Mathematical Modeling for Game Design

Paul Bouthellier and Saeed Dubas
Department of Mathematics and Computer Science
University of Pittsburgh-Titusville
Titusville, PA 16354
FAX: 814-827-5574
dubasis@pitt.edu (Ph: 814-827-5672)
pbouthe@pitt.edu (Ph: 814-827-4432)

In this paper we shall use the package *Flash* to study the mathematics required to create a simple shooting game. In each level of the game moving targets are created and then "hit" by the mouse-which is represented by a crosshair. After all the targets are hit the next level is loaded in which the targets move faster. The player is given 20 shots in the first round and 20 additional shots in each additional round. The game is concluded when the player runs out of shots.

Targets may be created using recursive methods (such as the Koch snowflake in *Geometer's Sketchpad*), "circles" using Bezier curves, or 3-dimensional objects created by rotating a curve about an axis.

The paths of the objects may be created by using parametric equations, Bezier curves, interpolation polynomials, vector fields, random variables, and trigonometric functions, amongst others. The paths may be created in R^2 or R^3. If creating the paths in 3-dimensional space one can also discuss the geometry necessary to scale and alter the apparent speed of the objects.

Students were not required to learn ActionScript, the programming language in *Flash*-just to be able to alter the code and re-run the program. After studying a few examples this proved to be quite easy.

The mathematics used in game design is diverse and extensive. Hence we only can begin to touch on the subject in this paper. However, the examples given here have proven useful in classes to illustrate an additional application of mathematics and statistics.

A Simple Game:

Here the objective was to show an application of parametric equations. The parametric equations are used to move simple circles across the screen. The students can alter the equations to create different paths. In class, various standard functions y=f(x) can be translated into parametric equations. A useful exercise is to reverse the process and study how parametric equations can be written as a standard y=f(x) function.

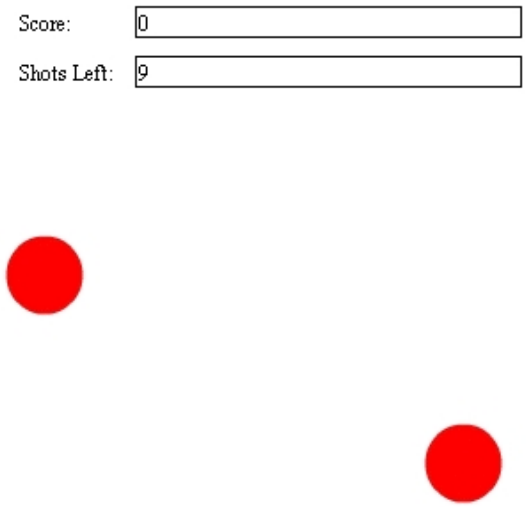

Figure 1-An Application of Parametric Equations-Moving Objects

In *Flash* the "circles" are actually approximated by 2nd order Bezier curves. To illustrate this, the following program, using 8 2nd Bezier curves, was written in ActionScript to approximate a circle.

```
onLoad () {

_root.createEmptyMovieClip("tri",1);
var xs, xe, ys, ye, xi, yi, ms, me;
var r=200;
var ix=300;
var iy=300;
```

```
var num=8;//If odd can get infinite slopes which can causes problems

_root.tri.duplicateMovieClip("test",2);

with(_root.test) {
lineStyle(1,0xff0000,100);
moveTo(_root.ix+_root.r*Math.cos(Math.PI/(_root.num)),_root.iy+_root.r*Math.sin(Mat
h.PI/(_root.num)));
for (var k=1;k<=_root.num;k++) {
var angkm1=Math.PI/(_root.num)+(k-1)*2*Math.PI/(_root.num);
var angk=Math.PI/(_root.num)+k*2*Math.PI/(_root.num);
_root.ms=-1/Math.tan(angkm1);
_root.me=-1/Math.tan(angk);
_root.xs=_root.ix+_root.r*Math.cos(angkm1);
_root.xe=_root.ix+_root.r*Math.cos(angk);
_root.ys=_root.iy+_root.r*Math.sin(angkm1);
_root.ye=_root.iy+_root.r*Math.sin(angk);
_root.xi=((_root.ye-_root.ys)+(_root.ms*_root.xs-_root.me*_root.xe))/(_root.ms-
_root.me);
_root.yi=_root.ys+_root.ms*(_root.xi-_root.xs);
curveTo(_root.xi,_root.yi,_root.xe,_root.ye);}}}
```

The resulting “circle” is given below in Figure 2.

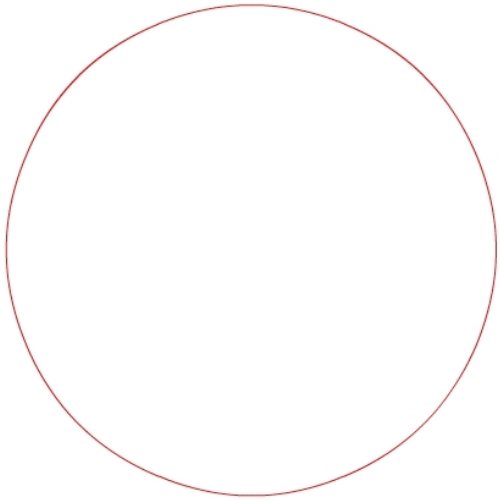

Figure 2-A Bezier Approximation of a Circle

One can then discuss the maximum error of approximation of using Bezier curves to approximate a given circle.

The advantage of modeling objects mathematically (using vector graphics) is illustrated by the fact that they can be scaled with little loss of accuracy. In Figure 3, the circle on the left-stored as a bitmap-was scaled-up with a loss of accuracy. The circle on the right was stored as a vector graphic-Bezier curves-and scaled-up. As the mathematical equations are easy to scale the result is more accurate.

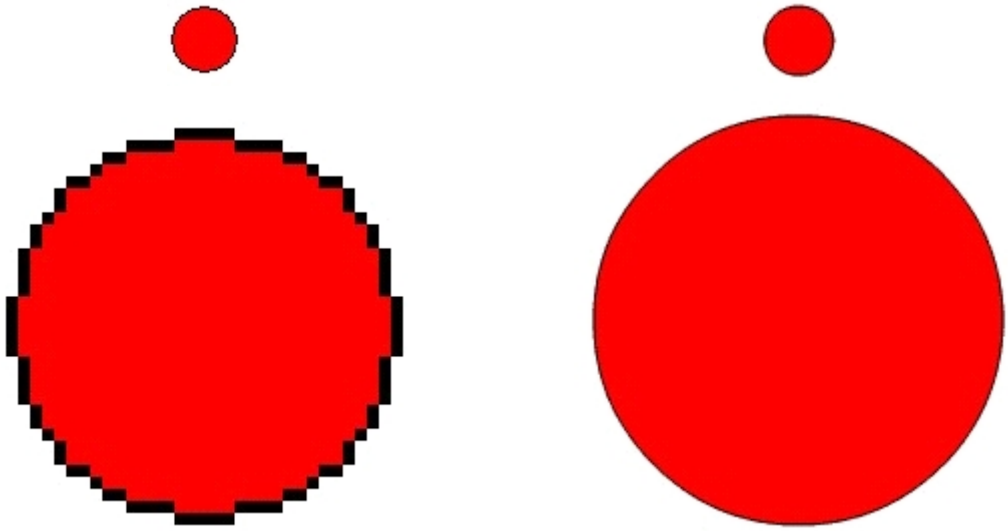

Figure 3- Raster verses Vector Representation of a Circle

3-dimensional targets for the game-as shown in Figure 4-were then created.

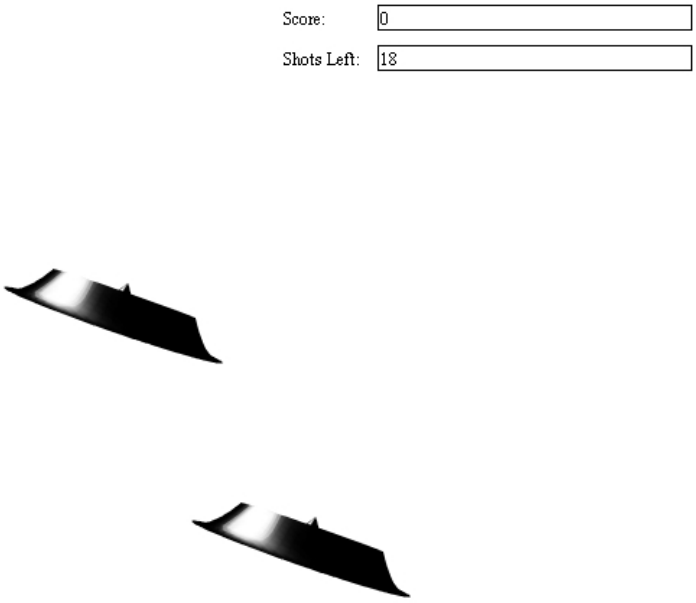

Figure 4- 3-Dimensional Objects Created by Rotating Curves about an Axis

A good start for students in creating mathematical models of objects is to begin with something that can be created by rotating a curve about an axis. The flying saucers illustrated in Figure 4 were created by rotating straight lines and a 2nd order Bezier curve about the y-axis in the art package *SwishMax*. This is shown in Figure 5.

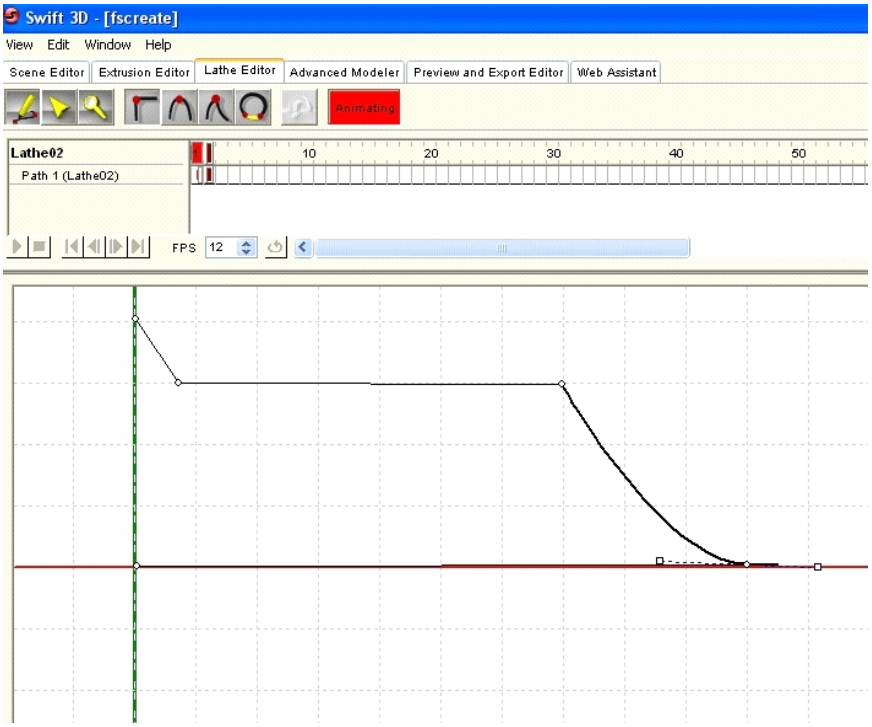

Figure 5-Rotating Curves about the y-Axis to Create a Flying Saucer

A more elaborate example of mathematical modeling is to use Bezier curves to mathematically model leaves, as shown in Figure 6. A related example would be to create a snowfall in 3-space.

Figure 6- Modeling Leaves using Bezier Curves

Another example of modeling a well-known object is shown in Figures 7a-b. The chess piece was created in *SwishMax*.

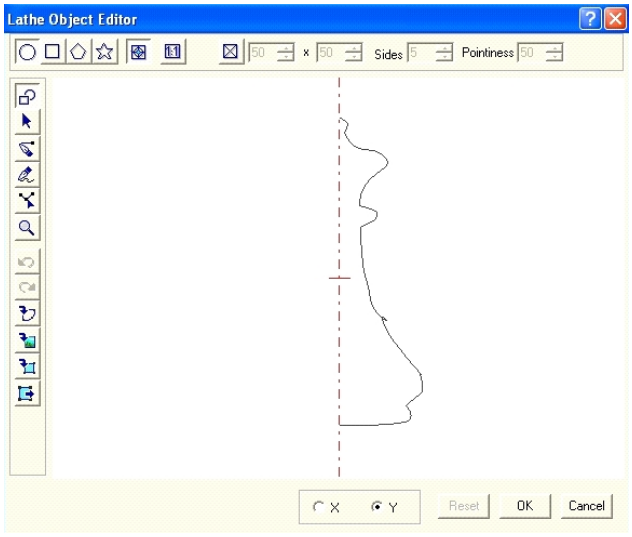

Figure 7a-Bezier Outline of a Chess Piece

Figure 7b-Rotating about the y-Axis

Conclusion:

The applications of mathematics and mathematical modeling to game design are immense. In classes from algebra to graduate courses, mathematical and statistical concepts can be illustrated and experimented with. The only limitations are time, imagination, and the availability of the necessary hardware and software.