CSC 494F – Computer Graphics Research Project

Deformation using Bezier Curves

Research Paper

Student:	Sam Yip	973 398 520
Instructor:	Alejo Hausner	

Research Period:	September to December, 2001
Paper Completed on:	January, 2002
Document Release:	ver 1.0

(This page left intentionally blank)

Table of Contents

1.0 Abstract

In a paragraph, describe the problem you are solving, your approach, and what you concluded.

This paper documents the process and result of my computer graphics implementation project. The problem investigated is how Free Form deformation can be implemented using Bezier Patches. Free Form Deformations and Bezier Patches will be defined in more detail in Section 2.1.

I created a program that can show the real-time effects of Free Form Deformation. The user of the program inputs the vertices of the object they want to deform. The program will render that object on the computer screen. At that time, the user can use the mouse to deform that object and observe the effects of Free Form Deformation in real-time. The program was developed using Microsoft Visual C++ and the OpenGL libraries.

The program was implemented successfully and it is evaluated in Section 3.5 Evaluation.

2.0 Introduction

2.1 What is the Research Area?

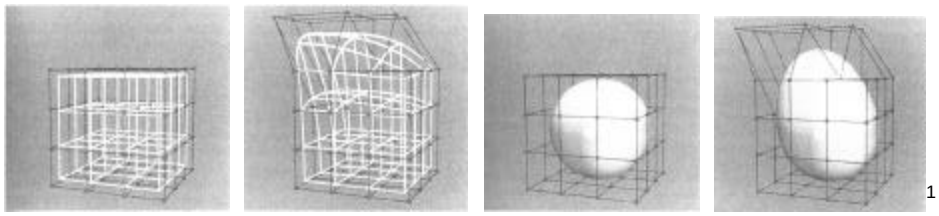
My project deals with Free Form Deformation in Computer Graphics. Let us first define Deformation.

2.1.1 Deformation

Deformation, by definition, is to alter the shape of an object by pressure or stress. Deformation, in the context of computer graphics, is to alter the shape of a computer-generated object. The pressure applied would be simulated by, for example, the drag of the mouse.

2.1.2 Free Form Deformation

Free Form Deformation is one of the techniques of deforming computer-generated objects. Instead of manipulating the object directly, Free Form Deformation deforms a lattice that was built around the object. The lattice is a space, in the shape of a cube, which wraps around the object. When we move the control points of the lattice, the lattice is deformed. At the same time, the object inside the lattice is deformed following the properties of Bezier Patches. The illustrations below demonstrate how FFD works.

 [1]

2.1.3 Bezier Volume

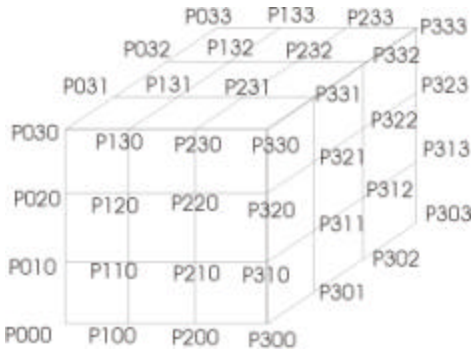
Free Form Deformation is based on the altering of a lattice around the object. This lattice is basically a composite of Bezier patches called a Bezier Volume. Bezier patches are made of Bezier curves. For more information on the Bezier curves and Bezier patches, please refer to *Advanced Animation and Rendering Techniques* by Watt, or the CSC418 fall 2000 Course Notes. We will only describe the formula we used for defining a Bezier Volume. The following is the formula of a tricubic Bezier patch.

$$Q(u,v,w) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} p_{ijk} B_i(u) B_j(v) B_k(w)$$

[1] p. 403. *Advanced Animation and Rendering Techniques – Theory and Practice, Watt. Addison-Wesley, 1992*.

$B_i(u)$, $B_j(v)$, and $B_k(w)$ are the Bernstein polynomials of degree 3 and p_{ijk} represents the 64 control points that define the Bezier Volume. Finally, the parameters u, v, w are between 0 and 1 inclusive.

The 64 control points are labeled like the following on the Bezier Volume.



2.2 What are you implementing?

This project involves the design and implementation of a program that perform Free Form Deformation on objects. The user will input the vertices and faces of the object he/she wants to deform. After that, the program will generate the object, as well as the Bezier Volume around it. At that time, the user can manipulate the 64 control points of the lattice and deform the object within it. The effects of Free Form Deformation are produced in real-time. That is, as the user clicks on a control point and drag it, the result of the object's deformation will be displayed instantaneously.

2.3 Why is it interesting?

One of the major motivations of deformation is to allow computer-animated objects to achieve the 'Squash and Stretch' effect in traditional animation. An example of the 'Squash and Stretch' in traditional animation is how Disney's Mickey Mouse jumps. One can see that his body stretches as he jumps up and squashes as he lands on the ground. This effect has the ability of adding more emotion and impact to dynamic actions. Without deformation, a computer generated Mickey Mouse will jump like a robot, rather than like a spirited mouse.

Free Form Deformation is interesting in that it does not directly deform the object. This can be seen as advantage since altering one control points affects an area of the object. You do not have to go through the time consuming task of changing every vertex of the object. Another advantage of Free Form Deformation is that it can deform any shape, as long as you can wrap a Bezier Volume around it.

Free Form Deformation is also a function that is available in 3D programs like AliasWavefront's Maya and Newtek's Lightwave. The users of those programs can model their own object and then apply Free Form Deformation to it. The implementation of this project will help me understand how the function is actually achieved in those programs.

3.0 The Project

3.1 Outline

Below is a brief overview of the subsequent sections.

References – describe what books I have read about the topic.
Implementation – describe what I steps I took to complete the project.
Results – describe the resulting program by a series of screen shots.
Evaluation – describe the problems I overcame, or did not overcame.

3.2 References

My main reference is the book *Advanced Animation and Rendering Techniques – Theory and Practice, by Alan and Mark Watt. Addison-Wesley, 1992*.

The Internet was also a good source of information. The following WebPages are also my references:

Bezier Volume
http://www.dgp.toronto.edu/~ah/csc418/fall_2001/notes/lectures.html

OpenGL, GLUT and GLUI
http://www.opengl.org/
http://www.cevis.uni-bremen.de/~uwe/opengl/opengl.html
http://www.dgp.toronto.edu/~ah/csc418/fall_2001/a1/q7/

oogl file format reference
http://www.geomview.org/docs/oogltour.html

3.3 Implementation

The implementation process involves the development of the code. Let me breakdown the coding process into several parts and describe each.

3.3.1 Method of Input

The input files must follow the oogl convention. The oogl is a file format that basically lists all the vertices (in three dimensions, x, y, z) and all faces. The oogl convention is described at *http://www.geomview.org/docs/oogltour.html*.

3.3.2 User Interface

The user interface was designed to allow the user to adjust variables in the program. There are basically two types of variables, the environment variables and the control points. The environment variables determine what appears within the program window. For example, whether the lights are turned on, whether the grid lines and control points are visible, or whether the user wants to zoom in towards the object. The control points determine the shape of the

Bezier Volume; hence in turn determine the shape of the object. The mouse manipulates these control points. A click and drag on the control points will move it to a new location. Notice that the program provides 4 different views of the object, one along the x-axis, one along the y-axis, one along the z-axis and one that is a viewing window that the user can manipulate. In the viewing window, the user can rotate the object and view it from their desired angle.

3.3.3 Implementing the Bezier Volume

The calculation of the Bezier Volume is done by helper functions within the program. The control points are kept in a 4x4x4 array.

3.3.4 Programming Logic

From a very high level, the following is how the program flows:

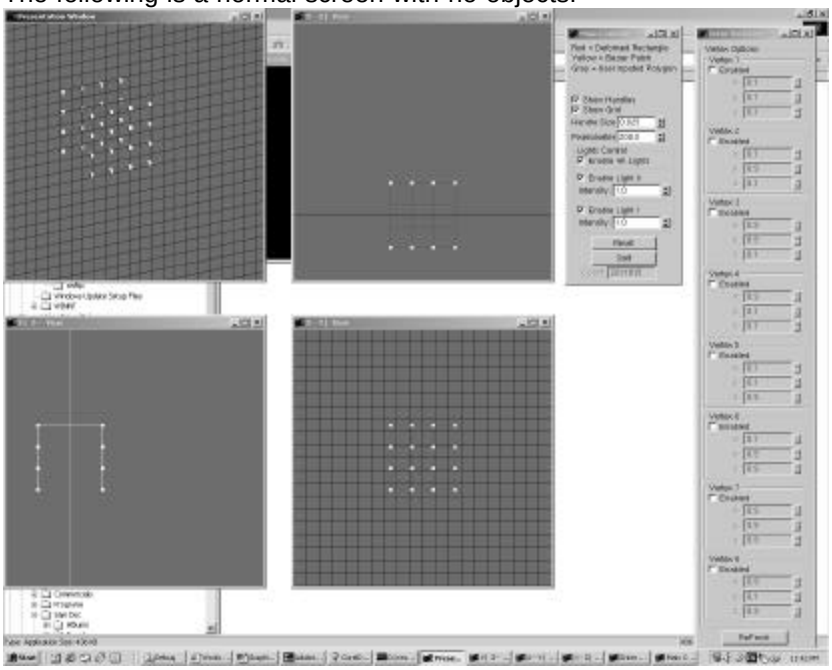
1. Determine the positions of the vertices in lattice or Bezier Volume.
2. Wait for the user to drag the control points
3. After the control points are moved, we calculate the new positions of each vertices according to the altered lattice.

For more details on the implementation, please refer to the actual code.

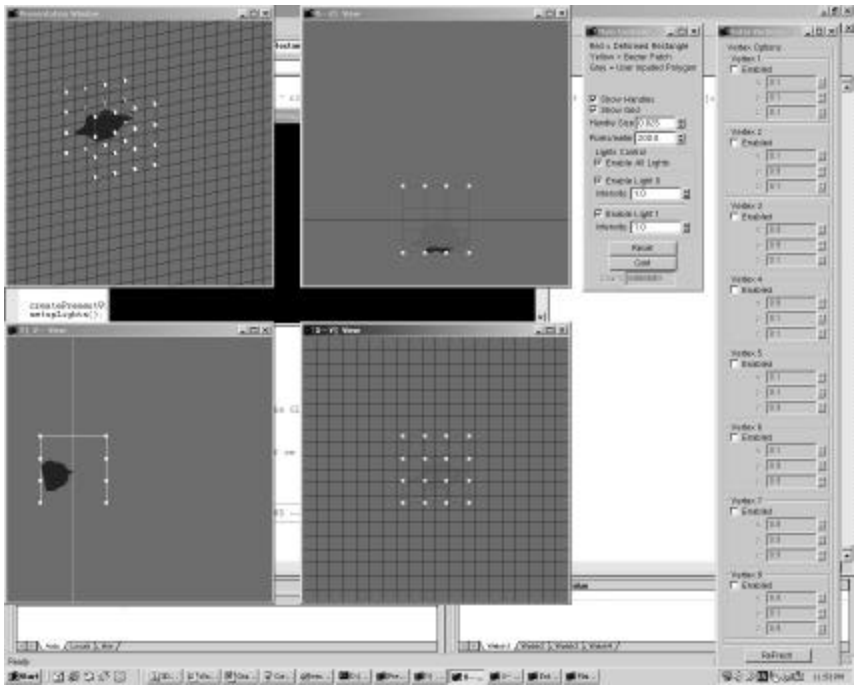
3.4 Results

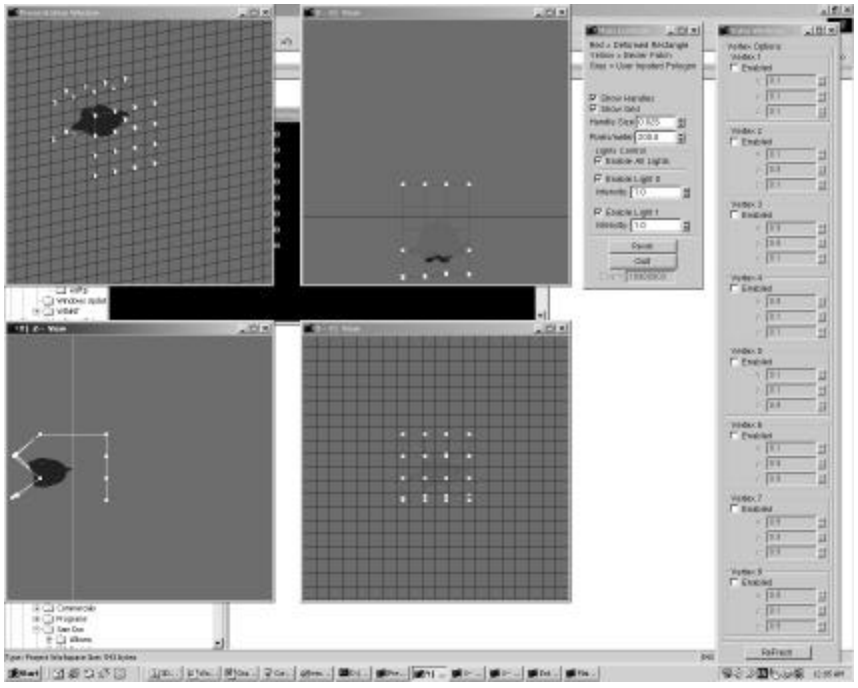
3.4.1 The Program

The following is a normal screen with no objects.



The following are two screenshots of the program, one with the original dinosaur and one with a deformed dinosaur.





3.4.2 Sample Input oogl file format

```
556 1041 0
```

```
-1.25 -0.89 -5.94
-1.17 -1.30 -6.16
-0.86 -0.94 -6.19
1.87 -2.44 -2.87
1.30 -3.04 -3.48
1.18 -2.89 -3.10
0.95 -2.71 -2.71
4     0 1 2 2
4     3 4 5 5
4     3 5 6 6
4     3 6 7 7
```

For explanation of the oogl convention, please go to http://www.geomview.org/docs/oogltour.html

3.5 Evaluation

3.5.1 Successes.

The program itself was successful in achieving its objective, which is the implementation of Free From Deformation on a virtual object. The deformation was correct in the sense that it followed the formula of the Bezier volume.

3.5.2 Remaining Issues.

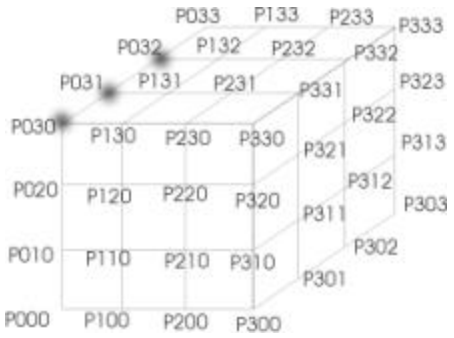
3.5.2.1 Performance

The problem I encountered was that the program gets sluggish when the input has many vertices and faces. For example, an object that has 50 vertices and 50 faces, the program responds quickly. Yet when there are more then 100 vertices and 100 faces, then the program responds very slowly. I have tried to think of ways to increase the efficiency of the algorithms and calculations, yet there are no successes so far. This is because the nature of the algorithm depends on calculating all the points over again and there is no way to escape that step.

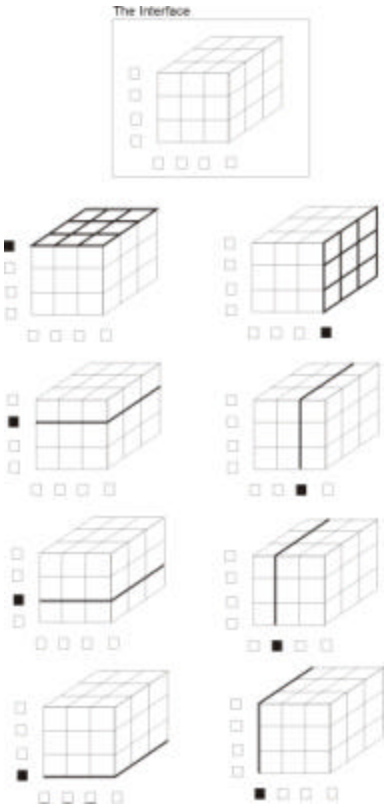
As of now, there are two solutions. First, is to run the code on a fast computer. Secondly, I can change the code so that the program only draws the object after the completion of a click and drag action. That is, the object does not actually get rendered until the user stopped altering the Bezier Volume. Currently, the program runs very slow because it calculates the all the intermediate steps when the user manipulates the control points. Thanks to Professor Alejo Hausner for suggesting this solution.

3.5.2.2 User Interface

The user interface comes into problem when the control points overlap. For example, if you want to move point P032 (below) and you are looking at the Bezier Volume dead on, you must first move away the points that lay on top of it, that is P030 and P031. This is confusing, inconvenient and restricts the user.



As a solution, I plan to add a feature to the User Interface. This feature allows you to select the layer of control points that the user wants to manipulate. There will be 8 layers, each with 16 control points. While you have selected a layer, the user can only manipulate the 16 control points in that layer. Visually, the control points will still overlap, yet there would no longer be the confusion of manipulating overlapping control points because the user have chosen their specific layer. This is because at least one of the three display windows will show that layer in a view that has no overlap of the 16 control points selected. The following could be a possible interface for this menu.



3.5.2.3 Other enhancements

To make the program more interesting, one of the features we could implement in the future is to give the option of constructing a more complex cube. For example, a 5x5, 6x6 Bezier Volume, depending on what the user needs. This feature would be helpful in that the user will be able to generate more detailed Free Form Deformations.

4.0 Conclusion

In summary, this project is the implementation of Free Form Deformation on virtual objects. Mathematically, I used the Bezier Volume to do the calculations. Technically, I used MS Visual C++ and the OpenGL libraries to construct the code. The program has achieved this by first allowing the user to input a file that defines their object. Then the user can alter the control points by dragging the mouse on the program window.

Functionally, the program was able to achieve is goal, yet it becomes slow with large inputs. So efficiency of the program has much room for development.

Finally, I want to make a note that it was a very rewarding and fun experience to work on this project. The schedule of its development is recorded on the website: http://www.cdf.toronto.edu/~g8samyip/home/