 * s / (pow(factor, l - 1.0) * d), ↙
          ↳ 0.0, 1.0);
      o[i] = c * k;
    }
    gl_FragColor = vec4(vec3(0.5 + 0.5 * mix(o[1], o[0], f)), 1.0);
  }

);

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 = 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 = 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);
  }
}

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

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutCreateWindow("graphpaper");
  glewInit();

  GLint success;
  int prog = glCreateProgram();
  int frag = glCreateShader(GL_FRAGMENT_SHADER);
  glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
  glCompileShader(frag);
  glAttachShader(prog, frag);
  glLinkProgram(prog);
  glGetProgramiv(prog, GL_LINK_STATUS, &success);
  if (!success)
  {
    debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
    debug_program(prog);
    exit(1);
  }
  glUseProgram(prog);
  GLuint utwist = glGetUniformLocation(prog, "twist");
  GLuint uloxodrome = glGetUniformLocation(prog, "loxodrome");

  GLuint tex;
  glGenTextures(1, &tex);
  glBindTexture(GL_TEXTURE_2D, tex);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, GL_UNSIGNED_BYTE, 0);
  glBindTexture(GL_TEXTURE_2D, 0);

  GLuint fbo;
  glGenFramebuffers(1, &fbo);
  glBindFramebuffer(GL_FRAMEBUFFER, fbo);
  glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, tex, 0);

  glViewport(0, 0, width, height);
  glLoadIdentity();
  gluOrtho2D(0, 1, 1, 0);

  unsigned char *buffer = malloc(width * height);
  float twist[8][4] =
    { { 1, 0, 8, 2 }
    , { 1, 1, 8, 2 }
    , { 2, 1, 8, 2 }
    , { 3, 2, 8, 2 }
```

```
      , { 1, 0, 6, 3 }
      , { 1, 1, 6, 3 }
      , { 2, 1, 6, 3 }
      , { 3, 2, 6, 3 }
195   };
  for (int k = 0; k < 8; ++k) {
    for (int l = 1; l < 2; ++l) {
      glUniform4fv(utwist, 1, &twist[k][0]);
      glUniform1i(uloxodrome, l);
200   glBegin(GL_QUADS); {
        float u = size / 2.0;
        float w = u * height / width / 0.825223;
        glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
        glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
205     glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
        glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
      } glEnd();
      glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
      char fname[200];
210   snprintf(fname, 100, "square-%d-%d.pgm", k, l);
      FILE *f = fopen(fname, "wb");
      fprintf(f, "P5\n%d %d\n255\n", width, height);
      fflush(f);
      fwrite(buffer, width * height, 1, f);
215   fflush(f);
      fclose(f);
    }
  }

220 free(buffer);
  glDeleteFramebuffers(1, &fbo);
  glDeleteTextures(1, &tex);
  glDeleteShader(frag);
  glDeleteProgram(prog);
225 glutReportErrors();
  return 0;
}
```

16 title-page.tex

```
% diary -- generate printable diary PDF
% Copyright (C) 2018  Claude Heiland-Allen
% License GPL3+ <http://www.gnu.org/licenses/>

5 \documentclass[10pt]{extarticle}
\usepackage[paperwidth=6.08in,paperheight=8.51in,margin=20mm,left=40mm]{geometry}
\usepackage{tabularx}
\usepackage{booktabs}
\usepackage{multirow}
10 \usepackage{adjustbox}
\usepackage{lmodern}
\renewcommand{\familydefault}{\sfdefault}

\makeatletter
15 \newcommand\cellwidth{\TX@col@width}
\makeatother
```

```
\begin{document}
\pagestyle{empty}
\topskip0pt
\vspace*{\fill}
\begin{adjustbox}{width=\columnwidth}{\begin{tabularx}{1.1\columnwidth}[t]{X @{\extracolsep{\fill}} X @{\extracolsep{\fill}} X}
\multicolumn{3}{c}{\Huge \bf 2019} \\
\\
\\
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf January} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &1 &2 &3 &4 &5 &6 \\
 7& 8& 9&10&11&12&13\\
14&15&16&17&18&19&20\\
21&22&23&24&25&26&27\\
28&29&30&31&  &  &  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf February} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  &  & 1& 2& 3\\
 4& 5& 6& 7& 8& 9&10\\
11&12&13&14&15&16&17\\
18&19&20&21&22&23&24\\
25&26&27&28&  &  &  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf March} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  &  & 1& 2& 3\\
 4& 5& 6& 7& 8& 9&10\\
11&12&13&14&15&16&17\\
18&19&20&21&22&23&24\\
25&26&27&28&29&30&31\\
  &  &  &  &  &  &  \\
\end{tabularx}} \\
\\
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf April} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
 1& 2& 3& 4& 5& 6& 7\\
 8& 9&10&11&12&13&14\\
15&16&17&18&19&20&21\\
22&23&24&25&26&27&28\\
29&30&  &  &  &  &  \\
```

```
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf May} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  & 1& 2& 3& 4& 5\\
 6& 7& 8& 9&10&11&12\\
13&14&15&16&17&18&19\\
20&21&22&23&24&25&26\\
27&28&29&30&31&  &  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf June} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  &  &  & 1& 2\\
 3& 4& 5& 6& 7& 8& 9\\
10&11&12&13&14&15&16\\
17&18&19&20&21&22&23\\
24&25&26&27&28&29&30\\
  &  &  &  &  &  &  \\
\end{tabularx}} \\
\\
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf July} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
 1& 2& 3& 4& 5& 6& 7\\
 8& 9&10&11&12&13&14\\
15&16&17&18&19&20&21\\
22&23&24&25&26&27&28\\
29&30&31&  &  &  &  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf August} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  & 1& 2& 3& 4\\
 5& 6& 7& 8& 9&10&11\\
12&13&14&15&16&17&18\\
19&20&21&22&23&24&25\\
26&27&28&29&30&31&  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf September} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  &  &  &  & 1\\
```

```
 2& 3& 4& 5& 6& 7& 8\\
 9&10&11&12&13&14&15\\
16&17&18&19&20&21&22\\
23&24&25&26&27&28&29\\
30&  &  &  &  &  &  \\
\end{tabularx}} \\
\\
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}
    ↪ }} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\
    ↪ fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf October} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &1 &2 &3 &4 &5 &6 \\
 7& 8& 9&10&11&12&13\\
14&15&16&17&18&19&20\\
21&22&23&24&25&26&27\\
28&29&30&31&  &  &  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}
    ↪ }} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\
    ↪ fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf November} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  &  & 1& 2& 3\\
 4& 5& 6& 7& 8& 9&10\\
11&12&13&14&15&16&17\\
18&19&20&21&22&23&24\\
25&26&27&28&29&30&  \\
  &  &  &  &  &  &  \\
\end{tabularx}} &
{\begin{tabularx}{\cellwidth}[t]{c @{\extracolsep{\fill}} c @{\extracolsep{\fill}
    ↪ }} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\fill}} c @{\extracolsep{\
    ↪ fill}} c @{\extracolsep{\fill}} c}
\multicolumn{7}{c}{\Large \bf December} \\
\bf M&\bf T&\bf W&\bf T&\bf F&\bf S&\bf S\\
  &  &  &  &  &  & 1\\
 2& 3& 4& 5& 6& 7& 8\\
 9&10&11&12&13&14&15\\
16&17&18&19&20&21&22\\
23&24&25&26&27&28&29\\
30&31&  &  &  &  &  \\
\end{tabularx}}\end{tabularx}}
\end{adjustbox}
\vspace*{\fill}
\end{document}
```

17 triangle-conformal-graph-paper.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

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

```
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) "#define REAL float\n" "#define VEC2 vec2\n" "#define VEC4 vec4\n" #s

static const int width = 3648;
static const int height = 5106;
static const float size = 4.0;


static const char *frag_src = GLSL(

  const float pi = 3.141592653589793;
  uniform float factor;
  uniform int formula;

  vec2 crecip(vec2 x) {
    return vec2(x.x, -x.y) / dot(x, x);
  }

VEC2 cExp(VEC2 z) {
        return exp(z.x) * VEC2(cos(z.y), sin(z.y));
}

REAL cAbs(VEC2 z) {
        return length(z);
}

VEC2 cAdd( VEC2 a, REAL s ) {
  return VEC2( a.x+s, a.y );
}

VEC2 cAdd( REAL s, VEC2 a ) {
  return VEC2( s+a.x, a.y );
}

VEC2 cAdd( VEC2 a, VEC2 s ) {
  return a + s;
}

VEC2 cSub( VEC2 a, REAL s ) {
  return VEC2( a.x-s, a.y );
}

VEC2 cSub( REAL s, VEC2 a ) {
  return VEC2( s-a.x, -a.y );
}

VEC2 cSub( VEC2 a, VEC2 s ) {
  return a - s;
}

REAL cArg(VEC2 a) {
        return atan(a.y,a.x);
```

```
}

VEC2 cSqr(VEC2 z) {
        return VEC2(z.x*z.x-z.y*z.y,2.*z.x*z.y);
}

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 cConj(vec2 a)
{
  return vec2(a.x, -a.y);
}

vec2 cDiv(vec2 a, vec2 b)
{
  return cMul(a, cConj(b)) / dot(b, b);
}

vec4 cExp(vec4 a)
{
  return vec4(cExp(a.xy), cMul(a.zw, cExp(a.xy)));
}

vec4 cSqr(vec4 a)
{
  return vec4(cSqr(a.xy), 2.0 * cMul(a.xy, a.zw));
}

vec4 cMul(vec4 a, vec4 b)
{
  return vec4(cMul(a.xy, b.xy), cMul(a.xy, b.zw) + cMul(a.zw, b.xy));
}

vec2 cSqrt( vec2 z ) {
  float m = length(z);
  return sqrt( max(vec2(0.0), 0.5*vec2(m+z.x, m-z.x)) ) *
    vec2( 1.0, sign(z.y) );
}

vec4 cSqrt( vec4 z ) {
  return vec4(cSqrt(z.xy), 0.5 * cDiv(z.zw, cSqrt(z.xy)));
}

vec2 cLog(vec2 a)
{
  return vec2(log(cAbs(a.xy)), cArg(a.xy));
}

VEC4 cLog(VEC4 a) {
        return VEC4(cLog(a.xy), cDiv(a.zw, a.xy));
}

VEC4 cAsin(VEC4 z) {
  const VEC4 I = VEC4(0.0, 1.0, 0.0, 0.0);
  return cMul(-I, cLog(cMul(I, z) + cSqrt(vec4(1.0, vec3(0.0)) - cSqr(z))));
```

```
}

VEC2 cPow(VEC2 z, VEC2 a) {
        return cExp(cMul(cLog(z), a));
}

VEC4 cPow(VEC4 z, REAL a) {
        return cExp(cLog(z) * a);
}

  void main() {
    vec2 p = gl_TexCoord[0].xy;
    if (formula == 0) p *= mat2(0.0, -1.0, 1.0, 0.0);
    float d = length(vec4(dFdx(p), dFdy(p)));
    vec4 q4 = vec4(p, d, 0.0);
    if (formula == 0) q4 = cAsin(q4) / pi * sqrt(3.0)/2.0;
    if (formula == 1) q4 = cSqrt(q4);
    if (formula == 2) q4 = cPow(q4, 1.0/3.0);
    vec2 q = q4.xy;
    d = length(q4.zw);
    q.x /= sqrt(3.0) / 2.0;
    q.y += q.x / 2.0;
    float l = ceil(-log(d)/log(factor));
    float f = l + log(d)/log(factor);
    l -= factor < 2.5 ? 9.0 : 5.0;
    float o[2];
    for (int i = 0; i < 2; ++i) {
      l += 1.0;
      vec2 u = q * pow(factor, l);
      u *= (factor < 2.5 ? 8.0 : 6.0) / pow(factor, 2.0);
      u -= floor(u);
      float r = min
        ( min(length(u), length(u - vec2(1.0, 0.0)))
        , min(length(u - vec2(0.0, 1.0)), length(u - vec2(1.0, 1.0)))
        );
      float c = clamp(pow(2.0, factor)/8.0 * r / (pow(factor, l) * d), 0.0, 1.0)↙
          ↳ ;
      vec2 v = q * pow(factor, l - 1.0);
      v *= (factor < 2.5 ? 8.0 : 6.0) / pow(factor, 2.0);
      v -= floor(v);
      float s = min(min(min(v.x, v.y), min(1.0 - v.x, 1.0 - v.y)), abs(v.x - v.y↙
          ↳ ));
      float k = clamp(pow(2.0, factor)/8.0 * s / (pow(factor, l - 1.0) * d), ↙
          ↳ 0.0, 1.0);
      o[i] = c * k;
    }
    gl_FragColor = vec4(vec3(0.5 + 0.5 * mix(o[1], o[0], f)), 1.0);
  }

);

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 = 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 = 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);
  }
}

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

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutCreateWindow("graphpaper");
  glewInit();

  GLint success;
  int prog = glCreateProgram();
  int frag = glCreateShader(GL_FRAGMENT_SHADER);
  glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
  glCompileShader(frag);
  glAttachShader(prog, frag);
  glLinkProgram(prog);
  glGetProgramiv(prog, GL_LINK_STATUS, &success);
  if (!success)
```

```
    {
      debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
      debug_program(prog);
      exit(1);
    }
    glUseProgram(prog);
    GLuint ufactor = glGetUniformLocation(prog, "factor");
    GLuint uformula = glGetUniformLocation(prog, "formula");

    GLuint tex;
    glGenTextures(1, &tex);
    glBindTexture(GL_TEXTURE_2D, tex);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, ↙
        ↳ GL_UNSIGNED_BYTE, 0);
    glBindTexture(GL_TEXTURE_2D, 0);

    GLuint fbo;
    glGenFramebuffers(1, &fbo);
    glBindFramebuffer(GL_FRAMEBUFFER, fbo);
    glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, ↙
        ↳ tex, 0);

    glViewport(0, 0, width, height);
    glLoadIdentity();
    gluOrtho2D(0, 1, 1, 0);

    unsigned char *buffer = malloc(width * height);
    for (int k = 2; k <= 3; ++k) {
      for (int l = 0; l <= 2; ++l) {
        glUniform1f(ufactor, k);
        glUniform1i(uformula, l);
        glBegin(GL_QUADS); {
          float u = size / 2.0;
          float w = u * height / width / 0.825223;
          glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
          glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
          glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
          glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
        } glEnd();
        glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
        char fname[200];
        snprintf(fname, 100, "triangle-conformal-%d-%d.pgm", k, l);
        FILE *f = fopen(fname, "wb");
        fprintf(f, "P5\n%d %d\n255\n", width, height);
        fflush(f);
        fwrite(buffer, width * height, 1, f);
        fflush(f);
        fclose(f);
      }
    }

    free(buffer);
    glDeleteFramebuffers(1, &fbo);
    glDeleteTextures(1, &tex);
    glDeleteShader(frag);
    glDeleteProgram(prog);
    glutReportErrors();
```

```
  return 0;
}
```

18 triangle-log-polar-graph-paper.c

```
/*
diary -- generate printable diary PDF
Copyright (C) 2018  Claude Heiland-Allen
License GPL3+ <http://www.gnu.org/licenses/>
*/

#include <math.h>
#include <stdio.h>
#include <string.h>

#include <GL/glew.h>
#include <GL/glut.h>

#define GLSL(s) #s

static const int width = 3648;
static const int height = 5106;
static const float size = 4.0;

static const char *frag_src = GLSL(

  uniform vec4 twist;
  uniform bool loxodrome;

  vec2 clog(vec2 x) {
    return vec2(log(length(x)), atan(x.y, x.x));
  }

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 cConj(vec2 a)
{
  return vec2(a.x, -a.y);
}

vec2 cDiv(vec2 a, vec2 b)
{
  return cMul(a, cConj(b)) / dot(b, b);
}

  vec4 clog(vec4 x)
  {
    return vec4(clog(x.xy), cDiv(x.zw, x.xy)) * 0.15915494309189535;
  }

  vec3 stereo(vec2 x) {
    return vec3(2.0 * x, dot(x, x) - 1.0) / (dot(x, x) + 1.0);
  }

  vec2 unstereo(vec3 x) {
```

```
    return x.xy / (1.0 - x.z);
  }

  vec3 rotate(vec3 x) {
    return x.zyx * vec3(1.0, 1.0, -1.0);
  }

  void main() {
    float factor = twist.w;
    vec2 p = gl_TexCoord[0].xy;
    p *= mat2(0.0, -1.0, 1.0, 0.0);
    if (loxodrome)
    {
      p = unstereo(rotate(stereo(p)));
    }
    vec4 q4 = clog(vec4(p, length(vec4(dFdx(p), dFdy(p))), 0.0));
    float a = atan(twist.y, twist.x);
    float h = length(vec2(twist.x, twist.y));
    q4.xy *= mat2(cos(a), sin(a), -sin(a), cos(a)) * h;
    q4.zw *= mat2(cos(a), sin(a), -sin(a), cos(a)) * h;
    q4.x /= sqrt(3.0) / 2.0;
    q4.y += q4.x / 2.0;
    float d = length(q4.zw);
    float l = ceil(-log(d)/log(factor));
    float f = l + log(d)/log(factor);
    l -= factor < 2.5 ? 9.0 : 5.0;
    float o[2];
    for (int i = 0; i < 2; ++i) {
      l += 1.0;
      vec2 u = q4.xy * pow(factor, l);
      u *= twist.z / pow(factor, 2.0);
      u -= floor(u);
      float r = min
        ( min(length(u), length(u - vec2(1.0, 0.0)))
        , min(length(u - vec2(0.0, 1.0)), length(u - vec2(1.0, 1.0)))
        );
      float c = clamp(pow(2.0, factor)/8.0 * r / (pow(factor, l) * d), 0.0, 1.0)↙
        ↳ ;
      vec2 v = q4.xy * pow(factor, l - 1.0);
      v *= twist.z / pow(factor, 2.0);
      v -= floor(v);
      float s = min(min(min(v.x, v.y), min(1.0 - v.x, 1.0 - v.y)), abs(v.x - v.y↙
        ↳ ));
      float k = clamp(pow(2.0, factor)/8.0 * s / (pow(factor, l - 1.0) * d), ↙
        ↳ 0.0, 1.0);
      o[i] = c * k;
    }
    gl_FragColor = vec4(vec3(0.5 + 0.5 * mix(o[1], o[0], f)), 1.0);
  }

);

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 = 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 = 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);
  }
}

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

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutCreateWindow("graphpaper");
  glewInit();

  GLint success;
  int prog = glCreateProgram();
  int frag = glCreateShader(GL_FRAGMENT_SHADER);
  glShaderSource(frag, 1, (const GLchar **) &frag_src, 0);
  glCompileShader(frag);
  glAttachShader(prog, frag);
  glLinkProgram(prog);
  glGetProgramiv(prog, GL_LINK_STATUS, &success);
```

```
    if (!success)
    {
      debug_shader(frag, GL_FRAGMENT_SHADER, frag_src);
      debug_program(prog);
      exit(1);
    }
    glUseProgram(prog);
    GLuint utwist = glGetUniformLocation(prog, "twist");
    GLuint uloxodrome = glGetUniformLocation(prog, "loxodrome");

    GLuint tex;
    glGenTextures(1, &tex);
    glBindTexture(GL_TEXTURE_2D, tex);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, width, height, 0, GL_RED, ↙
        ↳ GL_UNSIGNED_BYTE, 0);
    glBindTexture(GL_TEXTURE_2D, 0);

    GLuint fbo;
    glGenFramebuffers(1, &fbo);
    glBindFramebuffer(GL_FRAMEBUFFER, fbo);
    glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, ↙
        ↳ tex, 0);

    glViewport(0, 0, width, height);
    glLoadIdentity();
    gluOrtho2D(0, 1, 1, 0);

    unsigned char *buffer = malloc(width * height);
    double a1 = atan2(sqrt(3)/2, 3./2);
    double r1 = sqrt(3);
    double c1 = r1 * cos(a1);
    double s1 = r1 * sin(a1);
    double a2 = atan2(sqrt(3)/2, 5./2);
    double r2 = hypot(sqrt(3)/2, 5./2);
    double c2 = r2 * cos(a2);
    double s2 = r2 * sin(a2);
    double a3 = atan2(sqrt(3), 4);
    double r3 = hypot(sqrt(3), 4);
    double c3 = r3 * cos(a3);
    double s3 = r3 * sin(a3);
    float twist[8][4] =
      { { 1,  0,  8, 2 }
      , { c1, s1, 8, 2 }
      , { c2, s2, 8, 2 }
      , { c3, s3, 8, 2 }
      , { 1,  0,  6, 3 }
      , { c1, s1, 6, 3 }
      , { c2, s2, 6, 3 }
      , { c3, s3, 6, 3 }
      };
    for (int k = 0; k < 8; ++k) {
      for (int l = 1; l < 2; ++l) {
          glUniform4fv(utwist, 1, &twist[k][0]);
          glUniform1i(uloxodrome, l);
          glBegin(GL_QUADS); {
            float u = size / 2.0;
            float w = u * height / width / 0.825223;
```

```
            glTexCoord2f( u / 0.889977, -w); glVertex2f(1, 0);
            glTexCoord2f( u / 0.889977,  w); glVertex2f(1, 1);
220         glTexCoord2f(-u / 0.760470,  w); glVertex2f(0, 1);
            glTexCoord2f(-u / 0.760470, -w); glVertex2f(0, 0);
          } glEnd();
          glReadPixels(0, 0, width, height, GL_RED, GL_UNSIGNED_BYTE, buffer);
          char fname[200];
225       snprintf(fname, 100, "triangle-%d-%d.pgm", k, l);
          FILE *f = fopen(fname, "wb");
          fprintf(f, "P5\n%d %d\n255\n", width, height);
          fflush(f);
          fwrite(buffer, width * height, 1, f);
230       fflush(f);
          fclose(f);
        }
      }

235   free(buffer);
      glDeleteFramebuffers(1, &fbo);
      glDeleteTextures(1, &tex);
      glDeleteShader(frag);
      glDeleteProgram(prog);
240   glutReportErrors();
      return 0;
    }
```