Designing an algorithm that transforms each pixel back to motif in a fundamental region.

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

DNYANESHWARI SUBODH CHANDARANA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Adviser: Dr. Douglas Dunham

September 2011

Acknowledgements

I would sincerely like to thank Dr. Douglas Dunham for persistent guidance, and support at times of doubt. I also thank Dr. Gary Shute and Dr. Marshall Hampton for agreeing to serve on my committee. Appreciation is also due for the help I received from faculty members in the Computer Science department.

Finally, I would like to thank my mother and my fiancé for constant support and patience.

Dedicated to

my Mother

Mrs. Kirti Chandarana,

and my Fiancé

Aditya Mone

Abstract

Current algorithms to create repeating hyperbolic patterns transform the motif about the hyperbolic plane to points in the Poincaré circle model. This is inefficient near the bounding circle since the entire motif is drawn, even though it covers only few pixels.

To avoid this shortcoming, we designed another algorithm that transforms each pixel in a motif in a fundamental region and then colors the original pixel using a color permutation of the color of the final pixel. This solves the inefficiency problems of the previous algorithms.

Table of Contents

List of Figures

Chapter 1

Introduction

Euclidean geometry is a mathematical system attributed to the Greek mathematician Euclid of Alexandria, who's "Elements" is the earliest known systematic discussion of geometry. Euclid's method consists in assuming a small set of intuitively appealing axioms and deducing many other propositions from these.

While many of Euclid's findings in the Elements had been previously stated by earlier Greek mathematicians, Euclid is credited with developing the first comprehensive deductive system. Euclid's approach to geometry consisted of proving all theorems from a finite number of postulates (axioms).

Euclidean Geometry is the study of flat 2-dimensional space, the plane. We can easily illustrate these geometrical concepts by drawing on a flat piece of paper or a chalkboard.

The concepts in Euclid's geometry remained unchallenged until the early 19th century. During that time, other forms of geometry started to emerge; they were called non-Euclidean geometries. It was no longer assumed that Euclid's geometry could be used to describe all physical space.

The main goal of this thesis is to design a pixel-based algorithm that replicates a repeating hyperbolic pattern by transforming each pixel back to motif in a fundamental region and then coloring the original pixel based on the transformed pixel within the motif.

Chapter 2

Different types of Geometry

2.1 Hyperbolic Geometry

In mathematics, hyperbolic geometry is a non-Euclidean geometry, meaning that the parallel postulate of Euclidean geometry is replaced. The parallel postulate in Euclidean geometry is equivalent to the statement that, in two dimensional space, for any given line R and point P not on R, there is exactly one line through P that does not intersect R; i.e., that is parallel to R. In hyperbolic geometry there are at least two distinct lines through P which do not intersect R, so the parallel postulate is false.

Basically, in hyperbolic geometry, all of Euclid's postulates hold, other than fifth postulate. Instead of the fifth postulate, an axiom called the *Hyperbolic axiom* is used.

According to the Hyperbolic axiom: If there exists a line l and a point P not on the line l, there are at least two distinct lines parallel to l passing through P.

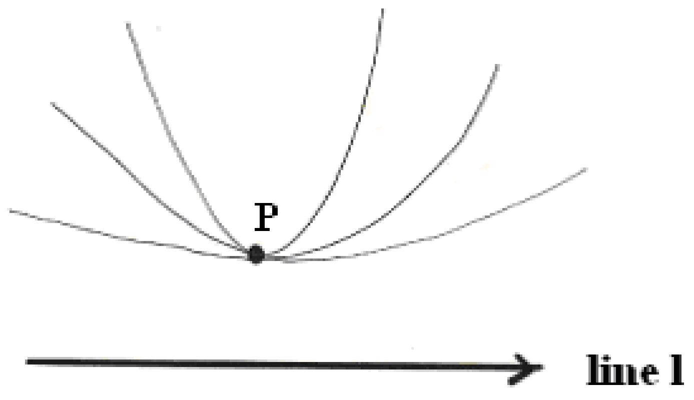

Figure 2.1: An illustration of the Hyperbolic Axiom

This postulate in turn, is used to prove and obtain many other useful results including:

- There exists a triangle whose angle sum is less than 180°.
- All quadrilaterals have angle sum less than 360°.
- If two triangles are similar, they are congruent, that is, if the angles of the triangle are equal, so are the sides.

To prove that any geometry is as consistent as Euclidean geometry, we need a Euclidean model of that geometry. A *model* is an interpretation of the primitive terms under which the axioms become true statements. Here interpretation does not allude to the "understanding of the meaning", but in a more basic sense of "giving a meaning".

2.2 Euclidean Geometry

Euclidean geometry is a mathematical system attributed to the Greek mathematician Euclid of Alexandria. Euclid's text "*Elements*" was the first systematic discussion of geometry. It has been one of the most influential books in history, as much for its method as for its mathematical content. The method consists of assuming a small set of intuitively appealing axioms, and then proving many other propositions (theorems) from those axioms. Although many of Euclid's results had been stated by earlier Greek mathematicians, Euclid was the first to show how these propositions could be fitted together into a comprehensive deductive and logical system.

For over two thousand years, the adjective "Euclidean" was unnecessary because no other sort of geometry had been conceived. Euclid's axioms seemed so intuitively obvious that any theorem proved from them was deemed true in an absolute sense.

Today, however, many other self-consistent geometries are known, the first ones having been discovered in the early 19th century. It also is no longer taken for granted that Euclidean geometry described physical space. An implication of Einstein's theory of general relativity is that Euclidean geometry is only a good approximation to the properties of physical space if the gravitational field is not too strong.

Euclidean geometry is an axiomatic system, in which all theorems are derived from a finite number of axioms. Euclid gives five postulates (axioms)

1) Any two points can be joined by a straight line.
2) Any straight line segment can be extended indefinitely in a straight line.
3) Given any straight line segment, a circle can be drawn having the segment as radius and one endpoint as center.
4) All right angles are congruent.
5) Parallel Postulate: If two lines intersect a third line in such a way that the sum of the inner angles on one side is less than the two right angles, then the two lines inevitably must intersect each other on that side if extended far enough.

2.3 Non Euclidean Geometry

A non-Euclidean geometry is the study of shapes and constructions that do not map directly to any n-dimensional Euclidean system characterized by a non-vanishing Riemann curvature tensor. Examples of non-Euclidean geometries include hyperbolic and elliptic geometry, which are contrasted with Euclidean geometry.

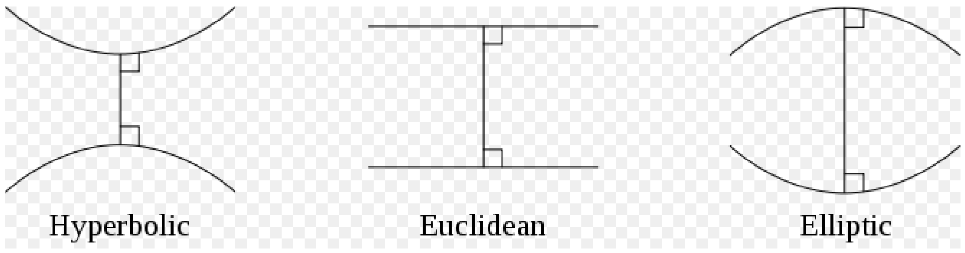

Fig 2.2: Lines bending away from, staying constant, and bending toward a common perpendicular in hyperbolic, Euclidean, and elliptic geometries.

There is a significant difference between the Euclidean and the hyperbolic geometry. In Euclidean geometry there exist parallel lines, lines that remain at a constant distance from each other even if extended to infinity; in hyperbolic two non-intersecting lines that have a common perpendicular curve away from each other by increasing distances as one moves farther from the points of the intersection with the common perpendicular; such lines are called ultra parallel.

There is significant difference between the Euclidean and non-Euclidean geometry. Non-Euclidean geometry uses a modification of Euclid's fifth postulate, which is also called the parallel postulate.

Euclid’s fifth postulate had always been thought of by mathematicians as special. Mathematicians never doubted its truth, but always thought of it as a theorem that could be proved from the other four.

By the latter half of the 18th century, this problem of proving the fifth postulate had become really famous and many mathematicians had attempted to prove it. It was only a matter of time before the difficulty of the problem would cause some to conclude that this problem was unsolvable. It did not in any way mean that Postulate 5 was not provable. But this unproven idea made the discovery of non-Euclidean geometry inevitable. The logical transition behind it was that since neutral geometry, the geometry of Euclid’s first four axioms, itself did not imply

Postulate 5, there must be a new geometry different from Euclid's based on the first four axioms and the negation of Postulate 5.

It has been observed many times in the history of science and mathematics, that when many people are working on the same problem, and when the communication between them is infrequent, it leads to multiple independent discoveries. In this sense, it seems that non-Euclidean geometry was discovered about four times in a span of twenty years. Apparently, Carl Friedrich Gauss was the first to discover non-Euclidean geometry. But he did not publish any papers related to his discoveries. He worked on this new geometry for many years and discovered many theorems.

In the meanwhile, he received a letter from Ferdinand Schweikart which indicated that Schweikart had himself discovered non-Euclidean geometry and had basically reached similar conclusions and results as Gauss.

Neither of them published any papers though, and hence, when Janós Bolyai published his work on non-Euclidean geometry in the

Appendix of his father’s book, it established the field of non-Euclidean geometry.

Bolyai was not the first person to have a paper published on non-Euclidean geometry. A Russian mathematics professor, Nocolai Lobachevsky had already published a paper on the topic. But since the paper was published in Russian, it was not well known in the European mathematical circles. Its translation into French and German later on reaffirmed Lobachevsky’s work and discoveries in the field of non-Euclidean geometry.

Non-Euclidean geometry is technically any geometry which is not Euclidean. One of the most useful non-Euclidean geometries is hyperbolic geometry, the geometry of hyperbolic space. This is the geometry discovered by Bolyai, Gauss, Lobachevsky and Schweikart [4]. Spherical or Elliptical geometry, the geometry of sphere, is another non-Euclidean geometry.

Chapter 3

Two Models of Hyperbolic geometry

3.1 The Weierstrass Model

Weierstrass demonstrated the consistency of hyperbolic geometry by constructing a model for it on a certain surface in Euclidean 3-space [1].

We now provide a brief development of this model following [1]. We will look at the interpretation of the primitive terms like “point”, “line”, “lies on” and “between” given by it. Before that, we need some preliminaries.

A “point” in real 3-dimensional space is represented by a triplet of real numbers such as $X = (x, y, z)$. Such a triplet can also represent a vector in 3-space. This vector can be visualized as an arrow from the origin $O = (0, 0, 0)$ to X. This lets us use the terms “vector” and “point” interchangeably.

Consider two vectors $X_1 = (x_1, y_1, z_1)$ and $X_2 = (x_2, y_2, z_2)$. Their *Euclidean inner product or dot product* is defined as:

$X_1.X_2 = x_1x_2 + y_1y_2 + z_1z_2$

To construct the model we also need another inner product, called the *hyperbolic inner product*. For the above two vectors it can be defined as

$<X_1,X_2> = x_1x_2 + y_1y_2 - z_1z_2$

(3.1)

Two vectors X and Y are said to be *e-orthogonal* if $X.Y = 0$ and *h-orthogonal* if $<X, Y> = 0$.

Now the equation of the plane in R^3 is of the form $<A, X> = b$, where A is an *h-normal* of the plane. This means that A is h-orthogonal to any point that lies on this plane [5].

We are now ready to develop the model. As noted earlier we provide interpretations of “point”, “line”, “lies on” and “between”. We will also use the symbol R^3 to mean the Euclidean 3-space.

A “point” is defined to be a point (or vector) $X = (x, y, z)$ of R^3 such that

$<X, X> = -k^2$ and $z > 0$. Such points comprise the “upper sheet” of a two sheeted hyperboloid in R^3. We call this sheet H^2. The constant k is a measure of the curvature of the hyperbolic plane.

A “line” is defined to be the intersection of H^2 with a plane through the origin of R^3. We know from Euclidean geometry that such intersection is one branch of a hyperbola (Fig. 3.1).

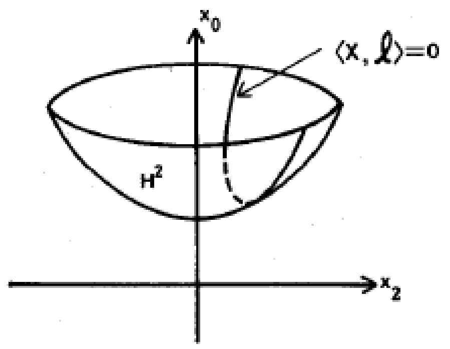

Figure 3.1: A “line” in the Weierstrass Model [1]

Now, a plane through the origin is given by an equation of the form $<X,l> = 0$, where l is an h-normal of the plane. Since l is an h-normal, any scalar multiple of l will also be an h-normal to the plane. We shall choose l in such a way that $<l, l> = k^2$. This means that l is a point of the single-sheeted hyperboloid with the equation $<X, X> = k^2$. (Fig. 3.2).

Summarizing, a "line" is a section of H^2 by a plane through the origin of R^3. There is a vector l, such that $<l, l> = k^2$ and the line consists of all X of H^2 such that $<X, l> = 0$.

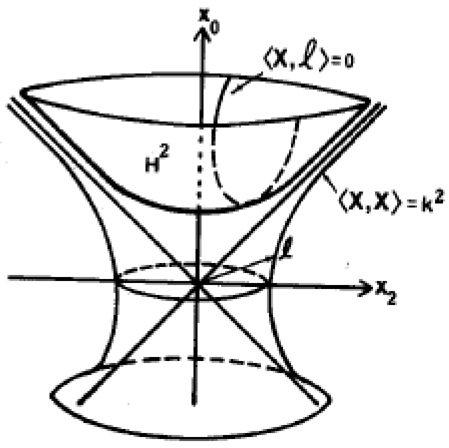

Figure 3.2: The vector l on the hyperboloid of one sheet $<X, X> = k^2$ corresponds to a "line" in the Weierstrass Model [1].

"Lines" are thus in one-to-one correspondence with the vectors l on $<X, X> = k^2$. We shall now refer to a "line" by either giving its equation $<X, l> = 0$ or by just giving its vector l.

A point X "lies on" the line if and only if $<X, l> = 0$.

Given three distinct points A, B and C on the line, we will say that C "lies between" A and B if, when the line is traversed in either direction, the three points are encountered in one of the orders *ACB* or *BCA*.

Now we define a distance on H^2.

The Lobachevskian distance between two points *P* and *Q* of H^2 is the unique non–negative number $d = d_{PQ}$ satisfying

$$\cosh \frac{d}{k} = -\frac{1}{k^2} < P, Q >$$

(3.2)

With distance thus defined, we can look at the concept of an *equidistant curve.* We saw that a “line” in the Weierstrass model is defined by the section of H^2 cut by a plane passing through origin. *Equidistant curves* to one such line are defined as sections of H^2 cut by planes that are parallel to the defining plane of the line. Note that the term “parallel” is used in the Euclidean sense here.

This completes discussion of the concepts of the Weierstrass model that are relevant to the present work. The inquisitive reader may refer to [1] for a detailed treatment of the model.

3.2 The Poincaré Disk Model

This model was formulated by the French mathematician Henri Poincaré. It is an example of a finite model for hyperbolic geometry and is conformal in nature. This means that angles are represented faithfully in this model. However, distance is distorted. We will now look at how this model interprets the primitive terms [5].

A “point” in the Poincaré Model is defined to be a point $X = (x, y)$ in a Euclidean 2-space (R^2), such that $x^2 + y^2 < 1$. Such points comprise the interior of a “bounding” circle in R^2 with center at the origin and unit radius. Note that the points on the circumference are not included in the model. We shall refer to this interior as D or “The Poincaré Disk” interchangeably.

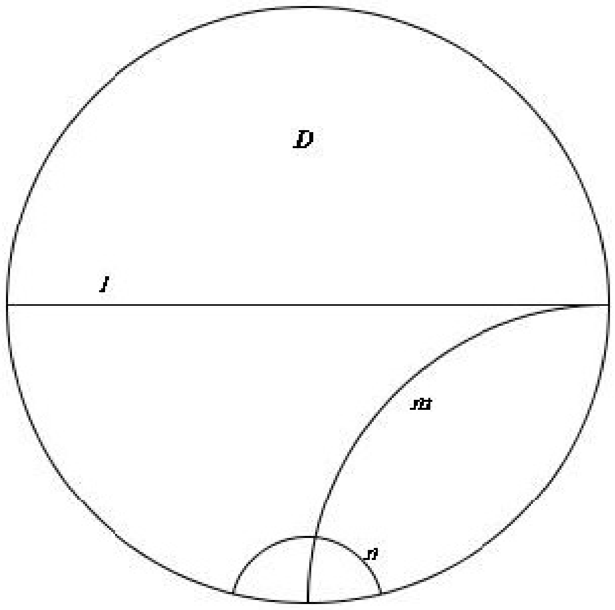

Figure 3.3: The Poincaré model, showing lines *l*, *m* and *n*. We say *l* and *n* are *divergently parallel or ultra-parallel*, *m* and *n* are intersecting and *l* and *m* are *asymptotically parallel*.

A "line" is defined to be a set of interior points of *D* that constitute an arc of a circle that is orthogonal to *D* (Fig 3.3). Open diameters of *D* are also defined as "lines" (open diameters can be thought of as arcs of a circle of infinite radius and obviously intersect *D* orthogonally).

The term "lies on" and "between" have the same meaning as in the Euclidean case in this model.

Since the Poincaré Model is conformal, the angle between two intersecting “lines” can be measured by measuring the angle between the tangents to the “lines” at the point of intersection.

For more on hyperbolic geometry and the Poincaré model, see [2].

Chapter 4

Research Problem

4.1 Description

The problem is to design an algorithm which transforms each pixel back to a motif in a fundamental region and then color the original pixel based on the transformed pixel within the motif.

Initially a PPM file was generated. Just for convenience the PPM file is filled up with black color and then a bounding circle is drawn. The next step is to draw a motif made up of polygons, filling those polygons using a polygon fill algorithm.

4.2 Pixel Algorithm

Pixel replicate ( )

For each V=(x,y) in circle

Apply the transformations across the edges of the central p-gon (which define the symmetry group of the pattern) to get new points $V_1, V_2, \ldots, V_p$

Then new V= The one of (V_1,V_2,…,V_p) that is closest to center of the disk, that is (0,0)

Repeat until V is in the central p-gon

Then color the original point (x, y) based on the color at the final location. The color of (x,y) is the inverse of the color permutation part of the cumulative transformation applied to the color of the final point.

So the algorithm is:

```
For each $V_{orig}$ (x, y) in the bounding circle
    Let V = $V_{orig}$
    While (V not in p-gon)
        For i=1 to P
            Apply transformation (edgeTrans) $T_1$,$T_2$,….$T_p$ to
            V to get $V_i = T_i$ times V
        Let new V= $V_i$ that is closest to the origin
Color $V_{orig}$ according to the color of the final V
```

Certain functions that are called in the algorithm which use vector based data files are listed below

1) readMotif () – reads the original vector-based motif file.
2) genPgonPat ( ) – builds the vector pattern within a p-gon using copies of the motif.
3) drawPgon ( ) – Converts the vector p-gon pattern to a pixel version.

In the past Dr.Dunham and his students have created many vector based data files containing motifs that work with a vector- based replication program.

4.3 Executing the Program

The program needs a data file on which to operate. The name of the data file can be entered on the command line (e.g.: % design cl1.dat). A brief description of the format of the data file is contained in the file DATA_FORMAT as described in the appendix.

Step 1: Compile the program by using the command “make” on command line

Step 2: Then type the following command “./design -s 800 c12.dat”

The user can use any of the .dat files in place of cl2.dat to get desired output.

Chapter 5

Results

We show the output of the previously existing vector based program for each data file, followed by the output of the new pixel-based program, using the same data file. Note that the vector based algorithm does not fill the bounding circle.

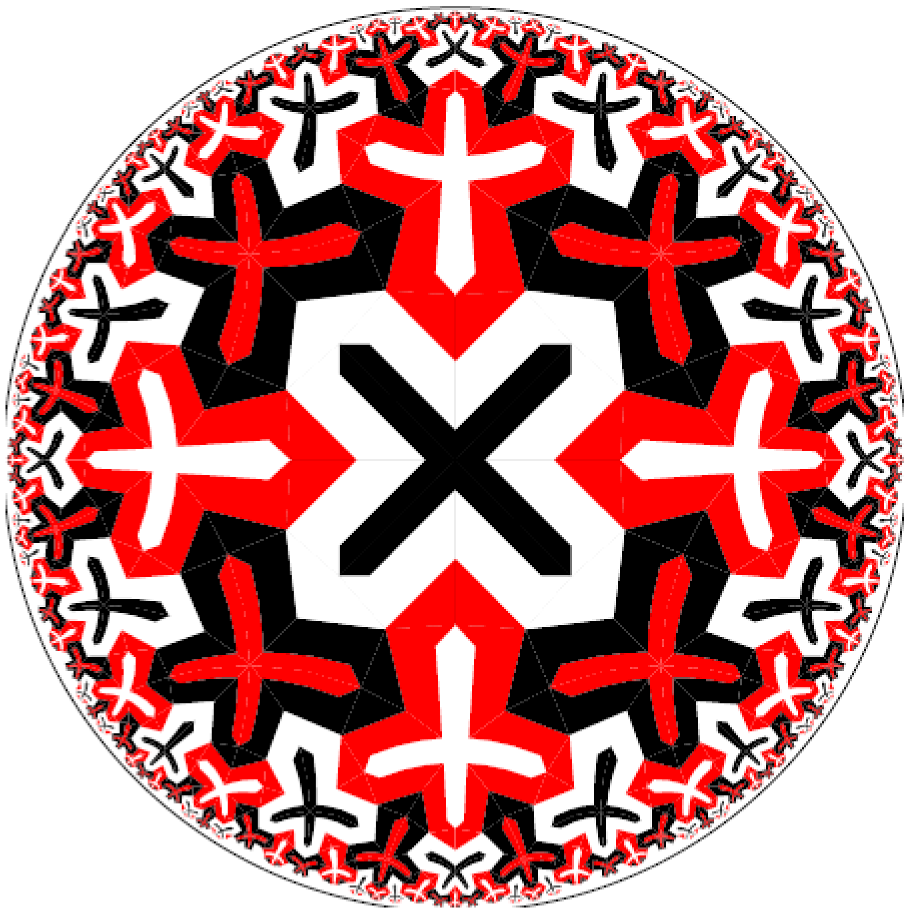

Figure 5.1: Escher ‘s circle Limit II pattern [Vector version]

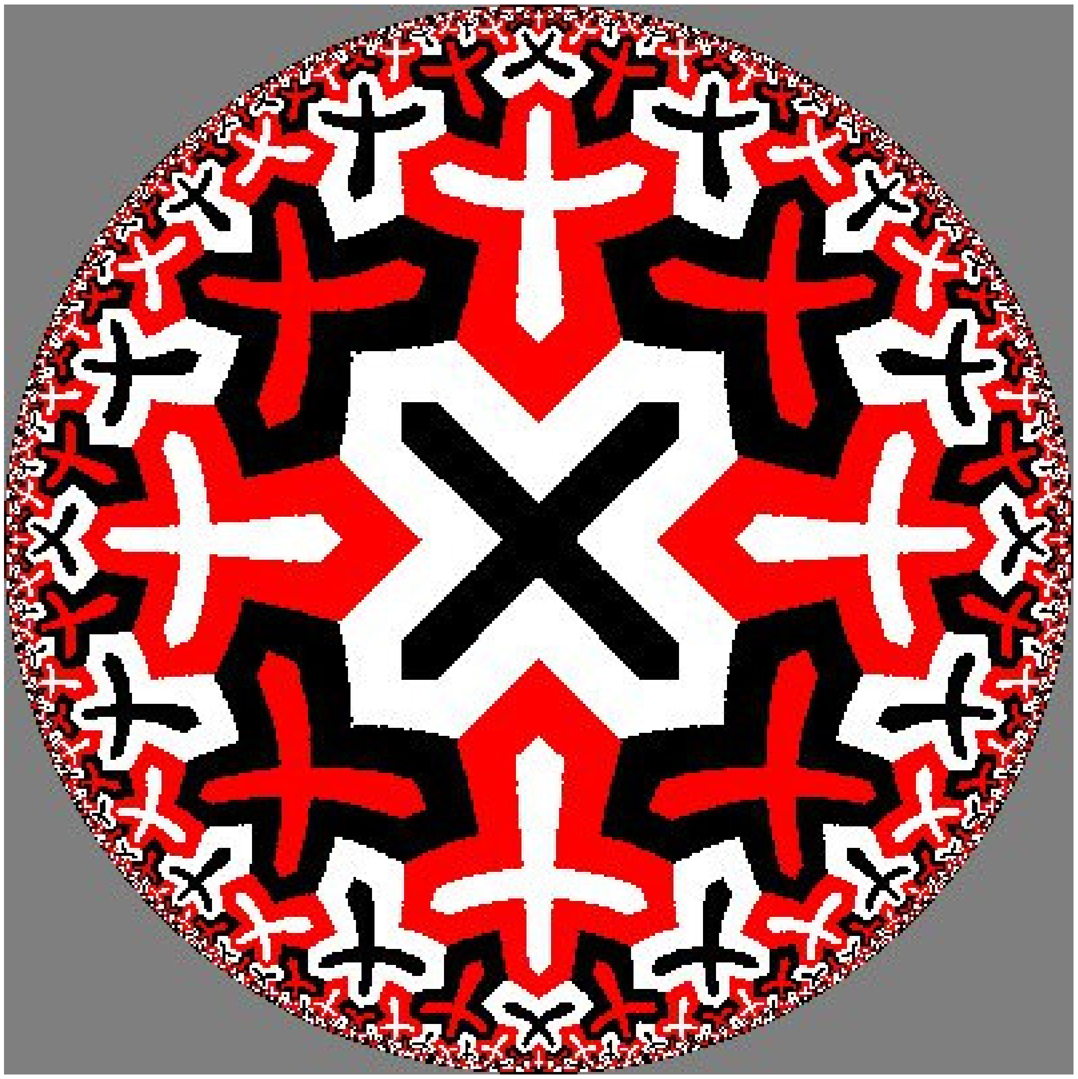

Figure 5.2: Escher‘s circle Limit II pattern [Pixel version]

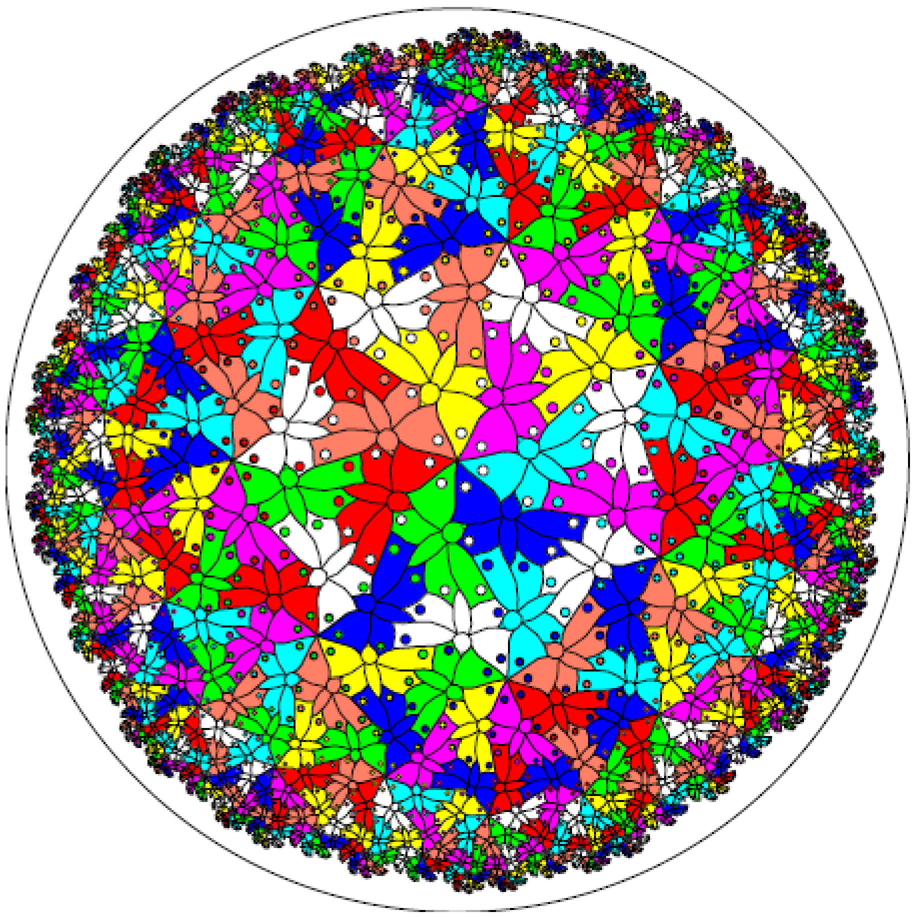

Figure 5.3: A pattern of butterflies based on Escher's Regular Division Drawing 70 [Vector version]

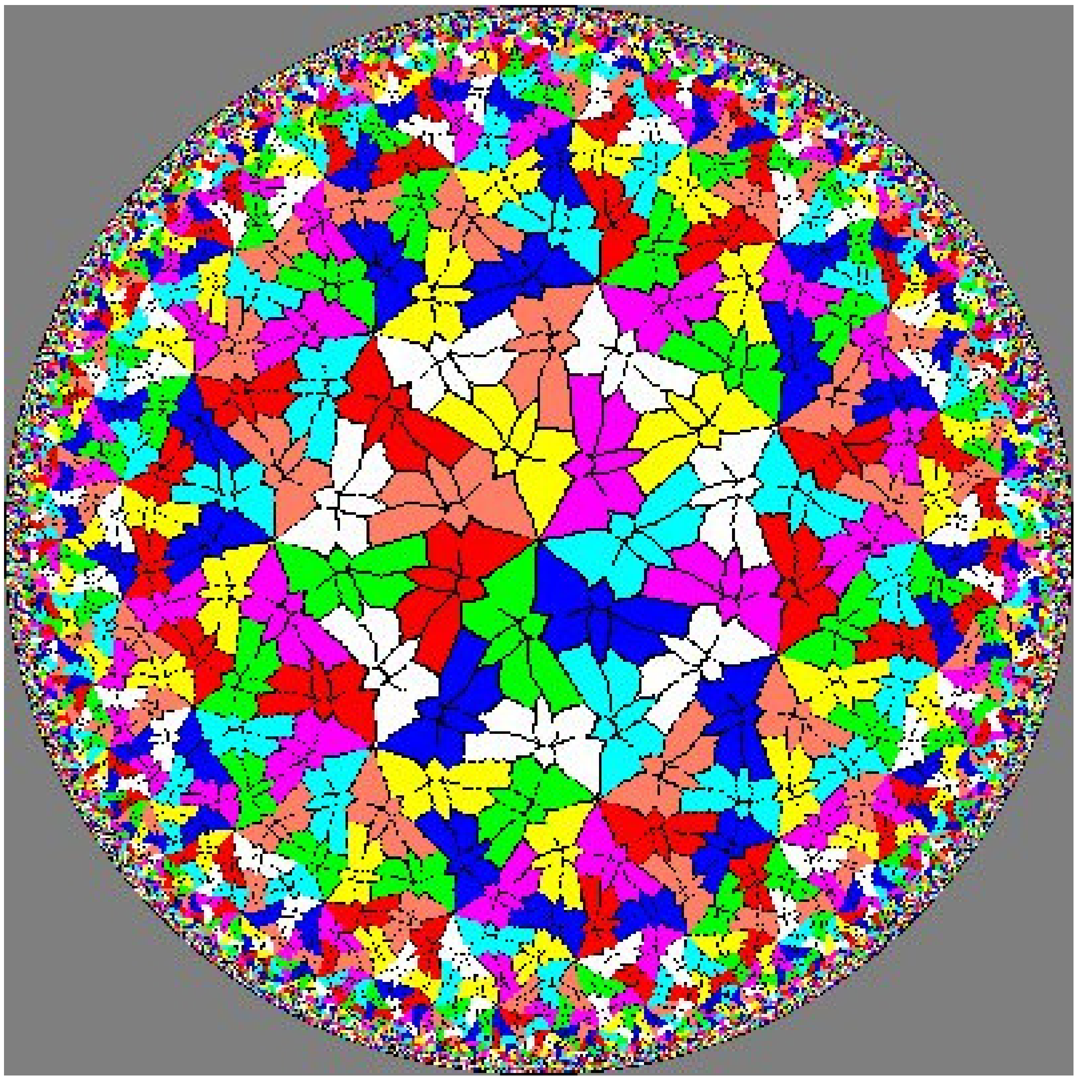

Figure 5.4: A pattern of butterflies based on Escher's Regular Division Drawing 70 [Pixel version]

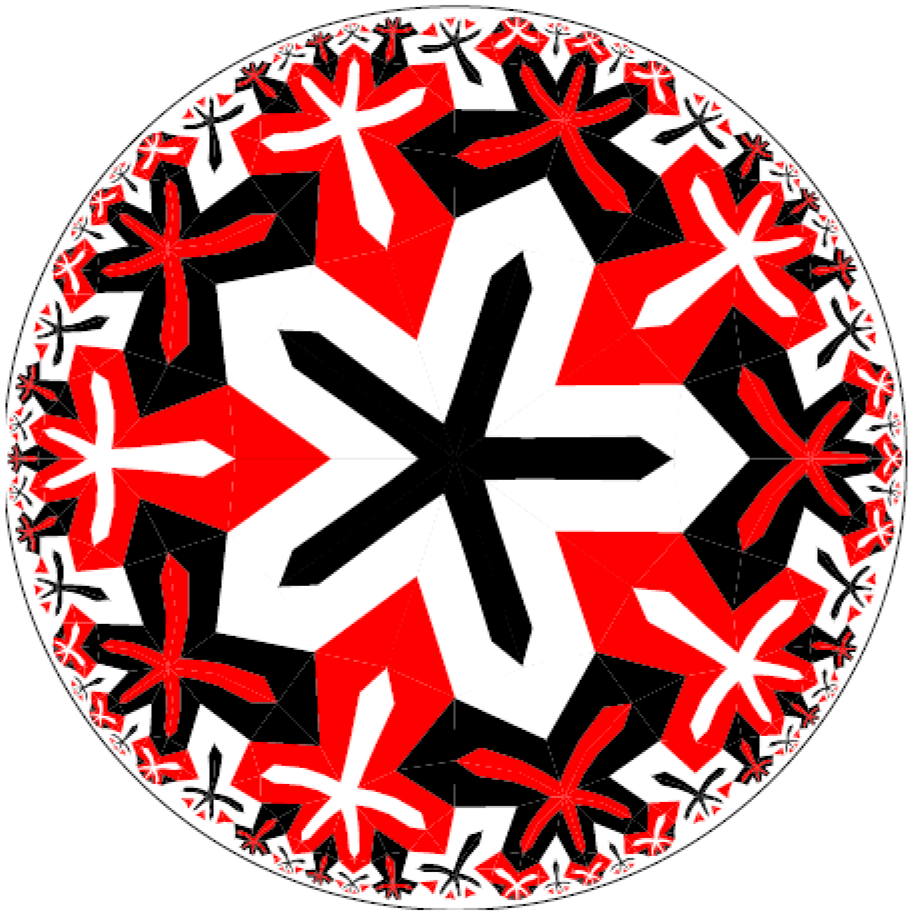

Figure 5.5: A five-armed cross pattern similar to Escher's Circle Limit II pattern [Vector version]

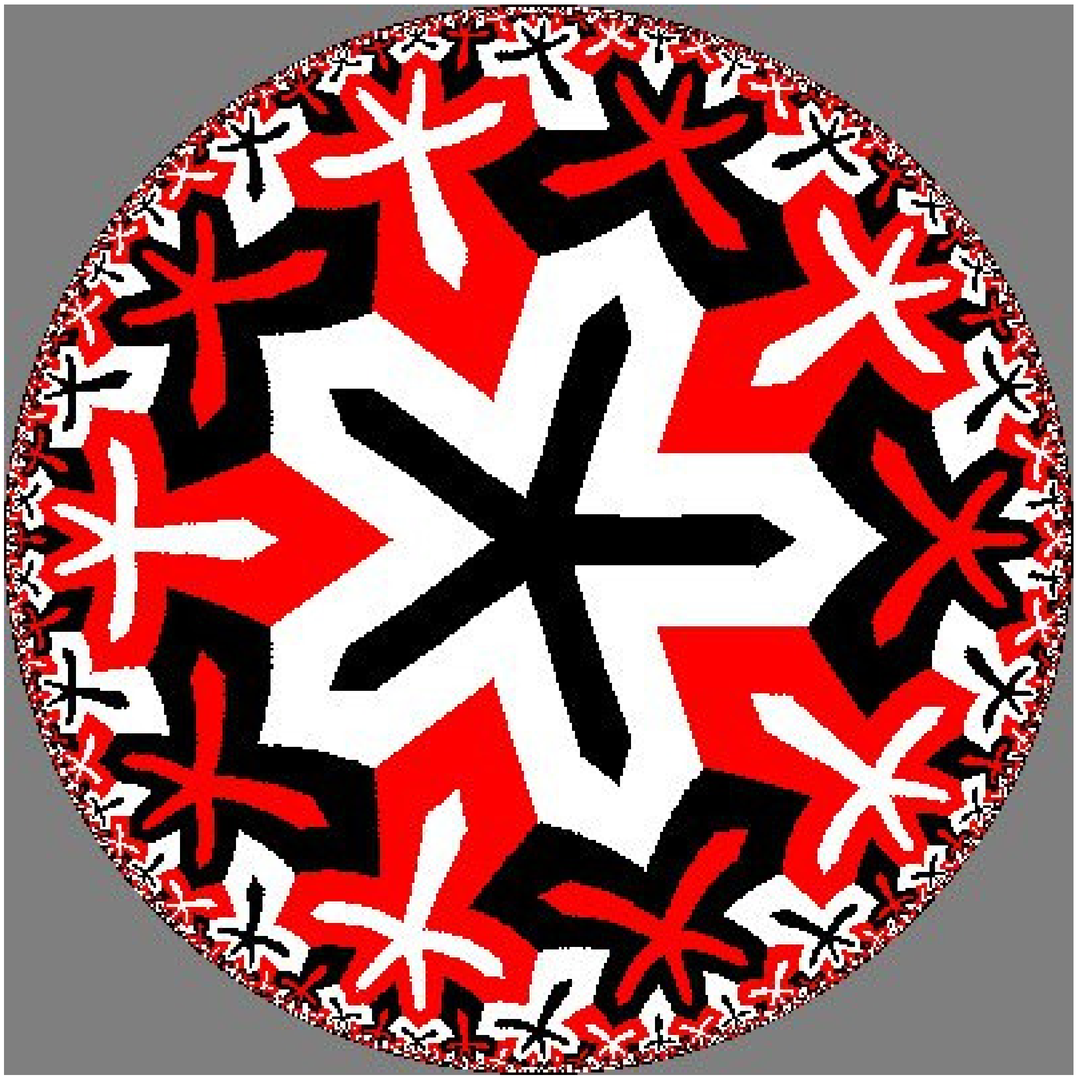

Figure 5.6: A five-armed cross pattern similar to Escher's Circle Limit II pattern [Pixel version]

Chapter 6

Conclusion

We have designed a pixel-based algorithm that replicates repeating hyperbolic patterns, including those with color symmetry, that are based on regular hyperbolic tessellations {p,q}.

This involved first converting vector-based data files to raster data that covers the enough of the center of the bounding circle that when the algorithm maps a point to be colored into that central region, the color of the original point can be determined.

The algorithm itself works by trying all the transformations across the edges of the central p-gon and using the one that moves the point closest to center of the bounding circle. This process is repeated until the "moving point" is in the central region where the color is known. The color of the original point can then be determined by inverting the color permutation part of the cumulative transformation.

Chapter 7

Future Work

We would like to add the capability for importing any pixmap into the central region so that the image of that pixmap could be replicated.

For example, the image of a photograph could be replicated. Currently, only motifs that are defined by vector data files can be replicated.

Also, it may be more efficient to iterate over a larger number of transformations at each step. This seems to be worth investigating

References

[1] R. Faber, "Foundations of Euclidean and Non-Euclidean Geometry",
Marcel Dekker, 1983.

[2] M. Greenberg, "Euclidean and Non-Euclidean Geometries: Development and History", 5th Ed., W.H. Freeman and Company, 2008.

[3] Kapoor T. K. (2005). Generating Repeating Hyperbolic Patterns based on {p, ∞}.

[4] Trudeau Richard J. (1987). The Non-Euclidean Revolution. Birkhäuser, Boston.

[5] Ajit Marathe (2007). Incorporating Points at Infinity in a Hyperbolic Drawing Program

Appendix 1

Data Format

Here we explain the format of the vector based data file that the program uses. In most cases, it is not necessary to understand how it works. Unfortunately, the design program is not at all robust, so that any error in a data file will probably cause the program to crash.

This program can only create hyperbolic patterns that are based on the regular tessellations (p, q) of regular p-sided polygons meeting q at a vertex ((p-2) (q-2) must be > 4 to be hyperbolic). For simplicity, we will call a regular p-sided polygon a "pgon" and we always arrange that one of them, the "central pgon" is centered within the bounding circle.

Here is a sample data file, "cl2.dat" that creates Escher's Circle Limit II pattern:

```
8 3 2 0 3 1
1 2 3
1 2 3
2 2 3 1
1 3 1 2
```

```
4 2 3 1
3 3 1 2
6 2 3 1
5 3 1 2
8 2 3 1
7 3 1 2
20
0.000000e+00 0.000000e+00 1 4 6
1.368406e-01 1.379250e-01 1 5 6
2.575625e-01 2.575625e-01 1 5 6
2.575625e-01 1.893719e-01 1 5 6
1.837032e-01 1.160219e-01 1 5 6
1.215594e-01 5.387817e-02 1 5 6
6.756561e-02 0.000000e+00 1 5 6
0.000000e+00 0.000000e+00 1 6 6
6.756561e-02 0.000000e+00 2 4 6
1.215594e-01 5.387817e-02 2 5 6
1.837032e-01 1.160219e-01 2 5 6
2.575625e-01 1.893719e-01 2 5 6
2.575625e-01 2.575625e-01 2 5 6
3.747407e-01 1.552227e-01 2 5 6
2.188499e-01 0.000000e+00 2 5 6
6.756561e-02 0.000000e+00 2 6 6
2.188499e-01 0.000000e+00 3 4 6
```

```
3.747407e-01 1.552227e-01 3 5 6
3.648853e-01 0.000000e+00 3 5 6
2.188499e-01 0.000000e+00 3 6 6
```

In the first line, 8 3 2 0 3 1

The first number is the value of p, i.e. p = 8 in this case.

The second number is the value of q, i.e. q = 3 in this case (thus this pattern is based on the tessellation (8.3)).

The third number, 2 in this case, is the number of "different" sides of the central p-sided polygon that are used to form the fundamental region that contains the motif (the other sides of the fundamental region are two radii from the center to two vertices of that p-sided polygon separated by 2*(2*pi/p)). This number must divide p, and p divided by this number is the number of copies of the motif that appears in the central p-sided polygon.

The fourth number, 0 is not used (it is there to maintain compatibility with older versions of this program).

The fifth number, 3 in this case, must be the highest "color" number of the colors used. The colors are numbered starting at 1 (black), 2(white), 3(red), 4(green), and so on.

The sixth number, 1 in this case, indicates the kind of reflection symmetry the pattern has within the central p-sided polygon:

0 indicates that there is no reflection symmetry (only rotation symmetry).

1 indicates that there is reflection symmetry across the perpendicular bisector of one of the edges of the p-sided polygon.

2 indicates that there is reflection symmetry across a radius (from the center to a vertex of the p-sided polygon).

The second line, 1 2 3, is the color permutation induced by rotating by 2*(2*pi/p) (i.e., the third number of line 1 times 2*(2*pi/p). Note that this is the “array” representation of permutations (not the “mathematical”) one using cycles): the values listed are the values of perm[1], perm[2], perm[3], etc., so 1 2 3 represents the identity permutation in this case.

The third line, 1 2 3, is the color permutation induced by the reflection if the sixth number of line 1 is 1 or 2 (it is just the identity if the sixth number is 0). In this case, 1 2 3 is the identity permutation.

The next p lines consist of a first number followed by a color permutation.

The first number of the first line indicates which edge (edge 2 in this case) of the transformed pgon should lie next to edge 1 of the central pgon. In general, if this first number is positive, the transformed pgon is rotated into position; if the number is negative, a reflection is used to move the transformed pgon into position. Note that the edges are numbered from 1 to p, not from 0 to p-1.

The color permutation of the first line, 2 3 1 (i.e. perm[1] = 2, perm[2] = 3, perm[3] = 1) is non-trivial in this case (i.e. black --> white --> red --> black)

The first number of the second line indicates which edge (edge 1 in this case) of the transformed pgon should lie next to edge 2 of the central pgon. In this case, the color permutation is 3 1 2.

The first number of the third line indicates which edge (edge 4 in this case) of the transformed pgon should lie next to edge 3 of the central pgon. In this case the color permutation is 2 3 1.

The first number of the fourth line indicates which edge (edge 3 in this case) of the transformed pgon should lie next to edge 4 of the central pgon. In this case, the color permutation is 3 1 2.

The first number of the eighth line indicates which edge (edge 7 in this case) of the transformed pgon should lie next to edge 8 of the central pgon. In this case, the color permutation is 3 1 2.

The next line consists of a single number, the number of points that make up the motif. It is 20 in this case.

Following that are 20 lines of 5 numbers each; each lines specifies one point. Each line has the following format:

x-coordinate, y-coordinate, color, type-of-point, number-of-layers

where:

the x- and y-coordinates are within the central pgon (and hence within the unit circle).

the color is one of 1, 2, ... , 10

the type-of-point is one of:

1 -- a "Move To"

2 -- a "Draw To"

3 -- a "Circle" (there should be two of these in succession)

4 -- start a (Euclidean) "Filled Poly"

5 -- continue a (Euclidean) "Filled Poly"

6 -- end a (Euclidean) "Filled Poly"

7 -- a "Hyperline" (there should be two of these in succession)

8 -- a "Filled Circle" (there should be two of these in succession)

9 -- start a (Euclidean) "Polyline"

10 -- continue a (Euclidean) "Polyline"

11 -- end a (Euclidean) "Polyline"

12 -- start a (hyperbolic) "Filled pgon"

13 -- continue a (hyperbolic) "Filled pgon"

14 -- end a (hyperbolic) "Filled pgon"

the number-of-layers is not used (is there for compatibility with previous versions of the program).

PPM basically stands for “Portable Pixel Map”. A PPM file consists of a sequence of one or more PPM images. There are no data, delimiters, or padding before, after, or between images.

Each PPM image consists of the following:

1. A "magic number" for identifying the file type. A ppm image's magic number is the two characters "P6".
2. Whitespace (blanks, TABs, CRs, LFs).
3. A width, formatted as ASCII characters in decimal.
4. Whitespace.
5. A height, again in ASCII decimal.
6. Whitespace.

7. The maximum color value (Maxval), again in ASCII decimal. Must be less than 65536 and more than zero; we use 255.
8. A single whitespace character (usually a newline).
9. A raster of Height rows, in order from top to bottom. Each row consists of Width pixels, in order from left to right. Each pixel is a triplet of red, green, and blue samples, in that order. Each sample is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first.

   A row of an image is horizontal. A column is vertical. The pixels in the image are square and contiguous.

   Here is an example of a small image in this format.

```
P3

# feep.ppm

 4 4
 15
  0  0  0    0  0  0    0  0  0   15  0 15
  0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

There is a newline character at the end of each of these lines.

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a PPM image.

All characters referred to herein are encoded in ASCII. "newline" refers the the character known in ASCII as Line Feed or LF. A "white space" character is space, CR, LF, TAB, VT, or FF (I.e. what the ANSI standard C isspace() function calls white space).

Actual code used in the program:

```
 /* This is the main algorithm that fills interior of bounding circle in
  *  FrameBuf[][] with colors calculated by iteration.
  */
void IntReplicate ()
{
   int i, j, jstart, jfinish, k, kmin, imin, jmin, color ;
   double xc, yc, r, root ;
   double x, y, xw, yw, zw, s, xwt, ywt, zwt, xt, yt, xmin, ymin ;
   transformation t, tcum ;
   colorpermtype colorperm ;

   xc = yc = (ARRAY_SIZE - 1) / 2.0 ;      /* Center of the circle */
   r = (ARRAY_SIZE - 1)/2.0 - 0.0001 ;  /* Make circle a bit smaller */

   /* For each pixel inside the circle, map it back to a known color */
   for ( i = 1 ; i <= ARRAY_SIZE - 2 ; i++ ) {
      root = sqrt( r*r - (i - yc)*(i - yc) ) ;
      jstart  = (int)ceil( xc - root ) ;
      jfinish = (int)floor( xc + root ) ;

      for ( j = jstart ; j <= jfinish ; j++ ) {
         /* Find the color of  FrameBuf[y][x] */
         /* Convert FrameBuf indices i, j to disk Poincare coords */
         y = (2.0 * i) / ((double)(ARRAY_SIZE - 1)) - 1.0 ;
         x = 1.0 - (2.0 * j) / ((double)(ARRAY_SIZE - 1)) ;

         /* s is used for loop test and to find 3D Weierstrass coords
*/
         s = x*x + y*y ;
         kmin = 0 ; /* intialize kmin - may not be needed */
         tcum = IDENT ; /* initialize the cumulative transformation */

         while ( s > xqpt*xqpt + yqpt*yqpt ) {

            /* Convert unit disk Poincare coords to
               3D Weierstrass coords */
            xw =   2.0 * x / (1.0 - s) ;
            yw =   2.0 * y / (1.0 - s) ;
            zw = (1.0 + s) / (1.0 - s) ;

            for ( k = 0 ; k < p ; k++ ) {
               /* Try each of the transformations across p-gon edges */
               t = edgeTran[k] ;
               /* multiply the matrix part of t times
                *(transpose of) vector [xw, yw, zw] to get
                *(transpose of) [xw', yw', zw']
                * map this back down to Poincare disk via:
                   x' = xw' / (1.0 + zw')
                   y' = yw' / (1.0 + zw')
                * see if point (x', y') is closer to origin than (x,y)
                * or any other previous (x', y').
                * if so, let (xmin, ymin) = (x', y') & kmin = k
                */
         xwt = t.matrix[1][1]*xw+t.matrix[1][2]*yw+t.matrix[1][3]*zw ;
         ywt = t.matrix[2][1]*xw+t.matrix[2][2]*yw+t.matrix[2][3]*zw ;
         zwt = t.matrix[3][1]*xw+t.matrix[3][2]*yw+t.matrix[3][3]*zw ;
               xt = xwt / (1.0 + zwt) ;
               yt = ywt / (1.0 + zwt) ;

               /* xt, yt is closer, update */
               if ( xt*xt + yt*yt < s ) {
                  s = xt*xt + yt*yt ;
                  x = xt ; y = yt ; kmin = k ;
                  /* (x,y) is updated to the new minimum point */
               }
            }
            tmult ( &tcum, edgeTran[kmin], tcum ) ;
         }

         /* convert (x, j) to FrameBuf (imin,jmin) coords */
         jmin = Round( (x + 1.0)*(ARRAY_SIZE - 1)/2.0 ) ;
         imin = Round( (1.0 - y)*(ARRAY_SIZE - 1)/2.0 ) ;

         /* find color permutation from tcum & invert it */
         invertperm ( &colorperm, tcum.colorperm ) ;

         /* Write the pixel in FrameBuf[][] based on the source pixel
*/
         WritePixel( i, j, colorperm[ FBufSrc[imin][jmin] ] ) ;

      }

   }

}
```