Direct Manipulation of Free-Form Deformations

William M Hsu[1]
Cambridge Research Lab
Digital Equipment Corporation

John F. Hughes[2] and **Henry Kaufman**[3]
Department of Computer Science
Brown University

Abstract

Free-form deformation (FFD) is a powerful modeling tool, but controlling the shape of an object under complex deformations is often difficult. The interface to FFD in most conventional systems simply represents the underlying mathematics directly; users describe deformations by manipulating control points. The difficulty in controlling shape precisely is largely due to the control points being extraneous to the object; the deformed object does not follow the control points exactly. In addition, the number of degrees of freedom presented to the user can be overwhelming. We present a method that allows a user to control a free-form deformation of an object by manipulating the object directly, leading to better control of the deformation and a more intuitive interface.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Curve, Surface, Solid, and Object Representations; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques.

Additional Keywords: Direct manipulation, free-form deformations.

1 Introduction

Geometric modeling of complex objects is a difficult task. Sophisticated techniques for shaping and creating complex objects are generally awkward and tedious to use [8]. Free-form deformation [15] falls into this category. It is a powerful modeling technique that enables the deformation of objects by deforming the space around them, but using this technique is sometimes difficult. The deformations are defined by parametric functions (3D splines) whose values are determined by the location of control points. Describing a free-form deformation (FFD) in conventional modeling systems is done by manipulating these control points, an interface that reflects the underlying mathematics of the modeling method. This type of interface can be confusing because the control point movement merely hints at the type of deformation the object will be subjected to. The following examples will help to clarify this.

Although the movement of the control points gives an indication of the resulting deformation, some shapes are not intuitive to form. As a first example, to create a bulge with a flat top one may think to align the control points to a plane, as shown in Figure 1a. However, it is actually necessary to position the control points as shown in Figure 1b to create the flat top. As a second example, Figures 6 and 7 show the prongs of a ring modeled with free-form deformations. Precise placement of the prongs is needed to ensure that they do not penetrate the gem stone.

Complex deformation operations often require a large number of control points resulting in screen clutter. They also tend to get buried within the model being deformed. As a result, it is virtually impossible to select or manipulate the control points efficiently.

Thus we can see four problems in manipulating deformations via control points.

1. Exact shape is difficult to achieve.
2. Exact placement of object points is difficult to achieve.
3. Users unfamiliar with splines do not understand the purpose of the control points and the results of their movement.
4. The control points become difficult to manipulate when occluded by the object being deformed, or when there are so many they clutter the screen.

One way to improve the usability of this technique is to move control points in groups, and then apply linear and nonlinear transformations to them, similar to the group control point manipulation presented in [5] for spline surfaces. While helping the user move many control points at one time, this

[1] One Kendall Square, Cambridge, MA 02139. email: hsu@crl.dec.com. phone: 617-621-6645

[2] Box 1910, Providence, RI 02912. email: jfh@cs.brown.edu. phone: 401-863-7638

[3] Current address: 3D Ltd. 4 Belinson Street, Tel-Aviv 63567, Israel

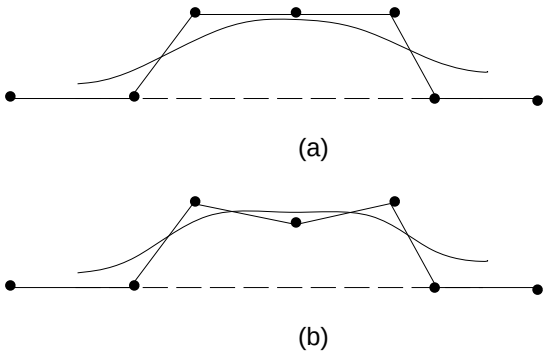

Figure 1: An FFD in the plane. The dashed line shows the original shape, and the solid line shows the shape after the deformation. (a) shows the result of a flat line of control points. (b) shows the control point configuration to create a flat top.

does nothing to alleviate the shape and placement problems. It is unclear which control points should be moved and how transformations will affect the object. The limited usefulness of this approach for spline surfaces was noted by [14]; the 3D volume of control points for FFDs (in contrast to the 2D mesh of control points for spline surfaces) exacerbates the difficulties of deciding how an aggregate move should be performed.

Another approach to an easier and more intuitive interface is the Extended Free-Form Deformation (EFFD) technique of [6]. With EFFDs, the user configures the initial lattice of control points to the approximate shape of the intended deformation, instead of starting with the FFD's parallelepiped of control points. EFFDs are quite effective for creating impressions, reliefs, and other fairly simple deformations that might otherwise be difficult to achieve with FFDs. However, the user must know the general shape of the deformation before starting to model, and the interface is still a direct representation of the underlying mathematics.

Both FFDs and EFFDs are based on the notion of deforming the underlying space in which an object lies. This has the advantage that it can be applied to any parametric or polygonal model, and is therefore not restricted to any class of objects. On the other hand, the control lattice used to manipulate the underlying space is not directly related to the object being deformed. Therefore, a control point that happens to be close to the surface of the object (which is, after all, the focus of the user's attention) may be far from the object surface after the deformation. Thus, these methods may surprise a user who does not understand the distinction between the object and the space in which it lies.

In this paper, we develop a direct manipulation technique which makes formation and placement of deformations easier. The essential idea is that the user selects (with some sort of pointer) a point on an object and then moves the pointer to a location where that object point should be. Our technique computes the necessary alteration to the control points of the FFD spline that will induce this change. This alteration is generally under-determined; we use a least squares approach to select a particular alteration.

The rest of the paper is structured in 4 sections. Section 2 describes FFDs, and introduces B-spline FFDs. Section 3 describes a direct manipulation interface to B-spline FFDs, in which the user describes actions, and these actions are converted into control point displacements that will effect the actions. Section 4 discusses related work in direct manipulation interfaces, possible applications and directions for future research. Section 5 summarizes the results of the paper.

2 Free-Form Deformation

The FFD method deforms an object by first assigning to each of its points within the deformation lattice a set of local coordinates. The local coordinate system is defined by a parallelepiped-shaped lattice of control points with axes defined by the orthogonal vectors **s**, **t**, and **u**, as shown in Figure 2. All object points within this parallelepiped are assigned local coordinates through a mapping applied to their xyz-coordinates; we describe this mapping later.

Once the control points are moved, the new location of an object point is then determined by a weighted sum of the control points. The weights are functions of the local coordinates originally assigned to the point. Hence, a positional change of the control points changes the location of the object point.

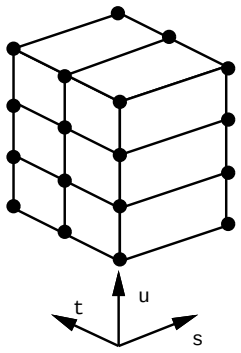

Figure 2: A lattice of control points. The $\mathbf{s}$, $\mathbf{t}$, and $\mathbf{u}$ vectors define the local coordinate system

In our implementation, the deformation function is a trivariate B-spline tensor product. We use the B-spline basis instead of the Bernstein polynomials used by Sederberg and Parry because of the local control properties of B-splines. Local control is desirable for both aesthetic value and for efficient computation with large control point lattices. We also prefer B-splines for its guaranteed continuity when any of its control points are moved, in contrast to, for example, Bézier splines.

In summary, then, the deformed position, $\mathbf{q}$, of any arbitrary point with local coordinates, (s, t, u), is given by

$$\mathbf{q}_{i,j,k}(s,t,u) = \sum_{l,m,n=-3}^{0} \mathbf{P}_{i+l,j+m,k+n} B_l(s) B_m(t) B_n(u) \quad (1)$$

where $\mathbf{P}_{i,j,k}$ is the i^{th}, j^{th}, k^{th} control point in the $\mathbf{s}$, $\mathbf{t}$, and $\mathbf{u}$ direction, respectively, and the Bs are the B-spline blending functions.

In our implementation of FFD we allow both direct manipulation of the object and manipulation of the control points. A drawback of using B-splines is that the image of the B-spline does not fill the convex hull of the control lattice, if the control lattice is evenly spaced and all control points have multiplicity one. We compensate for this by giving the outer control points of the lattice a multiplicity of three, which ensures that the image of the B-spline is the convex hull of the control lattice.[4] Phantom control points could be used as well; constraining their positions guarantees C^2 continuity along the borders. See [3] for more details.

Before the deformation is applied, object points must first be assigned local (s, t, u) coordinates, as already mentioned. When the control lattice is in its initial position, it defines an injective map from its domain to the convex hull of the lattice. Thus each point $\mathbf{w}$ within this hull is $\mathbf{q}(s_0, t_0, u_0)$ for some s_0, t_0, u_0 in the parameter space of the B-spline. The numbers s_0, t_0, u_0 are the local coordinates we assign. To compute them, we must invert the B-spline map. We first determine the spline segment $\bar{\mathbf{q}}_{i,j,k}$ that contains the object point. Then we compute s_0, t_0, and u_0 by explicitly solving the cubic equations $\bar{\mathbf{q}}_{i,j,k}(s_0, t_0, u_0) = \mathbf{w}$.[5] Note that the local coordinates need only be computed once for a given lattice, and not for each deformation calculation.

3 Direct Manipulation

In this section, we describe an interaction technique that converts a user action of the form "move *this* point of the object to *there*" and finds control point positions that will effect this action. We first describe the method in the case where the user wants to move a single *selected object point* to a new position, or *target point*. We then build upon this technique to describe how multiple selected points can be moved simultaneously. Although we demonstrate this method with the B-spline FFD, it can be used in conjunction with any other spline basis.

[4] If control points are not displayed at all, then all the control points in the lattice can be of multiplicity one, and the region deformed can be represented by the border of the B-spline image. This simplifies the deformation equation, and the latter portion of section 3.1 can be dismissed.

[5] General root finding is needed only for the outer two segments due to the tripling of control points at the borders. Otherwise, (s_0, t_0, u_0) can be found by the position of the object point in relation to the segments that contain it directly, as was done in the original FFD paper [15].

3.1 Single point constraint

As the user moves a target point our goal is to configure the control points such that the deformed location of the selected point matches the target point location. This problem is under-determined; there are many control-point configurations that will yield the same deformed location for the selected point. One obvious, but not very useful, solution is to simply translate all the control points by the target point's translation. Another solution is to choose the nearest control point and translate it until the target point reaches the desired location. A more natural solution is one that moves the control points the least (in the least-squares sense). The blending functions of Equation (1) assign weights to the control points for a given target point. The closer the control point is to the target point the greater the weight, or influence, the control point has. By using a least squares solution, control points are moved such that the resulting surface reaches its intended destination while the effect of the deformation smoothly tapers off. This effect provides predictable and physically intuitive behavior. We begin with some linear algebra.

Recall from Equation (1) that the deformed object point location, $\mathbf{q}$, is a linear function of 64 control points, $\mathbf{P}$, which can be written in matrix form as $\mathbf{q} = \mathbf{BP}$, where $\mathbf{B}$ is a single row matrix of the blending functions, and $\mathbf{P}$ is an 64×3 array whose rows are control point coordinates. (Henceforth we write coordinates of all points as row vectors.) A new location for the point $\mathbf{q}$, $\mathbf{q}_{new}$, is then $\mathbf{q}_{new} = \mathbf{B}(\mathbf{P} + \Delta\mathbf{P})$, or

$$\Delta\mathbf{q} = \mathbf{B}\Delta\mathbf{P} \quad (2)$$

where $\Delta\mathbf{P}$ is the change in position of the control points and $\Delta\mathbf{q}$ is the change in position of the object point. We are given $\Delta\mathbf{q}$ (the difference between the target point and the selected point), and wish to find a value of $\Delta\mathbf{P}$ satisfying Equation 2. To do this we use the *pseudoinverse* (often referred to as the *generalized inverse*) $\mathbf{B}^+$ of $\mathbf{B}$.

Digression on Pseudoinverses Given a system of linear equations $\mathbf{y} = \mathbf{Bx}$, the pseudoinverse $\mathbf{B}^+$ is a matrix where $\mathbf{x}_0 = \mathbf{B}^+\mathbf{y}$ is the best solution, in the least squares sense, to the system of equations, (i.e., for which $||\mathbf{Bx}_0 - \mathbf{y}||$ is minimized and $||\mathbf{x}_0||$ is as small as possible [12]). The pseudoinverse is computed by first representing the $m \times n$ matrix $\mathbf{B}$ in the form $\mathbf{B} = \mathbf{CD}$, where $\mathbf{C}$ is $m \times k$ and $\mathbf{D}$ is $k \times n$, so that all three matrices $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ have rank k. The general formula for the pseudoinverse $\mathbf{B}^+$ of $\mathbf{B}$ is then given by

$$\mathbf{B}^+ = \mathbf{C}^T(\mathbf{CC}^T)^{-1}(\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T \quad (3)$$

This formula can be used for both under-determined and over-determined systems of equations. When the problem is under-determined, as with the single target point constraint, only $(\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T$ is needed to compute the pseudoinverse, and $\mathbf{B} = \mathbf{D}$. Likewise, the pseudoinverse for the over-determined case is computed by $\mathbf{C}^T(\mathbf{CC}^T)^{-1}$. $(\mathbf{DD}^T)^{-1}$ reduces to $1/||\mathbf{D}||^2$, and the pseudoinverse of the single-row matrix $\mathbf{B}$ can now be found by the equation

$$\mathbf{B}^+ = \frac{1}{\|\mathbf{B}\|^2}\mathbf{B}^T \quad (4)$$

(end of digression).

Once the pseudoinverse of $\mathbf{B}$ is determined, the change in position of the control points based on the movement of the target point can be expressed as

$$\Delta\mathbf{P} = \mathbf{B}^+\Delta\mathbf{q} \quad (5)$$

Because the pseudoinverse gives a least-squares solution, the change in control point positions is minimized.

This solution, however, applies only when all control points are allowed to move independently. Recall from Section 2 that in our implementation the control points on the outer border have a multiplicity of three, and therefore must be coincident. To formulate the pseudoinverse equation to reflect this constraint, a matrix, $\mathbf{S}$, which selects the proper control point position is added to Equation (2), so that the deformed object point location is defined by

$$\Delta\mathbf{q} = \mathbf{B}\mathbf{S}\Delta\mathbf{P} \quad (6)$$

The matrix $\mathbf{S}$ is the identity matrix, if all control points are allowed to move freely. Control points that must be coincident with one another have the one in their row shifted to the column that corresponds to the control point it must follow. For example, in the one-dimensional border case, if $\mathbf{P} = [\mathbf{p}_{-2}\mathbf{p}_{-1}\mathbf{p}_0\mathbf{p}_1]^T$, where $\mathbf{p}_{-2}$ and $\mathbf{p}_{-1}$ are required to be coincident with $\mathbf{p}_0$ (i.e., $\mathbf{p}_0$ has a multiplicity of three), then

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The equation for the pseudoinverse $(\mathbf{B}\mathbf{S})^+$ is

$$(\mathbf{B}\mathbf{S})^+ = (\mathbf{D}\mathbf{S})^T(\mathbf{D}\mathbf{S}\mathbf{S}^T\mathbf{D}^T)^{-1} \quad (7)$$

For efficiency, $\mathbf{S}$ can be compressed to a vector, and $\mathbf{B}^+$ need be computed only once for a given target point.

3.2 Multiple target point constraints

The same technique is used to move several selected points to new targets simultaneously. Precise control over shaping objects becomes easier. When the multiple selected points are independent (i.e., when they share no control points), solving for control point position is a straightforward extension of the single target point method.

When selected points are influenced by the same control point, the system of equations must be designed so that each control point only appears once in the array $\mathbf{P}$. The number of columns of $\mathbf{B}$ is the number of distinct control points affecting the selected points. The number of rows of $\mathbf{B}$ will be the number of target points. In a one-dimensional analog of this situation, if we want to move two selected points that share three control points, then the dimension of $\mathbf{B}$ is 2×5 and $\mathbf{P}$ would list 5 control points. The blending functions in $\mathbf{B}$ are arranged in accordance to the listing of the control points. In this example, the equation becomes

$$\begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \end{bmatrix} = \begin{bmatrix} b_0^0 & b_1^0 & b_2^0 & b_3^0 & 0 \\ 0 & b_0^1 & b_1^1 & b_2^1 & b_3^1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix}$$

where $\mathbf{q}_0$ is affected by control points 0-3 and $\mathbf{q}_1$ is affected by control points 1-4, and b_j^i are the blending functions used to compute the location of the i^{th} selected point.

Once again, the pseudoinverse of $\mathbf{B}$ is calculated using Equation 3 and the new control point locations are determined by Equation 5. Figures 3 to 5 show how multiple constraints can quickly effect a change in the shape of an object.

As more target points are added, the problem *can* become over-determined. For example, if a user tries to create a wavy surface with more undulations than is possible to generate with the given B-spline, then the pseudoinverse cannot provide a complete solution. The pseudoinverse has, however, the property of providing the solution with the least squared error, which is the best solution considering the given constraints. Furthermore, the failure to move the selected points to the target points can be quantified; large errors suggest to the user the need to use a B-spline with a finer mesh.

4 Discussion

4.1 Other direct manipulation techniques

Direct manipulation has long been used as a 3D modeling technique for polygonal meshes [13]. However, we find that coupling the free-form deformation technique with direct manipulation is a richer modeling tool with several advantages over polygonal and purely spline-based modeling methods. FFDs work independently of the underlying data structure of the object being deformed, and hence can be applied to any parametric or polygonal model. An implication of this is that FFDs are "resolution" independent. Complex objects can be modeled in real-time by rendering them in low resolution, which can later be rendered at high resolution using the same deformation description. Though a procedural language may provide similar capabilities for a polygonal modeler, some restrictions apply. For example, vertices moved by the user in one level of mesh refinement must have a corresponding, coincident vertex in every other level of refinement [1].

Since the FFD technique deforms the space within it, another advantage is that the same description can be used for several objects. The deformation is dependent on the relative position of the control points. The control points undergo rigid transformations and scaling without affecting the general shape of the deformation, which is useful when applying the same deformation definition to objects of different size. If more than one object lies within the deformation

space, the deformation can be applied to all objects, preserving automatically their relative position and spacing.

Recent developments have been made in the direct manipulation of B-splines. Forsey and Bartels allow direct manipulation of hierarchical B-spline surfaces [9], but only at the B-spline joints, severely limiting the possible shapes that can be formed. The method was extended by Bartels and Beatty to manipulate spline curves at arbitrary points [2]. Their method is based on the Householder transformation, which computes a weighting function that relates positional changes in the target point to positional changes in the control points. In [10], Fowler and Bartels have extended the technique to include the manipulation of the first and second derivatives of the function at an arbitrary point as well.

Recently, [16] independently developed a system for direct manipulation of B-spline surfaces, based on their differential manipulation technique. This technique uses the Jacobian to "suggest" the direction of movement, and through least square projection uses the inverse of the Jacobian to solve for the position of the control points. Though this method for direct manipulation is similar to the method presented in this paper, it is applied only to B-spline surfaces. In contrast, the method described in this paper merely requires that the substrate in which the model lives (namely 3-space) be the image of a 3D spline; this is a property of the substrate and not of the model, and hence lets the technique apply to all polygonal models as well. Also, since our FFD technique is an "indirect" method of modeling, lattices of different size and resolution can be used on the same object to create a multitude of different curvatures.

4.2 Application

In addition to modeling static models, the direct manipulation technique can automate some forms of animated deformations. For instance, the technique can be used to simulate "Play-Doh®[6] physics," where objects deform when they are pressed against other objects, but without the complexities of simulating momentum transfer and non-rigid behavior. This level of simulation is useful to animators who want full object motion control, while still desiring automatic deformation in response to interpenetration or object collisions. In addition, this technique could be used to construct the final deformation lattices for Animated Free-Form Deformations (AFFD) [7]. In general, direct manipulation could be easily incorporated into EFFD (which AFFD is based upon) as a means for interactive shape control.

4.3 Future Research

Though the general technique for direct manipulation of free-form deformations has been implemented, further research is needed to provide a complete and robust user interface. Intuitive and easy to use techniques for moving aggregates of object points are needed. Some widgets we have developed are based on the idea of using a magnet or suction cup to move several points at a time [11]. It would be desirable for users (especially naive users) not to deal with control points at all. The proper metaphors for controlling the resolution of the lattice of control points and the spacing between the points must therefore be developed. Other aids, such as highlighting the area affected by the deformation can convey information that was previously conveyed by displaying control points. In general, a comprehensive metaphor needs to be developed to fully hide the details of the FFD technique and make the interface as transparent as possible. Creating a metaphor that is both believable and general enough to encompass all operations is a difficult task and will require further study [4]. We envision an environment where users will be able to sculpt objects using a Dataglove-like input device. The finger tips, digits, and palm of the hand will be tracked to offset selected points in a malleable object, with smooth valleys and hills attained by the FFD operation. Different elasticities can be assigned to the object by varying the resolution of the control-point lattice. Perhaps a metaphor of molten metal or glass may be appropriate, where a blow torch and cold air are used to heat and cool the object to give it different molding properties. By making modeling as natural as possible, or by imitating the ways it is done in the real world, a greater number of users can be reached and an increase in expressiveness in modeling attained.

With the technique described in this paper, there are occasions when the user can create over-constrained situations, and although the resulting solution has the minimum error it may not be what the user expects. A more gracious solution needs to be found, perhaps one that reconfigures the lattice of control points automatically, without disturbing the previous deformations.

5 Conclusion

With direct manipulation, using FFDs for modeling complex objects becomes more intuitive. Better control over the shape and placement of the deformation is gained. By eliminating the need to display control points (and its associated control lattice) the interface is more transparent, allowing the user to concentrate on his or her work. With the proper metaphor, users no longer need to understand splines in order to use this powerful modeling tool. By adding greater control over how an object is shaped, new modeling paradigms and environments can be explored.

6 Acknowledgments

The authors of this paper would like to thank the members of the Graphics Group at Brown for their helpful comments and support, especially Daniel C. Robbins for creating the gargoyle bust. This paper is based on the Master's thesis of the first author, whose attendance at Brown University was made possible by Digital Equipment Corporation's Graduate Engineering Education Program. Special thanks to Richard

[6] Play-Doh is a registered trademark of Tonka Corporation. It is a soft modeling compound similar to clay.

Szeliski for his review and suggestions of this work, and to the other members of the Visualization group at DEC's Cambridge Research Lab for their review of this paper. This work was supported in part by grants from Digital Equipment Corporation, NSF, DARPA, IBM, NCR, Sun Microsystems, and HP.

References

[1] Allen, Jeff B., Wyvil, Brian, and Witten, Ian H. A Method for Direct Manipulation of Polygon Meshes. *Proceedings of Computer Graphics International '89*, pages 451–469, 1989.

[2] Bartels, Richard H. and Beatty, John C. A Technique for the Direct Manipulation of Spline Curves. *Proceedings of Graphics Interface '89*, June 1989.

[3] Bartels, Richard H., Beatty, John C., and Barsky, Brian A. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[4] Carroll, John M. and Thomas, John C. Metaphor and the Cognitive Representation of Computing Systems. *Trans Systems, Man, and Cybernetics*, 12(2):107–116, March/April 1982.

[5] Cobb, Elizabeth S. *Design of Sculptured Surfaces Using the B-spline Representation*. PhD thesis, University of Utah, June 1984.

[6] Coquillart, Sabine. Extended Free-Form Deformation: A Sculpting Tool for 3D Geometric Modeling. *Proceedings of SIGGRAPH '90*, In *Computer Graphics*, 24, 4, pages 187–196, August 1990.

[7] Coquillart, Sabine and Jancène, Pierre. Animated Free-Form Deformation: An Interactive Animation Technique. *Proceedings of SIGGRAPH '91*, In *Computer Graphics*, 25, 4, pages 23–26, July 1991.

[8] Csuri, Charles A. Art and Animation. *IEEE Computer Graphics and Applications*, pages 30–35, January 1991.

[9] Forsey, David R. and Bartels, Richard H. Hierarchical B-spline Refinemen. *Proceedings of SIGGRAPH '88*, In *Computer Graphics*, 22, 4, pages 205–212, August 1988.

[10] Fowler, Barry M. and Bartels, Richard H. Constraint Based Curve Manipulation. *SIGGRAPH '91 course 25 notes, Topics in the Construction, Manipulation, and Assessment of Spline Surfaces*, pages 4.0–4.16, July 1991.

[11] Hsu, William M. Direct Manipulation of Free-Form Deformations. Master's thesis, Brown University, March 1991.

[12] Noble, Ben and Daniel, James W. *Applied Linear Algebra*. Prentice-Hall, 2nd edition, 1977.

[13] Parent, Richard E. A System for Sculpting 3D Data. *Computer Graphics*, 11(2):138–147, August 1977.

[14] Riesenfield, Richard F. Design Tools for Shaping Spline Models. In Lyche and Schumaker, editors, *Mathematical Models in Computer Aided Geometric Design*, pages 499–519. Academic Press, 1989.

[15] Sederberg, Thomas W. and Parry, Scott R. Free-Form Deformation of Solid Geometric Models. *Proceedings of SIGGRAPH '86*, In *Computer Graphics*, 20, 4, pages 151–160, August 1986.

[16] Welch, William, Gleicher, Michael, and Witkin, Andrew. Manipulating Surfaces Differentially. Technical Report CS-91-175, CMU, September 1991.

Figure 3: An example of multiple constraints. The red and white object is a deformation tool which projects all points which lie within it against the red plane.

Figure 4: The deformation is created by positioning the control points according to the displacement of several of the vertices of the green object.

Figure 5: The results of the deformation at a higher resolution.

Figure 6: A ring with prongs shaped by free-form deformation.

Figure 7: A close-up of the prongs in the ring.

Figure 8: An elongated sphere is used as the foundation for a gargoyle bust. The resolution of the deformation lattice is 20x20x20.

Figure 9: An intermediate stage of the gargoyle bust.

Figure 10: The resulting gargoyle bust. The entire model, except for the eyes, was modeled using the free-form deformation modeling technique with direct manipulation.