QMUL Game AI Research Group Seminar

General Board Geometry

Cameron Browne
Digital Ludeme Project
Maastricht University (DKE)

Queen Mary University of London
19/8/2020

Maastricht University

Overview

Context: Ludii

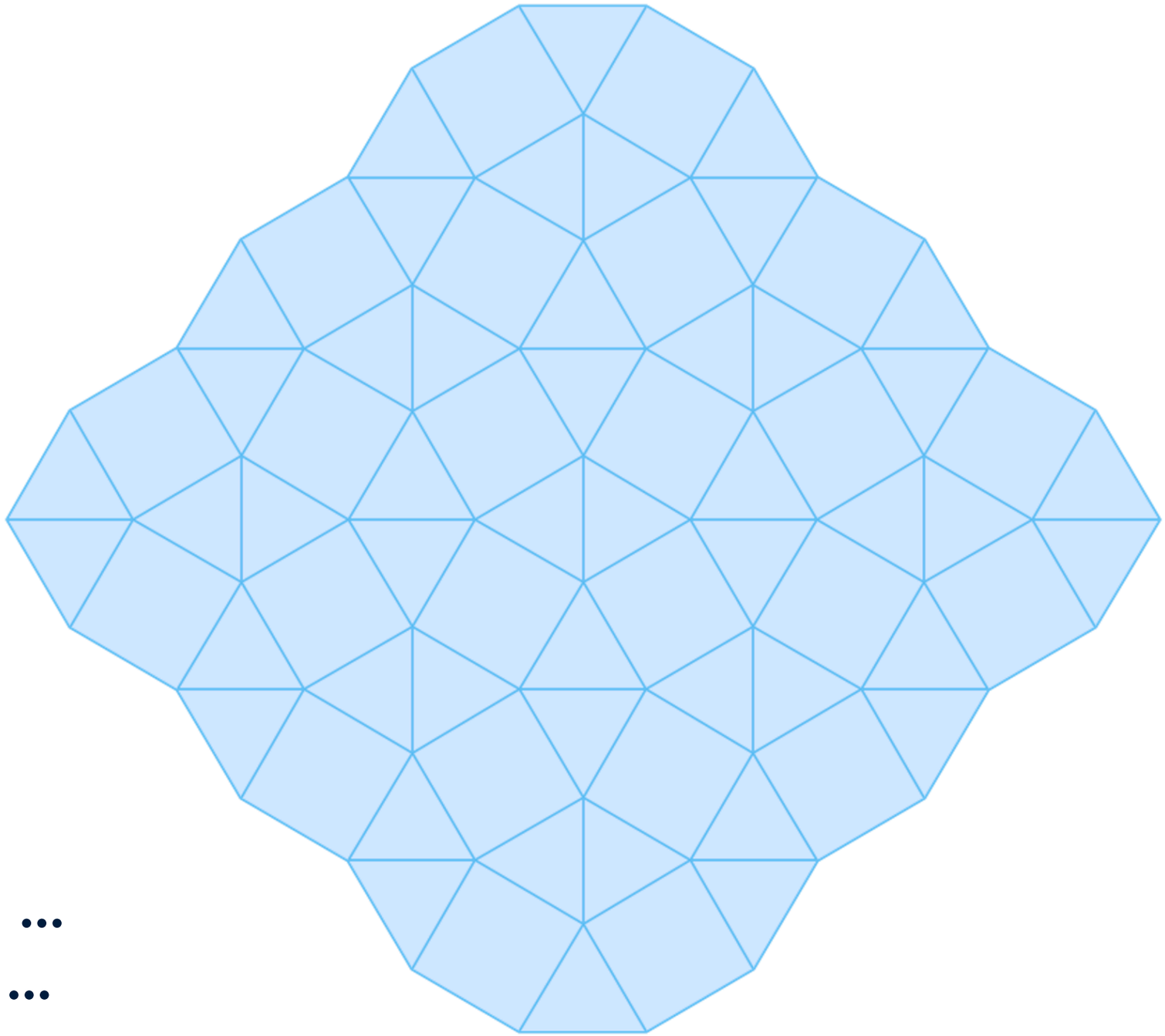
Game boards as graphs
- Board geometry

Design process

Graph functions for
  game boards

Properties:
- Steps, walks, radials, …
- Coordinates, phases, …

- Lots of examples!



Digital Ludeme Project

Five-year research project
- Funded by the ERC (€2m)
- Maastricht University

Aim is to digitally
- Model
- Reconstruct
- Map

world's traditional games

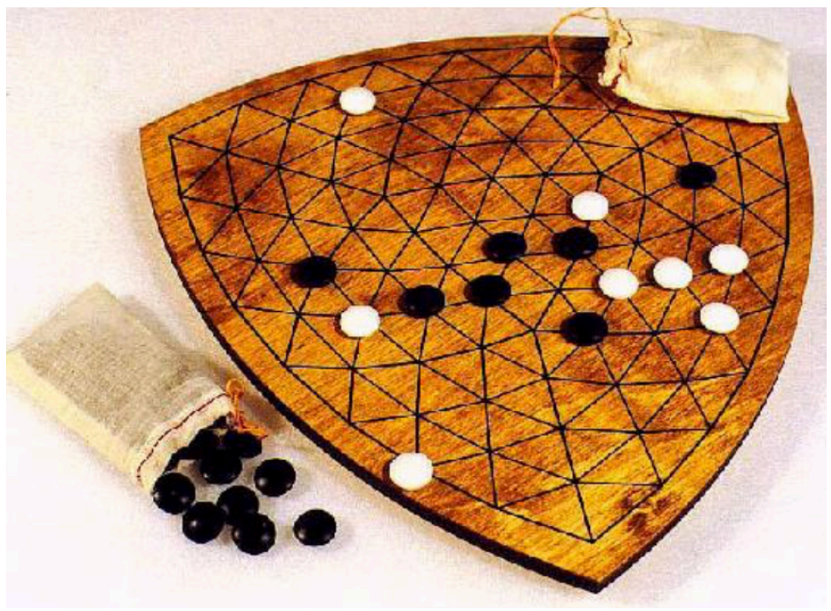
http://ludeme.eu

Scope

1,000 most important games
- Last 4,500 years
- Also modern games

Single consistent format

How to support full range of games and board types?



Ludemes

Ludeme = game “meme”:

- Unit of game-related information
- Building block (DNA) of games
- Encapsulates key game concept

```
(game “?”
   (players 2)
   (equipment {
      (board (square 3))
      (piece “Disc”) })
   (rules
      (play (move Add (to (sites Empty))))
      (end (if (is Line 3)
           (result Mover Win)))
   )
)
```

Ludemes

Ludeme = game “meme”:

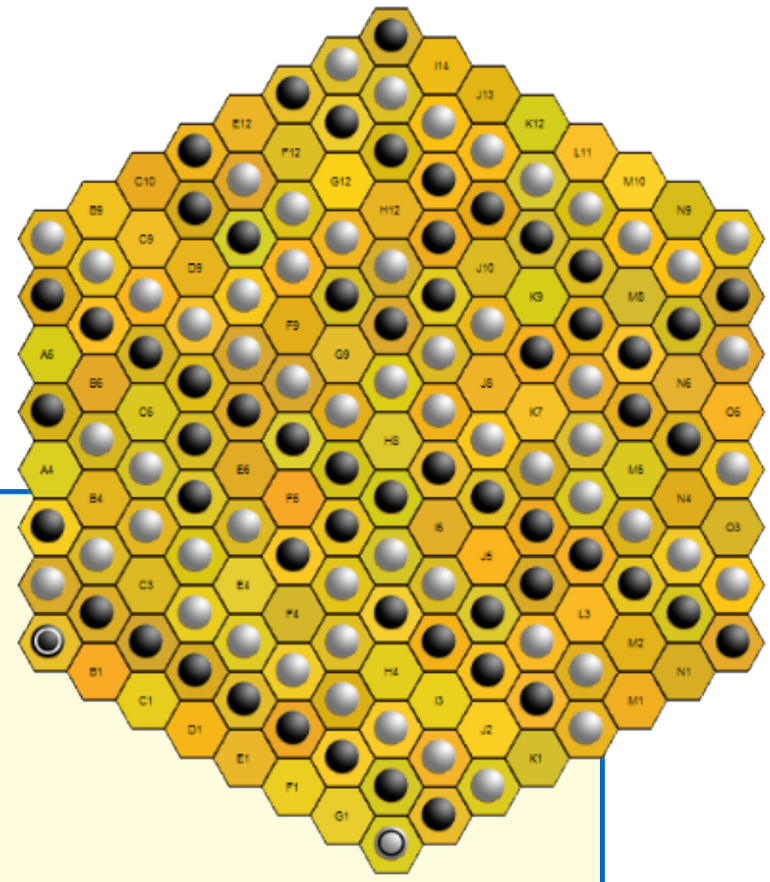
- Unit of game-related information
- Building block (DNA) of games
- Encapsulates key game concept

```
(game “Tic-Tac-Toe”
   (players 2)
   (equipment {
      (board (square 3))
      (piece “Disc”) })
   (rules
      (play (move Add (to (sites Empty))))
      (end (if (is Line 3)
           (result Mover Win)))
   )
)
```

Ludemes

Simplify complex game descriptions

- e.g. Havannah



```
(game "Havannah"
    (players 2)
    (equipment {
        (board (hex 8))
        (piece "Ball") })
    (rules
        (play (move Add (to (sites Empty))))
        (end (if (or {
                (is Loop)
                (is Connected 2 Corners)
                (is Connected 3 SidesNoCorners)
            })
            (result Mover Win))
        )
    )
)
```

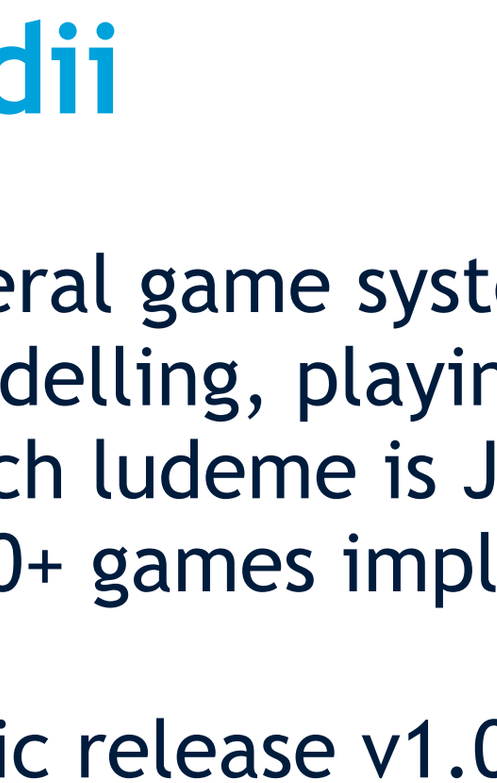
Ludii

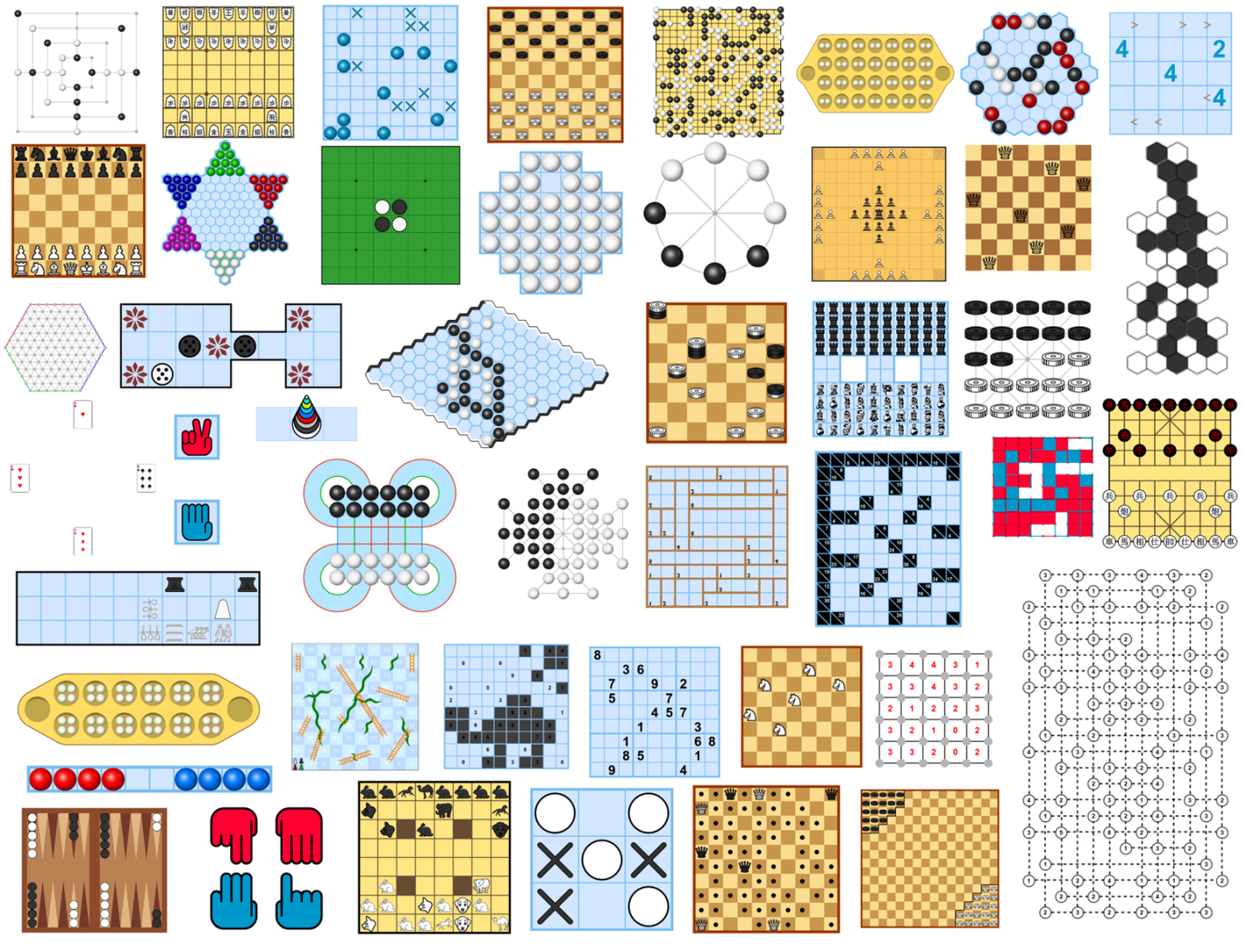
General game system:

- Modelling, playing, analysing, generating, reconstructing, …
- Each ludeme is Java class
- 250+ games implemented

Public release v1.0.0

- July 2020
- http://ludii.games



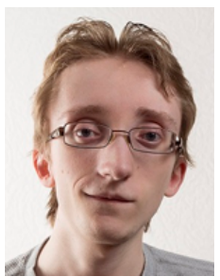


Team

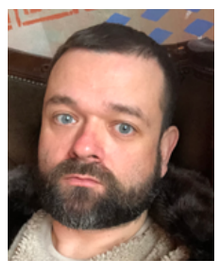
Cameron Browne (PI)
- Technical lead

Eric Piette (Postdoctoral Researcher)
- Game logic

Matthew Stephenson (Postdoctoral Researcher)
- GUI + networking



Dennis Soemers (PhD Candidate)
- AI + feature learning



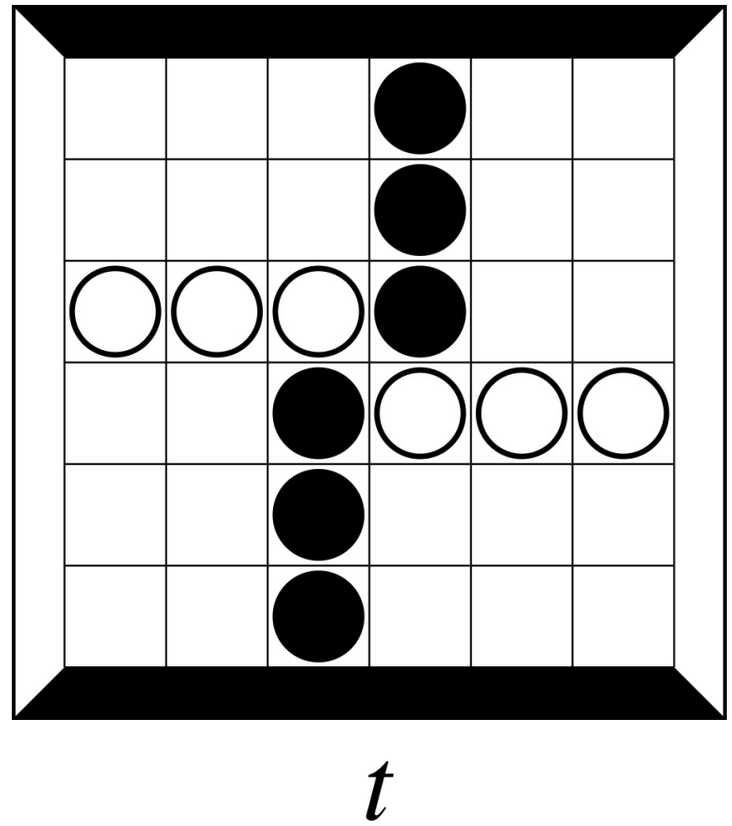
Walter Crist (Postdoctoral Researcher)
- Anthropologist/archaeologist (dispersal of games)

Many hours spent on board geometry…

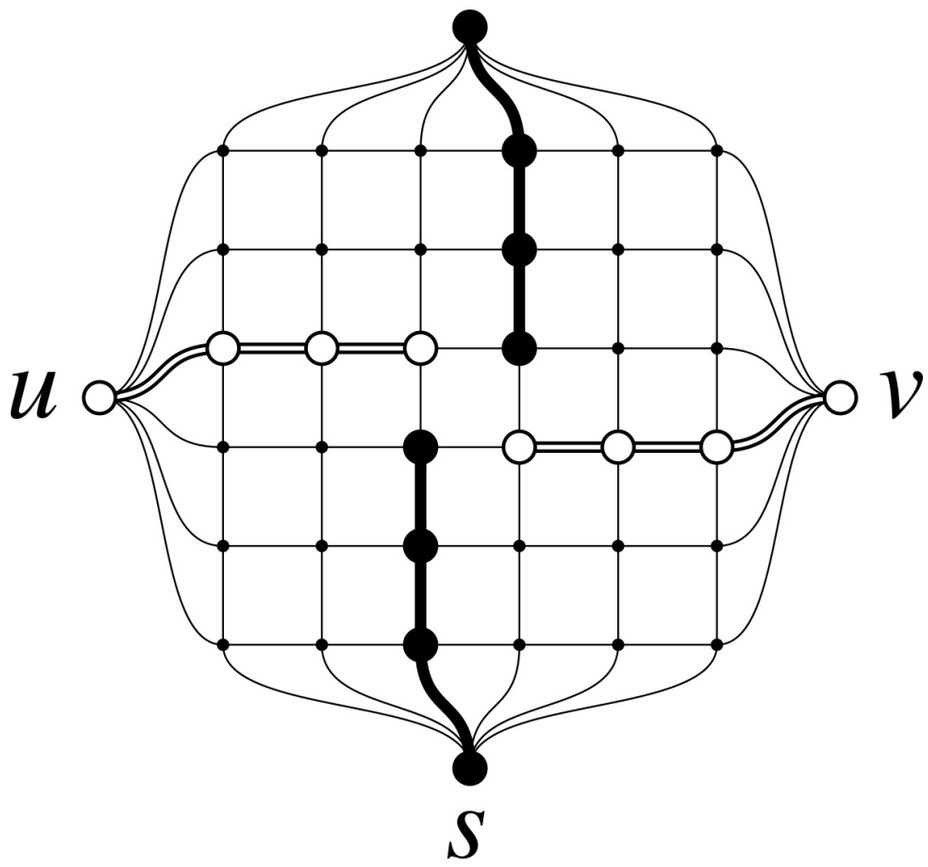
Boards as Graphs

C. Browne (2005) *Connection Games*
- Study of 200+ connection games

Games as graphs
- Vertices = playable sites
- Edges = adjacency

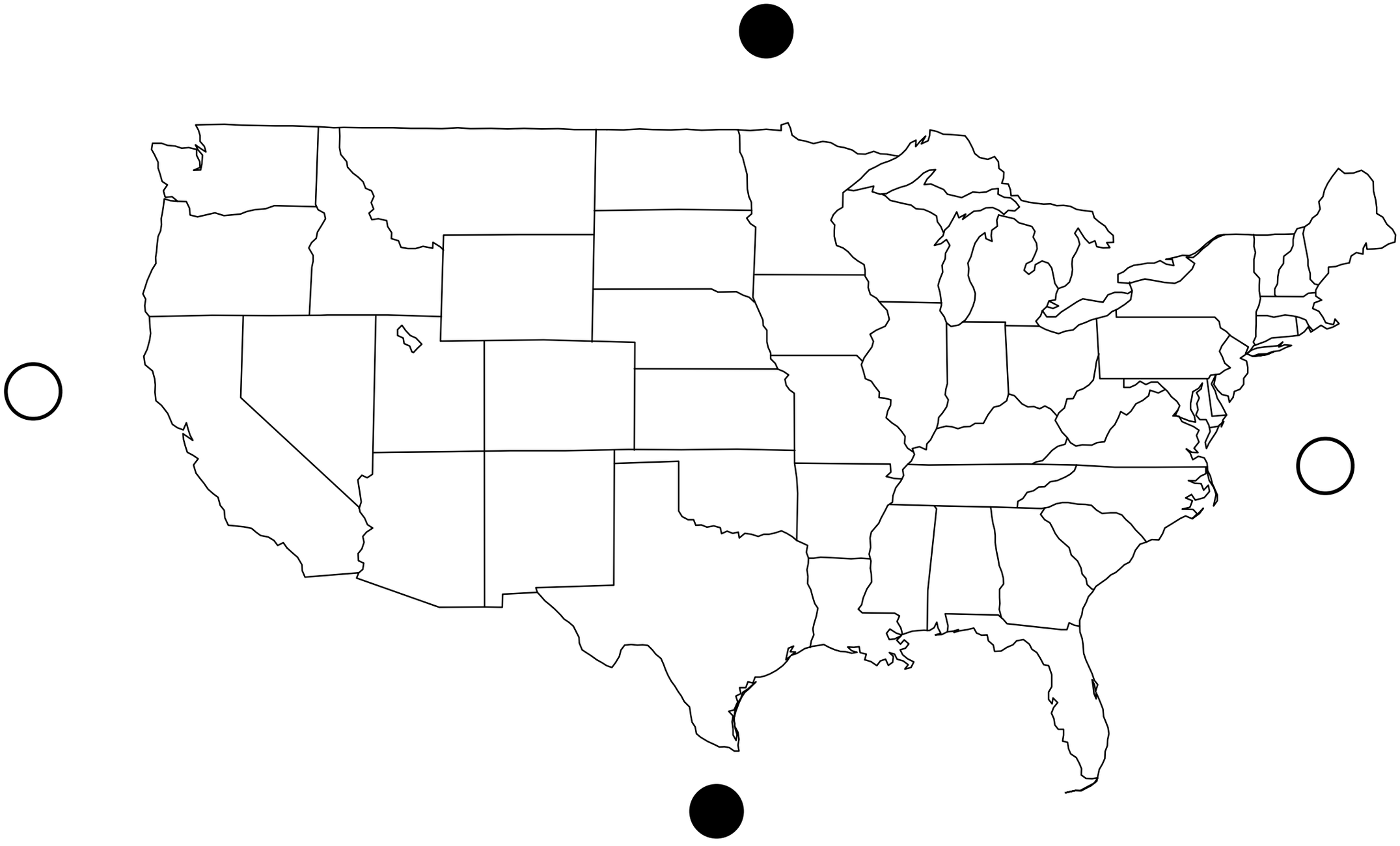


Strips games down to reveal underlying structure

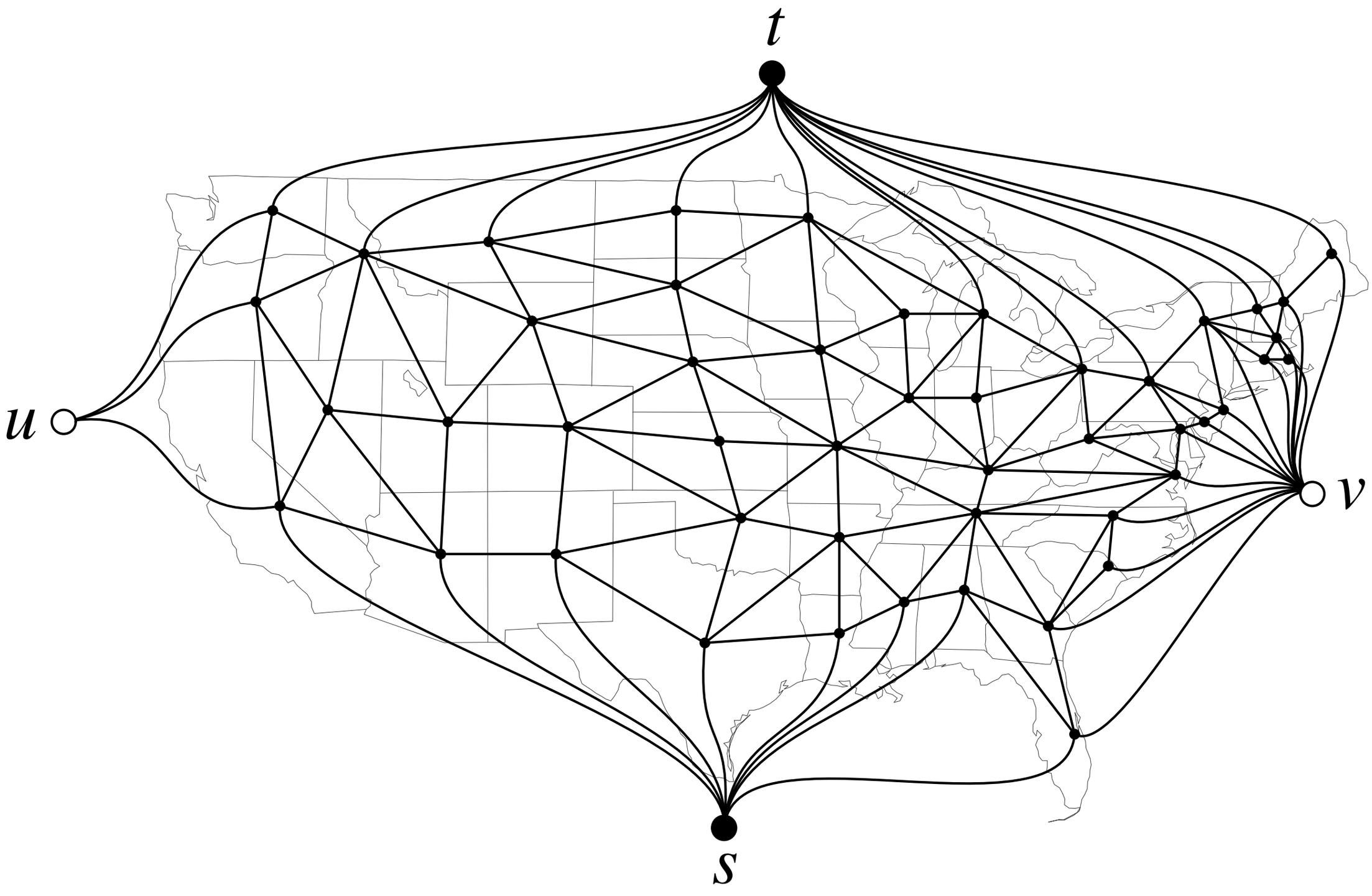


Games as Graphs

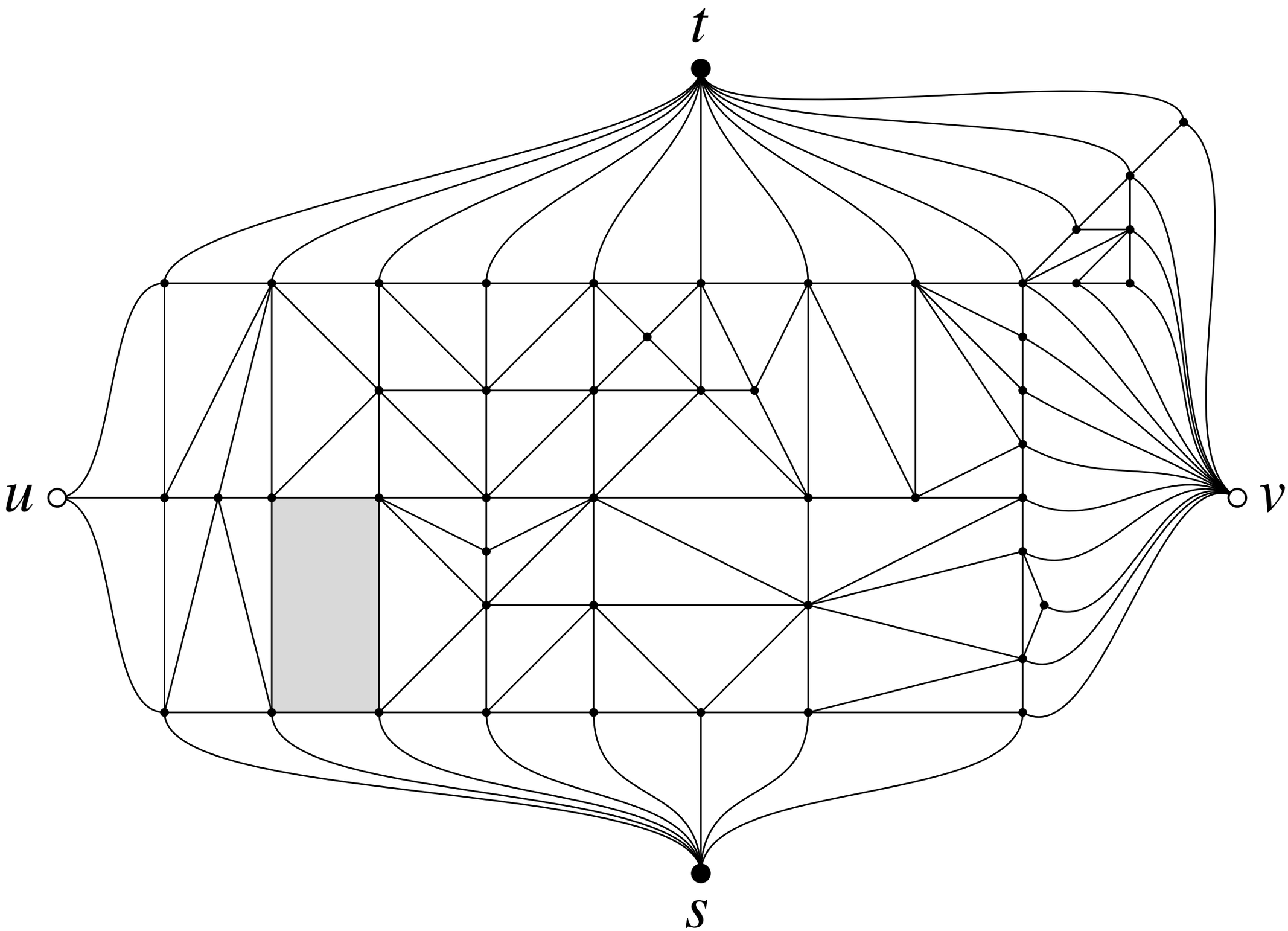
e.g. Map Hex



Games as Graphs

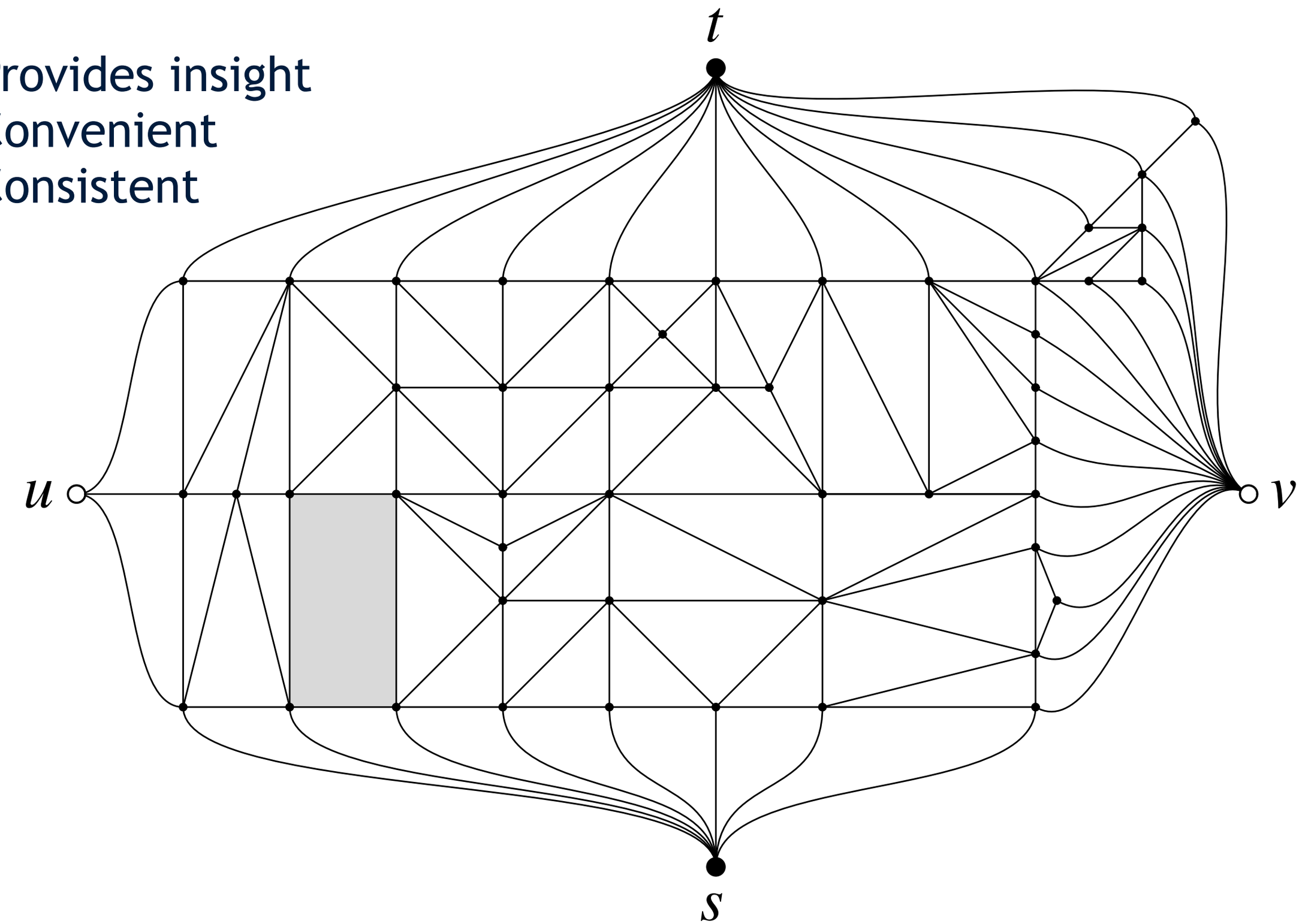


Games as Graphs



Games as Graphs

Provides insight
Convenient
Consistent



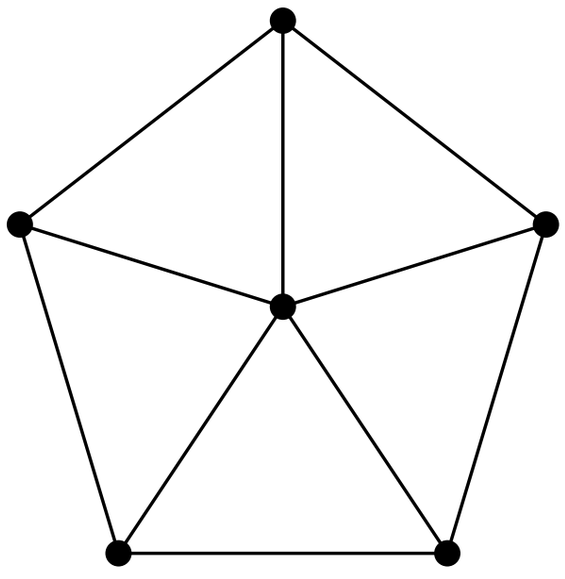
Basic Graph Theory

A **graph** G is a set of vertices and edges $G = \{V, E\}$

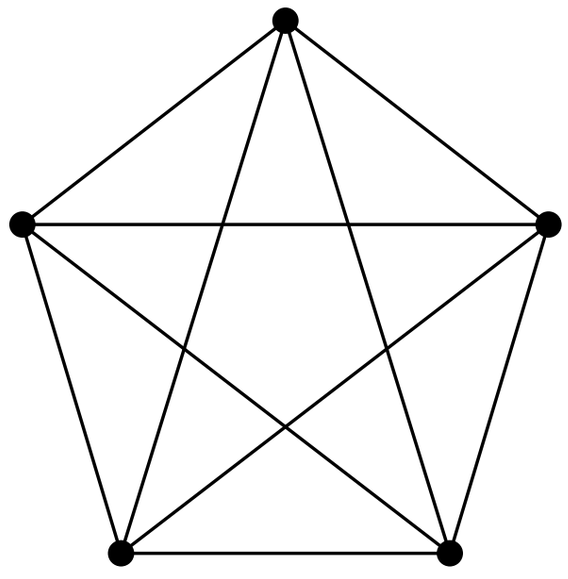
A **vertex** V_n is a point in Euclidean space

An **edge** E_n is defined by two vertices (end points)

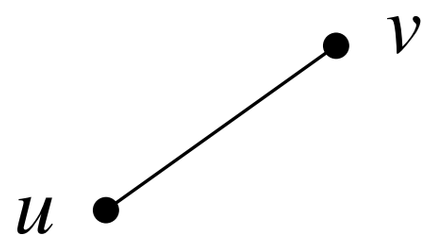
A **face** F_n is a closed region bounded by edges



5 faces



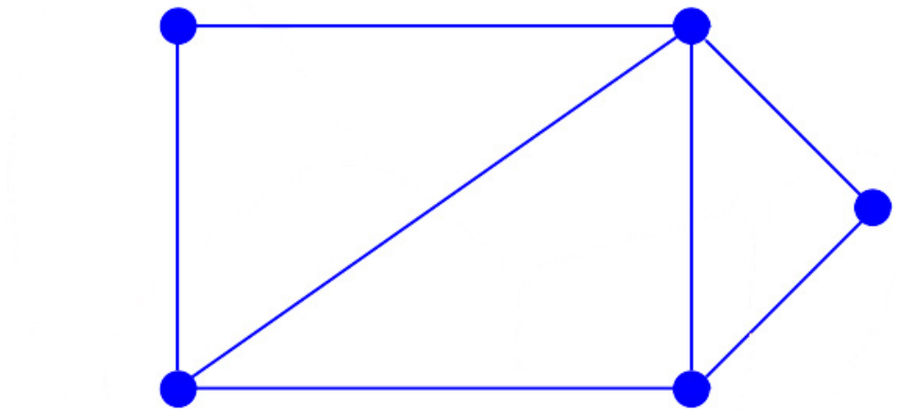
16 faces



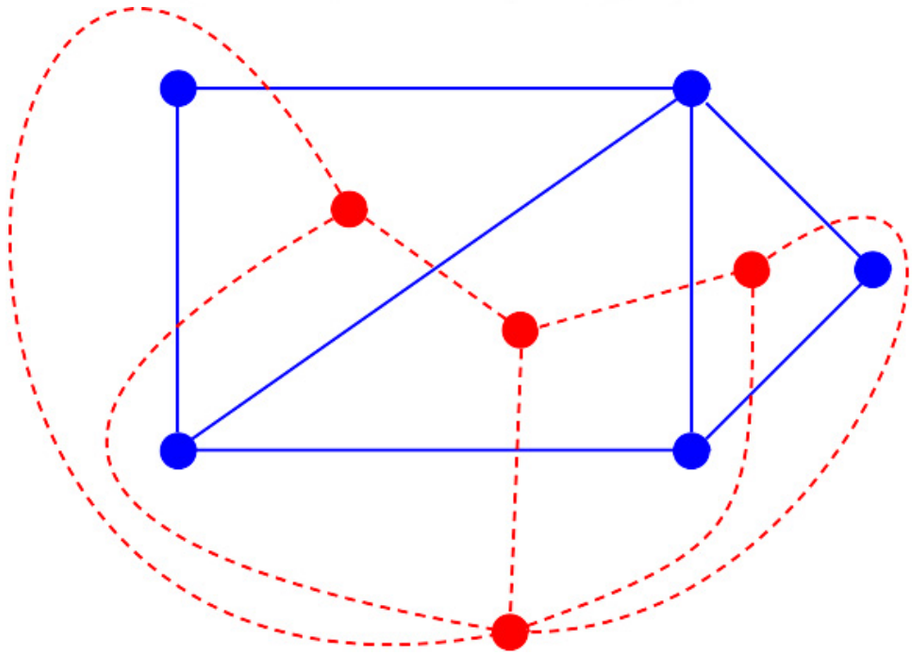
0 faces

Duals

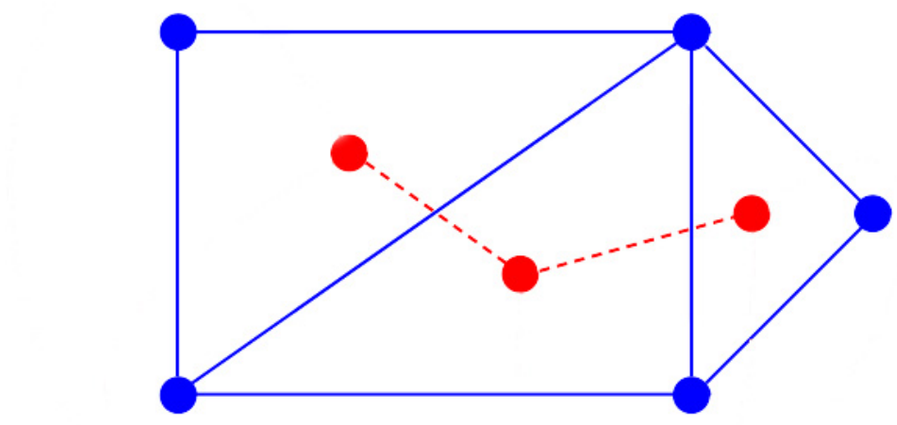
The **dual** of a graph *G* is the graph that has a vertex for each face of *G*, with edges between vertices of faces that share an edge



The **dual** includes a vertex for the outer region as an infinite face



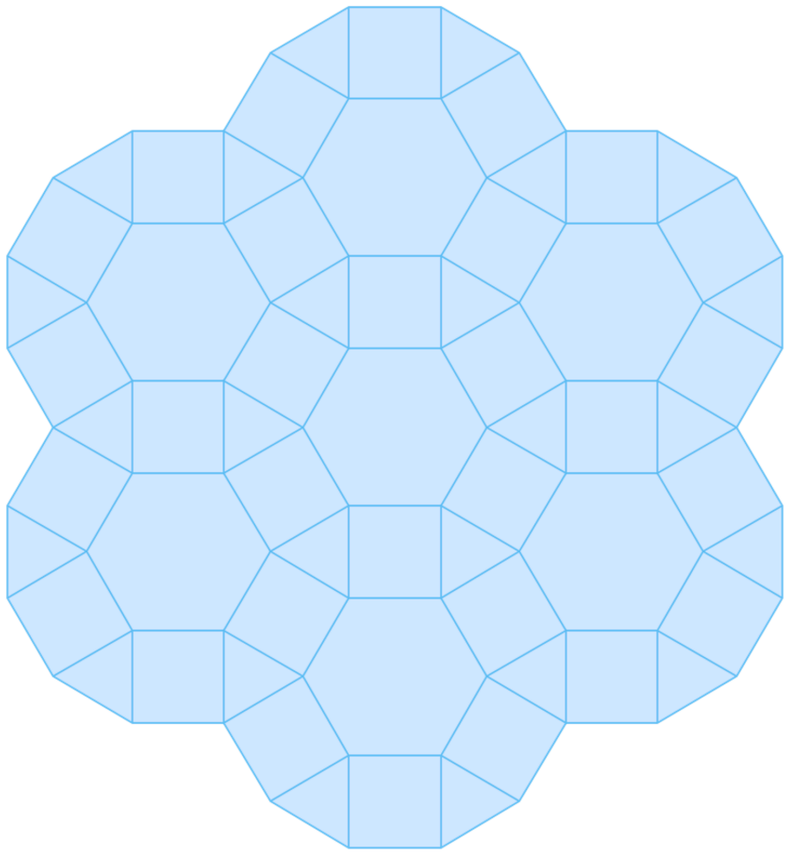
The **weak dual** of a graph *G* is the dual whose vertices correspond to **bounded faces only**

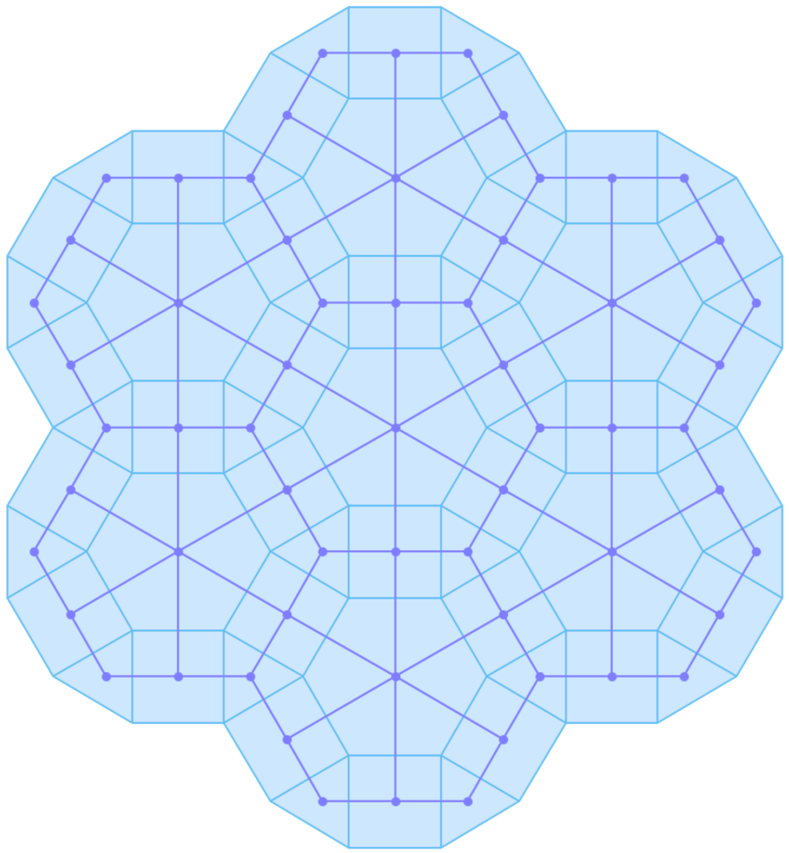


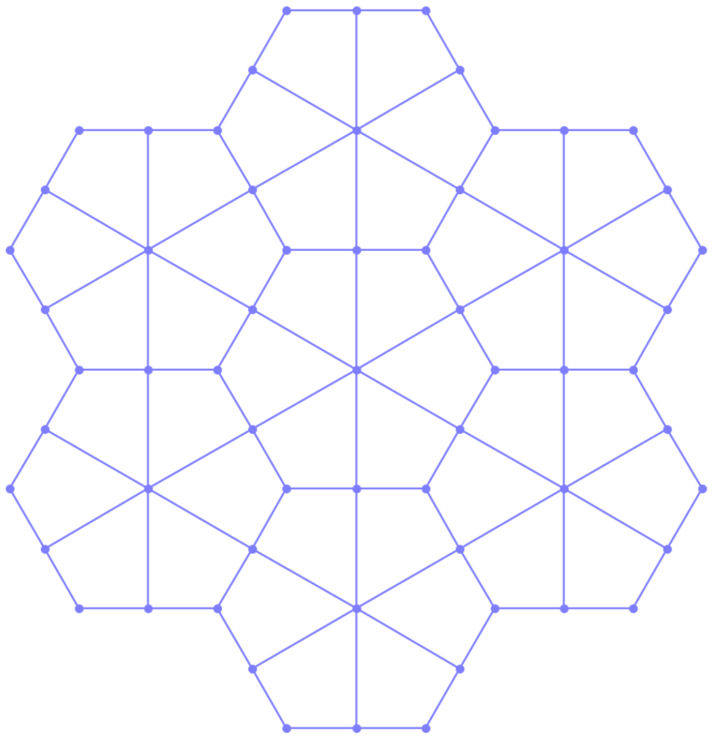
Weak Dual

Useful for defining:

- Where pieces go (*playable sites*)
- How they can move (*adjacency*)



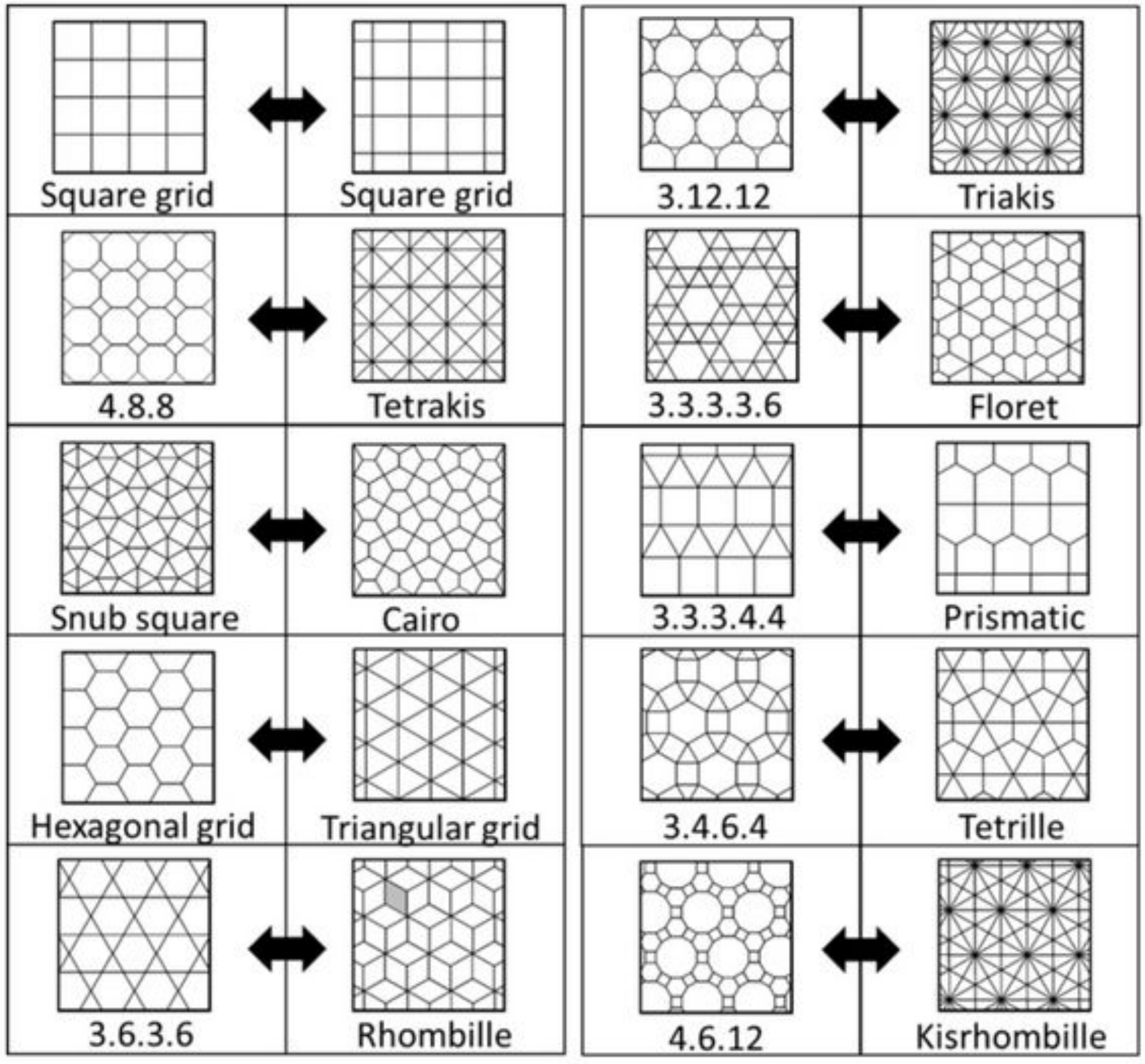




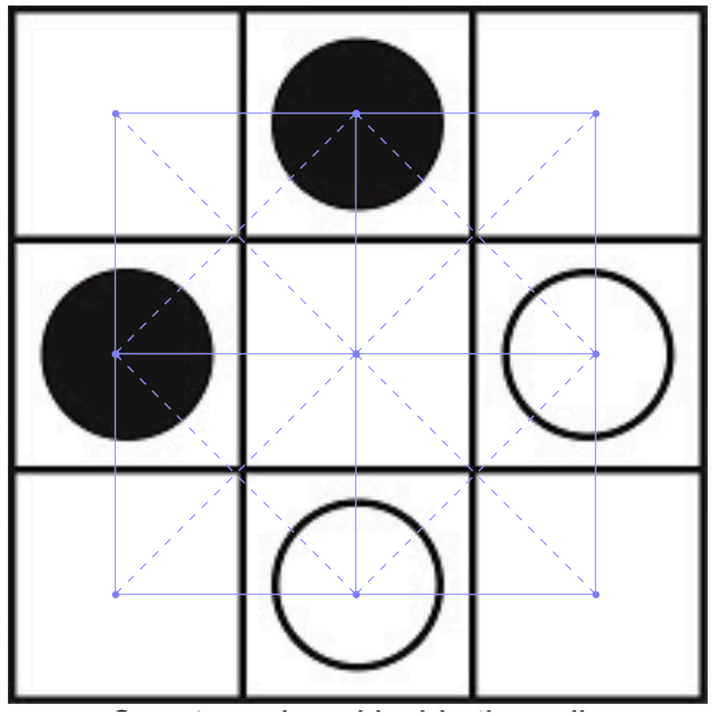
Weak Dual

Also useful for generating exotic boards from simple tilings

Very powerful!



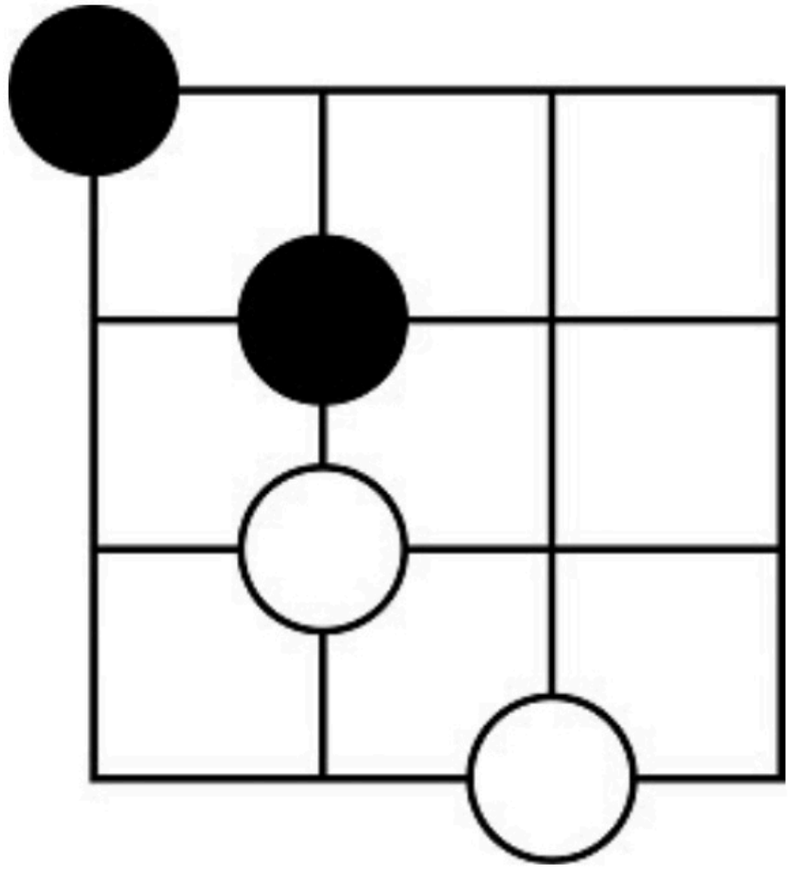
Ludii Board Representation



Initially maintained two separate graphs:

1. Weak dual of actual board
   - For game logic
     (if played on the cells)

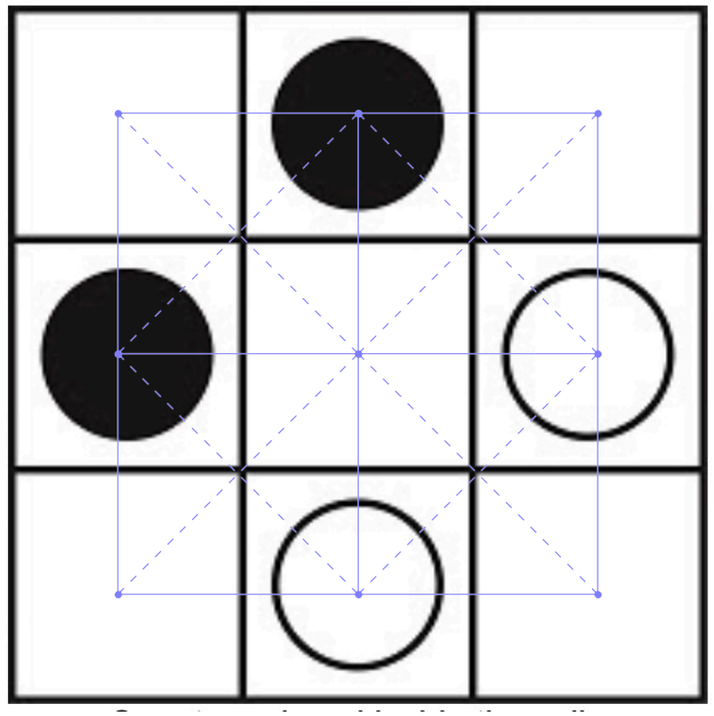
2. Graph of actual board
   - For drawing purposes
   - Also for game logic
     (if played on the intersections)



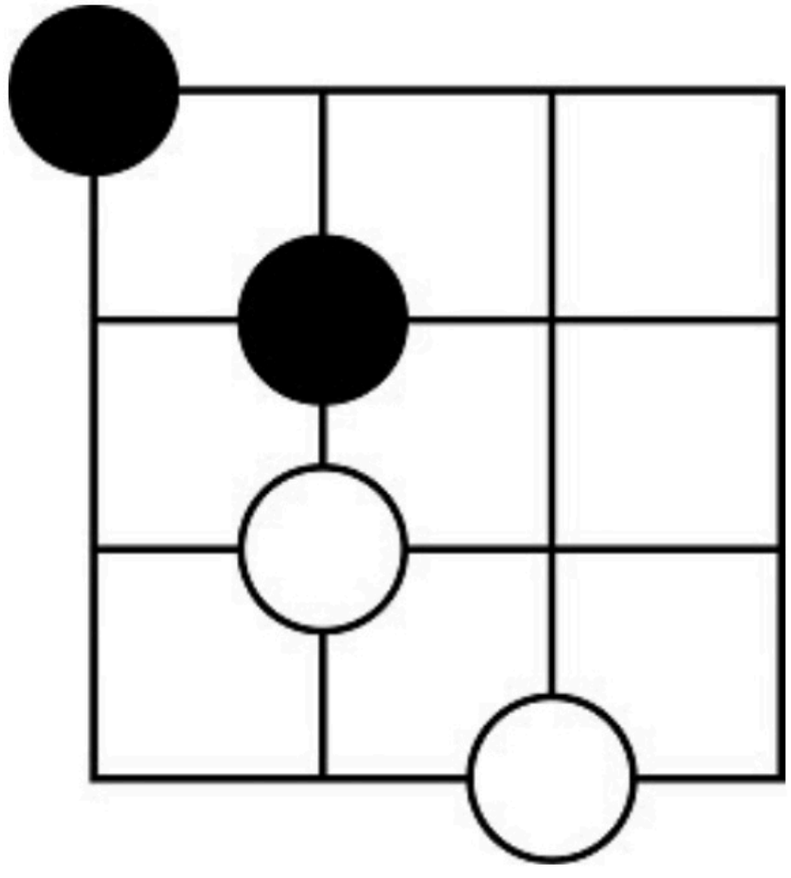
Issues

Duplication
Often just needed one graph

Size Confusion
Same board is:
- 3x3 if played on the cells
- 4x4 if played on the intersections



Ambiguity
Dual of the square grid is also a square grid – which to use?
e.g. Mancala (more hours…)



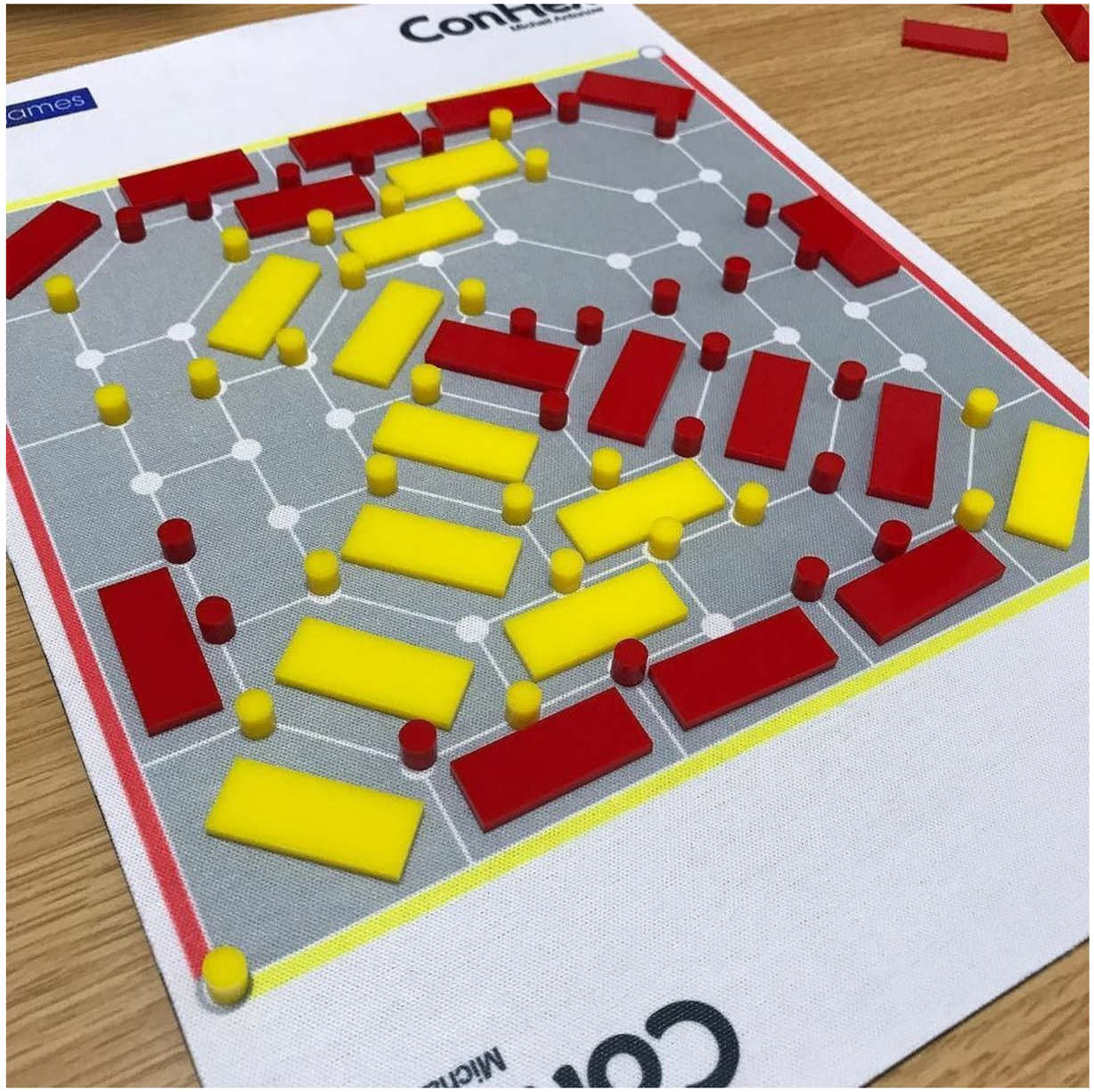
Issues

Games can be played on:
- Cells
- Vertices
- Edges
- Any combination of these
  e.g. Conhex

Two element types?
- Spread across both graphs

Three element types?
- No

Persisted about a year
- Not a good general solution



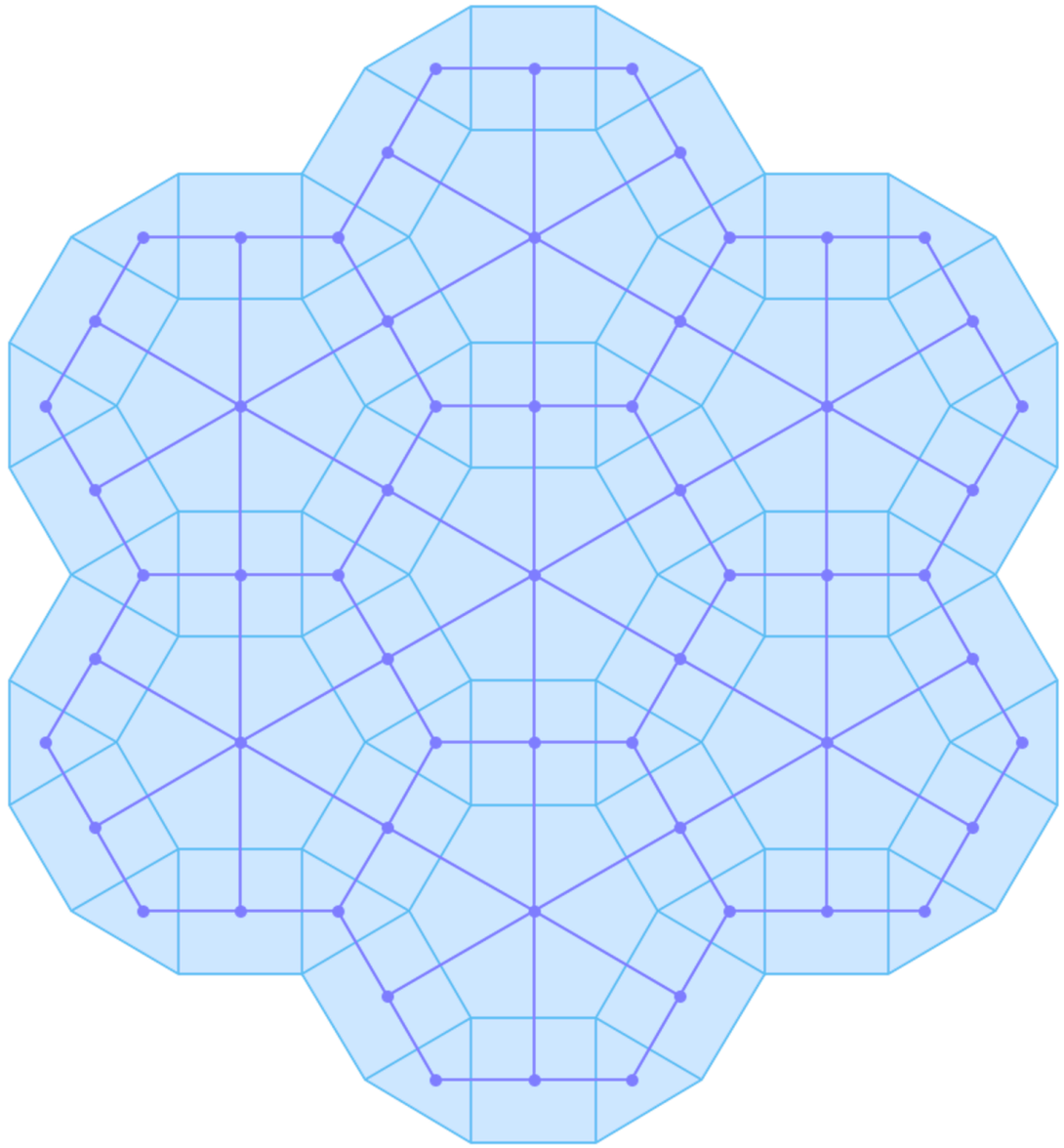
Solution

Just use one graph (thanks Matthew!)

Single graph with playable sites:

- Vertices
- Edges
- Cells

Use any combination of sites at any time



Implementation

Use “game design” term

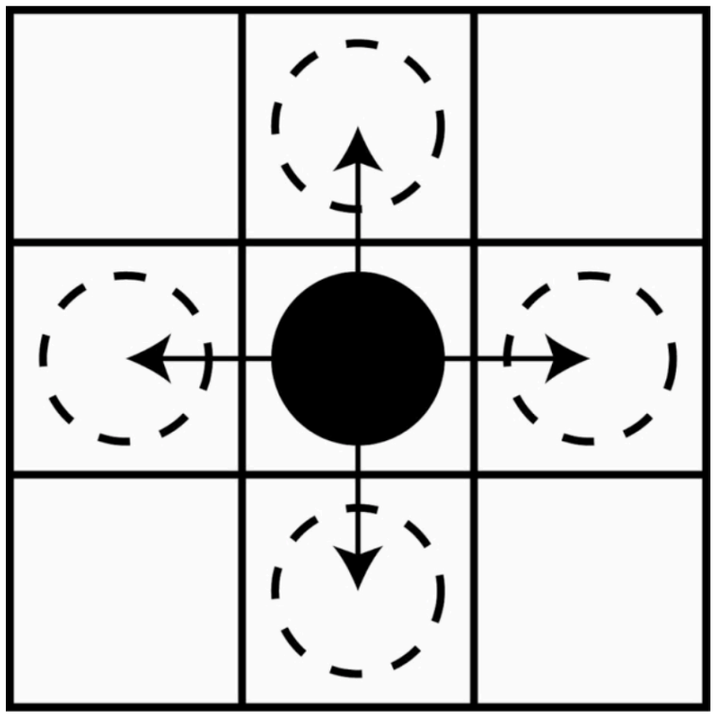
```
public class Topology implements Serializable
{

    /** List of cells. */
    private final List<Cell> cells = new ArrayList<Cell>();

    /** List of edges. */
    private final List<Edge> edges = new ArrayList<Edge>();

    /** List of vertices. */
    private final List<Vertex> vertices = new ArrayList<Vertex>();

    /** Reference to generating graph, so board styles and designs can determine underlying basis. */
    private Graph graph = null;
```

Will describe generating graph function shortly

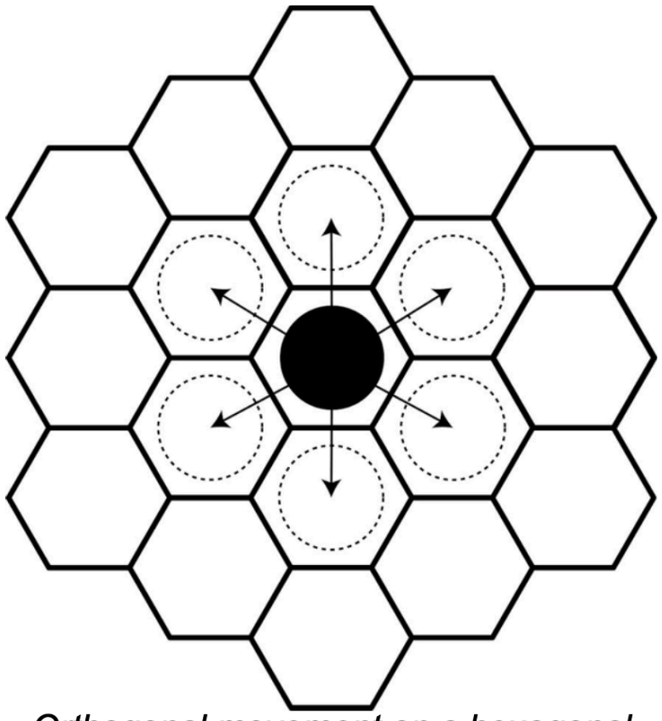
Naming Convention

Policy to use “game design” terms:
- Unless confusing
- If give better game descriptions

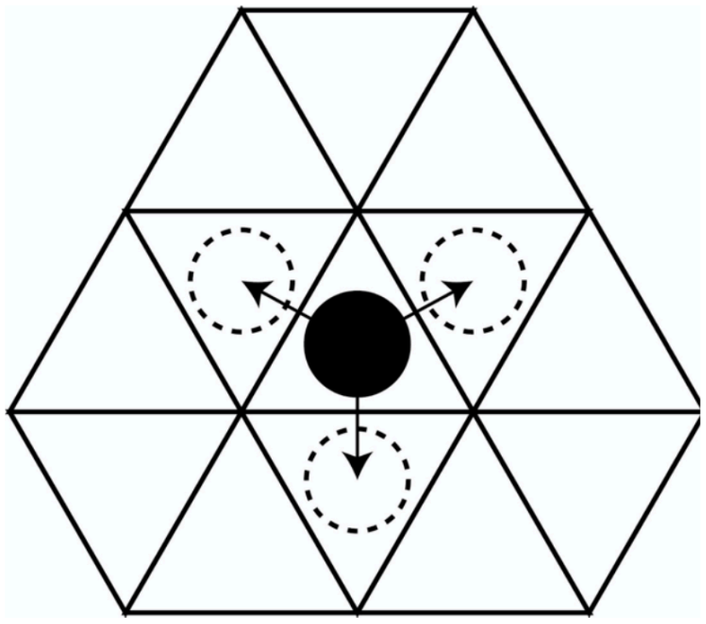
e.g. ***Orthogonal***
Game design interpretation: *Cells that share an edge*



Orthogonal movement on a square grid



Orthogonal movement on a hexagonal grid

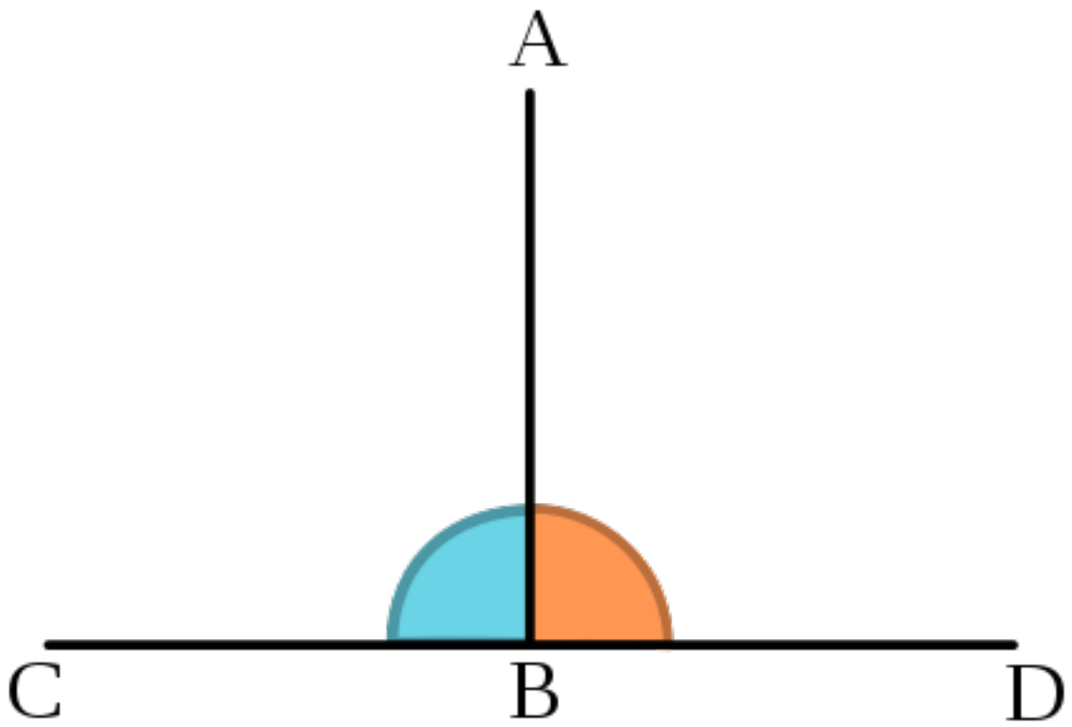


Orthogonal movement on a triangular grid

Orthogonality

Orthogonal

Mathematical definition: *Two lines that meet at right angles*

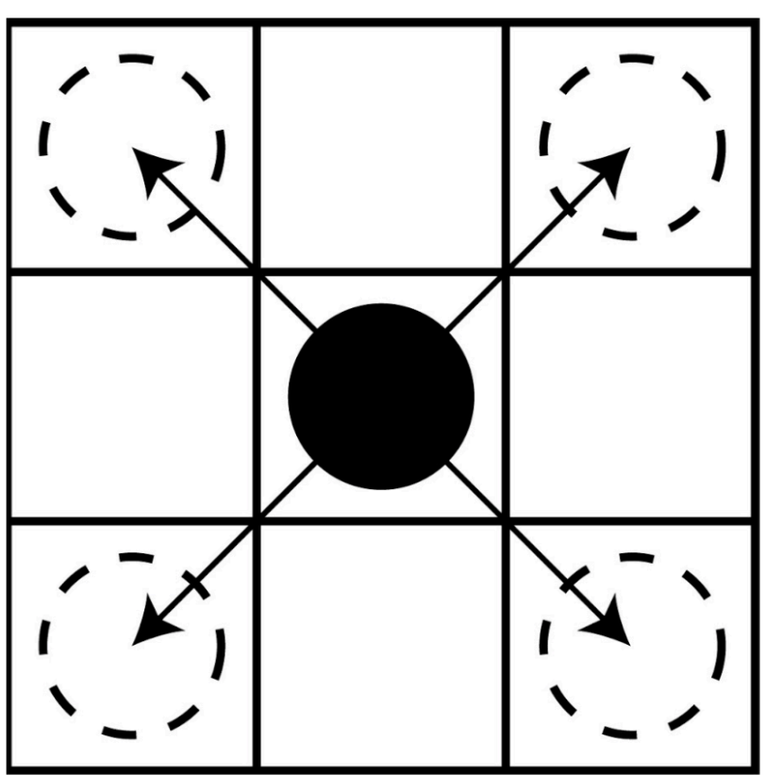


Square grid: 90°
Hexagonal grid: 60°
Triangular grid: 30°

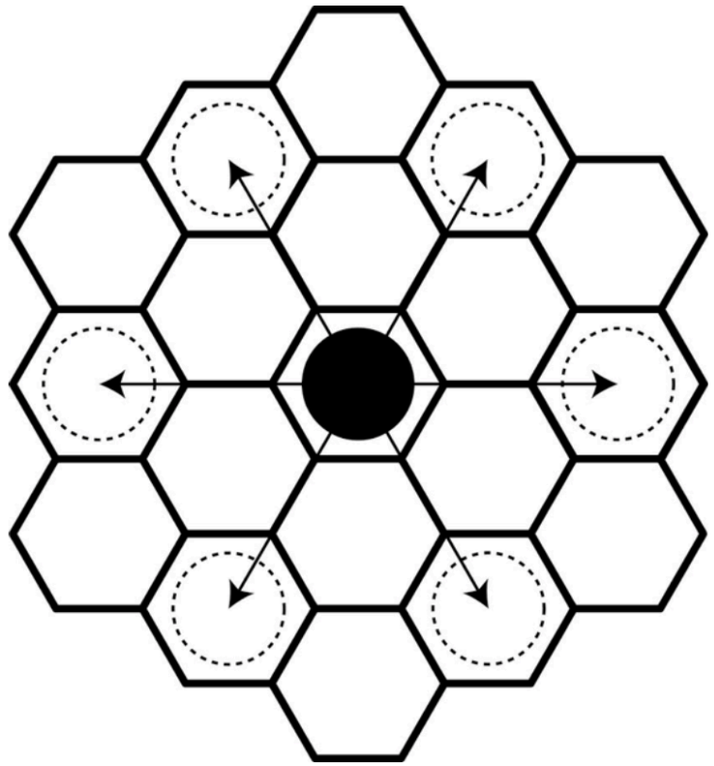
Diagonality

Diagonal

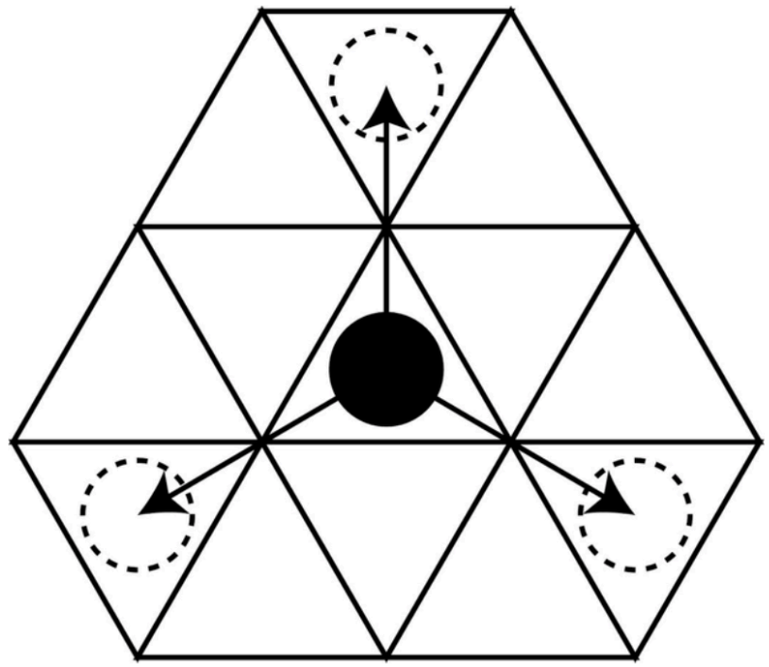
Game design: *Closest non-orthogonal cell through each vertex*



Diagonal movement on a square grid



Diagonal movement on a hexagonal grid



Diagonal movement on a triangular grid

Can be non-adjacent, e.g. hexagonal

Diagonality

Diagonal

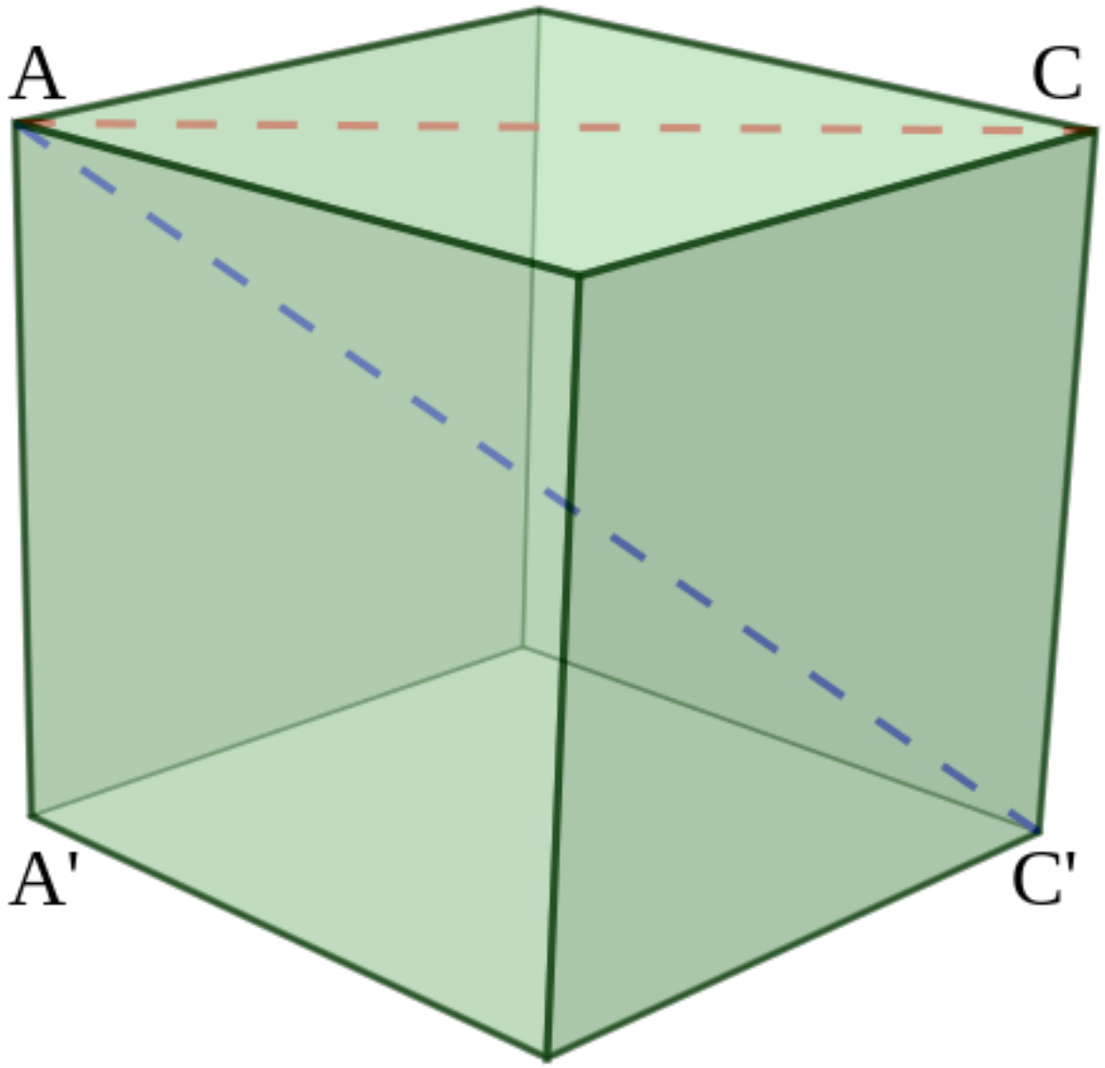
Mathematical definition: *Line segment between two vertices that do not define an edge*

“Mathematical” diagonals occur within the face

“Game design”diagonals project outwards from cells

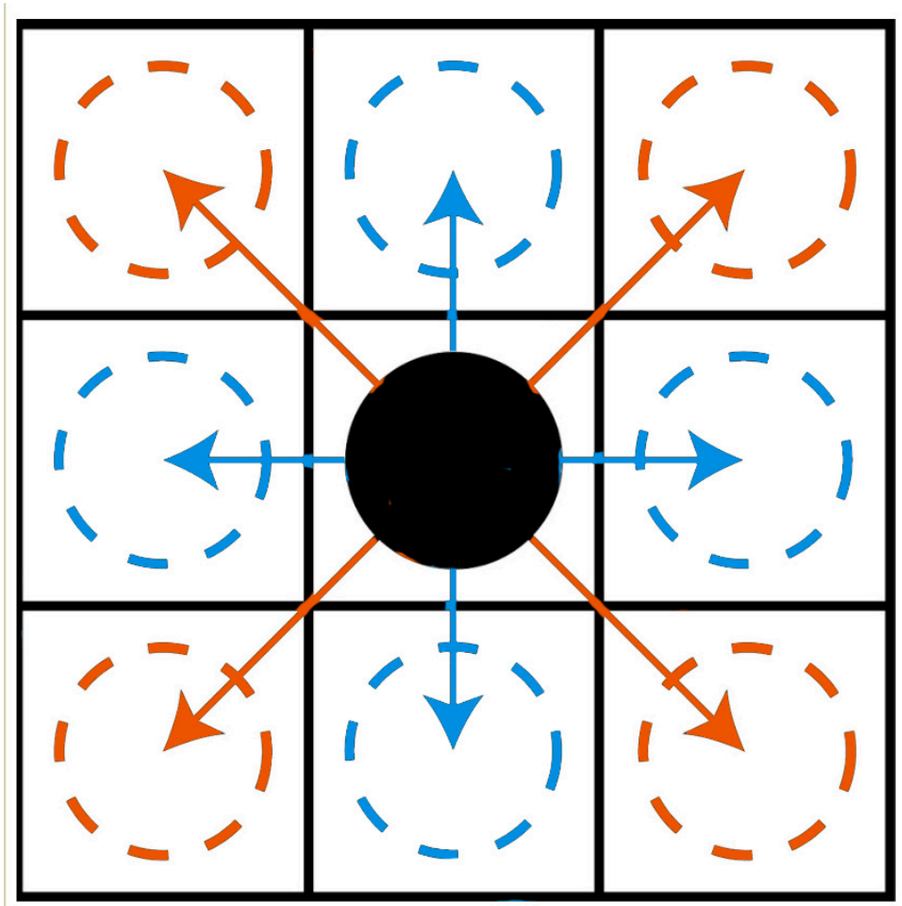
Triangular grid:

- 3 sides hence no true diagonals!



Alternative Naming Schemes

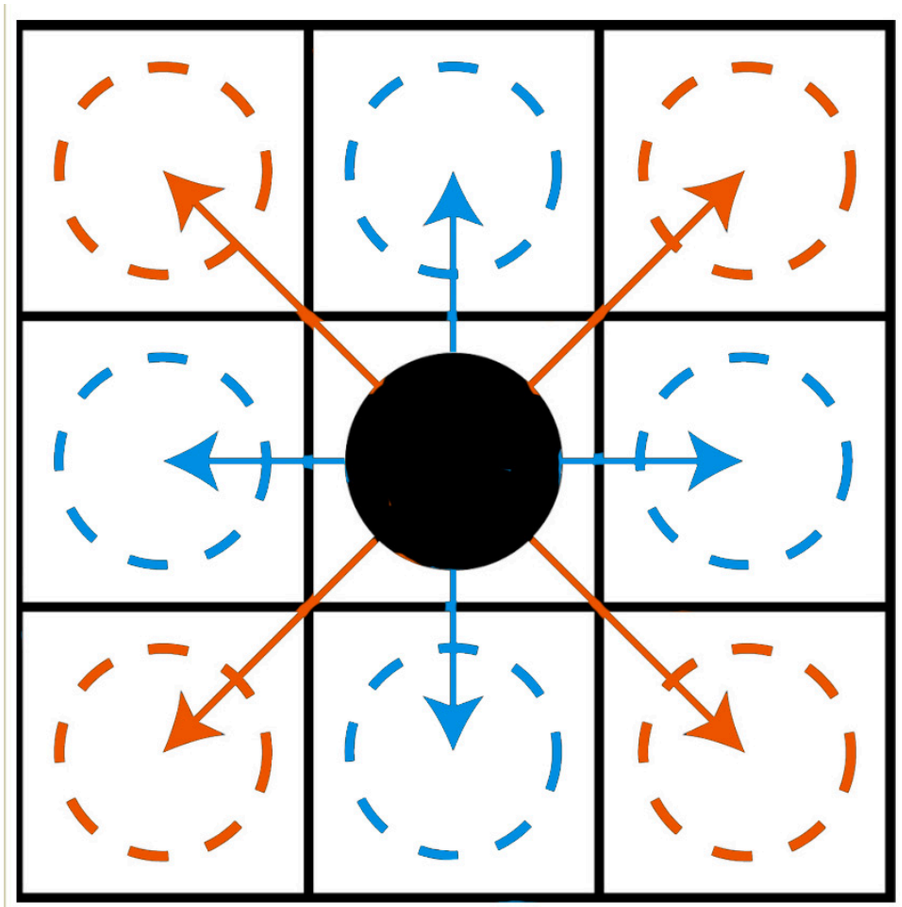
Orthogonal => “Axial” = lines along major axes
Diagonal => “Angled” = other lines



Axial Angled

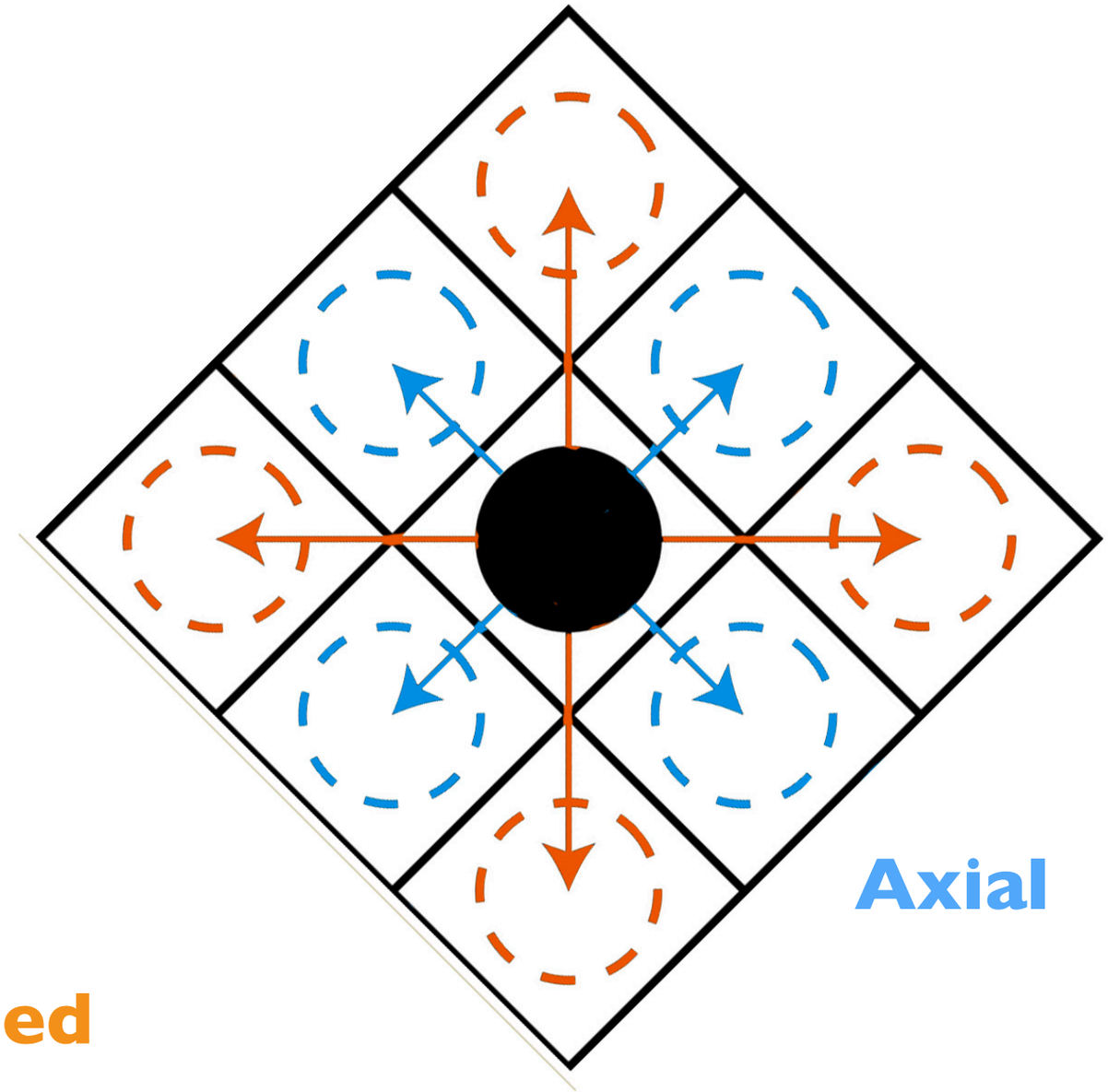
Alternative Naming Schemes

Orthogonal => “Axial” = lines along major axes
Diagonal => “Angled” = other lines



Axial

Angled



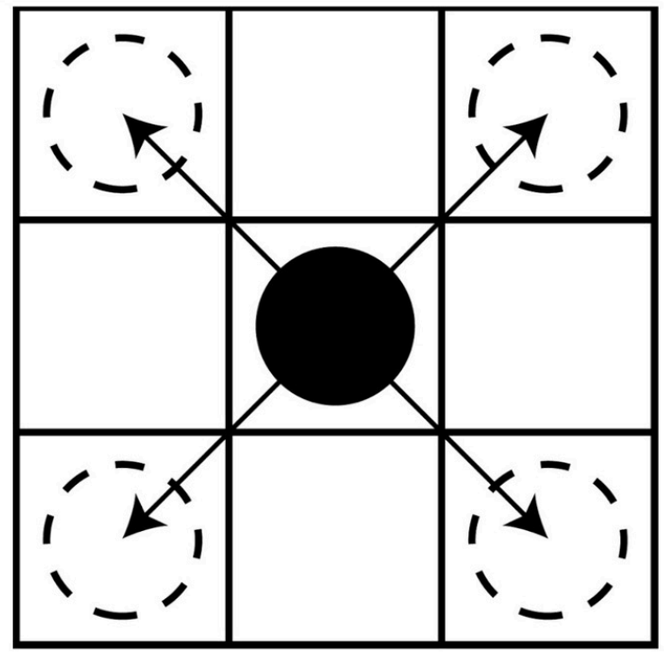
Axial

Angled

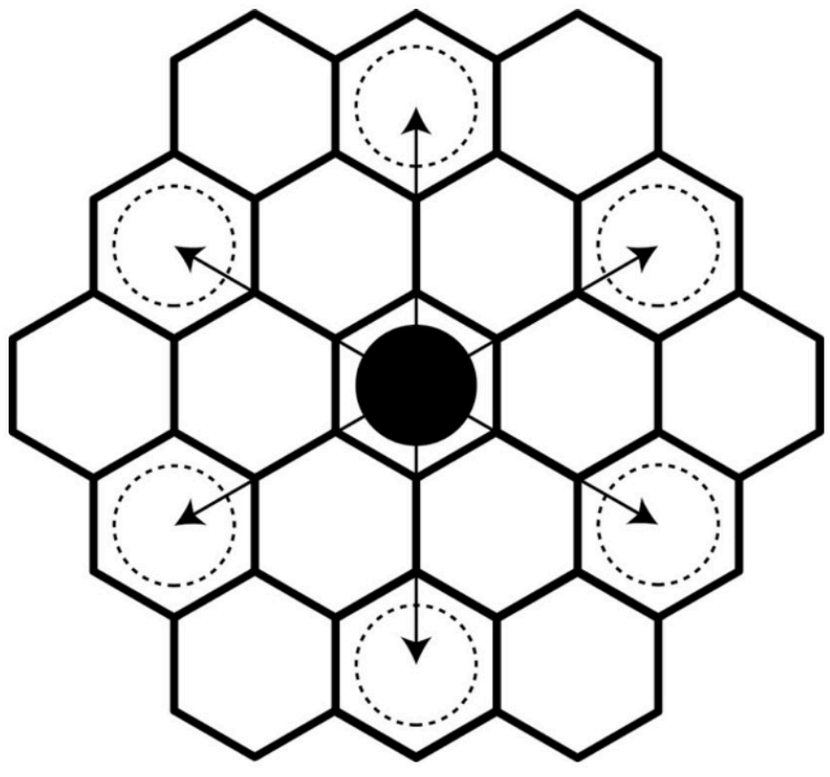
Types of “Diagonals”

Different types of “diagonals”:

1. *Connected* e.g. square grid
   - Vertex-connected but not edge-connected



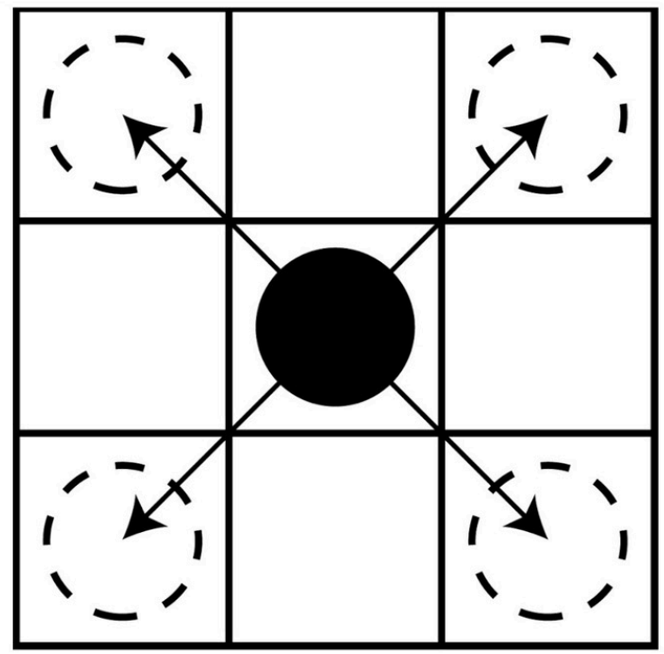
2. *Disconnected* e.g. hex grid
   - Neither vertex connected nor edge-connected



No term in graph theory describes this distinction!

Call them “adjacent” and “non-adjacent”

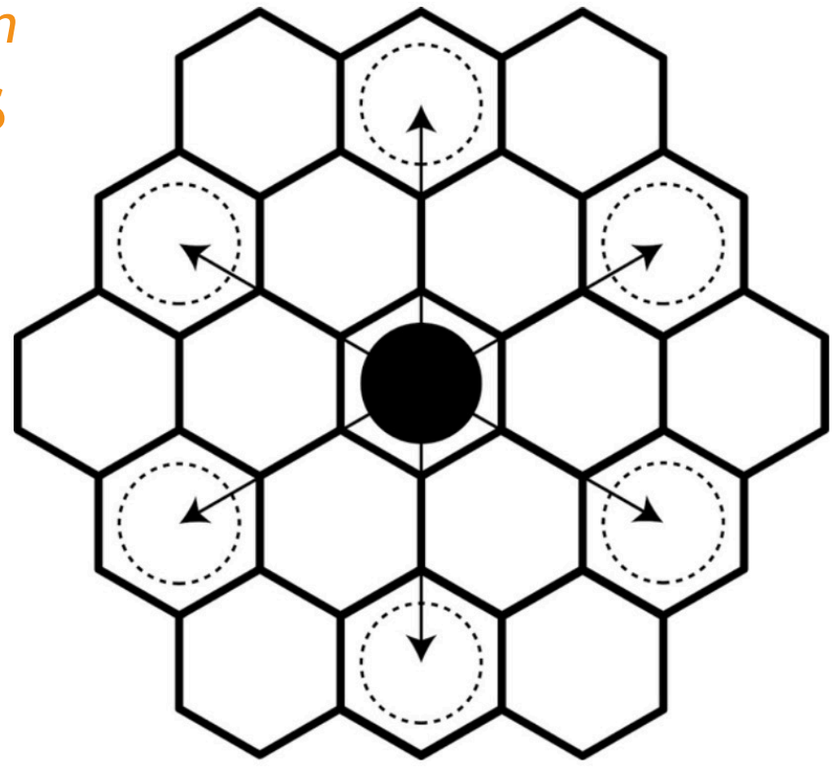
Diagonal Connectivity



Our Definitions

- *Orthogonal*: Share an edge
- *Diagonal*: Cells in line through a vertex

i.e. for each vertex V_n of each cell C, project a line from midpoint C_m through V_n and find the closest cell whose midpoint is approximately coincident

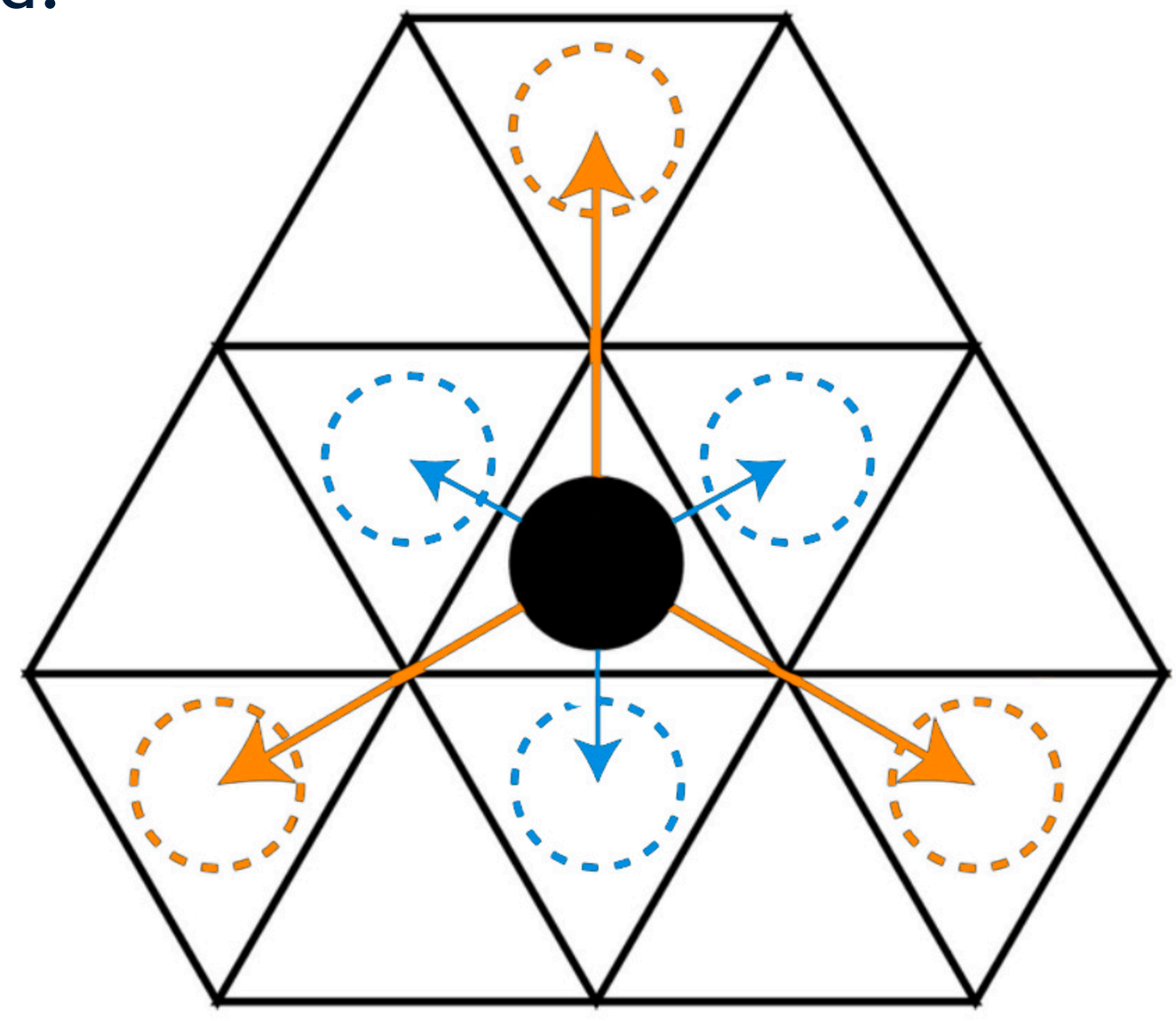


- *Adjacent*: Share at least one vertex
- *Non-adjacent*: No shared vertex
- *Neighbour*: Any orthogonal or diagonal

Off-Diagonals

Relations on the triangular grid:

- Orthogonal
- Diagonal
- ?



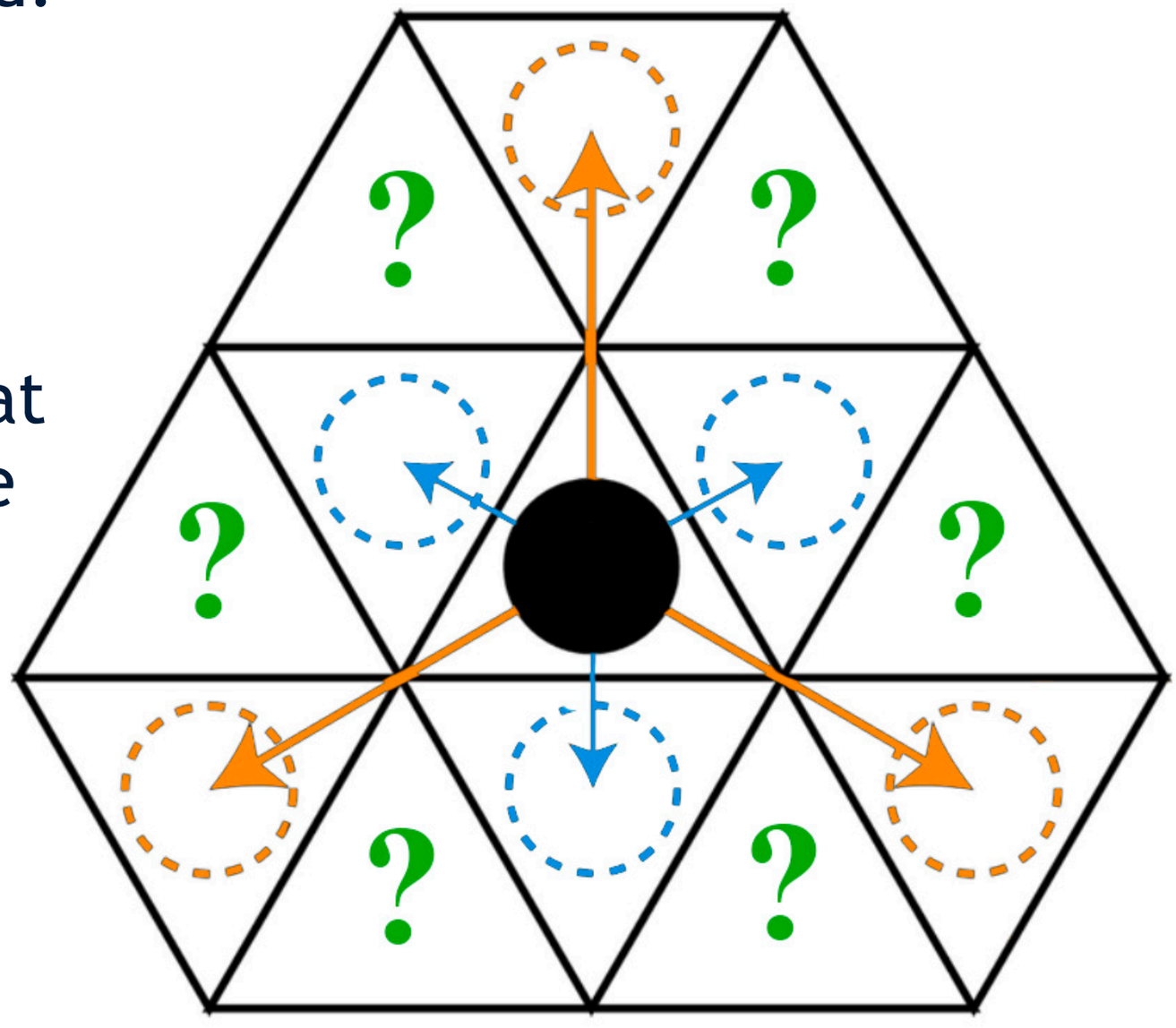
Off-Diagonals

Relations on the triangular grid:
- Orthogonal
- Diagonal
- Off-diagonal

Off-diagonal : those nbors that share a vertex but not an edge and are not true diagonals

Form of “secondary” diagonal

Important for lines of play on the triangular grid (more later)



Directions

Each site has a reference point (*x*,*y*,*z*):
- *Vertex*: Location
- *Edge*: Midpoint
- *Cell*: Centroid

Defines directions between sites:

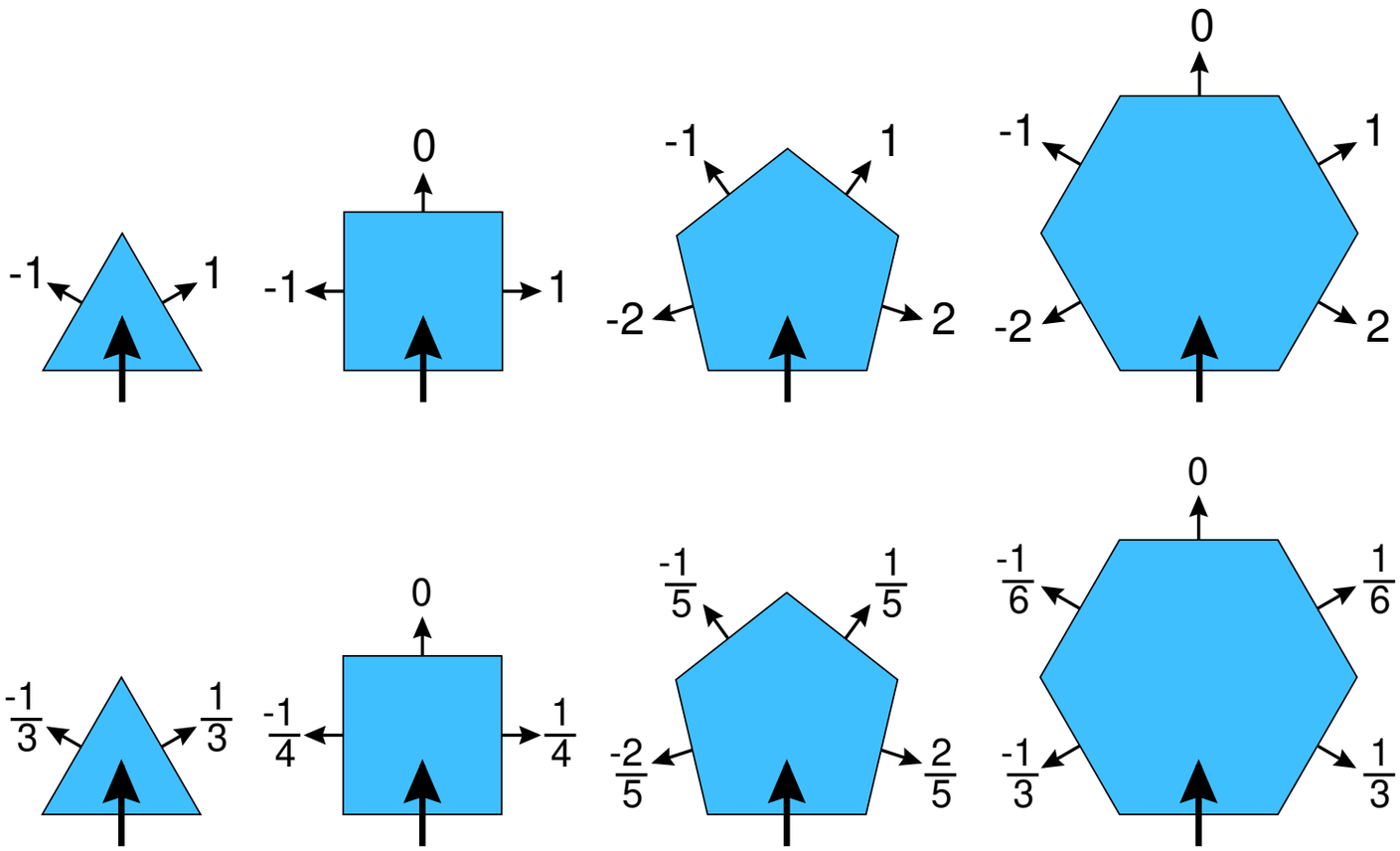
1. **Absolute**
   a) **Specific**
      i) *Compass* N, E, S, W, NE, NNE, …
      ii) *Rotational* In, Out, CW, CCW, …
      iii) *Spatial* U, D, UN, UNE, DN, DNE, …
   b) **Relational** Orthogonal, Diagonal, Off, Adjacent, …

2. **Relative** (to player and/or component)
   - F, B, L, R, Forward, Leftward, Leftward, Rightward, …

Steps and Walks

Generate steps to adjacent elements

A walk is a set of steps from a site in a direction

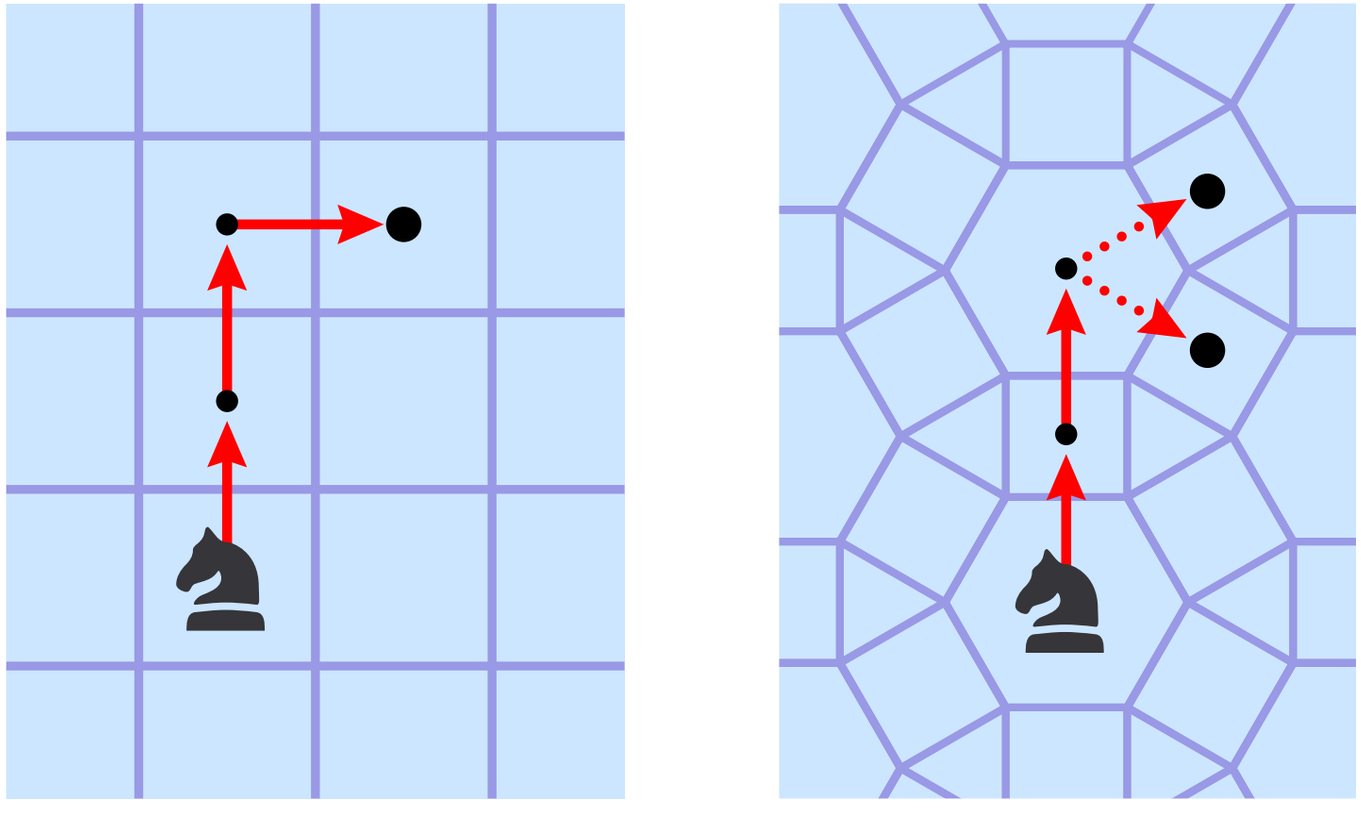


e.g. Knight walk:

{ {F F R} {F F L} }

Maps to any topology

- Can create ambiguity

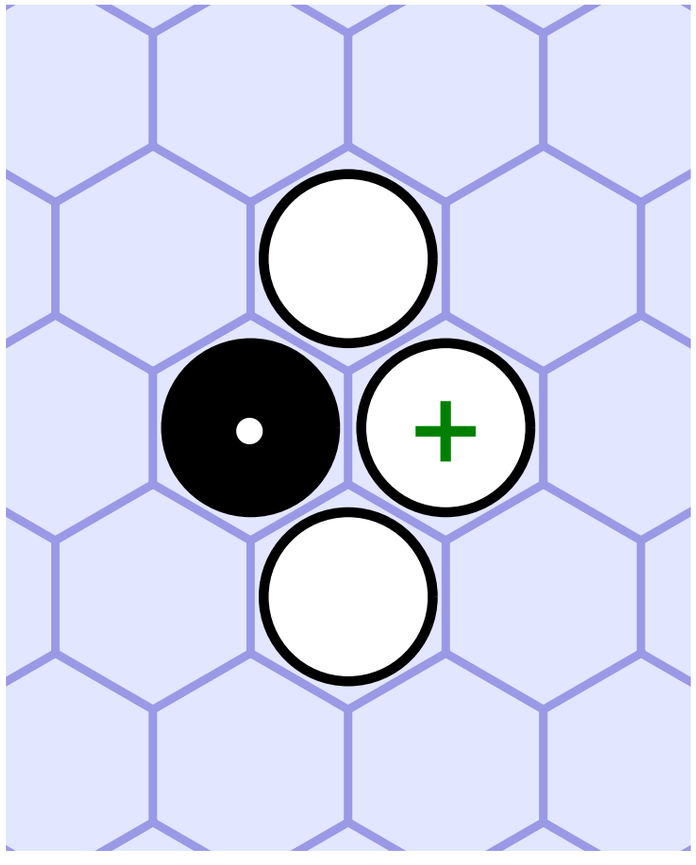


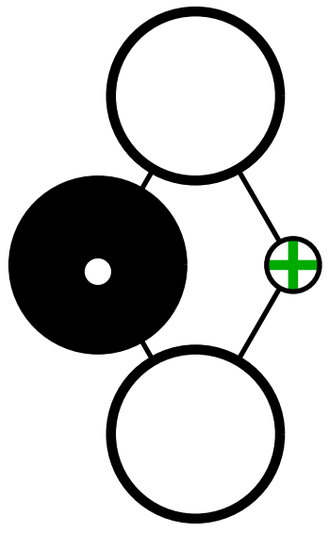
Features

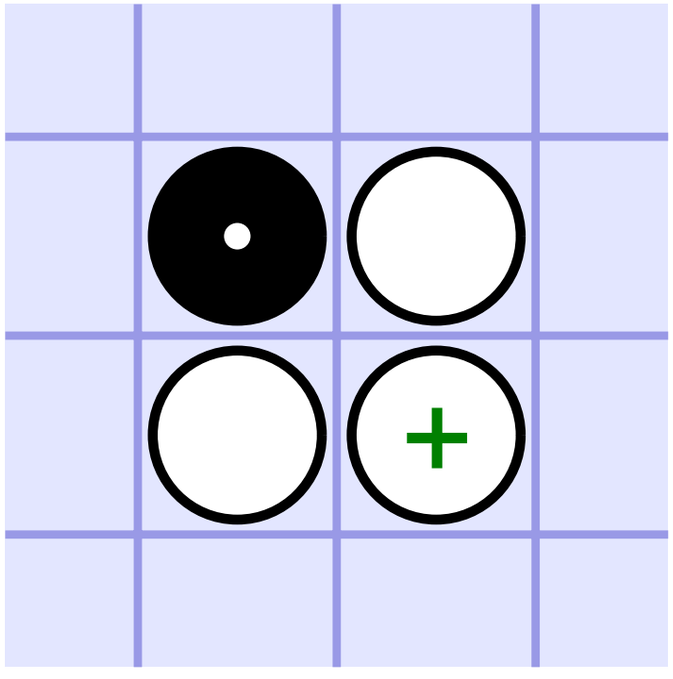
MCTS agent biases playouts according to learnt features:

- Geometric piece patterns defined by walks

e.g. “Bridge completion” pattern for connection games





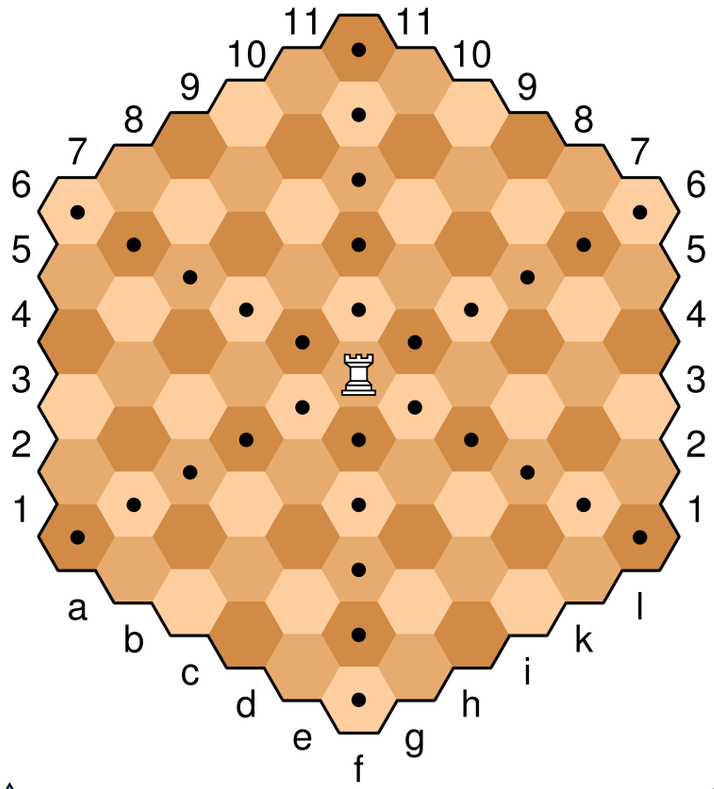


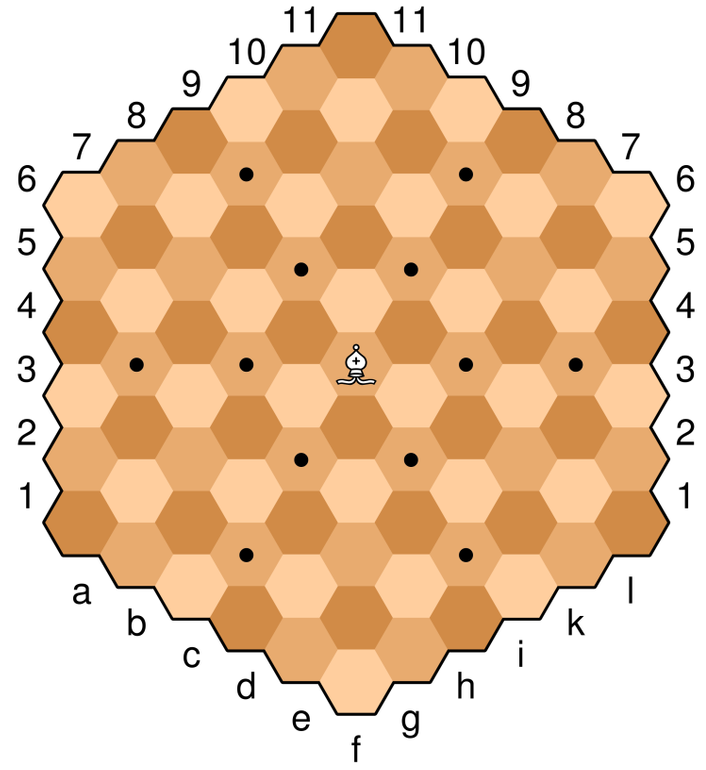
Maps easily between topologies

Radials

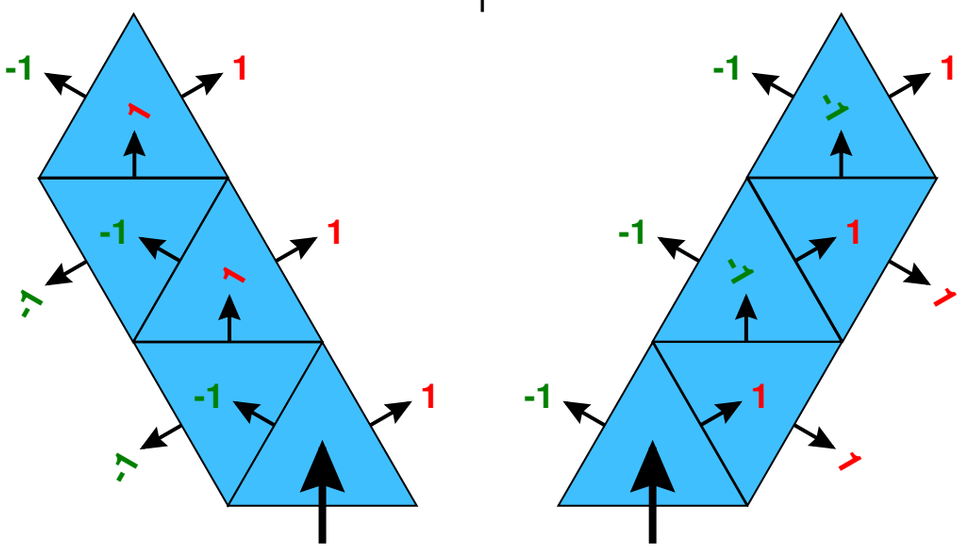
Radials are lines of play defined by regular step patterns

- Not necessarily contiguous





- Can branch



Radials

Radial generate(site S_0, dirn D_0)
{
    for (each nbor S_N from S_0 that satisfies D_0)
        $S = S_0$
        while (site != null)
        {
            store S
            $D = S_N - S$ <= current local direction
            find nbor(s) S_N from S that satisfies D_0
                and minimises deviation from D <= can branch
            $S = S_N$
        }
}

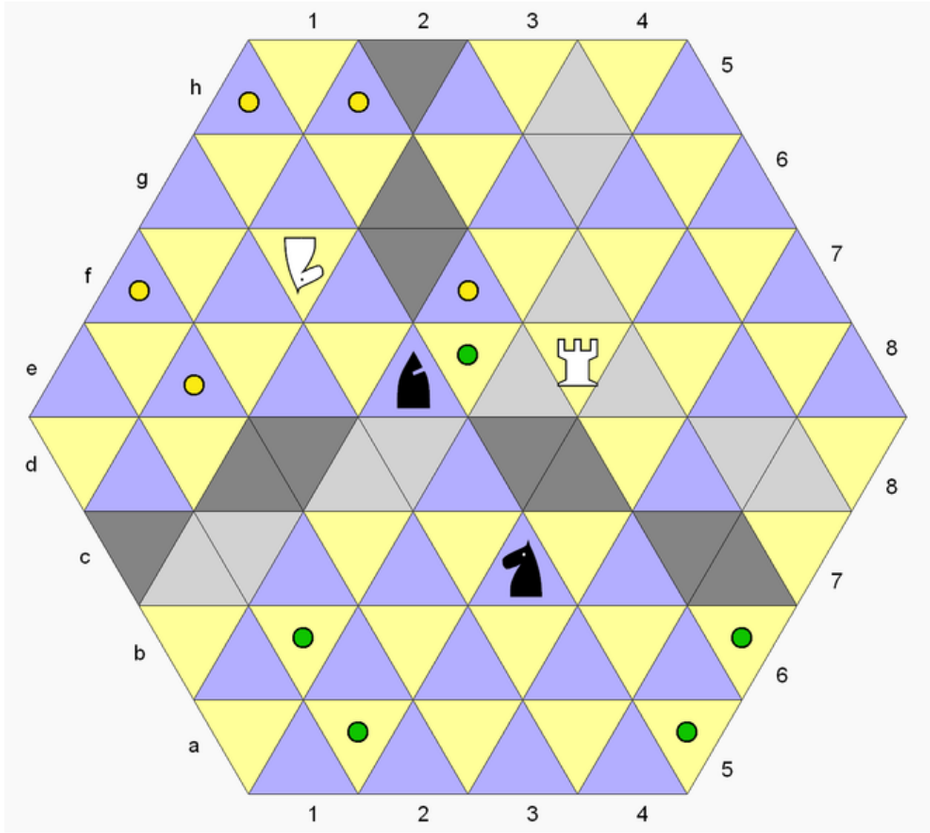
Triangular Radials

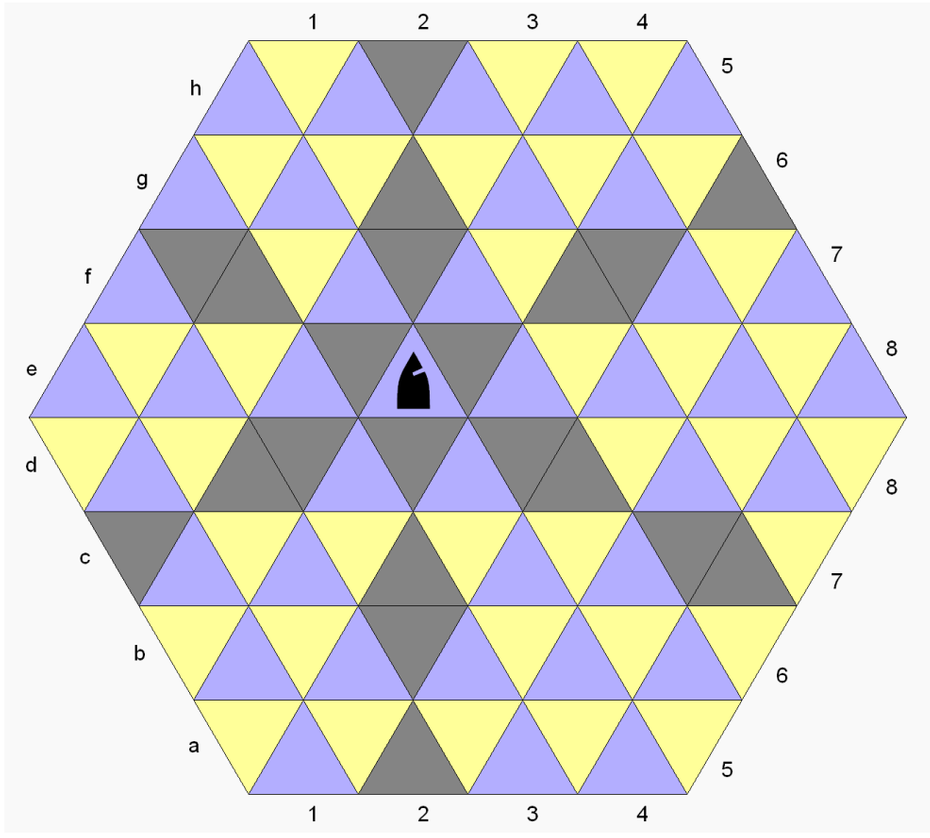
Triangular grid is a ~~bastard~~ problem

Steps within slides can alternate

1. Bishops in Triangular Chess slide diagonally alternating:
   - Diag, Orth, Diag, Orth, …

2. Bishops in Tri Chess slide orthogonally alternating:
   - Orth, Diag, Orth, Diag, …

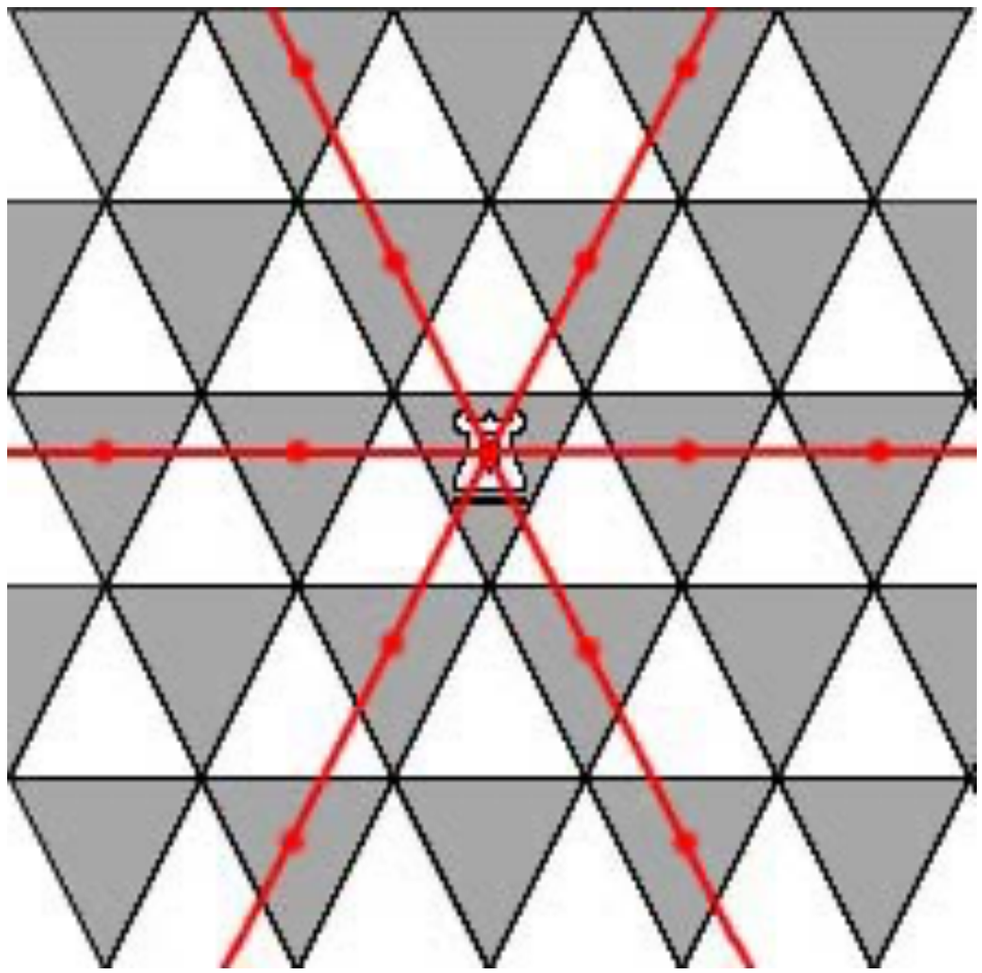




Triangular Radials

…while some slide regularly:

3. Rooks in Triangular Chess slide off-diagonally:
   - Off, Off, Off, Off, …



Triangular grid is usually the worst case scenario

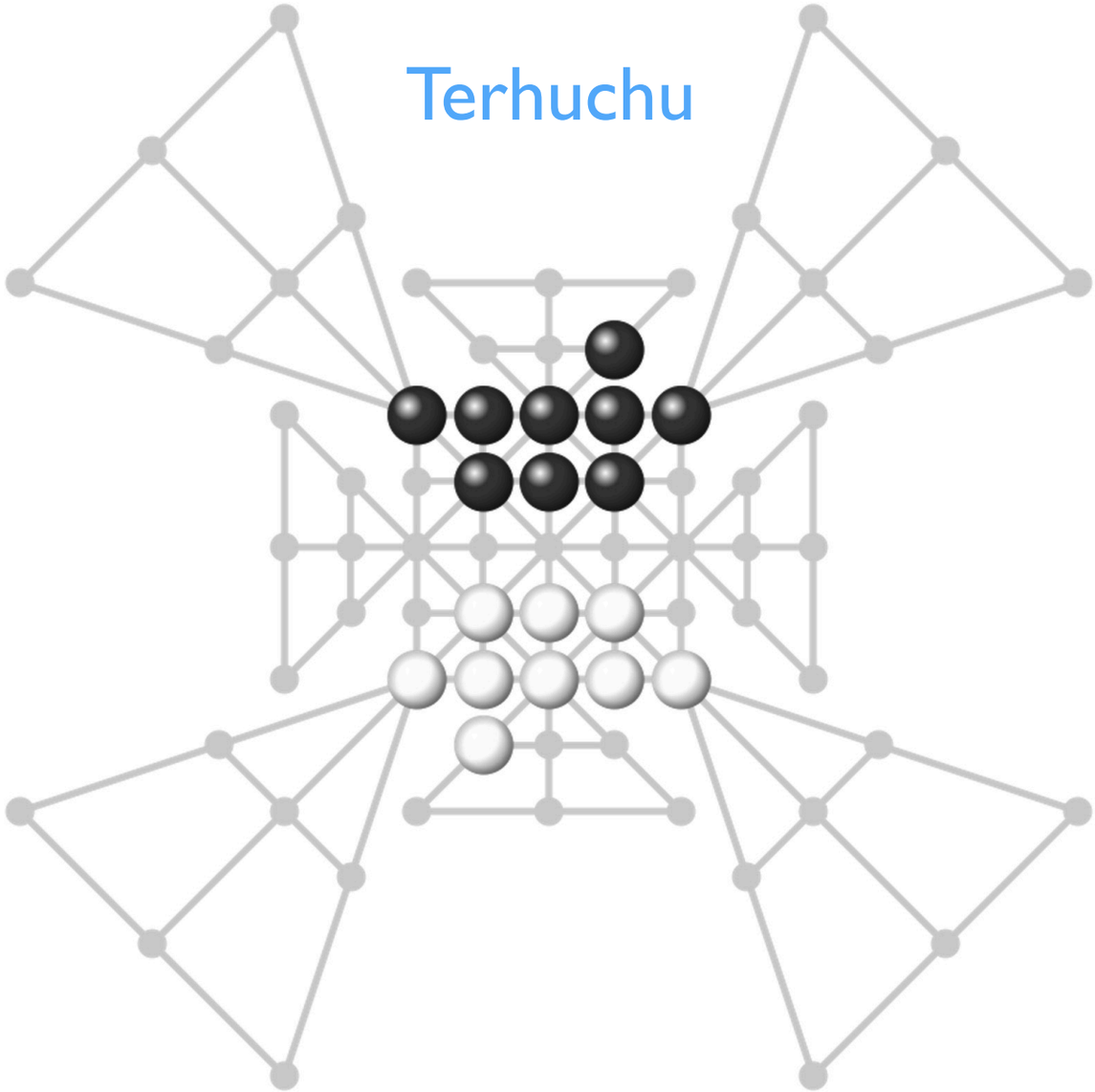
For any new design idea, first question to ask:

"Will it work on the triangular grid?"

Reality Check

Graph representation:
- Vertices
- Edges
- Cells

Relations:
- Directions
- Steps
- Walks
- Radials
- …

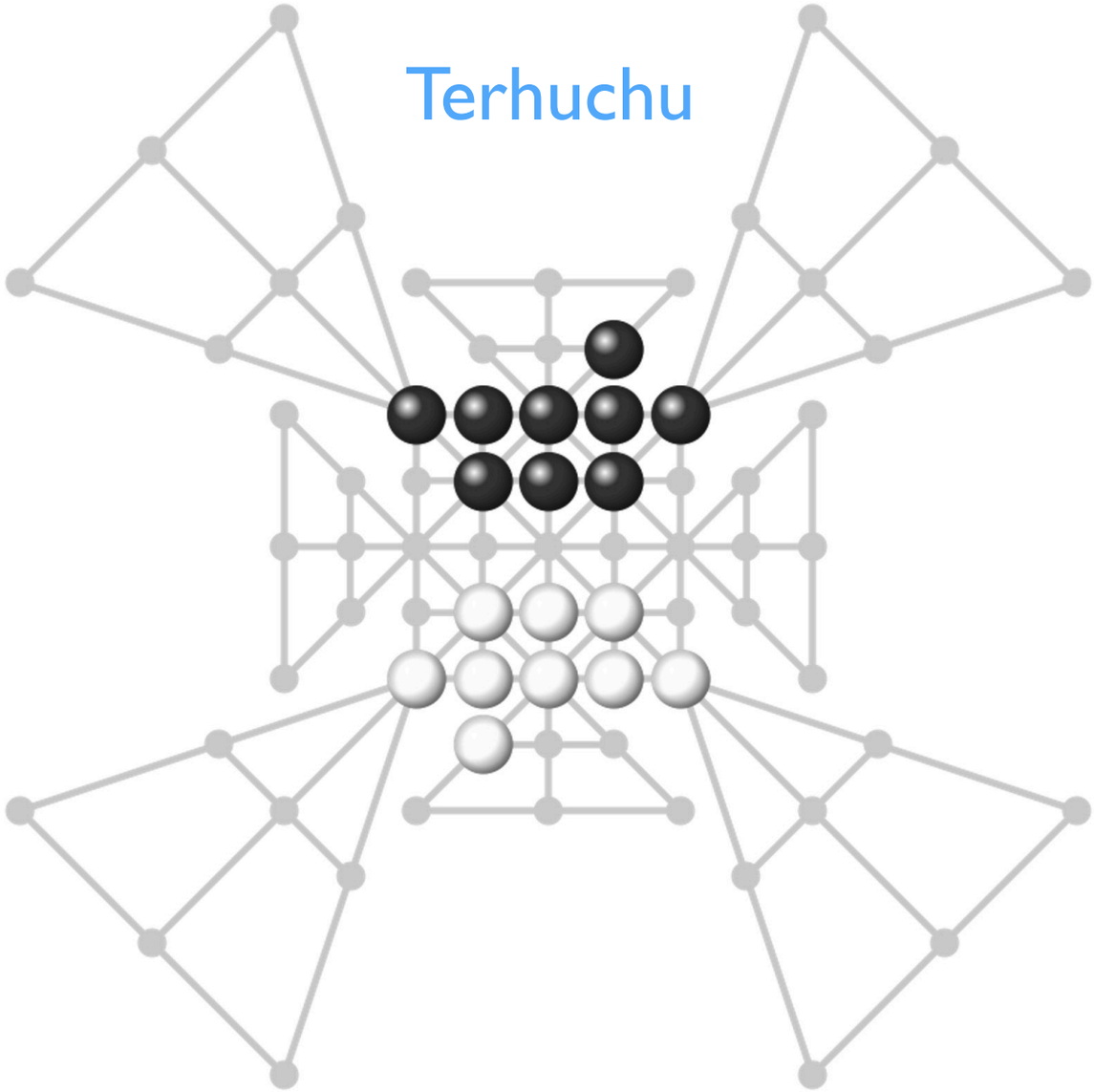


Reality Check

Graph representation:
- Vertices
- Edges
- Cells

Relations:
- Directions
- Steps
- Walks
- Radials
- …

BUT
- Required ~50 ludeme “board” classes
- Specialised for particular boards




```
(peralikatumaBoard
   5
            top:true
            left:true
            right:true
            bottom:true
            bottomLeft:true
            bottomRight:true
            topLeft:true
            topRight:true
)
```

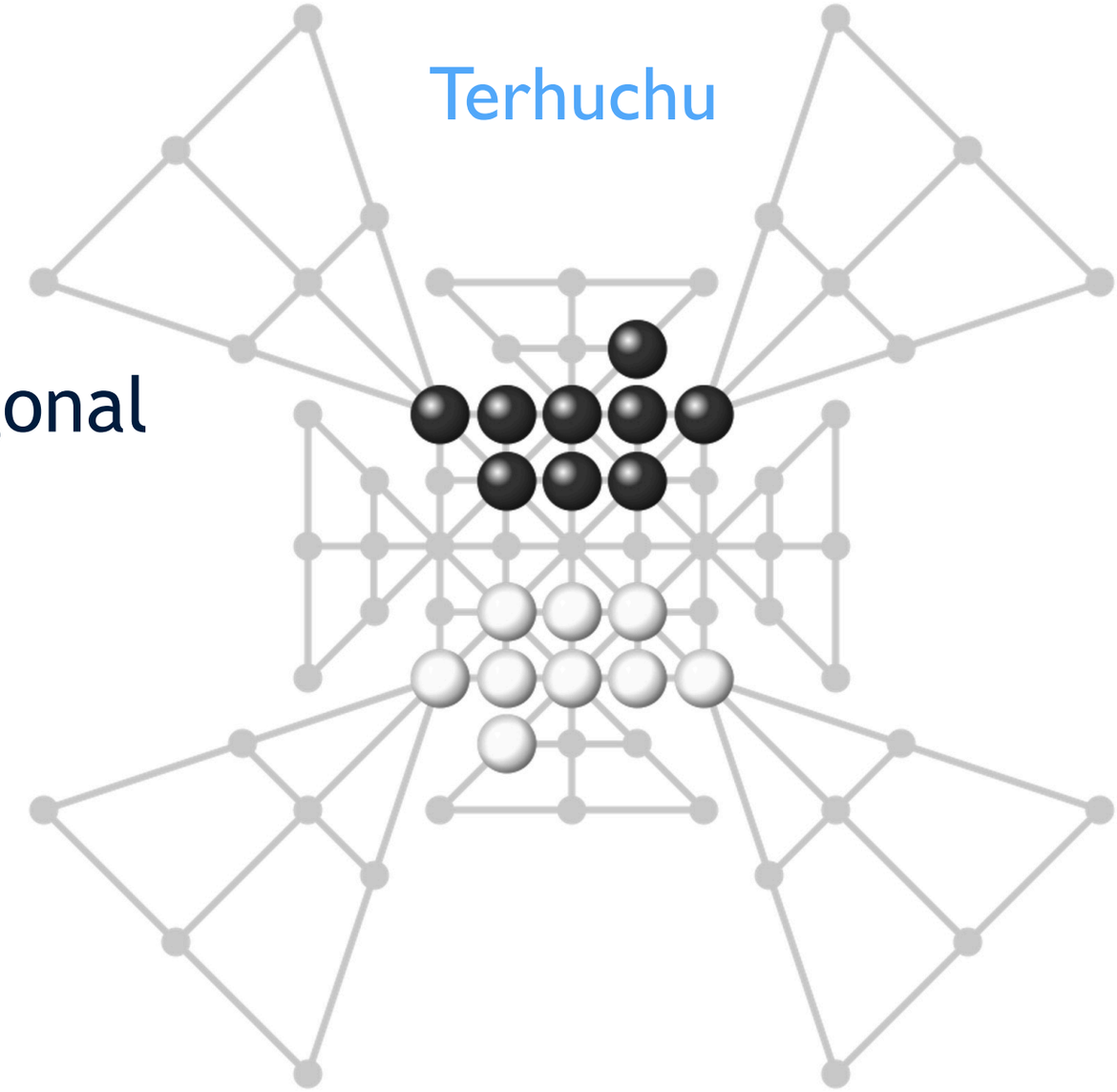
Overspecialisation

Any change:

- Different base tiling? e.g. hexagonal
- Different arm shape?
- Additional sites?
- Truncated corners?
- Different connectivity?

All would require ludemes
to be updated or added

Violates principle of **generality**




```
(peralikatumaBoard
   5
            top:true
            left:true
            right:true
            bottom:true
            bottomLeft:true
            bottomRight:true
            topLeft:true
            topRight:true
)
```

Graph Functions

Solution: Graph functions!

1. **Generators**
   Define simple graphs as **tilings** in certain **shapes**

2. **Operators**
   Combine graphs in simple ways

Can define boards of arbitrary complexity
- More flexible, more general
- More in keeping with the “ludemic” approach

Implemented in June 2020
- Just before public Ludii v1.0.0 release

Graph Generators: Tilings

Regular
- Square
- Hexagonal
- Triangular

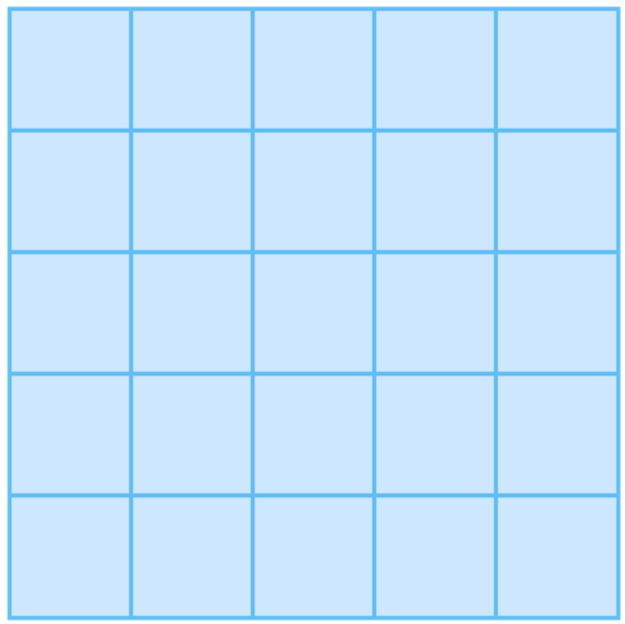
Semi-Regular
- 4.8.8
- 4.6.12
- 3.4.6.4
- 3.6.3.6
- 3.12.12
- 3.3.3.3.6
- 3.3.3.4.4
- 3.3.4.3.4

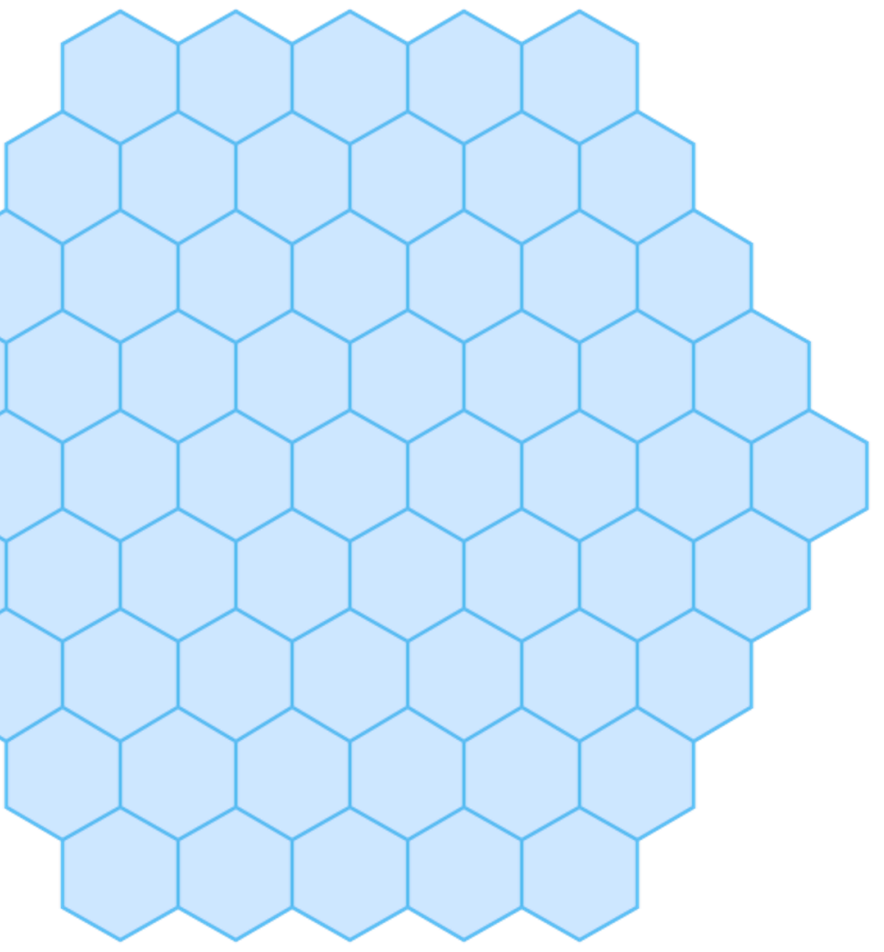
Custom
- Brick
- Celtic
- Morris
- Quadhex

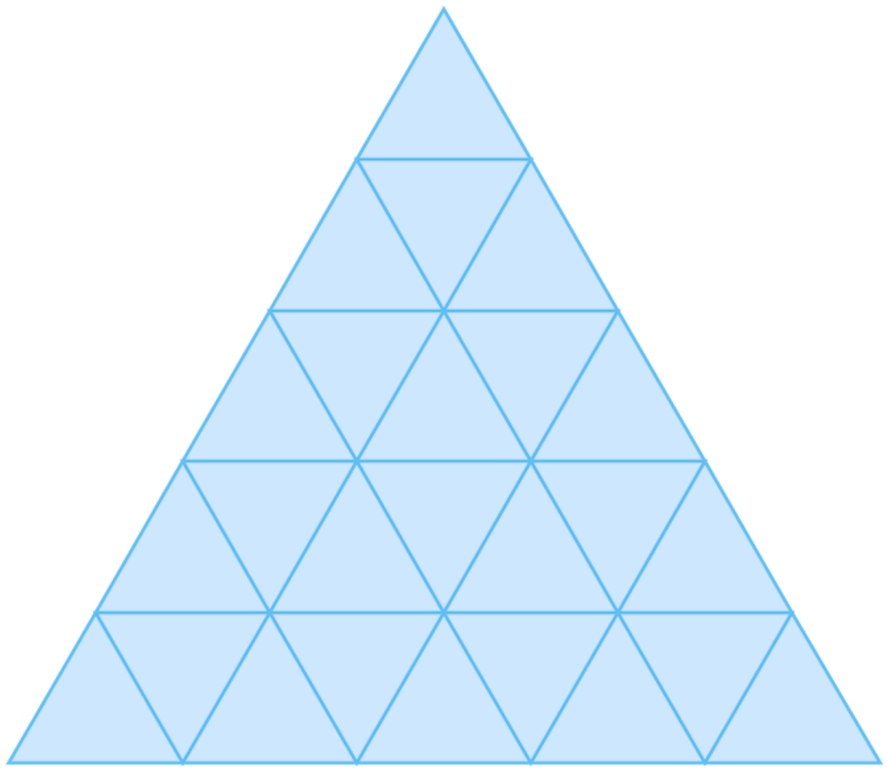
Regular Tilings

(square 5)
(hex 5)
(tri 5)

Number of cells per side
- Add 1 if “use:Vertex”

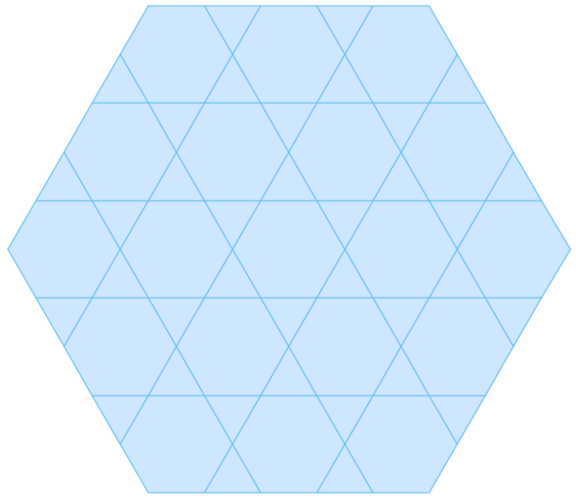


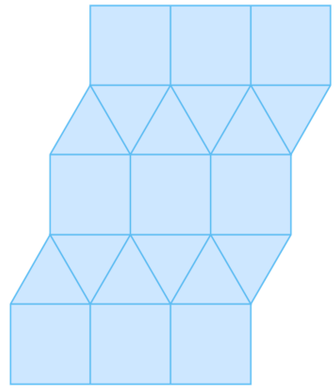


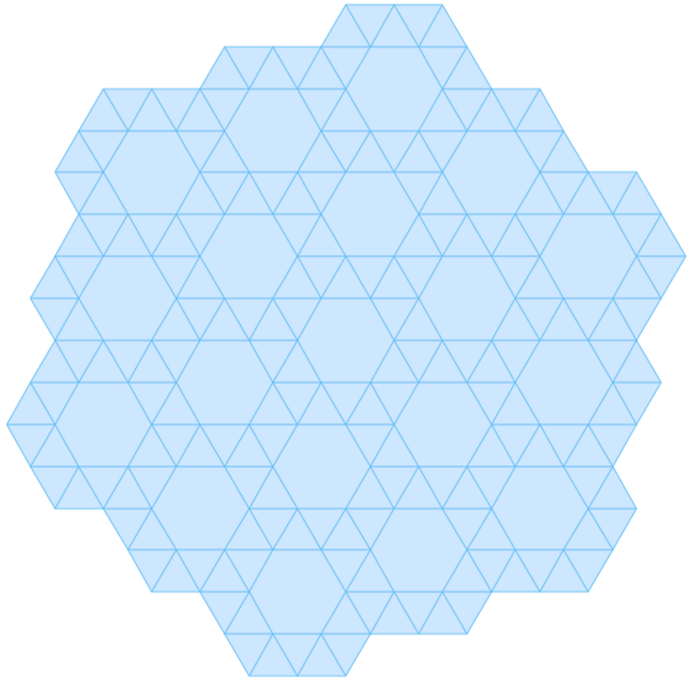


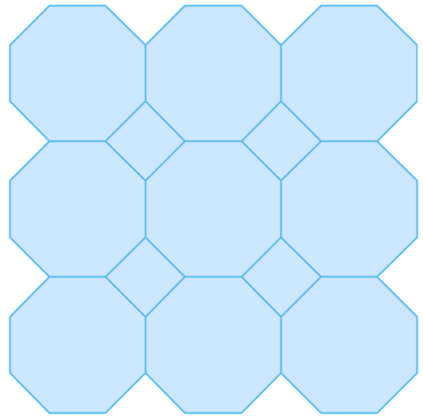
Semi-Regular Tilings

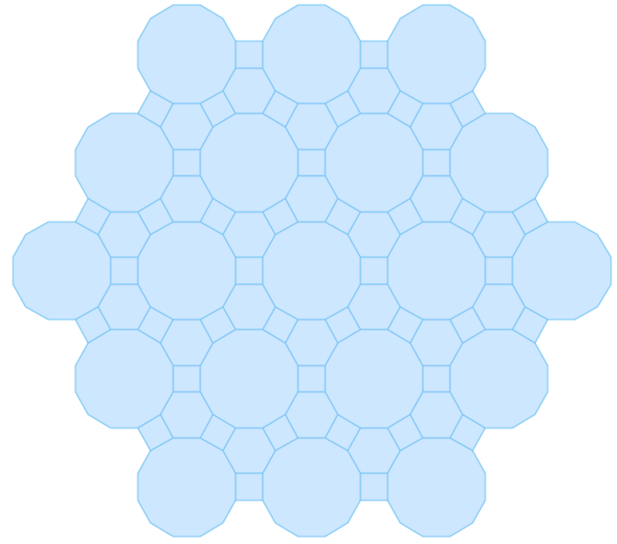
(tiling T488 3)
(tiling T4612 3)
(tiling T3464 3)
(tiling T3636 3)
(tiling T31212 3)
(tiling T33336 3)
(tiling T33344 3)
(tiling T33434 3)

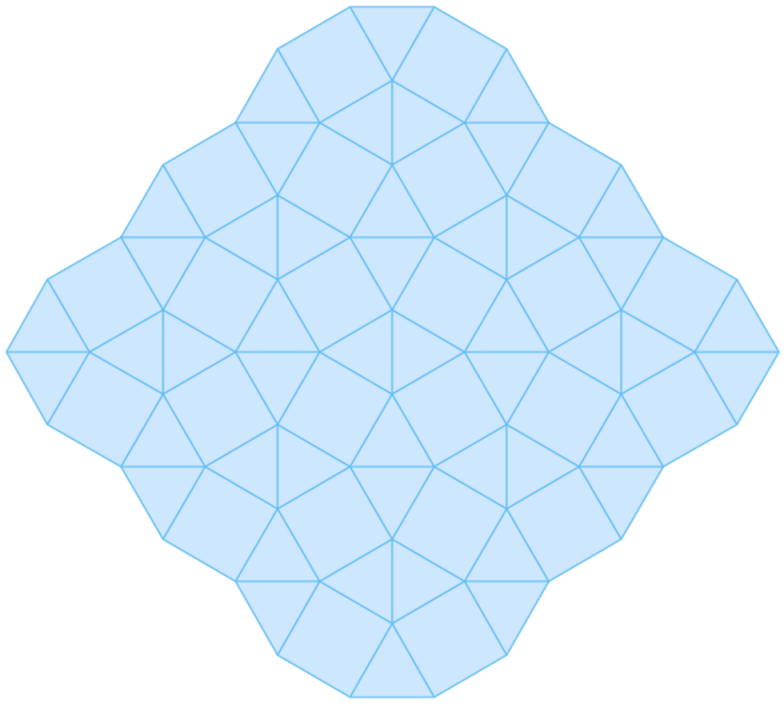


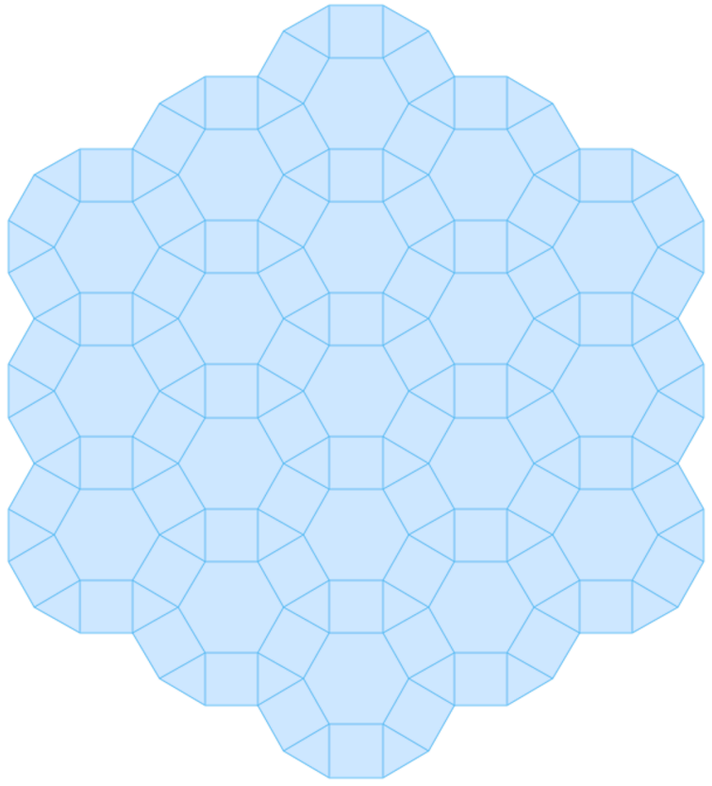


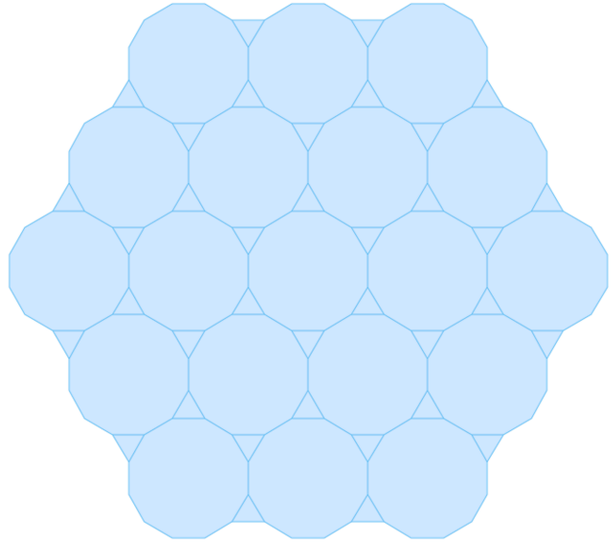












Semi-Regular Generation

(tiling T3464 2)

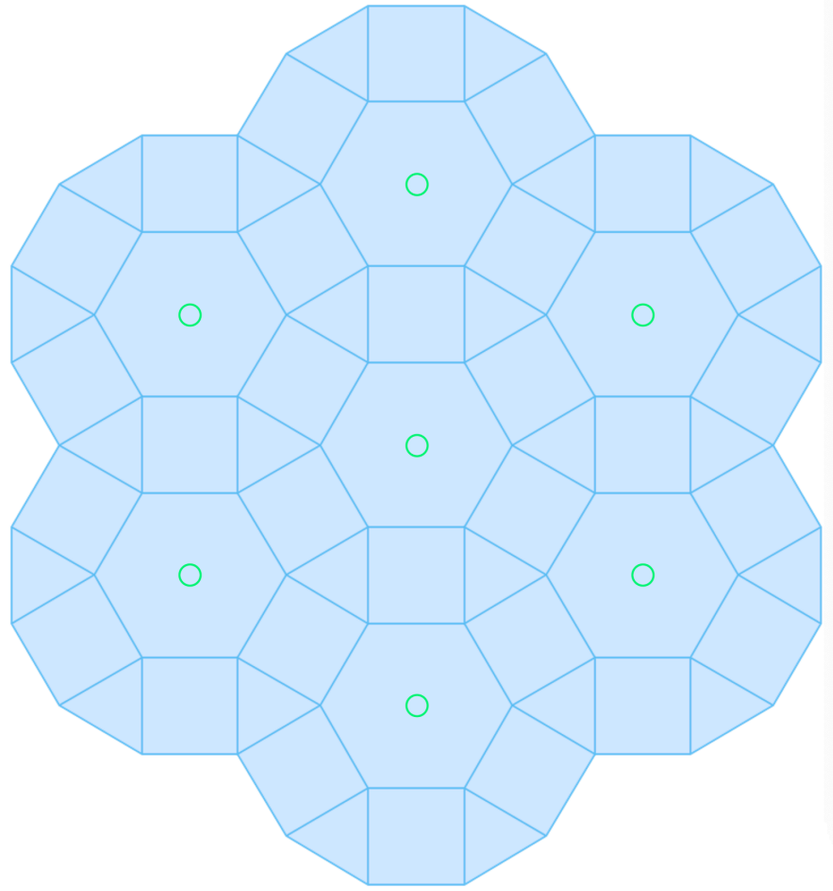
Generation based on:

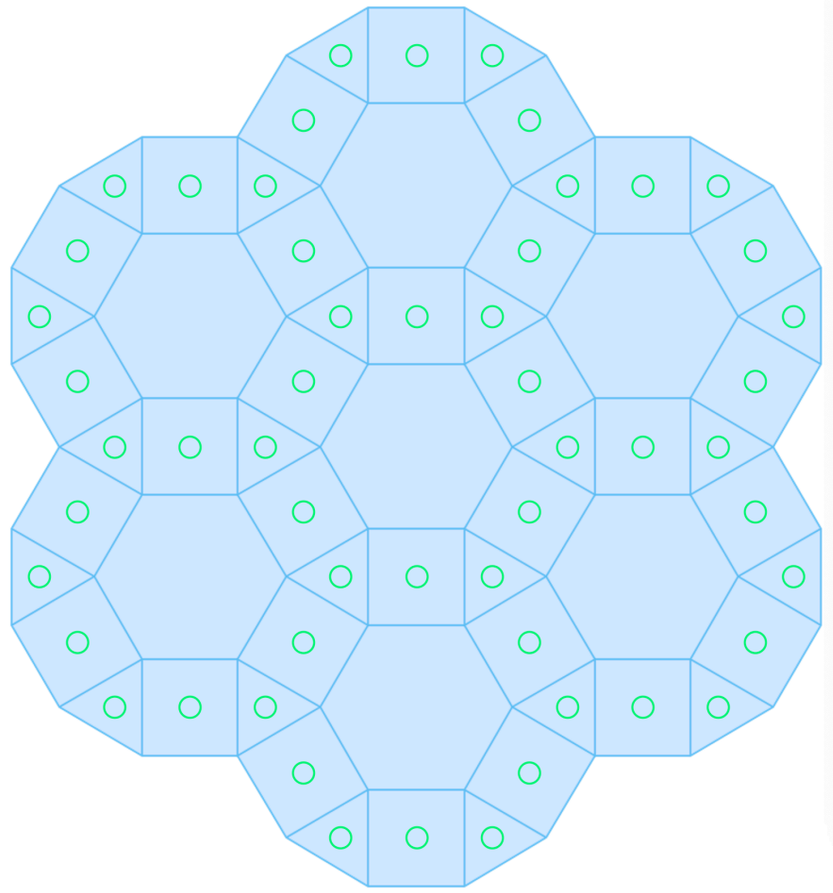
- ***Major*** Reference elements
  e.g. hexagons on hex grid

- ***Minor*** Satellite elements
  e.g. squares and triangles

Size refers to major elements:

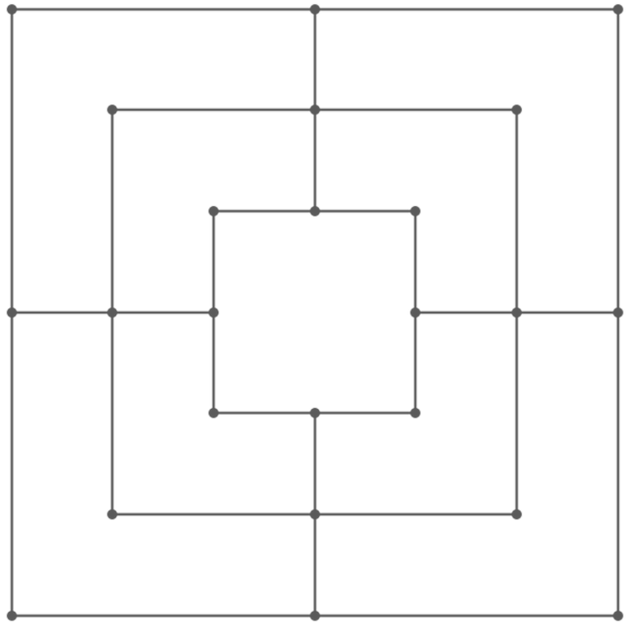
"3.4.6.4 board size 2"

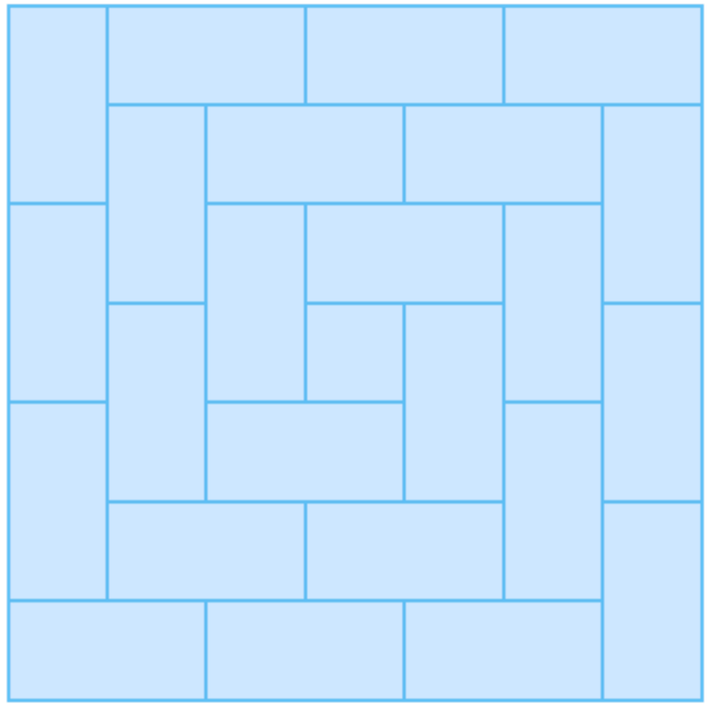


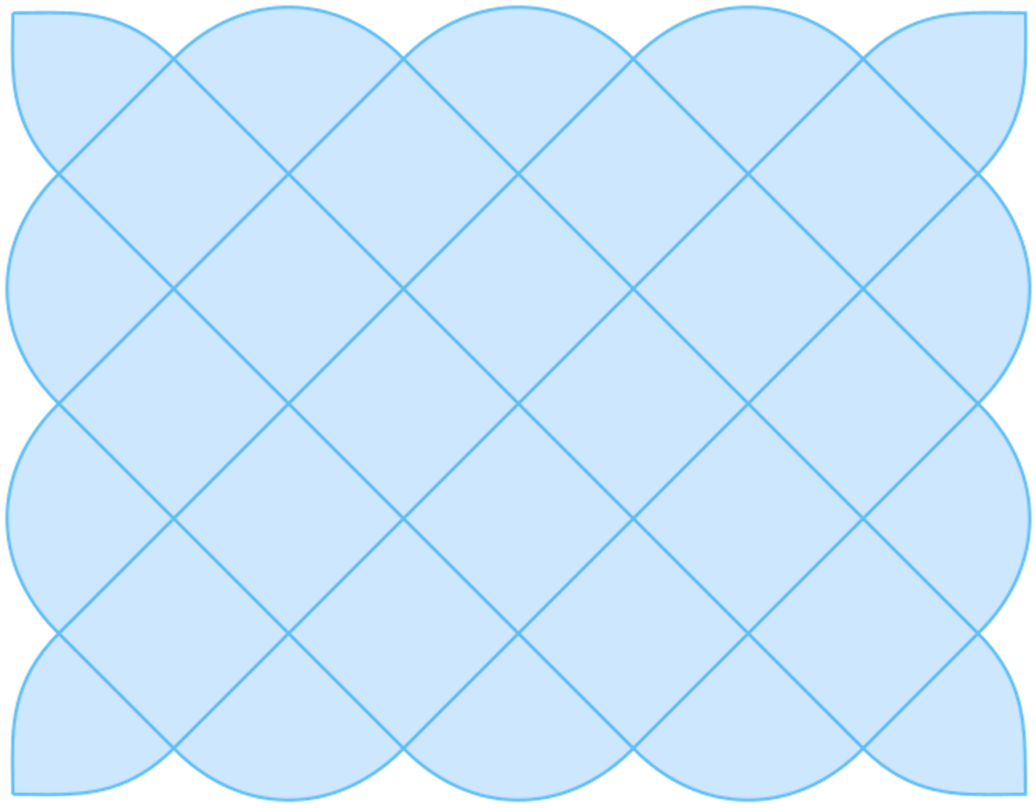


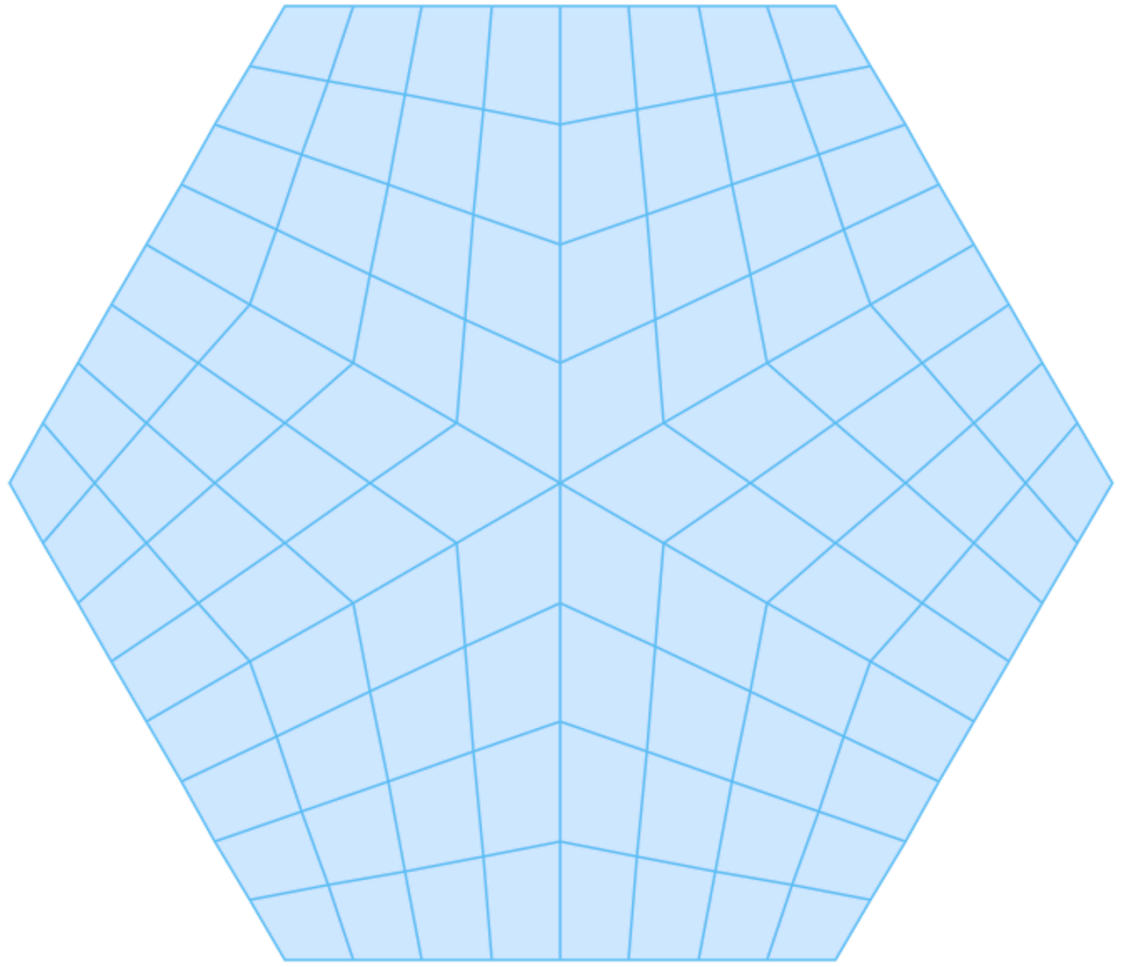
Custom Tilings

(morris 3)
(brick 4)
(quadhex 4)
(celtic 3 4)



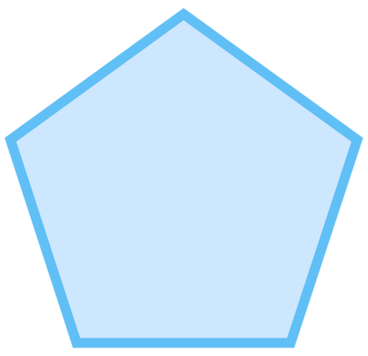


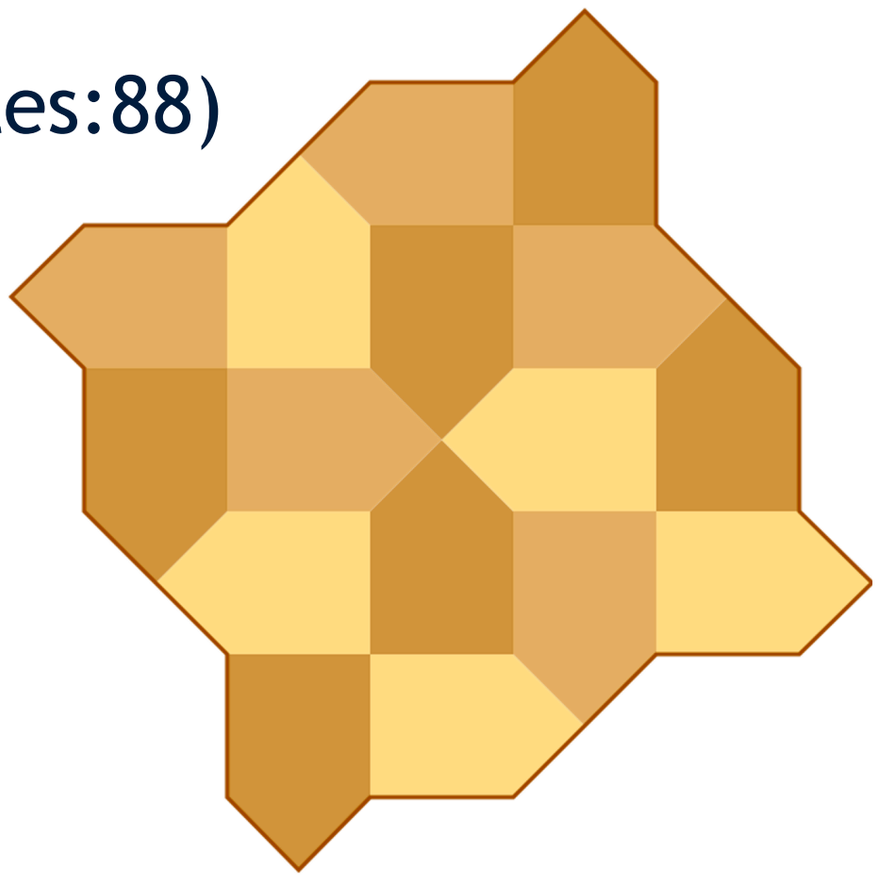


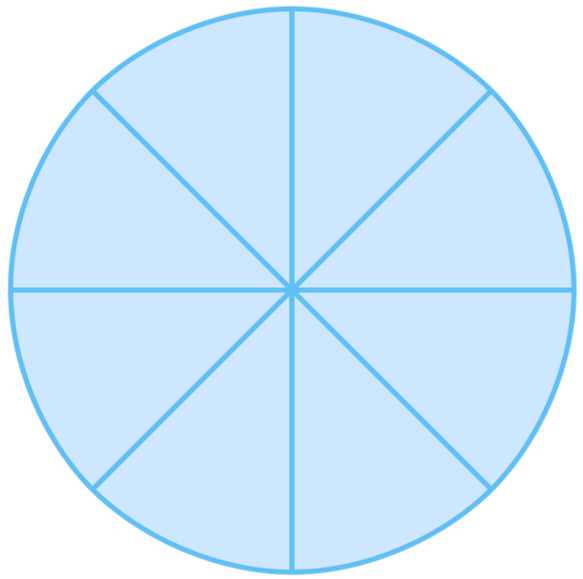


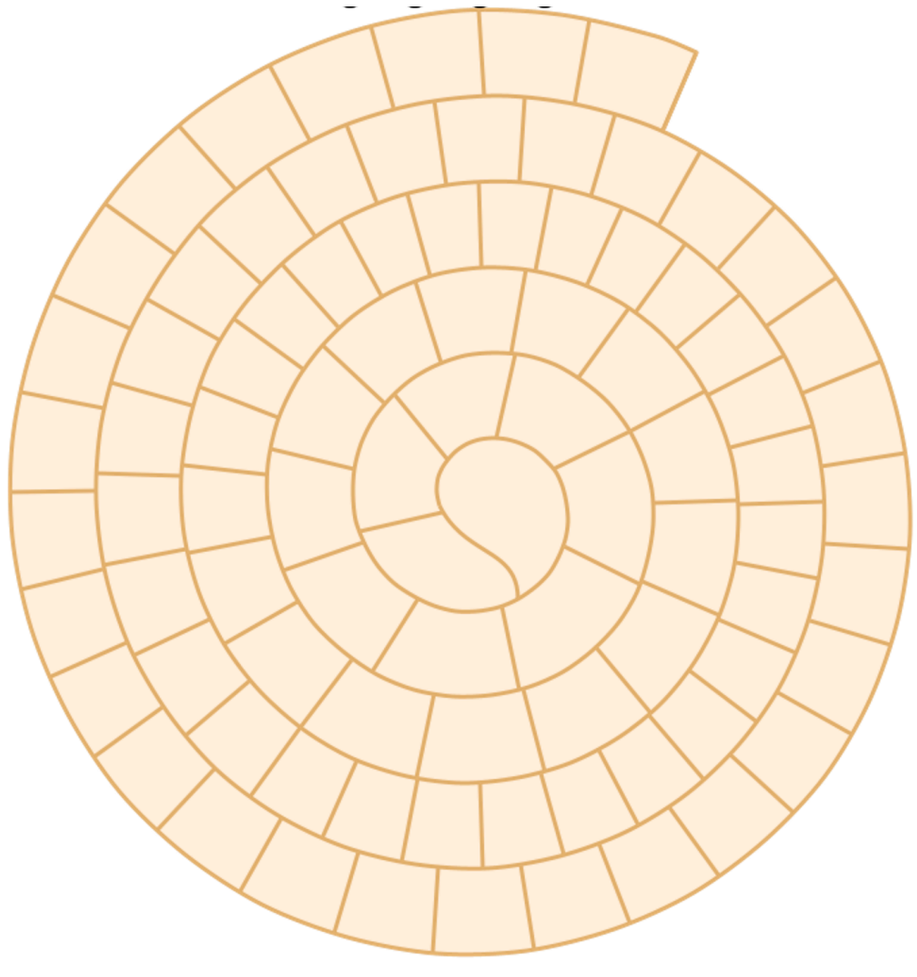
Graph Generators: Shapes

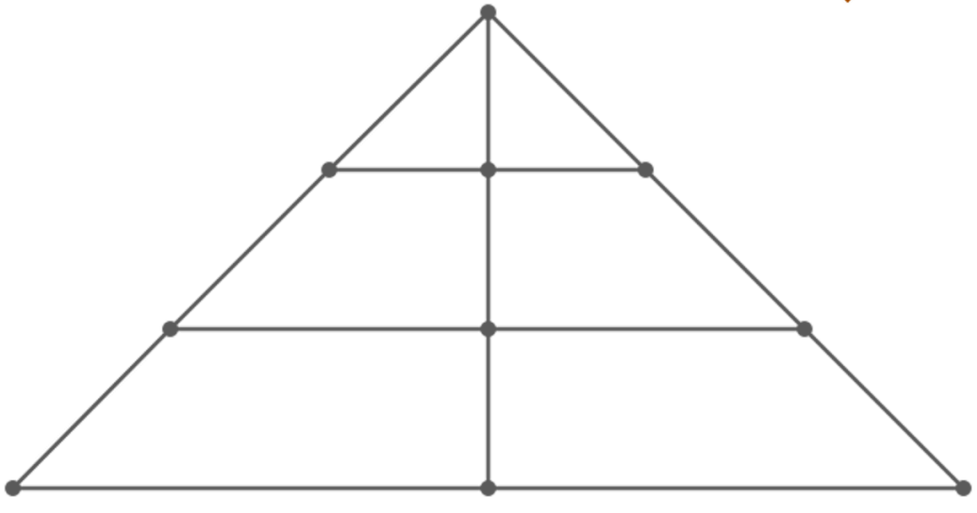
(shape 5)
(wedge 4)
(spiral turns:5 sites:88)
(circle 8)
(repeat 2 2 ...)





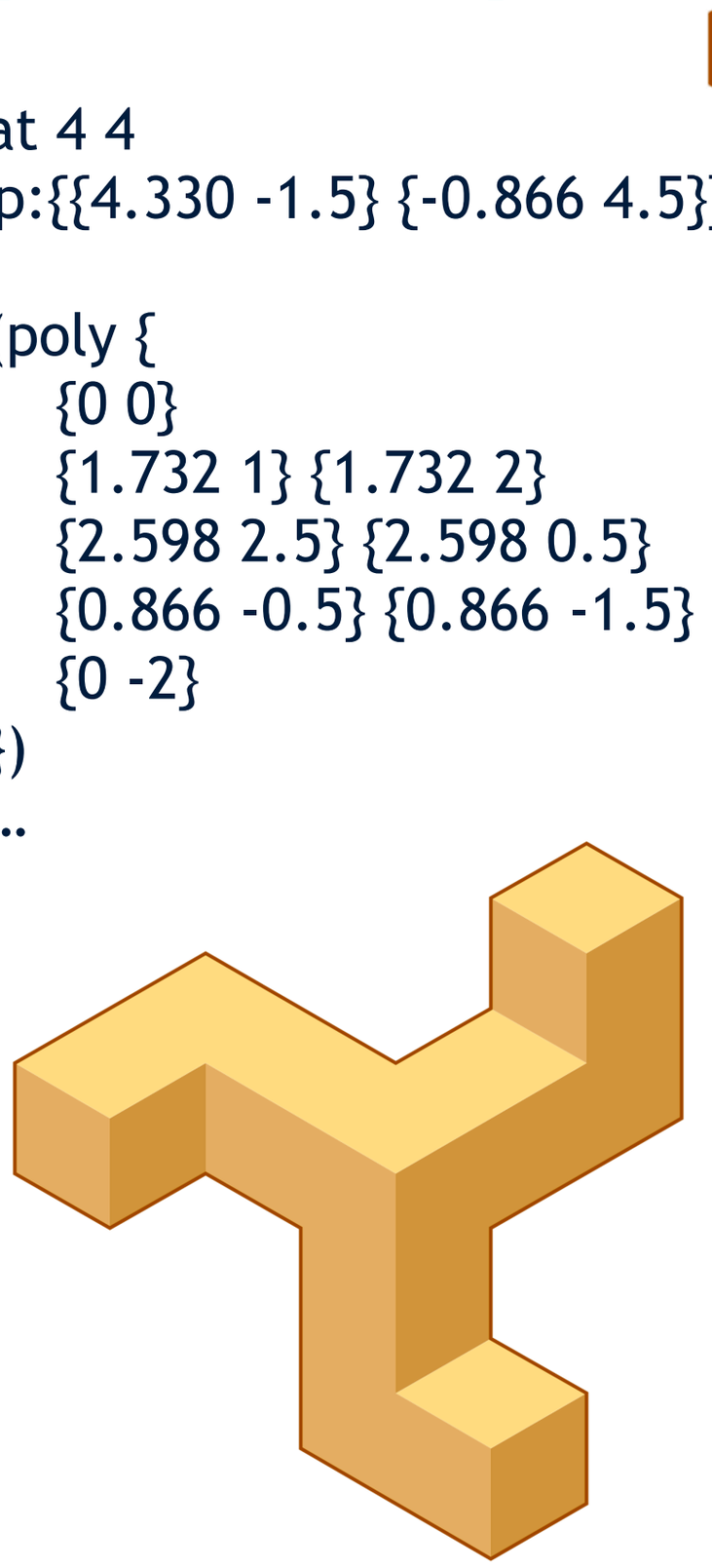


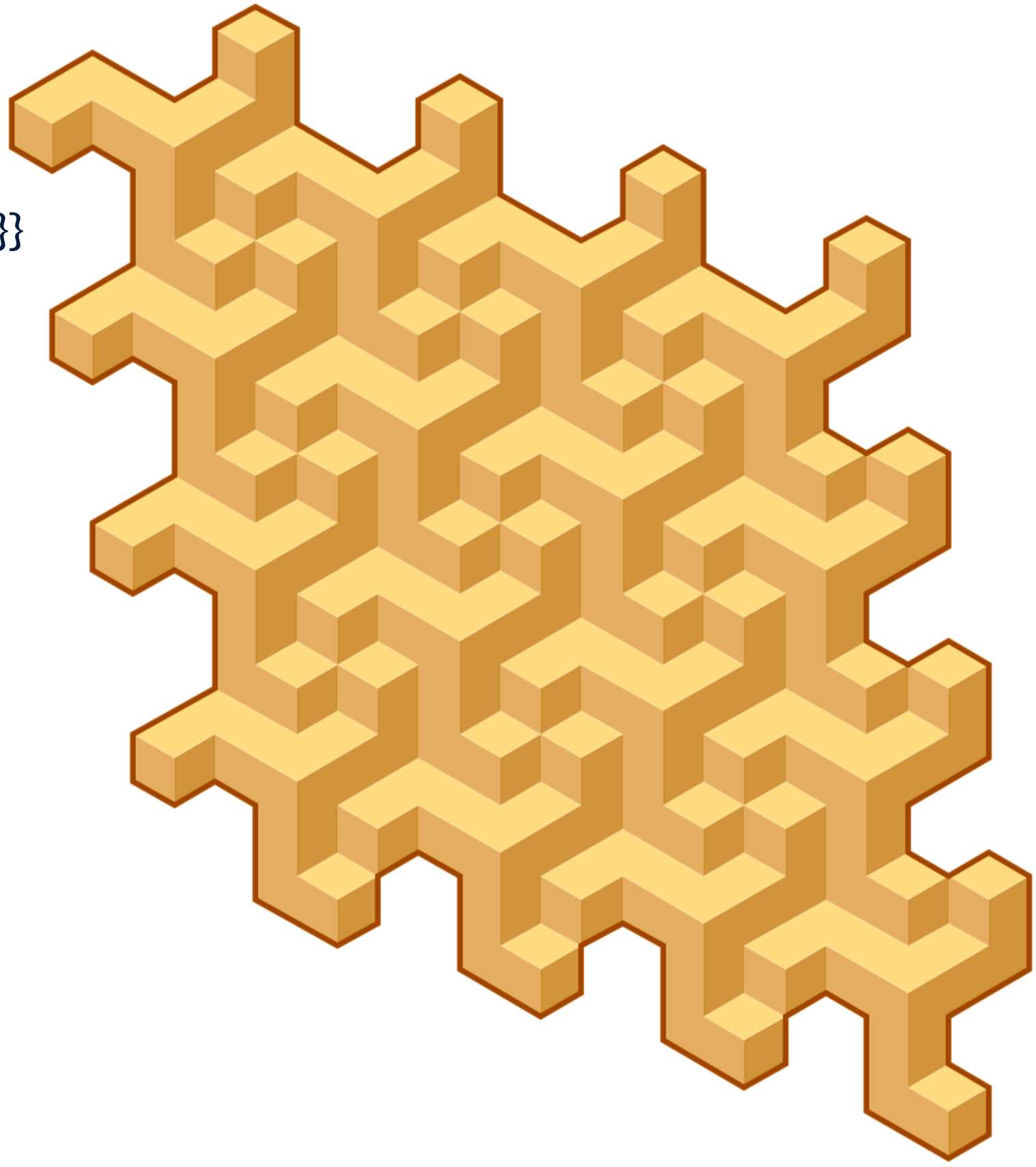




Repeat Shape

```
(repeat 4 4
    step:{{4.330 -1.5} {-0.866 4.5}}
    {
        (poly {
            {0 0}
            {1.732 1} {1.732 2}
            {2.598 2.5} {2.598 0.5}
            {0.866 -0.5} {0.866 -1.5}
            {0 -2}
        })
        ...
    }
)
```





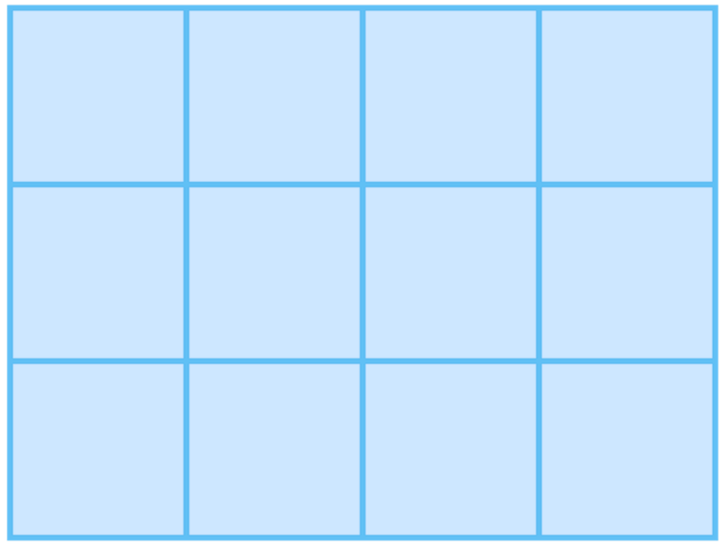
Limping Boards

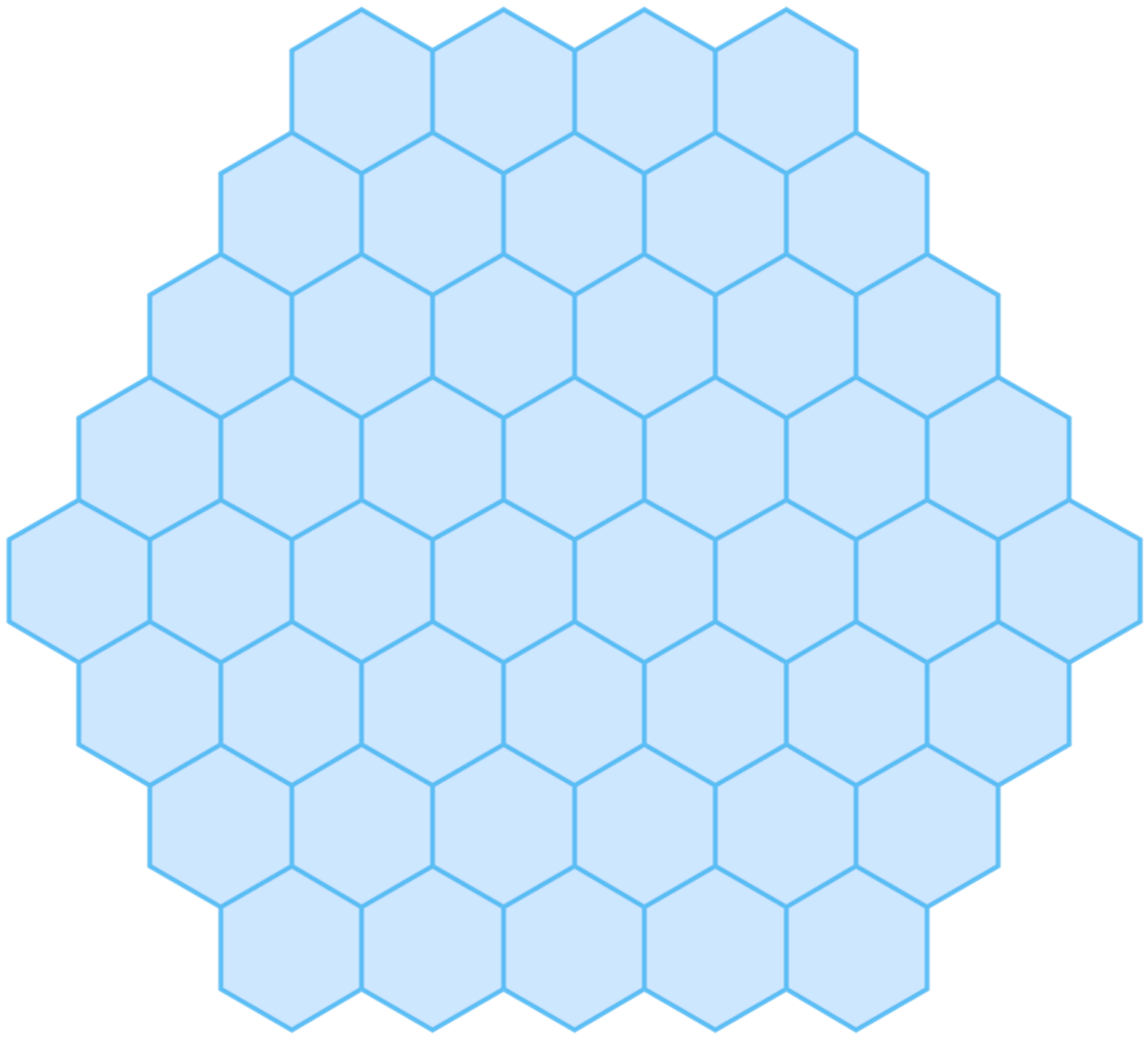
“Limping” board:
- Shape modifier
- Sides of {*n*, *n*+1, *n*, *n*+1, …}

e.g.

(square Limping 3)
(hex Limping 4)

Useful for projective boards
e.g. Projex

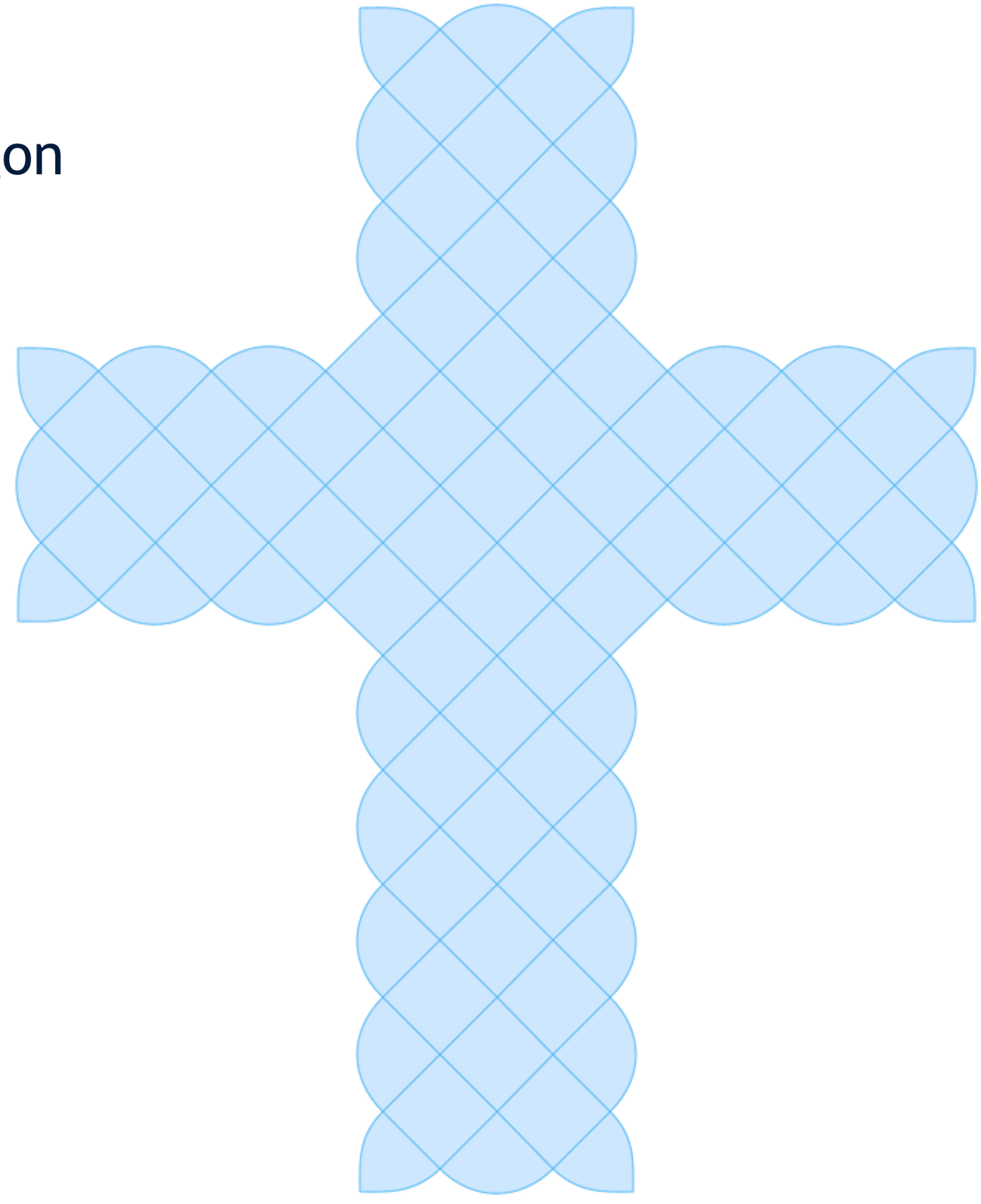




Custom Shapes

Celtic tiling clipped by polygon

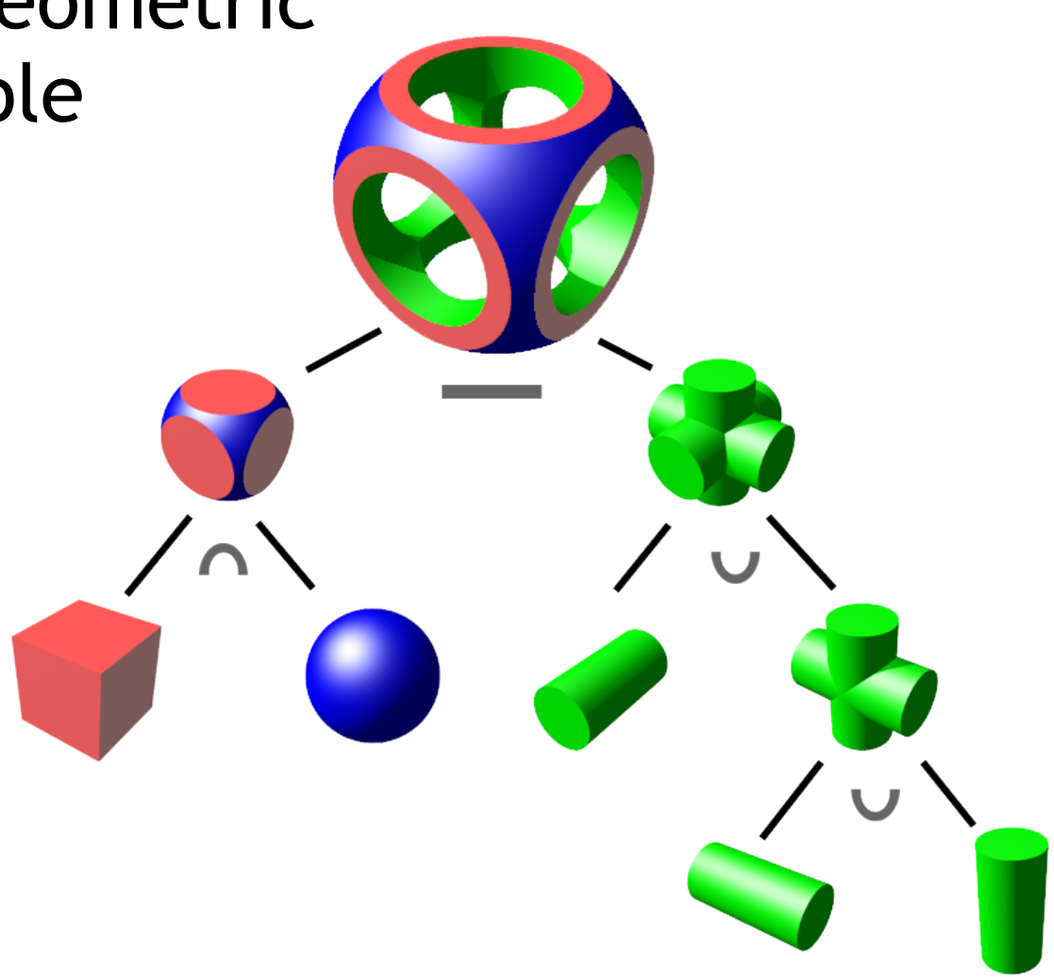
```
(celtic
   (poly {
      {3 0} {3 6} {0 6} {0 9}
      {3 9} {3 13} {6 13} {6 9}
      {10 9} {10 6} {6 6} {6 0}
   })
)
```

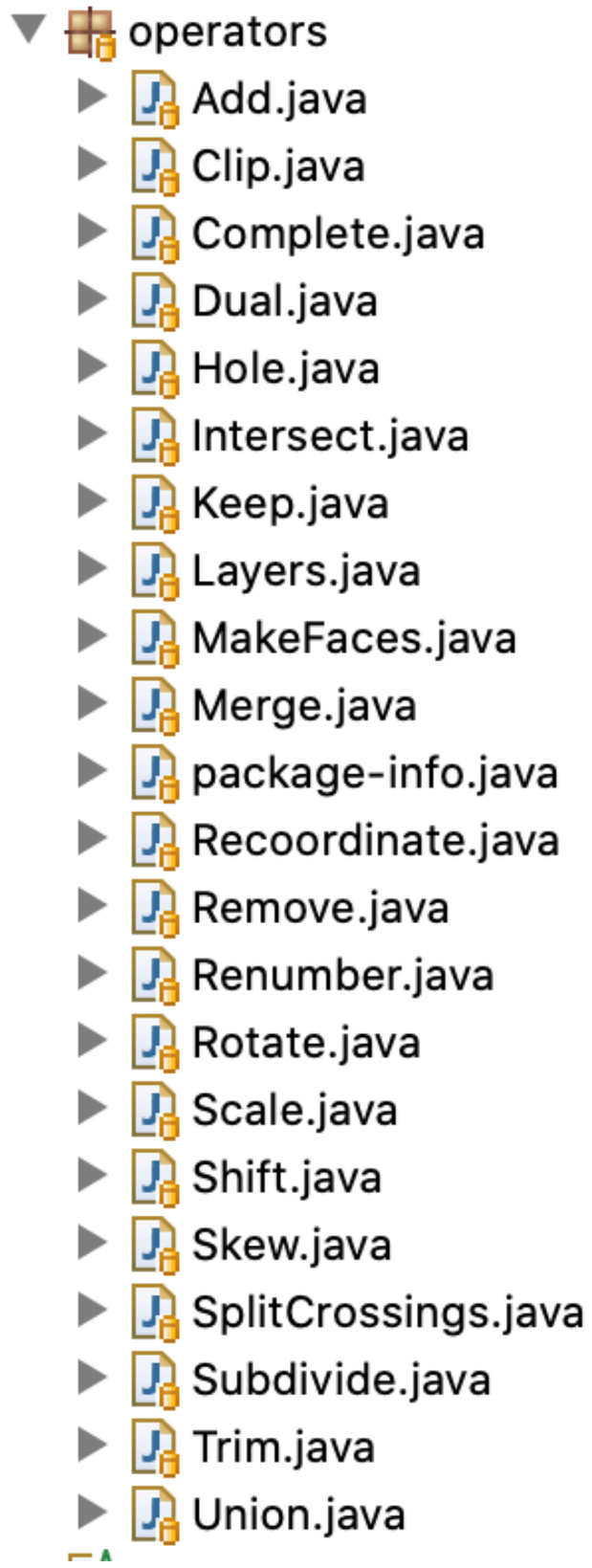


Graph Operators

Operators
- Take one or more graphs
- Return a modified or combined graph

Based on **Constructive Solid Geometry (CSG)**
- Build complex geometric forms from simple primitives and operations

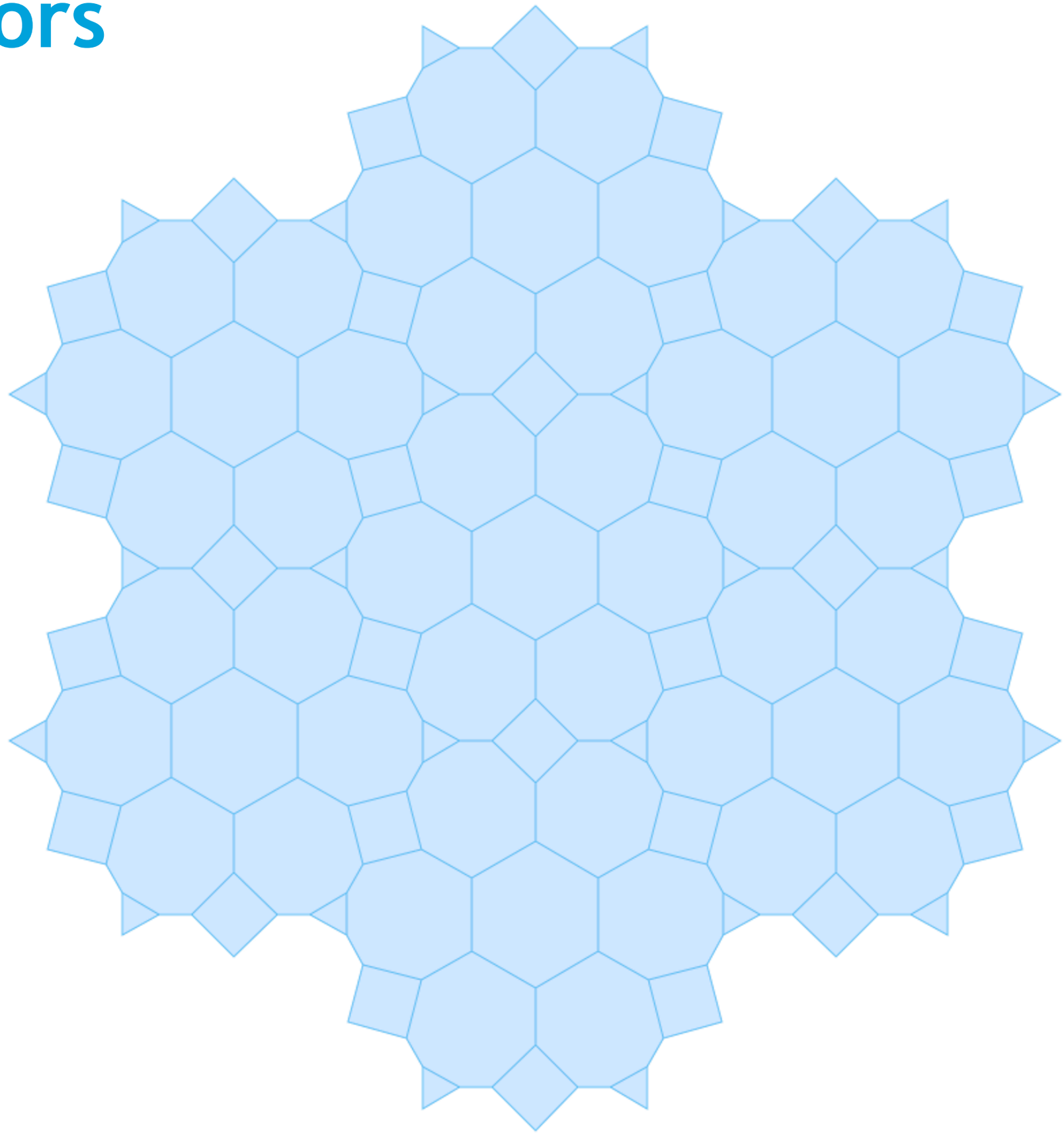




Graph Operators

Example

```
(dual
    (subdivide
        (tiling T3464 2)
    )
)
```



Weird but interesting!

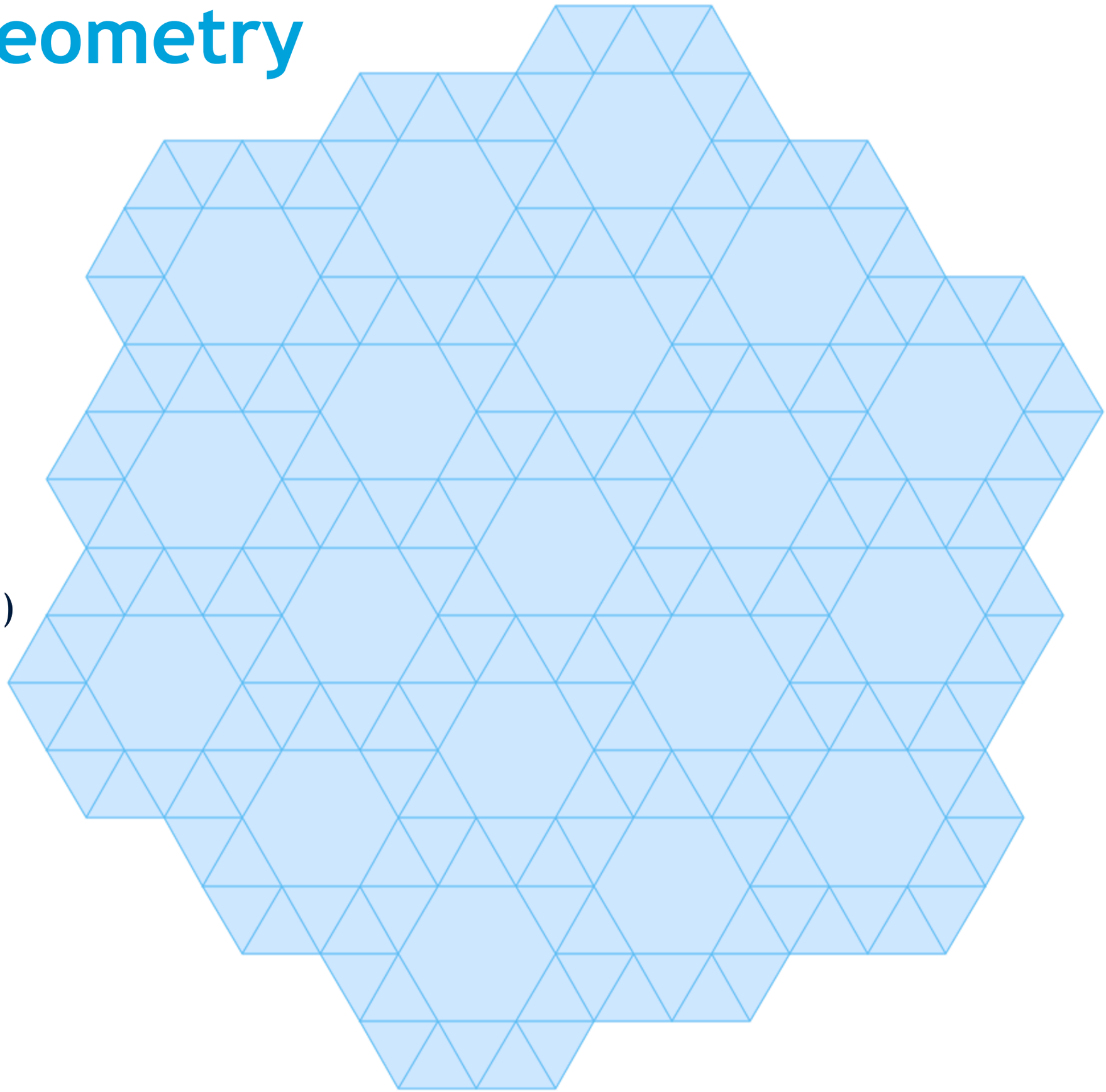
Few seconds to define

Graph Operators

Add	Add elements
Clip	Clip off elements outside a shape
Complete	Add edges between all vertex pairs
Dual	Take the weak dual of the graph
Hole	Remove some interior cells
Intersect	Keep elements shared by two or more graphs
Keep	Discard everything outside the specified region
Layers	Make Z layers for 3D boards
MakeFaces	Create all simple faces
Merge	Combine graphs and merge coincident elements
Recoordinate	Regenerate coordinate labels
Remove	Remove specified elements
Renumber	Renumber elements from bottom left
Rotate	Rotate elements
Scale	Scale elements
Shift	Shift elements
Skew	Skew elements
SplitCrossings	Create vertices wherever edges cross
Subdivide	Insert vertices and edges to subdivide cells
Trim	Remove orphaned edges and cells
Union	Combine graphs without merging coincident elements

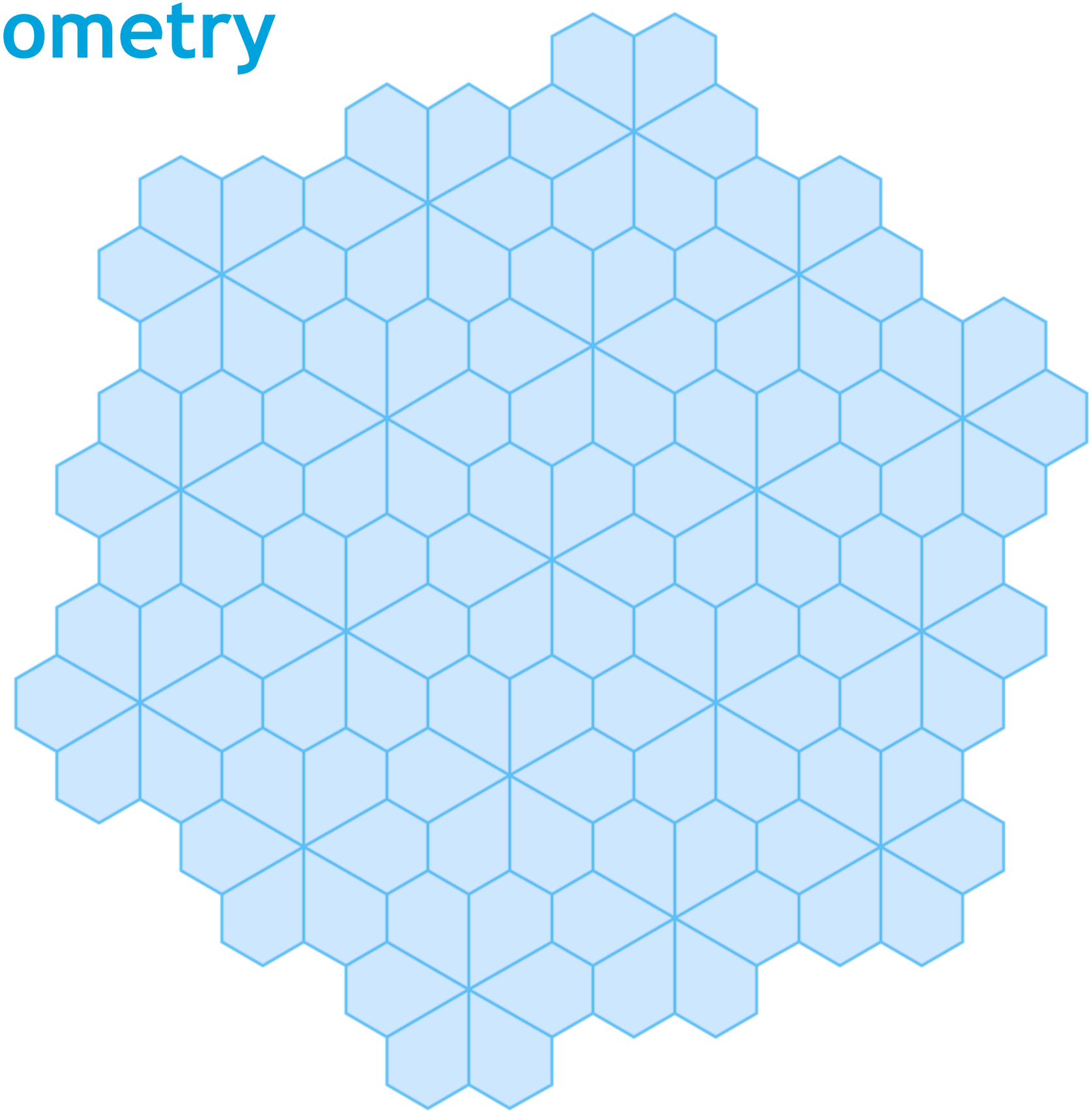
Using the Geometry

```
(game "X"
   (players 2)
   (equipment {
      (board
          tiling T33336 3))
      (piece “Disc") })
   (rules
      (play (move Add (to
              (sites Empty))))
      (end (if (is Line 4)
         (result Mover Win)))
   )
)
```



Using the Geometry

```
(game "X"
   (players 2)
   (equipment {
      (board (dual
         tiling T33336 3))
      (piece “Disc") })
   (rules
      (play (move Add (to
            (sites Empty))))
      (end (if (is Line 4)
         (result Mover Win)))
   )
)
```

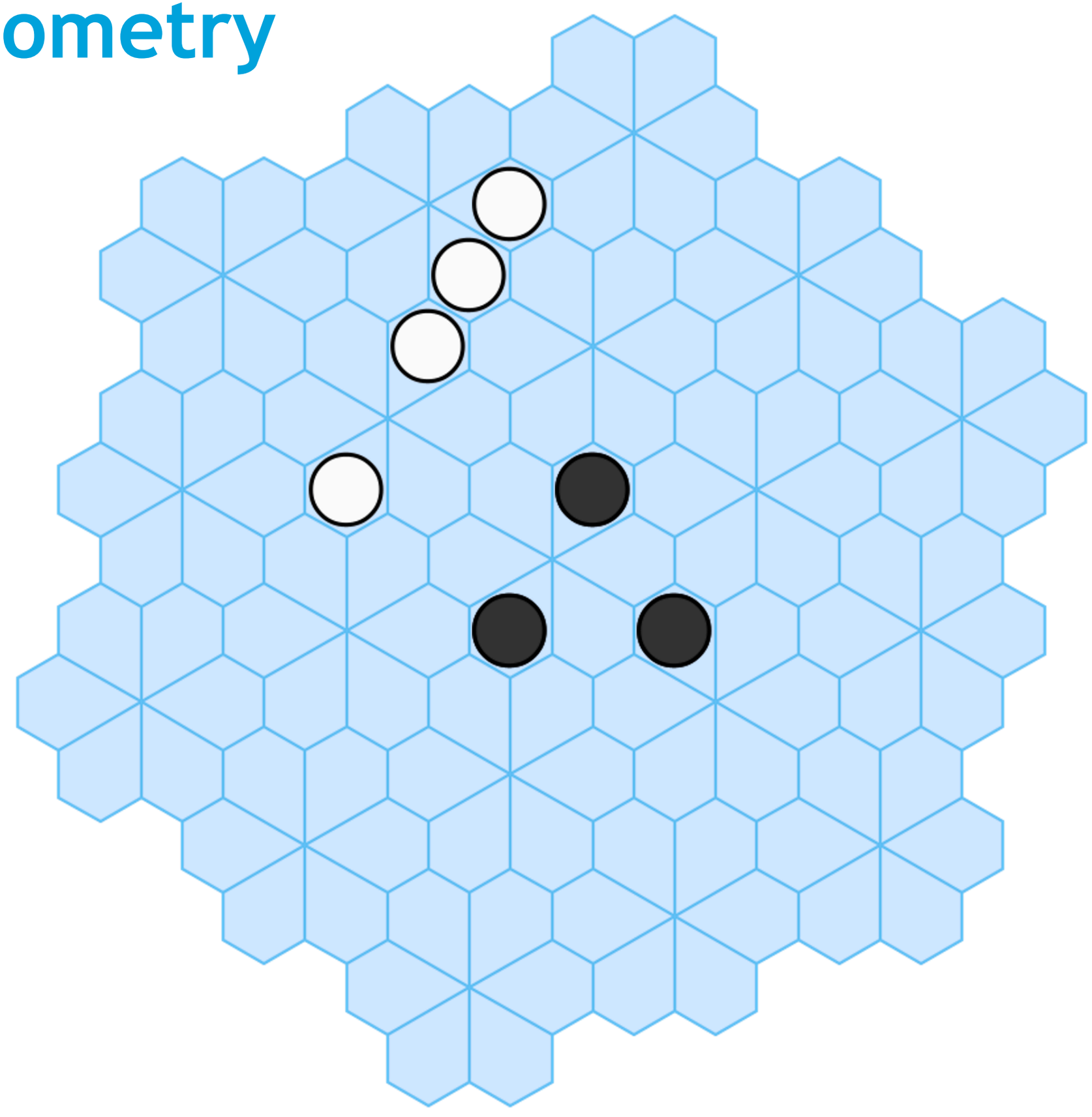


All cells equal size

Rosette patterns

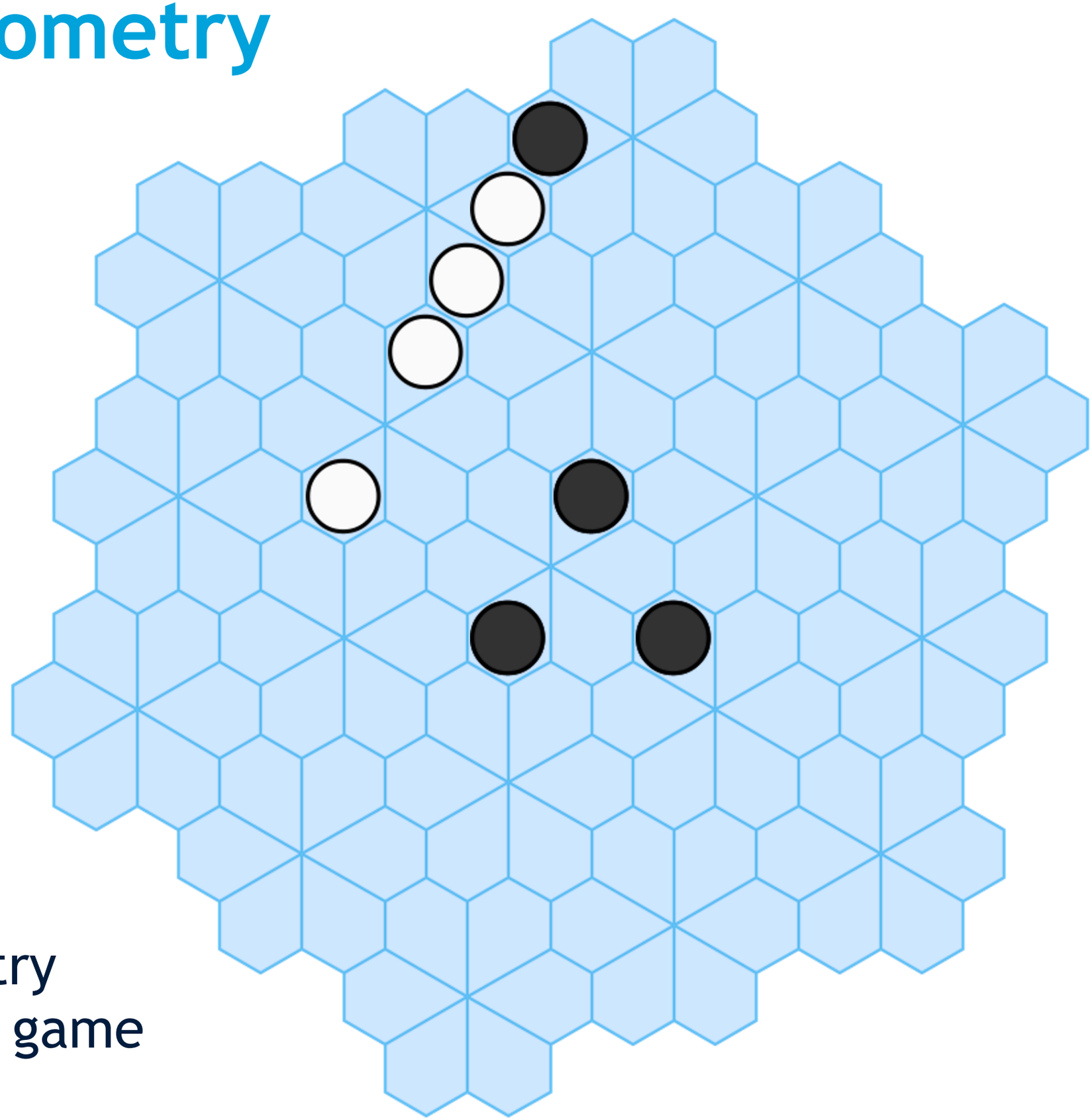
Using the Geometry

```
(game "X"
   (players 2)
   (equipment {
      (board (dual
          tiling T33336 3)))
      (piece “Disc") })
   (rules
      (play (move Add (to
             (sites Empty))))
      (end (if (is Line 4)
         (result Mover Win)))
   )
)
```



Using the Geometry

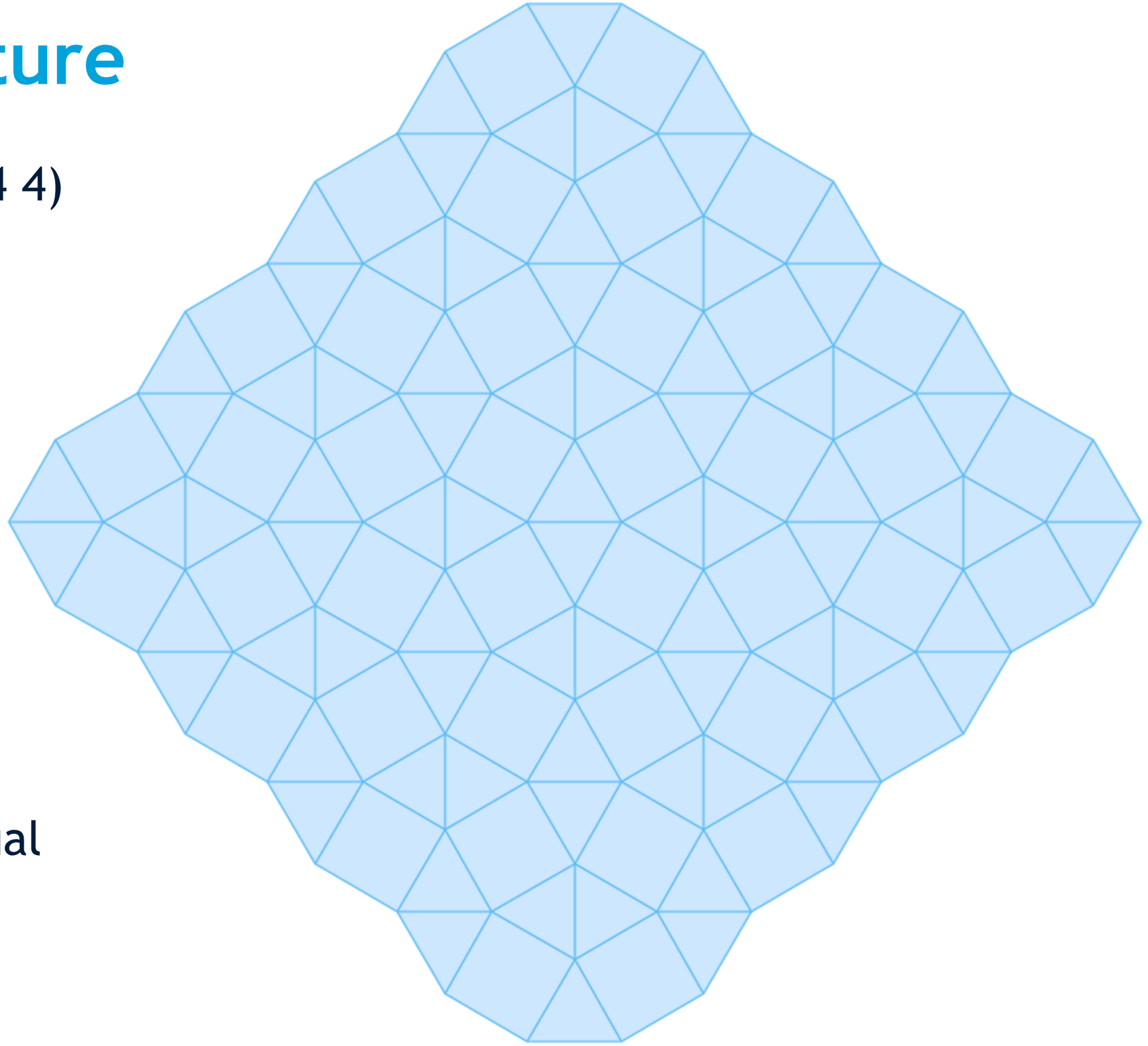
```
(game "X"
   (players 2)
   (equipment {
      (board (dual
          tiling T33336 3)))
      (piece “Disc") })
   (rules
      (play (move Add (to
             (sites Empty))))
      (end (if (is Line 4
             Orthogonal)
         (result Mover Win)))
   )
)
```



Incorporates geometry
into the rules of the game

Dual Nature

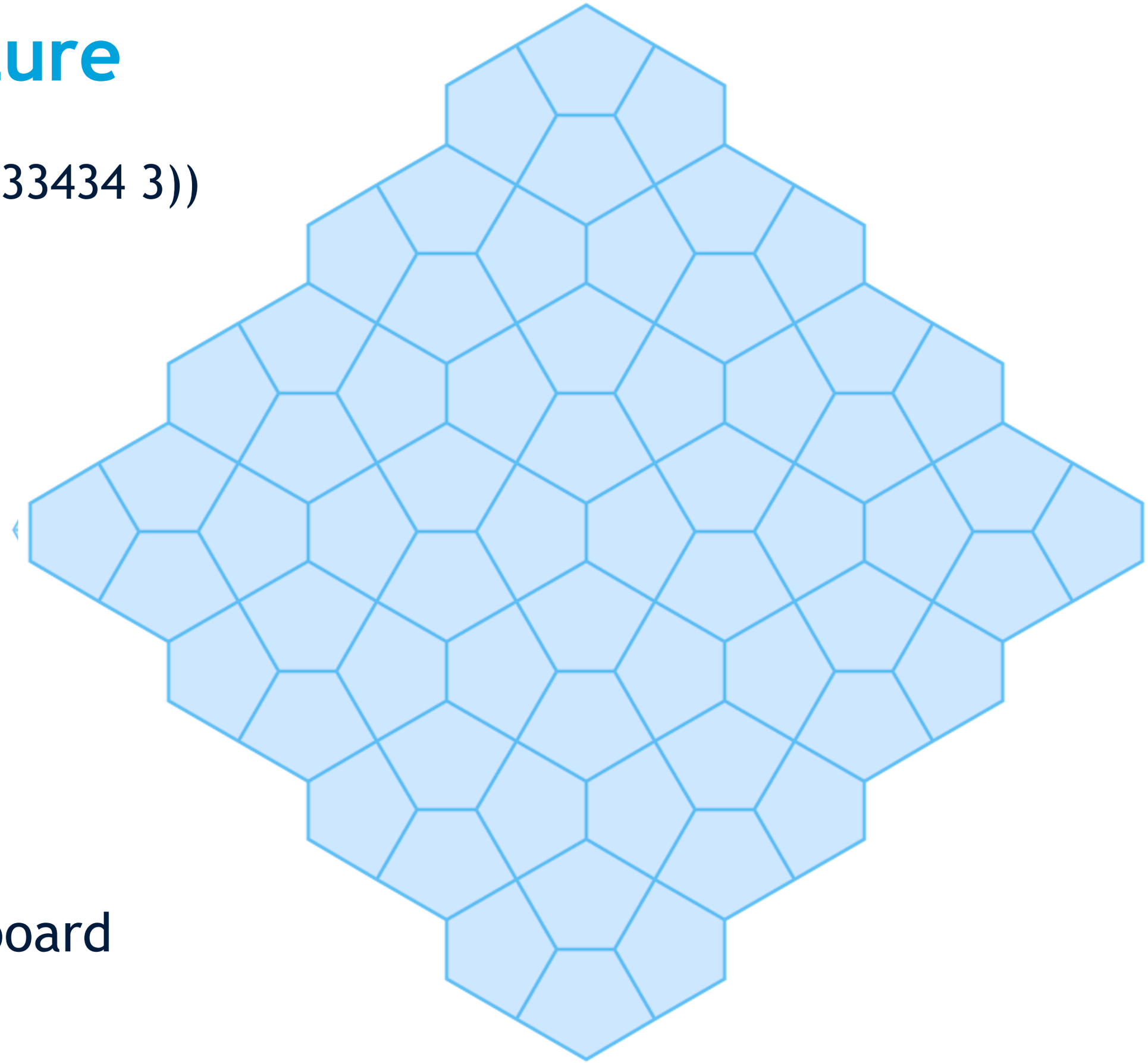
(tiling T33434 4)



What does dual look like?

Dual Nature

(dual (tiling T33434 3))

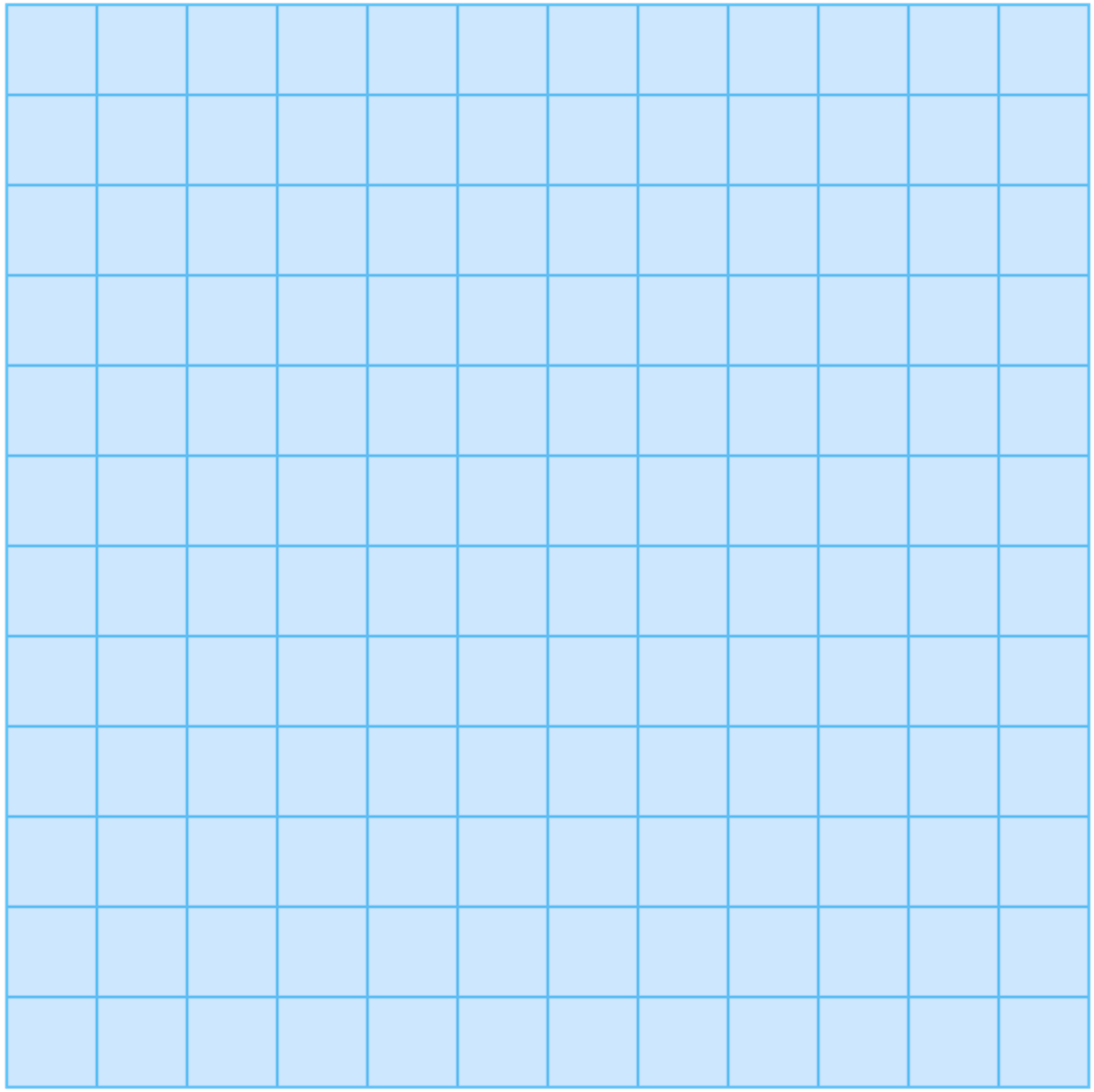


Cairo tiling!

Nice game board

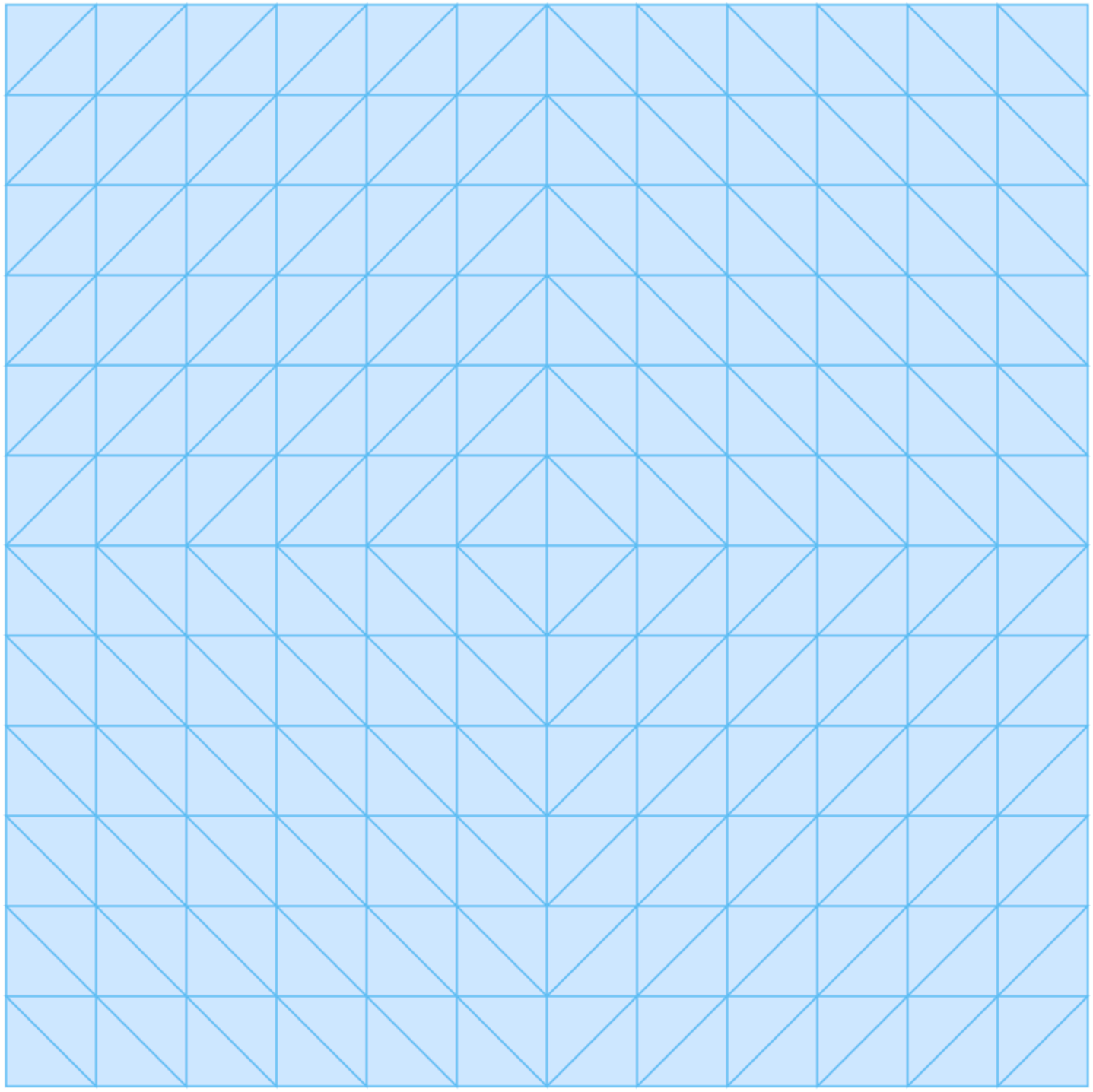
Conhex Tiling

(square 12)



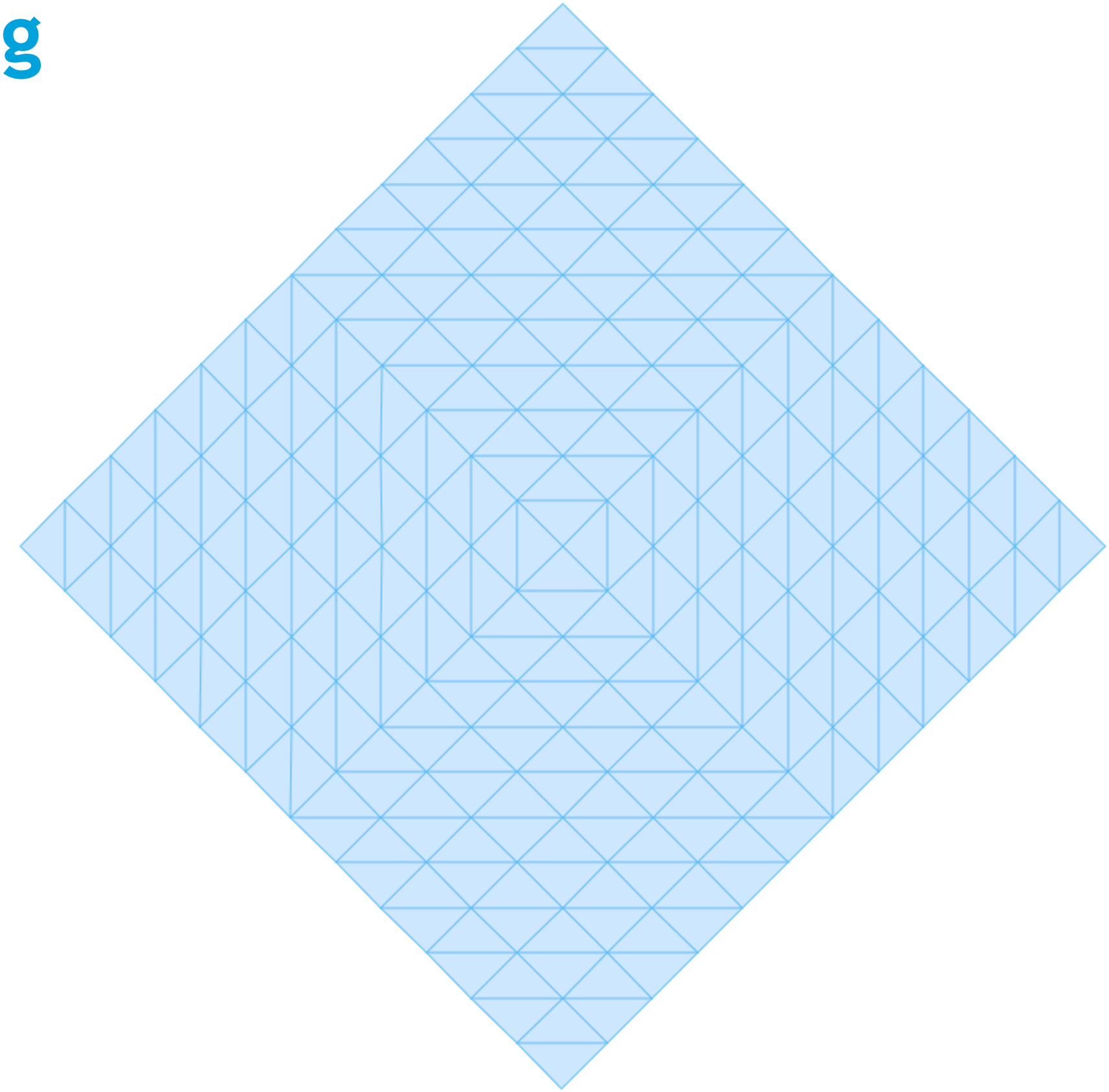
Conhex Tiling

(square 12)
diagonals:Concentric)



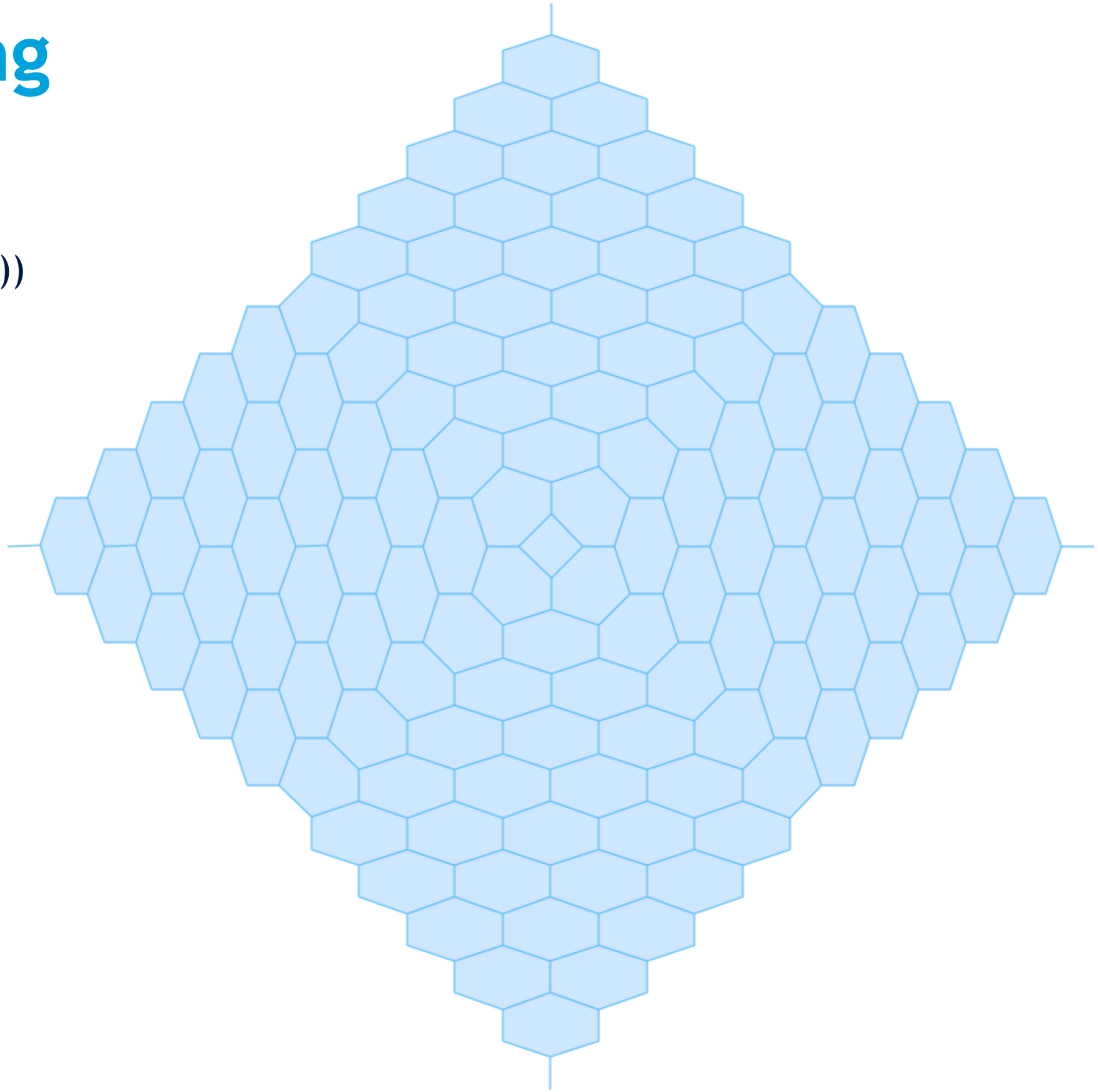
Conhex Tiling

```
(rotate 45
    (square 12
        diagonals:Concentric)
)
```



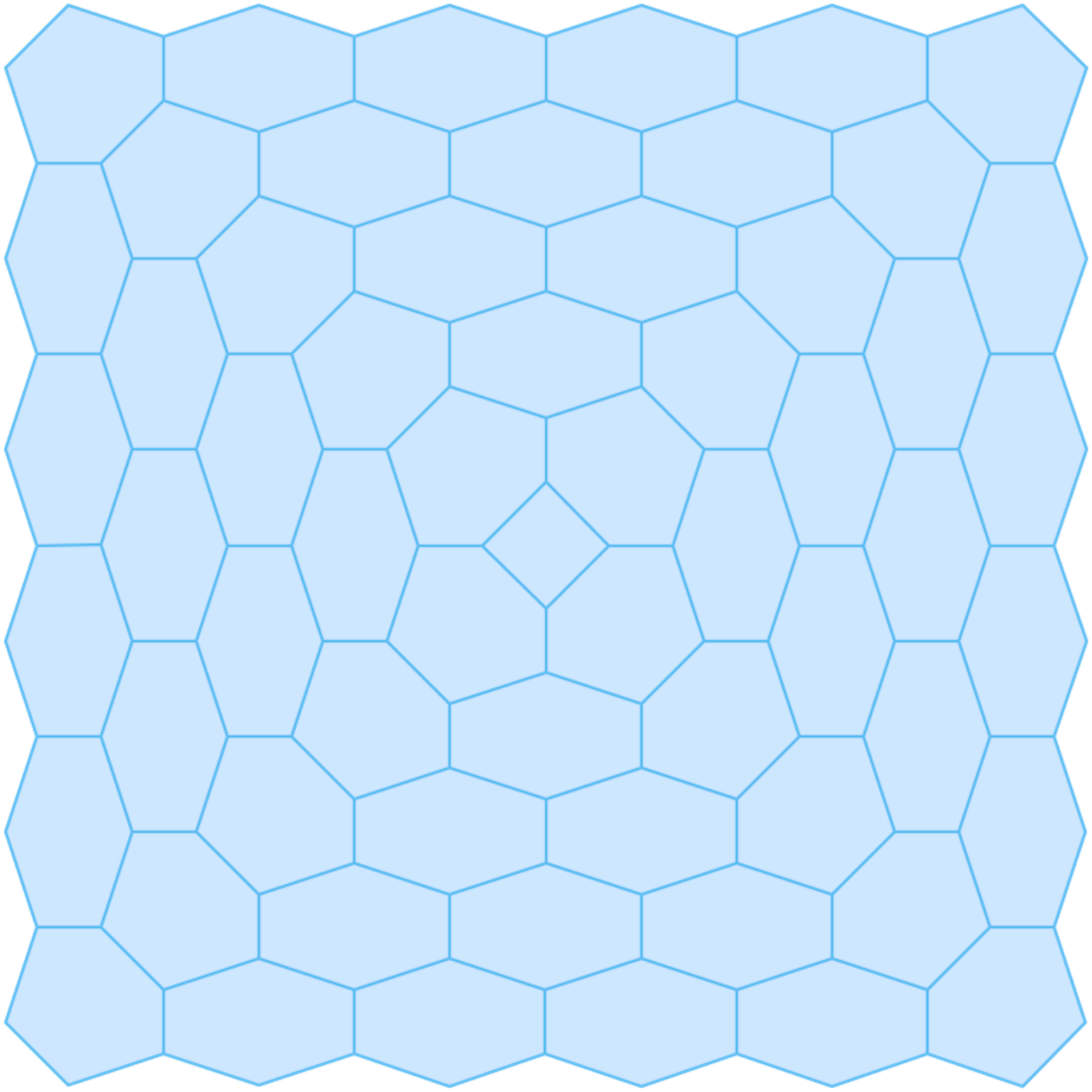
Conhex Tiling

```
(rotate 45
    (dual (square 12
        diagonals:Concentric))
)
```

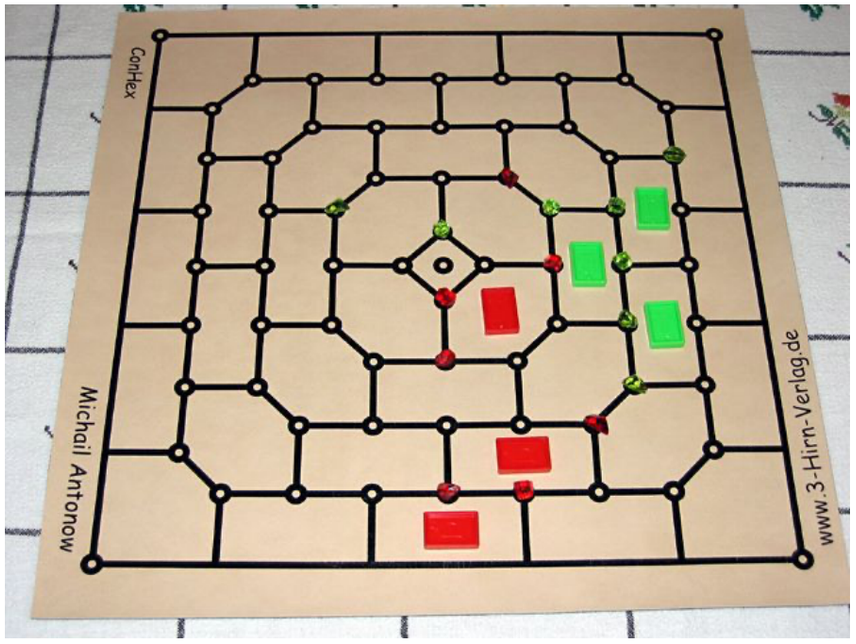


Conhex Tiling

```
(keep
    (rotate 45
    (dual (square 12
        diagonals:Concentric)))
    (poly { {2 2} {2 10}
        {10 10} {10 2} })
)
```



Conhex board:

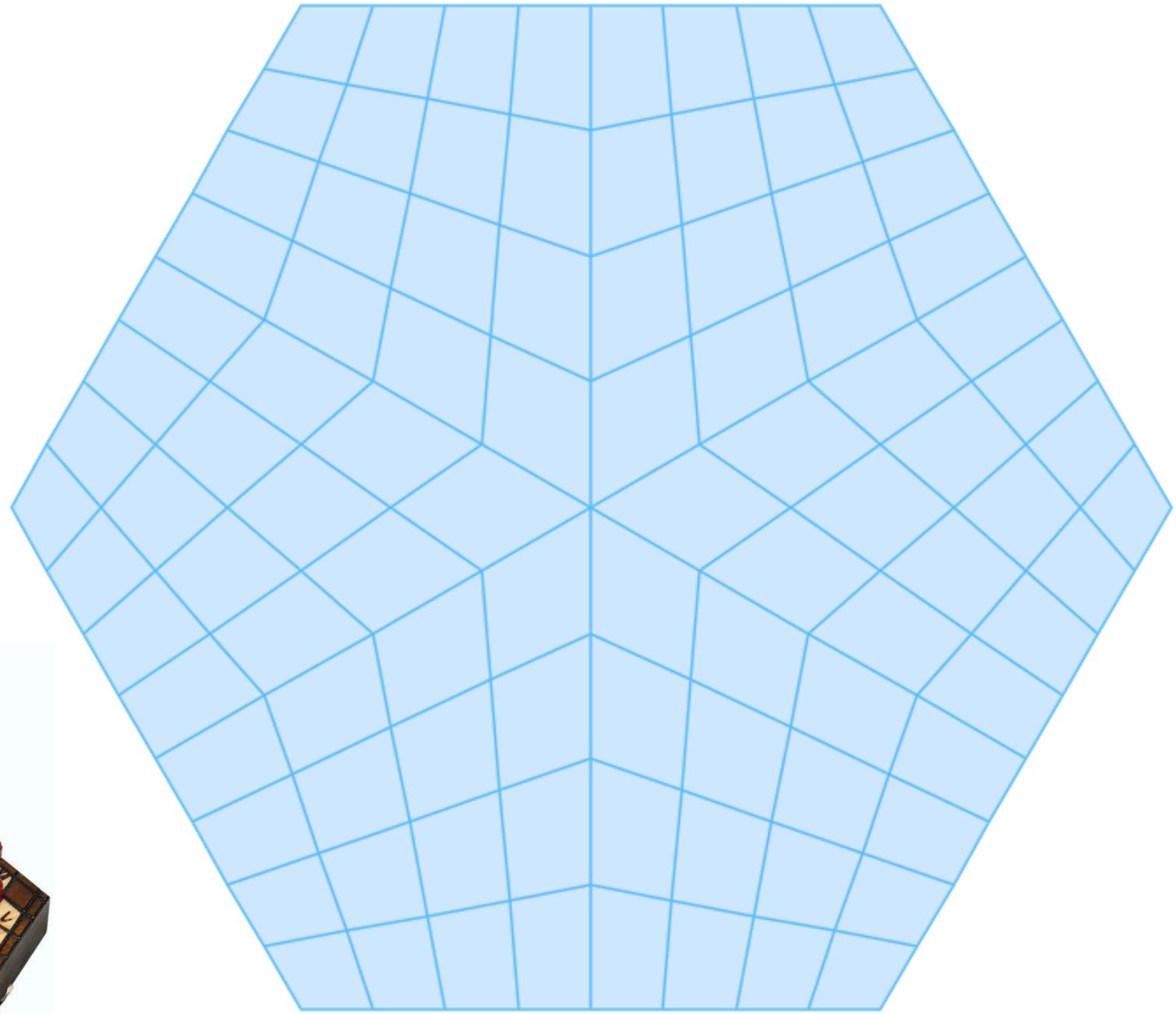


Is trivalent!

Quadhex Board

The “quad hex” board
- Quadrilateral tiling
- Hexagonal shape

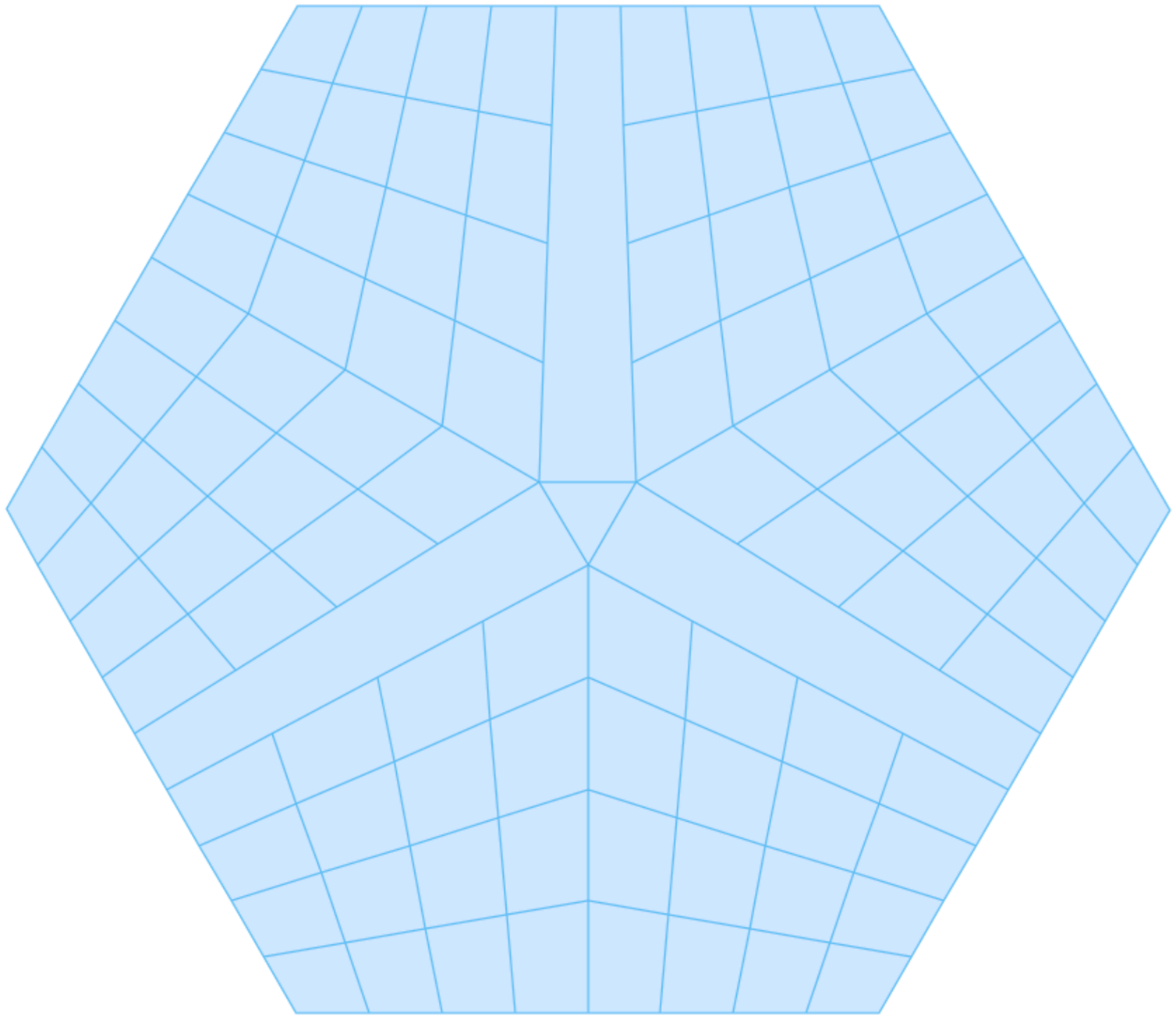
e.g. (quadhex 4)



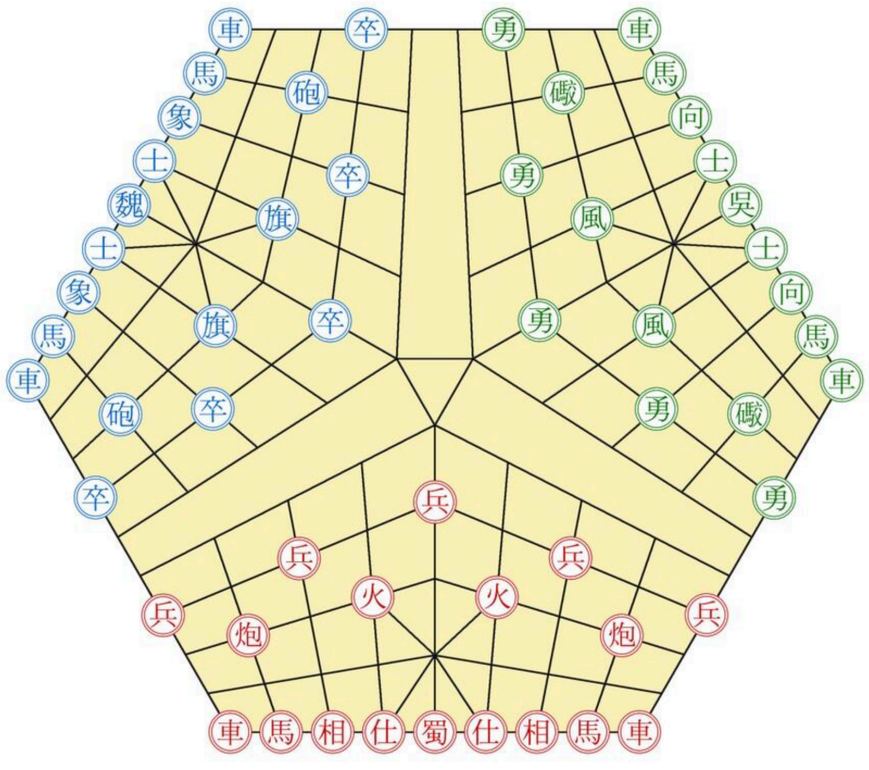
Three-player Chess:

Quadhex Board

(quadhex 4 thirds:true)

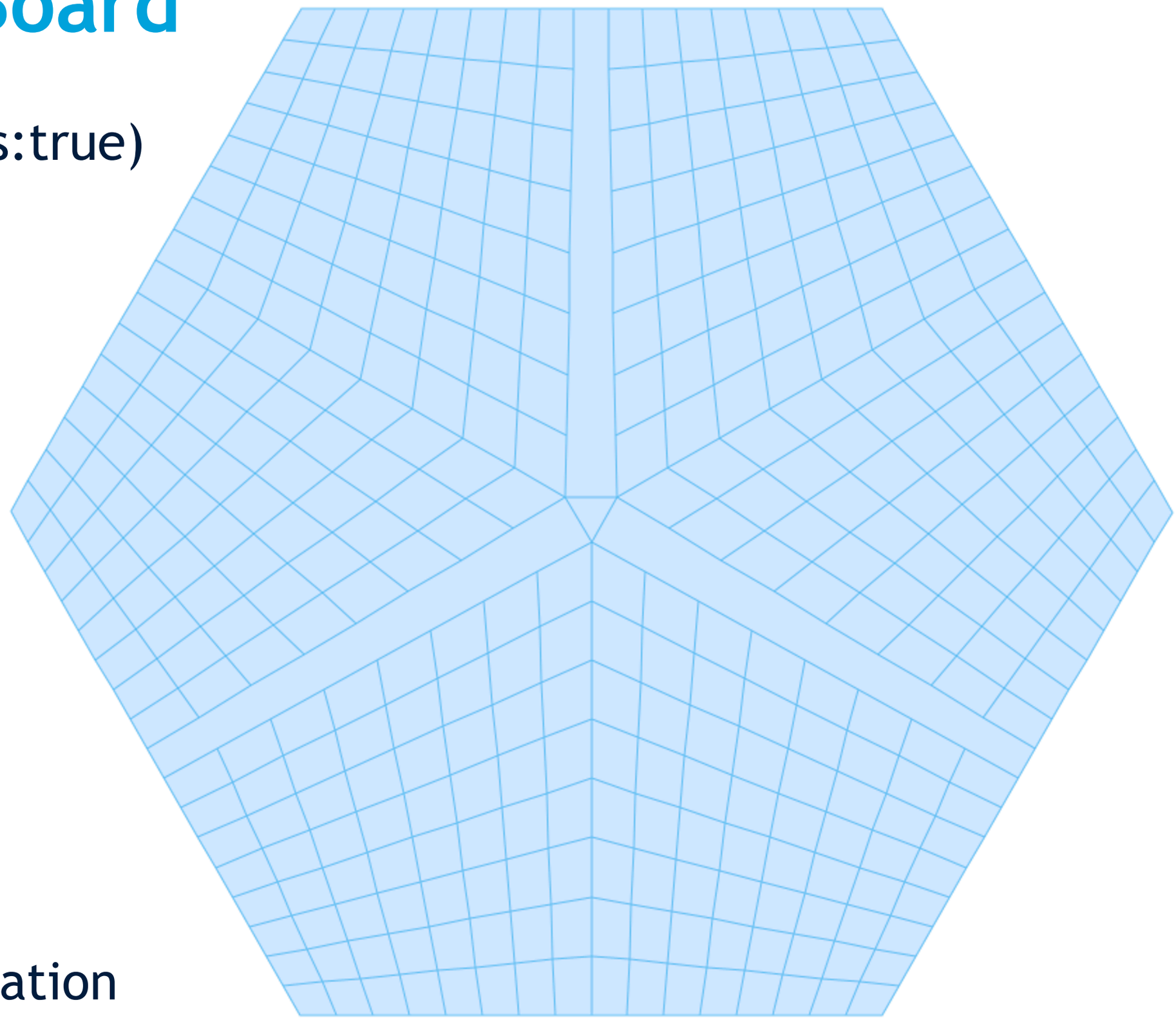


Three-player Chinese Chess:



Quadhex Board

(quadhex 8 thirds:true)

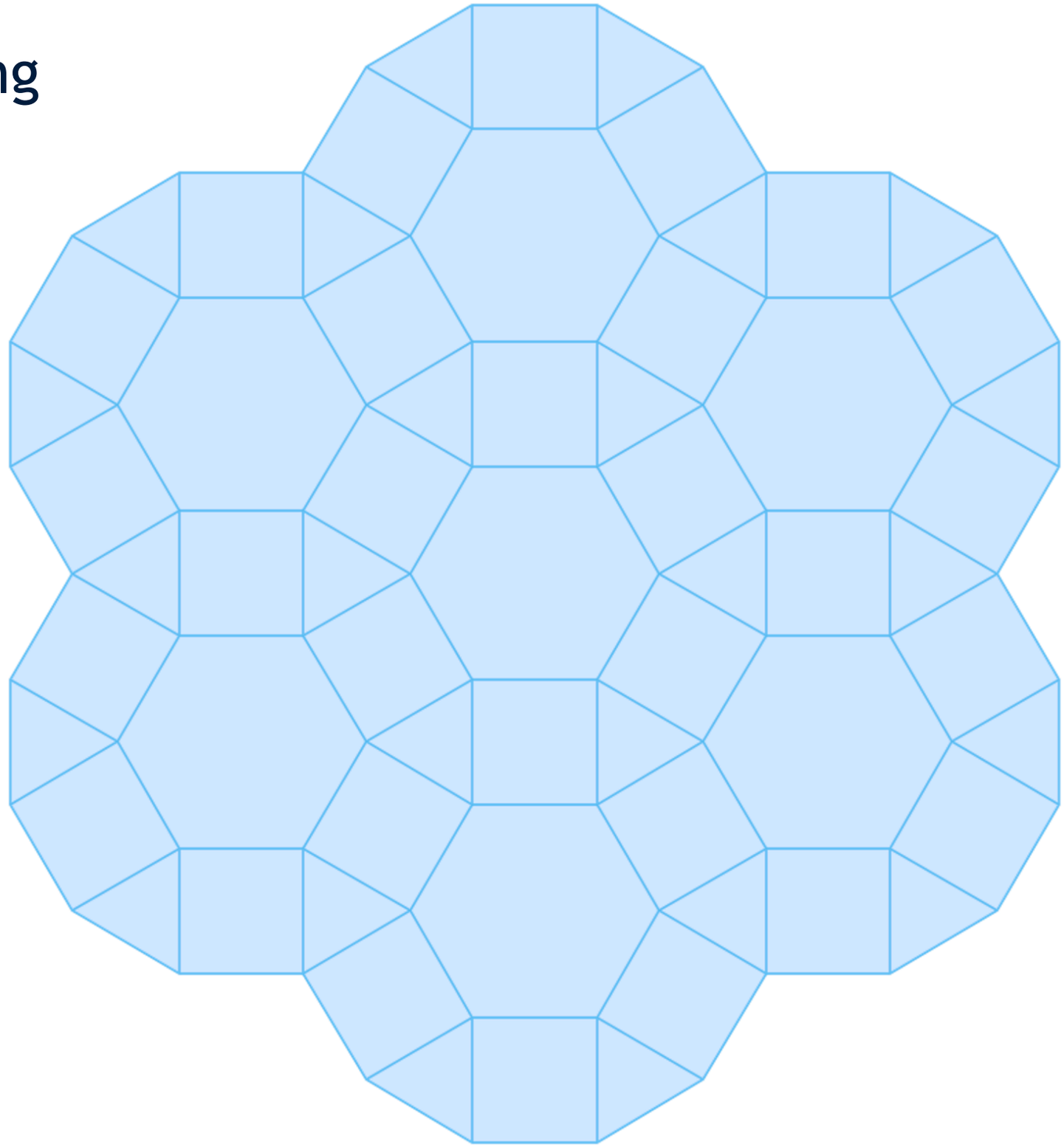


Easy parameterisation

Rhombitrihexahedral Dual Progression

Rhombitrihexahedral tiling
- Semi-regular 3.4.6.4
- Kensington board

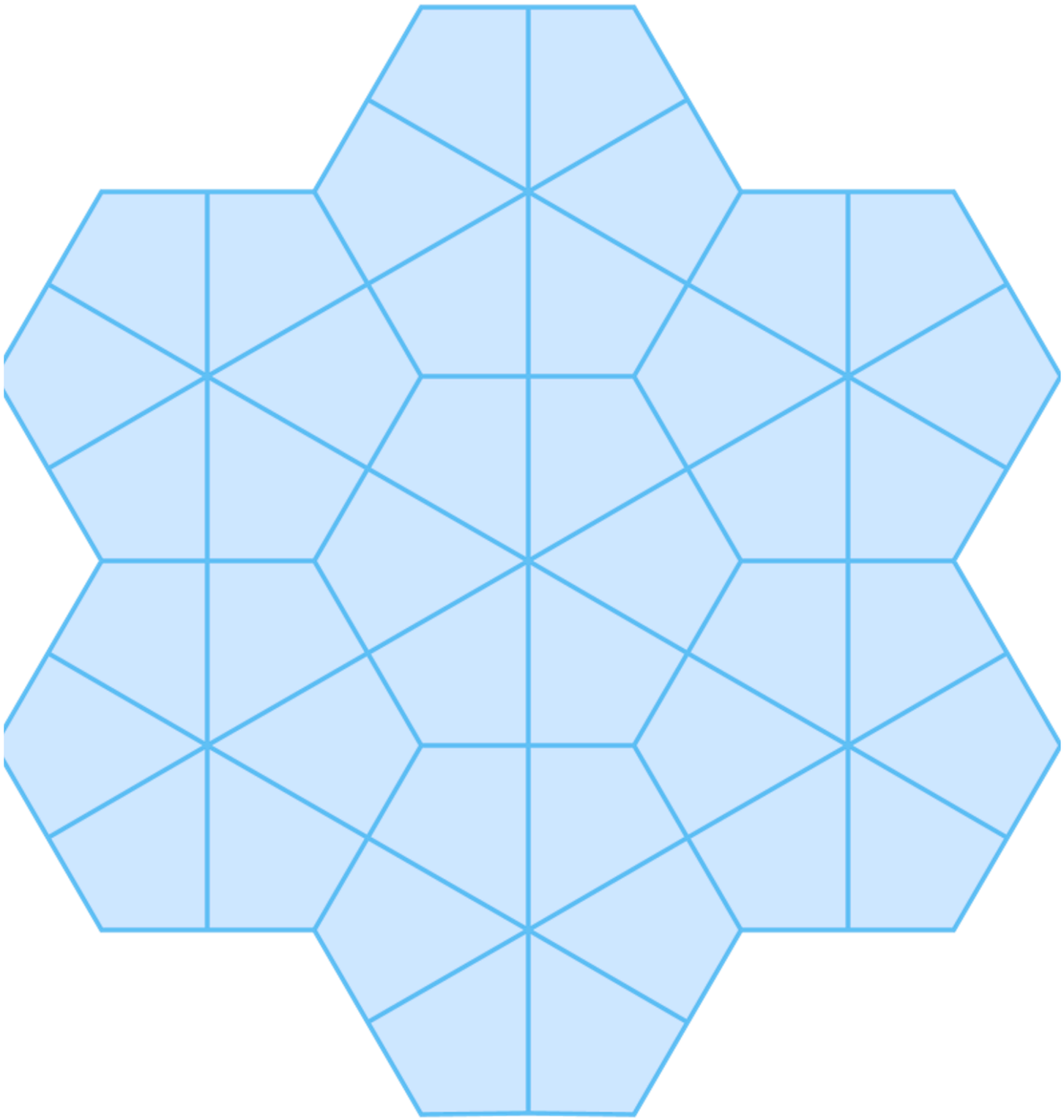
(tiling T3464 2)



Rhombitrihexahedral Dual Progression

(dual
   (tiling T3464 2)
)

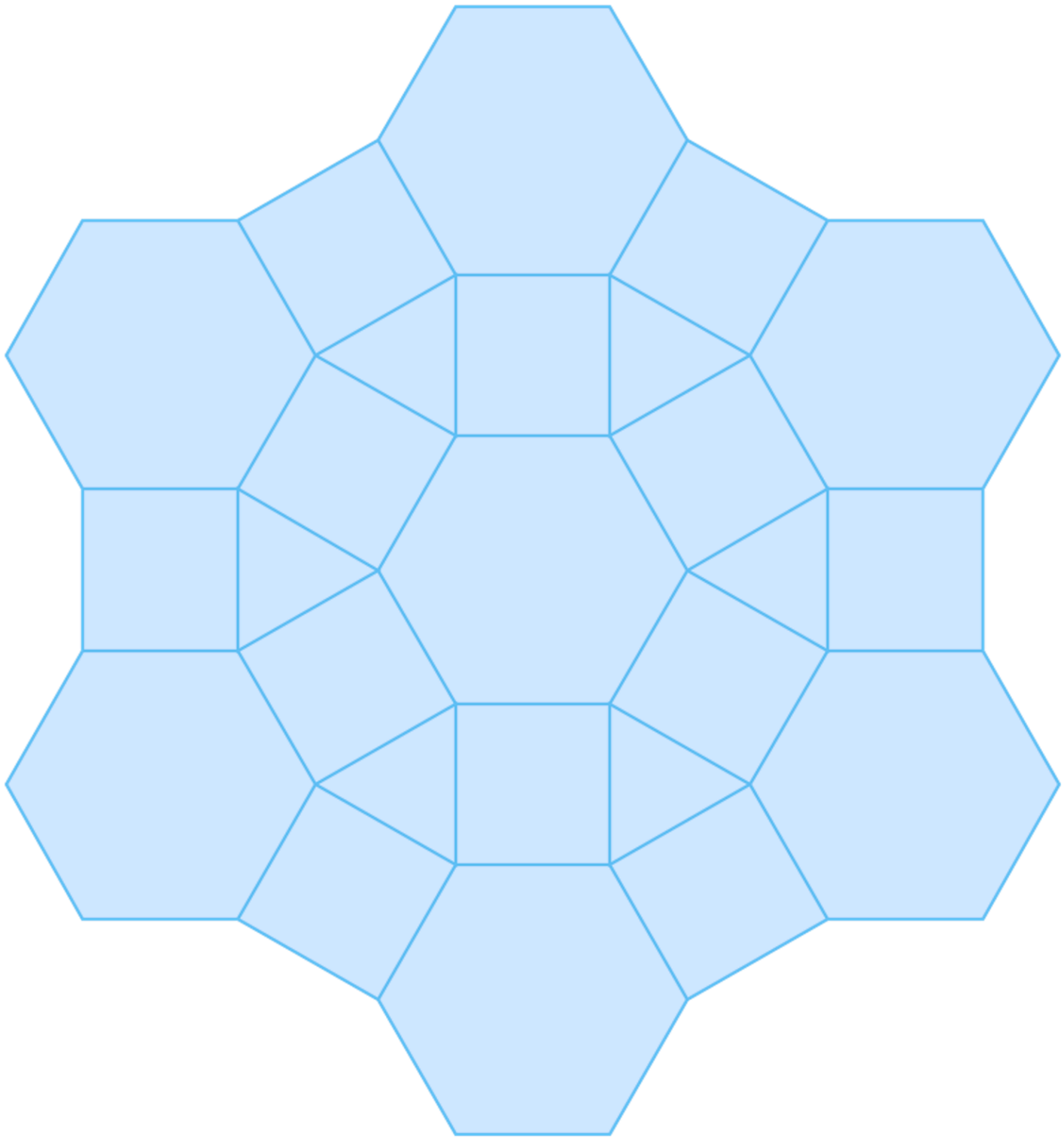
Quadrilaterals in
a triangular basis



Rhombitrihexahedral Dual Progression

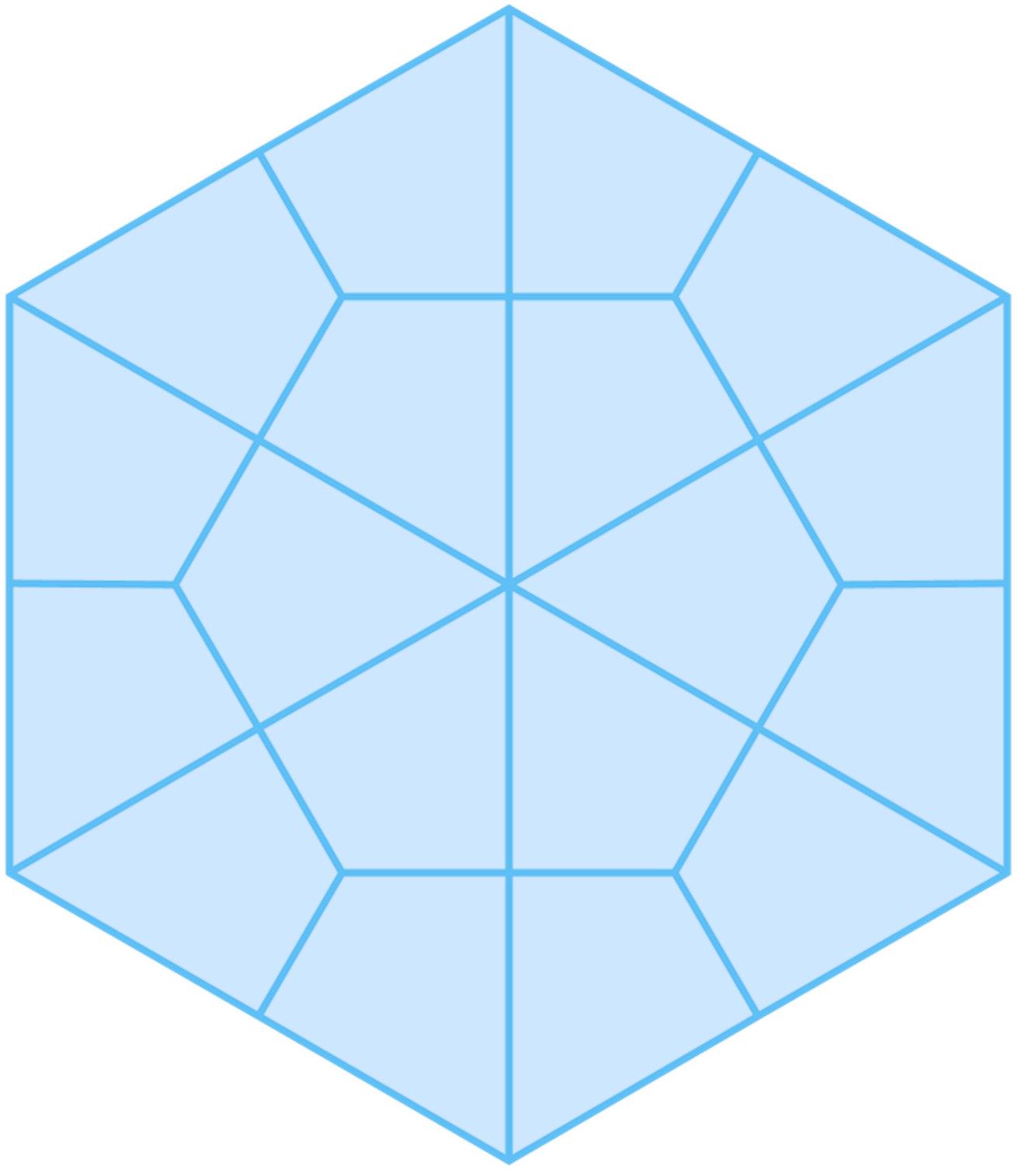
```
(dual
  (dual
    (tiling T3464 2)
  )
)
```

…back to 3.4.6.4



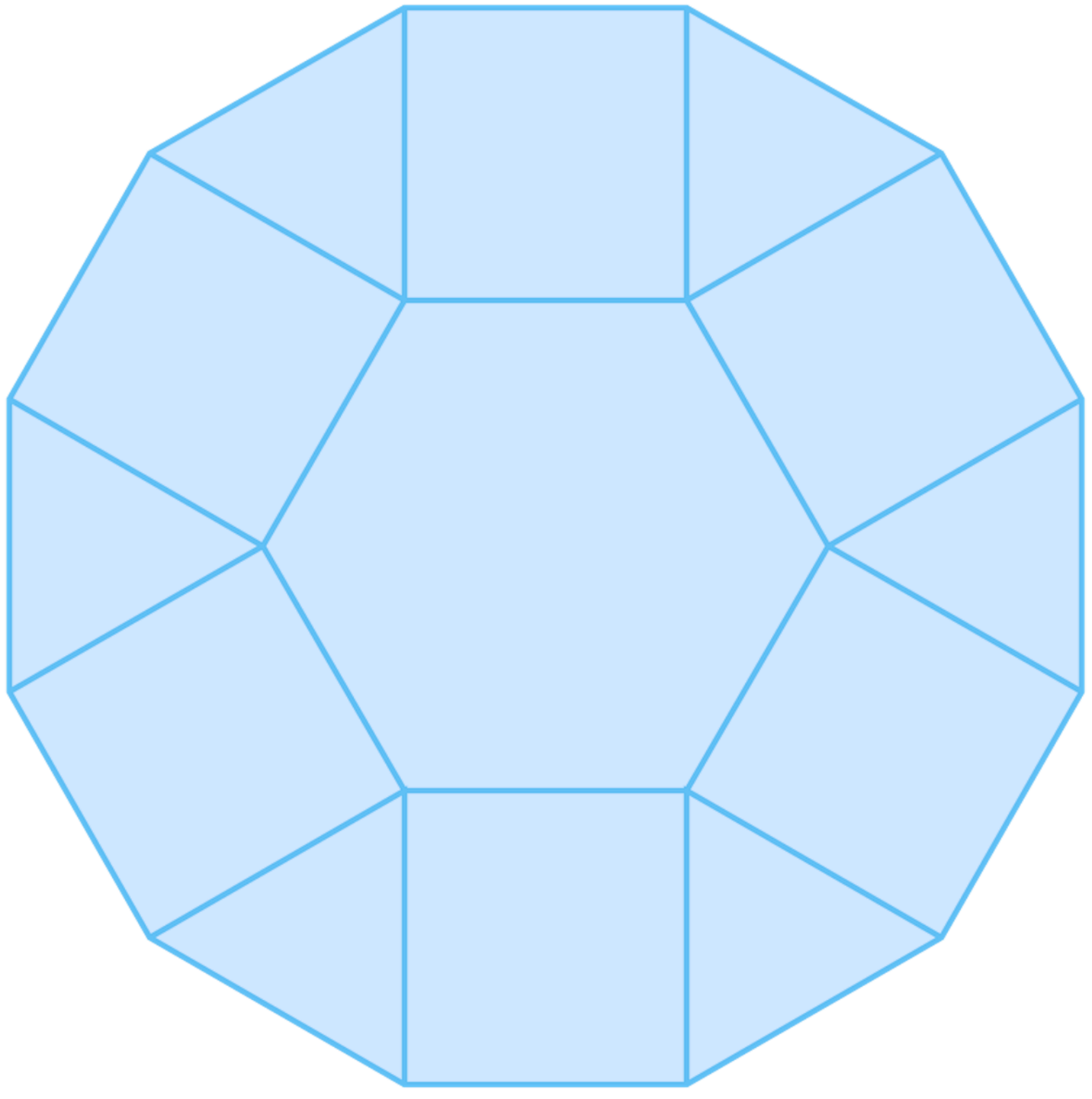
Rhombitrihexahedral Dual Progression

```
(dual
  (dual
    (dual
      (dual
        (tiling T3464 2)
      )
    )
  )
)
```



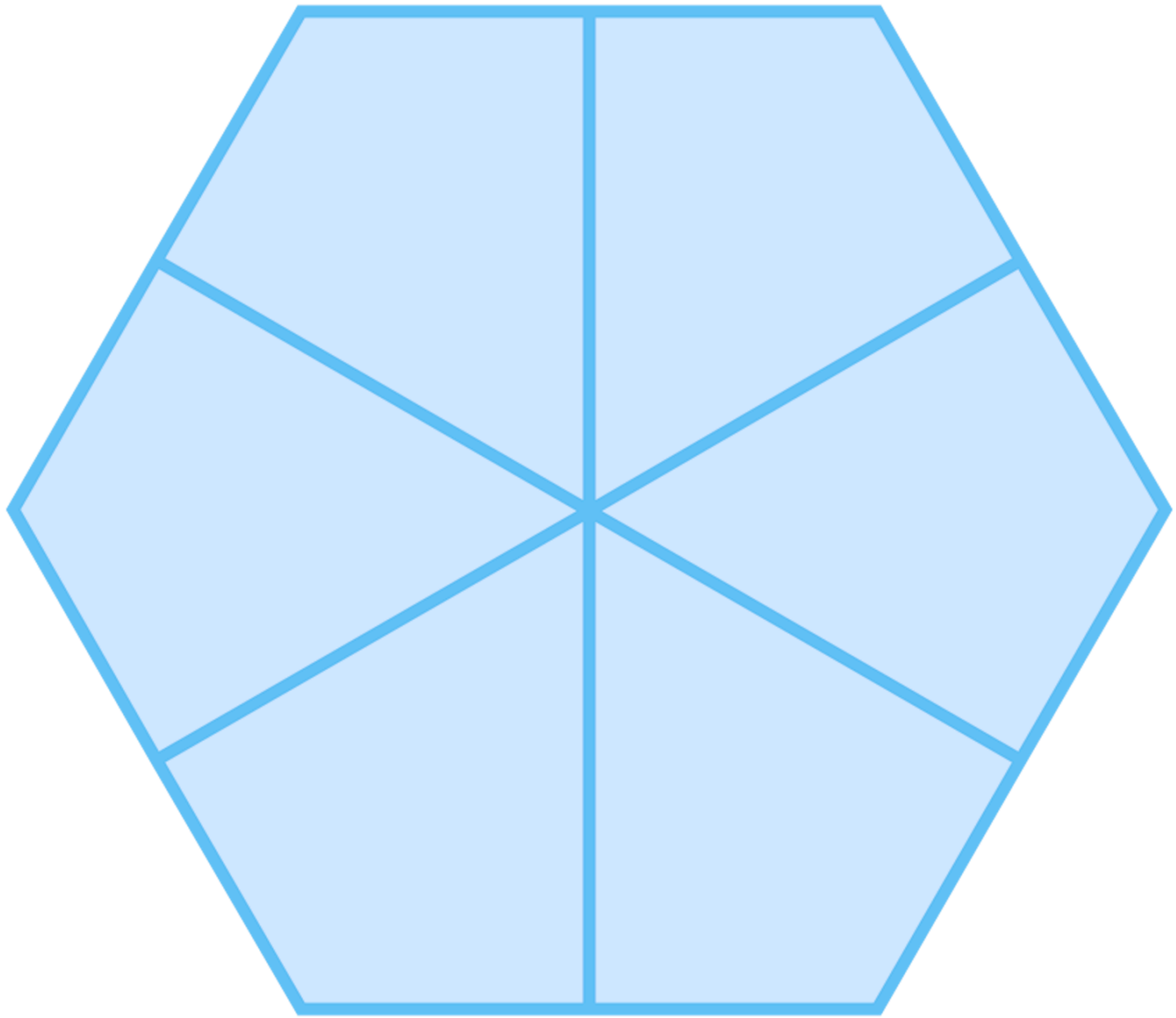
Rhombitrihexahedral Dual Progression

```
(dual
  (dual
    (dual
      (dual
        (dual
          (tiling T3464 2)
        )
      )
    )
  )
)
```



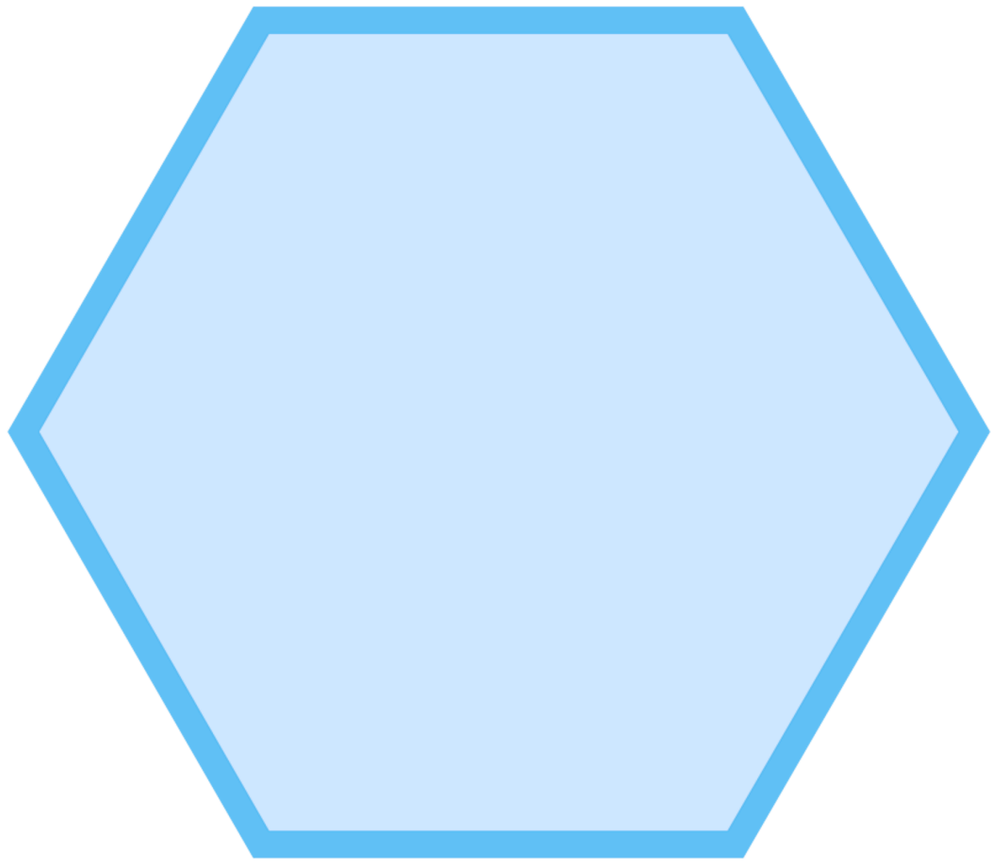
Rhombitrihexahedral Dual Progression

```
(dual
  (dual
    (dual
      (dual
        (dual
          (dual
            (tiling T3464 2)
          )
        )
      )
    )
  )
)
```



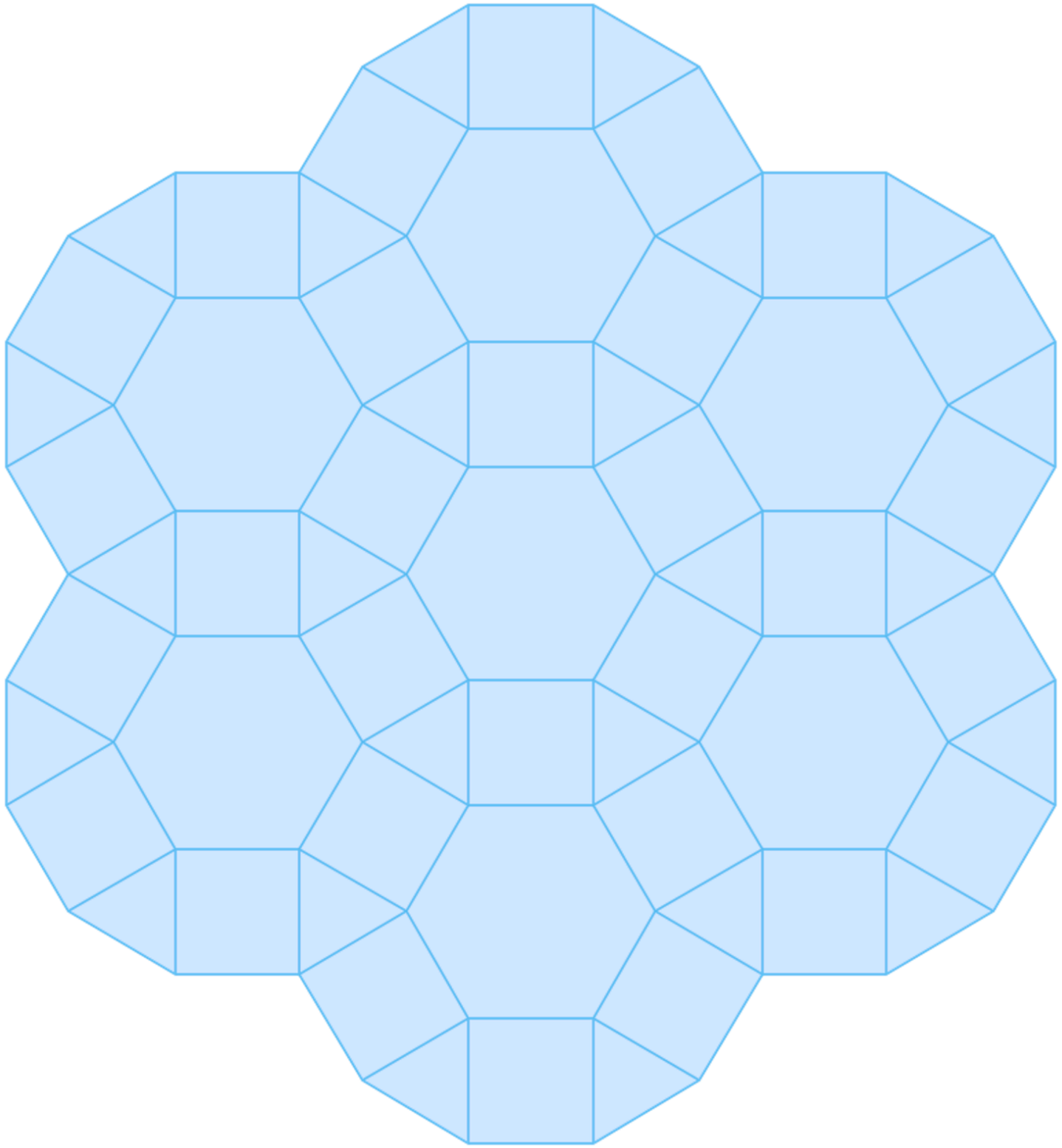
Rhombitrihexahedral Dual Progression

```
(dual
  (dual
    (dual
      (dual
        (dual
          (dual
            (dual
              (tiling T3464 2)
            )
          )
        )
      )
    )
  )
)
```



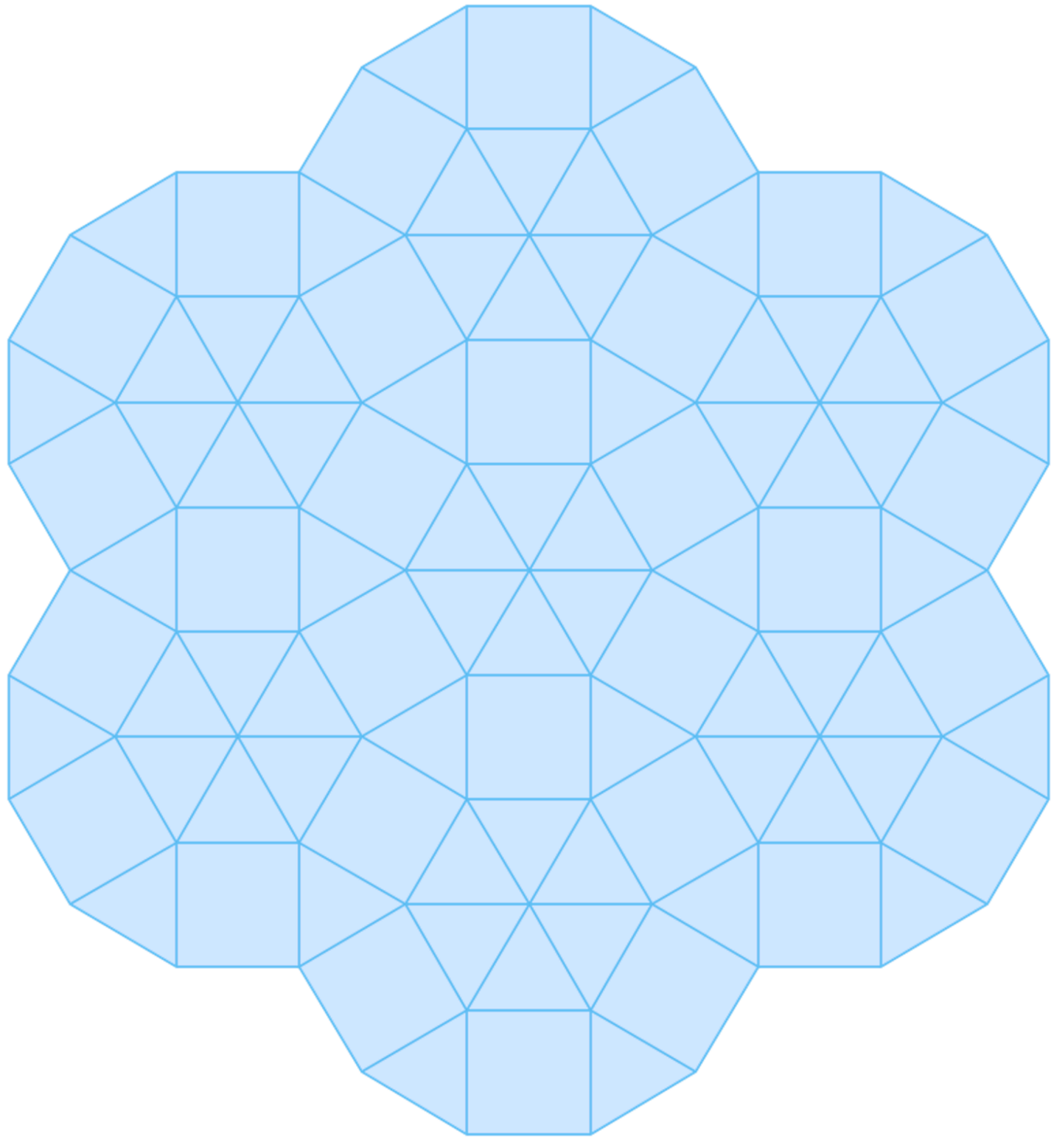
Rhombitrihexahedral Subdivision

(tiling T3464 2)



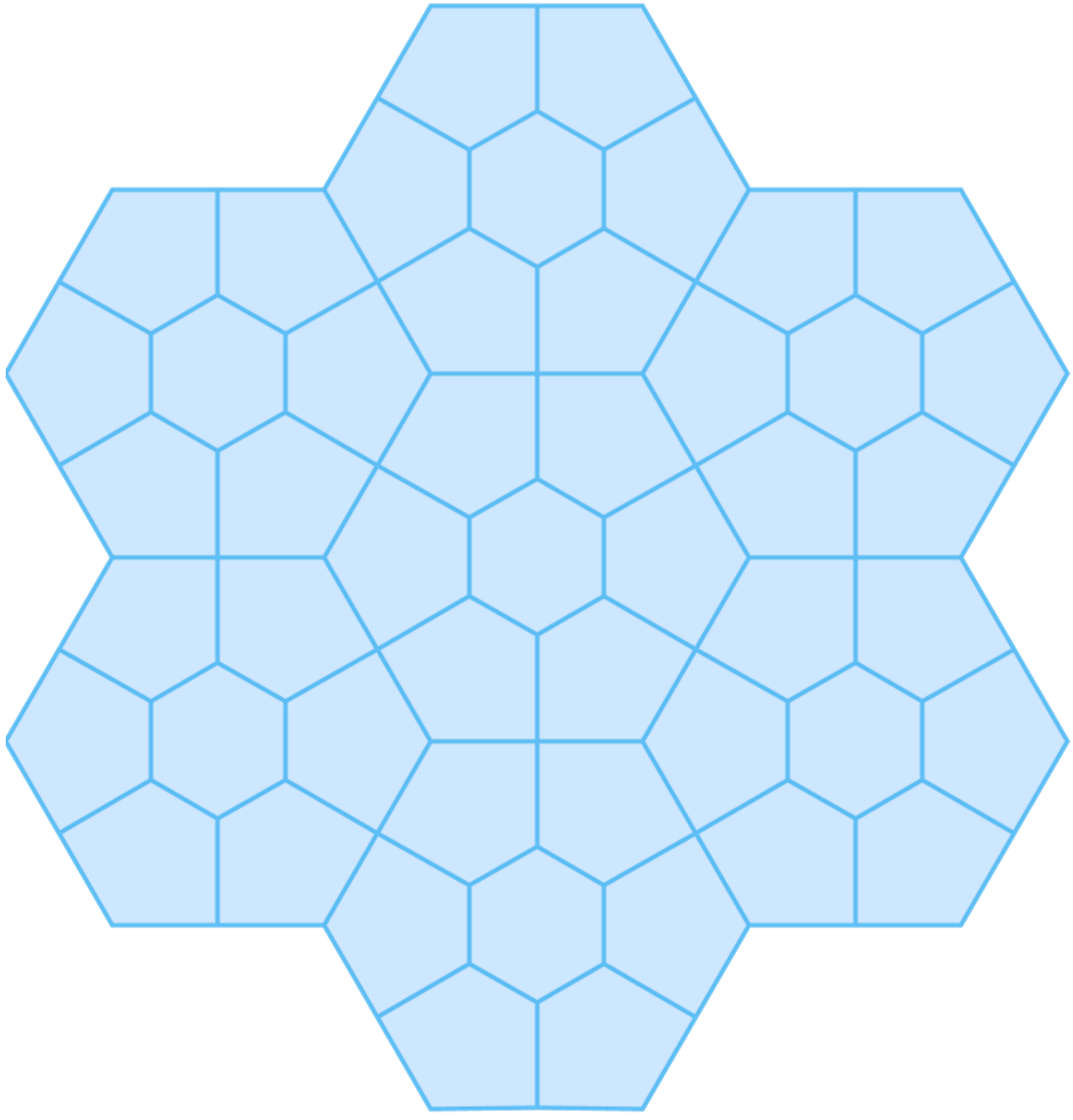
Rhombitrihexahedral Subdivision

```
(subdivide
   (tiling T3464 2) min:6
)
```



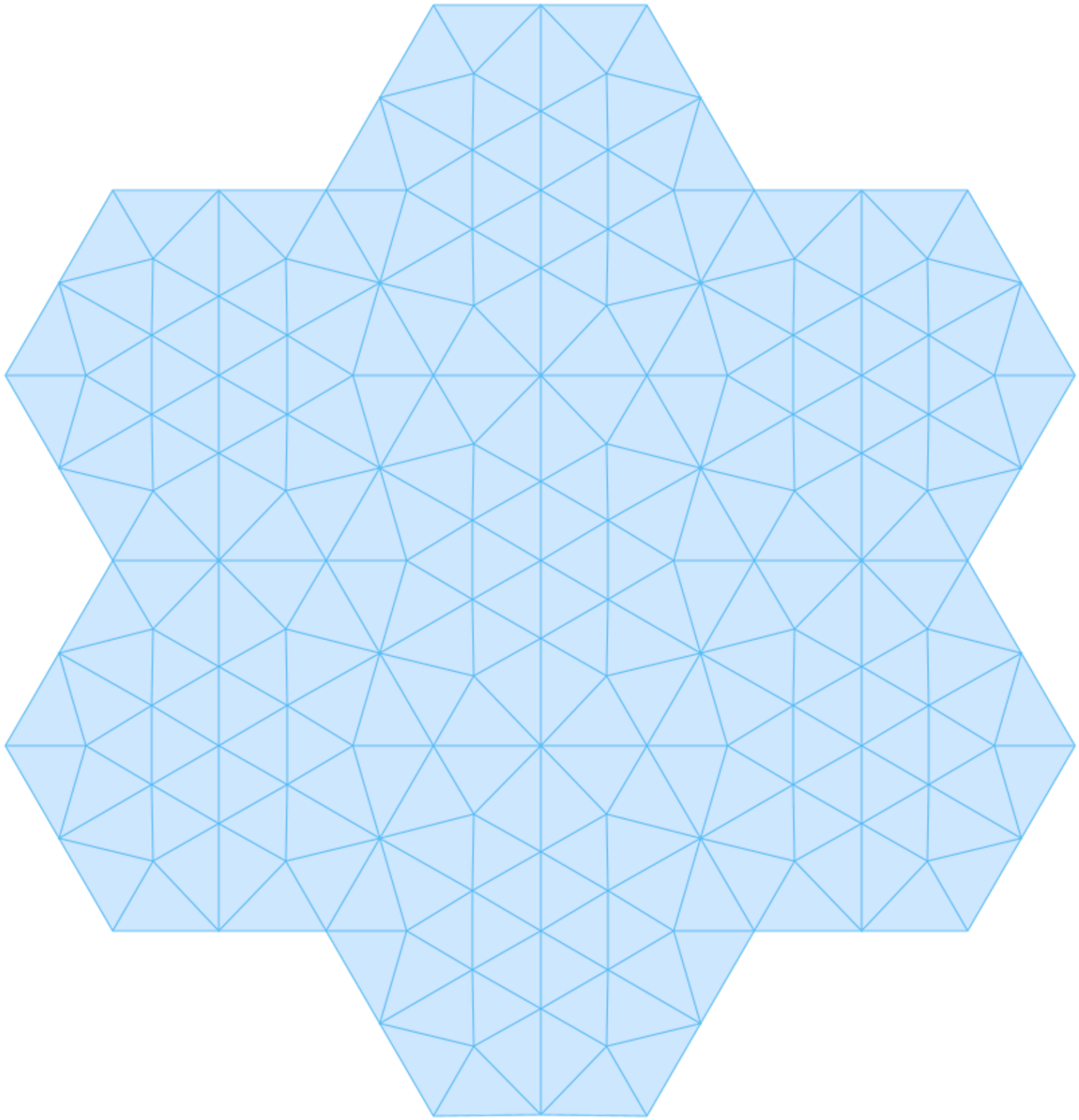
Rhombitrihexahedral Subdivision

```
(dual
   (subdivide
      (tiling T3464 2)
      min:6
   )
)
```



Rhombitrihexahedral Subdivision

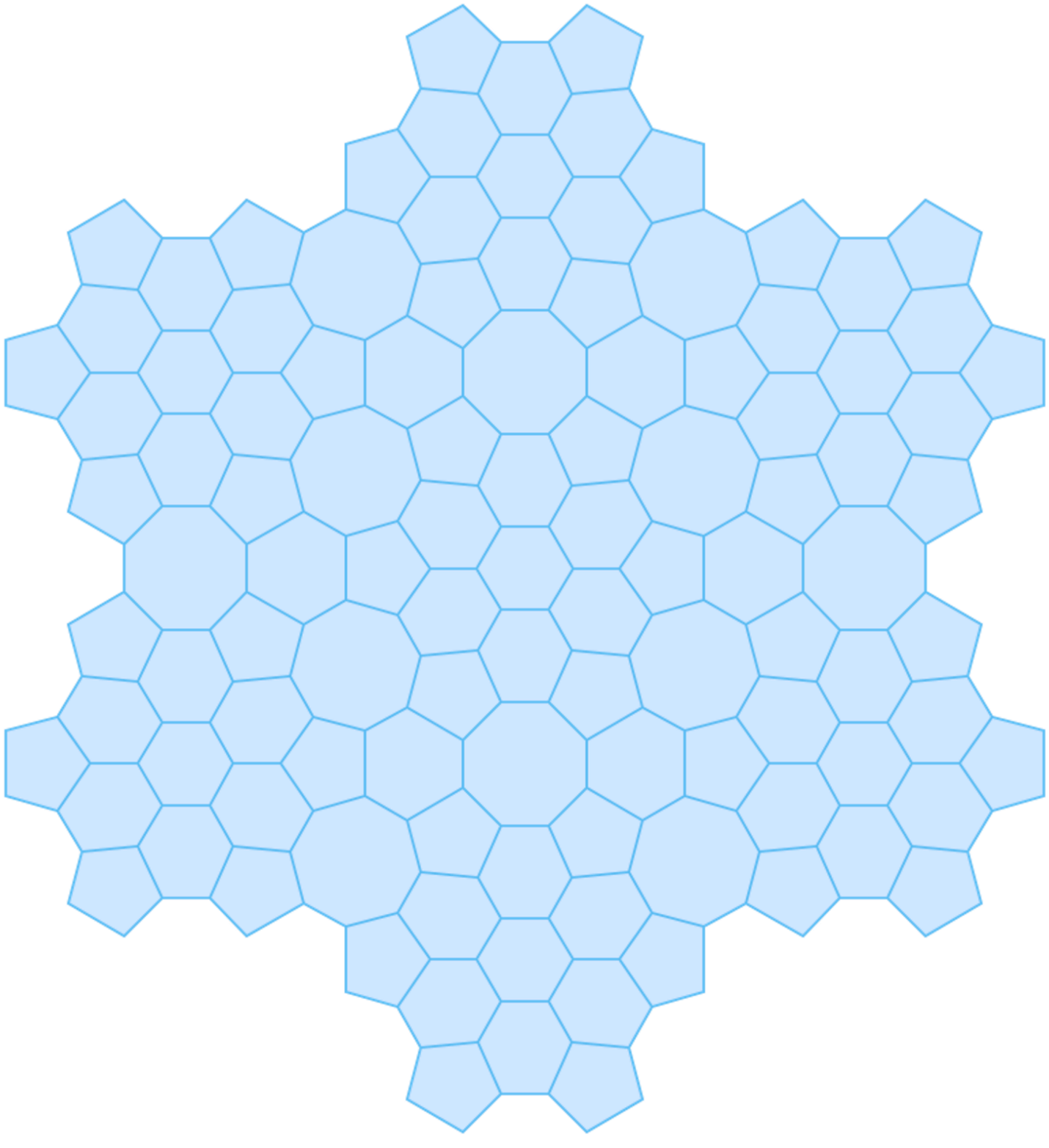
```
(subdivide
   (dual
     (subdivide
        (tiling T3464 2)
        min:6
      )
   )
)
```



Subdivision increases the number of cells again

Rhombitrihexahedral Subdivision

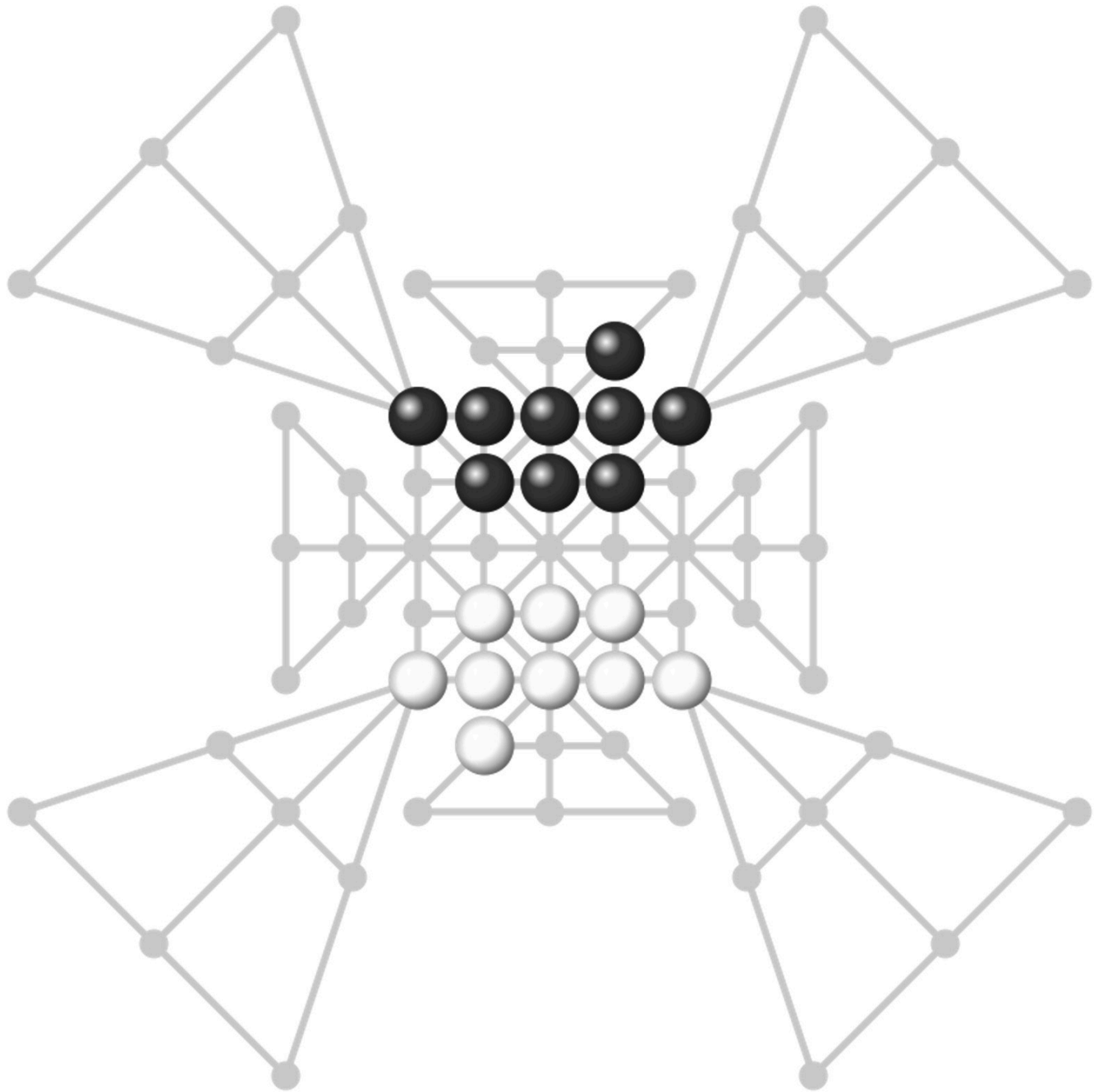
```
(dual
  (subdivide
    (dual
      (subdivide
        (tiling T3464 2)
        min:6
      )
    )
  )
)
```



Terhuchu

Tehuchu board, old format:

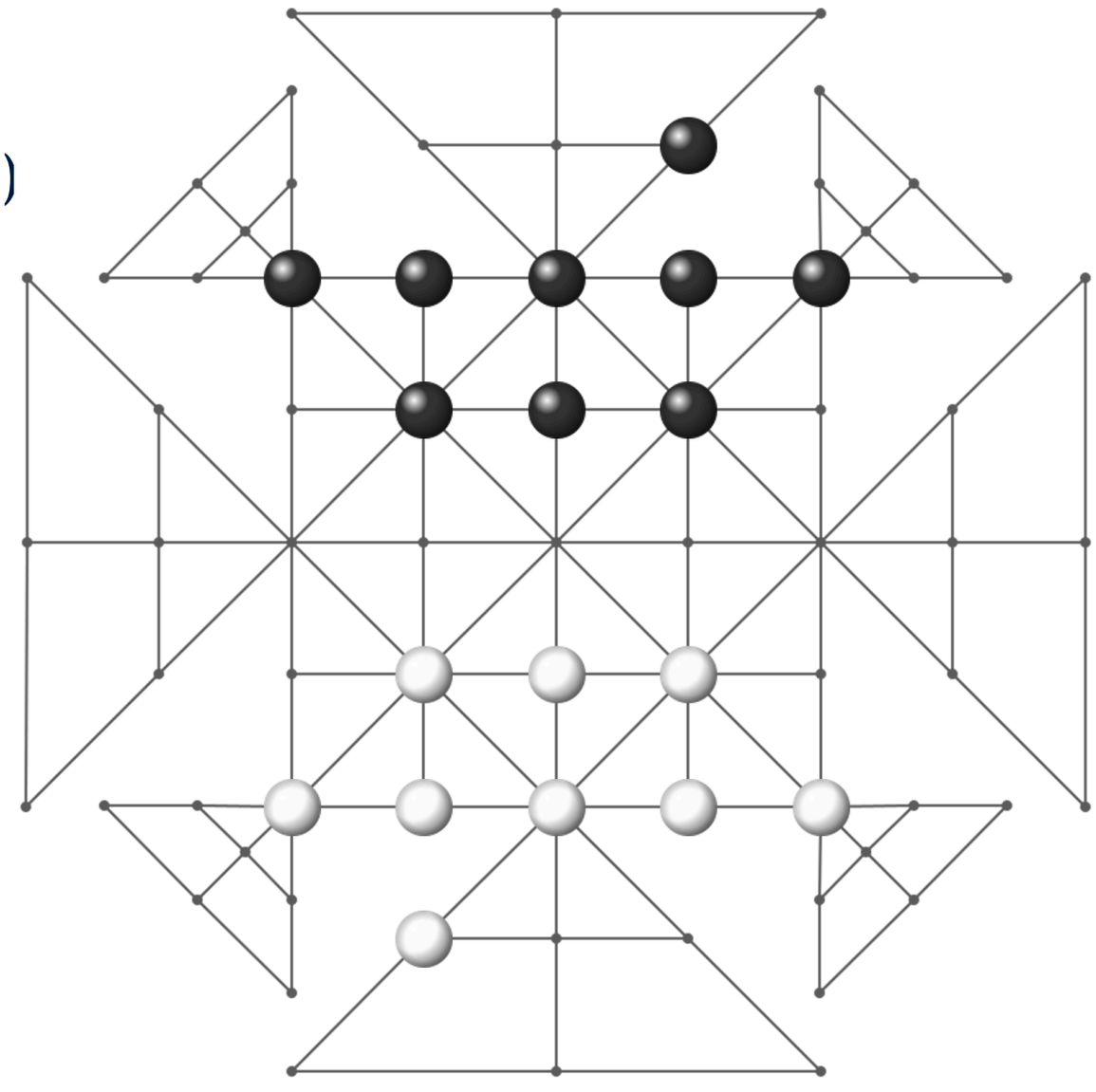
```
(peralikatumaBoard
   5
                top:true
                left:true
                right:true
                bottom:true
                bottomLeft:true
                bottomRight:true
                topLeft:true
                topRight:true
)
```



Terhuchu

Terhuchu board, new format:

```
(merge {
    (shift 2 2 (square 5 diagonals:Alternating))
    (shift 2 0 (wedge 3))
    (shift 5 3 (rotate 90 (wedge 3)))
    (shift 2 6 (rotate 180 (wedge 3)))
    (shift -1 3 (rotate 270 (wedge 3)))
    (shift 0.65 1.15 (scale 0.5
        (rotate -45 (wedge 3))))
    (shift 5.35 1.15 (scale 0.5
        (rotate 45 (wedge 3))))
    (shift 5.35 5.85 (scale 0.5
        (rotate 135 (wedge 3))))
    (shift 0.65 5.85 (scale 0.5
        (rotate -135 (wedge 3))))
    }
)
```

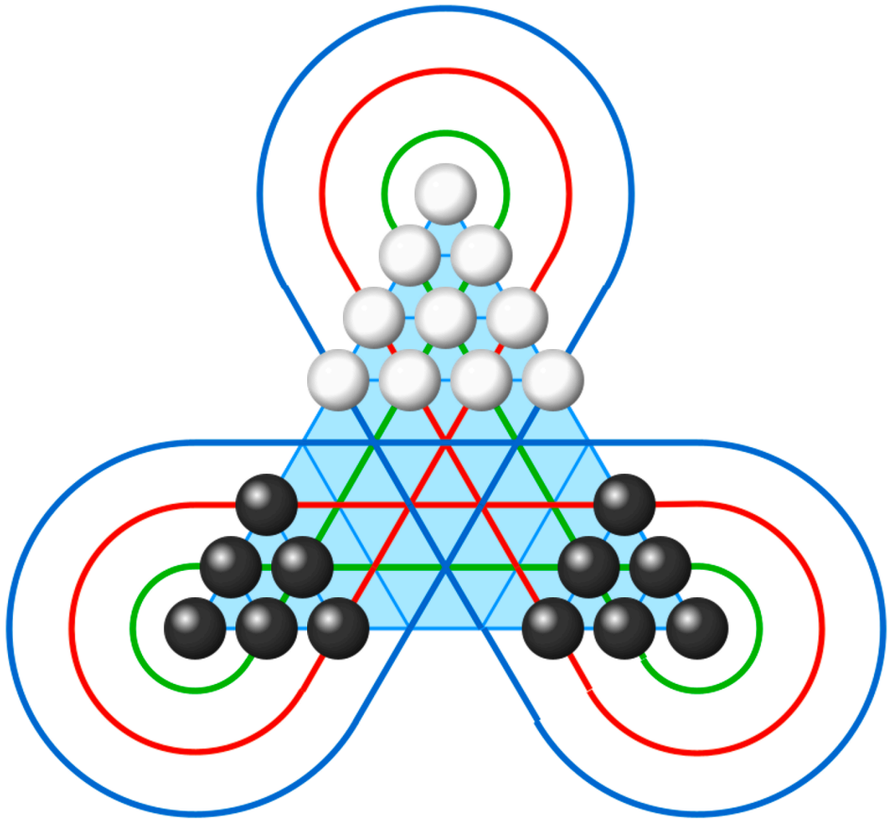


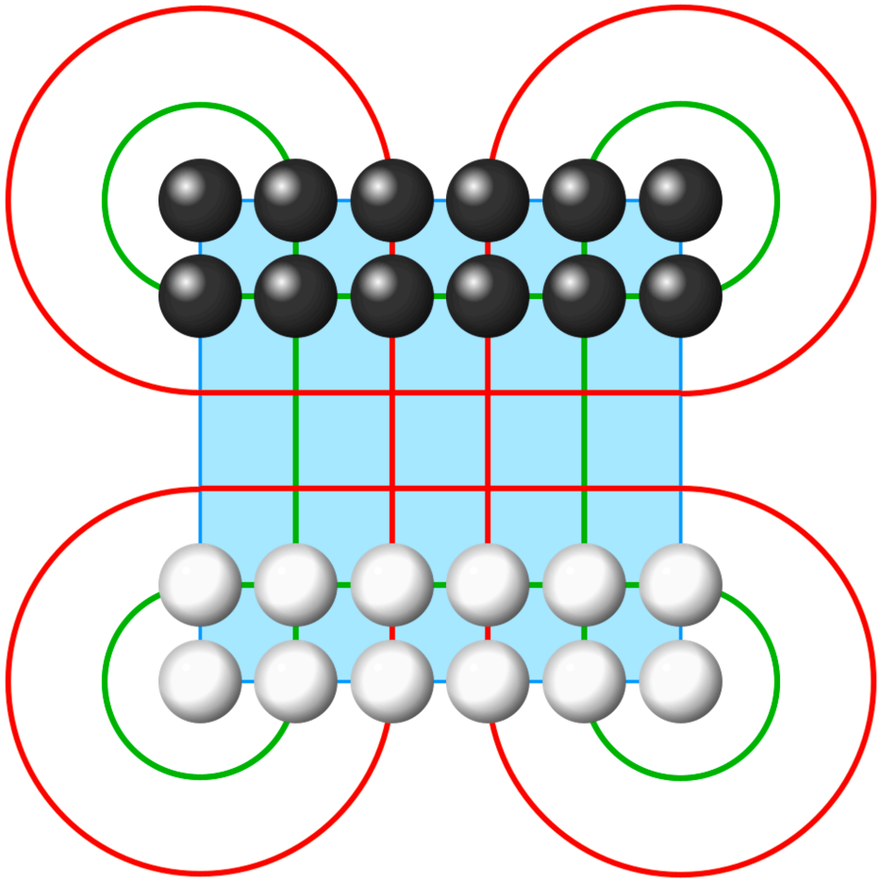
Less succinct but more general (square + wedges)

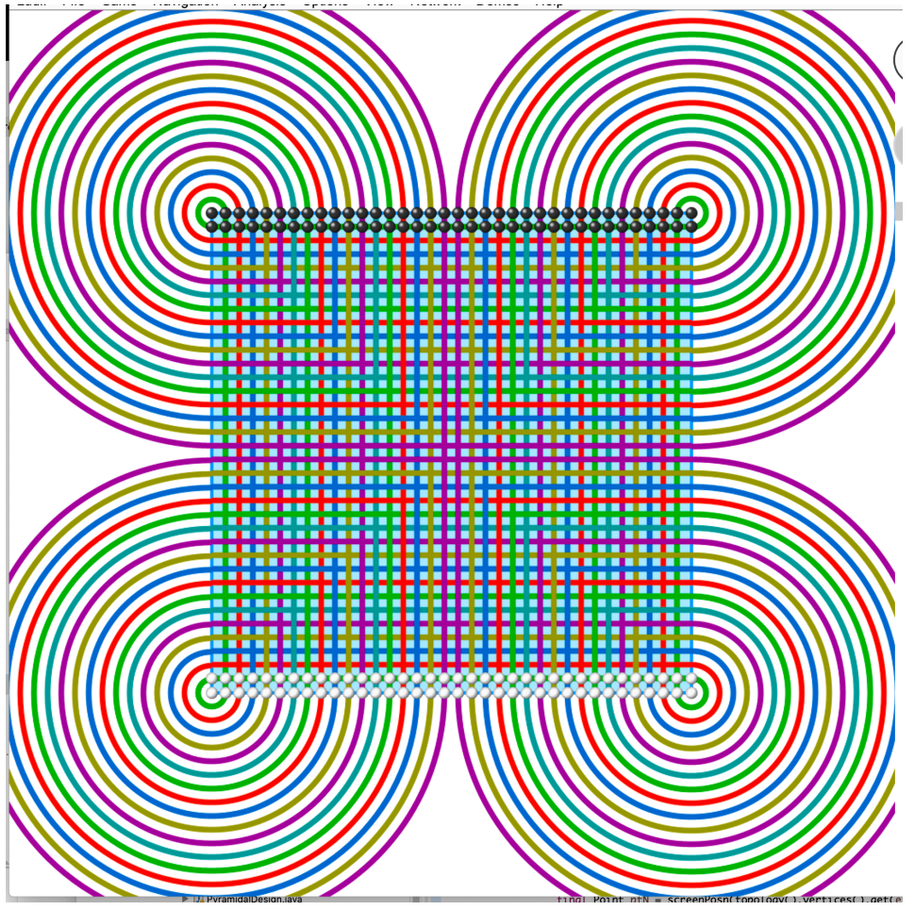
Custom Boards

1. Puzzles
2. Game of Surakarta

(surakartaBoard (square 6))
(surakartaBoard (tri 8))
(surakartaBoard (square 36))





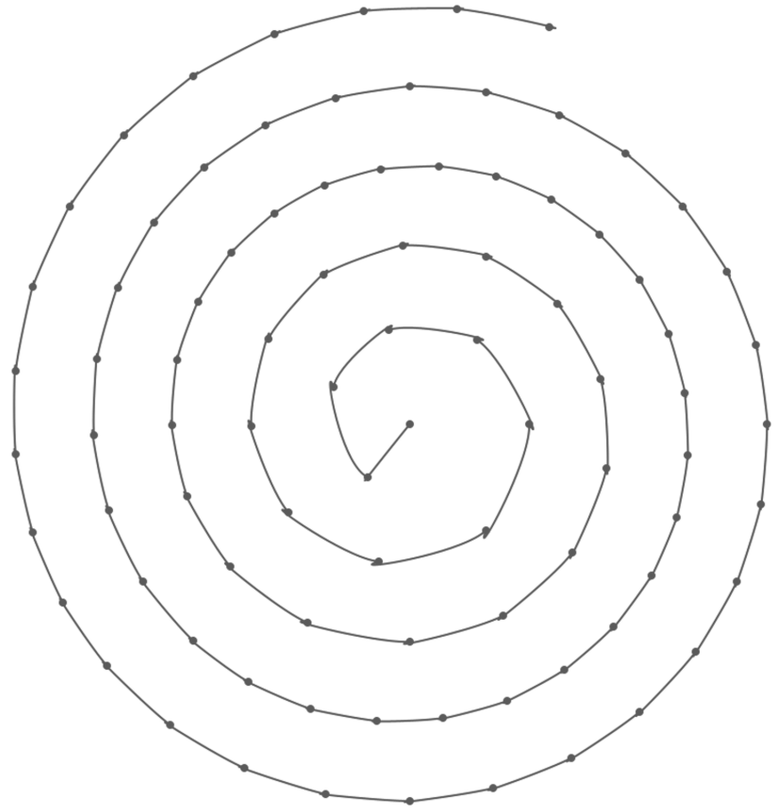


Graphics Metadata

Hints for drawing certain board types:

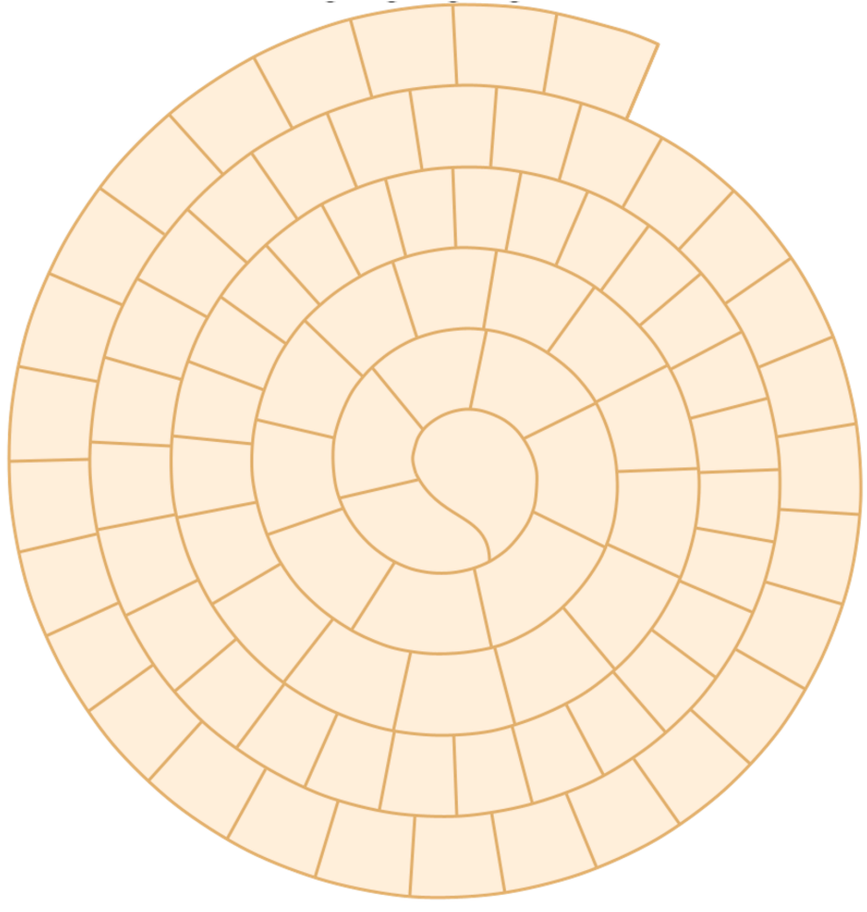
Plain spiral:

(spiral turns:5 sites:88)



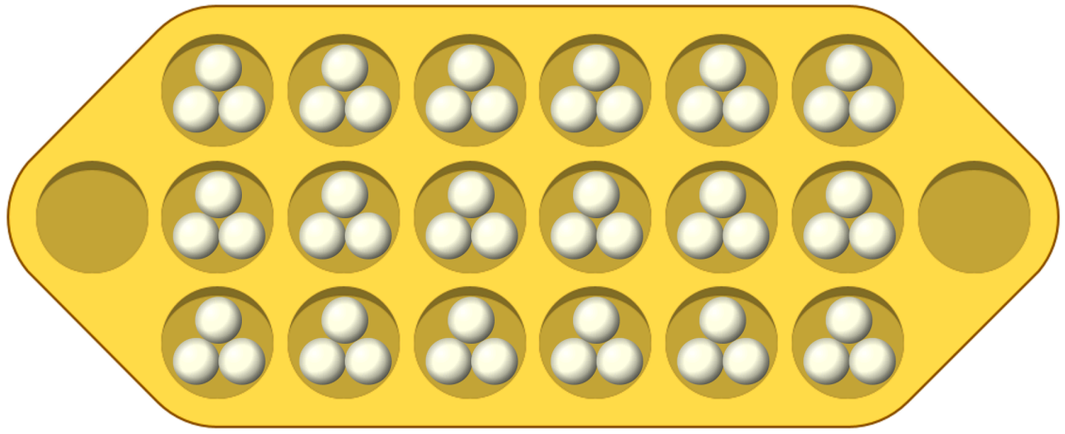
Decorate:

```
(metadata
   (graphics {(board Style Spiral)})
)
```

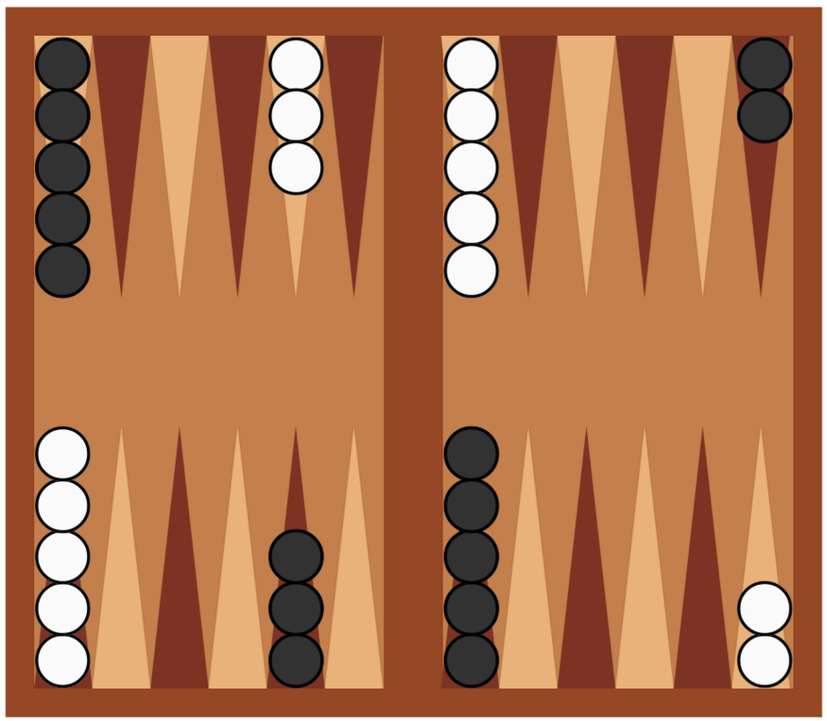


Graphics Metadata

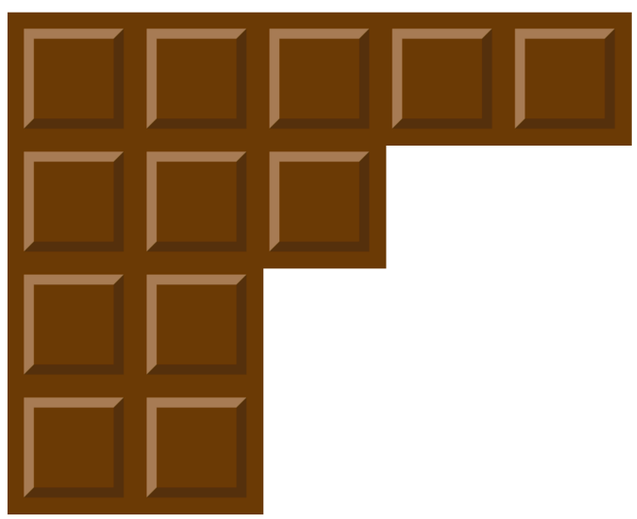
(graphics {(board Style Mancala)})



(graphics {
  (board Style backgammon)
  (stackType Backgammon)
})



(graphics {
  (player Colour Shared
    (colour 102 61 20))
  (no Board)
})



Graphics Metadata

Within game description (X.lud):

```
(game “X”
    // Game logic
)
(metadata
    (graphics
        // graphics hints
    )
)
```

Clear separation between game logic and graphics

Graphics metadata overrides:
1. Drawing style (board+components)
2. Behaviour

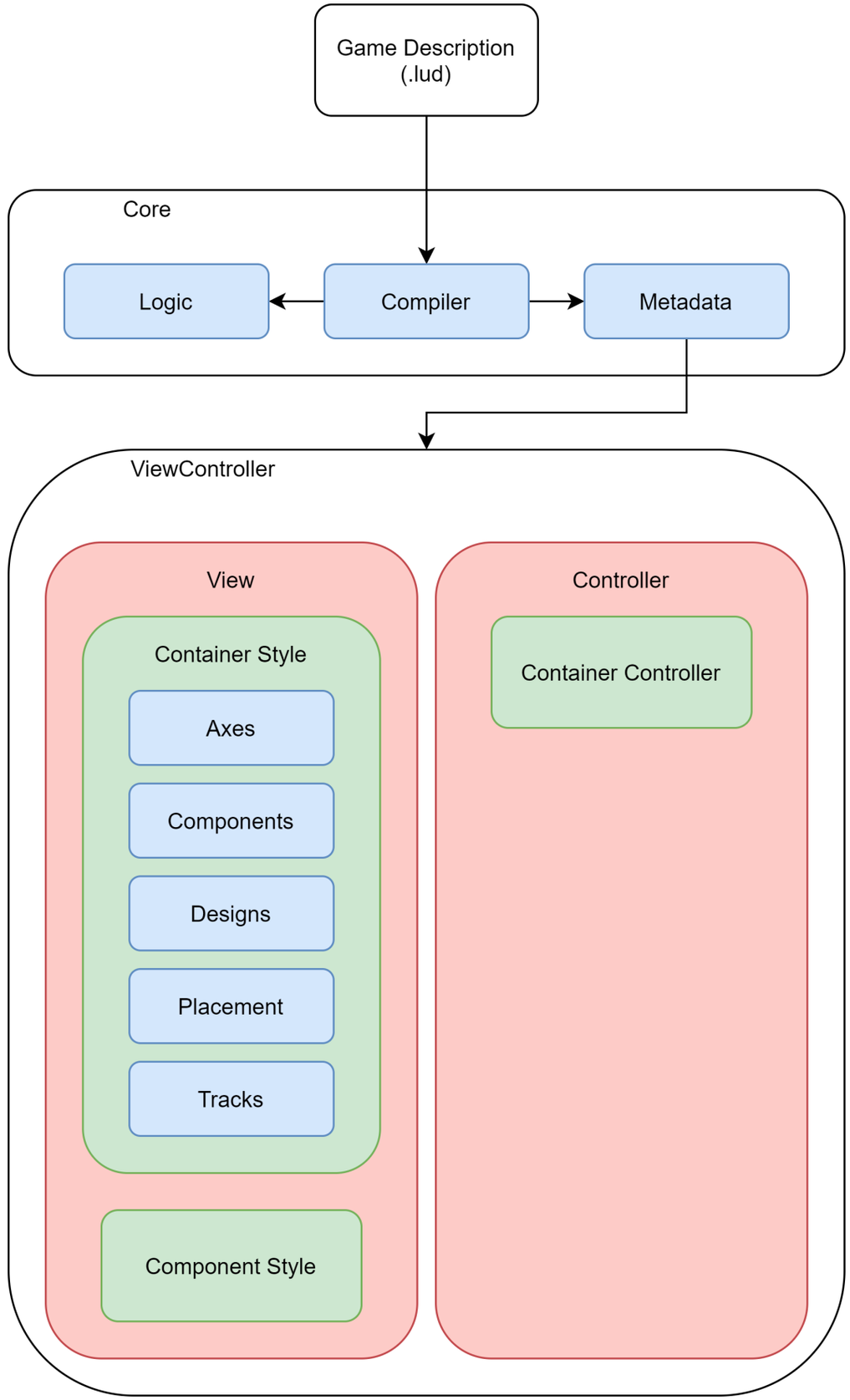
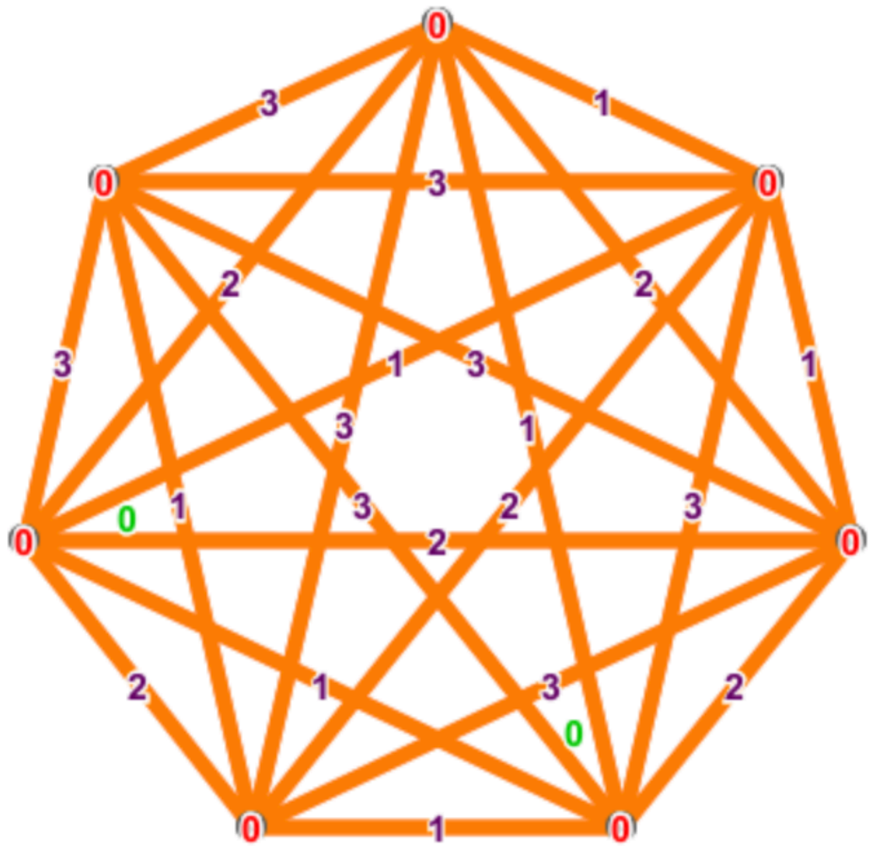


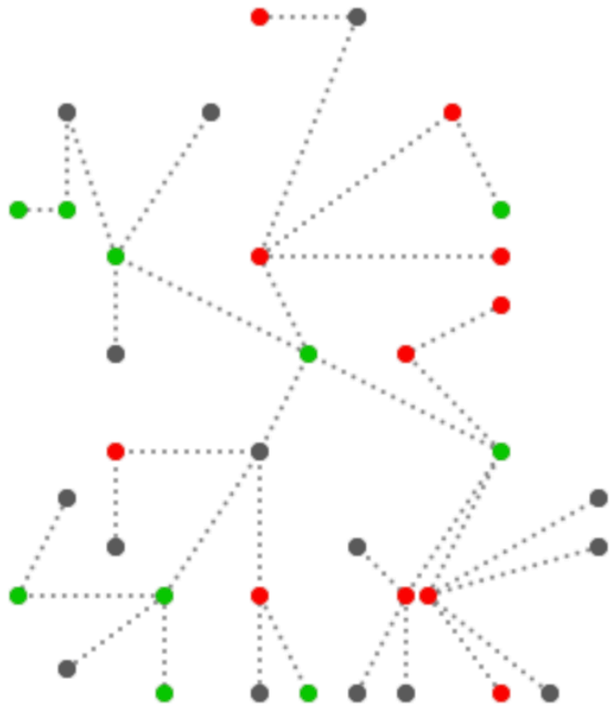
Diagram by Matthew Stephenson

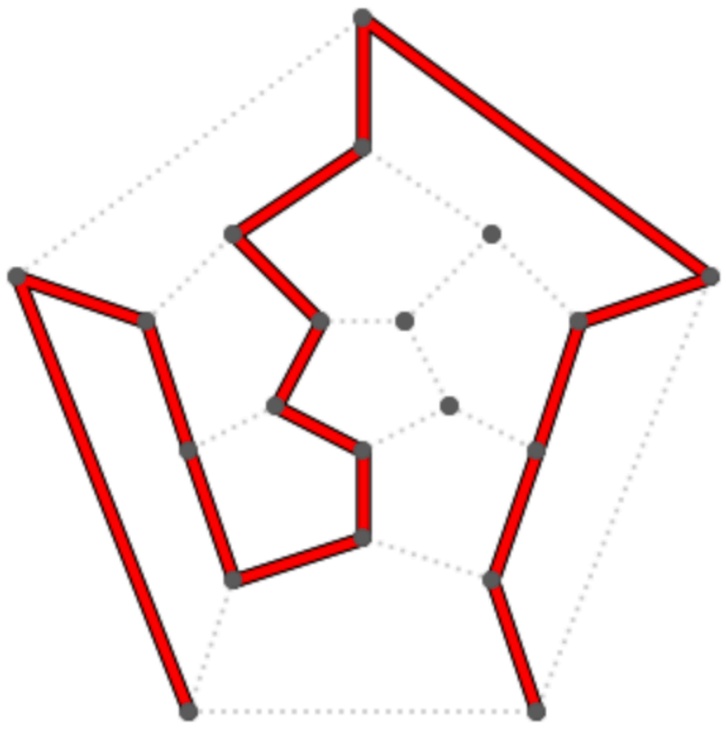
Graph Games

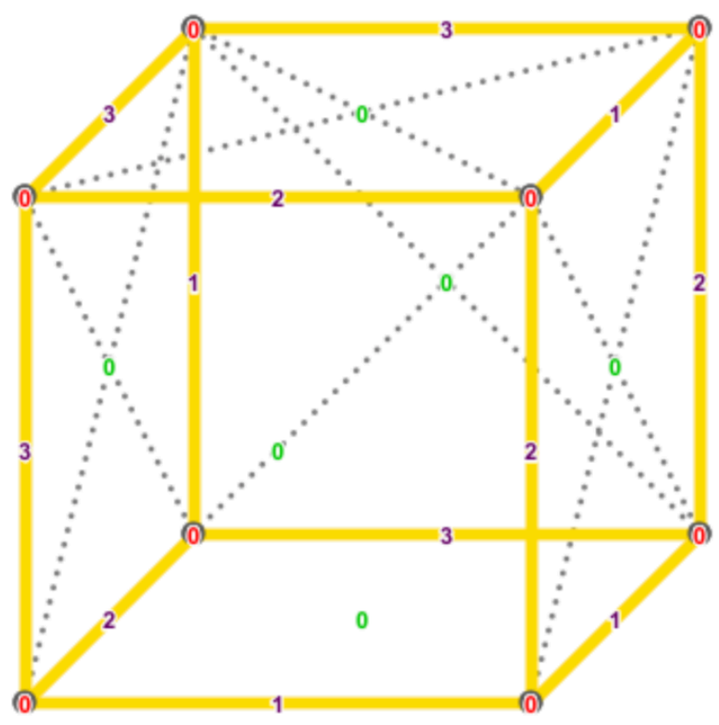
Tahmina Begum
- Masters thesis: *Graph Theory Games*

Graphs are immutable
- Site ownership
- Vertex/edge/cell colouring
- No components (pieces)





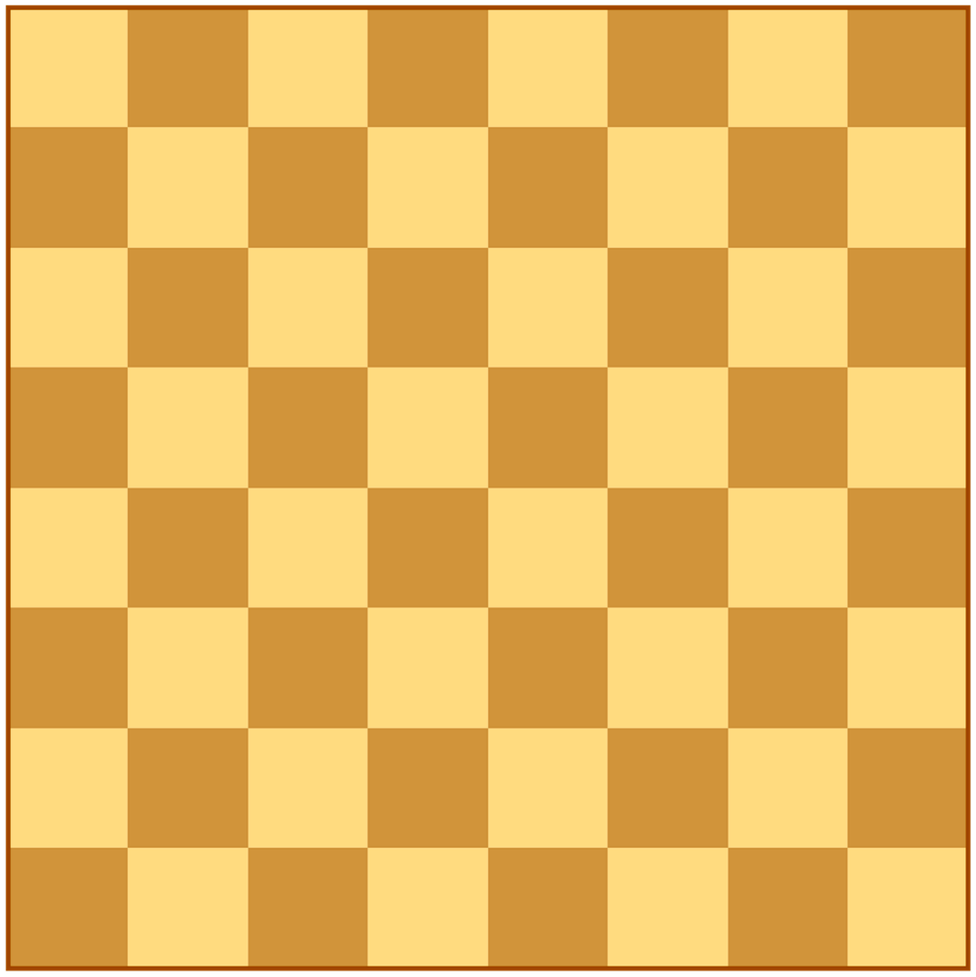


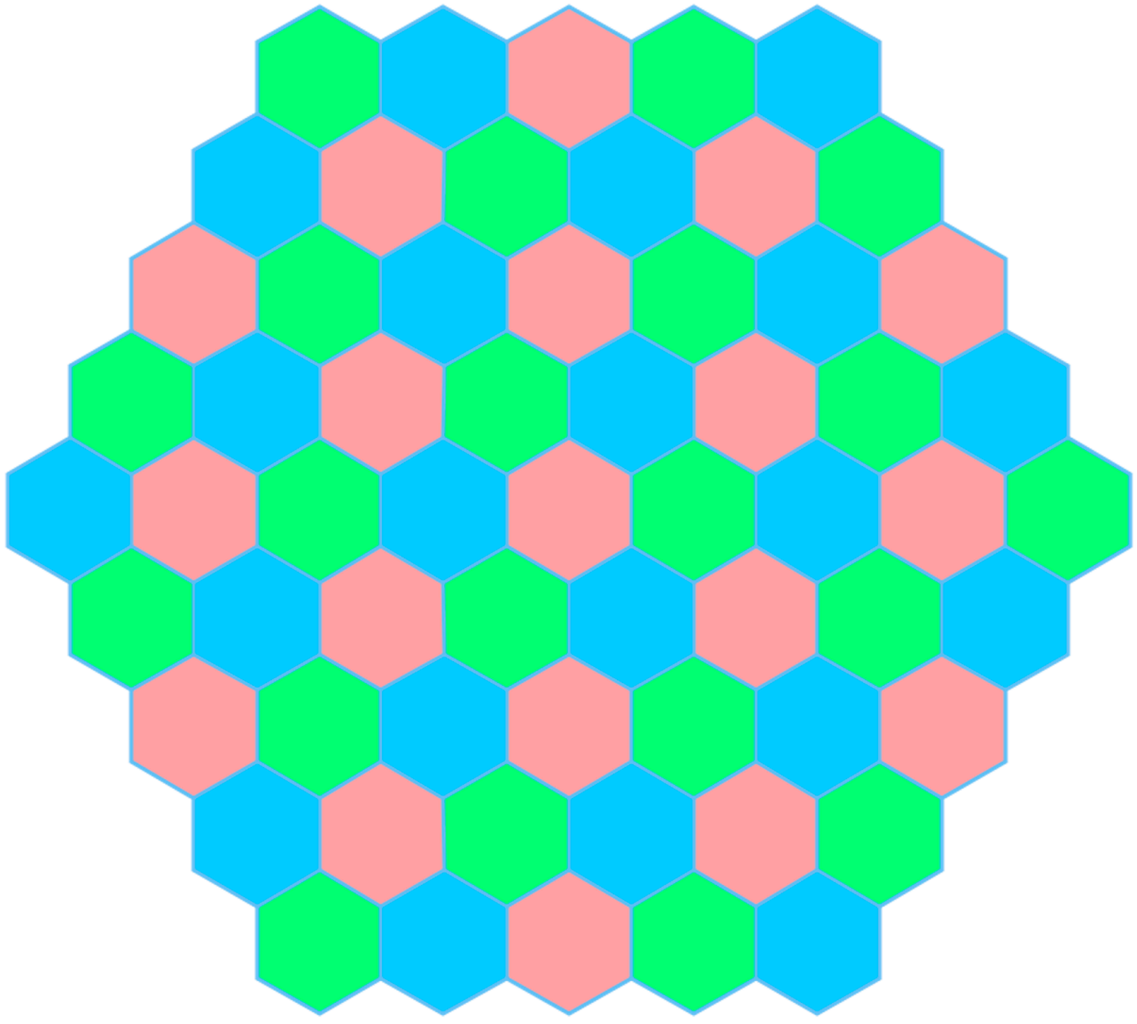


Cell Phases

Face Colourings:

- Colour each face 1..*N* such that no same-coloured faces share an edge
- Typically visual guide (Chess) but can also affect play (Triad)



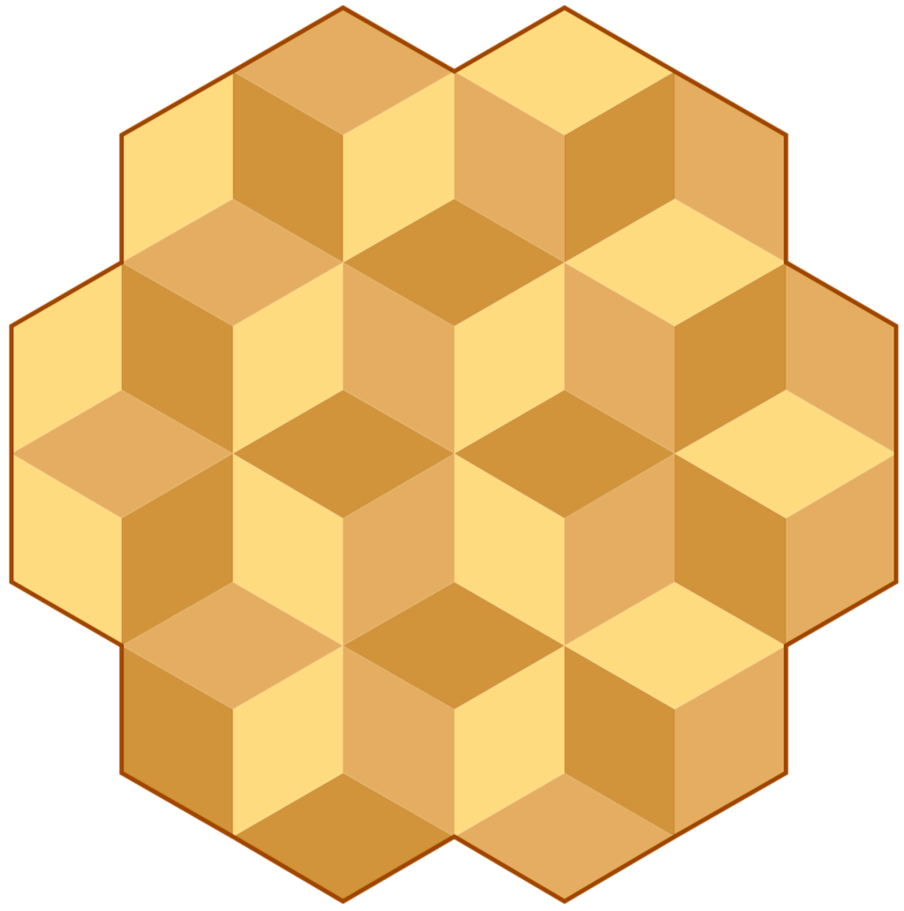


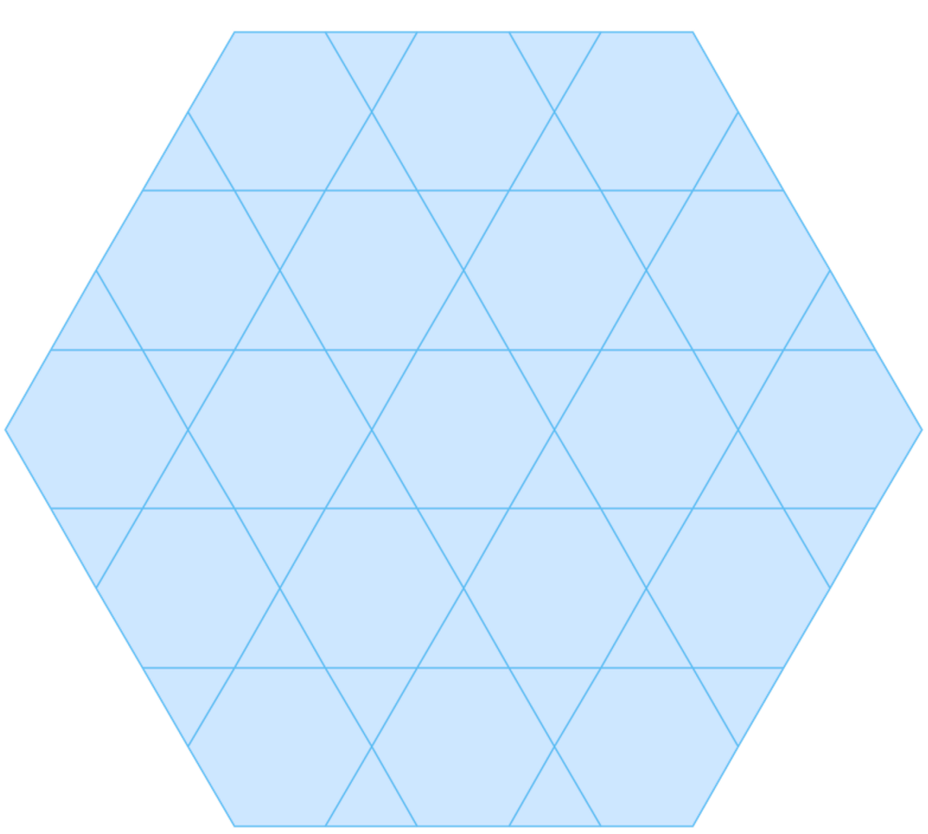
Cell Phases

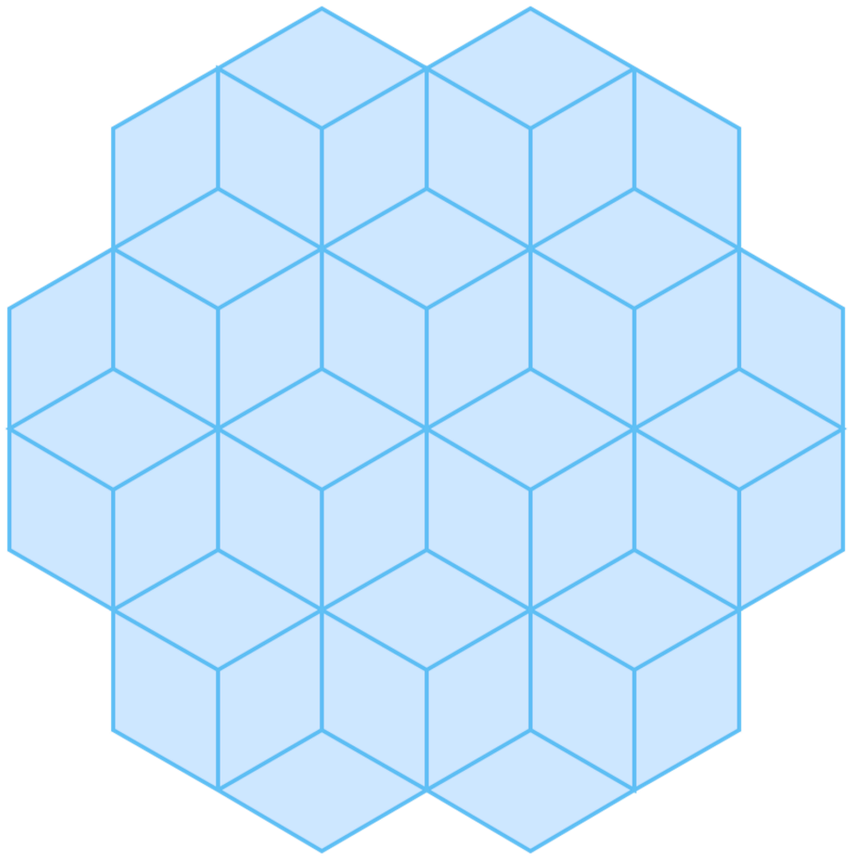
Problem:

- Symmetry is generally assumed
  e.g. 3.6.3.6 dual technically
  “correct” but not ideal

(dual (tiling T3636 3))



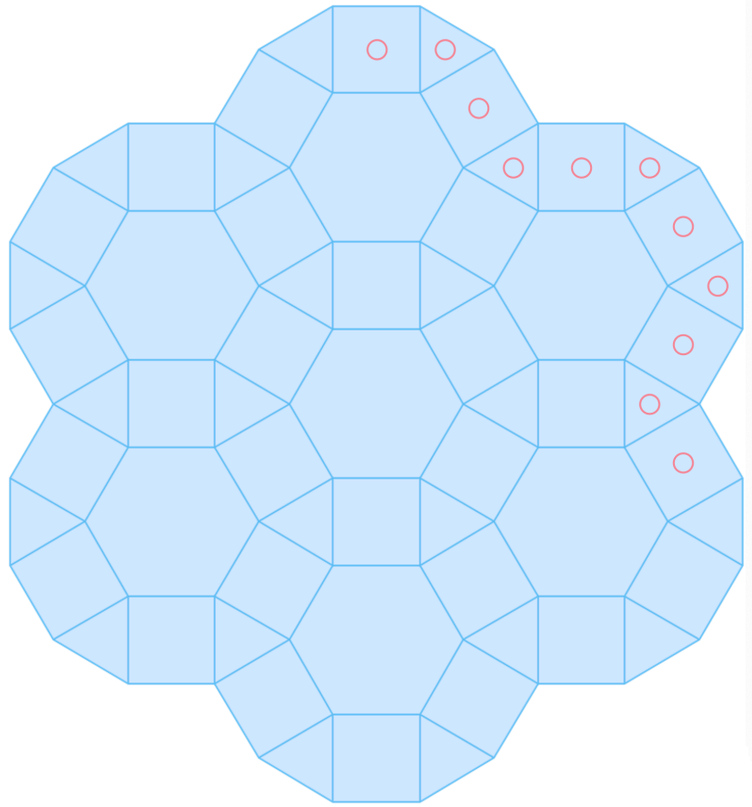


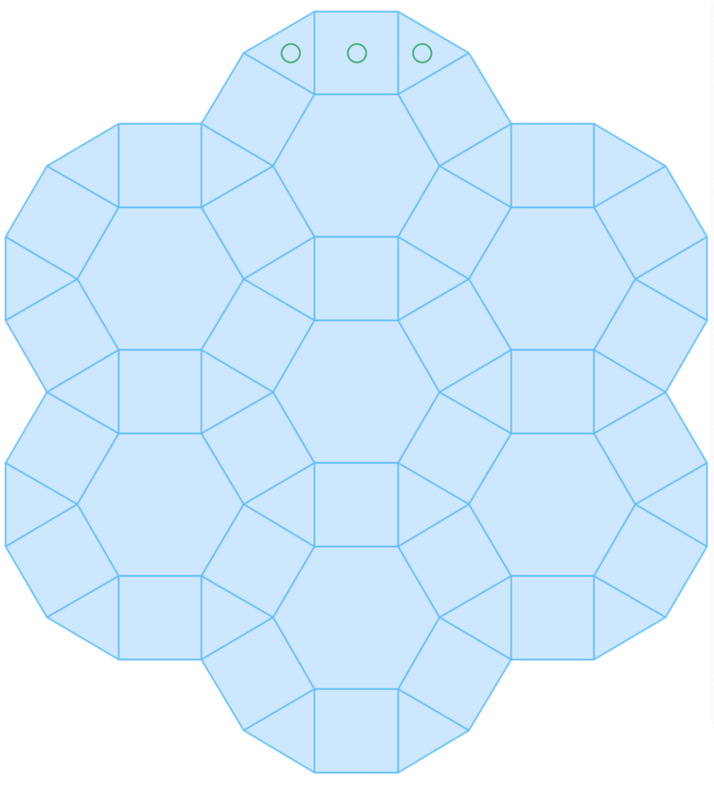


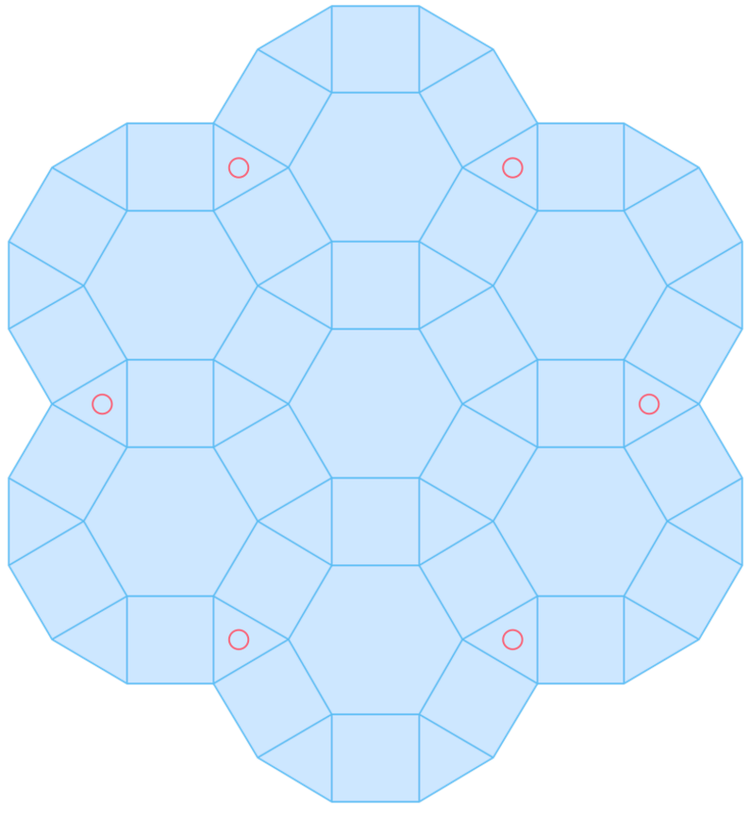
Regions

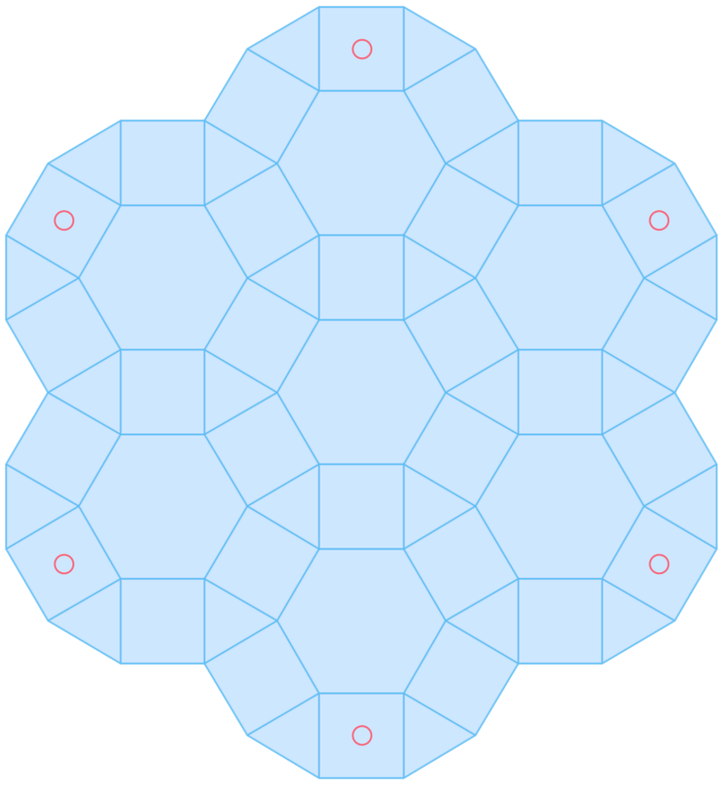
Region : a set of sites

1. *User defined* in the game logic
   e.g. goal regions
2. *Pre-defined* from the graph
   e.g. top, corners (concave),
   corners (convex), NE side, etc.







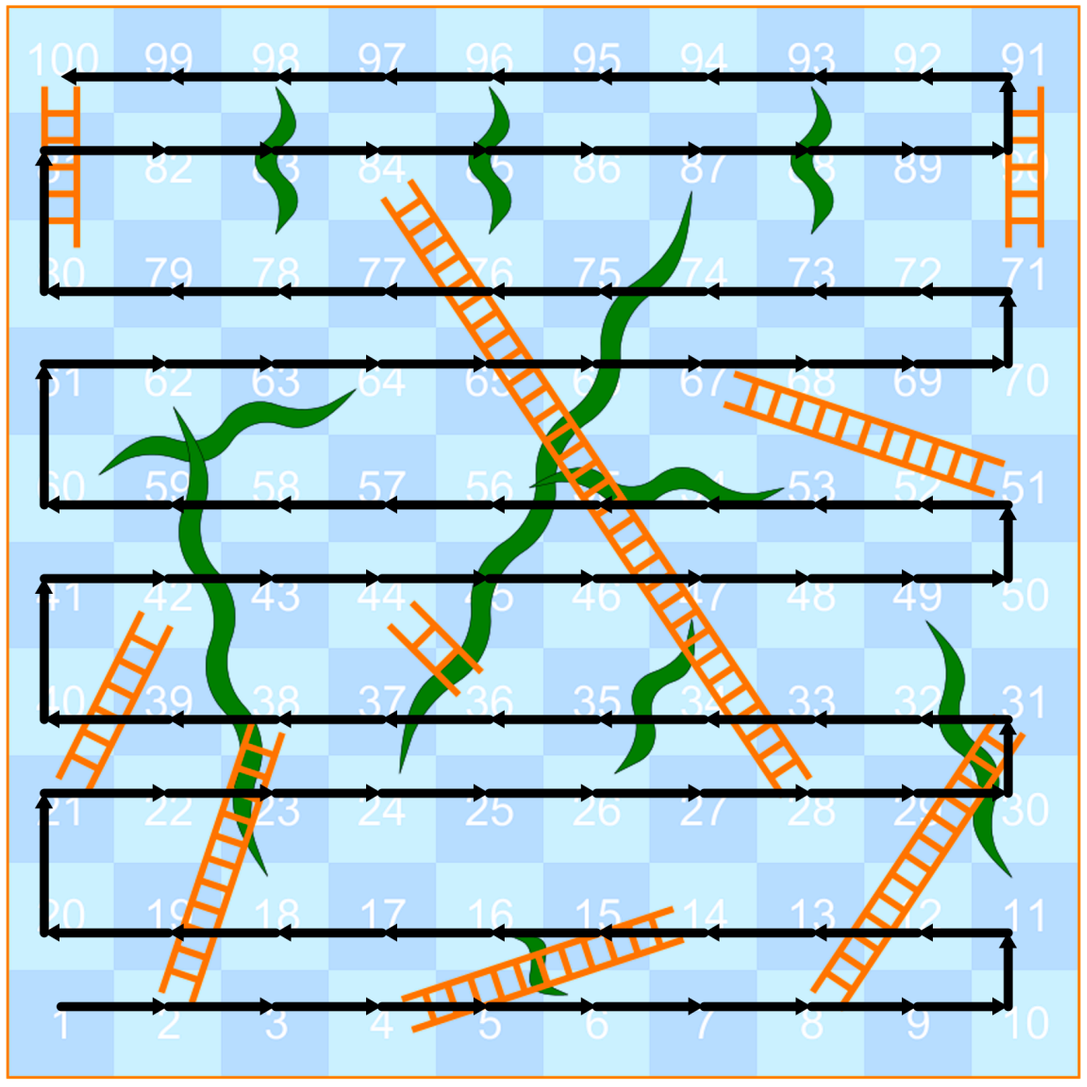


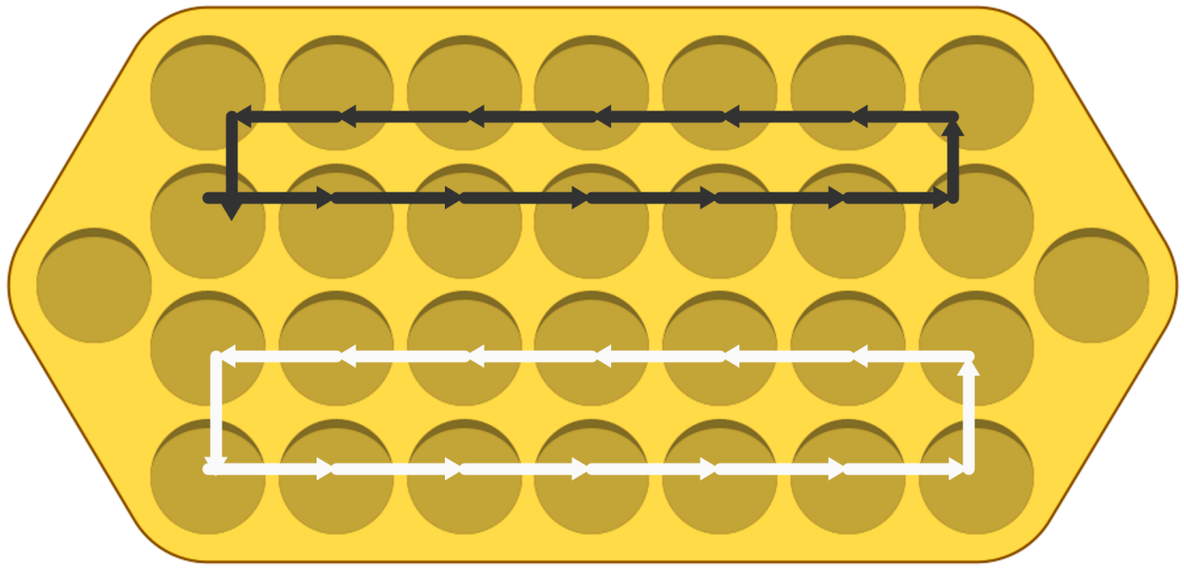
Tracks

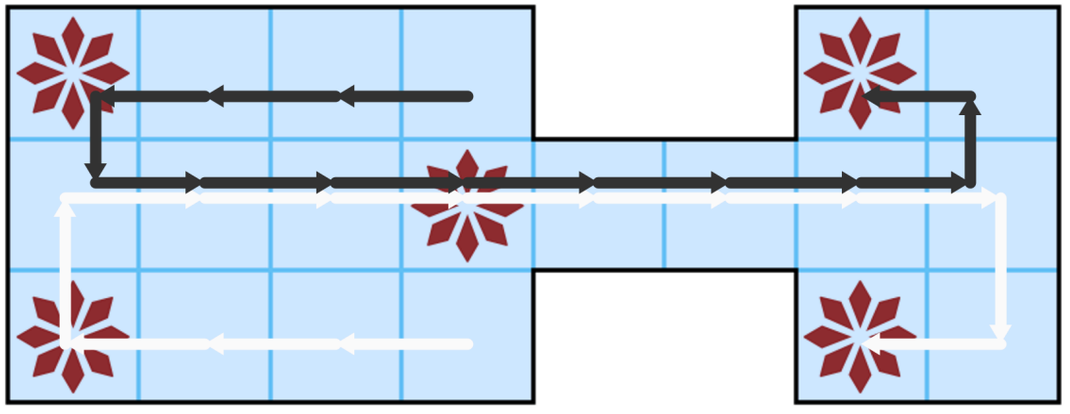
Track : ordered list of sites

User defined in the game logic

Can be assigned per player or shared (e.g. boustrophedon)



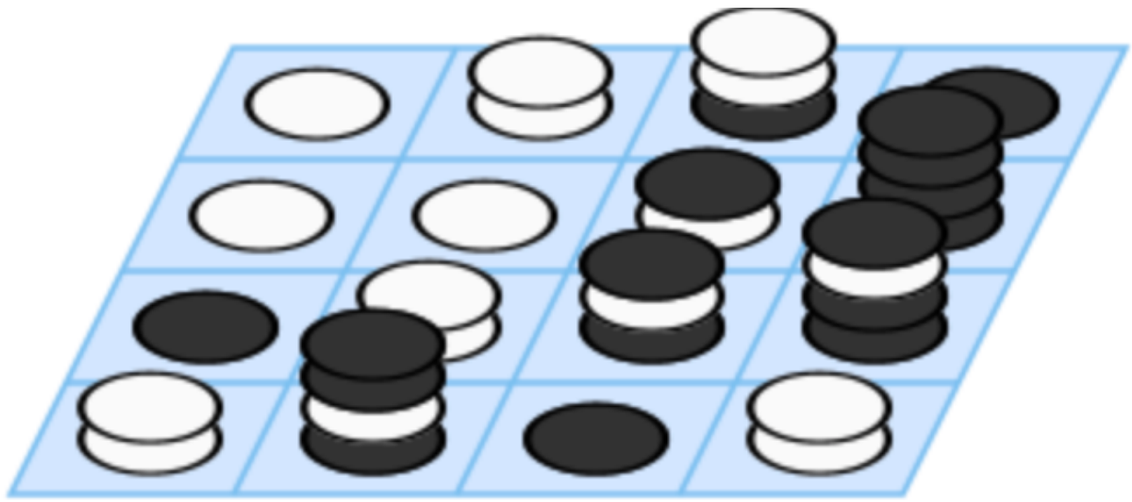




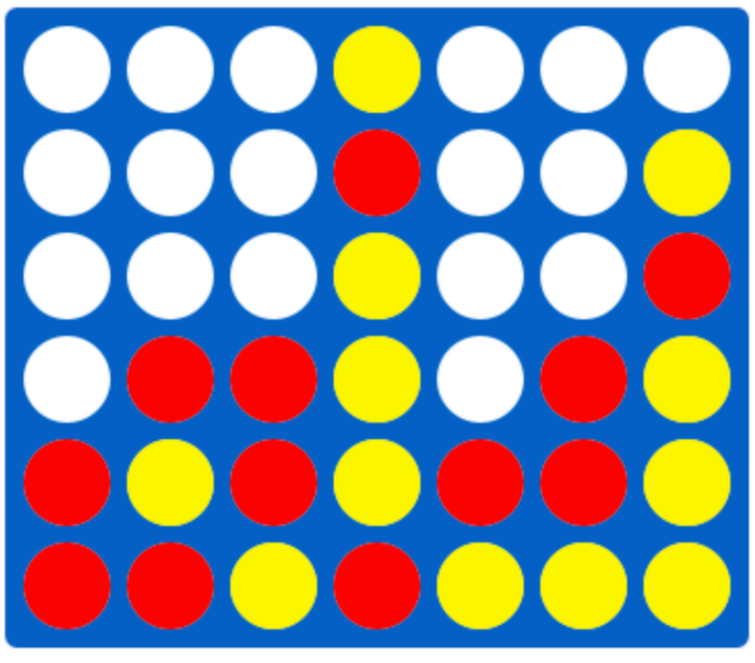
3D Boards

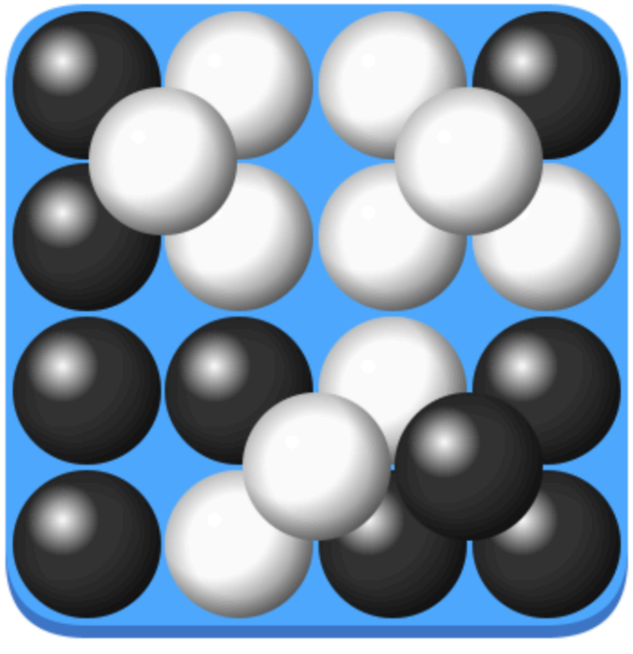
Simulated 3D:

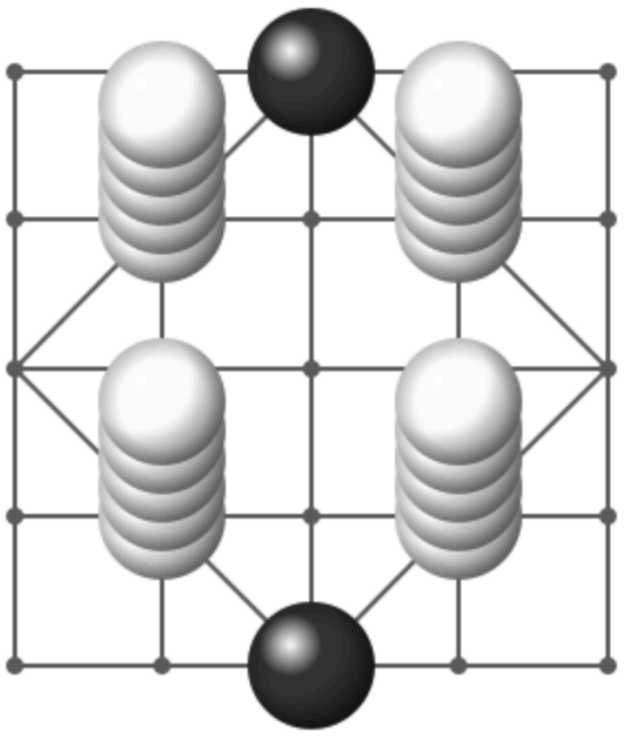
- Side view
- Offset stacking
- Overhead view
- Oblique view



Full 3D support to come (GUI)



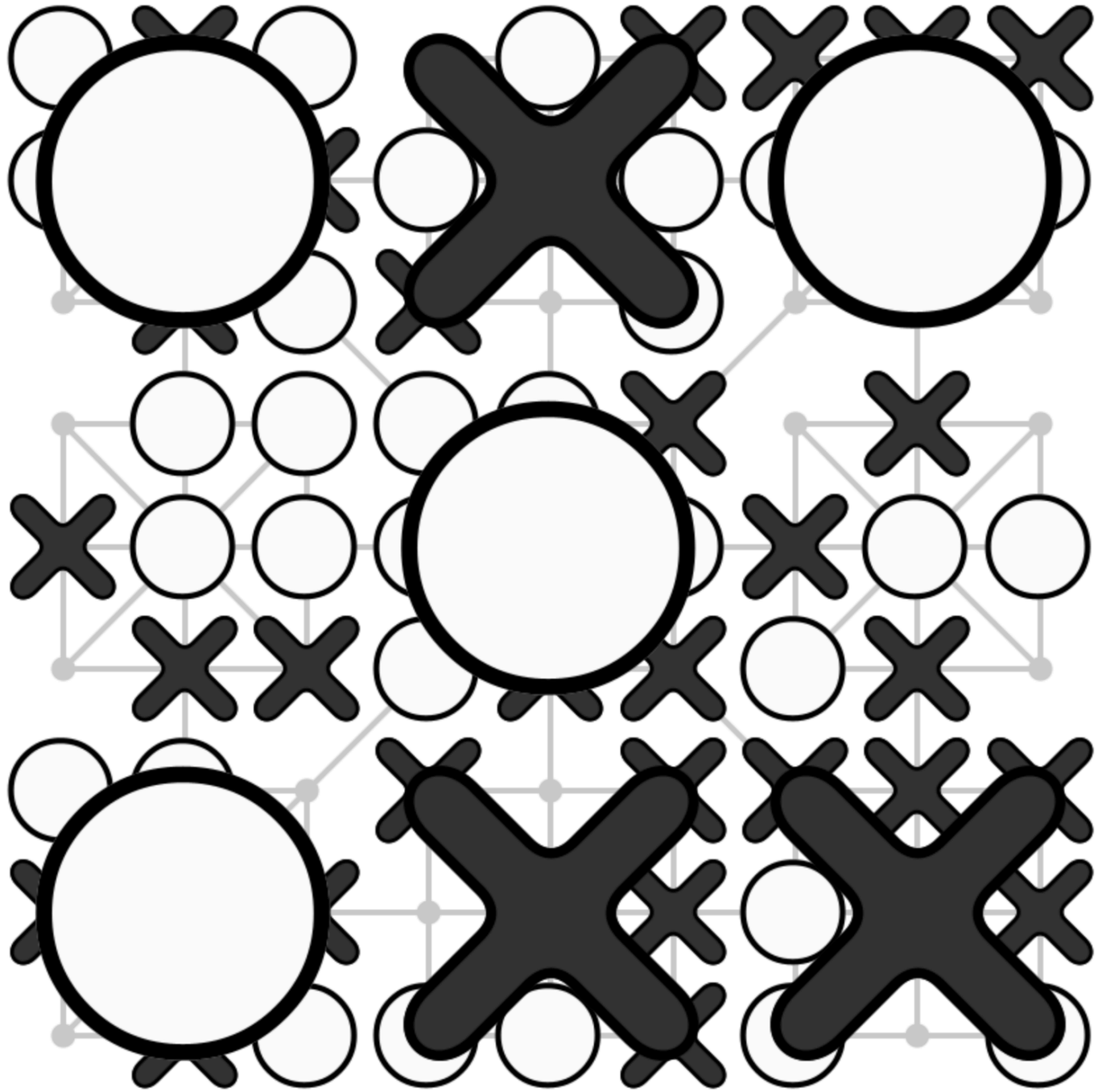




Shared Boards

Ultimate Tic-Tac-Toe:
- Nine 3x3 sub-games
- One 3x3 supergame

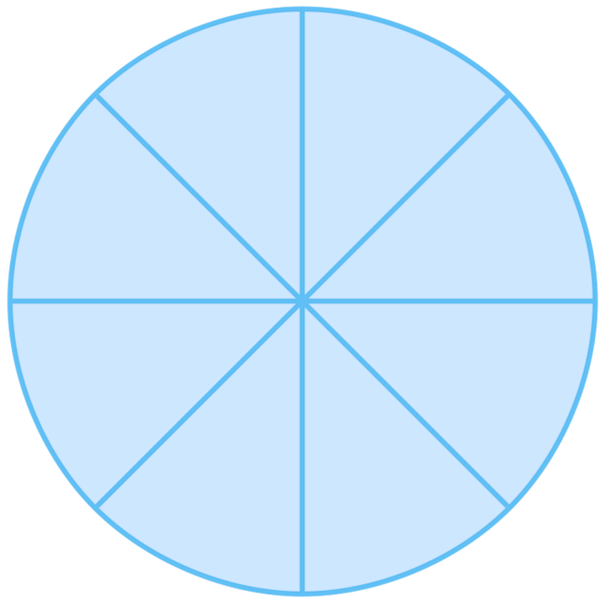
Modelled as a single graph:
- Sites 0..80 = sub-games
- Sites 81..88 = supergame

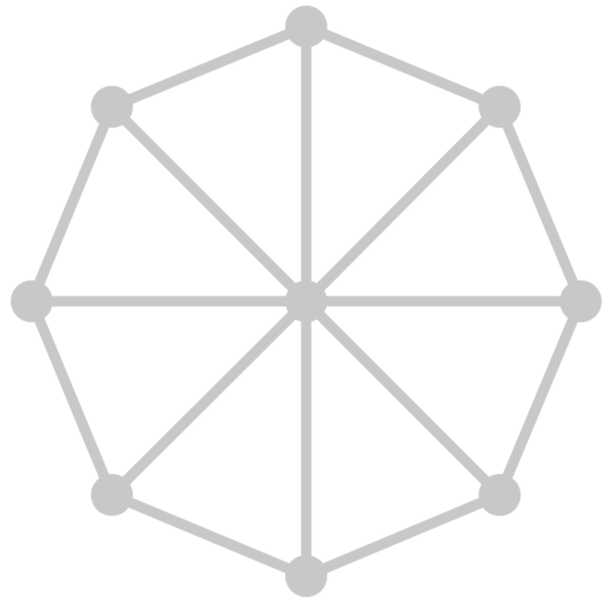


Circular Boards

(board (circle {8}))

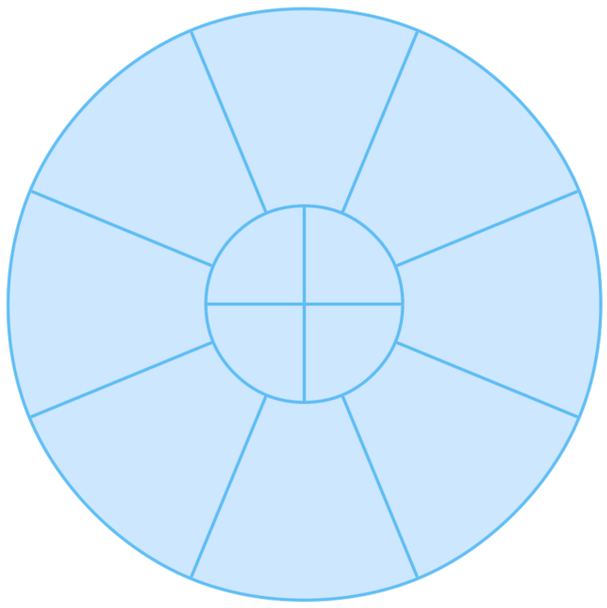
(board (circle {8}) use:Vertex)

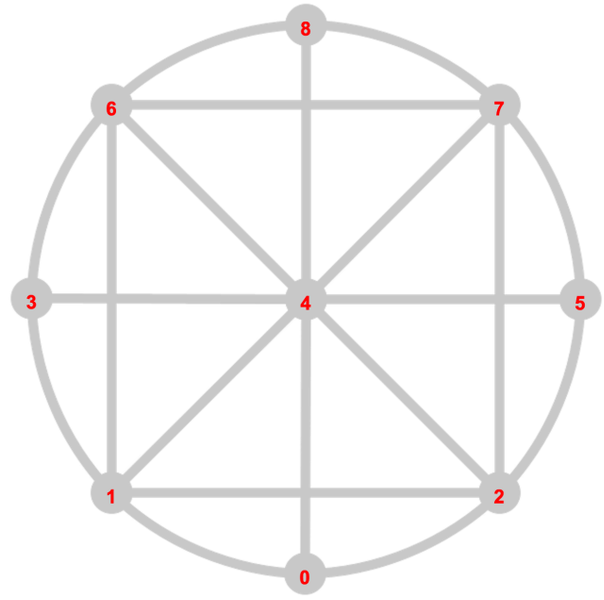




(board (circle {4 8} stagger:true)

```
(board
   (splitCrossings
      (merge
         (shift .5 .5 (scale 1.42
            (circle {8}))) (square 2)
      )
   )
   use:Vertex
)
```

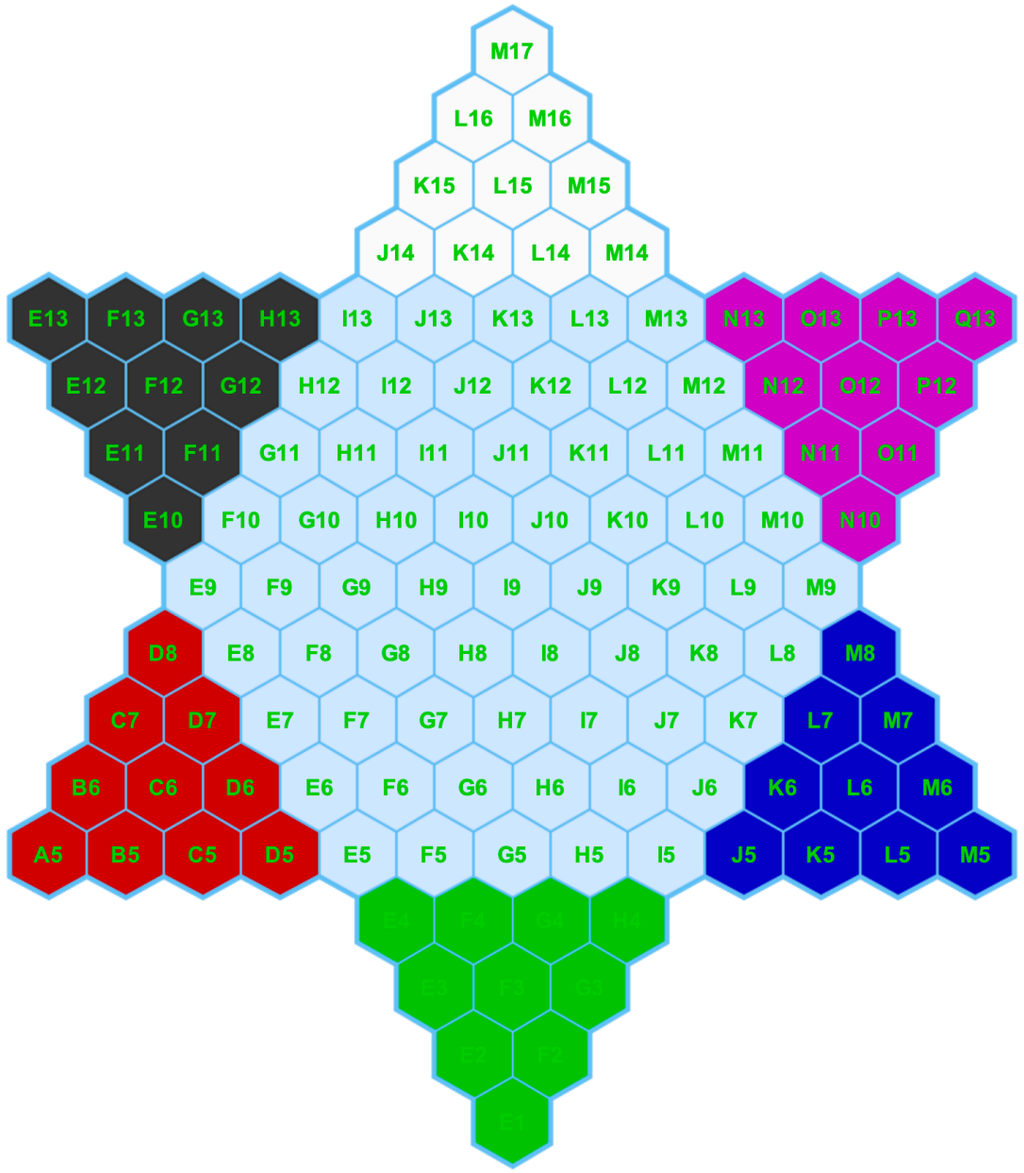


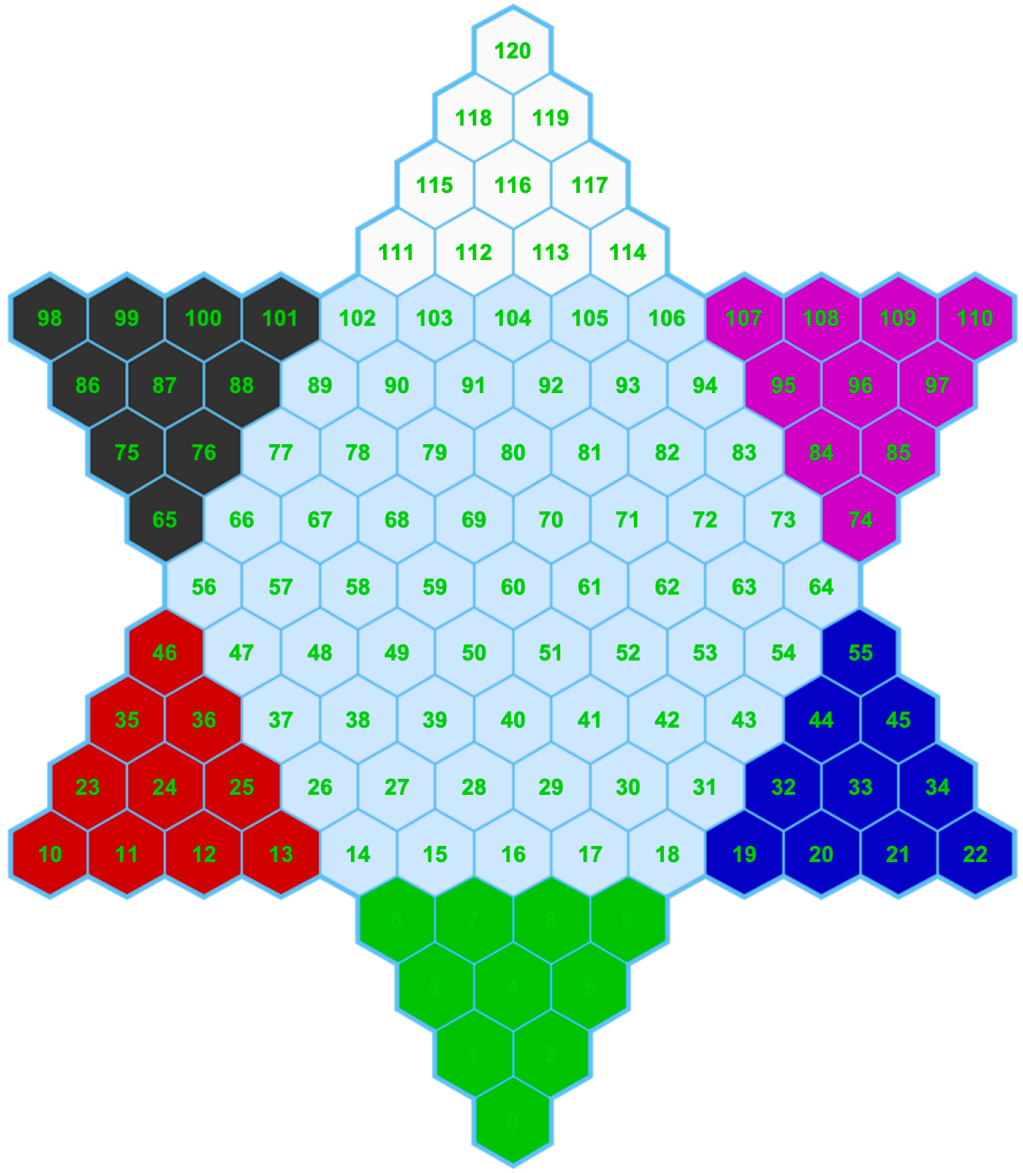


Coordinates

Policy to use coordinate labels to identify sites (not indices)

- e.g. Blue aims for E13 (not cell 98)
- More human readable





Coordinates

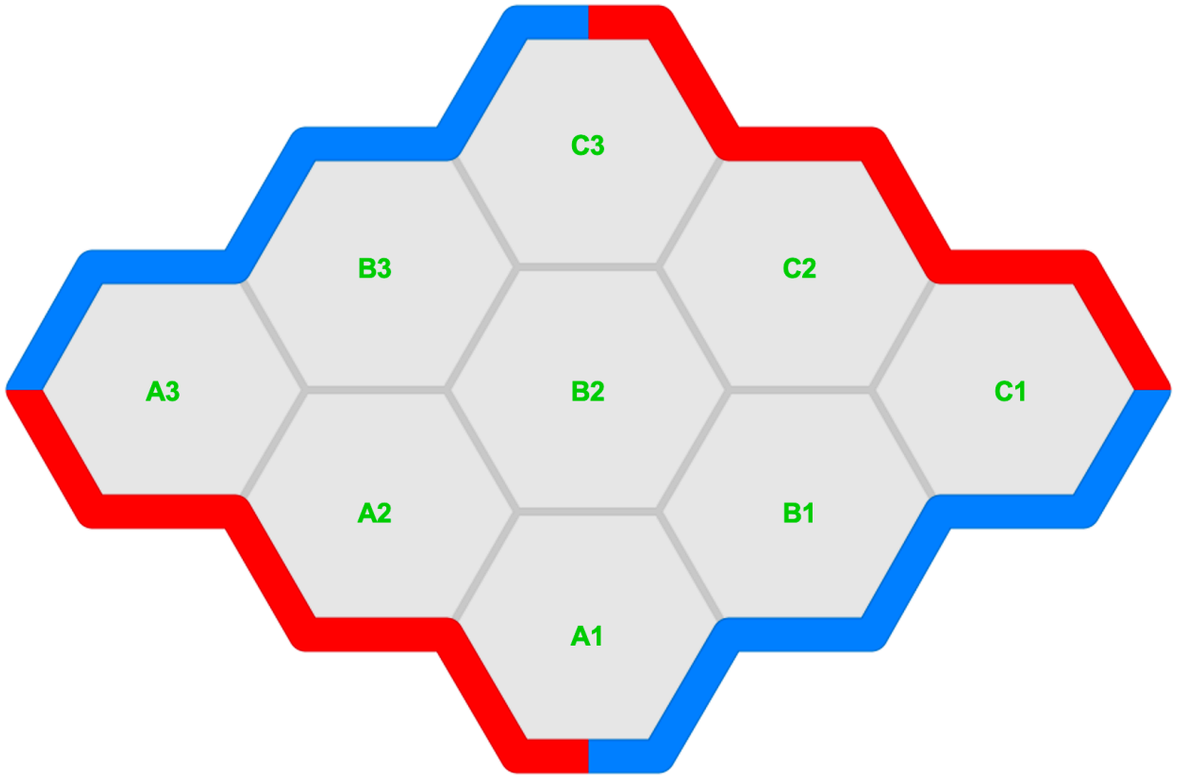
Coordinate generated using a general approachs:

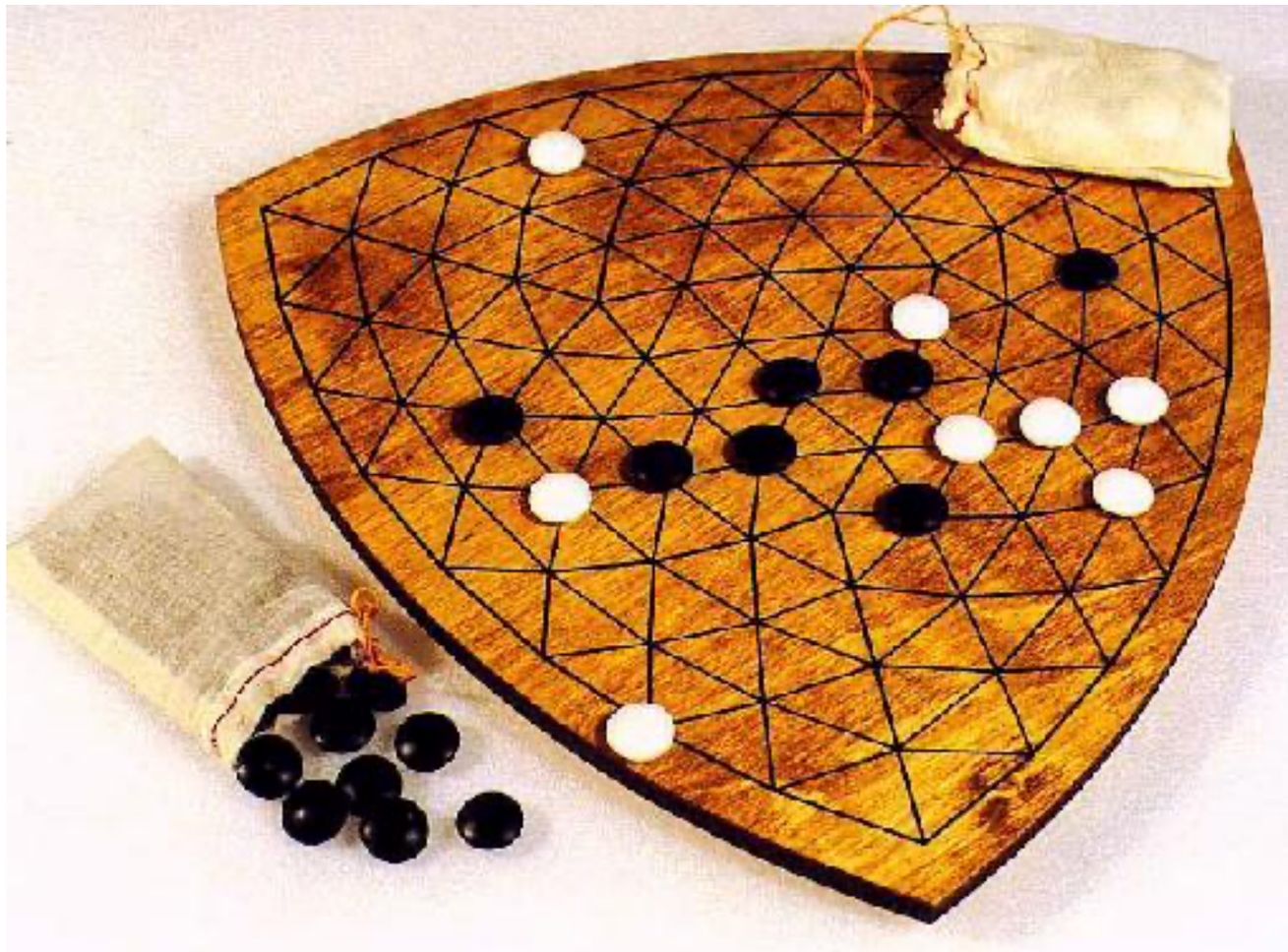
1. Find two principle axes (approx. orthogonal) that minimise clustering error

2. Cluster elements by projection onto each axis

Works for most cases
…not so well for others

Failsafe guarantees unique coords based on X and Y posn





Limits

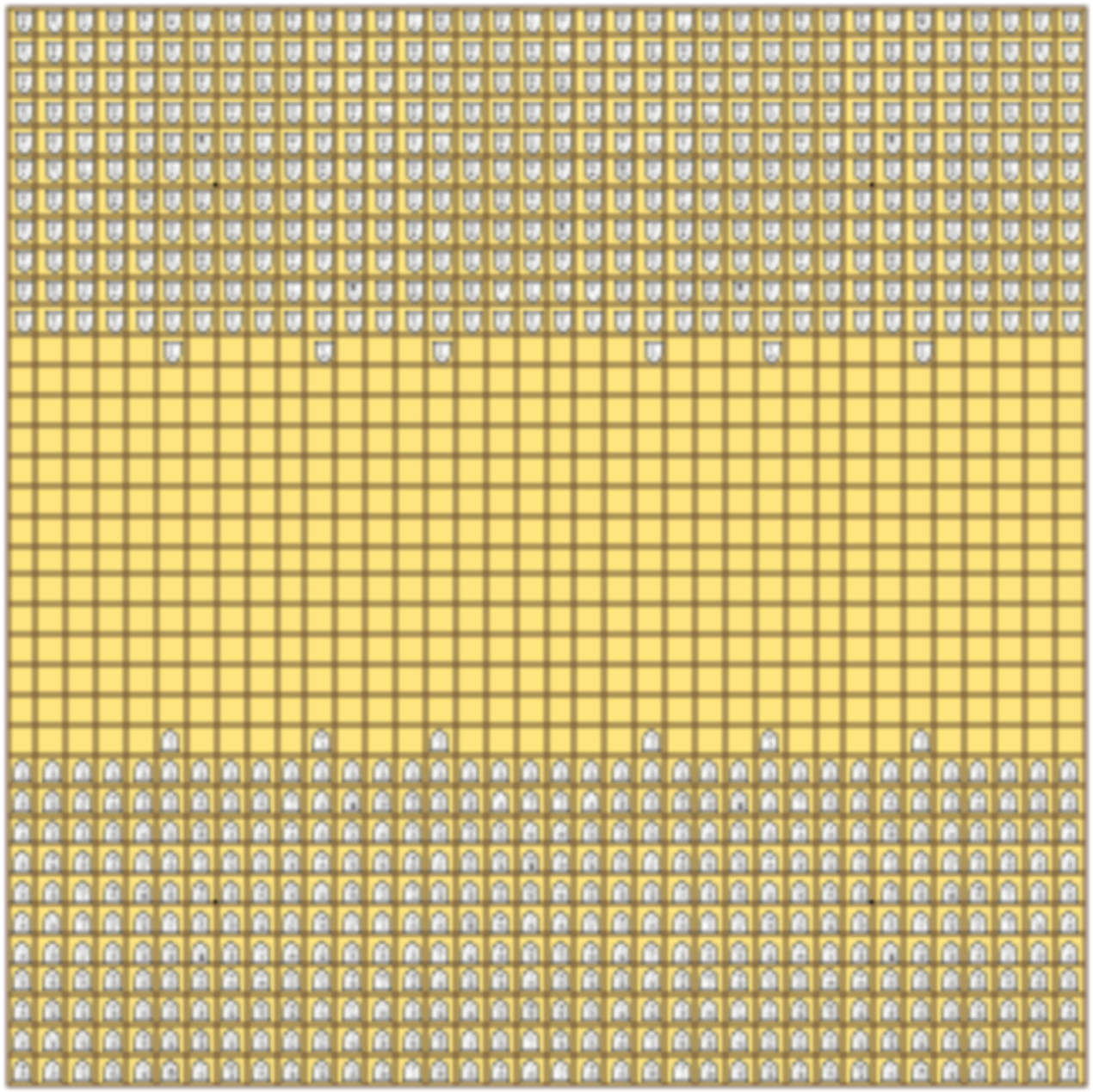
Ludii contraints:

- Maximum 8 dimensions

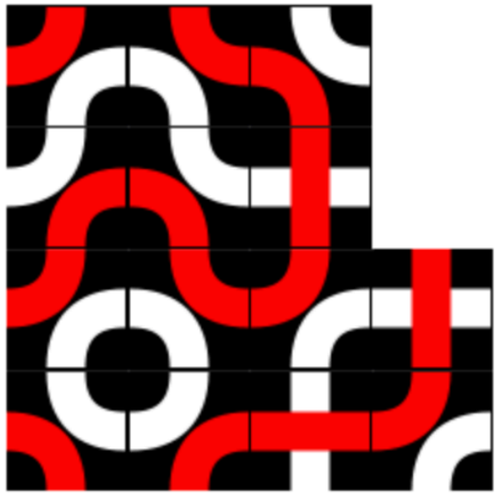
No other size limit, except:

- Memory
- Time

e.g. 36x36 board for Taikyoku Shogi

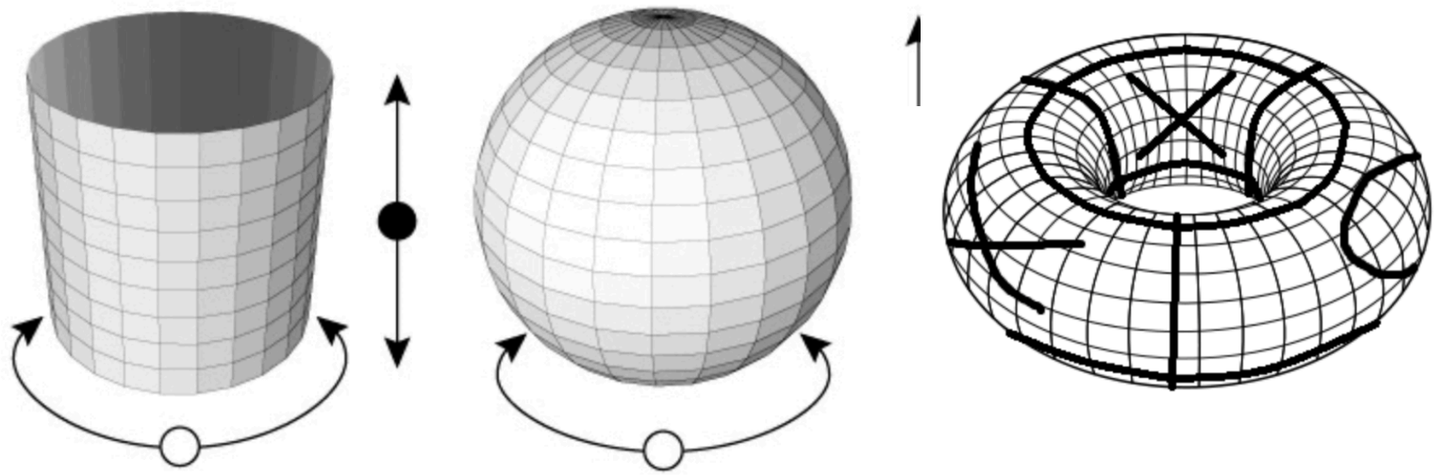


Future Work



Improve boardless games:
- Supported using virtual boards
- Inefficient

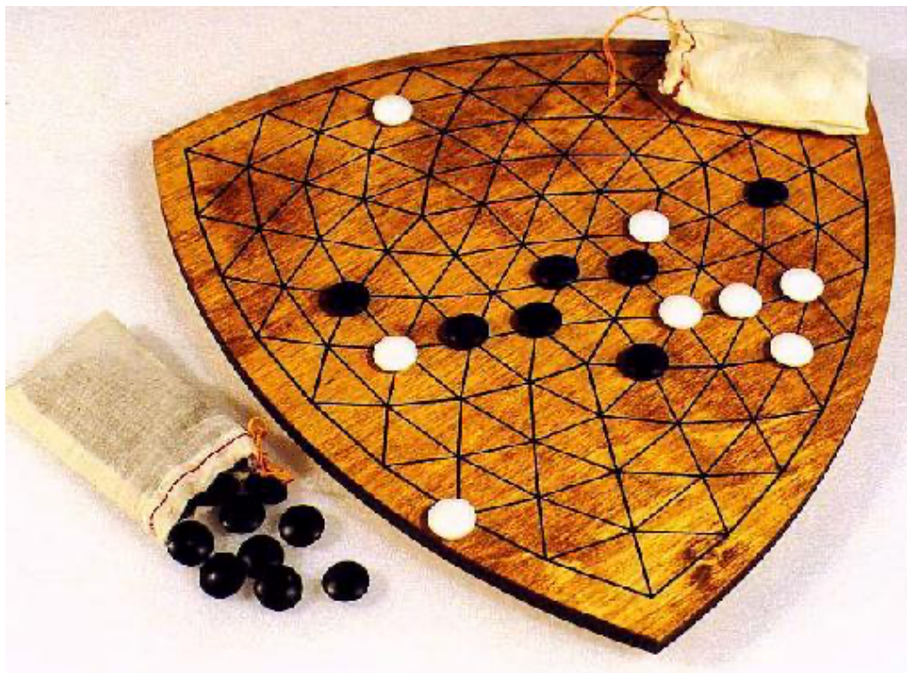
Wraparound boards:
- Cylinder
- Sphere
- Torus
- Projective plane



Better general coordinates:
- Based on contours (active snakes?)

Irregular and branching radials

Full support for 3D game boards



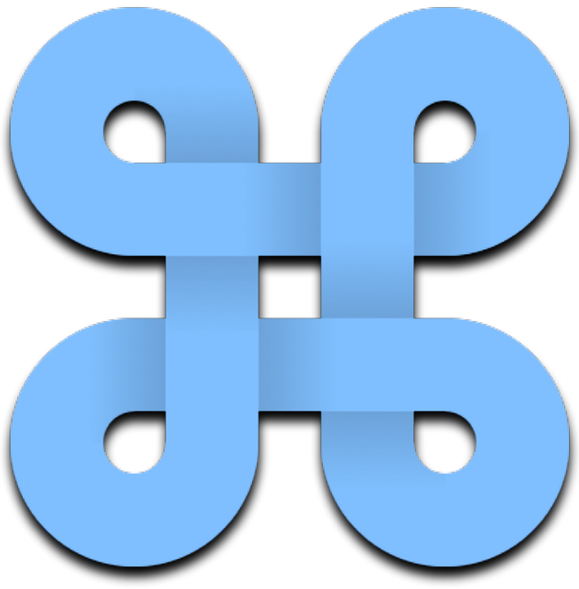
Conclusion

General board representation:

- Difficult task, many edge cases
- Largely achieved in Ludii
- A lot of work!

European Research Council

Established by the European Commission



http://ludii.games

Digital Ludeme Project

http://ludeme.eu