+

=

https://github.com/dr-jts/pg-util

PostGIS - current SVG

- **`ST_AsSVG( geom )`** - return geometry vertices as SVG `<path>` `data` attribute value

```
SELECT ST_AsSVG('POLYGON((0 0,0 10,10 10,10 0,0 0))');
```

```
M 0 0 L 0 -1 1 -1 1 0 Z
```

Usable SVG

- **`ST_AsSVG( geom )`** - return geometry vertices as SVG `<path>` `data` attribute value

```
SELECT svgDoc( ARRAY[
        svgShape( geom,
            style => svgStyle(
                'stroke', 'black', 'stroke-width', '1',     'fill', 'red'
))],
        svgViewbox( geom ) )
FROM (VALUES( 'POLYGON((0 0,0 10,10 10,10 0,0 0))'::geometry )) AS t(geom);
```

```
<svg viewBox="0 -1 1 1" xmlns="http://www.w3.org/2000/svg">
   <path style="stroke:black; stroke-width:1; fill:red; "
      fill-rule="evenodd"  d="M 0 0 L 0 -1 1 -1 1 0 Z" />
</svg>
```

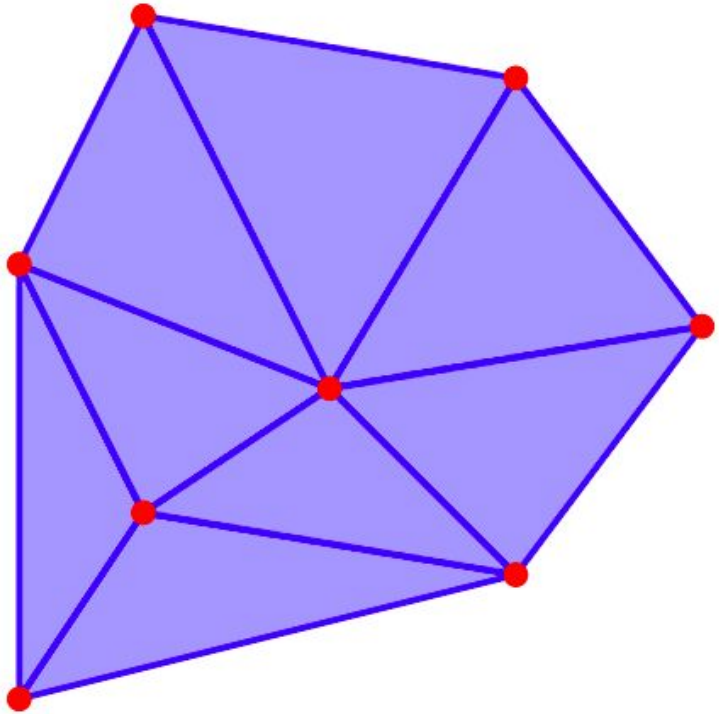
SVG add-on for Postgres/PostGIS

- **`svgDoc`** - `<svg>` element
- **`svgViewbox`** - viewbox attribute
- **`svgShape`** - geometry -> shape element
- **`svgPolygon`** - point array -> `<polygon>`
- **`svgStyle`** - list of CSS name/value pairs
- **`svgHSL`** - CSS hsl(H,S,L) function
- *more to come...*

https://github.com/dr-jts/pg-util

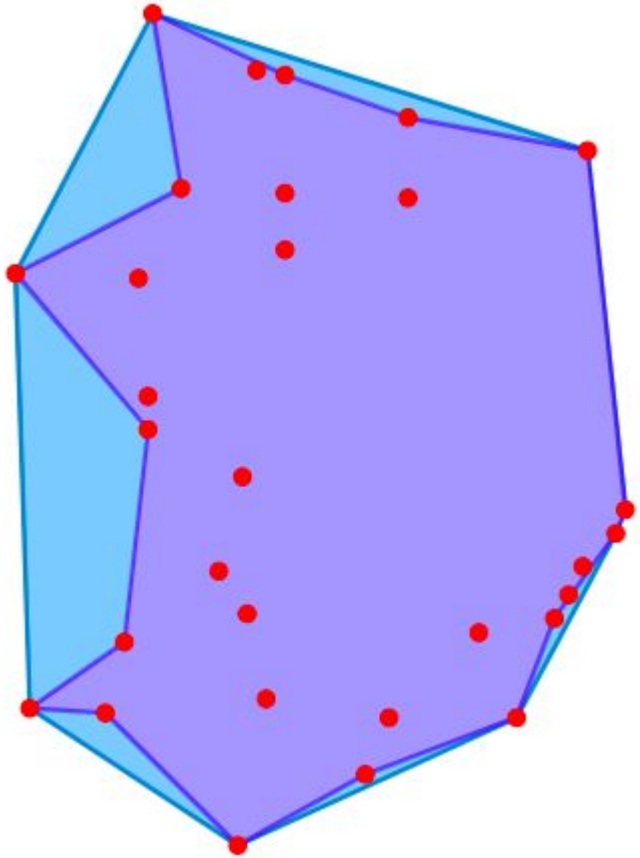
Delaunay Triangulation

```
WITH input AS (
 SELECT 'MULTIPOINT ((50 50), (50 120), (100 100), (130
70), (130 150), (70 160), (160 110), (70 80))'::geometry
geom
), result AS (
 SELECT ST_DelaunayTriangles( geom ) AS geom FROM input
), shapes AS (
 SELECT geom, svgShape( geom,
   title => 'Delaunay Triangulation',
   style => svgStyle('stroke', '#0000ff',
       'stroke-width', 1::text,
       'fill', '#a0a0ff',
       'stroke-linejoin', 'round' ) )
   svg FROM result
 UNION ALL
 SELECT geom, svgShape( geom, radius=>2,
   title => 'Site',
   style => svgStyle( 'fill', '#ff0000'  ) )
   svg FROM input
)
SELECT svgDoc( array_agg( svg ),
   viewbox => svgViewbox( ST_Expand( ST_Extent(geom), 5) )
 ) AS svg FROM shapes;
```



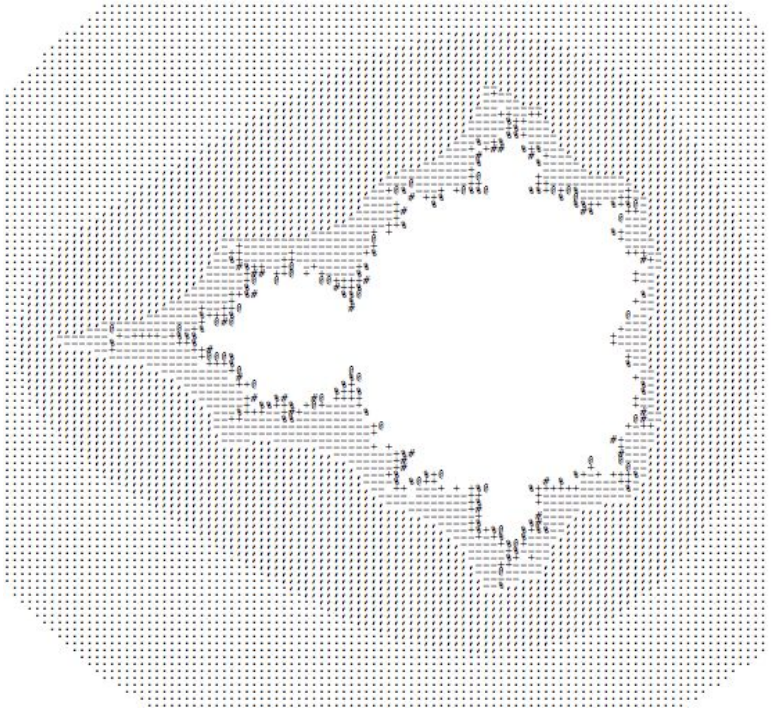
Convex and Concave Hull

```
WITH input AS (
  SELECT 'MULTIPOINT ((158 60), (105 64), (109 196), (87 172)
),
convex AS (
  SELECT ST_ConvexHull( geom ) AS geom FROM input
),
concave AS (
  SELECT ST_ConcaveHull( geom, 0.99 ) AS geom FROM input
),
shapes AS (
  SELECT geom, svgShape( geom,
    title => 'Convex Hull',
    style => svgStyle('stroke', '#0088cc',
        'stroke-width', 1::text,
        'fill', '#88ccff',
        'stroke-linejoin', 'round' ) )
    svg FROM convex
  UNION ALL
  SELECT geom, svgShape( geom,
    title => 'Concave Hull',
    style => svgStyle('stroke', '#0000ff',
        'stroke-width', 1::text,
        'stroke-opacity', 0.5::text,
        'fill', '#a0a0ff',
        'stroke-linejoin', 'round' ) )
    svg FROM concave
  UNION ALL
  SELECT geom, svgShape( geom, radius=>2,
    style => svgStyle( 'fill', '#ff0000' ) )
    svg FROM input
)
SELECT svgDoc( array_agg( svg ),
    viewbox => svgViewbox( ST_Expand( ST_Extent(geom), 5 ))
  ) AS svg FROM shapes;
```



Mandelbrot Set (ASCII-art)

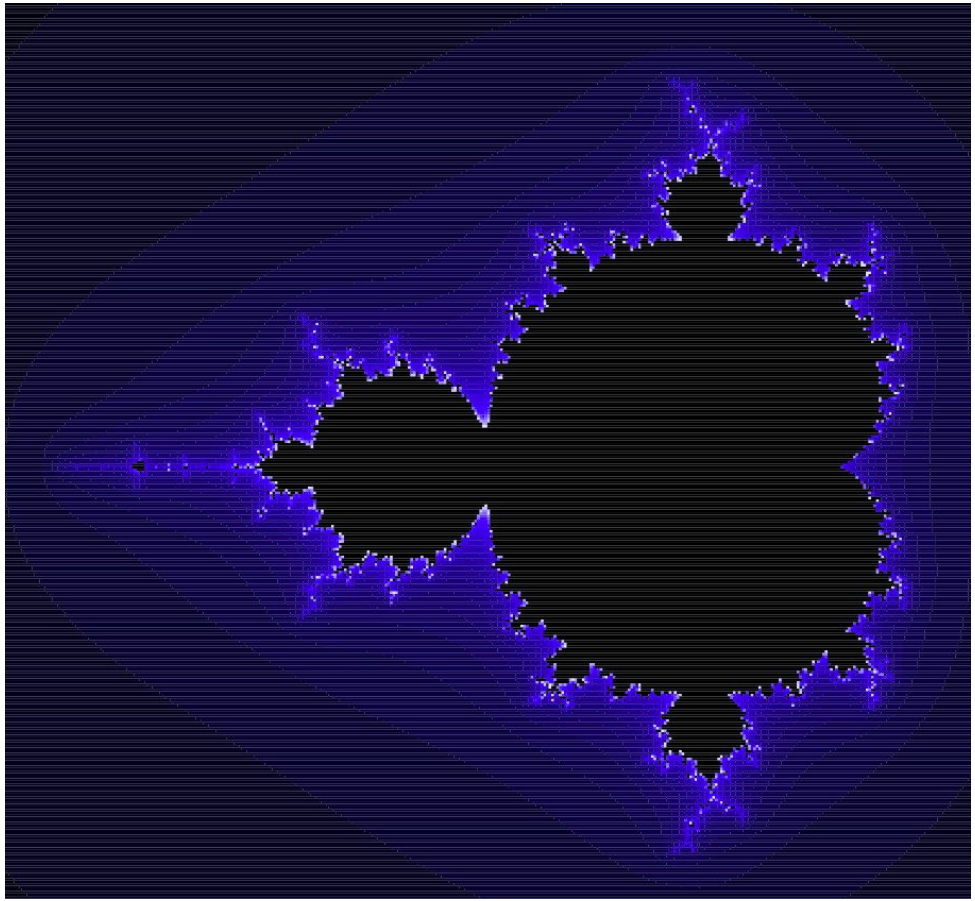
- Generated by standard recursive formula for Mandelbrot in complex plane
- SQL: recursive CTE (Common Table Expression)
- Iterate on each cell until it “escapes” to ∞
- Theme on # iterations



1969 called and it wants its printer back...

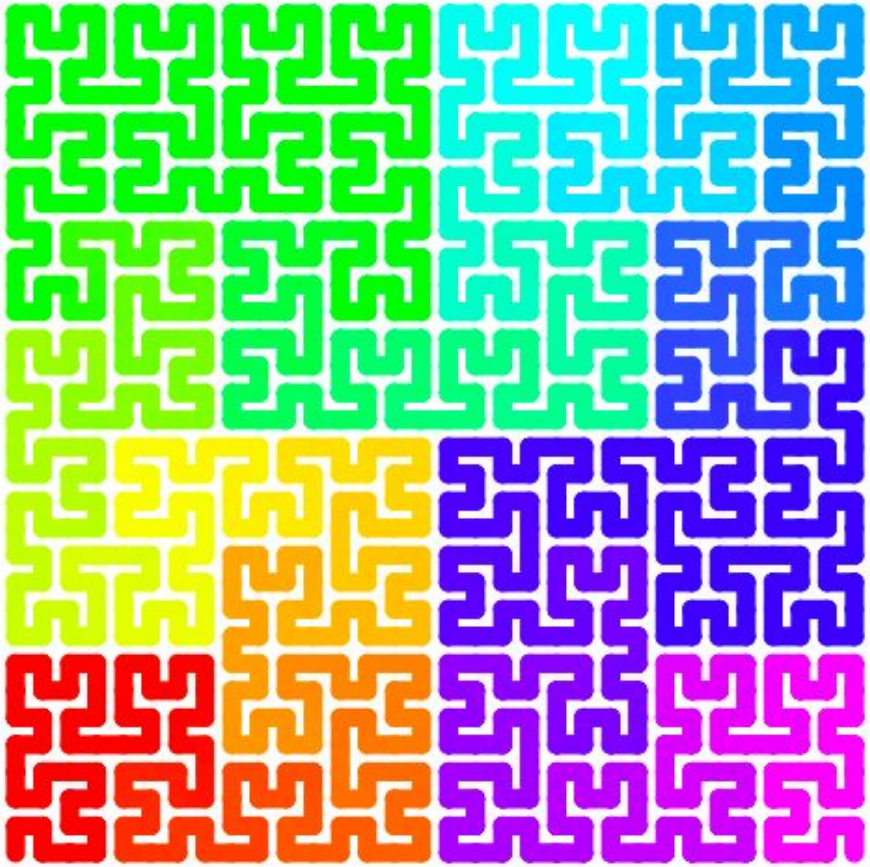
Mandelbrot Set (SVG with RLE)

- Reduce output size by combining same-value cells along rows (Run-Length Encoding)
- Output runs as SVG `rect` elements
- fill using palette on iteration #



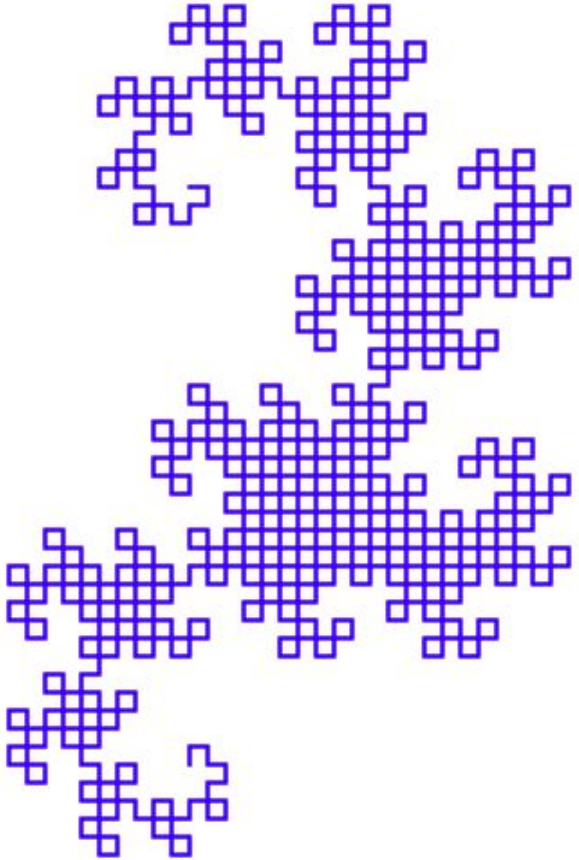
Hilbert Curve

- Generated by L-system
- SQL: recursive CTE (Common Table Expression)
- Each segment has `stroke` chosen from range of hues using CSS `hsl()` function



Dragon Curve

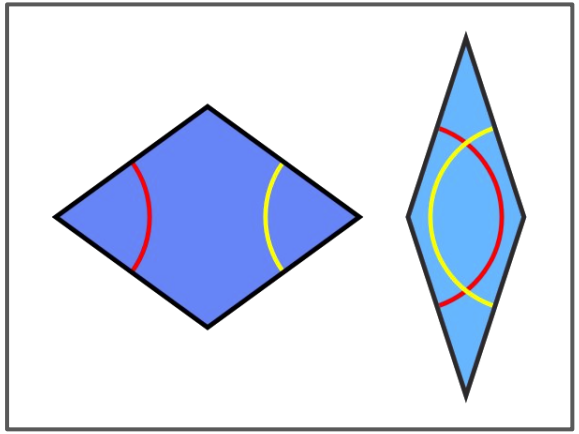
- Generated by L-system
- SQL: recursive CTE (Common Table Expression)
- Image captured from WKT output from PostGIS in JTS TestBuilder



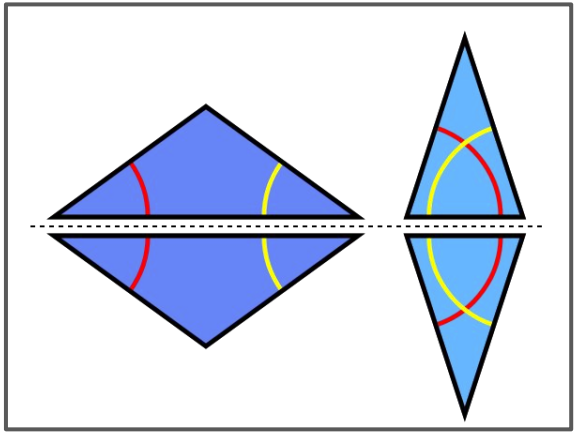
Penrose Tiling

- Aperiodic tiling of the plane (no translational symmetry)
- Type P3: two **rhombi** L and S
- Split rhombi to create **Robinson triangles**
- Generate tilings by ***inflation*** (subdivision) of triangles
- Result tiling depends on initial set of triangles

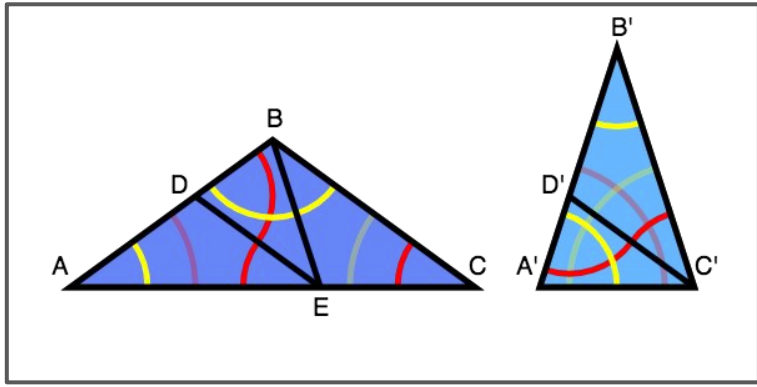
L and S



Robinson triangles

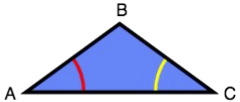


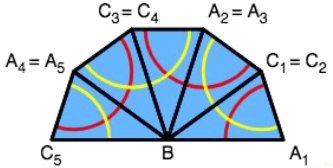
Inflation (subdivision)

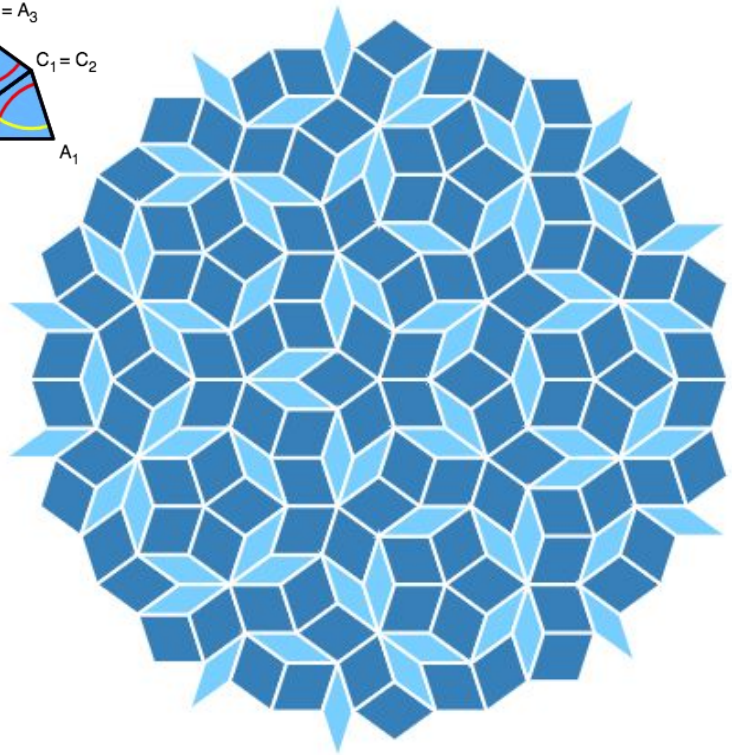


Penrose Tiling - SQL/SVG

- SQL: iterative CTE doing inflation of on initial triangles
- SVG: `polygon` elements



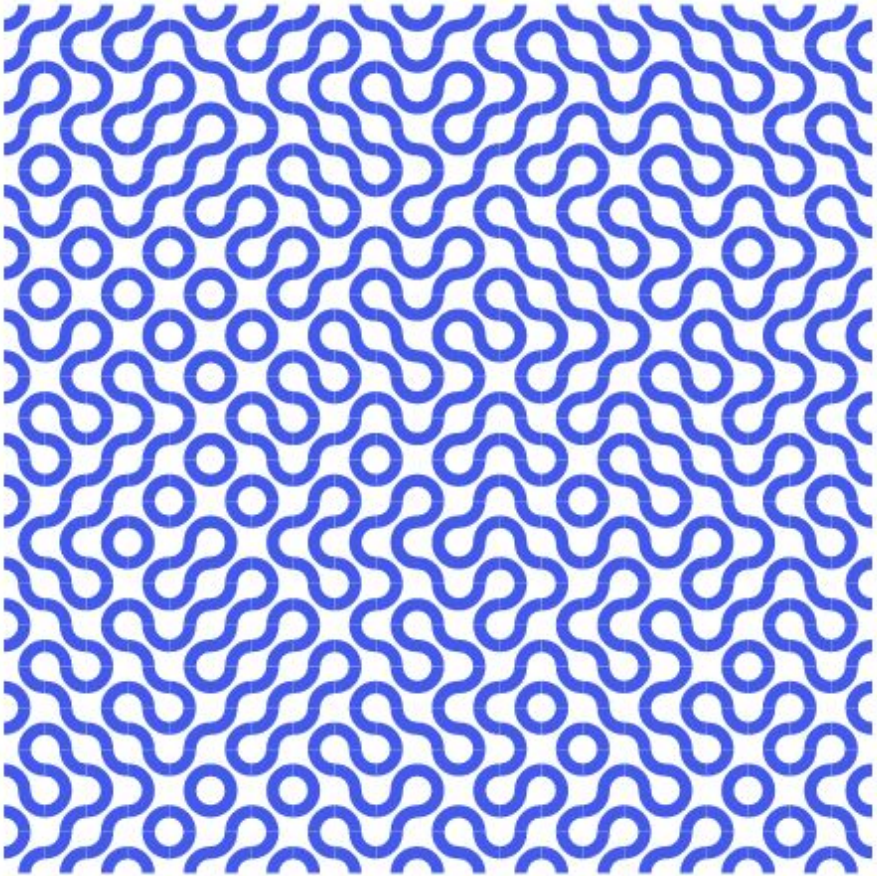




Truchet tiles

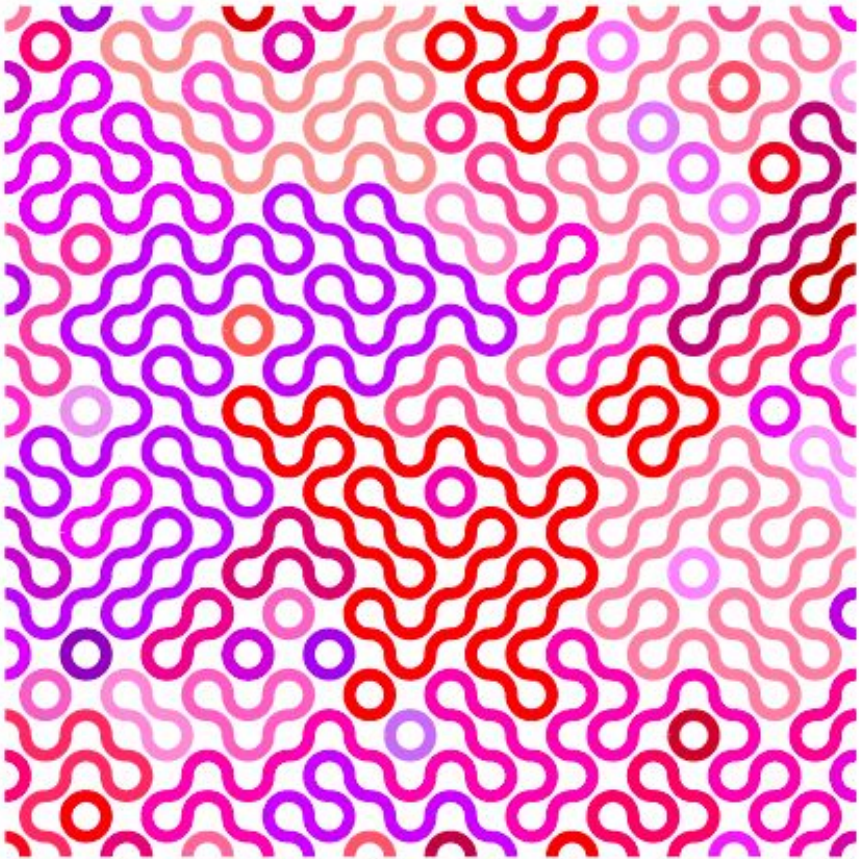
- Square tiling
- Tiles of two patterns, non-rotationally symmetric
- Random placement

Truchet tiles



Truchet tiles - lines

- Create quarter-circles using WKT `CIRCULARSTRING(sx,sy,mx,my,ex,ey)`
- Use `ST_LineMerge` to sew connected lines together



Truchet tiles - polygons

- Add half-circle links along border
- Use `ST_Polygonize` to create polygonal coverage