An Algorithm to Generate Repeating Hyperbolic Patterns

Douglas Dunham
Department of Computer Science
University of Minnesota, Duluth
Duluth, MN 55812-3036, USA
E-mail: ddunham@d.umn.edu
Web Site: http://www.d.umn.edu/~ddunham/
The \{8,3\} tessellation on *Circle Limit III*
An Islamic pattern based on the \(\{8,3\}\) tessellation
• **Points:** points within the **bounding circle**

• **Lines:** circular arcs perpendicular to the bounding circle (including diameters as a special case)
The Regular Tessellations \( \{p,q\} \)

There is a regular tessellation, \( \{p,q\} \) of the hyperbolic plane by regular \( p \)-sided polygons meeting \( q \) at a vertex provided

\[(p - 2)(q - 2) > 4\]

The tessellation \( \{6,4\} \) superimposed on the Circle Limit I pattern.
An arabesque pattern based on the \{6,4\} tessellation
The General Replication Algorithm

A *motif* is a basic sub-pattern, of which the entire repeating pattern is comprised.

*Replication* is the process of transforming copies of the motif about the hyperbolic plane in order to create the whole repeating pattern.

A *fundamental region* for the symmetry group of a pattern is a closed topological disk such that copies of it cover the plane without gaps or overlaps.

In Escher patterns the motif can usually be used as a fundamental region.

For a pattern with a finite motif, the fundamental region can be taken to be a convex polygon. This polygon will contain exactly the right pieces of the motif to reconstruct it.

Replication using copies of such a fundamental polygon will also create the entire pattern of motifs.
A Fundamental Polygon Tessellation

A quadrilateral can be used as the fundamental region for the *Circle Limit III* pattern, as shown below.
Layers of Fundamental Polygons

The fundamental polygons are arranged in layers (also called coronas in tiling literature), which are defined inductively.

The first layer consists of all polygons with a vertex at the center of the bounding circle.

The $k + 1^{st}$ layer consists of all polygons sharing an edge or vertex with the $k^{th}$ layer (and no previous layers).
A Polygon Tessellation Showing Layers

The polygon tessellation, with a fundamental polygon emphasized and parts of layers 1, 2, and 3 labeled.
Specification of the Fundamental Polygon

We use \( \{p; q_1, q_2, \ldots, q_p\} \) to denote the fundamental polygon with \( p \) sides and \( q_i \) polygons meeting at vertex \( i \) (so the interior angle at the \( i^{th} \) vertex is \( 2\pi / q_i \)).

The condition that a polygon is a fundamental polygon for a hyperbolic tessellation is that:

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

(which generalizes the condition \( (p - 2)(q - 2) > 4 \) for regular tessellations). If the “<” is replaced with “=” or “>”, one obtains a Euclidean or spherical tessellation respectively.

We say a polygon of a tessellation has \textit{minimal exposure} if it shares an edge with a previous layer; we say it has \textit{minimal exposure} if it shares a vertex with a previous layer.
Minimal and Maximal Exposure

The polygons with minimal exposure are marked with $m$’s, and those with maximal exposure are marked with $M$’s.
This figure shows how recursive calls in the replication work starting at polygon A. Polygon vertices are numbered in counter-clockwise order with vertex $i$ at the right end of edge $i$ looking outward.
The Top-level “Driver” for Replication

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 ) ;
    }
}
Utilities to Support Replication

Functions to compute transformations, based on tranMult() which multiplies two transformations and returns the product.

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

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

Arrays that control replication.

```
pShiftArray[] = { 1, 0 } ;
verticesToSkipArray[] = { 3, 2 } ;
qShiftArray[] = { 0, -1 } ;
polygonsToSkipArray[] = { 2, 3 } ;
exposureArray[] = { MAX_EXP, MIN_EXP } ;
```
The Recursive `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 Using \{6,4\}
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
Future Work

- Allow vertices at infinity.
- Create a program to transform between different fundamental polygons.
- Automatically generate patterns with color symmetry.