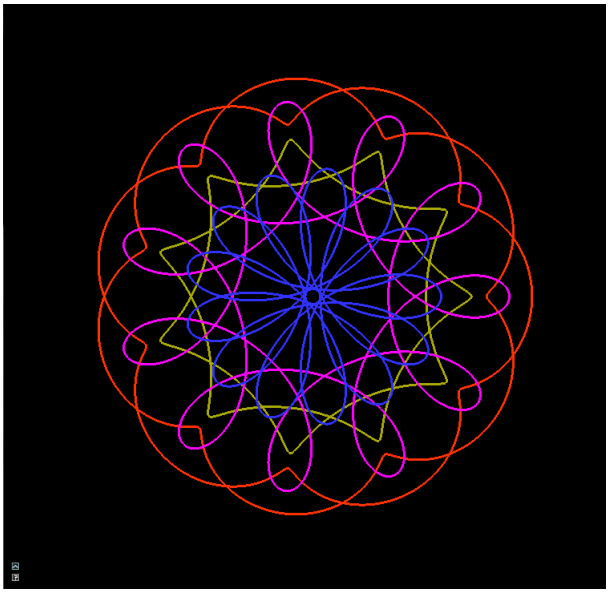
Goldsmiths University of London

Spirograph

Coursework 2

MSc in Computer Games and Entertainment
Mathematics and Graphics for Computer Games 1



Andrea Castegnaro, Aldo Curtis

26 January 2014

Abstract

This document present the result for the second assignment of the Maths and Graphics course.
Spirograph are geometric curves based on recursive functions and a set of parameters that control the output shape. The find very interesting application in art and design. Our proposal has been creating a fast tool to create shapes providing the user the possibility to have total control on the creation by changing function type, line drawing parameters and colours. Our best effort has been thinking on using math techniques to have a smooth and fast drawing.

Contents

List of Figures

Chapter 1

Spirograph

1.1 Introduction

Spirograph[1] was originally a geometric drawing toy that produces mathematical roulette curves of the variety technically known as hypotrochoids and epitrochoids. It is possible also to has custom function for this purpose as long as they agree with the current rules for a spirograph. In this chapter we will discuss the general math theory behind spirograph and we will describe also how to customize the function. The application has been developed using Andy Thomason's Octet Framework.

1.2 Mathematical Basis

Parametric curves[2] have a verity of equations that can represent them, and these equations can differ in both the values they require for input and how many values they output, for example in the generation of a 3D curve it will need return 3 values; for 2D 2 values are needed.

A Spirograph is formed by rolling a circle inside or outside of another circle[3]. The inner circle has control points that describes the curve. If the radius of fixed circle is R, the radius of moving circle is r, and the offset of the inner point in the moving circle is 0. The equation for the having a inner circle or the so called Hypotrochoid is the following:

$$x(\theta) = (R - r)\cos(\theta) + d\cos(\frac{R - r}{r}\theta) \tag{1.1}$$

$$x(\theta) = (R - r)\sin(\theta) - d\sin(\frac{R - r}{r}\theta) \tag{1.2}$$

$$\tag{1.3}$$

And in the same way we have the equation for the Epitrochoid or the outer circle such as:

$$x(\theta) = (R + r)\cos(\theta) - d\cos(\frac{R + r}{r}\theta) \tag{1.4}$$

$$x(\theta) = (R + r)\sin(\theta) - d\sin(\frac{R + r}{r}\theta) \tag{1.5}$$

And by combining the two of them we cna have nice configuration such as:

$$x(\theta) = (R - r)\cos(\theta) + r\cos(\frac{R - r}{r}\theta) \tag{1.6}$$

$$x(\theta) = (R - r)\sin(\theta) - r\sin(\frac{R - r}{r}\theta) \tag{1.7}$$

We added also a custom function with 5 parameters which gave us nice results. The equation is the following:

$$x(\theta) = \cos(a\theta) - (\cos^c(b\theta)) \tag{1.8}$$

$$x(\theta) = \sin(d\theta) - (\sin^f(e\theta)) \tag{1.9}$$

1.3 Implementation Idea

Our purpose has been creating a fast general tool [4] for creating more than one graph in real time and to use this we focused on optimizing three aspect:

- computation time for calculating the points of the curve
- optimization of resolution step using de Casteljau subdivision [5]
- fast GPU rendering using Bezier interpolation[6]

So in order to create a fast drawing tool we had to achieve a fast computational time for the points on the curve which will be drawn in the render step (using a custom shader) using an interpolation method.

1.3.1 Fast subdivision

To achieve the first two points we decide to use a big initial resolution for calculating the points given by the function and we increment the step size only when needed. In order to do we took inspiration from the Casteljau subdivision[7].

Given a big initial resolution we calculate our point along the curve and using fast algorithm we evaluate the curvature in order to see if we need more precision to describe it. The curvature of an area is calculated using three points along the line to construct a triangle and calculate its area Fig. 1.1, the points are sampled at regular intervals and the line is considered curved if the area is above a certain threshold [8]. The points are

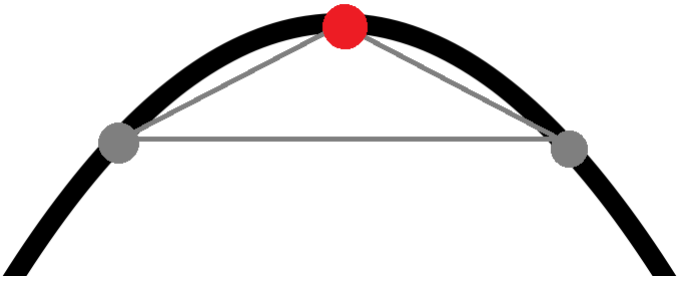


FIGURE 1.1: Schematic of the algorithm use for curve subdivision.

generated from the $t, t + dand((t + d) - t)/2$ with the third point being halfway between the other two. This is implemented recursively, upon finding a curved section beyond tolerance the algorithm is called using t and the half point, and then on the half point to $t + d$, successively dividing the curve. The result of this method can be found in Fig 1.2

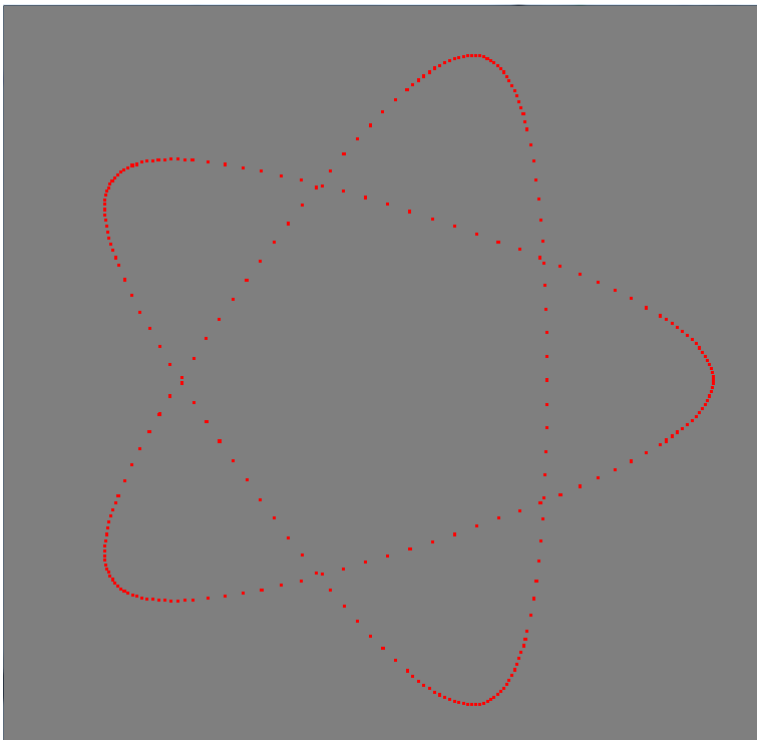


FIGURE 1.2: Effect of the subdivision algorithm implemented

1.3.2 Bezier interpolation

A Bezier curve is a parametric curve frequently used in computer graphics and related fields. It is used for smooth path calculation and it assure at least a C^0 continuity.

We used this method because we wanted to achieve very good looking result and we cannot use LERP method since we are not ending with simple linear motions[9].

The general equation for the Bezier curve is the following

$$B^n(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-1} s^i p_i \tag{1.10}$$

where $0 <= t <= 1$ is a parameter.

We used a third order Bezier curve to interpolate the points since it is not recommended to use a greater one.

We implement a custom feature for testing our Bezier result which consist of having a real time Bezier drawing system. A screenshot of the tool can be found in Fig. 1.3

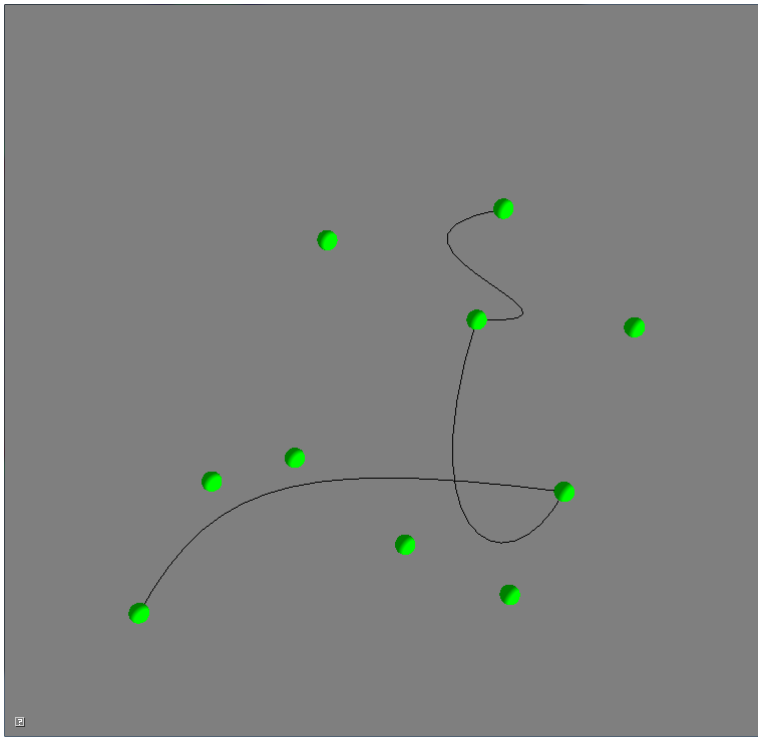


FIGURE 1.3: Effect of the subdivision algorithm implemented

1.3.3 UI interface

In order to customize the drawing real time we added a user interface to our project that had the following characteristics:

- Changing function equation
- T value parameter for the drawing
- Line thickness
- Line Color
- Add new spirograph
- Clear last spirograph

The user interface can be seen in figure 1.4

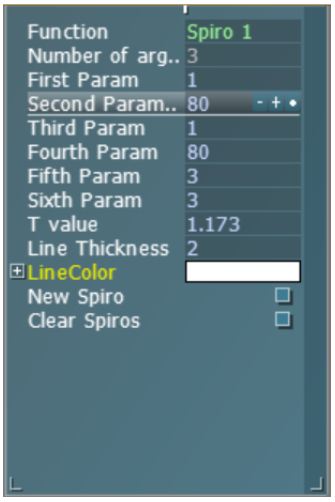


FIGURE 1.4: User Interface screenshot

We slightly go off from our initial idea since the target for this project was understanding the math and graphics technique to draw the lines, so we skipped the shader interpolation but we still put all the feature we want for the algorithm meant to be implemented.

1.4 Implementation

The actual implementation is schematized in figure 1.5

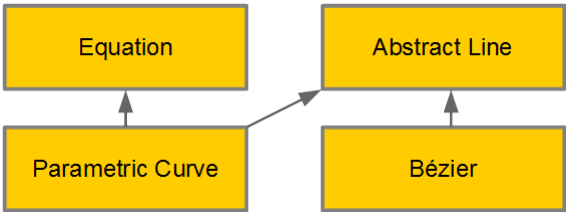


FIGURE 1.5: Schematic representation of the classes used for the implementation

The Equation class implements support for abstract functions, providing a vector of arguments and return values to suit any number of implementations. By storing a pointer to the function representing the equation it is easy to change between equations on the fly, as seen in the code where a vector of equation function pointers are stored and are swapped in and out at will.

The parametric converts the curve defined in equation form into geometry Line segments are used to represent the curve, performing dynamic sampling to accurately and efficiently represent the curve with the fewest lines possible. Areas with a low curvature or straight lines use less lines to represent them as they can be approximated by a single line segment where areas with high curvature are given higher resolution.

The Bezier line is implemented using 4 control points in 3D space. It uses a pool of control points which makes the insertion of new control points efficient and easy. The line is generated using the quadratic Bezier equation, and evaluating t from 0 to 1 to form the line from the start to the end.

AntTweakBar is free to use UI that can be used to tweak internal variables. The curve is regenerated every time a change is detected in the variables, and will update live with changes of the variables on the UI, this is simply done by saving the old state variables and comparing them to the current ones.

1.5 Drawing result

In this section we present some shape we manage to create with our system. We used a paper for interior design to produce many of this shape [10]

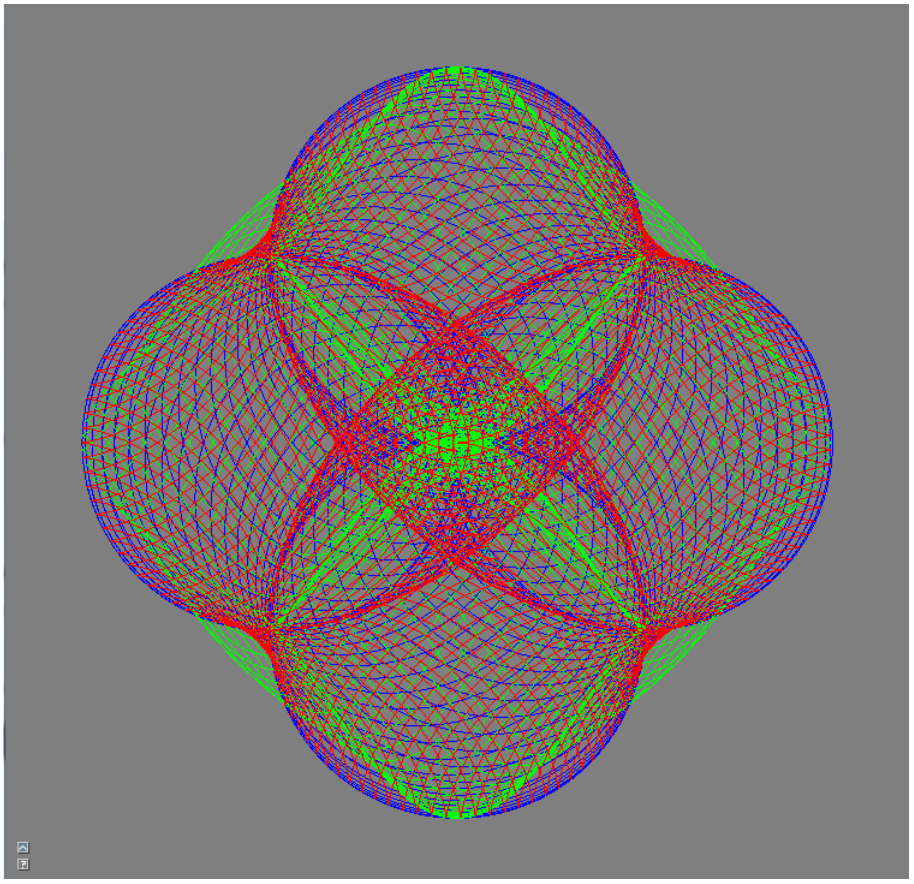


FIGURE 1.6: Example Spirograph.

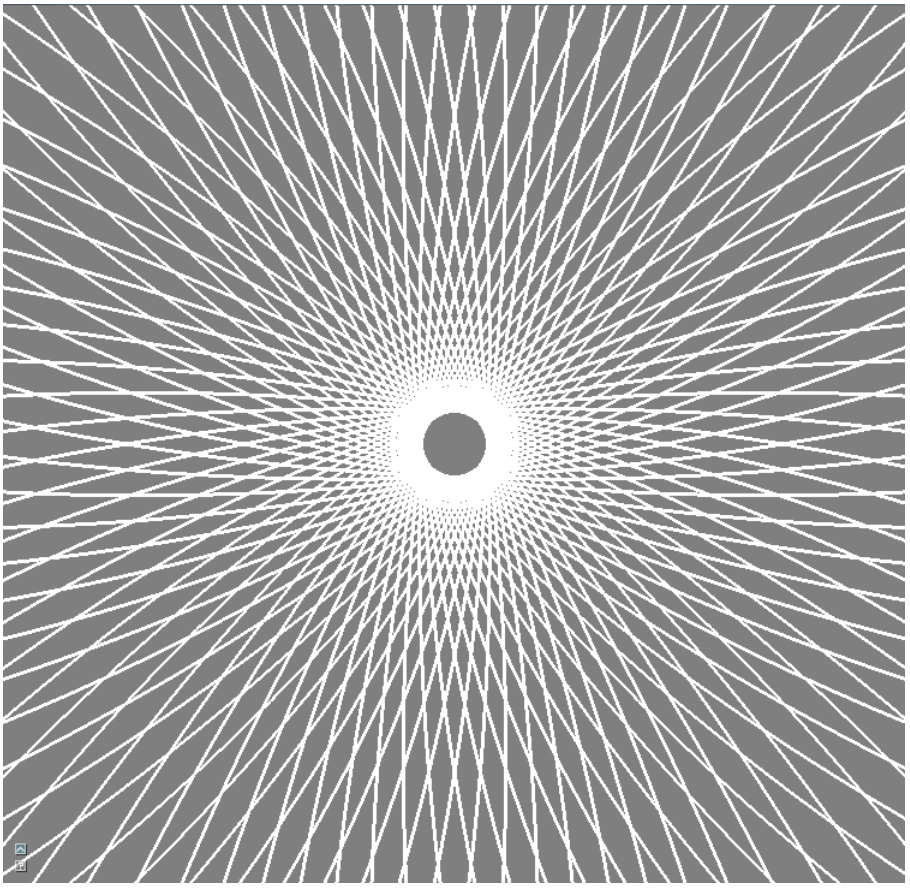


FIGURE 1.7: Example Spirograph.

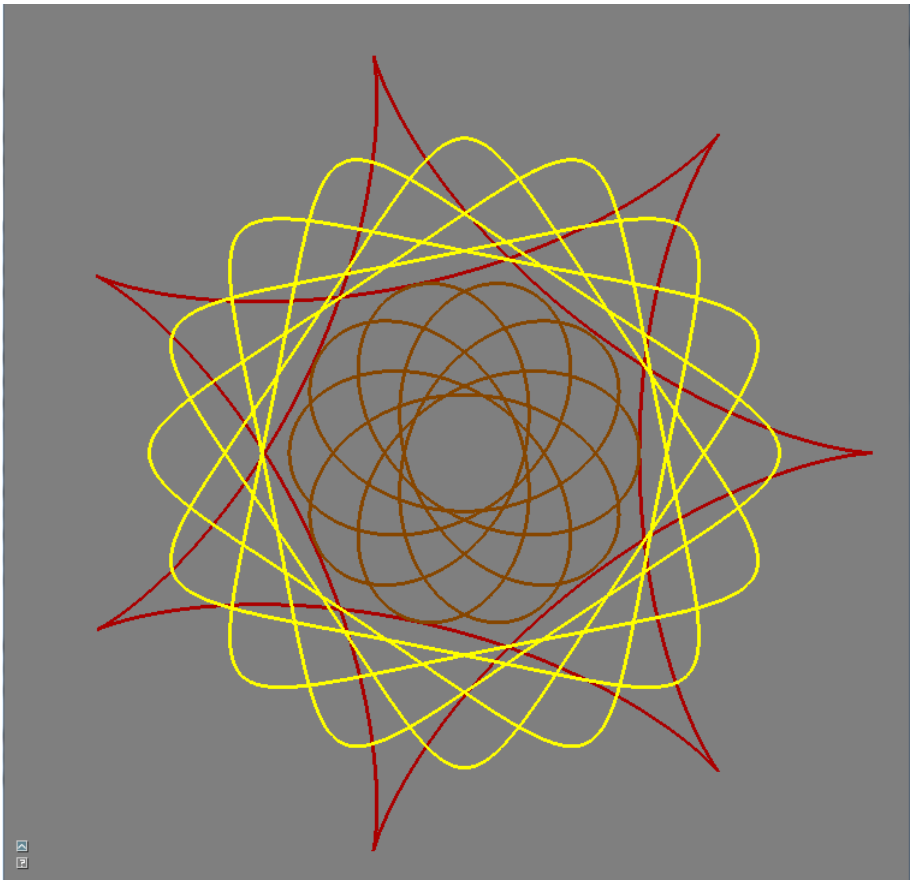


FIGURE 1.8: Example Spirograph.

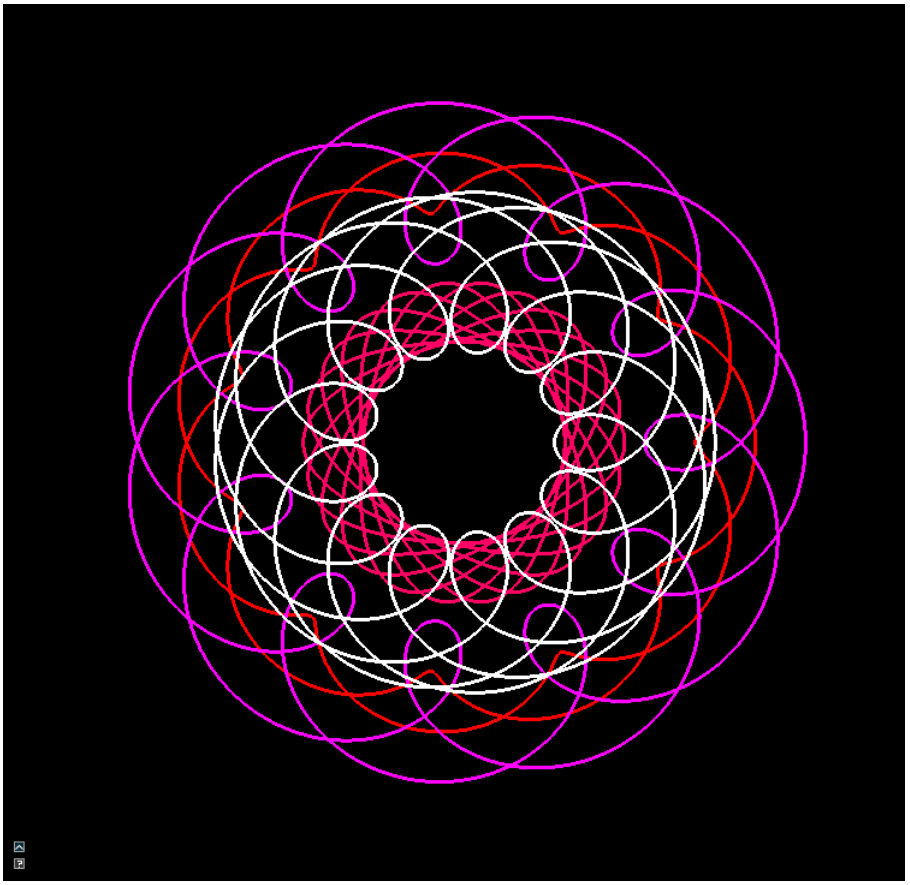


FIGURE 1.9: Example Spirograph.

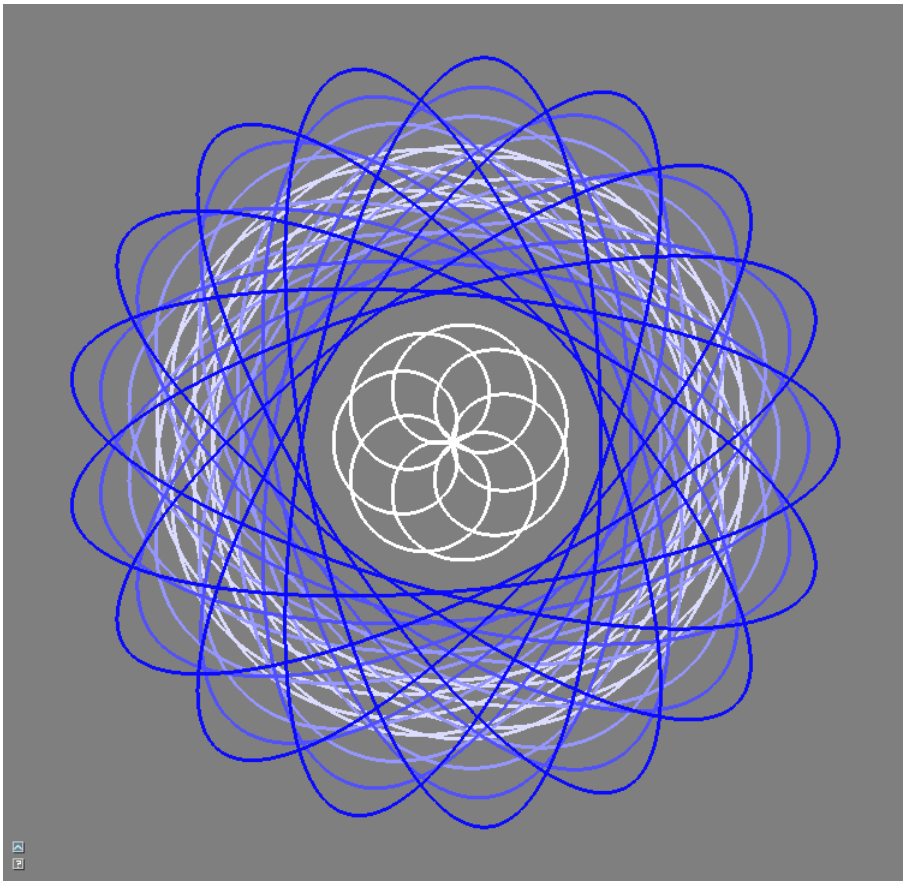


FIGURE 1.10: Example Spirograph.

Bibliography

[1] Wikipedia spirograph. . URL http://en.wikipedia.org/wiki/Spirograph.

[2] Dunn and Parberry. 3d math primer for graphics and game development.

[3] Ken Johnson. Spirograph shapes: Wpf bezier shapes from math formulae. April 2010. URL http://www.codeproject.com/Articles/76878/Spirograph-Shapes-WPF-Bezier-Shapes-from-Math-Form.

[4] Nathan Friend. Inspirograph. URL http://nathanfriend.io/inspirograph/.

[5] De casteljau's algorithm. . URL http://en.wikipedia.org/wiki/De_Casteljau%27s_algorithm.

[6] Bezier curves. . URL http://en.wikipedia.org/wiki/B%C3%A9zier_curve.

[7] Maxim Shemanarev. Adaptive subdivision of bezier curves. July 2005. URL http://antigrain.com/research/adaptive_bezier/.

[8] Sara Su. Computer graphics, lecture 18 parametric curves/ subdivision surfaces. URL http://www.cs.tufts.edu/~sarasu/courses/comp175-2009fa/pdf/comp175-18-surfaces.pdf.

[9] Juraj Onderik. Basic methods in computer animation. URL http://www.sccg.sk/~durikovic/classes/CGAnim/ca10_lesson02.pdf.

[10] Romain Vuillemot Ye Lin. Spirograph designs for ambient display of tweets. October 2013. URL https://hal.inria.fr/hal-00856824/document.