ACM Multimedia 2000

Intuitive and Effective Design of Periodic Symmetric Tiles

ERGUN AKLEMAN [*]
Visualization Laboratory
College of Architecture
Texas A&M University

JIANER CHEN [†]
Department of Computer Science
College of Engineering
Texas A&M University

BURAK MERIC
Knowledge Based
Information Systems, Inc.

Abstract

This paper presents a new approach for intuitive and effective design of periodic symmetric tiles. We observe that planar graphs can effectively represent symmetric tiles and graph drawing provides an intuitive paradigm for designing symmetric tiles. Moreover, based on our theoretical work to represent hexagonal symmetry by rectangular symmetry, we are able to present all symmetric tiles as graphs embedded on a torus and based on simple modulo operations. This approach enables us to develop a simple and efficient algorithm, which has been implemented in Java. By using this software, designers, architects and artists can create interesting symmetric tiles directly on the web. We also have designed a few examples of symmetric tiles to show the effectiveness of the approach.

1 Introduction

The symmetric patterns in Alhambra, Granada are probably the most well-known architectural usage of symmetric patterns. In fact, symmetric patterns have been a part of the architectural world throughout the history and frequently used by almost every civilization in wallpapers and wall decorations, ceilings, floor tiles, street pavements and even facades of the buildings [4] as shown in Figure 1, 2 and 3.

Although, there has been a great interest in art and architecture, the theoretical classification of periodic symmetric patterns did not began until the early twentieth century when Russian crystallographer E. S. Fedorov enumerated the seventeen two-dimensional periodic symmetry groups. These groups today are also known as *wallpaper groups, periodic*

[*]Address: 216 Langford Center, College Station, Texas 77843-3137. email: ergun@viz.tamu.edu. Supported in part by the Texas A&M, Scholarly & Creative Activities Program.

[†]Address: Department of Computer Science, College Station, TX 77843-3112. email: chen@cs.tamu.edu. Supported in part by the National Science Foundation under Grant CCR-9613805.

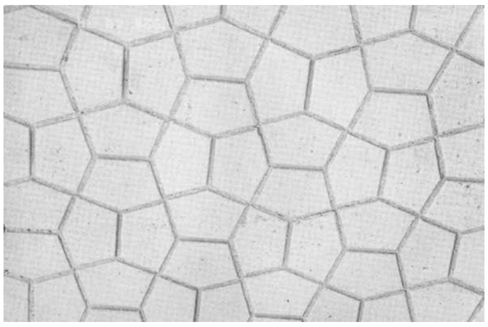

Figure 1: Street Pavements in Japan [2].

groups or *(plane) crystallographics groups* [3]. Fedorov's result shows that, mathematically, there are only seventeen distinct types of patterns which have different symmetries. Since the paper of Fedorov was written in Russian the classification of the 17 symmetry groups were not known until the work of Niggli and Polya in 1924.

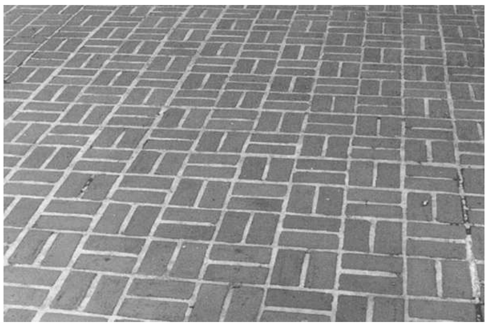

Figure 2: Paving patterns in Annapolis, Maryland [2].

From the designer's perspective, the most important implication of Fedorov, Niggli and Polya's work is the identification of the symmetry groups as a set of distinct symmetry operations. This identification of symmetry operations encouraged artists such as M. C. Escher, F. Briss and K. Mehmedov to discover new and interesting patterns. The most famous drawings of symmetric patterns were created by M. C. Escher [1] and his works are still extremely popular.

Although, the knowledge of 17 symmetry groups helps the design of symmetric patterns, it is still difficult to find *interesting tiles* with paper and pen. In other words, even with the knowledge of 17 symmetry groups when using only paper and pen to design symmetric patterns the artistic talent is still important. The idea of using interesting symmetric tiles has a great use in architecture, art, science and education. Therefore, it is important to find interactive computational approaches to simplify the design of interesting tiles.

With the development of computer graphics, many interactive systems to design symmetric patterns have been developed. Most of existing symmetric pattern design systems are based

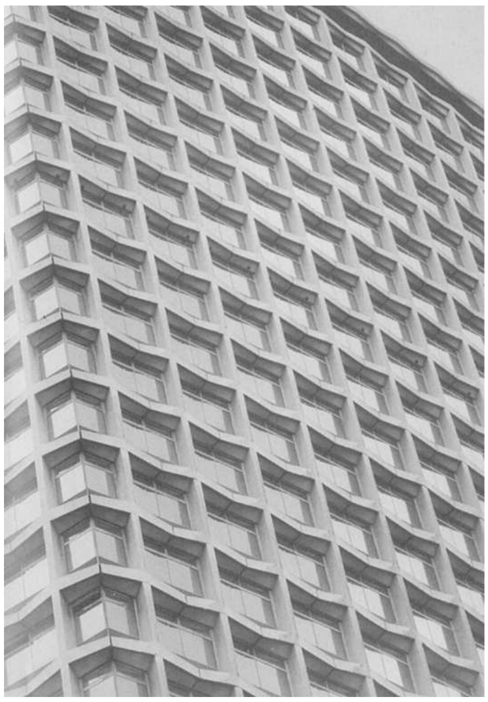

Figure 3: Uniform facades in Centre Point building in London, U.K. [2].

on painting paradigm. For designing tiles, these systems are not fundamentally different than paper and pen. In a painting system, even if the users want to make small changes, they must erase existing images and redraw new ones.

One alternative approach is to use drawing paradigm. Kali developed by N. Amenta at University of Minnesota Geometry Center is an example of a system that uses drawing paradigm [4]. Since drawing is based on objects such as points, lines and polygons that can be translated, scaled or rotated interactively, it is easy to change the shapes of the times. At the first look, this approach seems to be the appropriate choice. Unfortunately, we observe that even drawing paradigm is not appropriate for symmetric tile design. Tiles are polygons that come together without gaps and overlaps, on the other hand, drawing paradigm supports gaps and overlaps. The user has to be extremely careful not to include any gap or overlap when designing symmetric tiles.

We observed that symmetric tiles are inherently graphs embedded on surfaces without edge crossings, where lines are edges and line junctions are vertices, as shown in Figure 4. In other words, for symmetric tiles the internal representation must support not only points, lines and polygons but also graphs.

We, therefore, propose that graph drawing paradigm is the most appropriate approach to develop a graph based algorithmic approach to design symmetric tiles.

In this work we have developed a graph based approach for designing symmetric tiles. In

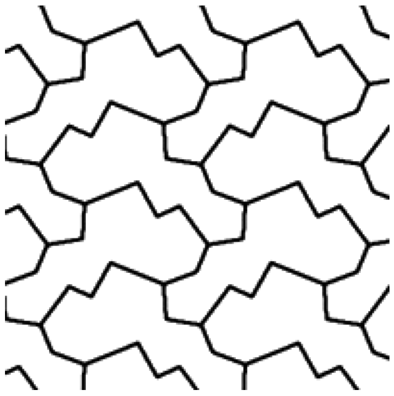

Figure 4: Symmetric tiles are inherently planar graphs.

our approach, the graphs that represent symmetric tiles are constructed by users by drawing line segments. Each line segment corresponds to an edge of the underlying graph and endpoints of the lines give the vertices of the graph. The graph representation is constructed by attaching endpoints. In other words, when two vertices come close, they will snap together and become one vertex. We also provide intersection prevention to ensure the graph edges never intersect. We also provide an option that removes this constraint to enhance flexibility and improve useability.

Based on this approach, we have developed a system. By using our system one can produce drawings composed of graphs. These graphs can be modified interactively by moving the vertices, by adding new edges, and by dividing edges. In order to create better drawings collisions need to be avoided. For implementation we have chosen the Java programming language. That allows us to make use of the object oriented aspects and to make our work available on the internet.

2 Seventeen Planar Symmetries

As we have mentioned in introduction, there exist seventeen distinct symmetries. In literature [3], the periodic symmetry groups are called as *p1, p2, p4, pm, pmm, p4m, p4m, cm, cmm, pg, pmg, pgg, p4g, p3, p6, p3m1, p31m* and *p6m* . Each one of these symmetry groups is a collection of symmetry operations: translation, rotation, reflection and glide reflection. The rotations can be either period 2, 3, 4 or 6. These operations are known as isometries which preserve the distance of any two points and they can uniformly be represented by a 3×3 matrix.

The complete list of the 17 symmetry groups [2] in plane can be classified in two categories: rectangular and hexagonal symmetries (12 of these 17 groups have rectangular symmetries and 5 of them have hexagonal symmetries.). The rectangular and hexagonal symmetries are shown in Figures 5 and 6.

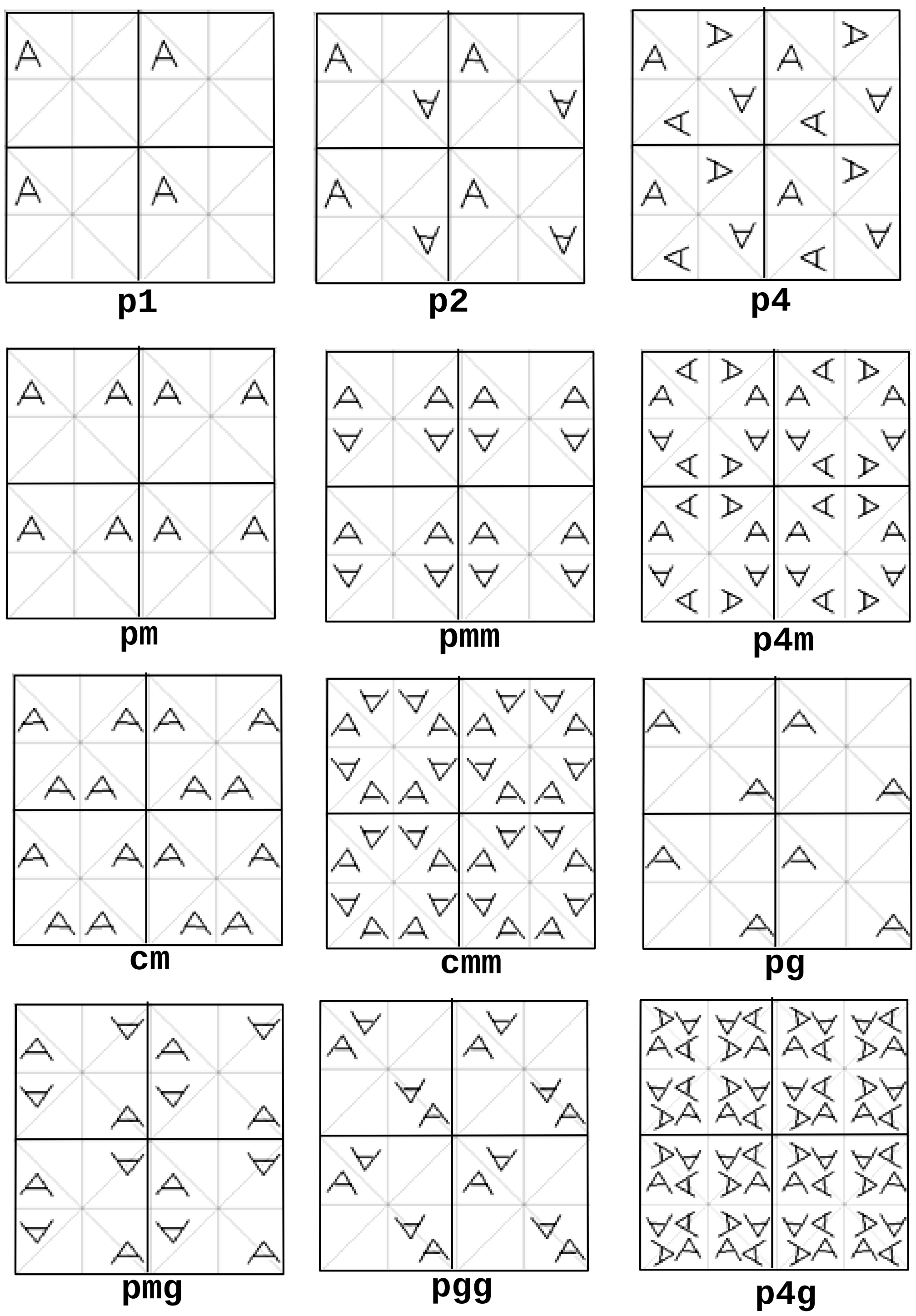

Figure 5: Rectangular symmetries.

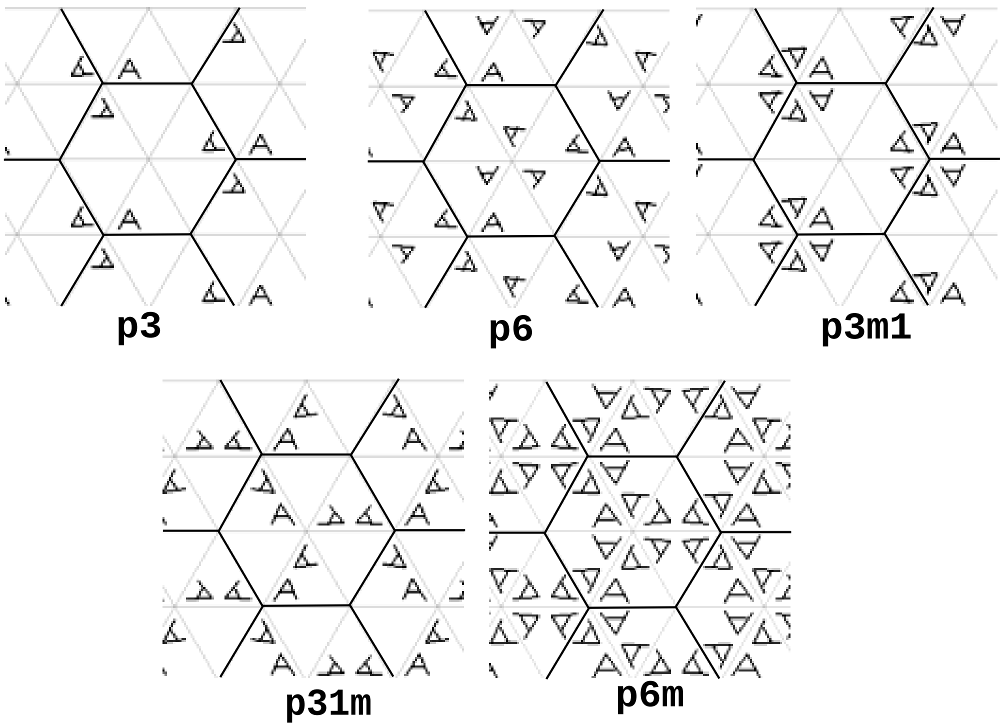

Figure 6: Hexagonal symmetries.

3 Methodology

For some symmetry groups, the drawing operations require checking all the vertices and edges. Checking all vertices and edges is not possible since the graphs that represent symmetric tiles have infinitely many vertices and edges. Fortunately, for symmetric tiles, it is not necessary to check all vertices and edges. We observe that since symmetric tiles can be generated by repeating a unit block, the whole graph can be represented by a simple unit graph and all the operations can be done on this simple graph.

1. Based on this observation, we have developed a simple algorithm (and supporting data structure) for drawing rectangular symmetries.

2. We show that the drawing algorithm for rectangular symmetries can also be applied to hexagonal symmetries with a minor modification.

3. We also show that, based on this drawing algorithm, collisions can easily be detected.

3.1 Drawing Algorithm for Rectangular Symmetries

As the names suggest, unit blocks for rectangular symmetries are rectangles and unit blocks for hexagonal symmetries are hexagons.

As known from topology [1], rectangular symmetric tiling is a covering space of the rectangle with identified opposite sides, which is topologically equivalent to a torus as shown in Figure7. Because of this property, rectangular symmetries can easily be obtained by modulo operations. It is, therefore, especially easy to develop an algorithm for rectangular symmetries since for rectangular symmetries it is possible to simply repeat a rectangle with modulo operations. This rectangle is called the unit cell. With the unit cell we get all information on the drawing and are able to perform operations based on this information. So, instead of dealing with the whole drawing we only need to consider the objects in a single cell.

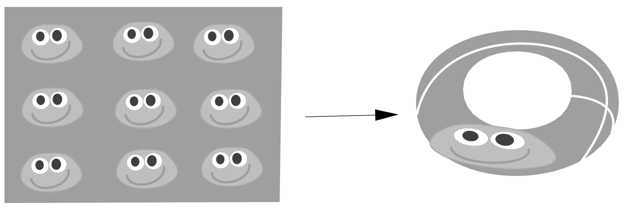

Figure 7: The repeating rectangles can be embedded to a toroidal surface.

Based on this observation, the algorithm becomes extremely simple. We use two graph data structures. The first one represents the actual drawing and the other one (unit graph embedded over toroidal surface) represents rectangular unit blocks. We call these internal representations ”drawing” and ”unit graph” representations. Unit graph representation is obtained in two stages from drawing representation.

1. For a given rectangular symmetry group, first related symmetry operations, i.e. a combination of translation, rotation, reflection and glide reflection operations are applied. As a result of these symmetry operations a set of new lines are computed.

2. By using mod operations in x and y directions, these lines are cut into shorter pieces that exactly fit inside of unit cell.

Figure 8 shows a collection of lines drawn by the user and Figure 9 shows the corresponding lines in the unit cell for symmetry group PG [3]. Once the unit graph is obtained, the final drawing is obtained by simply translating all the lines in unit graph as shown in Figure 10.

Note that both ”drawing” and ”unit graph” representations are internal representations and they are completely hidden from users. During the design process, users have only to deal with the symmetric tiles such as the one shown in Figure 10. As a result, they can focus mainly on the design of interesting tiles without the need to know the internal representations and the symmetry group they are currently using.

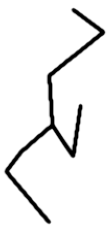

Figure 8: Drawing.

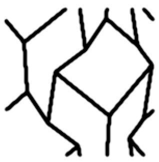

Figure 9: Unit graph related to drawing in Figure 8.

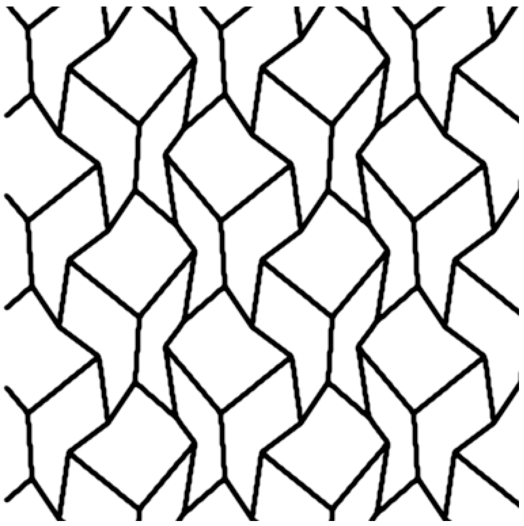

Figure 10: The symmetric tiles related to drawing in Figure 8.

3.2 Drawing Algorithm for Hexagonal Symmetries

Hexagonal unit blocks do not simply provide convenient operations as given in rectangular unit blocks. (In fact, It is well known [3] that rectangles with identified opposite sides give a torus while hexagon does not correspond to any valid topological surface which is a closed compact 2-manifold.)

One of our contributions in this paper is the development of a unified approach to include hexagonal symmetries. In order to avoid hexagons, we look for simpler unit cells for hexagonal symmetries and show that hexagonal symmetries can also be represented by a repeating rectangular unit block which is shown in Figure 12. In other words, we will prove that regardless of the symmetry group, all symmetric tiles can be represented by a simple *unit graph* which can be embedded over a toroidal surface, thus can be constructed by also using modulo operations.

Formally, let $\vec{v} = (x, y)$ be a vector. We say that a tiling T is *closed* under $\vec{v}$ if for any vector $\vec{w}$, the image at point $\vec{w}$ is the same as that at point $\vec{w} + \vec{v}$.

In particular, a *hexagonal symmetric tiling* H_r is a tiling that is closed under three vectors (where r is a fixed constant):

$$\vec{v}_1 = (0, r), \qquad \vec{v}_2 = (\frac{\sqrt{3}}{2}r, \frac{1}{2}), \qquad \vec{v}_3 = (\frac{\sqrt{3}}{2}r, -\frac{1}{2}) \tag{1}$$

A *rectanglar symmetric tiling* $R_{a,b}$ is a tiling that is closed under two vectors (where a and b are fixed constants):

$$\vec{v}_a = (a, 0), \qquad \vec{v}_b = (0, b) \tag{2}$$

Lemma 3.1 *A hexagonal symmetric tiling H_r is a rectanglar symmetric tiling $R_{\sqrt{3}r,r}$.*

PROOF. According to the definition, the hexagonal symmetric tiling H_r is closed under the vector $\vec{v}_r = (0, r)$. Thus, it suffices to show that H_r is also closed under the vector $\vec{v}_{\sqrt{3}r} = (\sqrt{3}r, 0)$.

It is easy to verify that

$$\vec{v}_{\sqrt{3}r} = (\sqrt{3}r, 0) = (\frac{\sqrt{3}}{2}r, \frac{1}{2}) + (\frac{\sqrt{3}}{2}r, -\frac{1}{2}) = \vec{v}_2 + \vec{v}_3$$

Thus, for any vector $\vec{w}$, we have

$$\vec{w} + \vec{v}_{\sqrt{3}r} = \vec{w} + \vec{v}_2 + \vec{v}_3 = \vec{w} + \vec{v}_3 = \vec{w}$$

The second equality is because H_r is closed under v_2 while the third equality is because H_r is closed under v_3.

This proves that the hexagonal symmetric tiling H_r is a rectanglar symmetric tiling $R_{\sqrt{3}r,r}$. See Figure 11 for a more intuitive illustration. $\square$

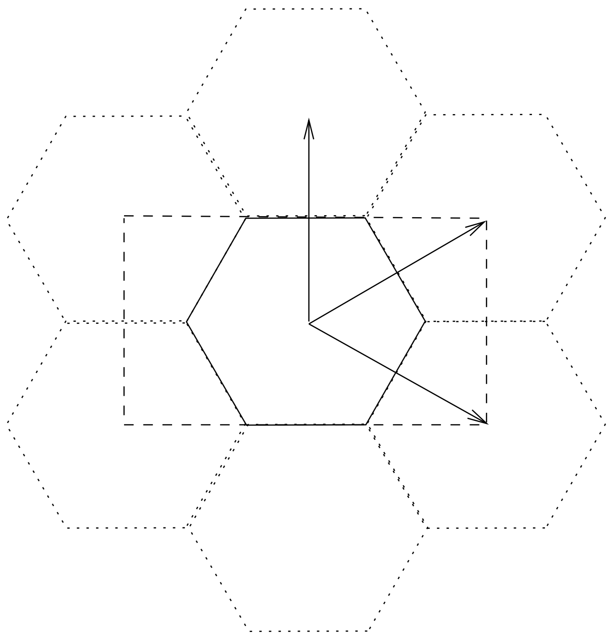

Figure 11: Hexagonal symmetry and rectanglar symmetry

An interesting question is the inverse of Lemma 3.1, i.e., when a rectanglar symmetric tiling corresponds to a hexagonal symmetric tiling. This, in fact, can also be investigated as follows.

We say that a rectangle of length $\sqrt{3}r$ and width r is *hexagonally symmetric* if it is symmetric in terms of its centered embedded hexagon (see Figure 12). In particular, in our construction of the rectanglar symmetric tiling $R_{\sqrt{3}r,r}$ for the hexagonal symmetric tiling H_r in Lemma 3.1, the basic rectangle of $R_{\sqrt{3}r,r}$ is hexagonally symmetric. In fact, we can prove the following stronger version.

Theorem 3.2 *A rectanglar symmetric tiling T is also hexagonally symmetric if and only if the basic rectangle of T is hexagonally symmetric.*

In this paper, we have omitted the formal proof of this theorem.

3.3 Collision Prevention

An important issue in creating symmetric planar graphs is the collision avoidance. In our framework, the collisions can be avoided by checking only the lines in the unit cell, not on the whole drawing. As a result, by preventing intersections we can come up with planar graphs. Following is a high level pseudo code of our system:

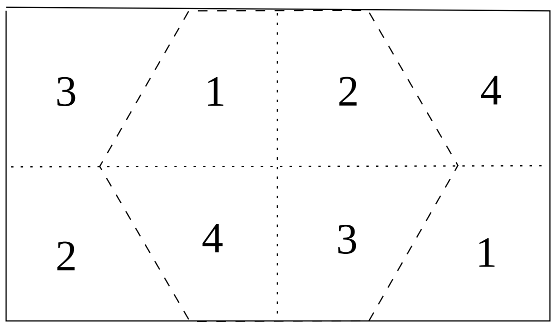

Figure 12: A hexagonally symmetric rectangle

```
**If** any modification on the drawing is made
**then if** there is no collision in the unit cell **then**
    translate the real line into the unit cell;
    create the converted lines;
    calculate symmetries;
    update data;
    generate the whole drawing by repeating the main unit cell
```

4 Implementation

We have implemented two versions of the drawing algorithm explained in the previous section, one with C and OpenGL and another one with Java. The Java version is available to the public. For anonymous review we have created an anonymous homepage that can be reached via internet from the address http://www.geocities.com/symmetrictiledesign/. Figure 13 shows the java interface of symmetric tile designer. As shown in Figure 13 interface supports moving existing lines, adding new lines, breaking lines into two lines. The system also provides collision avoidance.

The Java applet provides only drawing borders of the tiles. In order to color the tiles, users need to copy the resulting image and color the tiles by using a painting program such as Photoshop. Since the boundaries of each tile are clearly defined, it is easy to fill these tiles. Figures 14 and 15 show two colored examples of symmetric tiles we have designed using our software.

5 Conclusion

In this paper, we present a new approach for intuitive and effective design of periodic symmetric tiles. We observe that planar graphs can effectively represent symmetric tiles and

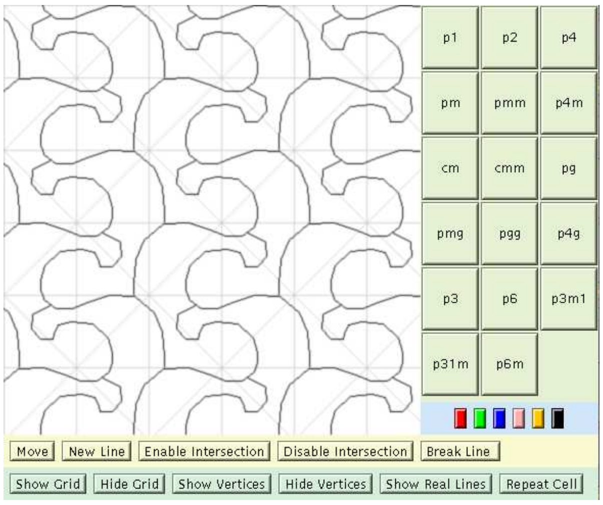

Figure 13: Java interface of Symmetric Tile Designer.

graph drawing provides an intuitive paradigm for designing symmetric tiles. Based on the observation that rectangular symmetry groups can be represented as graphs embedded on a torus, we have developed a simple drawing algorithm based on modulo operations. Moreover, based on our theoretical work to represent hexagonal symmetry by rectangular symmetry, we are able to present all symmetric tiles as graphs embedded on a torus. This result enables us to develop a simple and efficient drawing algorithm for all periodic symmetric groups. We also extended the algorithm to handle collision detection. We have implemented this algorithm both in Java and C. The java version is available in internet to all designers, architects and artists. By using this software, they can create interesting symmetric tiles directly on the web. We also have designed a few examples of symmetric tiles to show the effectiveness of the approach.

References

[1] Firby, P. A. and Gardiner C. F., *Surface Topology*, John Wiley and Sons Inc., New York, 1982.

[2] Hargittai I. and Hargittai M., *Symmetry, A Unifying Concept*, Shelter Publications, Inc. Bolinas, Ca, 1994.

[3] Grunbaum, B. and Shephard G. C., *Tilings and Patterns*, W. H. Freeman & Co., New York, 1987.

[4] *http://www.geom.umn.edu/java/Kali/*.

[5] Locher J. L., *M. C. Escher: His Life and Complete Graphic Work*, ed. Abrams, New York, 1982.

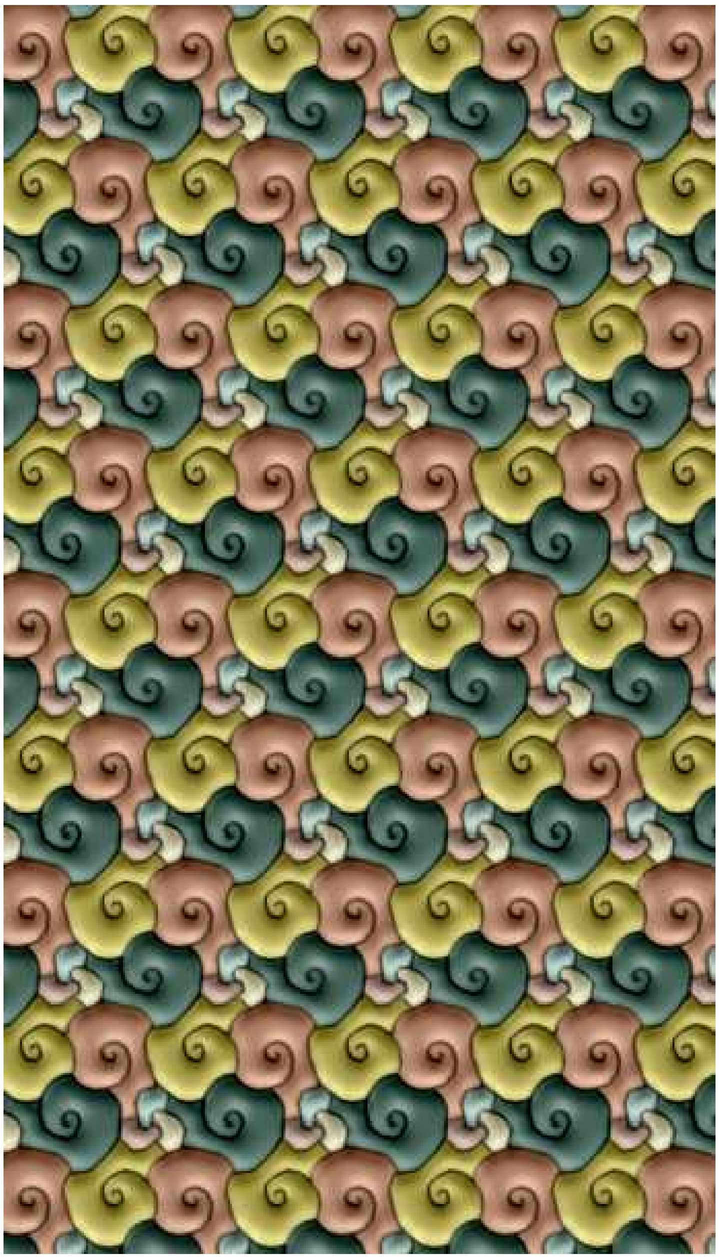

Figure 14: Snails.

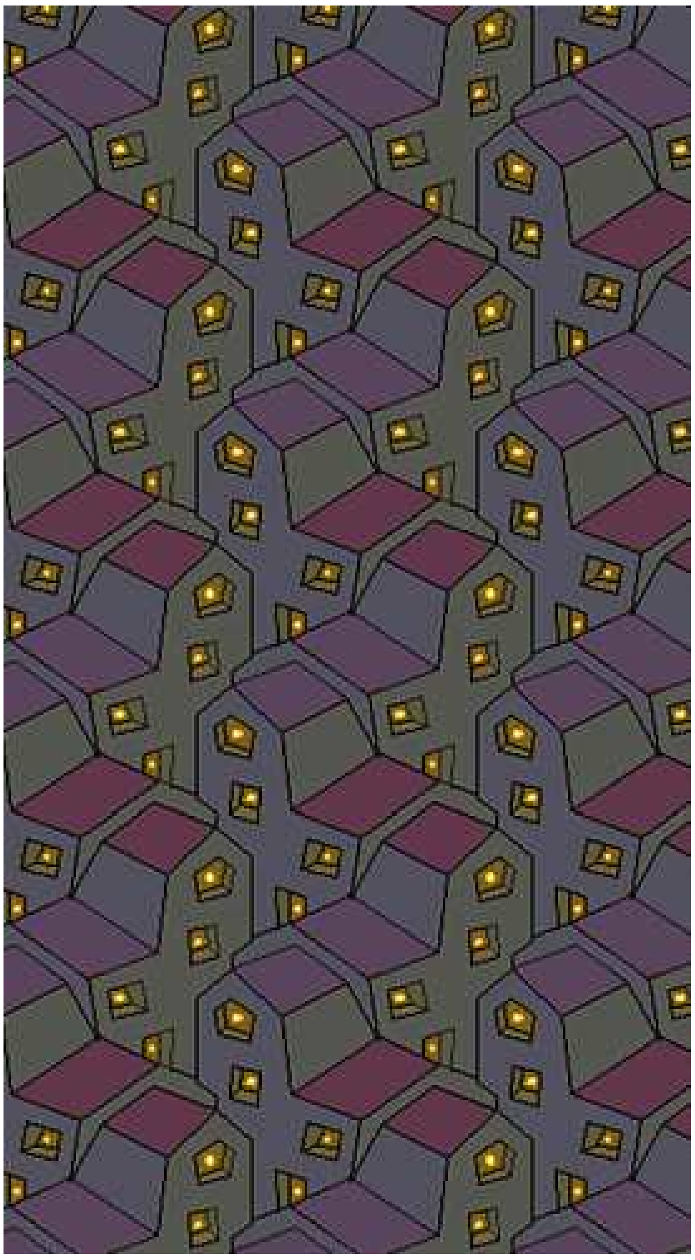

Figure 15: Mediterranean night.