Vaguely Islamic Aperiodic Patterns:
CSE 558 Project 1 Final Report

AJ Bernheim Craig Kaplan Doug Zongker

May 11, 1997

1 Problem Statement

Given an Islamic tiling on a periodic lattice, we want to create an aperiodic version of the tiling that preserves its visual appearance. We would like the resulting tiling to consist of the same basic polygons, but be aperiodic.

2 Difficult Aspects of the Problem

Given an arbitrary Islamic pattern it is not immediately apparent how to create an aperiodic version. For example, consider the Islamic pattern in Figure 1. What set of aperiodic tiles should be used to generate an aperiodic version? The pattern consists of three polygonal shapes: six-pointed stars, squares, and flattened rhombs; how should these polygons be placed onto the aperiodic tiles. Which polygons are crucial to the visual appearance of the tiling and which can be left out or modified if necessary?

For a particular pattern it may be possible to answer some of these questions using the general characteristics of patterns produced by the different aperiodic tile sets. For instance, Penrose rhombs produce patterns with a visual appearance of five-fold rotational symmetry while patterns using Amman rhombs have the appearance of eight-fold rotational symmetry. Figure 2 is an aperiodic design inspired by Figure 1 using the Penrose rhombs. The aperiodic version contains squares, flattened rhombs and five-pointed stars instead of the six-pointed stars of the original version, but still captures the flavor of Figure 1.

Thus, determining the appropriate tile set and decorations can sometimes be done by inspection, in general we know of no clear process for creating an aperiodic tiling reminiscent of a particular Islamic pattern.

Figure 1: An Islamic design from [1].

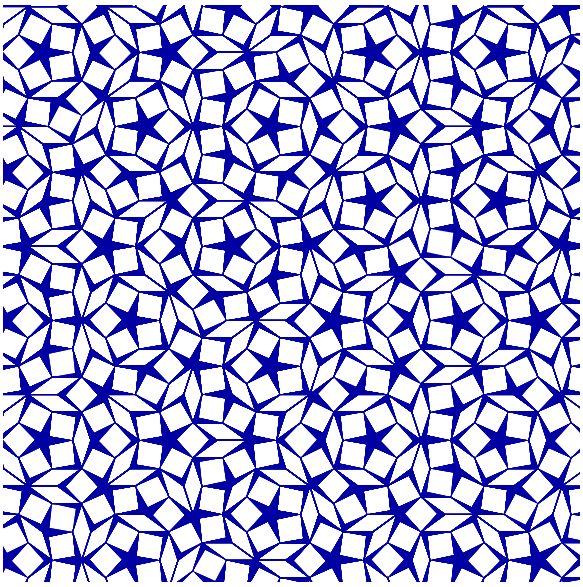


Figure 2: A tiling inspired by Figure 1, using Penrose rhombs.

3 Our Approach

In order to explore the set of possible aperiodic patterns and gain intuition about creating Islamic style aperiodic patterns, we implemented a system that allows the user to take tiles from various aperiodic tile sets and decorate them with lines and polygons. There are two major software components: the *editor* and the *tiler* (see Figure 3).

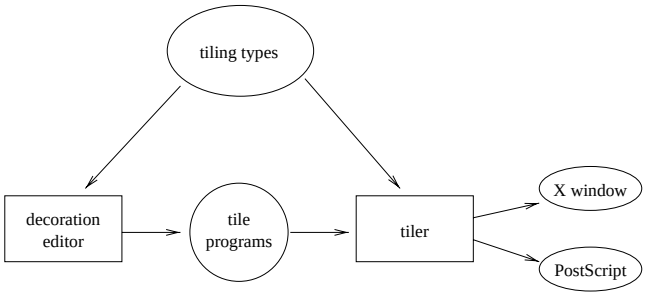


Figure 3: Block diagram of the system.

Each tiling type implemented has methods for giving information to the editor and the tiler. The editor needs to get the basic shape of the tile's outline from the tiling type. Each tile is represented in a standardized coordinate system convenient for that tile. The editor generates a list of instructions for drawing each type of tile (both the tile outline and the user's decoration) of the tiling.

The tiling type provides a list of tiles to the tiler. Each tile in the list has a transformation matrix mapping it from the tile's standard coordinate system to some location in the plane. The whole list of tiles covers the unit square $[-0.5, 0.5] \times [-0.5, 0.5]$. To draw the decorated tiling, the tiler iterates through this list. For each item in the list, it executes the tile-drawing program from the editor under the appropriate transformation matrix. Figure 4 illustrates this process.

The major difficulty in implementing this architecture was the generation of the tile list to cover the unit square. This was accomplished through deflation. A slider in the tiler window controls the number of iterations of deflation. Each tile provides an initial list of tiles which covers the desired square.

One deflation iteration consists of:

- Replacing each tile in the current list with a set of smaller tiles. This can be done by computing each new tile's transform matrix relative to the parent tile in the standard coordinate system, and composing this matrix with the parent tile's current transform. The child tiles must completely cover the parent tile (so that we preserve the invariant that the square is covered). They can extend outside the parent tile; see the next step.

- Eliminating duplicate tiles. Some tiling types, such as the Penrose kites and darts, break a parent into child half-tiles, then join the half-tiles to make the full child tiles. Our implementation has each parent generate a full tile (which extend outside the boundary of the parent), so a child tile lying on the boundary between two parents will be generated by both of them. We eliminate these by establishing an order on transformation matrices, sorting them, and checking adjacent pairs in the list to see if they are the same.

- Pruning tiles which lie outside the square. Since the deflation process leads to exponentially increasing number of tiles, we want to limit this as much as possible by throwing away those tiles that lie outside the square and will not contribute to the final picture.

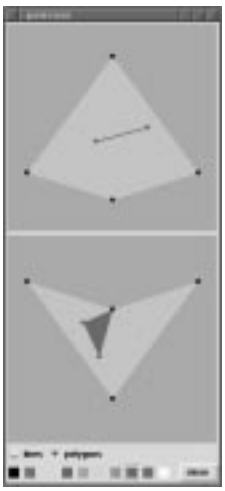
Figure 5 illustrates successive iterations of this process on the Penrose kites and darts tile set.

Since the tile list only needs to be reconstructed when the iterations slider changes, we can cache it to provide quick redraws when the user edits a tile.

Figure 6 shows designs created with the tool.

4 Future Work

Our tool is very general—it allows the user to create arbitrary decorations on the provided sets of aperiodic tiles. In a sense, this is a disadvantage. The user interface does not impose any constraints on the set of possible designs. As a result, it is potentially difficult to discover the subset of those designs that fit our intuitive notion of Islamic pattern. What is needed is a set of constraints in the user interface to guide the user in creating Islamic designs. At this time, it is



```
KITE:
  begin 0
  setcolour 0
  moveto -0.951063 0.309058
  lineto -0.000029 0.000049
  lineto 0.951005 0.309058
  lineto -0.000029 1.618043
  lineto -0.951063 0.309058
  begin 1
  setcolour 1
  moveto 0.391021 0.816867
  lineto -0.187733 0.660447
  <end>
```

```
DART:
  begin 0
  setcolour 0
  moveto -0.951063 1.309030
  lineto -0.000029 0.000044
  lineto 0.951005 1.309030
  lineto -0.000029 1.000020
  lineto -0.951063 1.309030
  begin 1
  fillcolour 5
  polybegin false
  polypoint -0.148628 0.466833
  polypoint -0.328511 0.850062
  polypoint -0.000029 1.000020
  polyend
  <end>
```

(c)

Figure 4: (a) A sample editor window. (b) The tile-drawing programs generated. (c) The tiling produced by executing the programs repeatedly with different transform matrices.

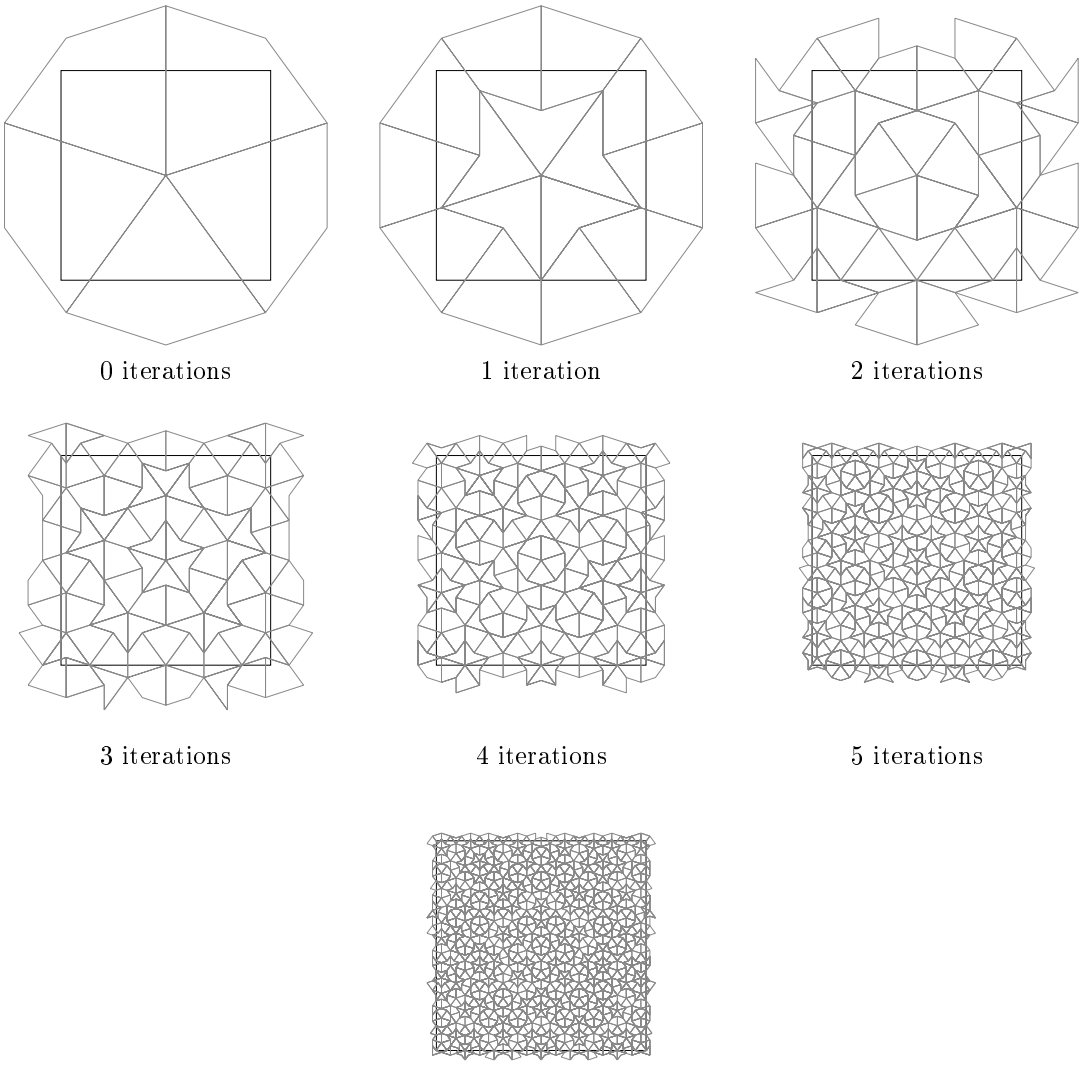


Figure 5: Results of successive iterations of deflation on the Penrose kites and darts patterns. The effect of throwing away tiles outside the square can be seen as the tiles get smaller.

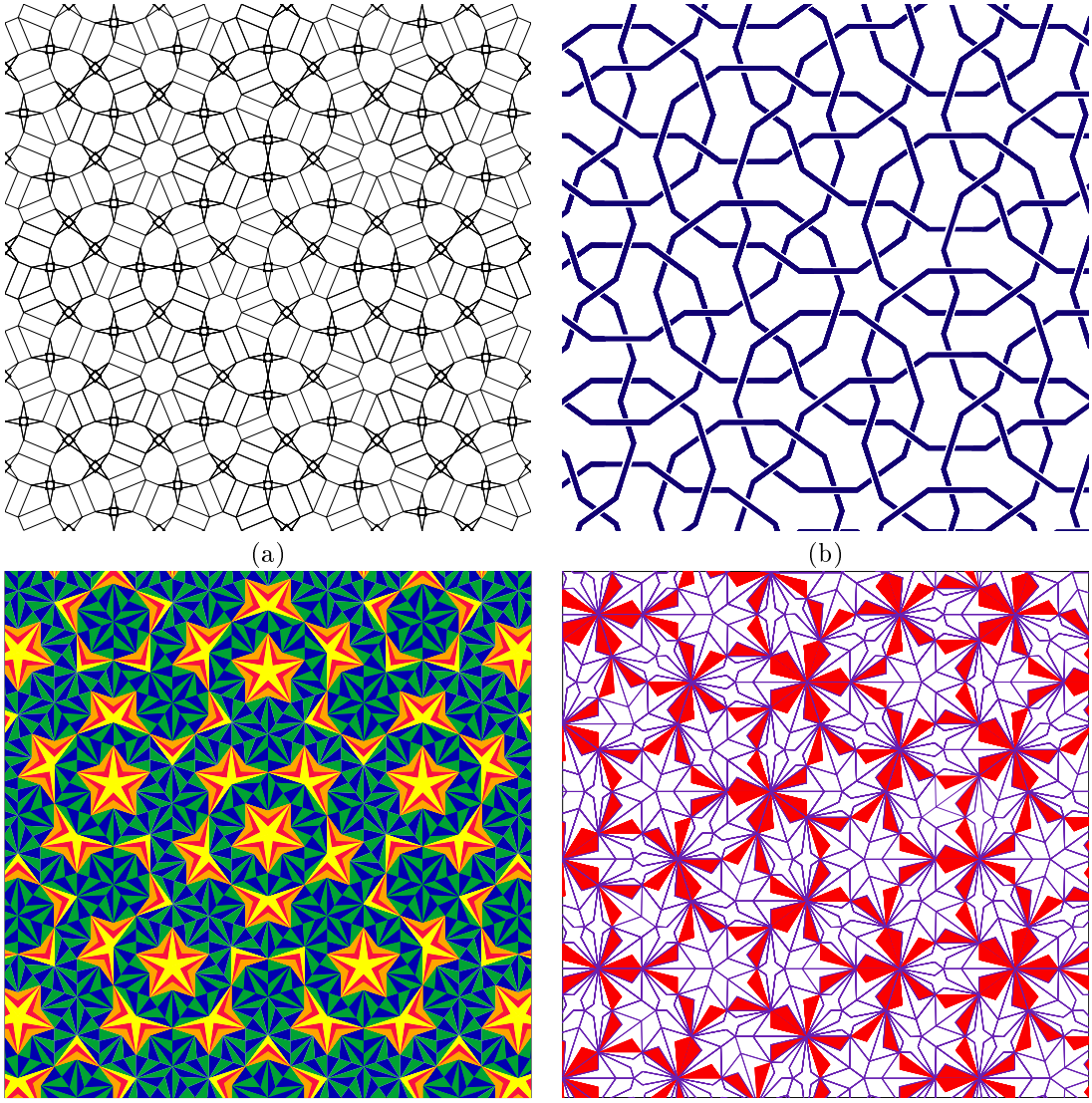


Figure 6: Some sample images created with the tool. The underlying tile sets are: (a) Amman rhombs, (b) and (d) Penrose rhombs, and (c) Penrose kites and darts.

unclear what these constraints should be. More experience with the existing tool would probably point the way to some heuristics that could be implemented.

Another shortcoming of our tool is that none of the currently implemented aperiodic tile sets contains a tile with more than four sides. We know of no aperiodic tile set that contains a convex tile of more than four sides and can tile an area through deflation. We believe that adding polygons with more sides would allow us to create a wider variety of Islamic designs, because so many existing designs are based on hexagons, octagons, and dodecagons [1].

Finally, we have not explored the possibility of implementing the grid method for creating quasiperiodic tilings, as described in lecture. It may be possible to create a large class of aperiodic designs simply by applying this simple transformation to any number of well-known tilings. Unfortunately, our current tiler relies on deflation to produce sets of tiles covering the unit square. While the grid method is conceptually simple and easy describe, determining the exact set of tiles produced and covering an area with those tiles is nontrivial.

References

[1] J. Bourgoin. *Arabic Geometrical Pattern & Design.* Dover Publications, New York, 1973.

[2] Branko Grünbaum and G. C. Shepard. *Tilings and Patterns.* W. H. Freeman, 1987.