PostGIS Pictures And Patterns

Martin Davis *PostGIS Day 2021*

November 2021

crunchy data

Martin Davis

- **Geospatial Engineer** at crunchy data
- Developer on:
  - **JTS Topology Suite**
  - **GEOS**
  - **PostGIS**
  - `pg_featureserv`

I ♥ Math & Geometry

Displaying PostGIS spatial data

- **Tools:**
  - Web map engines *(GeoServer, MapServer)*
  - External applications *(QGIS, PGAdmin)*
  - Geospatial vector formats *(MVT, GeoJSON)*
    - *Need a Web Map Library*
  - Programming Lang + Graphics API

- **PostGIS native?**
  - **SVG**

…wait, what?

SVG - Scalable Vector Graphics

- W3C Standard
- 2D Vector graphics language
- XML markup
- Advantages:
  - High-quality, scalable rendering
  - CSS styling
  - Widely implemented in web browsers
  - Standalone or HTML-embedded
  - Interactive & Dynamic (via Javascript)
  - Editable (e.g. Inkscape)

“SVG does for graphics what HTML does for text”

MDN

SVG Markup

View area

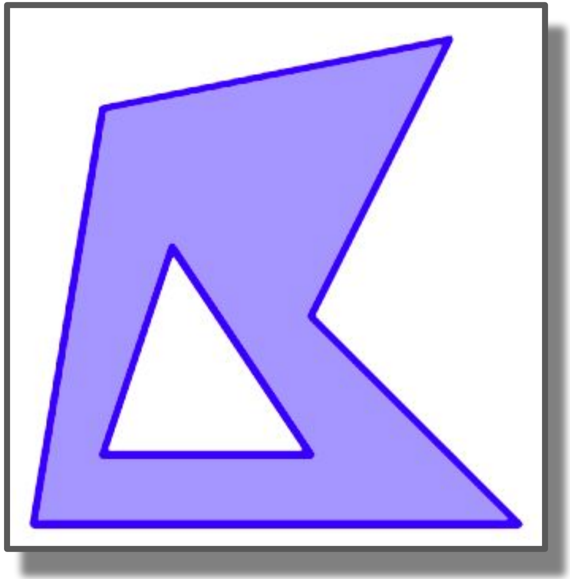
Graphic Element

Polygon fill

CSS styling

Drawing commands

```
<svg xmlns="http://www.w3.org/2000/svg"
     viewBox="-15 -35 50 50"
     >
  <path
     fill-rule="evenodd"
     style="fill:#a0a0ff;
            stroke:#0000ff;
            stroke-width:1;
            stroke-linejoin:round;"
     d="M 20 -20
     L 30 -80 80 -90 60 -50 90 -20 Z
     M 40 -60 L 30 -30 60 -30 Z"
     />
</svg>
```



SVG in PostGIS

- **ST_AsSVG(geometry)**
    - ⇒ drawing commands ONLY

```
SELECT ST_AsSVG('POLYGON ((20 20, 30 80, 80 90, 60 50, 90 20, 20 20))');


                 st_assvg
------------------------------------------
 M 20 -20 L 30 -80 80 -90 60 -50 90 -20 Z
```

Bad Idea

- Generate SVG markup via SQL string concatenation
    - Need detailed knowledge of SVG
    - Tedious, error-prone
    - Hard to read and maintain

```
SELECT '<path fill-rule="evenodd" '
|| ' style="fill: rgb(' || r || ',' || g ',' || b '); '
|| ' stroke:#000000; stroke-width:"' || width || ';"'
|| ' d="' || ST_AsSVG(geom) || '" />'
AS svg FROM data
```

PG-svg

- PL/pgSQL function library
- **Domain-specific language** (DSL) for SVG
  - named parameters, variadic arguments
- Produces text for SVG elements and attributes
- Functions for:
  - Shapes - `svgShape`, `svgPolygon`, `svgRect`, ...
  - Styling - `svgStyle`
  - Utilities - `svgRGB`, `svgHSL`, ...
  - SVG Document - `svgDoc`

https://github.com/dr-jts/pg_svg

delaunay-svg.sql

```sql
WITH data AS (
 SELECT 'MULTIPOINT ((50 50), (50 120), (100 100), (130 70), (130
150), (70 160), (160 110), (70 80))'::geometry geom ),

shapes AS (
```

SVG - Triangles

```sql
 SELECT svgShape( ST_DelaunayTriangles( geom ),
                  style => svgStyle('fill', '#a0a0ff',
                     'stroke', '#0000ff', 'stroke-width', 1::text )
)
   AS svg FROM result
 UNION ALL
```

SVG - Points

```sql
 SELECT svgShape( geom, radius => 2,
                 style => svgStyle( 'fill', '#ff0000' ))
   AS svg FROM data
)
```

SVG Document

```sql
SELECT svgDoc(  ARRAY_AGG( svg ),
         viewbox => svgViewbox(
                  ST_Expand((SELECT ST_Extent(geom) FROM data), 20))
) FROM shapes;
```

https://github.com/dr-jts/pg_svg/blob/master/demo/geom/delaunay-svg.sql

crunchydata

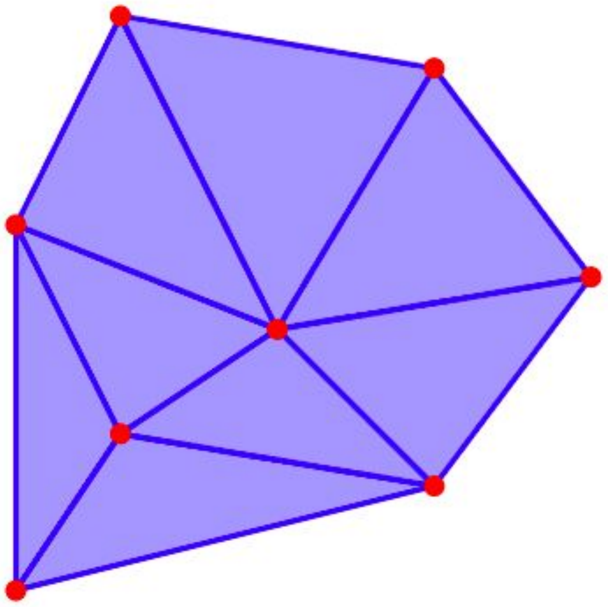
Run the code

```
psql -A -t -o delaunay.svg  < delaunay-svg.sql
```

- `-A` - unaligned table output mode
- `-t` - print rows only
- `-o` - output to file

Delaunay Triangulation

```
<svg viewBox="30 -180 150 150" xmlns="http://www.w3.org/2000/svg">
<g  style="fill:#a0a0ff; stroke:#0000ff; stroke-width:1;
stroke-linejoin:round; ">
<path  fill-rule="evenodd"   d="M 50 -120 L 50 -50 70 -80 Z"   />
<path  fill-rule="evenodd"   d="M 50 -120 L 70 -80 100 -100 Z"   />
<path  fill-rule="evenodd"   d="M 50 -120 L 100 -100 70 -160 Z"   />
<path  fill-rule="evenodd"   d="M 70 -160 L 100 -100 130 -150 Z"   />
<path  fill-rule="evenodd"   d="M 130 -150 L 100 -100 160 -110 Z"   />
<path  fill-rule="evenodd"   d="M 160 -110 L 100 -100 130 -70 Z"   />
<path  fill-rule="evenodd"   d="M 50 -50 L 130 -70 70 -80 Z"   />
<path  fill-rule="evenodd"   d="M 70 -80 L 130 -70 100 -100 Z"   />
<title>Delaunay Triangulation</title></g>
<g  style="fill:#ff0000; ">
<circle  r="2"  cx="50" cy="-50" />
<circle  r="2"  cx="50" cy="-120" />
<circle  r="2"  cx="100" cy="-100" />
<circle  r="2"  cx="130" cy="-70" />
<circle  r="2"  cx="130" cy="-150" />
<circle  r="2"  cx="70" cy="-160" />
<circle  r="2"  cx="160" cy="-110" />
<circle  r="2"  cx="70" cy="-80" />
<title>Site</title></g>
</svg>
```

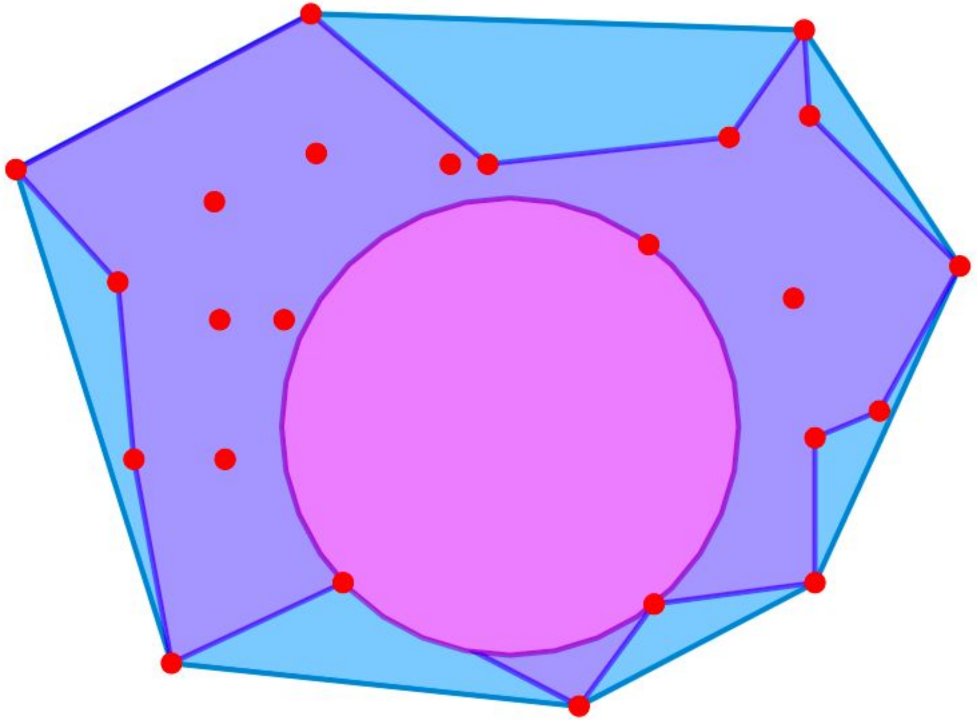


hulls-svg.sql

```
WITH data AS ( SELECT 'MULTIPOINT ((178 80), (174 133), (66 151), (162 163), (205 139), (147 143), (29 157),
(85 160), (79 129), (117 158), (110 158), (84 186), (134 57), (177 167), (178 107), (190 112), (58 65), (67 129), (68 103),
(176 183), (51 103), (90 80), (48 136), (148 76))'::geometry geom ),
shapes AS (
 SELECT svgShape( ST_ConvexHull( geom ),   title => 'Convex Hull',
   style => svgStyle('stroke', '#0088cc', 'stroke-width', '1', 'fill', '#88ccff' ) )
   svg FROM data

 UNION ALL SELECT svgShape( ST_ConcaveHull( geom, 0.99 ),
     title => 'Concave Hull',
     style => svgStyle('stroke', '#0000ff', 'stroke-width', '1', 'fill', '#a0a0ff' ) )
   svg FROM data

 UNION ALL SELECT svgShape( ST_Buffer((mic).center, (mic).radius),
     title => 'Maximum Inscribed Circle',
     style => svgStyle('stroke', '#6600aa', 'stroke-width', '1', 'fill', '#dd90ff' ) )
   svg FROM (SELECT ST_MaximumInscribedCircle( geom ) AS mic FROM data) AS t

 UNION ALL SELECT svgShape( geom, radius => 2,
                           style => svgStyle( 'fill', '#ff0000'  ) )
   svg FROM data
)
SELECT svgDoc( array_agg( svg ),
   viewbox => svgViewbox( ST_Expand( (SELECT ST_Extent(geom) from data), 20 ))
 ) FROM shapes;
```

https://github.com/dr-jts/pg_svg/blob/master/demo/geom/hulls-svg.sql

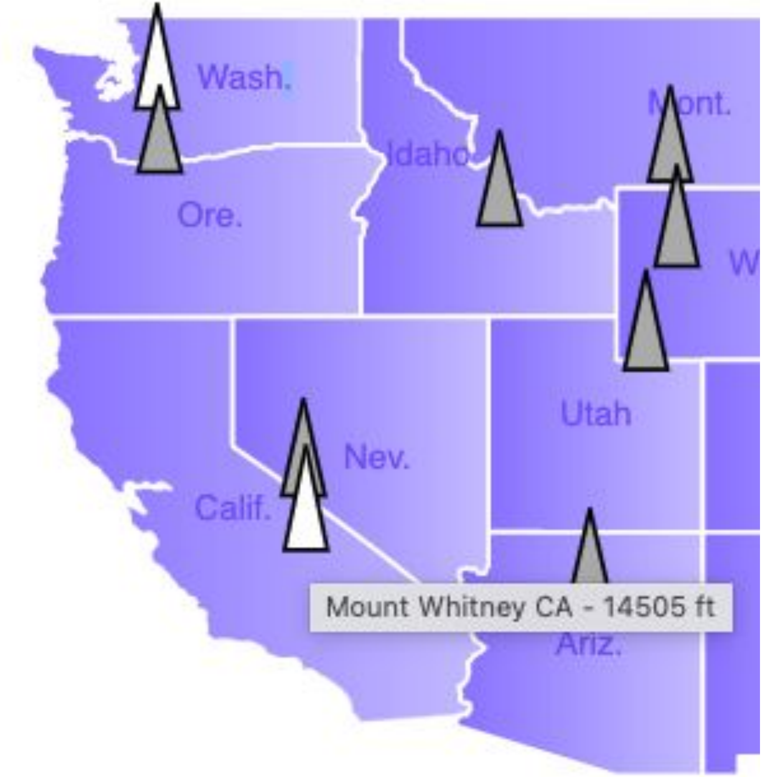
Convex/Concave Hulls, Maximum Inscribed Circle



Maps

US High Points Map

- State High point data
  - Triangle polygon, with height proportional to elevation
  - Fill color based on elevation
  - Name as `title` property
- Natural Earth boundary polygons
  - Fill with linear gradient
- State name
  - SVG text element



https://github.com/dr-jts/pg_svg/blob/master/demo/map/us-highpt-svg.sql

US highpt-svg.sql - Linear Gradient

Use as fill

```
,shapes AS (
 SELECT geom, svgShape( geom,
   title => name,
   style => svgStyle(  'stroke', '#ffffff',
                       'stroke-width', 0.1::text,
                       'fill', 'url(#state)',
                       'stroke-linejoin', 'round' ) )
   svg FROM lower48

                   .  .  .  .
```

Definition

```
SELECT svgDoc( array_agg( svg ),
   viewbox => svgViewbox( ST_Expand( ST_Extent(geom), 2)),
   def => svgLinearGradient('state', '#8080ff', '#c0c0ff')
 ) AS svg FROM shapes;
```

us-highpt-svg.sql - Elevation symbols

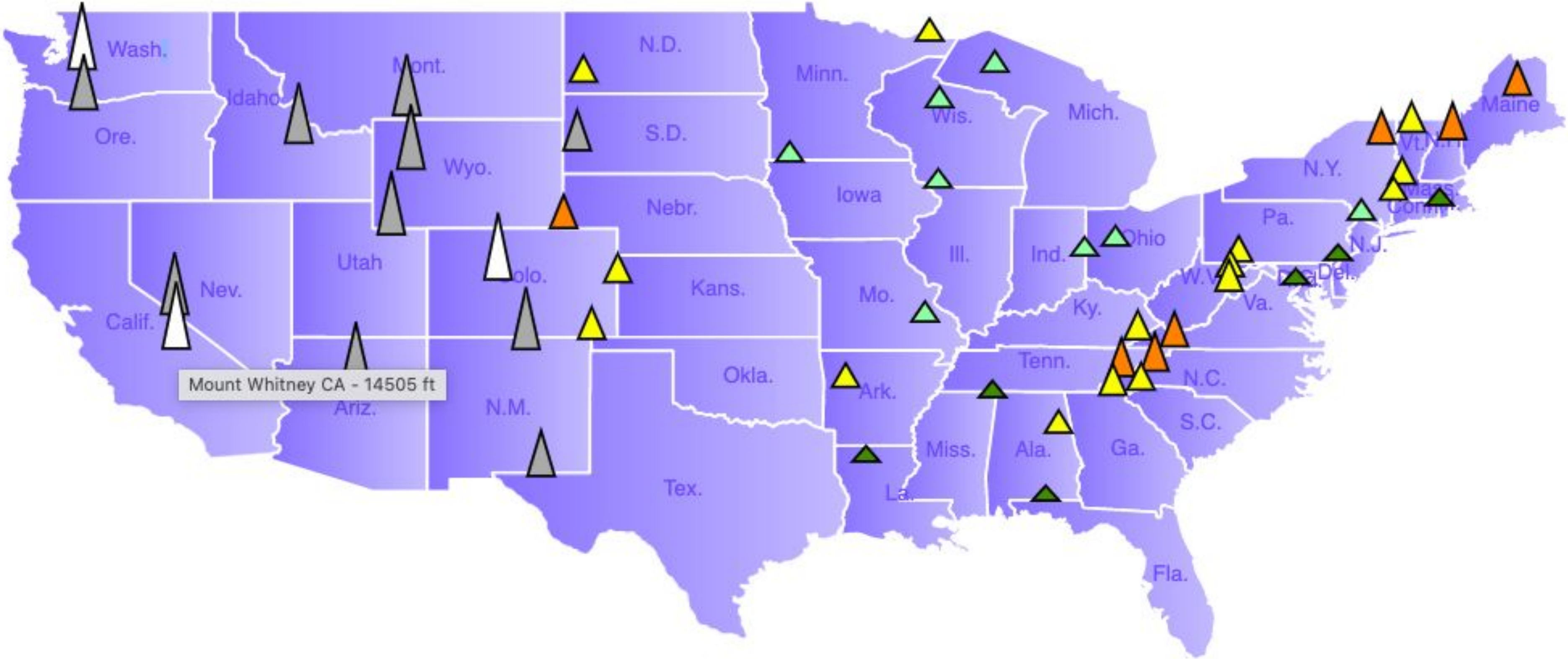
Symbol height & fill from elevation

Rendering order

Create polygon

```
,highpt_geom AS (SELECT name, state, hgt_ft, lon, lat,
   (2.0 * hgt_ft) / 15000.0 + 0.5 AS symHeight,
   CASE WHEN hgt_ft > 14000 THEN '#ffffff'
        WHEN hgt_ft >  7000 THEN '#aaaaaa'
        WHEN hgt_ft >  5000 THEN '#ff8800'
        WHEN hgt_ft >  2000 THEN '#ffff44'
        WHEN hgt_ft >  1000 THEN '#aaffaa'
                            ELSE '#558800'
   END AS clr
   FROM high_pt
     ORDER BY lat DESC)
. . . . .
 SELECT NULL,
     svgPolygon( ARRAY[ lon-0.5, -lat, lon+0.5, -lat, lon, -lat-symHeight ],
          title => name || ' ' || state || ' - ' || hgt_ft || ' ft',
          style => svgStyle(  'stroke', '#000000',
                       'stroke-width', 0.1::text,
                       'fill', clr  ) )
   svg FROM highpt_geom
```

US High Points Map

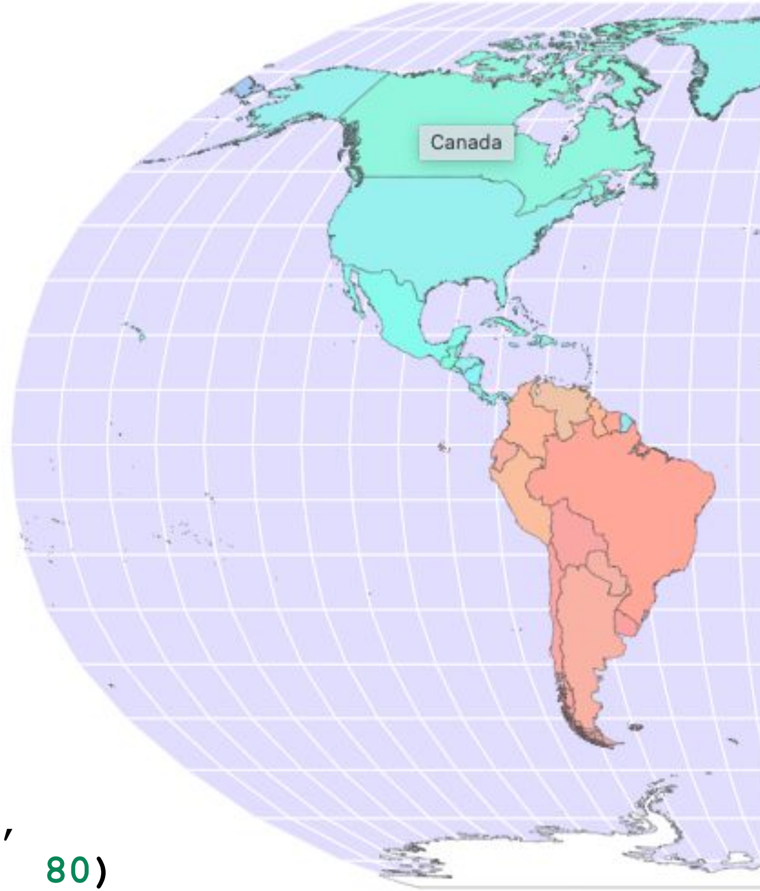


World Map

- Natural Earth admin boundaries
  - Reduce data size:

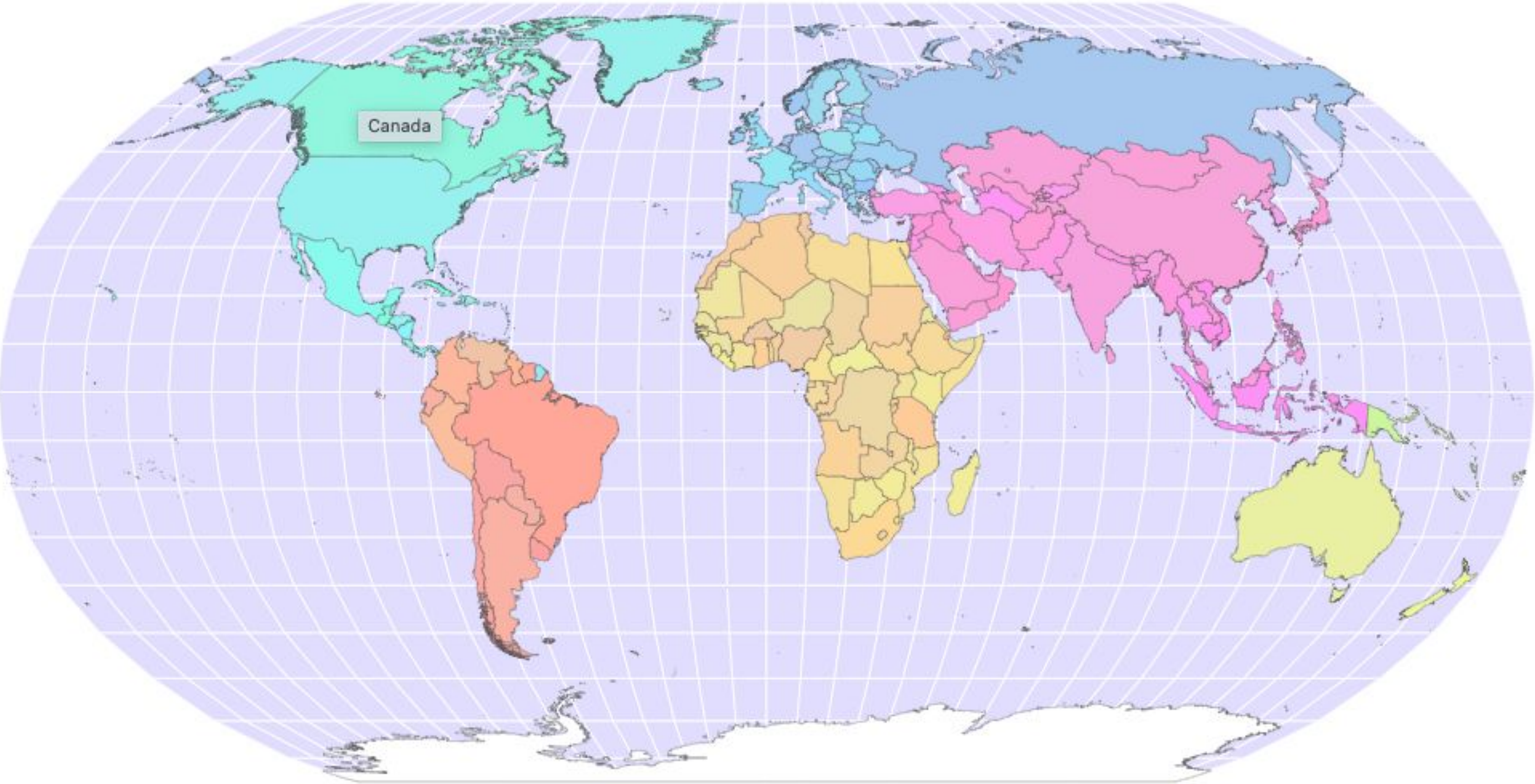
    `ST_SnapToGrid(geom, 0.1)`
- Generate geodetic grid
- Transform to Robinson projection
  - `ST_Transform(geom, 54030)`
- Country name as `title` property
- Style countries with HSL fill from continent and H and S “dither”

```
svgHSL( svgRandInt( hue, hue + 25),
        svgRandInt( 60, 70),        80)
```



https://github.com/dr-jts/pg_svg/blob/master/demo/map/world-svg.sql

World Map



Fractals

Mandelbrot Set

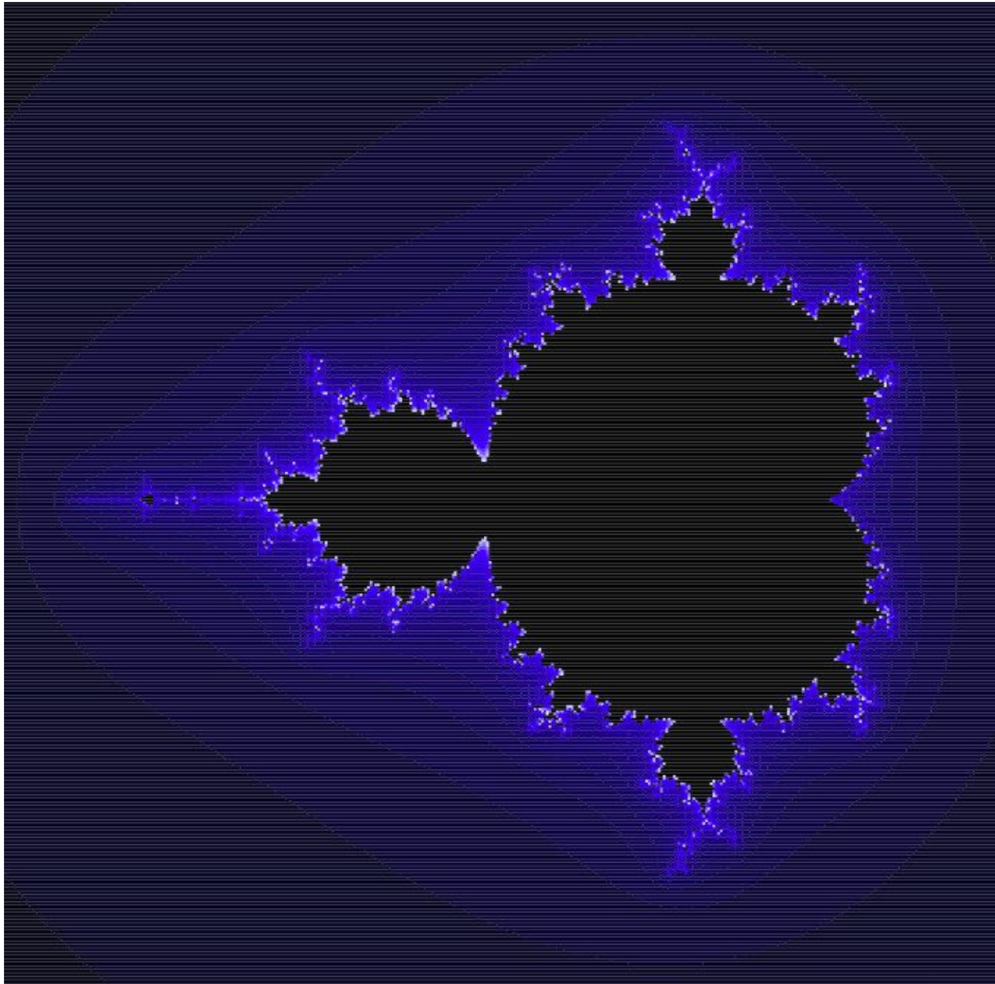
- Fractal filling XY plane ⇒ raster
- Definition:
  - ○ Set of complex numbers for which *f(z) = z^2 + c* does not diverge
- Classic example of SQL recursive CTE wizardry
  - ○ But… ASCII art!

```
                                                     *
                                                   ****
                                                  ******
                                                   *****
                                              * *********
                                         *** ****************
                                           ******************** **
                                        *************************
                                       ****************************
                                     *******************************
                                     ******************************
                        * *****     ********************************
                       ***********  ********************************
                      ************* *******************************
                  ** ************** ******************************
*****************************************************************
                  ** ************** ******************************
                      ************* *******************************
                       ***********  ********************************
                        * *****     ********************************
                                     ******************************
                                     *******************************
                                       ****************************
                                        *************************
                                           ******************** **
                                         *** ****************
                                              * *********
                                                   *****
                                                  ******
                                                   ****
                                                     *
```

Mandelbrot - SVG

- SVG - much better!
  - Draw pixels as `<rect>`

- Problem:
  - 400x400 grid
  - = 160,000 cells!
- Solution:
  - Run-Length Encoding (RLE)
  - ~12,000 rectangles



Mandelbrot Set SQL

1) Compute divergence at grid points

```sql
WITH RECURSIVE
-- Grid index
x(i) AS ( SELECT i FROM generate_series(0, 400) AS t(i) ),

z(ix, iy, cx, cy, x, y, iter) AS (
   -- Complex number values at grid points
   SELECT ix, iy, x::FLOAT, y::FLOAT, x::FLOAT, y::FLOAT, 0
     FROM        (SELECT -2.2 + 0.0074 * i, i FROM x) AS xgen(x, ix)
     CROSS JOIN  (SELECT -1.5 + 0.0074 * i, i FROM x) AS ygen(y, iy)
   UNION ALL
   -- Iterate Mandelbrot eqn at points until divergence or max
   SELECT ix, iy, cx, cy,
       x*x - y*y + cx AS x,
       y*x*2 + cy AS y,       iter + 1
   FROM z
   WHERE x*x + y*y < 16.0  -- divergence
   AND iter < 27           -- max iterations
   ),
-- Get final iteration count for each point
itermax (ix, iy, iter) AS (
   SELECT ix, iy, MAX(iter) AS iter
   FROM z GROUP BY iy, ix
   ),
```

https://github.com/dr-jts/pg_svg/blob/master/demo/fractal/mandelbrot-rle-svg.sql

Mandelbrot Set SQL 2) *Run-Length Encoding*

```
-- Run-Length Encoding across grid rows

-- mark start of run where iter value changes
runstart AS (
   SELECT iy, ix, iter,
   CASE WHEN iter = LAG(iter) OVER (PARTITION BY iy ORDER By ix)
       THEN 0 ELSE 1 END AS runstart
   FROM itermax
   ),
-- assign id number to runs in each row
runid AS (
   SELECT iy, ix, iter,
       SUM(runstart) OVER (PARTITION BY iy ORDER By ix) AS run
   FROM runstart
   ),
-- get run start and end X index
runs AS (
   SELECT iy, MIN(ix) ix, MAX(ix) ixend, MIN(iter) iter
   FROM runid
   GROUP BY iy, run
   ),
```

https://github.com/dr-jts/pg_svg/blob/master/demo/fractal/mandelbrot-rle-svg.sql

crunchydata

Mandelbrot Set SQL 3) *Create SVG*

```sql
-- Map grid cell iteration count to RGB color
plot(iy, ix, ixend, iter, b, g) AS (
   SELECT iy, ix, ixend, iter,
   CASE    WHEN iter < 18 THEN (255 * iter / 18.0 )::integer
           WHEN iter < 27 THEN 255
           ELSE 0 END AS b,
   CASE    WHEN iter < 18 THEN 0
           WHEN iter < 27 THEN (255 * (iter - 18) / (27 - 18
))::integer
           ELSE 0 END AS g
   FROM runs ORDER BY iy, ix
),
-- Create SVG rectangle for each run
svg AS ( SELECT svgRect( ix, iy, ixend-ix+1, 1,
                         style => svgStyle('fill', svgRGB(g, g, b) )
     ) AS svg
   FROM plot
)
SELECT svgDoc( array_agg( svg ),
         viewbox => '0 0 400 400',
         style => svgStyle('stroke-width', '0') )
 FROM svg;
```

https://github.com/dr-jts/pg_svg/blob/master/demo/fractal/mandelbrot-rle-svg.sql

Hilbert Curve

- Space-filling curve
- Generate with a ***Lindenmayer System*** (L-system)
    - Recursive rewrite rules
    - Produce drawing commands (Turtle Graphics)

Axiom: A

Rules: $A \rightarrow -BF+AFA+FB-$

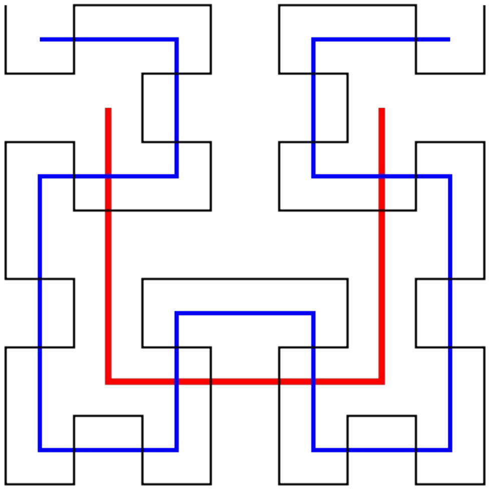
$B \rightarrow +AF-BFB-FA+$

F = forward, + = turn left 90°, − = turn right 90°

1: -F+F+F-

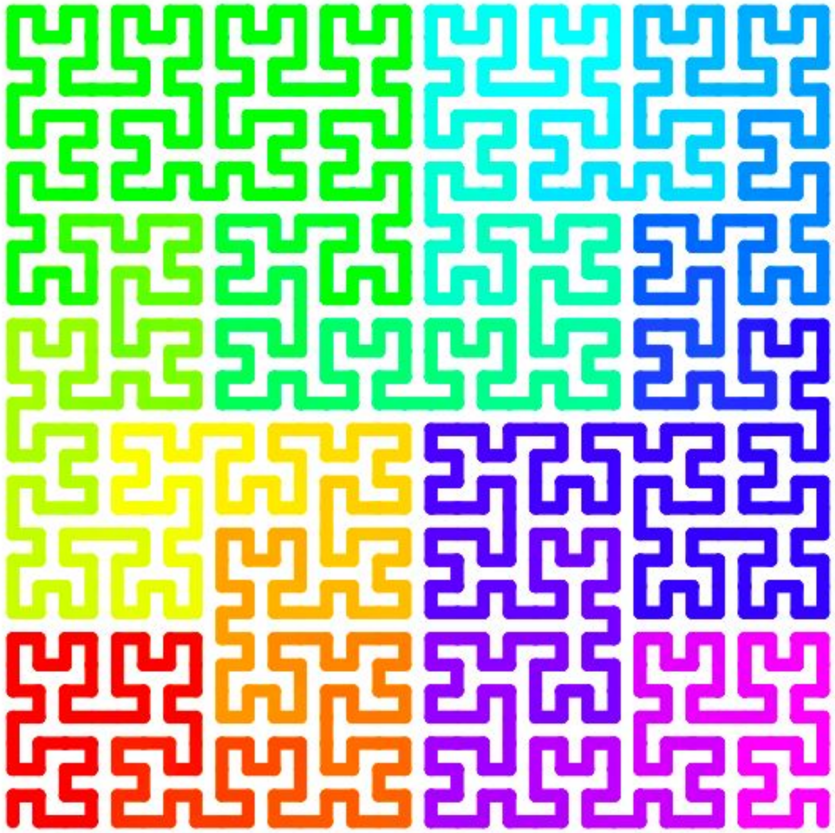
2: F-F-F+F+-F+F+F-F-F+F+FB+F+F-F-F

3: -F+F+F-FF-F-F+F+F-F-FF-F+F+FF+F-F-F+FF+F+F
-F-F+F+FF+F-F-FFF-F-F+FF+F+F-F-F+F+FF+F
-F-F+FF+F+F-FF-F-F+F+F-F-FF-F+F+F-



Hilbert Curve SVG

- Recursive CTE generates L-system to produce string of draw commands
- Recursive CTE “interprets” draw commands to produce line segments
- Style segments with HSL color with increasing hue value
  - `svgHSL(h,s,l)`



https://github.com/dr-jts/pg_svg/blob/master/demo/fractal/hilbert-curve-svg.sql

Hilbert Curve SQL

1) L-system evaluation

```sql
WITH RECURSIVE
-- recursively generate L-system output string
lsystem AS (
   SELECT 'A' AS state, 0 AS iteration
   UNION ALL
   SELECT replace(replace(replace(state, 'A', '-CF+AFA+FC-'),
                    'B', '+AF-BFB-FA+'), 'C', 'B'),
         iteration + 1 AS iteration
   FROM lsystem WHERE iteration < 5  -- Iteration parameter
  ),

-- clean output and optimize drawing commands
path(moves) AS ( SELECT replace(replace(replace(replace(state, 'A', ''),
                            'B', ''),
                            '+-', ''),
                            '-+', '')
  FROM (SELECT state FROM lsystem ORDER BY iteration DESC LIMIT 1) st
  ),
```

Hilbert Curve SQL

2) Interpret commands to create lines

```
-- iterate over draw commands to create segments
pts(moves, index, dir, xp, yp, x, y, dx, dy, len) AS (
   SELECT moves, 1, ' ',  0, 0, 0, 0, 1, 0, 0 FROM path
   UNION ALL
   SELECT moves, index+1 AS index, substr(moves, index, 1) AS dir,
     x AS xp, y AS yp,
     x + dx*len AS x, y + dy*len AS y,
     CASE substr(moves, index, 1)
              WHEN '-' THEN -dy WHEN '+' THEN  dy ELSE dx END AS dx,
     CASE substr(moves, index, 1)
              WHEN '-' THEN  dx WHEN '+' THEN -dx ELSE dy END AS dy,
     CASE substr(moves, index, 1) WHEN 'F' THEN 1 ELSE 0 END AS len
   FROM pts WHERE index <= length(moves) ),
```

```
-- create line segments, in numbered sequence
seg AS (
 SELECT row_number() OVER() AS id,
         ST_MakeLine( ST_Point(xp, yp), ST_Point(x, y)) geom
 FROM pts WHERE xp <> x OR yp <> y
),
```

Hilbert Curve SQL 3) Create SVG

```
-- SVG shapes for line segments, with spectrum fill
svg AS ( SELECT geom, svgShape( geom,
       style => svgStyle('stroke', svgHSL( 300*(id/ 1024.0 ), 100, 50))
     ) AS svg
   FROM seg
)
-- SVG document
SELECT svgDoc( array_agg( svg ),
         viewbox => svgViewbox( ST_Expand( ST_Extent(geom), 5 )),
         style => svgStyle('stroke-width', '0.5', 'stroke-linecap', 'round' ) )
 FROM svg;
```

https://github.com/dr-jts/pg_svg/blob/master/demo/fractal/hilbert-curve-svg.sql

Dragon Curve

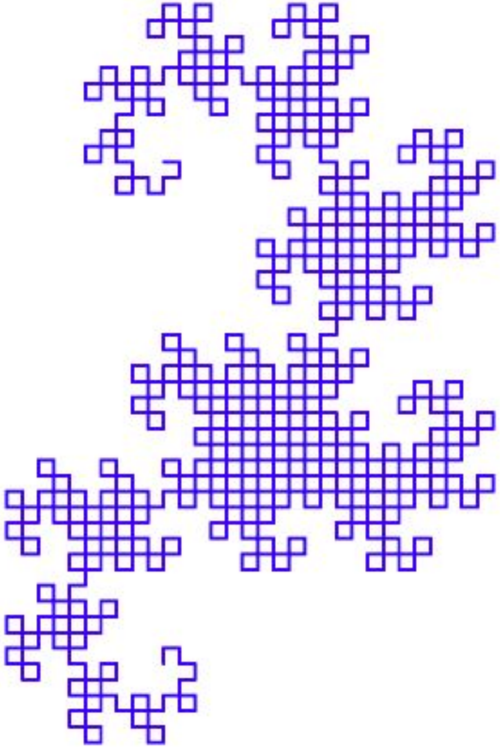
- Generate with simple L-system

Axiom: A

Rules: $A \rightarrow A + B\ \mathrm{F}$

$B \rightarrow \mathrm{F}\ A - B$

F = forward, **+** = turn left 90°, **–** = turn right 90°

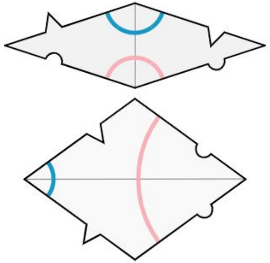


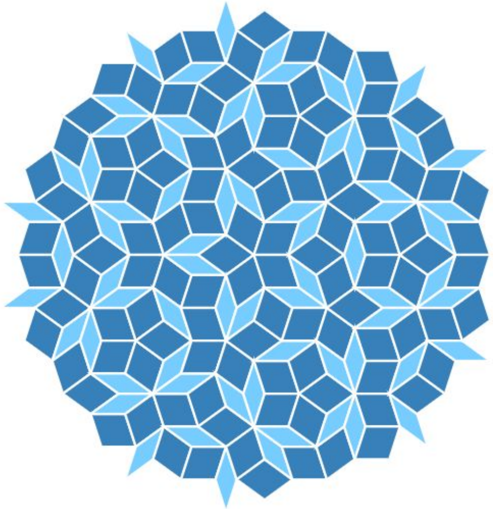
https://github.com/dr-jts/pg_svg/blob/master/demo/fractal/dragon-curve-svg.sql

Tilings

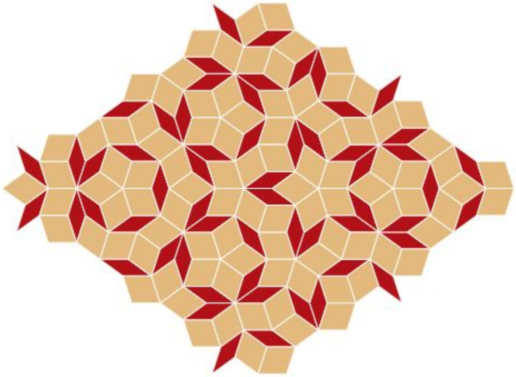
Penrose Tiling

- Aperiodic tiling of the plane
  - No translational symmetry
  - Tiles are two rhombs
  - Adjacency constraints ensure aperiodicity





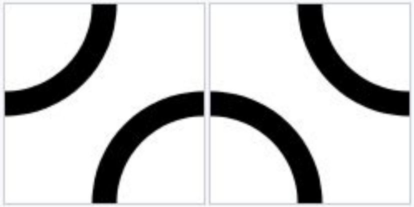
- Generate using ***deflation***
  - Start with simple tile arrangement
  - Recursively decompose tiles according to deflation rules



https://github.com/dr-jts/pg_svg/blob/master/demo/tiling/penrose3S-svg.sql
https://github.com/dr-jts/pg_svg/blob/master/demo/tiling/penrose3L-svg.sql

Truchet Tiling - Curves

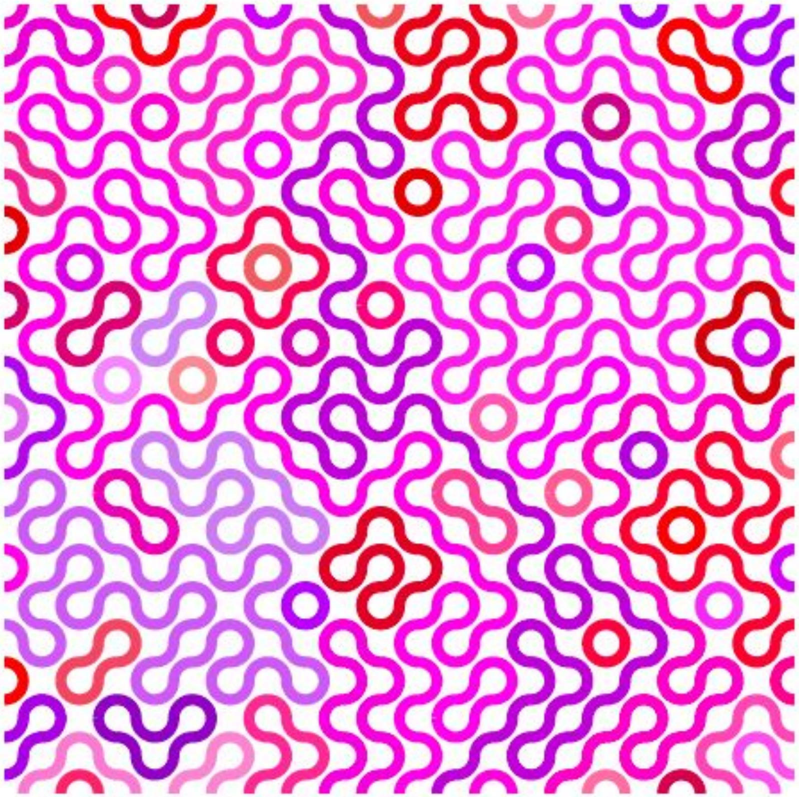
- Two tile types, placed randomly



The Truchet tile

Inverse of the Truchet tile, created by any 90° rotation or orthogonal flip

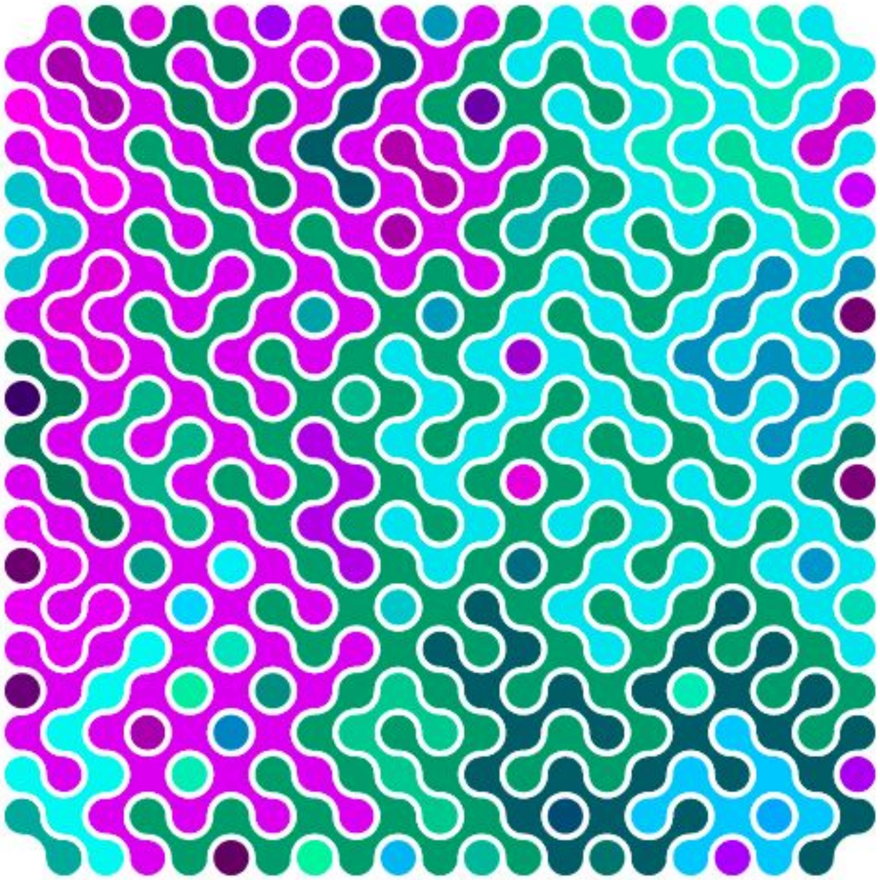
- Generate using PostGIS `CIRCULARSTRING` geometry
  - Convert to lines
  - Merge lines with `ST_LineMerge`
  - Stroke color is “dithered” HSL values



https://github.com/dr-jts/pg_svg/blob/master/demo/tiling/truchet-curve-svg.sql

Truchet Tiling - Polygons

- Generate using PostGIS **`CIRCULARSTRING`** geometry
  - Convert to lines
  - Add closing arcs at edges
  - Form polygons from lines withh **`ST_Polygonize`**
  - Fill color is dithered HSL values



https://github.com/dr-jts/pg_svg/blob/master/demo/tiling/truchet-poly-svg.sql

Future Work

- More SVG elements
  - **<use>** macros
  - **<filter>** definition
- Charting symbols
  - Pie charts, bar charts, etc.
- Legends, map surrounds
- Embed images from PG-Raster
- Non-spatial charting
- SVG output from **`pg-featureserv`**