# Free-Form Deformation

## 1 Introduction

Free-Form Deformation is a useful technique that enables the maniuplation of objects and surfaces, resulting in a 'sculpting' effect where the model being deformed is pushed or pulled by the movement of control points that control points along a bezier curve that relate to the position of the surfaceof the model. Moving the control points moves the position of the bezier curve which in turn manipulates related sections of the models surface.

Objects that are deformed, either in animation or the user of an application, are referred to as soft bodies, in contrast to rigid bodies, whose shape never changes. The most common uses of soft bodies are in films and videogames. Soft bodies can be used well in cartoon-style animation to add emphasis and exaggeration to characters and objects to convey expression and style. Another use is in realistic deformation of highly flexible or elastic object, one example of this is muscles, including those in the face as well as full body motion. Another use would be shape distortion to show dynamic interaction, a ball compressing as it hits a surface, a can being crushed or a vehicle being damaged in a collision.

Free-Form Deformation is one of the simpler techniques for soft body deformation, and was first documented by Sederburg and Parry (1986).

## 2 Mathematical Concepts of Free-Form Deformation

### 2.1 Coordinate Computation

Figure

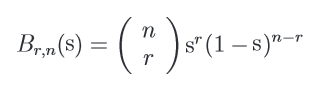
The First step to free-from deformation is to calulate the 'lattice space' coordinate of the point being deformed (Sederberg ,2014). Coordinates at the minima would be valued at 0, and those at the opposite end would be 1, the point being deformed therefore will have coordinates in the range between 0 and 1. Assuming a 3 dimensional shape the local (s,t,u) coordinates of a point with cartesian coordinates (x,y,z) would be:

Figure

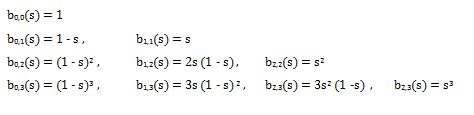
### 2.2 Bezier Curves

Figure

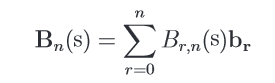
Bezier curves are core to the free-form deformation technique, the bezier curve's location and shape is determined by its control points, and the shape of the curve deforms the model's vertices. Bezier curves are parametric curves with coordinates defined by control shape and Berstein coefficients, calculated from the Berstein Polynomial.

The Bernstein Polynomial can be expressed as:

Figure

For the purposes of this report it will not be necessary to go above the cubic degree, the bernstein polynomials up to this order are:

Figure

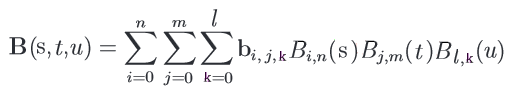
Knowing this, bezier curve of the cubic degree or lower can be found with the formula:

Figure

where Br,n(s) are the berstein polynomials from figure 5 and br are the coordinates of the control points. The (s) in all the above equations is the lattice space coordinate of the point being deformed, calculated in section 2.1.

### 2.3 Bezier Volumes

Figure

Bezier Volumes are an extension of Bezier curves to 3 dimensions, the concepts are the same and the only difference is the final bezier equations which now needs to accounts for all three dimensions:

Figure

## 3 Code Implementation

### 3.1 Initialising the Mesh

Before the mesh could be deformed it's lattice space coordinates need to be set up as mentioned in section 2.1. Using the width, depth, height and central point of the mesh the maximum and minimum points of the mesh are found. Then for every vertex in the mesh the equations form figure 2 are used to calculate the 'lattice space' coordinates which are then stored as a variable that will be needed later.

### 3.2 Initialising the Control Polygon

The Control Points are created in code by looping through all three axes a number of times that is equal to the number of control points that are required in that direction. The psedo-code for this initialisation is:

|  |
| --- |
|  |
| for (x = 0; x < 1; x += 1/xCPs) |
| for (y = 0; y < 1; y += 1/yCPs) |
| for (z = 0; z < 1; z += 1/zCPs) |
| create ControlPoint |
| controlPoint position = minVertex + x\*xScale + y\*yScale +z\*zScale |
| save controlPoint |
| } |
| } |
| } |

After this the connector strips are also added, which was a slightly more difficult task, but not one that is significantly relevant to this report as it has no bearing on functionality, only aesthetic.

### 3.3 Calculating Berstein Coefficients

for the sake of simplicity the number of control points was hard coded...

### 3.4 Deforming the Mesh

For every vertex on the mesh we have already stored it's lattice space coordinates as per section 3.1, we have also calculated the bernstein polynomial as described above in section 3.3. The only thing remaing in order to perform the evalution of the 3D point as described in section 2.3 is the location of the control points, this is simple as they are accessible throughout the script and can be accessed from any function within it.

This allows us to perform the equation in figure 8, summing up all the values of control points multiplied by the relevant polynomials to calculate the new position of the vertex. This value is returned from the function and checked against the current value, if they're different the new value replaces the old. This check is simply to save on processing when no control point has been moved.

### 3.5 Moving Control points

The process for moving the control points is very simple. The user simply holds the left mouse button down over the control point they wish to move, releasing when they no longer wish to move the control point.

## 4 Results

### 4.1 2D Surface



Figure Undeformed 2D Surface



Figure Deformed 2D Surface

As can be seen the 2 dimensional surface deformed well, the surface had 100 vertices (10 by 10)

### 4.2 3D Shape

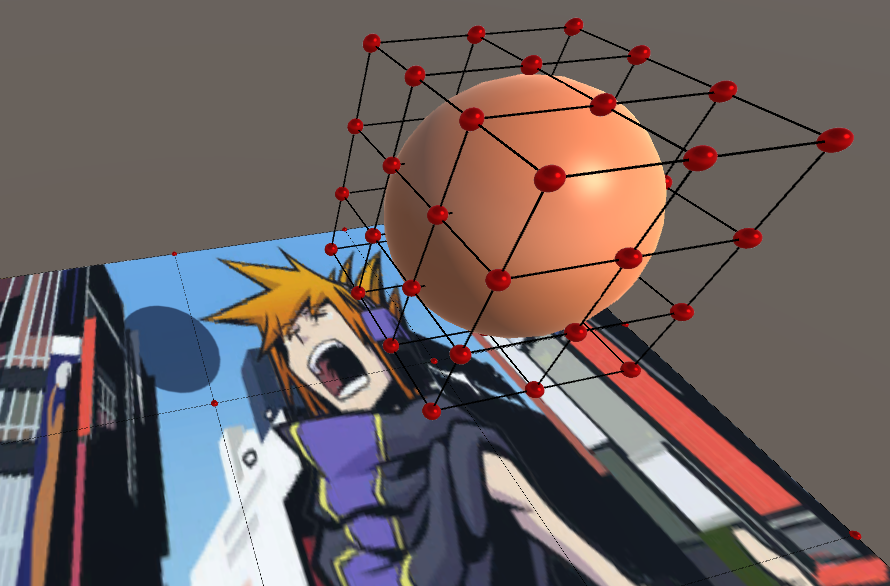


Figure Undeformed Textureless Sphere

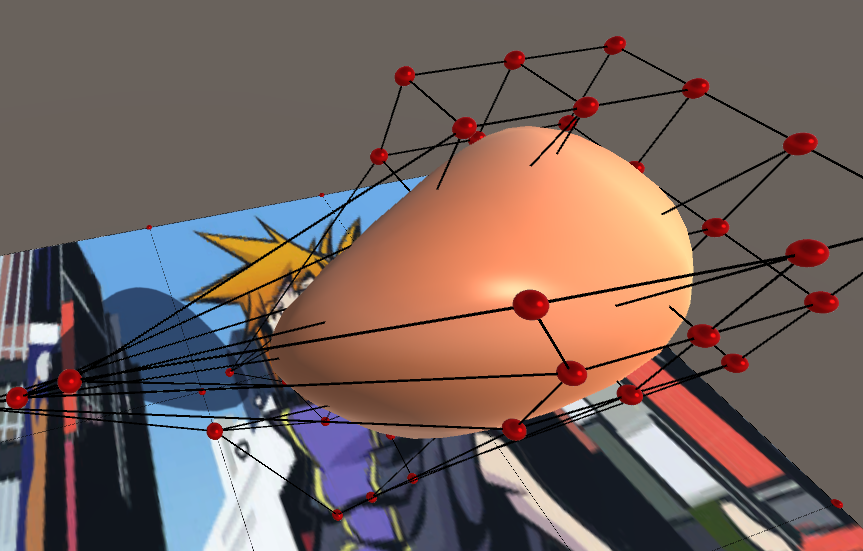


Figure Deformed Textureless Sphere

The application also had no trouble deforming a basic sphere, it should be mentioned that there are seperate scripts for free-form deformation of surfaces and for areas. the reason for this is that the 2D script for surfaces was developed first as a test, upon its success the 'proper' area deformation script was developed.4.3 3D Model

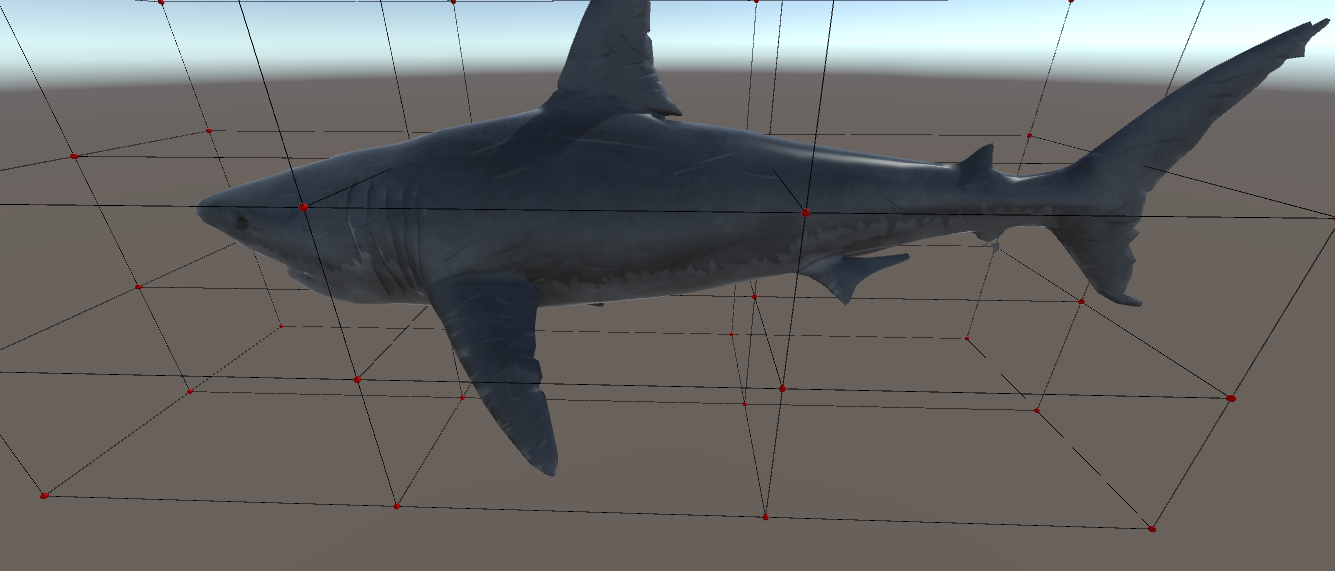


Figure Undeformed Textured Model

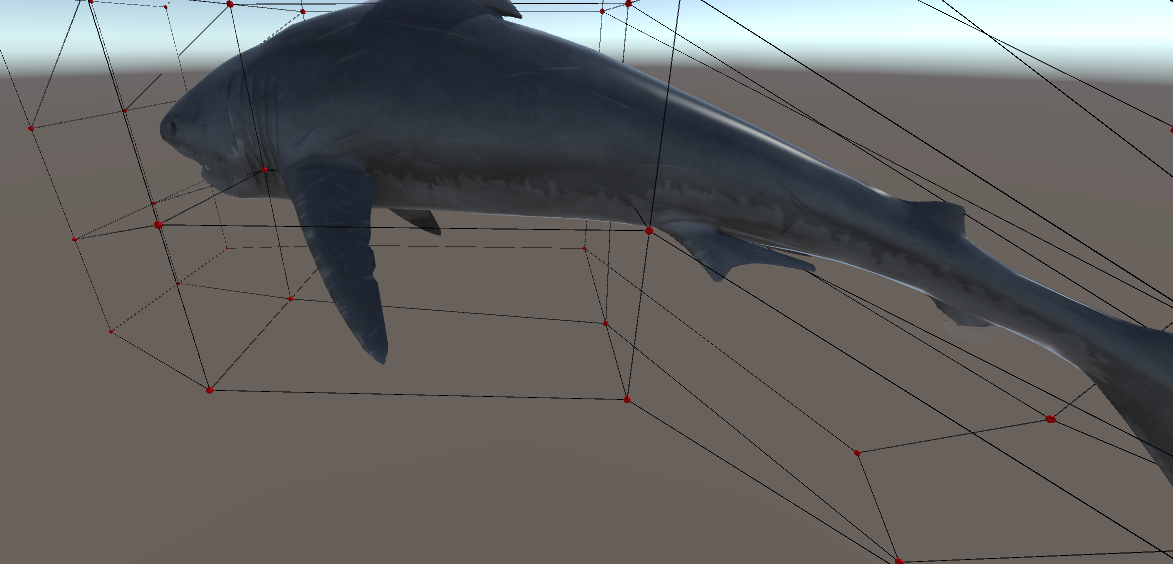


Figure Deformed Textured Model

The textured model also deformed well, maintaining its overall structure and texture. It is possible to deliberately 'break' it by moving a number of control points to the opposite side, but the model holds up fairly well and it would probably require someone deliberately trying to break it or deform the model in a very extreme way for it to start looking very weird.

## 3 Conclusion

### 3.1 A

a

Free-Form Deformation of Solid Geometric Models Thomas W. Sederberg Scott R. Parry t

http://tom.cs.byu.edu/~557/text/cagd.pdf

http://web.cs.wpi.edu/~matt/courses/cs563/talks/freeform/free\_form.html

http://www.darwin3d.com/gamedev/articles/col0600.pdf

https://classes.soe.ucsc.edu/cmps160/Spring05/finalpages/pati/

https://www.student.cs.uwaterloo.ca/~cs779/Gallery/Winter2013/Pharoah/

http://www.gamasutra.com/view/feature/131779/realtime\_softobject\_animation\_.php