# 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 originally proposed by Sederburg and Parry (1986).

## 2 Mathematical Concepts Behind Free-Form Deformation

### 2.1 Coordinate Computation

### 

Figure 1

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:  
2.2 Bezier Curves

Figure 2

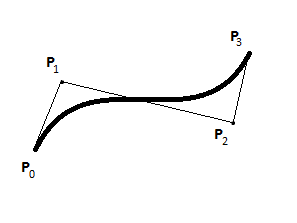


Figure 3

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 4

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 5

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

Figure 6

where are the berstein polynomials from figure 5 and is the coordinates of the control points. The 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 7

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 8

## 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 marginally 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 is a set value that cannot be changed by the user. For the 3D deformation a 4x4x3 control polygon is used, which gives good control of the object to the user, without making a too heavy sacrifce in computational time

Since the size of the control Polygon is known before runtime the bernstein coefficients are easier to calculate, as the equations will not change during program execution. As a result the equations from figure 5 can be used. If the size of the control polygon were not known, the Berstein coefficients would have to be calculated as shown in figure 4.

### 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 9 Undeformed 2D Surface



Figure 10 Deformed 2D Surface

As an initial test the deformation for a 2D surface was calculated, the surface is a 10 by 10 mesh with 100 vertices. The control polygon is a 4 by 4 grid with 16 control points to control the postion of the grid. A texture was applied to the surface to better demonstrate the deformation. as can be seen the deformation is succesful and looks smooth.

### 4.2 3D Shape

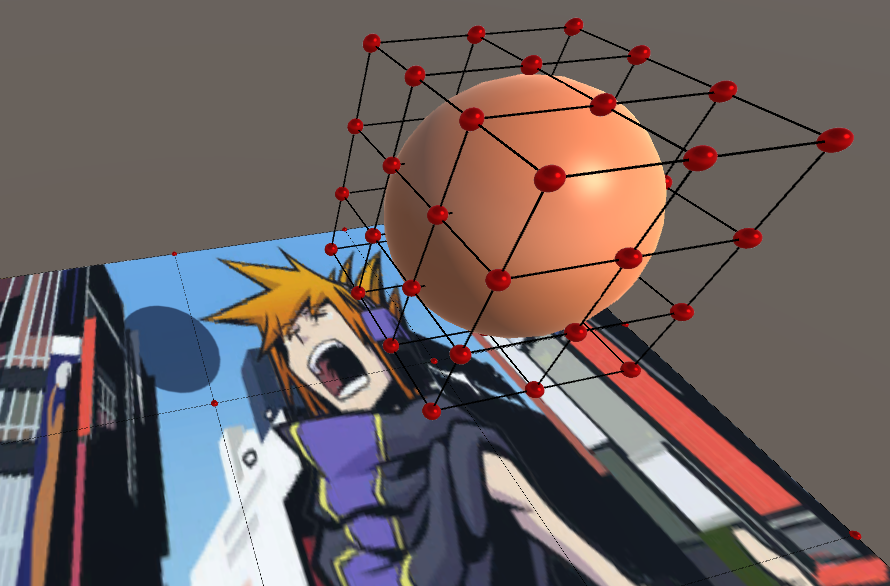


Figure 11 Undeformed Textureless Sphere

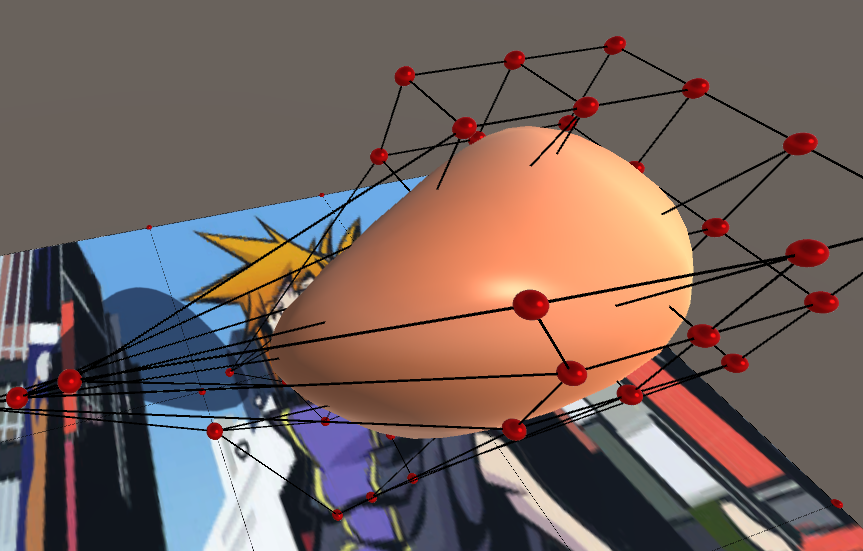


Figure 12 Deformed Textureless Sphere

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 area deformation script was developed as an extension of the concepts and code learned in the 2D case.4.3 3D Model

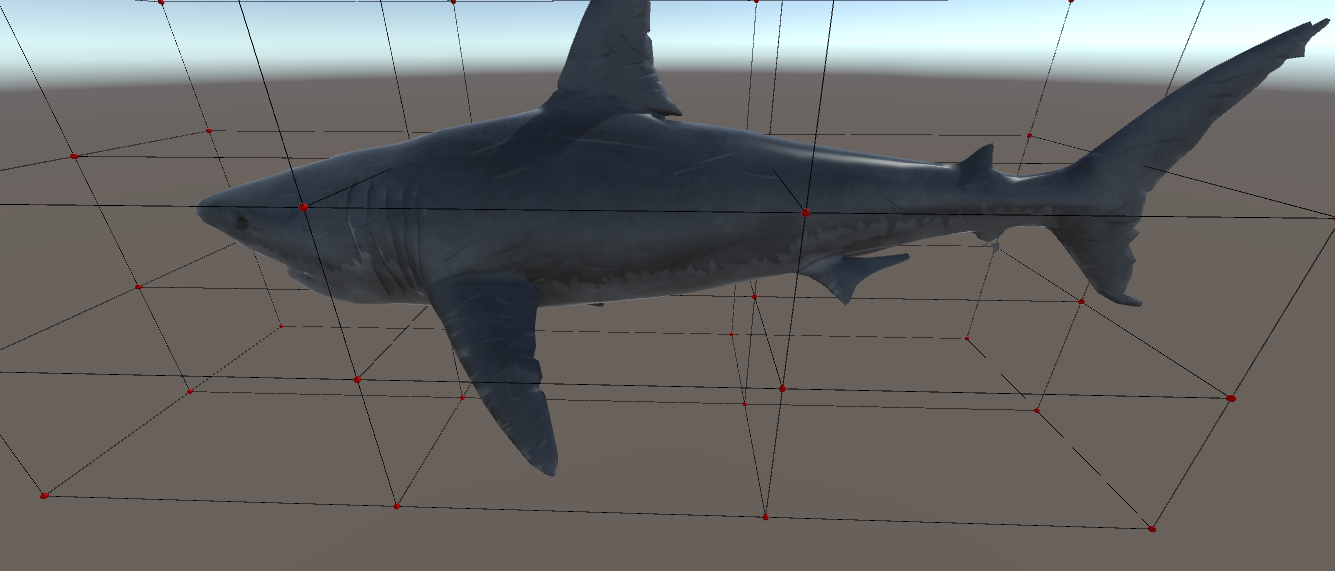


Figure 13 Undeformed Textured Model

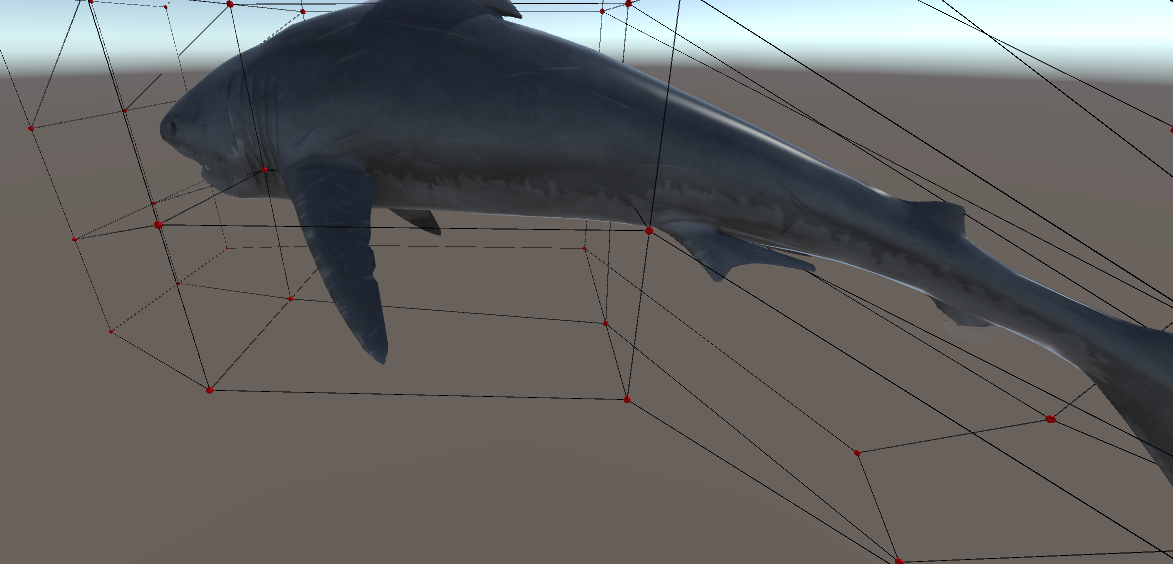


Figure 14 Deformed Textured Model

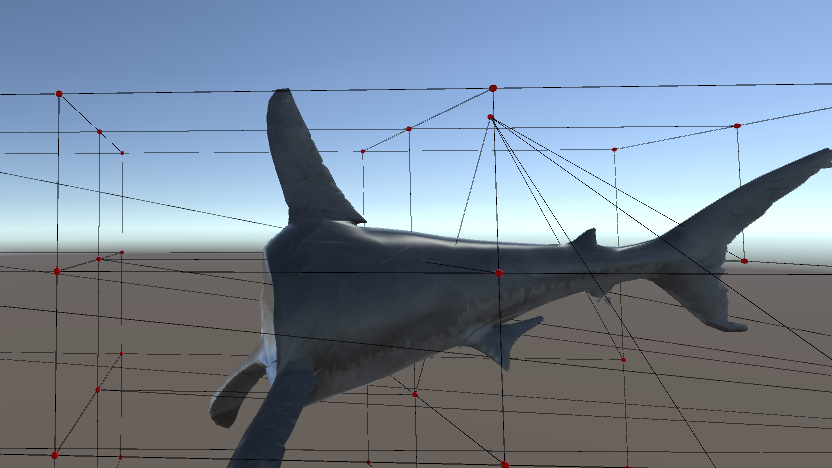
finally the deformation script was applied to a 3D textured model. The textured model also deformed well, maintaining its overall structure and texture. It is possible to deliberately 'break' it by moving a number of control points to the opposite side, but the model holds up fairly well under a general use case and it would probably require someone deliberately trying to break it or deform the model in an unusual way for it to start to look odd.

## 5 Extension Work

### 5.1 Volume Preservation

Free-Form deformation in no way considered the realistic constraints of the object being deformed, the model is simply moved as the control points dictate. However many objects that require deformation have inherent constraints such as internal structure, elasticity or other phyiscal forces. As a result it is difficult to to use Free-Form Deformation to to express deformation of objects with inherent constraints (Aubert & Bechman, 1997)

For example, our model shark can be deformed back in on itself in a way that is a physical impossibility for a real shark as shown in Figure 15.



Figure

From the laws of physics comes the principle of conservation of mass. If the density of a given matrial is constant, then this implies conservation of volume (Hirota, Maheshwari & Lin, 1999). One way that the realism of deformation and animation can be improved is by introducing volume preservation methods (Hahmann et al, 2011). This is a technique where after the deformation has been computed as normal, an additional step, the volume correction step, is introduced to recover lost volume (or reduce increased volume).

In the original paper(1986) it was noted that the change in volume after the deformation can be calculated with the Jacobian. The Jacobian matrix is a matrix which contains every first order partial derivatives for a given vector function. Total preservation of volume occurs therefore when the Jacobian matrix of the deformation is equal to 1. The Jacobian will be:

If we take the FFD to be given by:

The process of making the statement ' ' true, however is not so easy. The common method seems to be calculating the volume of the shape, calculating the volume derivative, then applying constraints to the control point configuration so as to prevent the change in volume (Aubert & Bechmann, 1997. Rappoport et al, 1996. Hirota, Maheshwari & Lin ,1999). In this volume correction step the FFD lattice is recomputed to be as close as possible to the user defined deformation. these techniques are either time consuimg, or will only work with small scale deformation, however improvements and optimisations to make the speed of this technique faster and increase its viability have been made (Hahmann at al, 2011).

## 6 Conclusion

Free Form Definition is a powerful technique for deforming meshes that allows the user to exert a lot of control over the psotion of vertices of a mesh in a user-friendly manner, rather than controlling every vertex individually the position of the vertexes is controlled by a control polygon.

The processing time for the maths required is not too long for 3D control shapes with bezier curves of the cubic order or lower, which requires 64 iterations of the main loop for every vertex. But as the number of control points increase the number of iterations rapidly increases, especially a problem for meshes with a high number of vertices.

Free-From Deformation is somehwat of a simple technique to implement, the code required is not too difficult to understand, with the entire deformation script being only 200 lines (this number could be optimised), less than 100 of which directly deal with the computation of the control nodes and mesh vertex locations.

Free-Form definition has a good 'feel' in that the mesh deforms as the user would expect, is intuitive for the user to operate, is not too challenging to code and runs efficiently provided the number of control points and model detail does not go too high. Overall, Free-From Deformation is a technique that should definitely be considered if deformation of soft objects is needed.

Furthermore, considering the improvements and research that has contributed to its improvement since its inception there are many benefits and strengths to using this technique.

## 7 References

Rappoport A, Sheffer A, Bercovier M. 1996. Volume-preserving free-form solids. IEEE Transactions on Visual Computer Graphics, vol 2(1), pp.19–27.

Aubert F & Bechmann D. 1997. Volume-preserving space deformation. Computer & Graphics, vol 21(5), pp.625–39.

Hahmann S et al. 2011. Volume Preserving FFD for Programmable Graphics Hardware. Visual Computer, 28(3), pp.231-245.

Hirota G, Maheshwari R & Lin M. 1999. Fast volume-preserving free-form deformation using multi-level optimization. In Proceedings of the fifth ACM symposium on Solid modeling and Applications, pp.234-245.

Sederberg T. 2014. Computer Aided Geometric Design. pp.149-154. [online] Available from: http://tom.cs.byu.edu/~557/text/cagd.pdf. [Accessed 27th April]

Sederberg T & Parry S. 1986. Free-form deformation of solid geometric models. Computer Graphics, vol 22(4), pp.151-160. SIGGRAPH 86.

## 7 Bibliography

Barr, A.H. Global and Local Deformations of Solid Primitives. Computer Graphics, vol 17(3), pp 21-30.

Coquillart S. 1990. Extended free form deformation: a sculpturing tool for3D geometric design. Computer Graphics, 24(4), pp.187–193. SIGGRAPH 90.

Ferrier A. Real-Time Soft-Object Animation Using Free-Form Deformation. [online] Gamasutra. Available from: www.gamasutra.com/view/feature/131779/realtime\_softobject\_animation\_.php [Accessed 27th April]