

# Repeating Hyperbolic Pattern Algorithms — Special Cases

**Douglas Dunham**

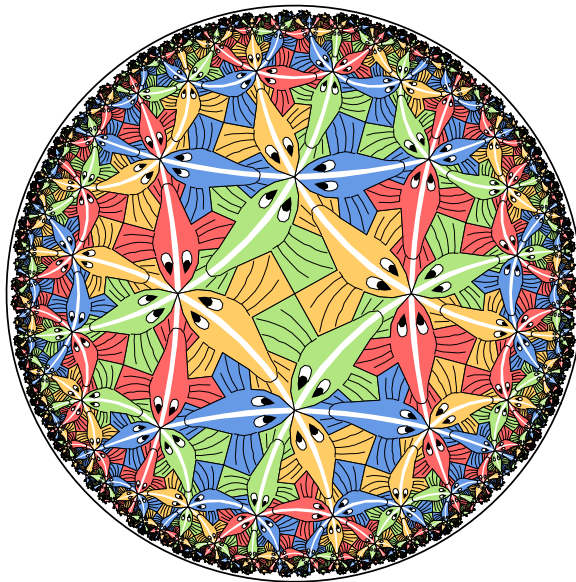
Department of Computer Science

University of Minnesota, Duluth

Duluth, MN 55812-3036, USA

E-mail: [ddunham@d.umn.edu](mailto:ddunham@d.umn.edu)

Web Site: <http://www.d.umn.edu/~ddunham/>



# Outline

- 1. History**
- 2. A Repeating hyperbolic pattern algorithm based on regular tessellations**
- 3. A Repeating hyperbolic pattern algorithm based on non-regular tessellations**
- 4. Future work**

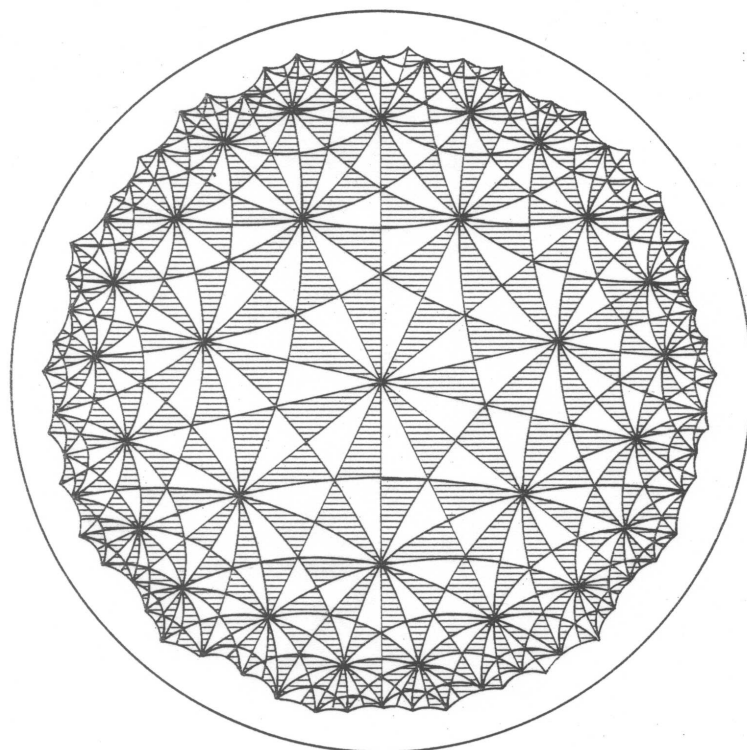
# **1. History**

**1. Pre-Escher**

**2. Escher's patterns**

**3. Post-Escher = Dunham, Ferguson, Sazdanovic, etc.**

**Triangle group (7,3,2) tessellation**  
**Originally in *Vorlesungen über die Theorie der***  
***elliptischen Modulfunctionen***  
**F. Klein and R. Fricke, 1890.**



**H.S.M. Coseter's Figure 7**  
**in *Crystal Symmetry and Its Generalizations***  
**Trans. Royal Soc. of Canada, 1957.**

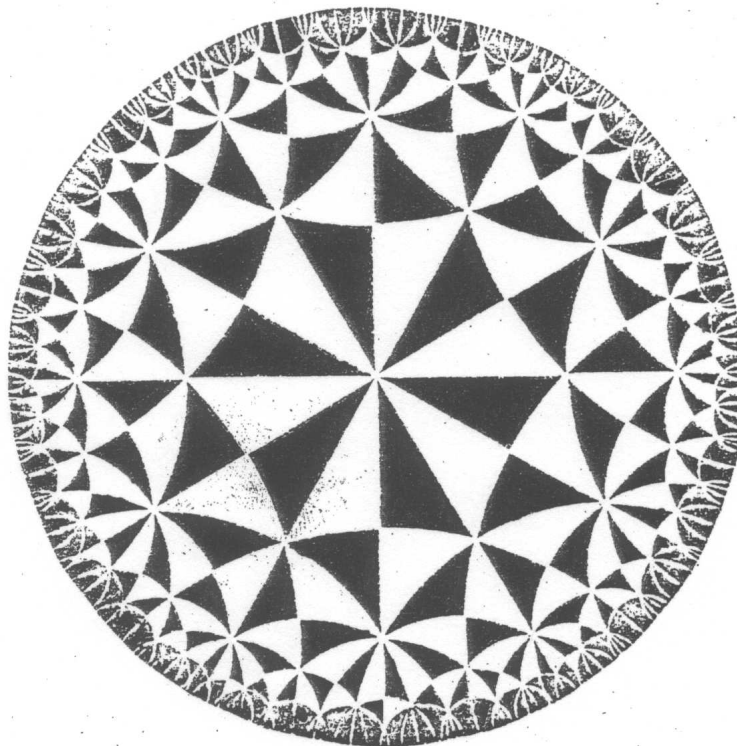
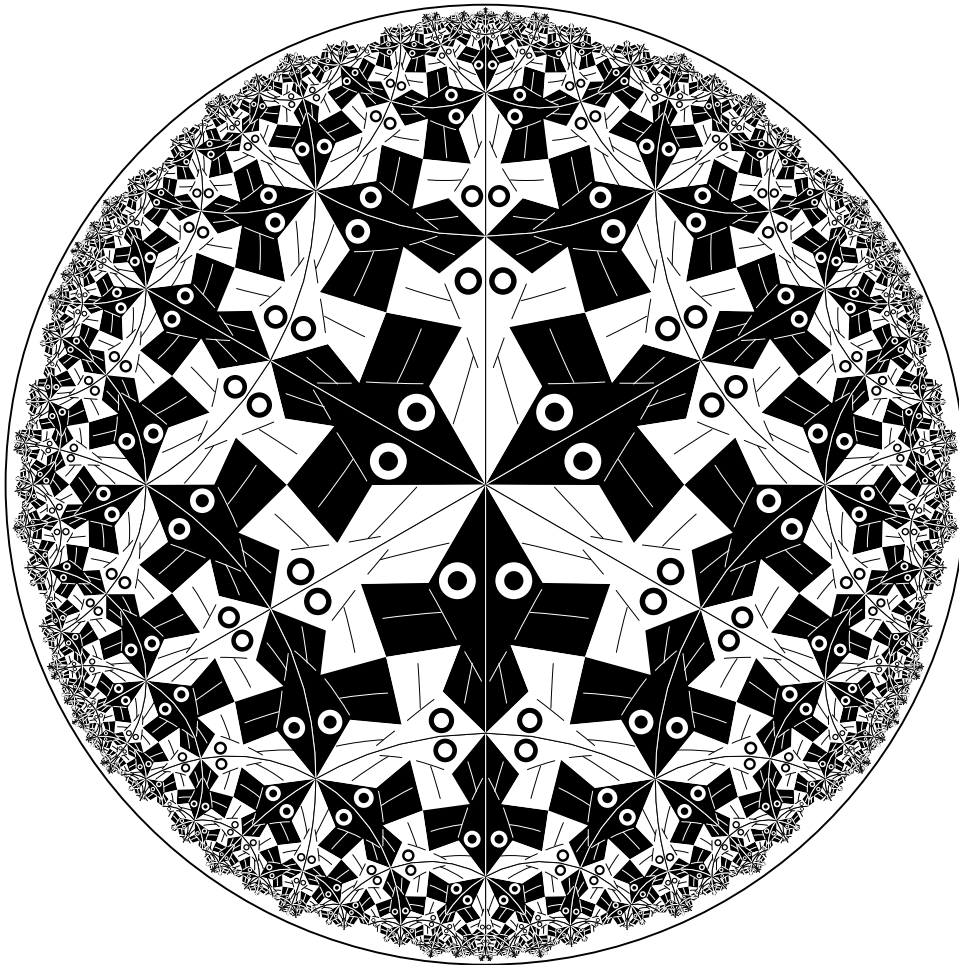


FIGURE 7

# M.C. Escher's "Circle Limit" Patterns

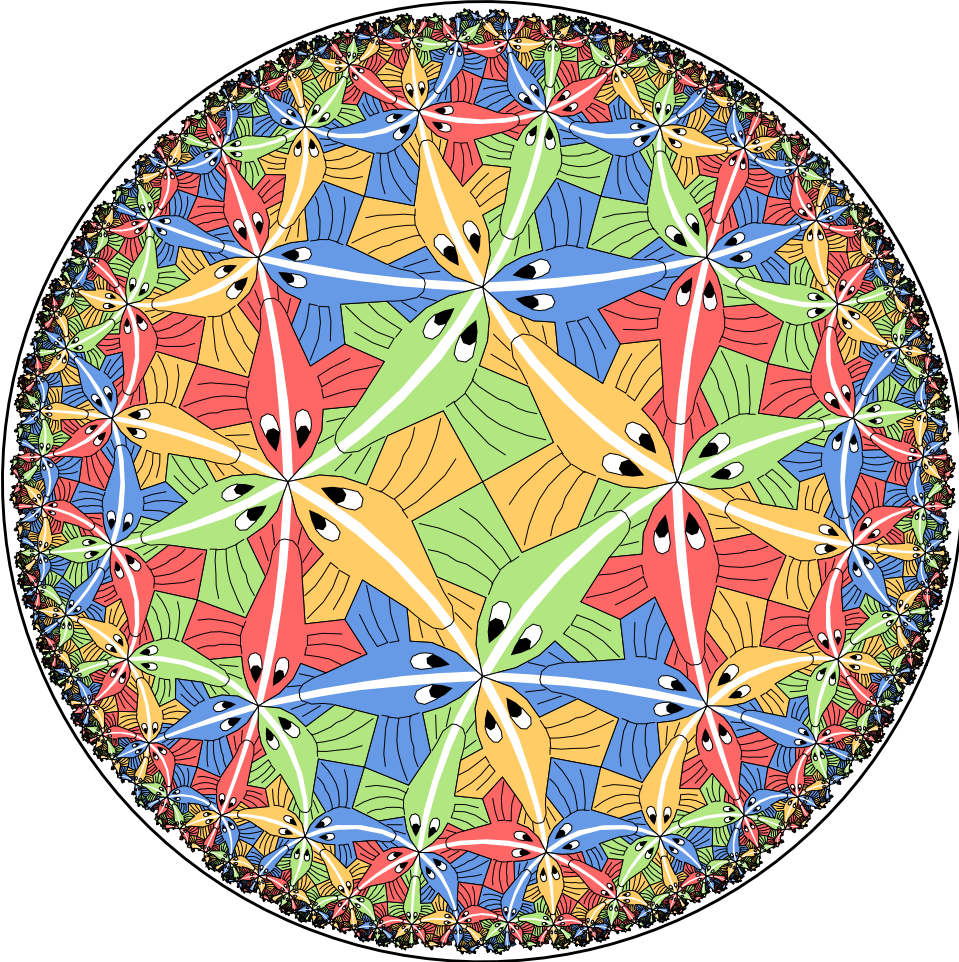
## *Circle Limit I*



# *Circle Limit II*

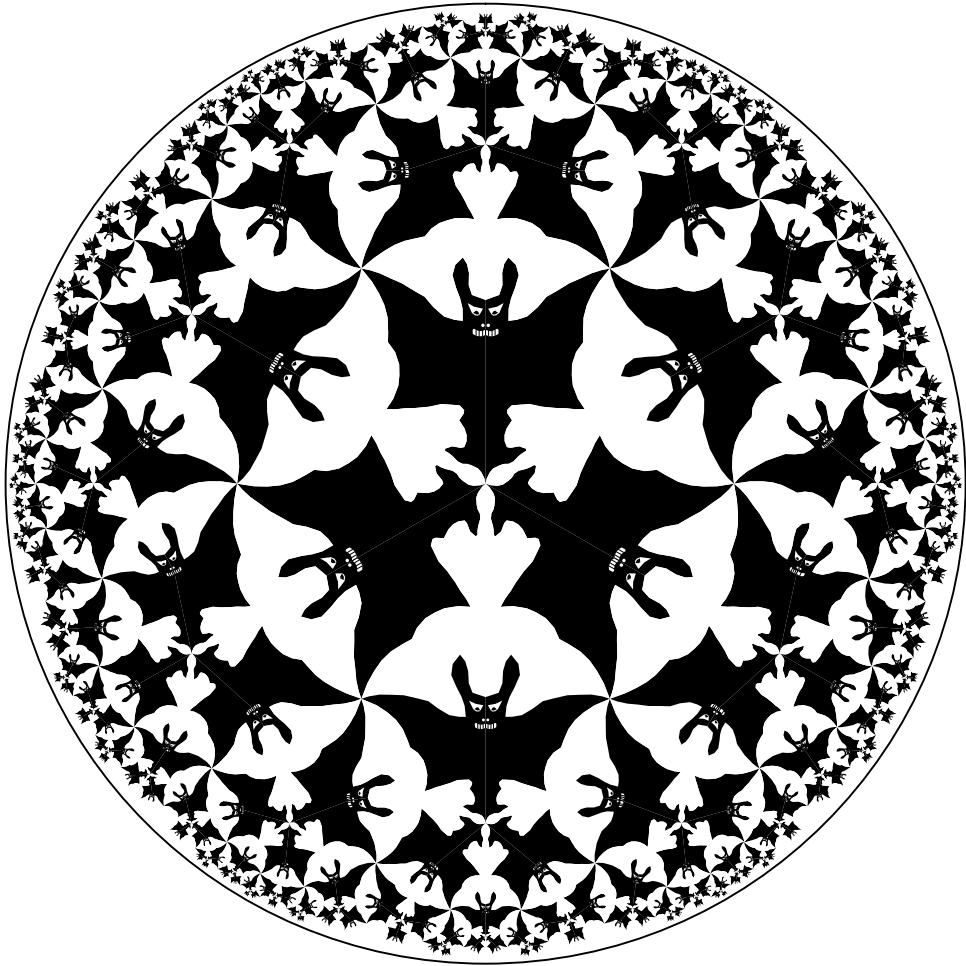


*Circle Limit III*



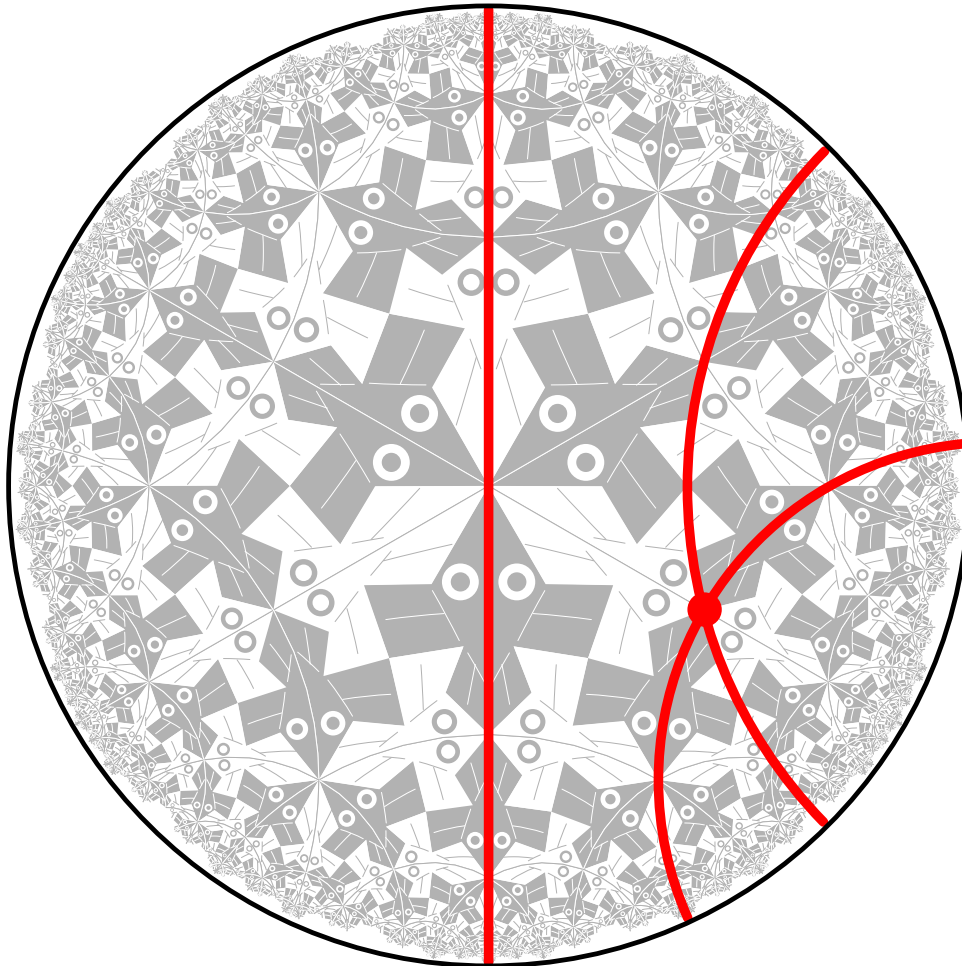


# *Circle Limit IV*



## 2. Generation of Repeating Hyperbolic Patterns

Following Escher, we use the Poincaré disk model of hyperbolic geometry.



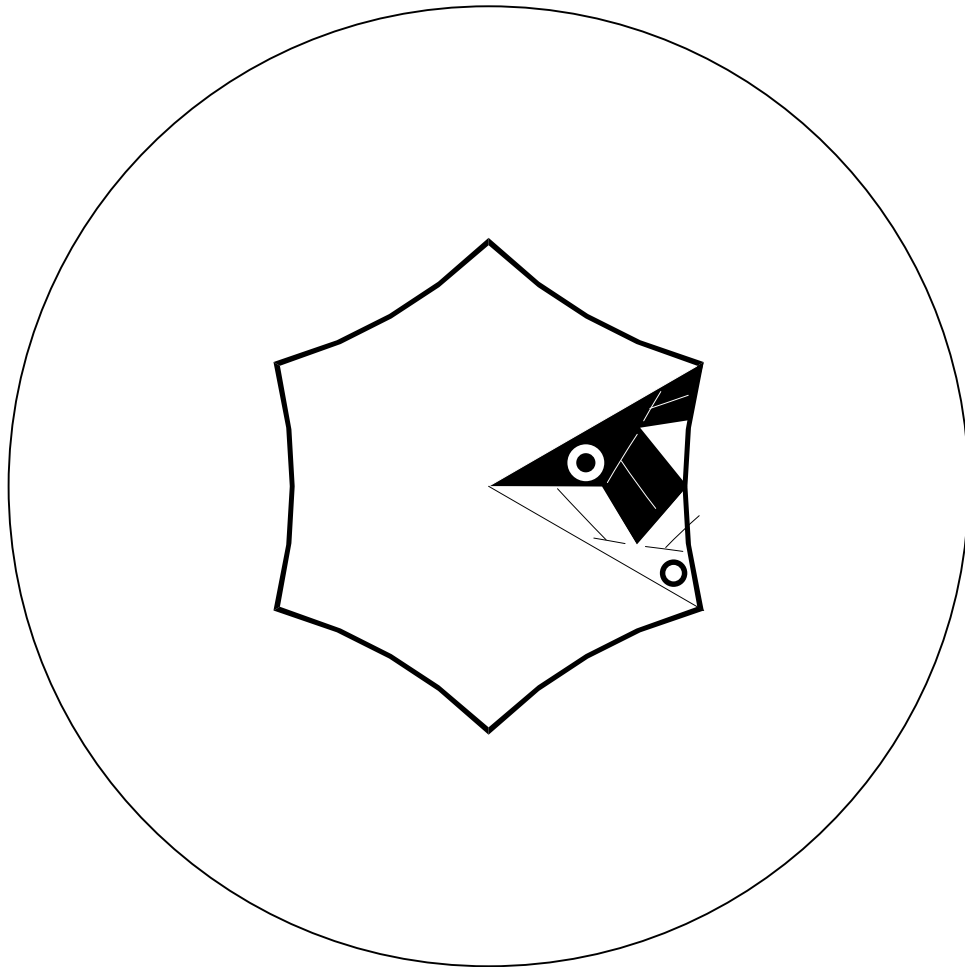
# The Pattern Generation Process

Consists of two steps:

1. Design the basic subpattern or *motif* — done by a hyperbolic drawing program.
2. Transform copies of the motif about the hyperbolic plane: *replication*

# Repeating Patterns

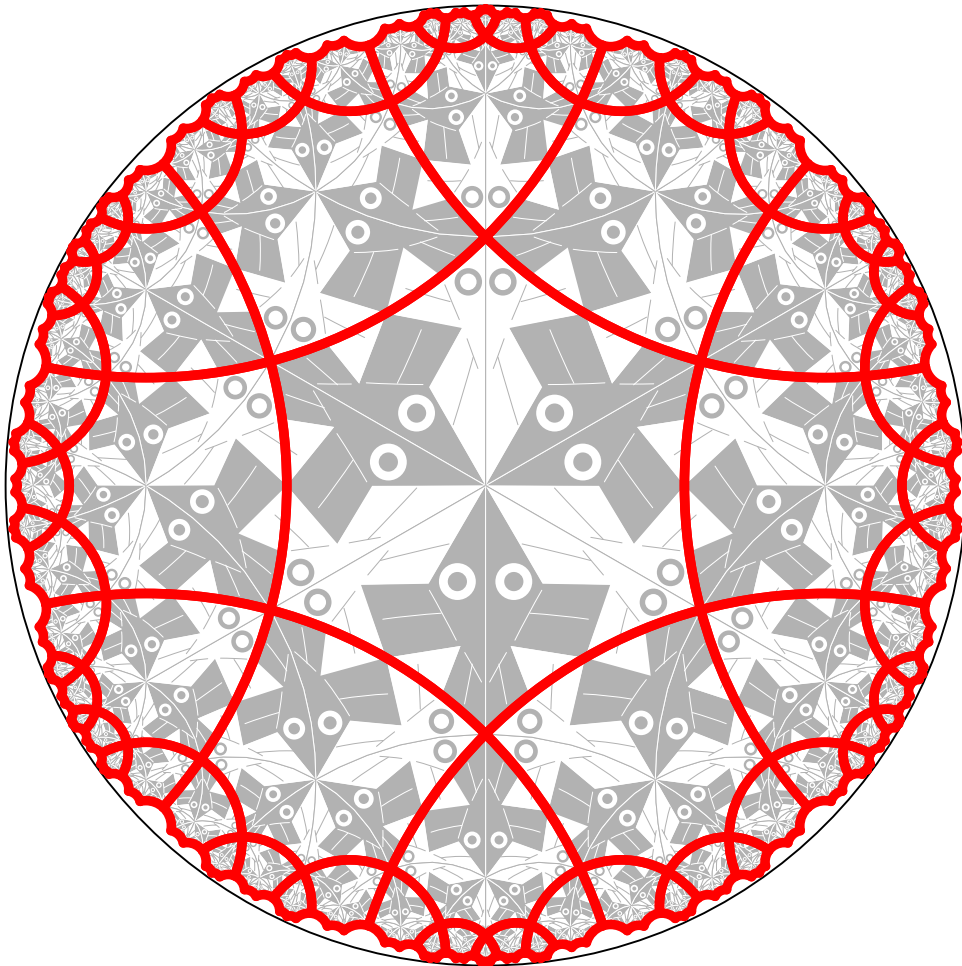
***A repeating pattern is composed of congruent copies of the motif. A motif for Circle Limit I.***



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

- Escher based his four “Circle Limit” patterns (and many of his Euclidean and spherical patterns) on regular tessellations.
- 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

$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
11		*	*	*	*	*	*	*	*	*	...		
10		*	*	*	*	*	*	*	*	*	...		
9		*	*	*	*	*	*	*	*	*	...		
8		*	*	*	*	*	*	*	*	*	...		
7		*	*	*	*	*	*	*	*	*	...		
$q$ 6		□	*	*	*	*	*	*	*	*	...		
5		○	*	*	*	*	*	*	*	*	...		
4		○	□	*	*	*	*	*	*	*	...		
3		○	○	○	□	*	*	*	*	*	...		
2													
1													
		1	2	3	4	5	6	7	8	9	10	11	...
		$p$											

□ - Euclidean tessellations  
 ○ - spherical tessellations  
 \* - hyperbolic tessellations

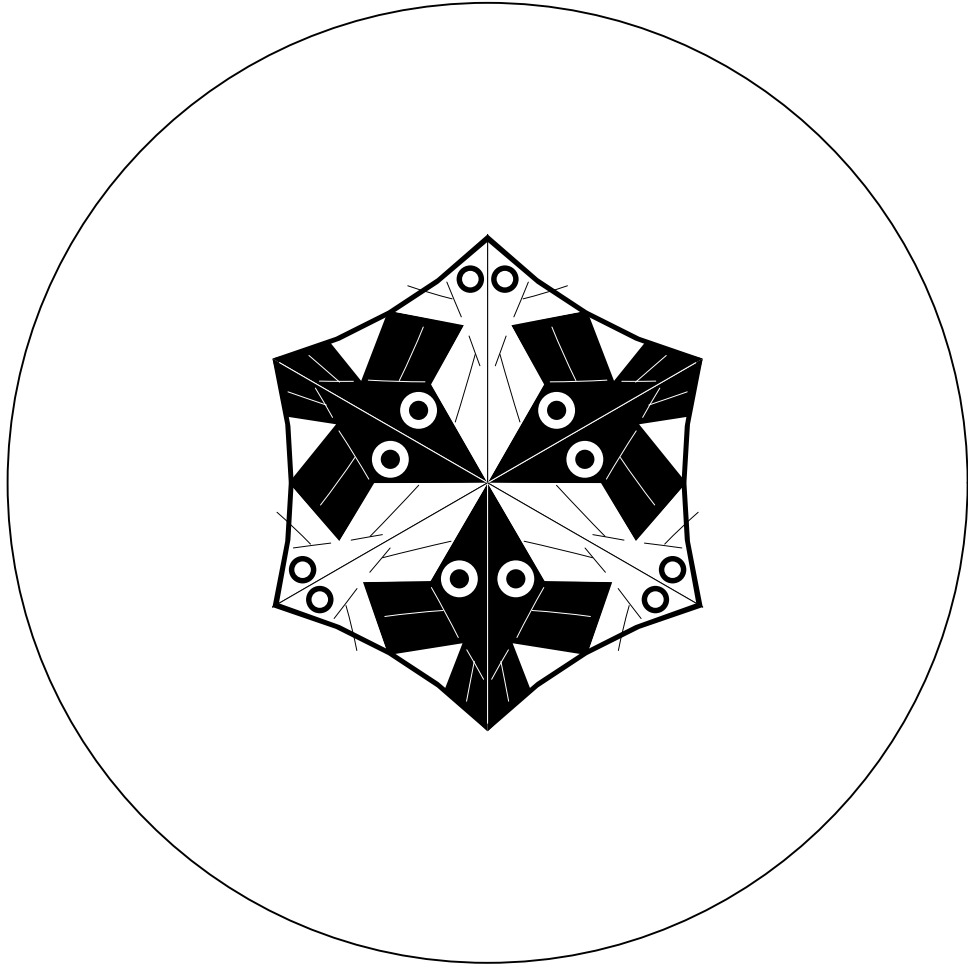
# Replicating the Pattern

In order 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 to replicate the pattern, we need only 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* for repeating Euclidean patterns.



# The p-gon pattern for *Circle Limit I*



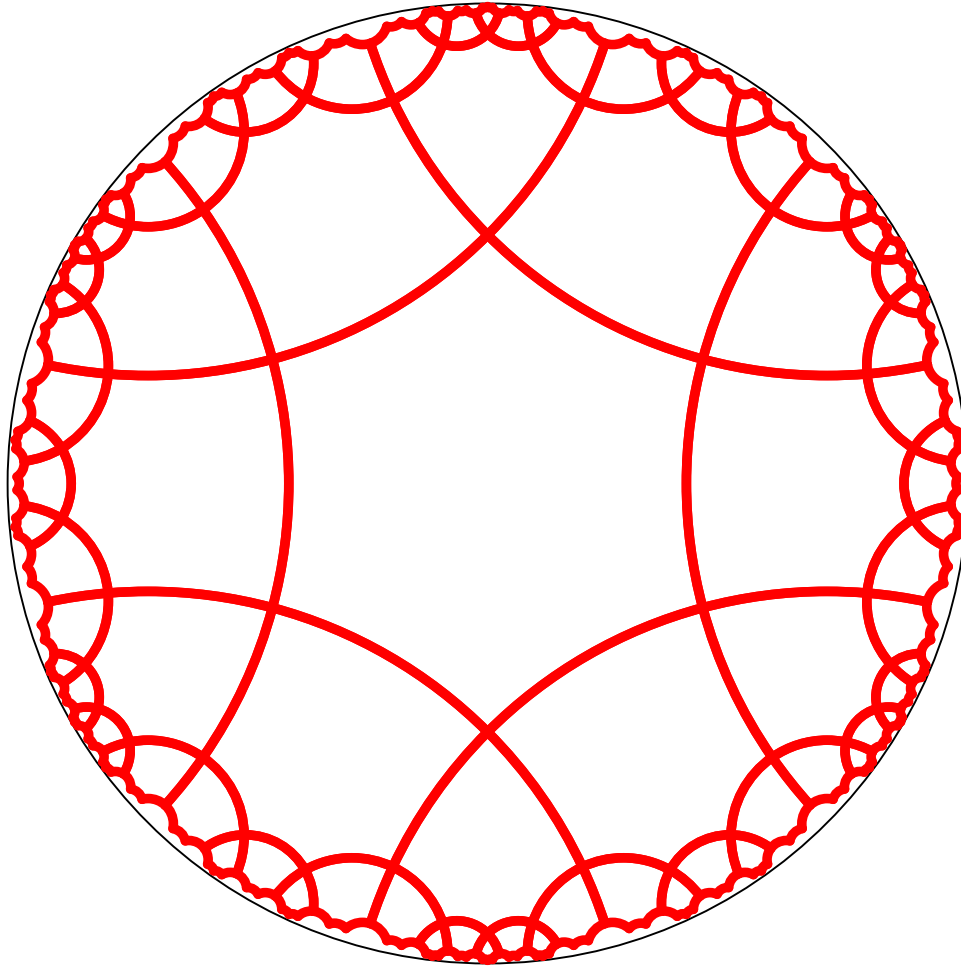
# Layers of p-gons

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 an 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.

# The Regular Tessellation $\{6, 4\}$ — Revisited

To show the layer structure and exposure of p-gons.



## 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.

- A p-gon has *minimum exposure* if it has the fewest edges in common with the next layer, and thus shares an edge 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.
- In the pseudo-code, we abbreviate these values as `MAX_EXP` and `MIN_EXP` respectively.

# The Replication Algorithm

The replication algorithm consists of two parts:

1. A top-level “driver” routine `replicate()` that draws the first layer, and calls a second routine, `recursiveRep()`, to draw the rest of the layers.
2. 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 < maxLayers ) { // If any more layers  
        pShift = ( exposure == MIN_EXP ) ? 1 : 0 ;  
        verticesToDo = ( exposure == MIN_EXP ) ? p-3 : p-2 ;  
  
        for ( i = 1 to verticesToDo ) { // Iterate over vertices  
            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 ) { // Iterate about a vertex  
                newExposure = ( i == 1 ) ? MIN_EXP : MAX_EXP ;  
                recursiveRep(motif, qTran, layer+1, newExposure);  
                qTran = addToTran ( qTran, -1 ) ;  
            }  
            pShift = (pShift + 1) % p ; // Advance 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.**

## The case $p = 3$

**In `replicate()` the calculation of exposure in the inner loop is the same as the general case.**

**In `recursiveRep()`:**

- `pShift = 1` **regardless of exposure.**
- `verticesToDo = 1` **regardless of exposure.**
- `qSkip` **is -1 for MIN\_EXP and 0 for MAX\_EXP.**
- `pgonsToDo` **is  $q - 4$  for MIN\_EXP and  $q - 3$  for MAX\_EXP.**
- `newExposure` **is the same as the general case.**

**In both `replicate()` and `recursiveRep()` at the last iteration of the inner loop, the call to `recursiveRep()` is replaced by a non-recursive call to `drawPgon()`.**

## The case $q = 3$

**In `replicate()`, `exposure = MAX_EXP` in the inner loop regardless of whether it is the first iteration or not.**

**In `recursiveRep()`:**

- `pShift` is 3 for `MIN_EXP` and 2 for `MAX_EXP`.
- `verticesToDo` is  $p - 5$  for `MIN_EXP` and  $p - 4$  for `MAX_EXP`.
- `qSkip = 0` for all cases.
- `pgonsToDo = 1` for all cases.
- `newExposure` is `MIN_EXP` if  $i = 1$  and `MAX_EXP` if  $i > 1$ .

# Some New Hyperbolic Patterns

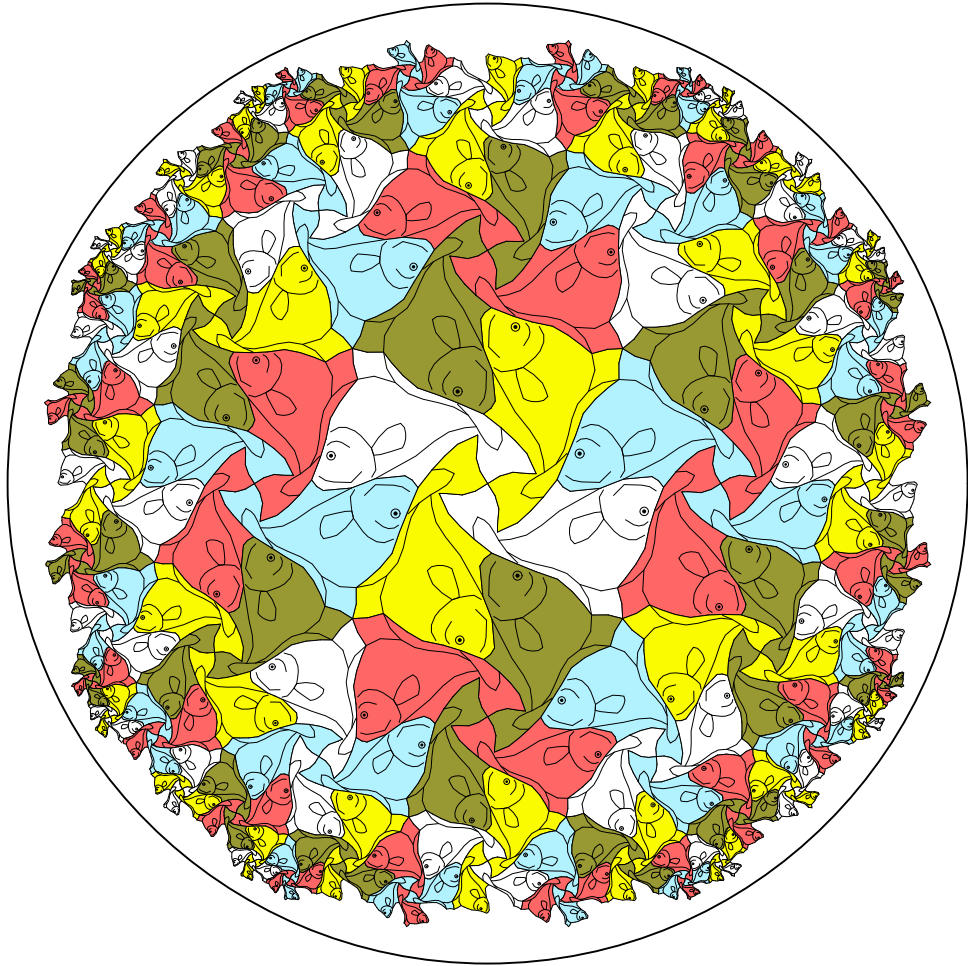
Escher's Euclidean Notebook Drawing 20, based on the  $\{4, 4\}$  tessellation.



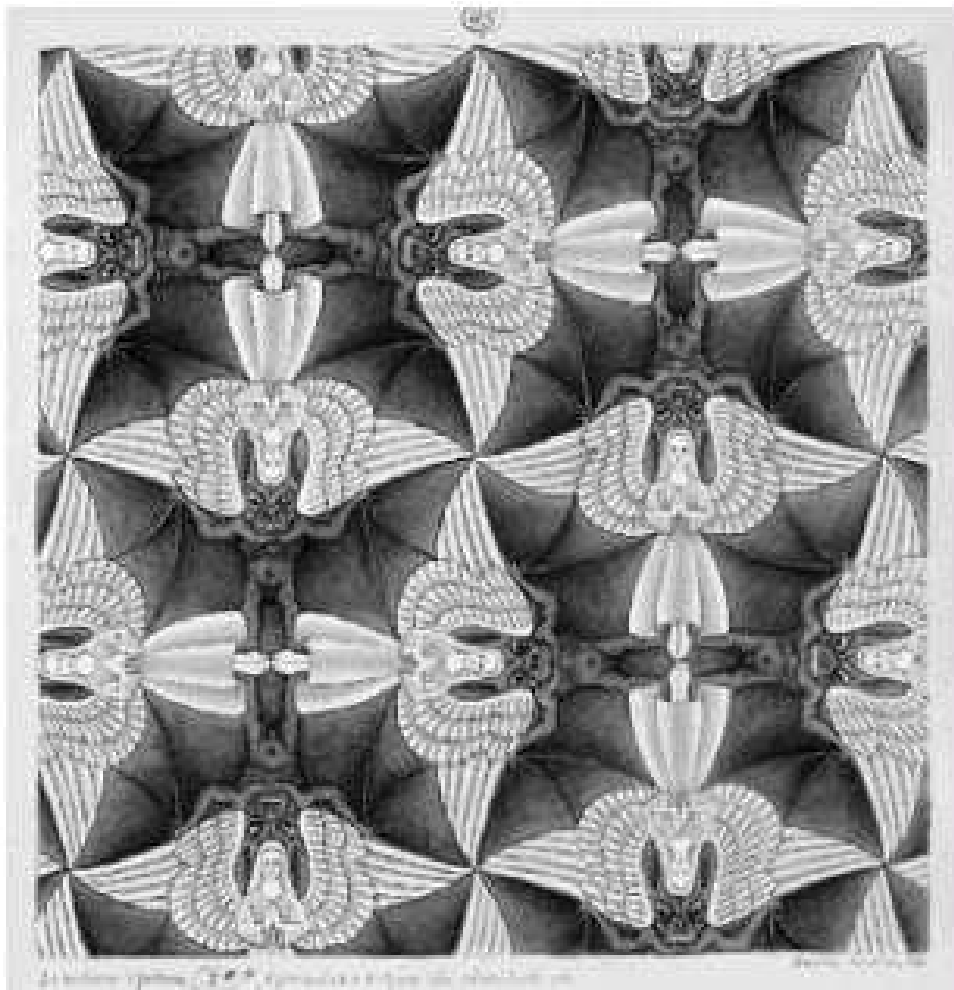
# Escher's Spherical Fish Pattern Based on $\{4, 3\}$



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



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

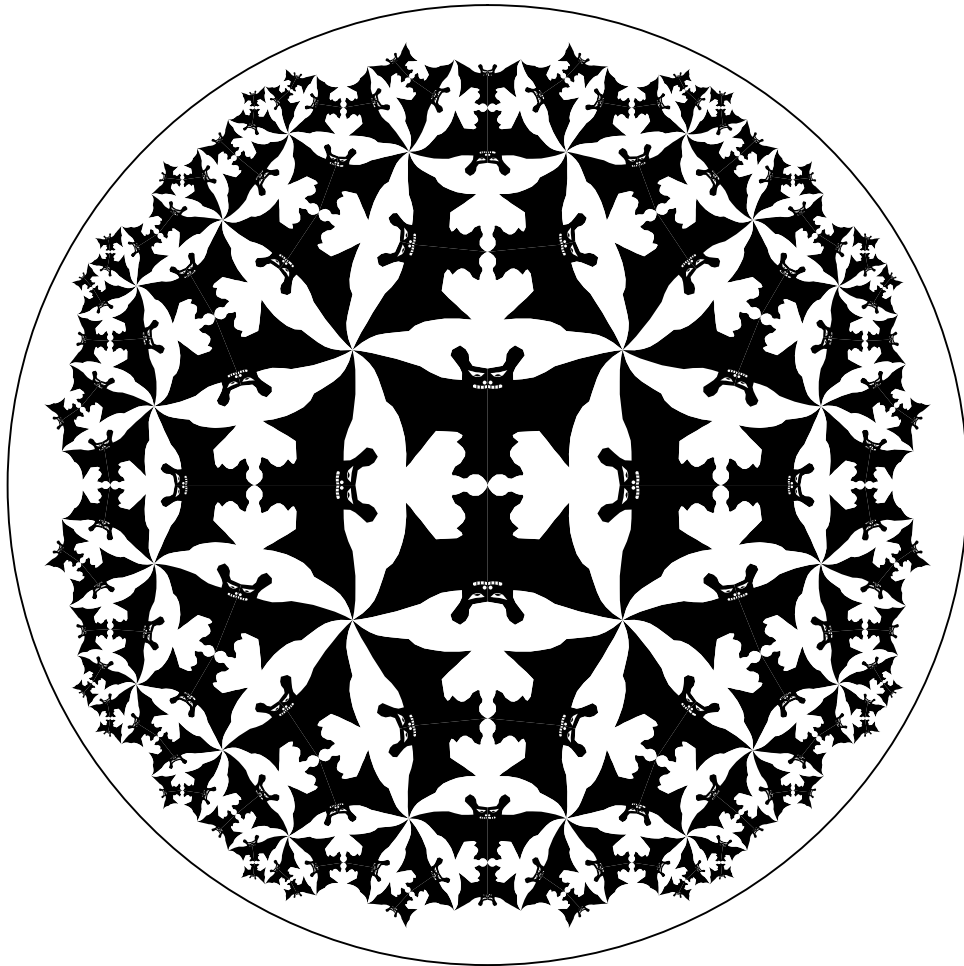


# Escher's Spherical "Heaven and Hell" Based on $\{4, 3\}$

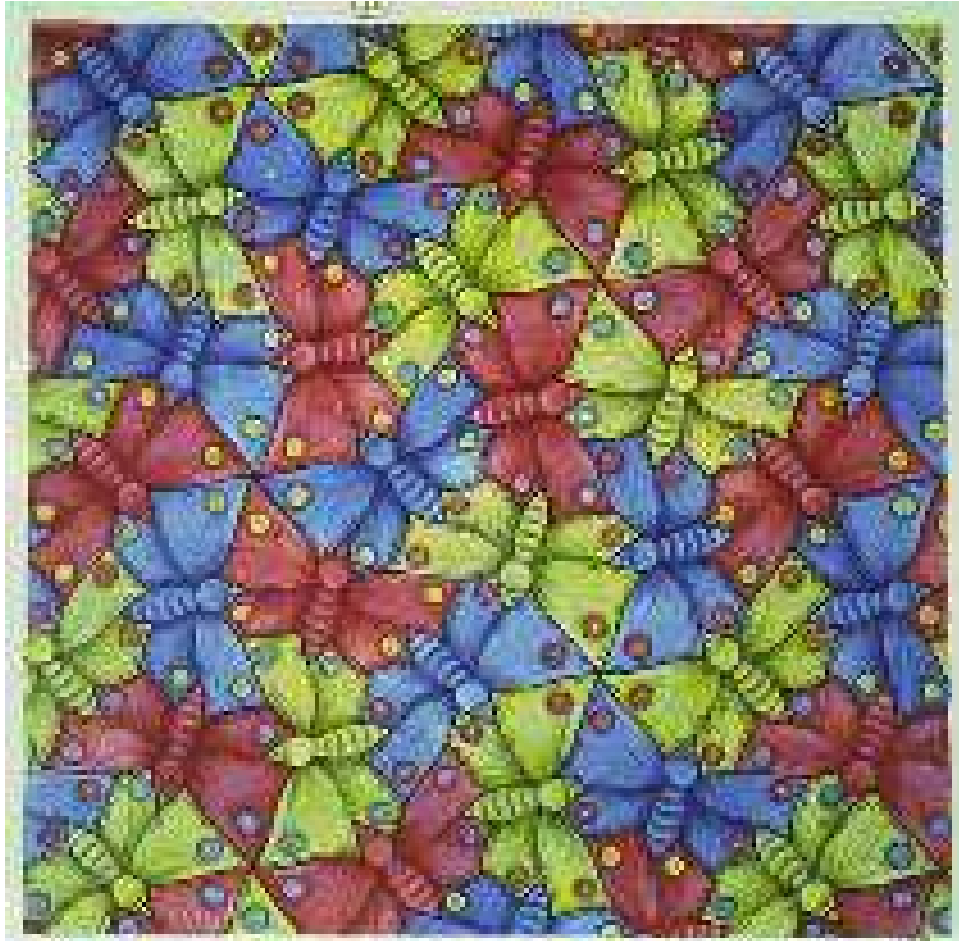




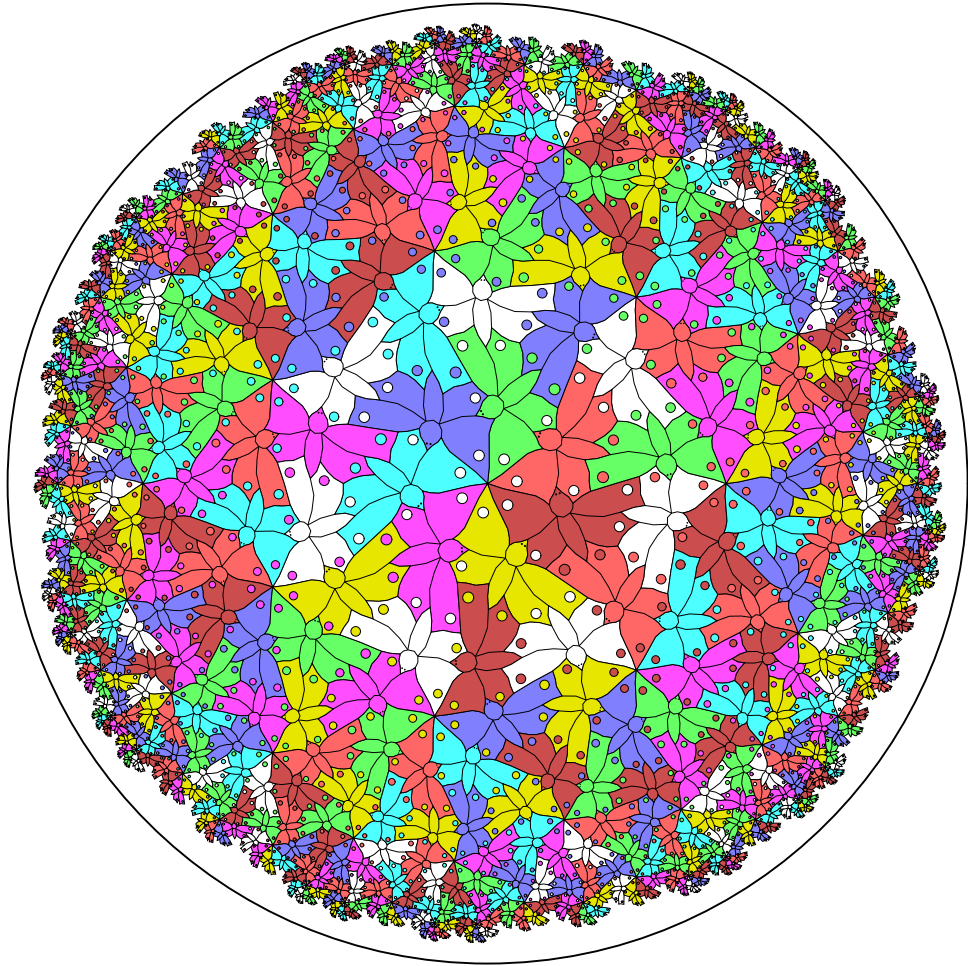
# A Hyperbolic “Heaven and Hell” Pattern Based on $\{4, 5\}$



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



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



### 3. Patterns Based on Non-Regular Polygon Tessellations

A non-regular  $p$ -sided polygon with  $q_1, q_2, \dots, q_p$  copies around the respective vertices forms a hyperbolic tessellation provided

$$\sum_{i=1}^p \frac{1}{q_i} < \frac{p}{2} - 1$$

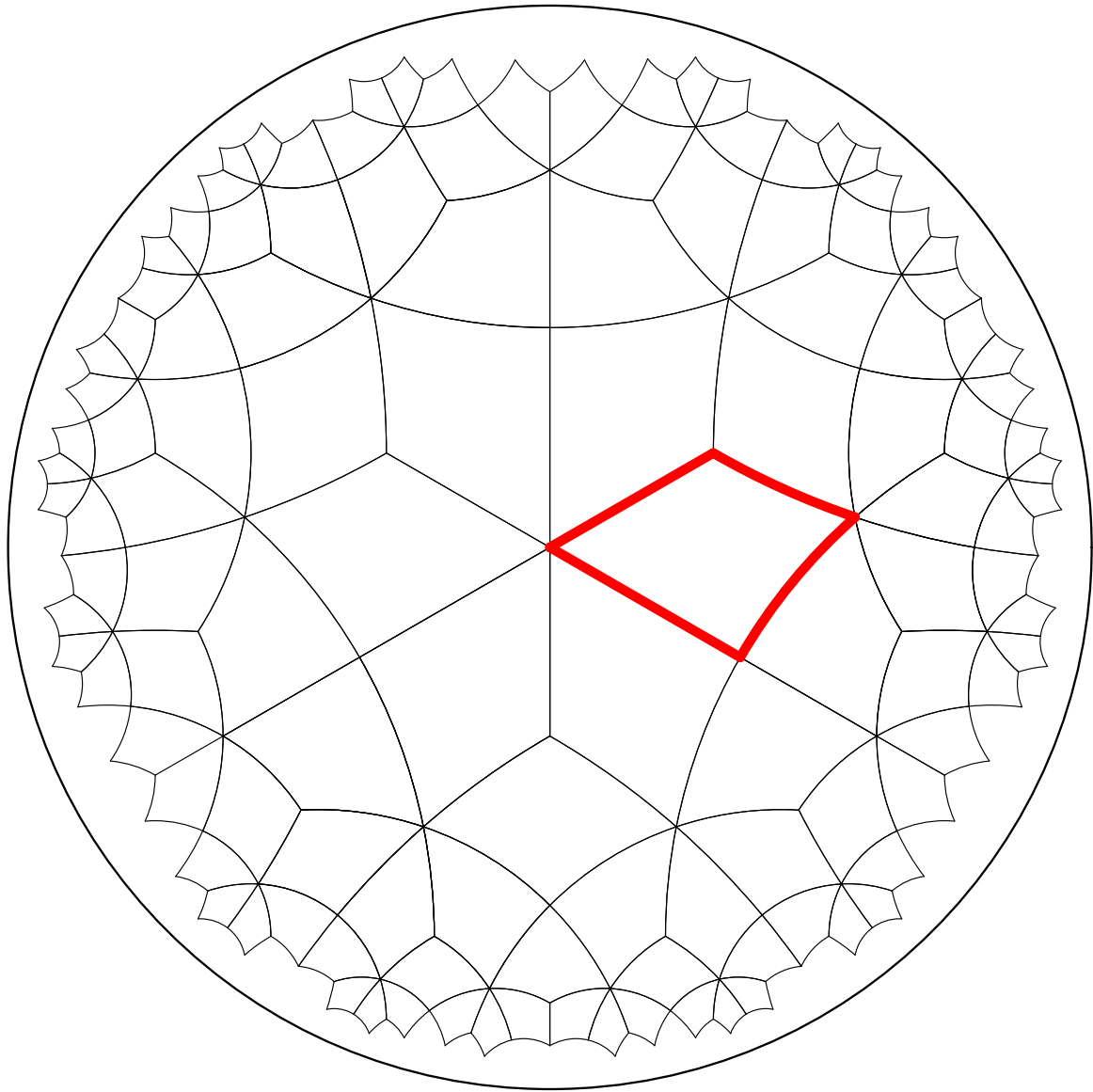
(so the interior angle at the  $i^{\text{th}}$  vertex is  $2\pi/q_i$ ).

This tessellation is denoted  $\{p; q_1, q_2, \dots, q_p\}$

The pattern drawing algorithm is similar to the case for regular tessellations: a non-recursive “driver”, `replicate()` calls a recursive routine `replicateMotif()`.

Unfortunately this algorithm draws multiple copies of the motif if  $p = 3$  or if any of the  $q_i = 3$ . There are only a few duplications near the center, but the number of them grows exponentially in the number of layers.

# A $\{4; 6, 3, 6, 4\}$ Polygon Tessellation



## The Top-level “Driver” replicate()

The replication process starts with the following top-level “driver”, which calls the recursive routine replicateMotif() to create the rest of the pattern.

```
replicate ( motif )
{
  for ( j = 1 to q[1] )
  {
    qTran = edgeTran[1] ;

    replicateMotif(motif, qTran, 2, MAX_EXP) ;

    qTran = addToTran ( qTran, -1 ) ;
  }
}
```

# The Recursive Routine replicateMotif()

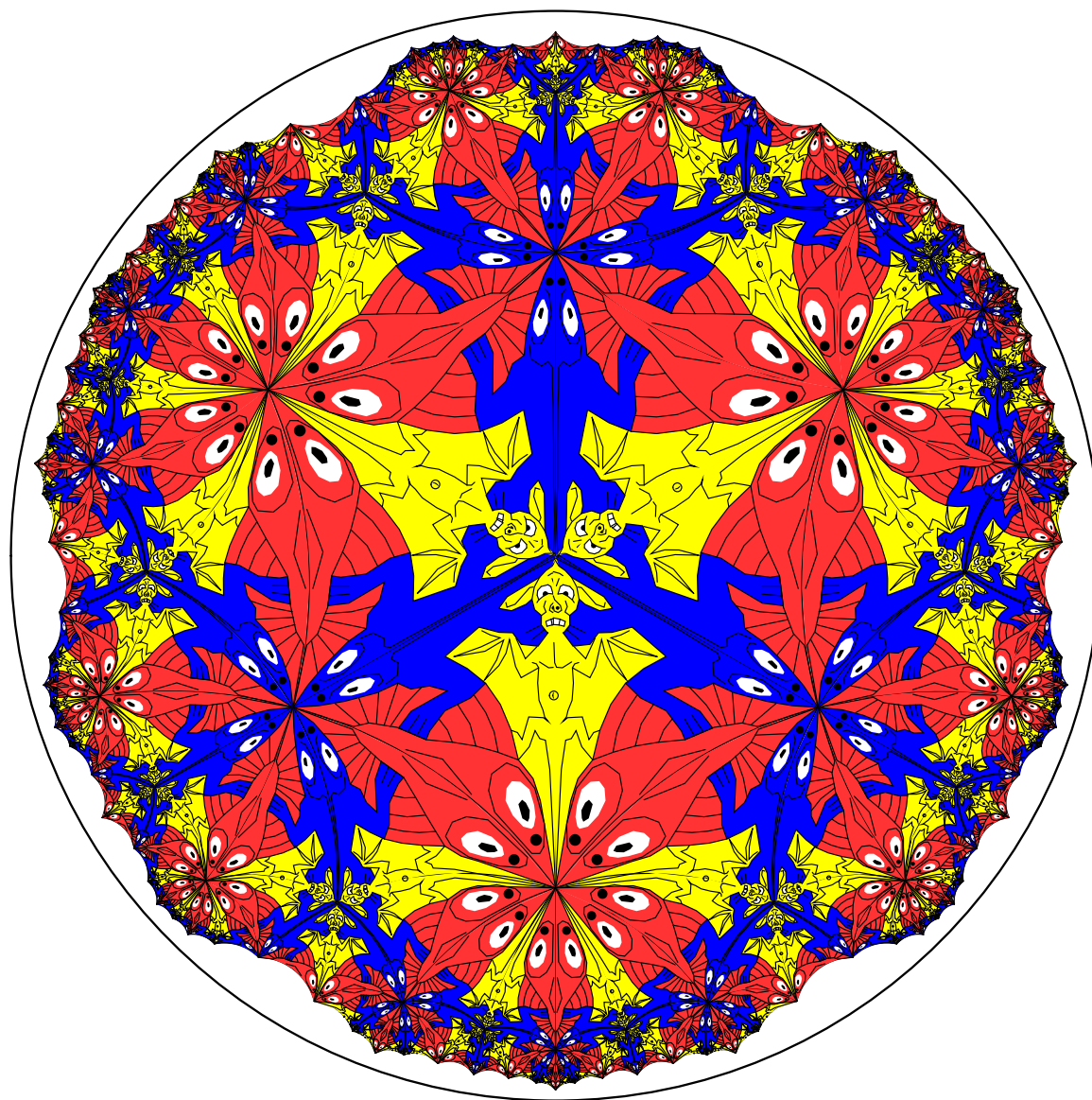
```
replicateMotif(motif, inTran, layer, exposure)
{
  drawMotif ( motif, inTran ) ;
  if ( layer < maxLayers )
  {
    pShift = pShiftArray[exposure] ;
    verticesToDo = p -
                  verticesToSkipArray[exposure] ;

    for ( i = 1 to verticesToDo )
    {
      pTran = computeTran(initialTran, pShift) ;
      first_i = ( i == 1 ) ;
      qTran = addToTran(pTran, qShiftArray[first_i]) ;
      if ( pTran.orientation > 0 )
      {
        vertex = (pTran.pposition-1) % p ;
      }
      else
      {
        vertex = pTran.pposition ;
      }
      polygonsToDo = q[vertex] -
                    polygonsToSkipArray[first_i] ;

      for ( j = 1 to polygonsToDo )
      {
        first_j = ( j == 1 ) ;
        newExpose = exposureArray[first_j] ;

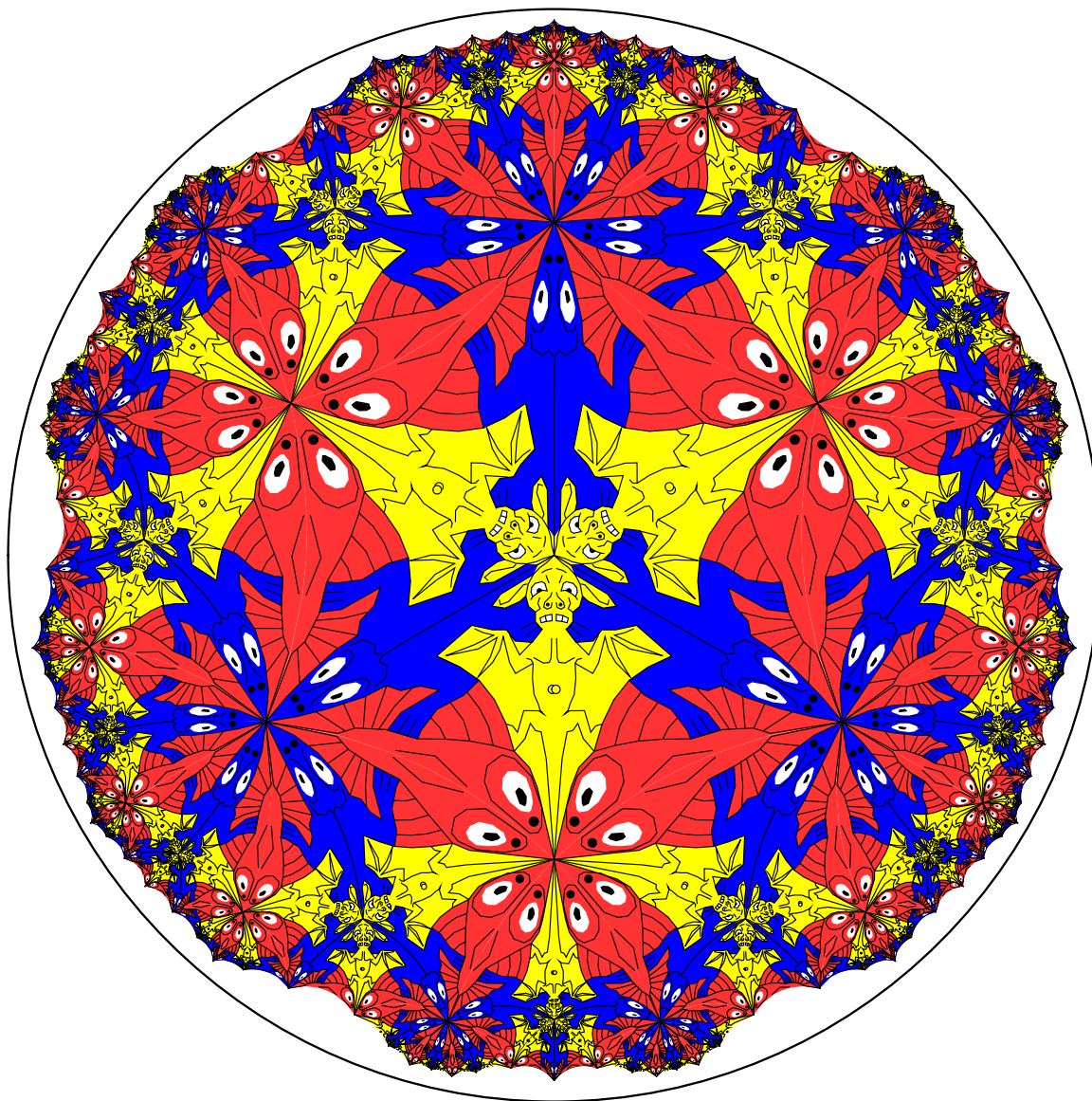
        replicateMotif(motif, qTran, layer+1, newExpose) ;
        qTran = addToTran ( qTran, -1 ) ;
      }
      pShift = (pShift + 1) % p ;
    }
  }
}
```

# A “Three Element” Pattern with Different Numbers of Animals Meeting at their Heads

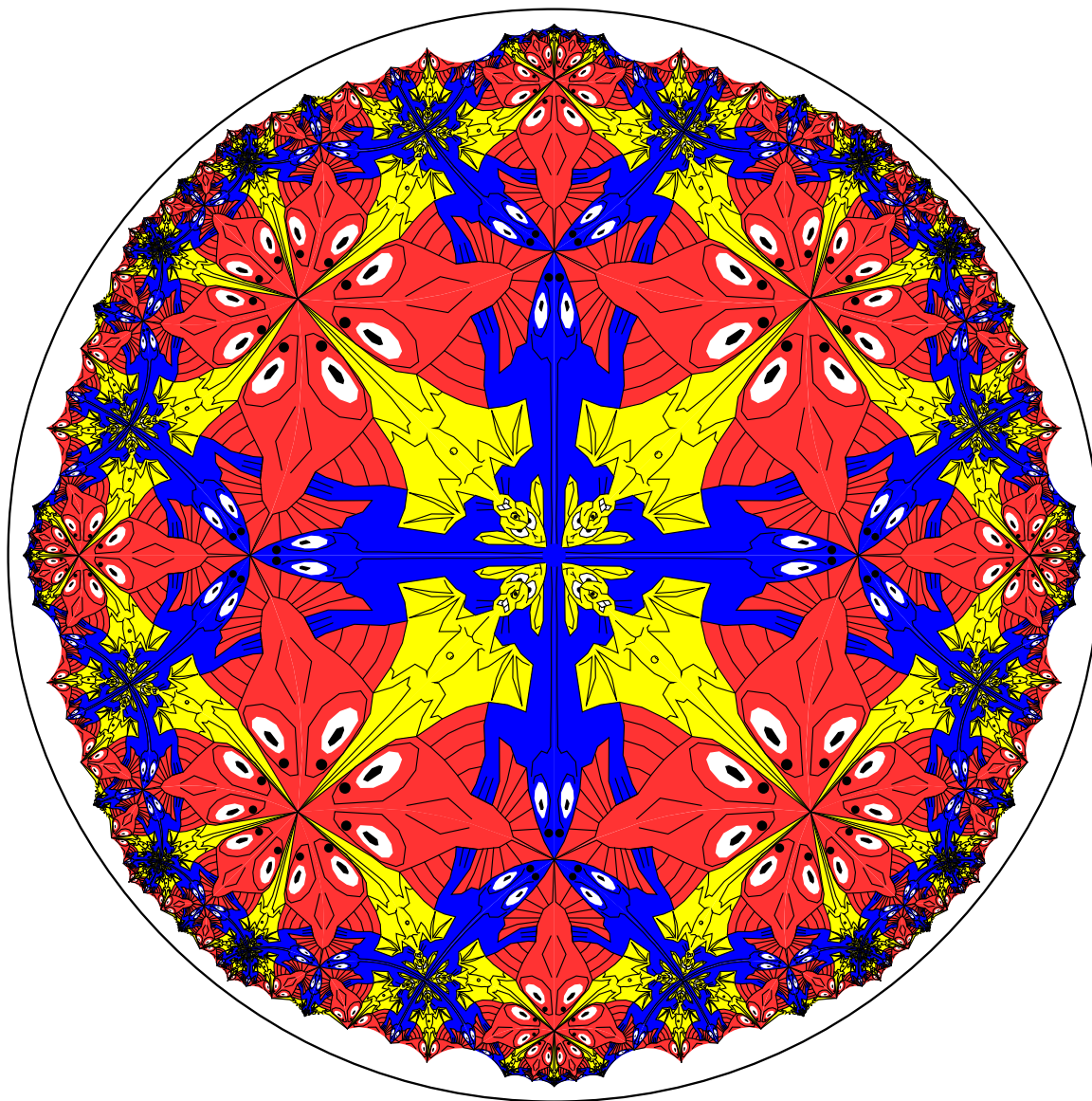




# A “Three Element” Pattern with 3 Bats, 5 Lizards, and 4 Fish Meeting at their Heads



# A “Three Element” Pattern with 3 Bats, 5 Lizards, and 4 Fish Meeting at their Heads



## 4. Future Work

- **Fix the non-regular polygon tessellation algorithm so that it does not make duplicate copies of the motif at some locations.**
- **Allow some or all of the vertices of the fundamental polygon to lie on the bounding circle.**
- **Automatically generate patterns with color symmetry.**

**The End**

**I hope not!**

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



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

