Constructing Mini-tools for Tessellations

Loe Feijs, Jun Hu, Matthias Funk
Dept. of Industrial Design,
Eindhoven University of Technology,
{l.m.g.feijs,j.hu,m.funk}@tue.nl

Mayra Goevaerts, Teun Keusters,
Caro Van Kessel
{w.f.goevaerts,t.w.keusters,
c.r.v.kessel}@student.tue.nl

Abstract

We present examples of student work from the course Golden Ratio at Eindhoven University of Technology (TU/e) where students learn about tessellations, fractals, splines and Processing. One of the difficulties in designing new tessellations is how to find a basic figure which is attractive or has a meaning and which satisfies the formal requirements of fitting in a complex tiling. In earlier work we combined turtle graphics and splines, which turned out useful for certain projects, yet did not complete solve the problem of how to make the difficult creative step. This year several students created their own interactive mini-tools for designing a type of tessellation of their own choice. The following strategy was found effective: choose a type of tessellation, typically a Heesch-Kienzle type, perhaps with an extra twist, and then build a mini-tool in Processing where the edges of the basic figure can be manipulated interactively using one of the variations of spline theory. We present coding details and examples.

Introduction

Creating tessellations is an art and design challenge which is rewarding for several reasons. First, there is the potential beauty of the results and the fascination for the way the figures interlock. Escher's work was ground-breaking and still inspires many people. The mathematics of tessellations has been described by Heesch and Kienzle [1] as a combination of wallpaper theory and topology. Designing tessellations is an opportunity for design students to refresh their mathematical knowledge and develop computer programming skills: useful later in serious projects of data visualization, digital aesthetics, and personalized product-service systems. This was the idea for the course *Golden Ratio for Industrial Design* (Master) students at TU/e, taught by various configurations of teachers since 2007. The pedagogics are described by Bartneck and Feijs [2]. Nowadays, learning goals include basic geometry, frieze- and wallpaper theory, turtle graphics, splines, tessellations and fractals.

This year we found that several students made their own combination of splines and tessellations. This was enabled by the way we have introduced spline-theory: presenting the polynomials (deriving their properties using calculus) and presenting a demo of a small Processing program where one can drag two anchor points and two control points to see the curve. Students took this demo program, modified it, taking more complex polynomials and embedding it in larger programs with a basic figure composed of several of these splines.

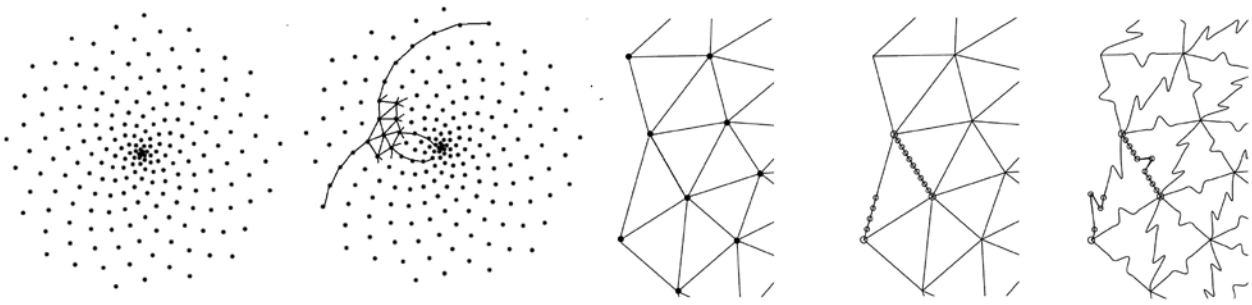

Figure 1: Bumgardner's phyllotaxy spiral, lines, triangles, anchor points, and edited C, G lines.

Implementation

A key tool for putting basic figures together and getting complete tessellations is the collection of examples coded by Jun Hu, available at github.com/iddi/oogway-processing/archive/master.zip, described by Feijs and Hu [3]. It includes descriptions of the Heesch-Kienzle types [1], for example no 28 ($CG_1CG_2G_1G_2$):

> "Glide-reflect the arbitrary line AB to CD (glide-reflection axis H_1I_1). Draw the arbitrary line BC and glide-reflect it in the glide-reflection axis H_2I_2 which is perpendicular to H_1I_1, towards FE (E arbitrary.) Connect D to F by a C-line and connect A to E in the same way."

There are 28 distinct tessellation types. The simplest is called TTTT, referring to two arbitrary so-called T lines, each of which has one Translated (T) copy. The abbreviations code the line types such as T (to be translated), C to be rotated 180 degrees around the center, C_3 (to be rotated 120 degrees), C_4 (to be rotated 90 degrees), C_6 (to be rotated 60 degrees), and G (to be glide-reflected).

The above-mentioned interactive spline demo works with an array of (*x*,*y*) pairs called `myPoints` such that `myPoints`[i][0] contains the *x*-coordinate of the i-th point and `myPoints`[i][1] the *y*-coordinate.

```
if (mousePressed){
    for (int i=0; i<4; i++){
        if (dist(myPoints[i][0],myPoints[i][1],mouseX,mouseY) < 20){
                myPoints[i][0] = mouseX;
                myPoints[i][1] = mouseY;
}   }    }
```

This is the core of a mini line-editor. Drawing splines is easy defining `polynomial`(x_0,x_1,x_2,x_3,t) as $x_0(1-t)^3+ 3x_1(1-t)^2t+3x_2(1-t)t^2+ x_3t^3$. To get a Bézier curve apply the polynomial to both *x* and *y*-coordinates in `myPoints`, looping on *t* from 0 to 1 in small steps. Processing also has *curveVertex* for Catmull Rom splines, but are not as smooth as Bézier. Students made different choices: Keusters took $CG_1CG_2G_1G_2$, Goevaerts CGG, turned into spirals and based on code inspired by Jim Bumgardner's basic phyllotaxy spiral [1]. Van Kessel made line art (no Heesch theory, Fig. 3). Keusters used Bézier curves with 3d order polynomial, Goevaerts used Catmull Rom splines with 10 control points for the G line and 4 for the C line. Van Kessel used Bézier curves with a 9th order polynomial (a nice exercise deploying Pascal's triangle). Van Kessel's editor loads a background image. The mini-tools help making the non-trivial creative step. As Teun Keusters puts it: "Now the editor was finished, it was time to start playing around with it, trying to get some recognizable shape out of it. Although the editor made sure that a $CG_1CG_2G_1G_2$ shape always remained, it was hard to make a something recognizable. Every line affects another which makes it hard to exactly get the shape I wanted" (see Figure 2).

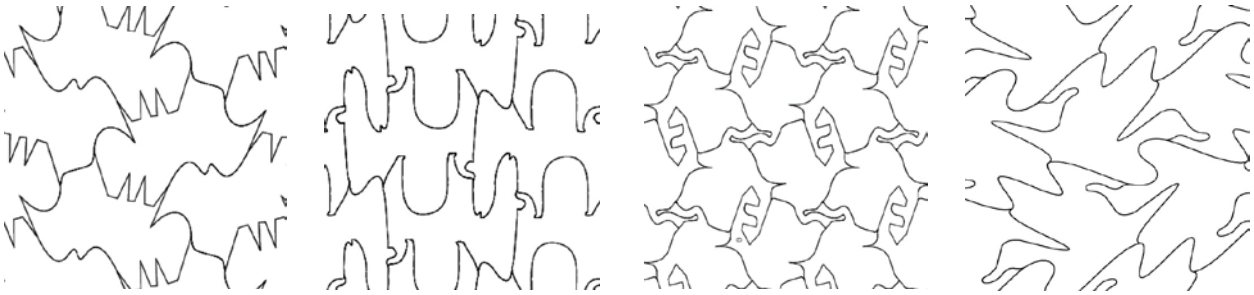

Figure 2: Rhinos, dromedaries, ostriches and geese.

Student Projects

The materials used for the final realization were different. As explained by Bartneck and Feijs [2], the physical realization adds significantly to the motivation and enthusiasm of the (industrial design) students. Teun Keusters had the goose-shaped tiles laser-cut in two kinds of plywood: ash and birch wood. He painted the backside of the ash wood black, but the backside of the birch looked nice after laser cutting, so this was kept in its original color. Then, the pieces were glued together in a hexagonal shape on a plate of 2mm MDF (Figure 3). This makes it possible to hang it on a wall, and to transport it easily.

Figure 3: *Geese tessellation of* $CG_1CG_2G_1G_2$ *type.*

Caro Van Kessel converted the generated pdf file into PES embroidery format and had it embroidered using a Brother Entrepreneur® PR655 embroidery machine. Using a cotton fabric, the work became a fashionable bag (Figure 4). Mayra Goevaerts used two materials: 3 mm thick acrylic glass and 3 mm MDF (Figure 5).

Figure 4: *Design and embroidered realized bag of line art.*

Conclusions and Outlook

The formula of dedicated mini-tools seemed to work well and the students improved their programming skills in Processing (Java). Other students (not reported in this article) took different approaches, such as making the creative step by traditional sketching, or turning to other mathematical ideas, still within the scope of the course. The code of Mayra is on https://git.io/vS7ly, Teun's code at git.io/vS7lX and Caro's code at git.io/vSAp8. We thank Chet Bangaru and Jasper Sterk for their expertise and support.

Figure 5: *Phyllotaxis-inspired swans tessellation of CGG type.*

References

[1] Heesch, H. and Kienzle, O. (1963). *Flachenschluss; System der Formen lückenlos aneinanderschliessender Flachteile*. Berlin: Springer.

[2] L.M.G. Feijs and C. Bartneck. Teaching geometrical principles to design students, Digital Culture & Education, Vol. 1(2009), No. 2, p. 104-115.

[3] Loe Feijs, Jun Hu. Turtles for tessellations. In: G. Hart and R. Sarhangi (Eds.), *Proceedings of Bridges 2013*, pp. 241-248. Online: archive.bridgesmathart.org/2013/bridges2013-241.pdf

[4] Jim Bumgardner. *Circles, Spirals and Sunflowers: A Processing.js tutorial.* krazydad.com/tutorials/circles_js/.