MATHEMATICAL MODELING OF A SNOWSTORM AND A GALAXY USING JAVASCRIPT AND FLASH

Paul R. Bouthellier
Department of Mathematics and Computer Science
University of Pittsburgh-Titusville
504 East Main Street
Titusville, PA 16354
pbouthe@pitt.edu

Introduction

Having taught both mathematics and web design, a logical intersection of these two areas is that of computer programming and mathematical modeling. Constructing mathematical models for 2- and 3-dimensional objects and showing how they are projected onto a computer screen shows computer science students another use of mathematics, and gives math students an additional application of their studies.

The study of computer graphics covers a wide range of mathematical topics: Fourier series, Bezier curves, B-splines, graph theory, cross products, quaternions, rotation matrices, and many more. Entire courses are devoted to the mathematical modeling of real-world objects for computer graphics.

In this paper mathematical modeling will be used to create a snowstorm and a galaxy in 3-dimensional space. Both require: rotation matrices, projections, geometry, trigonometry, and random number generators. The galaxy also used a variation of the Fibonacci series. The snowfall program can be used to illustrate the Monte Carlo method to approximate areas as well. Mathematical modeling for computer applications is so diverse that its study serves as an excellent source of practical examples and projects in many undergraduate (and graduate) courses.

The programs in this presentation were written in the ActionScript language of *Flash*. (Simpler 2-dimensional versions can be created in JavaScript.) The programs were saved in the .swf format and embedded in web pages. The graphics were created in the vector-based art packages *Swift3D* and *SwishMax*. Students were not required to learn how to program in ActionScript-just to be able to alter the code and re-run the program. This worked quite well after walking through a few examples.

A Three-Dimensional Snowfall

Wanting to illustrate vector fields and rotation matrices in multivariate calculus, an example which allowed both to be illustrated and easily altered was a snowfall in R^3. Most snowfall programs are basically 2-D, so this project allows students to see how to generalize these concepts into R^3 and to study mathematical modeling.

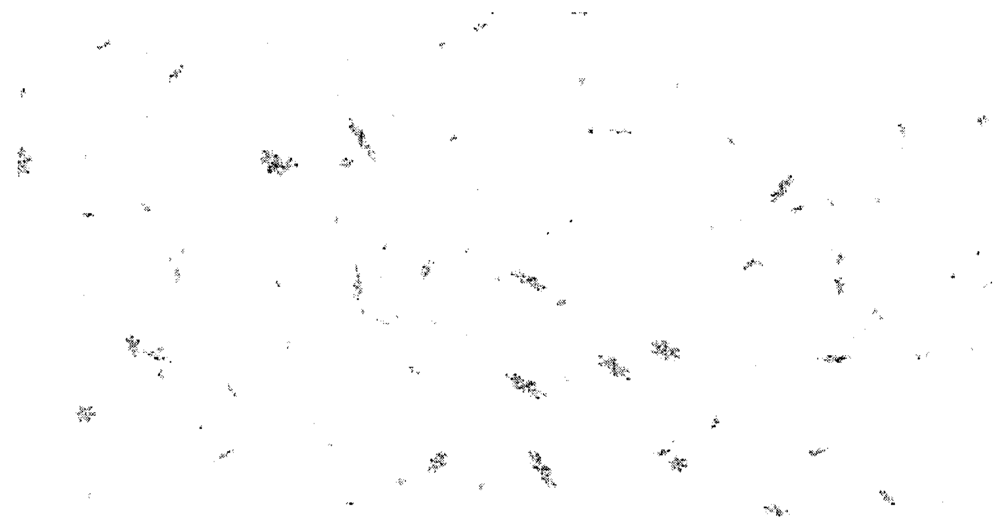


Figure 1. A 3-Dimensional Snowfall

To create the snowfall, snowflakes must first be created. The snowflake in Figure 2a below was created in *Swift3D* using symmetry and rotations. Creating such basic objects is a good first example of how to use mathematics to model real-world objects. *Swift3D* can also be used to create 3-dimensional snowflakes. This helps students visualize 3-dimensional space.

The Koch snowflake in Figure 2b was created using recursions with the package *The Geometer's Sketchpad.*

To create (the illusion of) 3-dimensional rotations of these flakes, the package *SwishMax* was used to rotate the flakes about various axes in R^3. This is an excellent place to discuss rotation matrices in classes such as multivariable calculus.

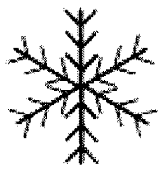


Figure 2a-Snowflake via Symmetry

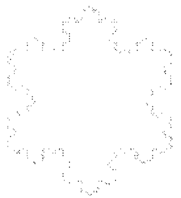


Figure 2b-Koch Snowflake-GSP

The next step is to translate these snowflakes into .swft files so that they can be imported into *Flash.* Finally, the snowflakes need to be placed randomly in a viewing frustum and then projected onto the viewer's computer screen.

The viewing frustum is created by placing an imaginary eye (point) in front of a rectangle that serves as a computer screen.

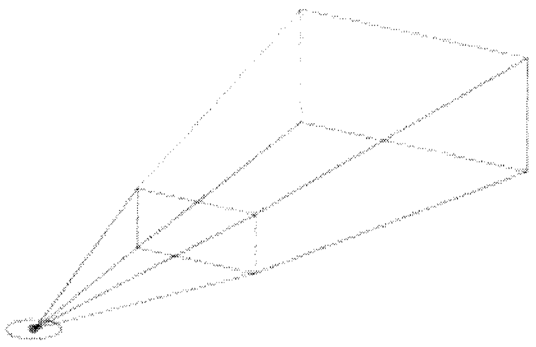


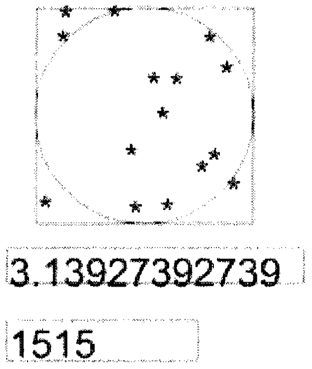
Figure 3-The Viewing Frustum

Four lines in 3-space are then created between the eye and the corners of the rectangle. These lines are used to create inequalities along the x- (horizontal), y- (vertical), and z- (depth) axes.

Depending on computer speed and memory, between 50 and 300 snowflakes are randomly placed in the frustum using uniform distributions and the inequalities described above. Using geometry, the size of the flakes are scaled depending on the distance from the viewer. The location of each flake on the computer screen is determined by the intersection of the line between the flake and the viewer's eye with the rectangle representing the computer screen.

Vector fields in R^3 are used to create motion for the flakes (to simulate wind) in the frustum and hence on the screen. When a flake moves out of the viewing frustum it is removed from the program and a new flake is placed in the frustum. *Flash's* rotate command can also be used to impart additional rotation (about the z-axis) if desired.

With simple alterations, this program can be used in calculus and numerical analysis classes to illustrate the Monte Carlo method. In Figure 4 the area of a circle is approximated.


```
3.13927392739

1515
```
Figure 4-Monte Carlo Method

Another variation of this program is to create the effect of falling leaves (Figure 5). Using 2nd order Bezier curves in the package *Swift3D,* the leaves were created and then extruded along their normals to create depth. The creation of objects such as leaves is a good beginning project in any class which uses polynomials as approximating functions.

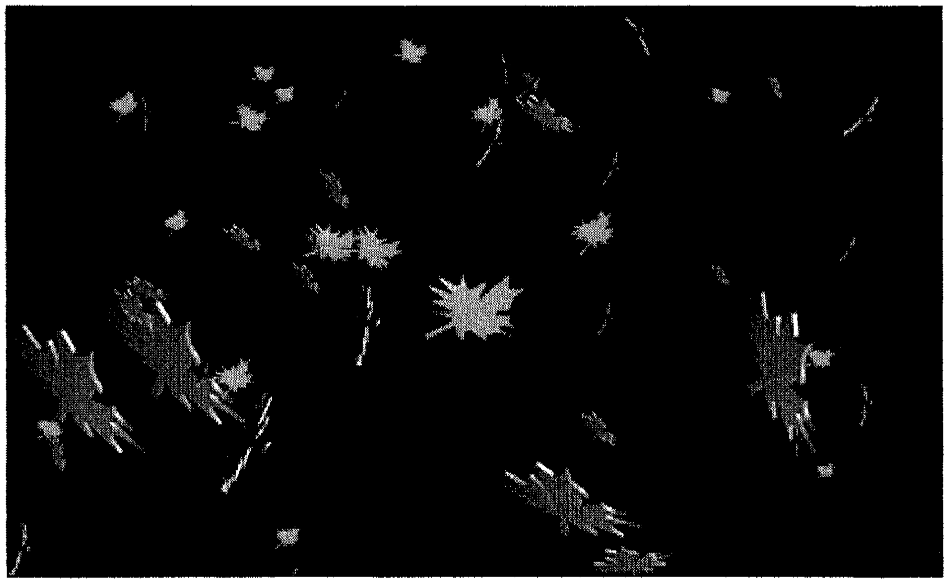


Figure 5-Bezier leafs

A Three-Dimensional Galaxy

To create a 3-dimensional galaxy, a 2-dimesional galaxy is first created. Using the parametric equations x(t)=$\sqrt{n}\cos(\theta n)$ and y(t)= $\sqrt{n}\sin(\theta n)$ where $\theta = 137.5^o$ for the nth star, the "galaxy" in Figure 6a is created. Figure 6a is also the classic sunflower pattern. The angle $\theta = 137.5^o$ is determined by using the limiting ratio of the elements of the Fibonacci sequence applied to a circle: $137.5 = (\frac{\sqrt{5}-1}{2}) * 360^{O}$.

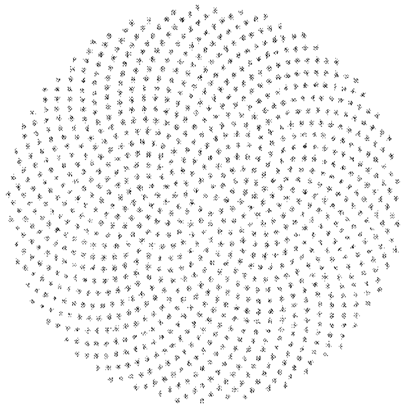


Figure 6a-The Fibonacci Galaxy $\theta = 137.5^{O}$

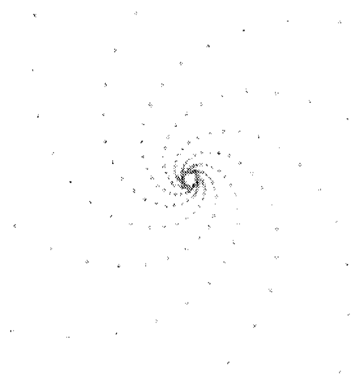


Figure 6b-
x(t)=$(1.02)^n\cos(\theta n)$ and y(t)=
$(1.02)^n\sin(\theta n)$ where $\theta = 136.5^{O}$

By altering θ and the radius we can get more realistic galaxies, as shown in Figure 6b. A good problem in classes, such as trigonometry, is to have students change θ and to explain mathematically what they see.

By replacing the stars with spheres and rotating the 2-dimensional galaxy about the vertical axes we can create the 3-dimensional galaxy shown in Figure 7. As in the snowfall program, it is necessary to take into account the depth of each star, making the stars which are further from the viewer's eye appear to be smaller. This can be used as an example in geometry to illustrate similar triangles or in trigonometry as an application of angles.

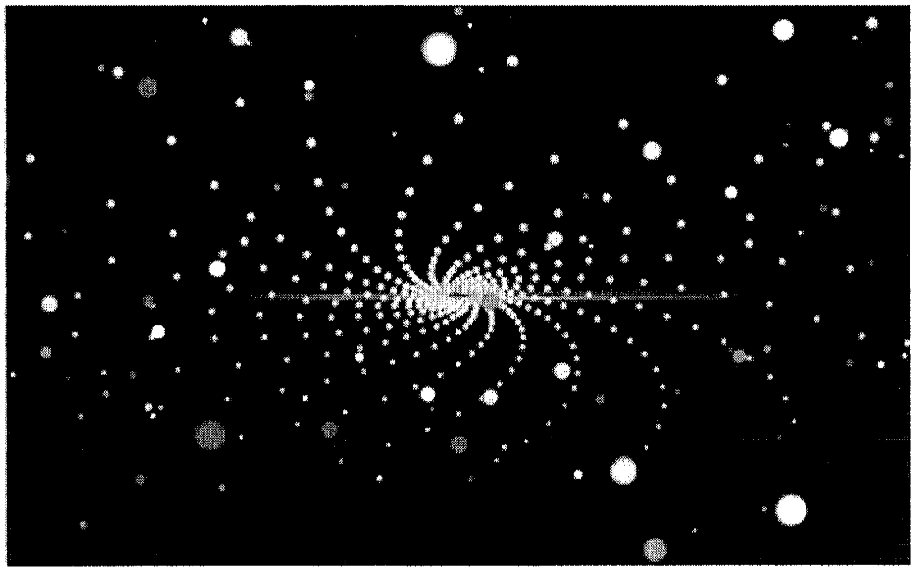


Figure 7-A 3-Dimensional Galaxy

By placing additional stars randomly in front of the galaxy we can create a more realistic scene. By using 2-D rotation matrices the program, via the left and right arrow keys, the users can rotate the galaxy clockwise and counterclockwise. The up arrow key allows the user to move forward through the galaxy. These applications have proven to be good examples in classes such as trigonometry and multivariable calculus.

Conclusion

Using computer programming and art packages, many examples and projects in mathematical modeling can be created in undergraduate and graduate courses. The only limitations are the imagination, the availability of the programs and computers, and time.