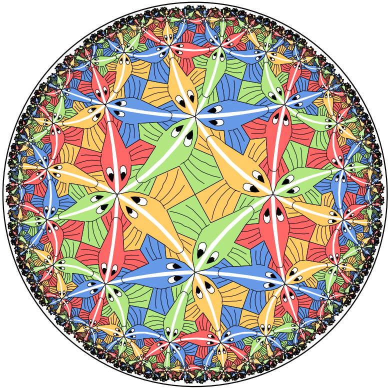
SubTile 2013, Marseille

An Algorithm to Create Hyperbolic Escher Tilings

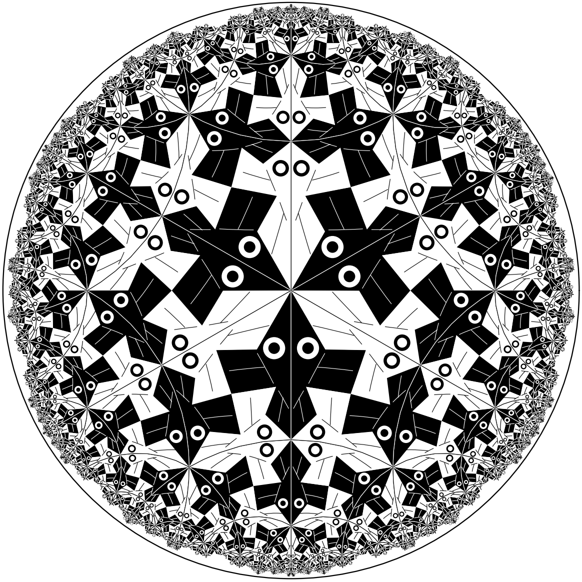
Douglas Dunham
University of Minnesota Duluth
Duluth, Minnesota USA
Email: ddunham@d.umn.edu
Web: http://www.d.umn.edu/~ddunham



Outline

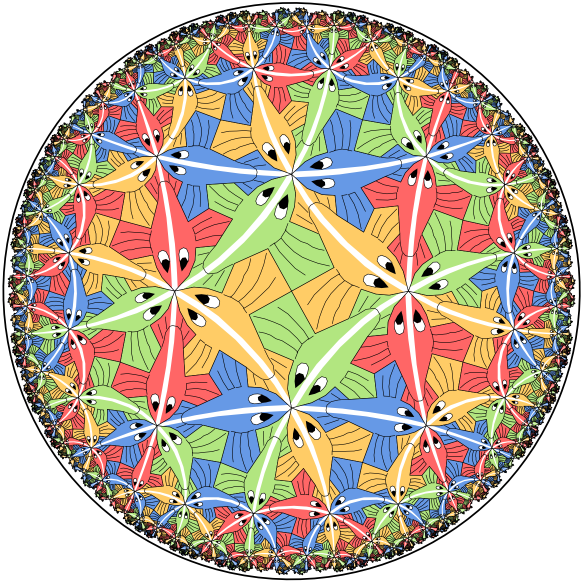
- Motivation — M.C. Escher examples
- Hyperbolic geometry, Repeating patterns, and regular tessellations
- The replication algorithm
- Other hyperbolic patterns inspired by Escher patterns
- Future research

Hyperbolic Art Pioneer: M.C. Escher
Four “Circle Limit” Patterns: Circle Limit I

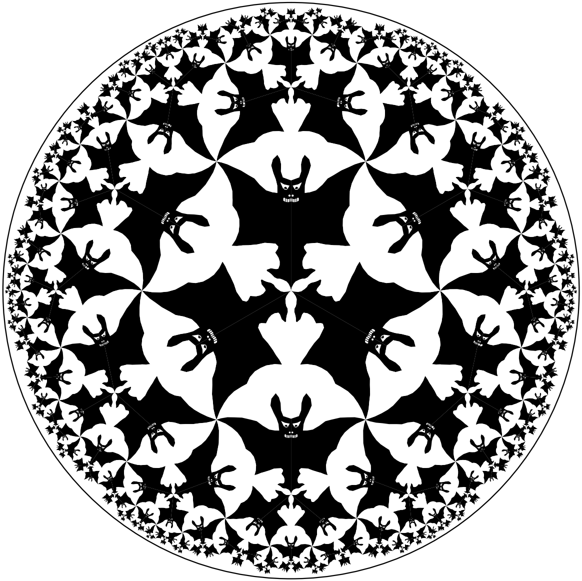


Circle Limit II

Circle Limit III



Circle Limit IV

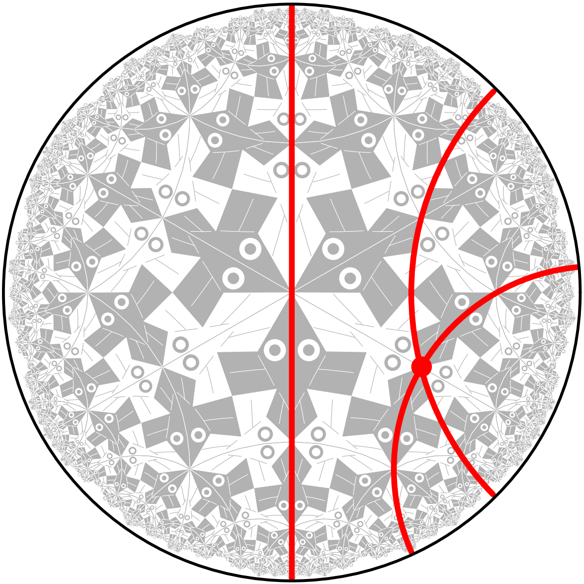


Creating Repeating Hyperbolic Patterns

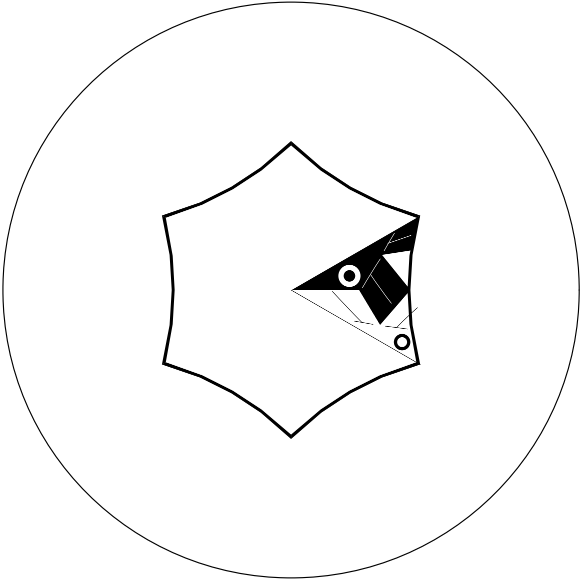
A two-step process:

1. **Design the fundamental tile or motif**
2. **Transform copies of the tile about the hyperbolic plane: replication**

Poincaré Disk Model of Hyperbolic Geometry



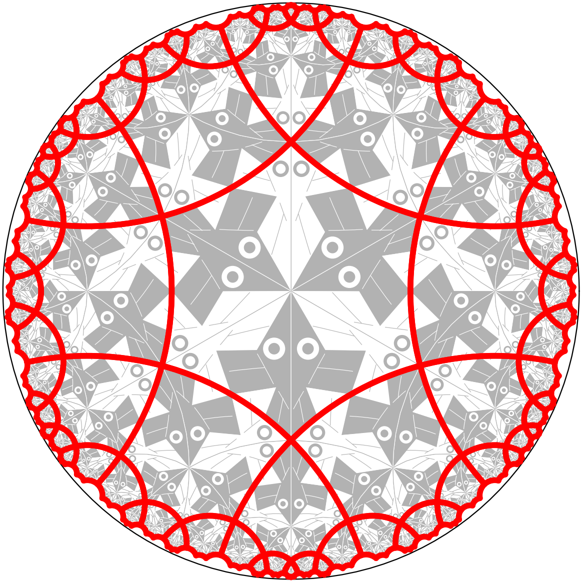
Repeating Patterns
A repeating pattern is composed of congruent copies of the motif.



The Regular Tessellations $\{p, q\}$

- **The regular tessellation $\{p, q\}$ is a tiling composed of regular p-sided polygons, or p-gons meeting q at each vertex.**
- **It is necessary that $(p-2)(q-2) > 4$ for the tessellation to be hyperbolic.**
- **If $(p-2)(q-2) = 4$ or $(p-2)(q-2) < 4$ the tessellation is Euclidean or spherical respectively.**

The Regular Tessellation $\{6, 4\}$



A Table of the Regular Tessellations

q \ p	3	4	5	6	7	8	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
8	*	*	*	*	*	*	⋯
7	*	*	*	*	*	*	⋯
6	□	*	*	*	*	*	⋯
5	○	*	*	*	*	*	⋯
4	○	□	*	*	*	*	⋯
3	○	○	○	□	*	*	⋯

□ - Euclidean tessellations

○ - spherical tessellations

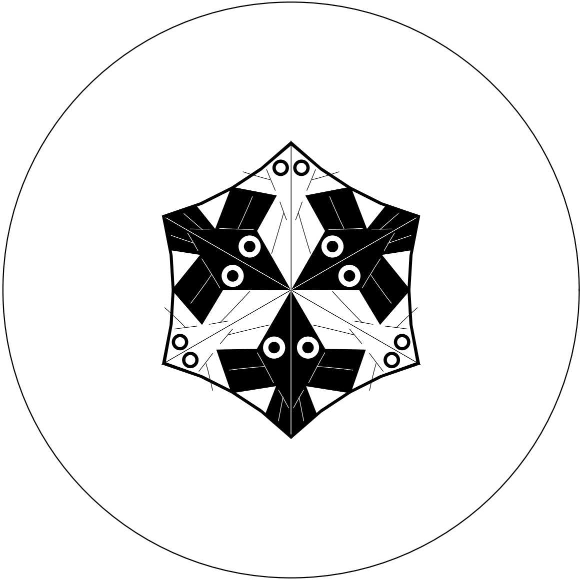
* - hyperbolic tessellations

The Replication Algorithm

To reduce the number of transformations and to simplify the replication process, we form the p-gon pattern from all the copies of the motif touching the center of the bounding circle.

- **Thus in order to replicate the pattern, we apply transformations to the p-gon pattern rather than to each individual motif.**
- **Some parts of the p-gon pattern may protrude from the enclosing p-gon, as long as there are corresponding indentations, so that the final pattern will fit together like a jigsaw puzzle.**
- **The p-gon pattern is often called the translation unit in repeating Euclidean patterns.**

The p-gon pattern for Circle Limit I



Layers of p-gons

We note that the p-gons of a $\{p, q\}$ tessellation are arranged in layers as follows:

- **The first layer is just the central p-gon.**
- **The $k + 1^{st}$ layer consists of all p-gons sharing and edge or a vertex with a p-gon in the k^{th} layer (and no previous layers).**
- **Theoretically a repeating hyperbolic pattern has an infinite number of layers, however if we only replicate a small number of layers, this is usually enough to appear to fill the bounding circle to our Euclidean eyes.**

Exposure of a p-gon

We also define the exposure of a p-gon in terms of the number of edges it has in common with the next layer (and thus the fewest edges in common with the previous layer).

- **A p-gon has maximum exposure if it has the most edges in common with the next layer, and thus only shares a vertex with the previous layer.**
- **A p-gon has minimum exposure if it has the least edges in common with the next layer, and thus shares an edge with the previous layer.**
- **We abbreviate these values as MAX_EXP and MIN_EXP respectively.**

The Replication Algorithm

The replication algorithm consists of two parts:

- **A top-level "driver" routine `replicate()` that draws the first layer, and calls a second routine, `recursiveRep()`, to draw the rest of the layers.**
- **A routine `recursiveRep()` that recursively draws the rest of the desired number of layers.**

A tiling pattern is determined by how the p-gon pattern is transformed across p-gon edges. These transformations are in the array `edgeTran[]`

The Top-level Routine `replicate()`

```
Replicate ( motif ) {
  drawPgon ( motif, IDENT ) ; // Draw central p-gon
  for ( i = 1 to p ) {    // Iterate over each vertex
    qTran = edgeTran[i-1] ;
    for ( j = 1 to q-2 ) { // Iterate about a vertex
      exposure = (j == 1) ?  MIN_EXP : MAX_EXP ;
      recursiveRep ( motif, qTran, 2, exposure ) ;
      qTran = addToTran ( qTran, -1 ) ;
    }
  }
}
```

The function `addToTran()` **is described next.**

The Function `addToTran()`

Transformations contain a matrix, the orientation, and an index, `pPosition`, of the edge across which the last transformation was made (`edgeTran[i].pPosition` is the edge matched with edge `i` in the tiling). Here is `addToTran()`:

```
addToTran ( tran, shift ) {
  if ( shift % p == 0 ) return tran ;
  else return computeTran ( tran, shift ) ;
}
```

where `computeTran()` is:

```
computeTran ( tran, shift ) {
  newEdge = (tran.pPosition +
            tran.orientation * shift) % p ;
  return tranMult(tran, edgeTran[newEdge]) ;
}
```

and where `tranMult ( t1, t2 )` multiplies the matrices and orientations, sets the `pPosition` to `t2.pPosition`, and returns the result.

The Routine `recursiveRep()`

```
recursiveRep ( motif, initialTran, layer, exposure ) {
  DrawPgon ( motif, initialTran ) ; // Draw p-gon pattern
  if ( layer < maxLayer ) { //   If any more layers
    pShift = ( exposure == MIN_EXP ) ? 1 :  0 ;
    verticesToDo = ( exposure == MIN_EXP ) ? p-3 :  p-2 ;
    for ( i = 1 to verticesToDo ) {// Do each vertex
      pTran = computeTran ( initialTran, pShift ) ;
      qSkip = ( i == 1 ) ?  -1 :  0 ;
      qTran = addToTran ( pTran, qSkip ) ;
      pgonsToDo = ( i == 1 ) ?  q-3 :  q-2 ;
      for ( j = 1 to pgonsToDo ) {// Go around a vertex
        newExposure = ( j == 1 ) ?  MIN_EXP : MAX_EXP ;
        recursiveRep(motif, qTran, layer+1, newExposure);
        qTran = addToTran ( qTran, -1 ) ;
      }
      pShift = (pShift + 1) % p ; // Go to next vertex
    }
  }
}
```

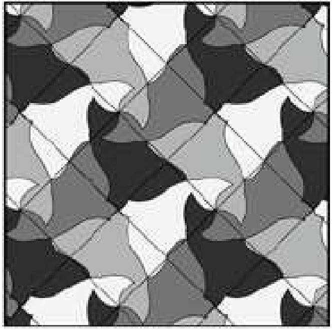
Special Cases

The algorithm above works for $p > 3$ and $q > 3$.

If $p = 3$ or $q = 3$, the same algorithm works, but with different values of `pShift`, `verticesToDo`, `qSkip`, **etc.**

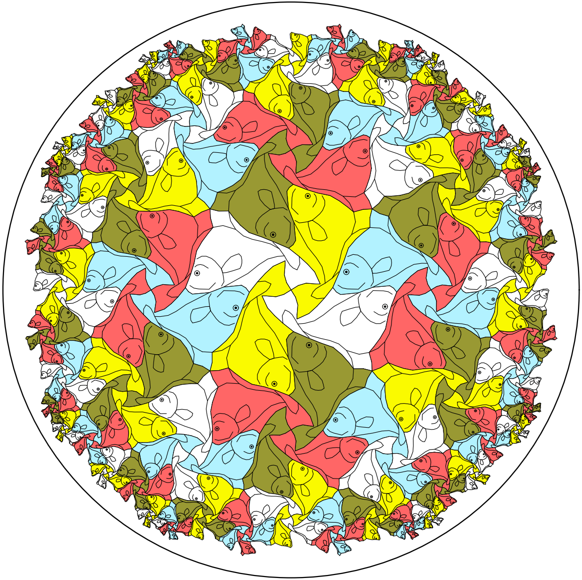
Sample Patterns

Escher's Euclidean Notebook Drawing 20, based on the $\{4, 4\}$ tessellation.

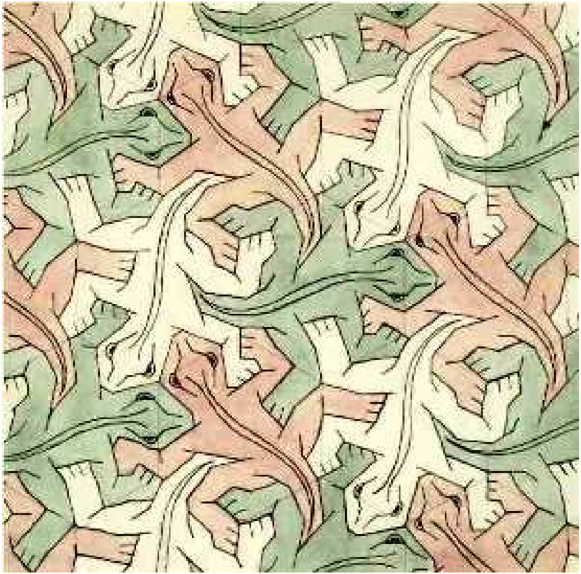


Escher's Spherical Fish Pattern Based on $\{4, 3\}$

A Hyperbolic Fish Pattern Based on $\{4, 5\}$

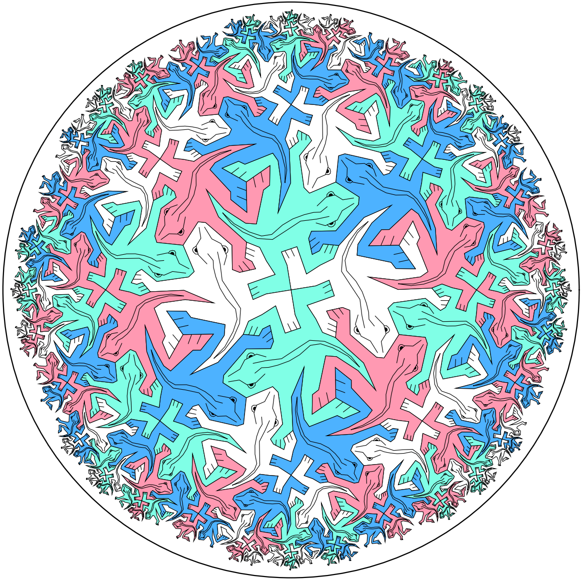


Escher's Euclidean Notebook Drawing 25, based on the $\{6, 3\}$ tessellation.



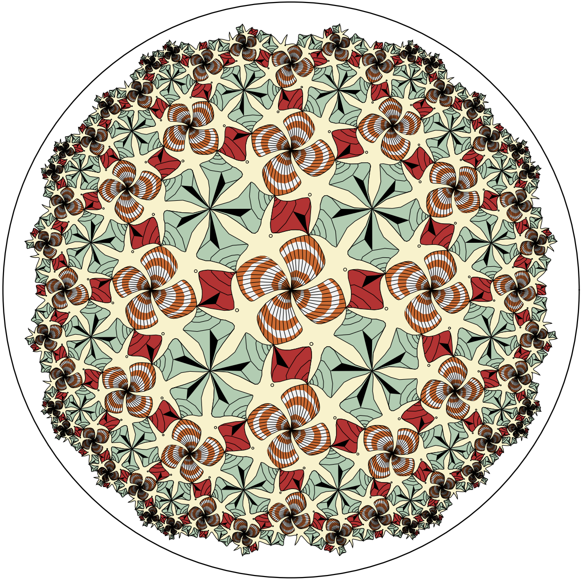
Escher's Print Reptiles based on Notebook Drawing 25

A Hyperbolic Lizard Pattern Based on $\{8, 3\}$

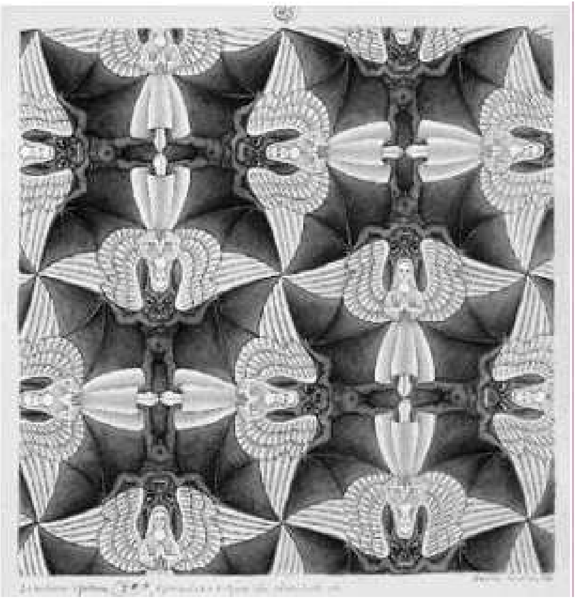


Escher's Euclidean Notebook Drawing 42, based on the $\{4,4\}$ tessellation.

A Hyperbolic Shell Pattern Based on $\{4, 5\}$

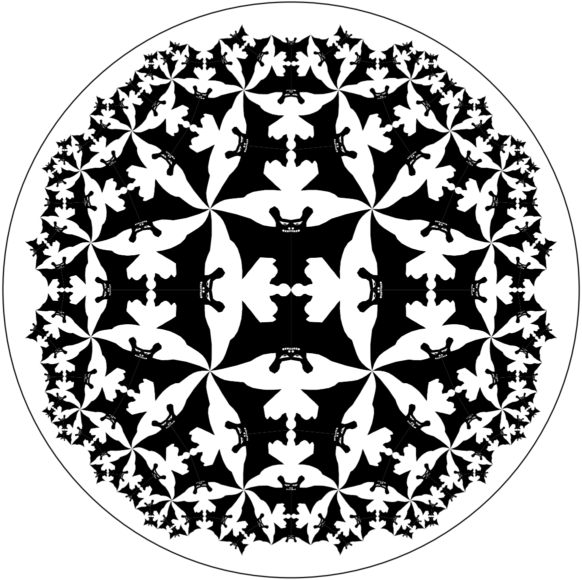


Escher's Euclidean Notebook Drawing 45, based on the $\{4,4\}$ tessellation.

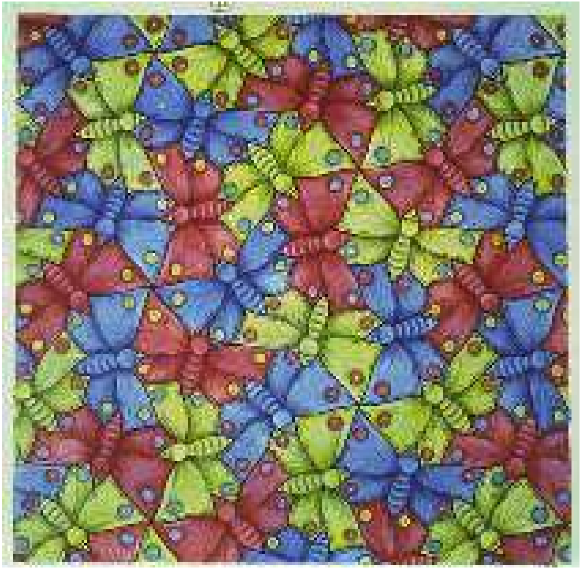


Escher's Spherical "Heaven and Hell" Based on $\{4,3\}$

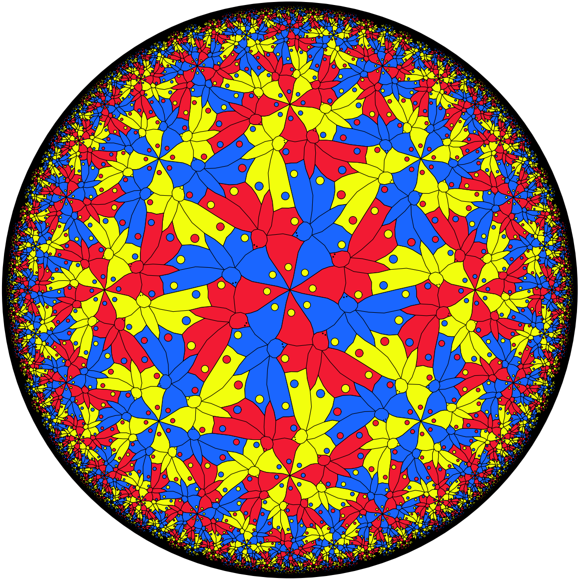
A Hyperbolic "Heaven and Hell" Pattern Based on $\{4, 5\}$



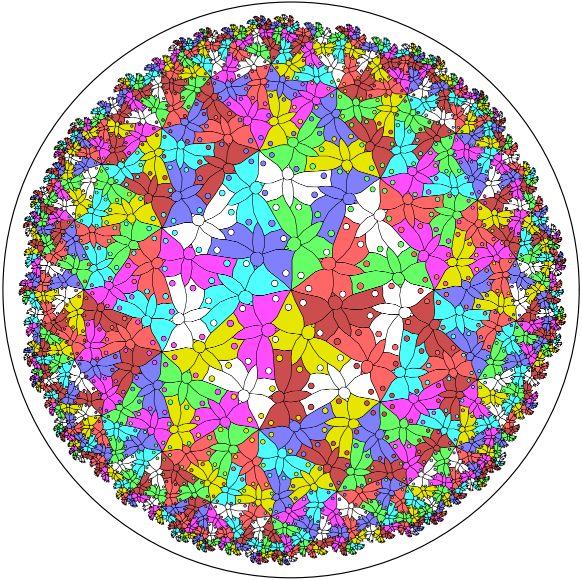
Escher's Euclidean Notebook Drawing 70, based on the $\{6, 3\}$ tessellation.



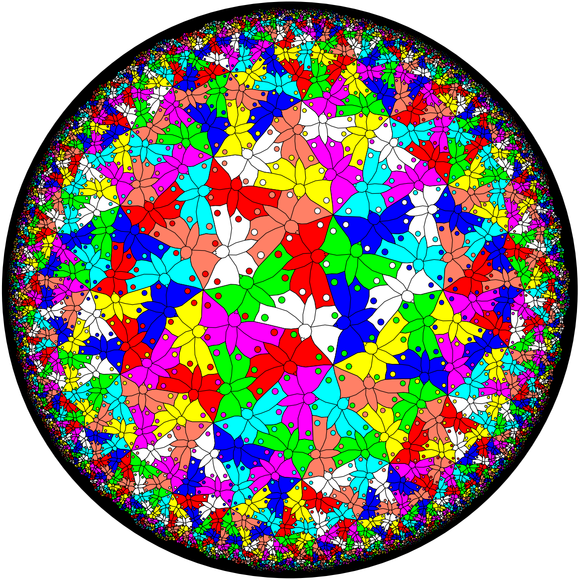
A Hyperbolic Butterfly Pattern Based on $\{8, 3\}$



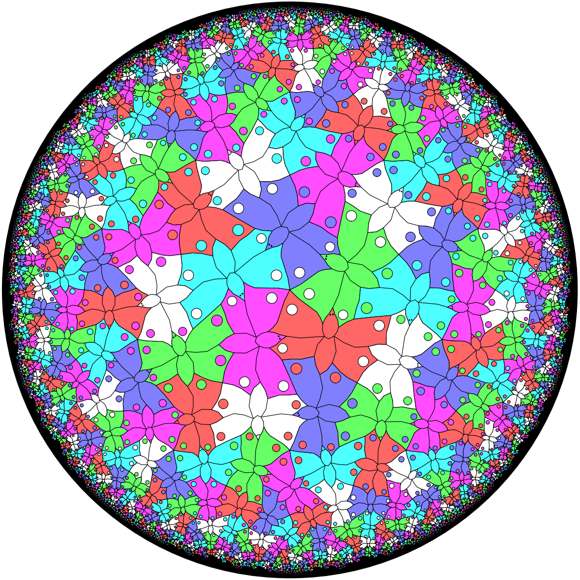
A Hyperbolic Butterfly Pattern Based on $\{7, 3\}$



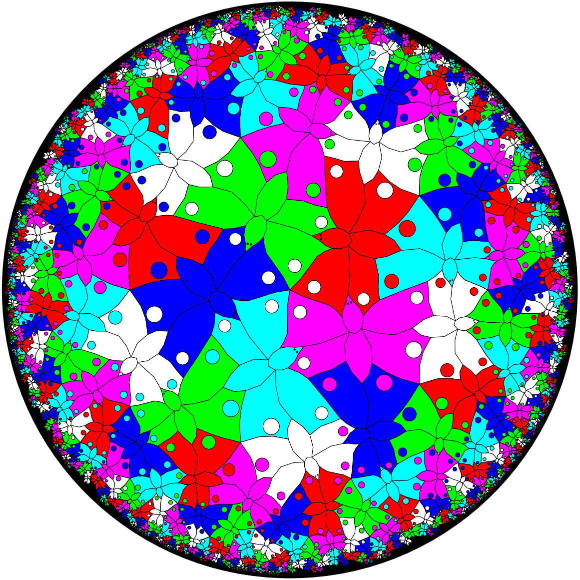
A Hyperbolic Butterfly Pattern Based on $\{3, 7\}$



A Hyperbolic Butterfly Pattern Based on $\{5,4\}$



A Hyperbolic Butterfly Pattern Based on $\{5, 5\}$



Future Work

- Extend the algorithm to handle tilings by non-regular polygons.
- Extend the algorithm to the cases infinite regular polygons: $\{p, \infty\}$ of infinite p-sided polygons, or $\{\infty, q\}$ of infinite-sided polygons meeting q at a vertex.
- Create a program to transform between different fundamental polygons.
- Automatically generate patterns with color symmetry.

Thank You

To CIRM and all the organizers of SubTile 2013

Contact Information:
Doug Dunham
Email: `ddunham@d.umn.edu`
Web: `http://www.d.umn.edu/~ddunham`