Algorithms for Morphing Escher-Like Tessellations

Kevin D. Lee
Math/CSCI Department
Normandale Community College
9700 France Avenue
Bloomington, MN, 55431, USA
E-mail: kevin.lee@normandale.edu

Abstract

Inspired by the way M.C. Escher combined metamorphosis and regular division in his art, I explore linear and non-linear algorithms that automatically morph tiles from the base polygon to a final shape. The morphing can be visualized as an animation or as a parquet deformation.. The final algorithm involves an interactive cubic spline, path-based editor that gives the artist fine control over the intermediate morph frames.

In his print, *Development I*, M. C. Escher shows the visual development of an outer ring of squares morphing through several intermediate rings into a inner ring of four lizards [4, page 30]. Inspired by Escher's print I used *TesselManiac!* [6], a software program I wrote for creating Escher-like tessellations, and a laser cutter to create a wooden chessboard (Figure 1). The chessboard was part of the art exhibit during Bridges 2014. I was not entirely pleased with the result so I have been researching and refining algorithms for morphing the base tile to the final shape in *TesselManiac!* The morphing can be seen on screen as an animation in real time or spatially as the tile evolves along a linear dimension as shown in Figure 7. So far, three different algorithms have been implemented. Much of my work has been based on Doris Schattschneider's book *Visions of Symmetry* [4] and two Bridges papers by Craig Kaplan, see [2,3].

Figure1: *Escher inspired chessboard shown at Bridges 2014*

To understand the morphing algorithms in *Tesselmaniac!* I first need to explain the tile editor. It allows the user to create their own Escher-like tessellations by choosing a tile type from one of 36 isohedral types based on an extended Heesch notation [5]. The user is presented with a fundamental polygon, which they can modify, by adding and dragging control points that reshape the edges. The edges are constrained by the geometric rules of the tile type. There are two types of control points the user can insert: sharp or smooth. Figure 2 shows the initial polygon for a Heesch TTTT tile. In Figure 3 the user

has added sharp points to the top edge, the bottom edge is a translated copy of the top. Figure 4 shows the piecewise cubic spline curves that result from adding two smooth points (and one sharp point) to the left edge. It is worth noting that Figure 4 is really a polygon, the program approximates the curves by adding additional hidden points.

Figure 2: *The initial tile.*

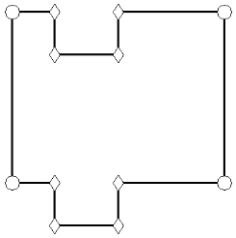

Figure 3: *Sharp points have been added to the top edge and automatically to the bottom edge.*

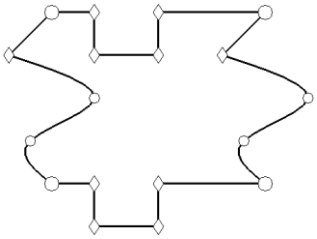

Figure 4: *Smooth points have been added to the left edge and automatically to the right edge.*

I should also note that in the tile data structure, half the control points are master points and the remaining half are transformed copies of the master points. The transformations are based on the tile type. The goal of a morphing algorithm is to generate intermediate polygons that deform smoothly from the initial to final polygon (tile). To illustrate the morping algorithms, I use an elephant tile (Heesch Type TCCTC) based on *Curly Elephants* by Bruce Bilney [1]. The tile is based on an underlying pentagon as shown in Figure 5. *TesselManiac!* includes drawing tools to add interior details to the tile, currently I do not morph these interior details.

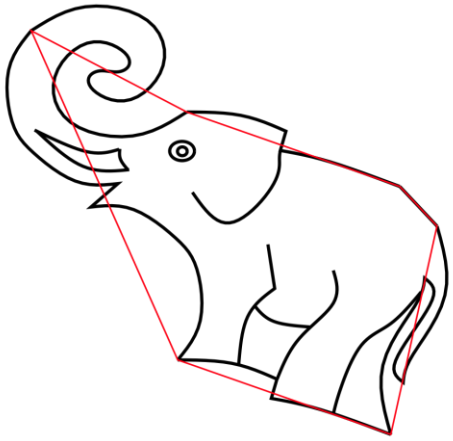

Figure 5: *Bilney's Elephant and Initial Pentagon*

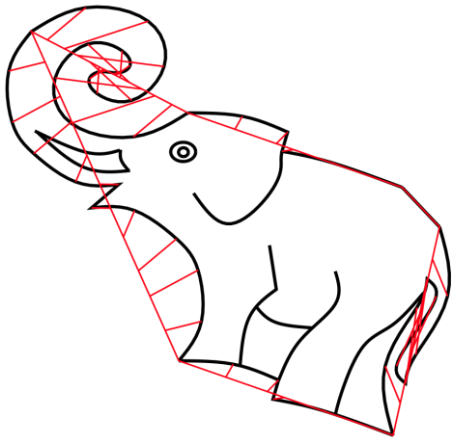

Figure *6*: *Linear Morph Paths*

In all of these algorithms moving the master control points generate the intermediate frames. The master points on final edge need to be connected with points on the initial edge; determining the location of these starting points is a problem in itself. A simple method is to equally space the points along the initial edge. For the algorithms in this paper I use a slightly more complicated method; the starting points are proportionately spaced based on the distance between corresponding points on the final edge.

Figure 6 illustrates the first algorithm, based on linear interpolation. Each master point on the final tile is connected to a point on the corresponding edge of the initial polygon by a line. A time parameter ***t*** is introduced. The equations of the lines are expressed in parametric form with x and y being linear functions of t. At $t = 0.0$ we get the points for the initial polygon, at $t = 1.0$ we get the final polygon. Intermediate frames are generated by values of t between 0.0 and 1.0.

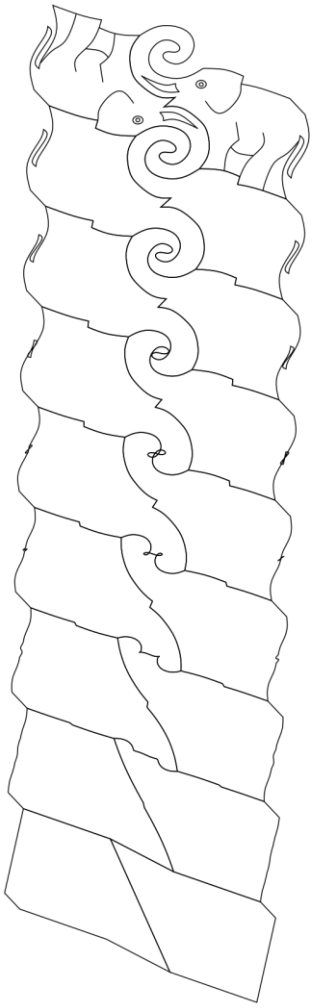

Figure 7: *Linear Morph*

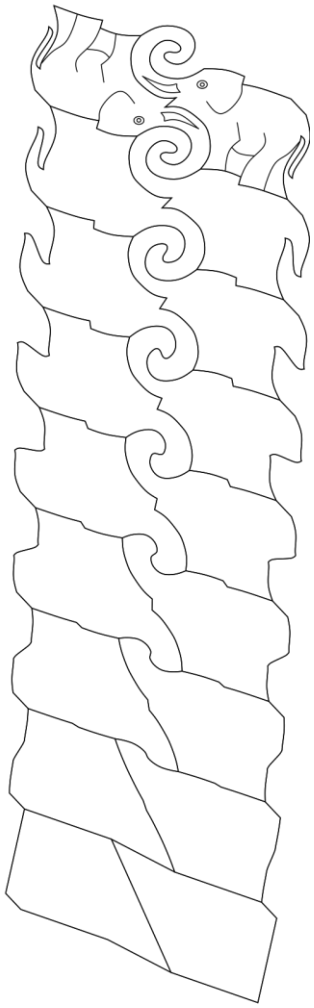

Figure 8: *Sederberg's Morph*

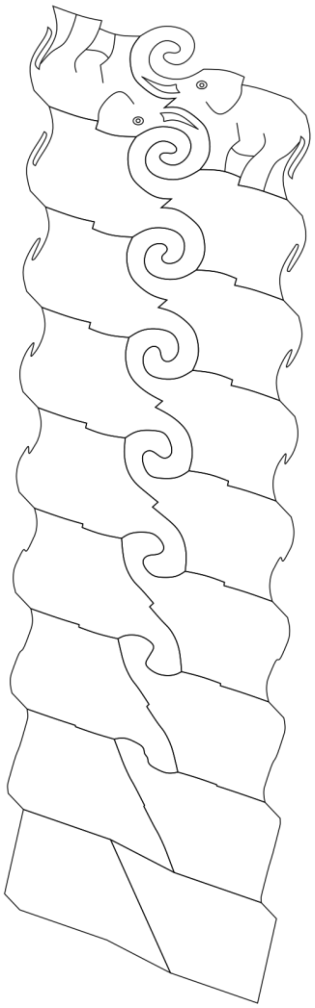

Figure 9: *User-Paths Morph*

The linear algorithm is simple and often gives nice results but the elephant tile illustrates a fatal flaw: intermediate frames often have edges that intersect where the parametric lines cross. See Figure 7 for examples of overlap of the elephant trunks.

For the second method I adapted an algorithm by Sederberg et al [7] for 2D shape blending. Sederberg's algorithm was developed for blending polygons. For morphing tiles the problem simplifies to blending two polylines (tile edges) that share endpoints. Following the spirit of Sedeberg's algorithm I describe a polyline using turtle graphics. From the coordinates of the points of the polyline I calculate an initial direction, distances and turn angles that describe the path a turtle would take to trace the polyline. I interpolate the turtle parameters from start edge to final edge to generate a turtle path for the intermediate polylines. The endpoint of an intermediate turtle path often does not interpolate the final point, to solve this I use an affine transformation to scale and rotate the intermediate polyline to match at the endpoint.

Sederberg's algorithm is illustrated in Figure 10. For most tiles this algorithm solves the intersecting edges problem but often the tiles are not aesthetically pleasing. In this particular case, as you can see from the paths the points trace, several of the tail points swing in wide arcs making the tail bulge in the intermediate frames. Figure 8 illustrates the bulging tail problem.

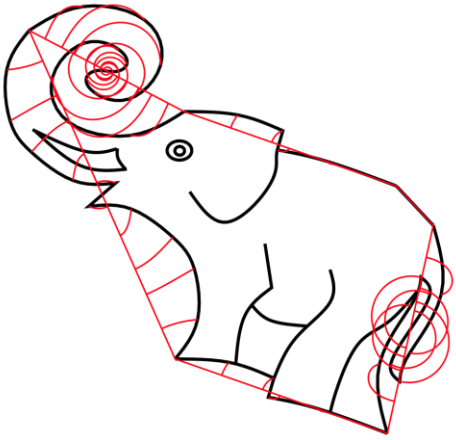

Figure 10: *Paths from Sederberg's Morph*

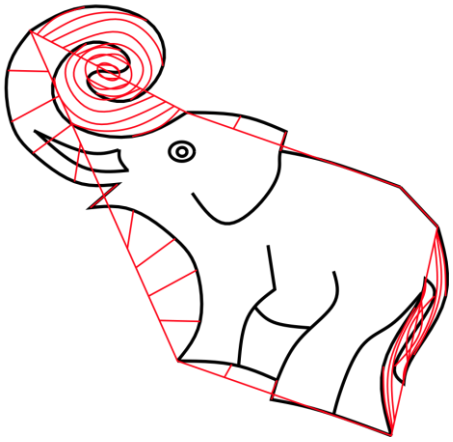

Figure 11: *User Edited Paths*

Figure 11 illustrates my final algorithm, user paths. Initially the paths are taken from the linear algorithm but the artist can click and drag to reshape the paths using spline curves. They can also edit the starting point on the initial edge. The editor includes a slider that runs from 0.0 to 1.0 so they can see intermediate frames as they edit the paths. The edited paths for the tail points in Figure 11 illustrate the ability the artist has to control the way the tail grows. The user can shape the paths to avoid the overlap problem and make the intermediate frames aesthetically pleasing (No unsightly bulges!), see Figure 9. Using spline curves is a common trick in generating paths for morphing graphics, but as far as I know *TesselManiac!* is the first software to apply it to morphing isohedral tiles.

Figures 7, 8, and 9 illustrate the three morphing algorithms and related problems. In the linear morph algorithm you can see several intermediate tiles where the trunks overlap. In Sederberg's morph you can see the bulging tail. The final algorithm gives the user the ability to create an aesthetically pleasing morph with no overlapping edges as shown. The Escher lizard chessboard, Figure 1, was created using Sederberg's algorithm. I intend to create another version using a user-path morph of the lizards, perhaps for Bridges 2016!

References

[1] Bruce Bilney, Curly Elephants Tessellations, http://www.ozzigami.com.au/tessellations.html (as of April 20, 2015)

[2] Craig S. Kaplan. Curve Evolutions Schemes for Parquet Deformations. In *Bridges 2010: Mathematical Connections in Art, Music and Science*, pages 95-102, 2010.

[3] Craig S. Kaplan. Metamorphosis in Escher's art. In *Bridges 2008: Mathematical Connections in Art, Music and Science*, pages 39–46, 2008.

[4] Doris Schattschneider. *M.C. Escher: Visions of Symmetry*. Harry N. Abrams, second edition, 2004.

[5] Kevin D. Lee. Tile Types in TesselManiac, http://www.tesselmaniac.com/tess/Tile_Types.html, (as of April 20, 2015)

[6] Kevin D. Lee *TesselManiac!* 2014 http://www.tesselmaniac.com (as of April 20, 2015)

[7] Thomas W. Sederberg, Peisheng Gao, Guojin Wang, and Hong Mu. 2D shape blending: An intrinsic solution to the vertex path problem. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 15–18, Aug 1993