Comput. & Graphics Vol. 14, Nos. 3/4, pp. 491–500, 1990
Printed in Great Britain.

0097-8493/90 $3.00 + .00

Technical Section

FREE-FORM SOLID MODELING USING DEFORMATIONS

UĞUR GÜDÜKBAY and BÜLENT ÖZGÜÇ
Bilkent University, Department of Computer Engineering and Information Sciences, Ankara, Turkey

Abstract—One of the most important problems of available solid modeling systems is that the range of shapes generated is limited. It is not easy to model objects with free-form surfaces in a conventional solid modeling system. Such objects can be defined arbitrarily, but then operations on them are not transparent and complications occur. A method for achieving free-form effect is to define regular objects or surfaces, then deform them. This keeps various properties of the model intact while achieving the required visual appearance. This paper discusses a number of geometric modeling techniques with deformations applied to them in attempts to combine various approaches developed so far.

1. INTRODUCTION

A system for deforming three-dimensional models to obtain objects with free-form surfaces is explained in this paper. The system is implemented using C language[8] on a Unix† workstation environment. In the implementation of the system, care has been taken to include user interface facilities to simplify the usage[7].

In advanced CAD/CAM applications, designers need to model solid objects with complex surfaces[9]. Objects whose surfaces are free-form defy description in terms of analytical surfaces such as planes, cones, spheres, or toroids[4]. There are various approaches to model objects with free-form surfaces in a solid modeling system.

One approach uses Boolean operations on arbitrary free-form surfaces. To implement Boolean operations, the computation of intersecting curves between two different free-form surfaces is required. This takes a long computation time since intersection algorithms compute points iteratively and perform some type of curve fitting that yields an approximate intersection curve. Because of this, the interpolating curve will never lie exactly on both surfaces. Consequently, intersection algorithms are unreliable.

The second approach involves generating free-form surfaces from a polyhedron. This approach uses rounding operations on polyhedral objects for integrating solid modeling and free-form surface modeling[5]. Since complex calculations are not needed, computation time and reliability are not problems. However, there are several restrictions in the range of shapes generated.

Another approach to model free-form surfaces is based on parametric polynomial functions[6]. This is a unified approach, and geometric operations can be performed with equal facility on simple primitives and complex sculptured geometries by using it. This approach combines a number of parametric polynomial geometry representations, such as Bézier, Coons, B-spline into a unified modeling system that is capable of interchanging between these representations through mathematical transformations.

A new approach, deformations, is similar to the second approach in the sense that both provide methods of changing the existing models to create irregular classes of objects. Deformations, first introduced by Alan Barr[2], are a highly intuitive and easily visualized set of operations. Deformations allow the user to treat a solid as if it were constructed from a special type of topological putty or clay, which may be bent, twisted, tapered, compressed, expanded, and otherwise transformed into a final shape. Deformations can be incorporated into traditional CAD/CAM solid modeling and surface patch methods, reducing the data storage requirements for simulating flexible geometric objects, such as objects made of metal, fabric, or rubber. Without deformations, to simulate an irregular object, one has to save every point on the object. However, one can create a regular object and then apply deformations to it to create an irregular object with much less data.

The system currently uses superquadric objects and Bézier surfaces to model regular classes of objects. There are two main approaches used to deform solid geometric models:

- Regular deformations[2];
- Free-form deformations (FFD) technique[10].

Our system combines these two approaches for deforming regular classes of objects to create objects with free-form surfaces. Both deformation techniques can be applied hierarchically and interchangeably in our system. The combination seeks to offer benefits of both regular deformations and FFD technique.

2. MODELING IN OUR SYSTEM

In the implementation of the system, Superquadrics and Bézier Surfaces are used to model regular classes of objects. They are explained briefly with some examples in the following sections for completeness.

2.1. *Superquadrics*

One long-term goal of computer graphics and numerical methods for three-dimensional design is a unified mathematical formalism. Such a unified mathematical formalism for geometric representation and computation provides a natural base for a geometric modeler of considerable versatility and robustness[6].

† Unix is a trademark of AT&T Laboratories.

Superquadric objects show potential to achieve this goal[1]. The superquadric objects are a new collection of smooth parametric objects producing a new spectrum of flexible forms. The chief advantage of superquadrics is that they allow complex solids and surfaces to be constructed and altered easily by changing a few interactive parameters. The superquadrics family consists of mainly superquadric ellipsoids, toroids, hyperboloids of one piece, and hyperboloids of two pieces. These shapes differ from the corresponding quadrics in the exponent of their terms. The exponents of their terms, which are two for the quadric shapes, are replaced by an arbitrary positive number. By changing the exponent of the terms, the shapes can be rounded, pinched, and can have different properties in different sections.

Superquadrics can be defined by either nonparametric or parametric equations. Our implementation uses parametric equations to generate superquadrics since generation of surface points is easier with this method.

The mathematics used to define superquadrics can be summarized as follows[1]. Given are two two-dimensional curves

$$\underline{h}(\omega) = \begin{bmatrix} h_1(\omega) \\ h_2(\omega) \end{bmatrix}, \quad \omega_0 \le \omega \le \omega_1,$$

and

$$\underline{m}(\eta) = \begin{bmatrix} m_1(\eta) \\ m_2(\eta) \end{bmatrix}, \quad \eta_0 \le \eta \le \eta_1.$$

The spherical product $\underline{x} = \underline{m} \otimes \underline{h}$ of the two curves is a surface defined as

$$\underline{x}(\eta, \omega) = \begin{bmatrix} m_1(\eta)h_1(\omega) \\ m_1(\eta)h_2(\omega) \\ m_2(\eta) \end{bmatrix}, \quad \begin{matrix} \omega_0 \le \omega \le \omega_1 \\ \eta_0 \le \eta \le \eta_1 \end{matrix}.$$

Geometrically, $\underline{h}(\omega)$ is a horizontal curve vertically modulated by $\underline{m}(\eta)$; $m_1(\eta)$ changes the relative scale of $\underline{h}$, while $m_2(\eta)$ raises and lowers it. η is a north-south parameter, like latitude, whereas ω is an east-west parameter, like longitude. Spherical product surfaces can be rescaled by a separate vector $\underline{a} = [a_1, a_2, a_3]^T$ where T denotes the transpose.

Position vectors for different types of superquadric objects follow[1].

2.1.1. *Position vector of surface for superellipsoids:*

$$\underline{x}(\eta, \omega) = \begin{bmatrix} a_1 C_\eta^{\epsilon_1} C_\omega^{\epsilon_2} \\ a_2 C_\eta^{\epsilon_1} S_\omega^{\epsilon_2} \\ a_3 S_\eta^{\epsilon_1} \end{bmatrix}, \quad \begin{matrix} -\frac{\pi}{2} \le \eta \le \frac{\pi}{2} \\ -\pi \le \omega \le \pi \end{matrix}$$

where $C_\eta = \cos(\eta)$, and $S_\eta = \sin(\eta)$.

2.1.2. *Position vector of surface for superhyperboloids of one piece:*

$$\underline{x}(\eta, \omega) = \begin{bmatrix} a_1 \sec^{\epsilon_1}\eta C_\omega^{\epsilon_2} \\ a_2 \sec^{\epsilon_1}\eta S_\omega^{\epsilon_2} \\ a_3 \tan^{\epsilon_1}\eta \end{bmatrix}, \quad \begin{matrix} -\frac{\pi}{2} < \eta < \frac{\pi}{2} \\ -\pi \le \omega < \pi \end{matrix}.$$

2.1.3. *Position vector of surface for superhyperboloids of two pieces:*

$$\underline{x}(\eta, \omega) = \begin{bmatrix} a_1 \sec^{\epsilon_1}\eta \sec^{\epsilon_2}\omega \\ a_2 \sec^{\epsilon_1}\eta \tan^{\epsilon_2}\omega \\ a_3 \tan^{\epsilon_1}\eta \end{bmatrix}, \quad \begin{matrix} -\frac{\pi}{2} < \eta < \frac{\pi}{2} \\ -\frac{\pi}{2} < \omega < \frac{\pi}{2} \quad \text{(piece 1)} \\ \frac{\pi}{2} < \omega < \frac{3\pi}{2} \quad \text{(piece 2)} \end{matrix}.$$

2.1.4. *Position vector of surface for supertoroids:*

$$\underline{x}(\eta, \omega) = \begin{bmatrix} a_1(a_4 + C_\eta^{\epsilon_1})C_\omega^{\epsilon_2} \\ a_2(a_4 + C_\eta^{\epsilon_1})S_\omega^{\epsilon_2} \\ a_3 S_\eta^{\epsilon_1} \end{bmatrix}, \quad \begin{matrix} -\pi \le \eta \le \pi \\ -\pi \le \omega \le \pi \end{matrix}.$$

Wireframe examples of superquadric objects from our system are shown in Fig. 1, and superquadric ellipsoids with different exponents are shown in Fig. 2.

2.2. *Bézier surfaces*

For an arbitrary curve, it may be difficult to devise a single set of parametric equations that completely defines the shape of the curve. However, any curve can be approximated by using different sets of parametric functions over different parts of the curve. Since finite degree polynomials are, in many respects, ideal forms for representing and approximating functions, they are used to form these approximations. The smoothness of a curve from one section to another can be described in terms of curve continuity between sections[3].

To define a curve or surface in design applications, a set of control points indicating the shape of the curve or surface is interactively specified. Bézier formulated a method for displaying curves specified with control points using the Bernstein polynomial basis. A useful feature of Bernstein basis is its convex-hull property, which means that any curve defined using it smoothly follows the control points without erratic oscillations.

Formulation of Bézier Surfaces can be found in [3]. Fig. 3 shows two Bézier Surfaces generated by our system.

3. DEFORMATIONS

Since the primary goal of our system is to create the free-form objects and scenes that the user desires, we have to supply the user with the operations that can be used to achieve this goal. We have implemented deformations for this purpose. Two deformation tech-

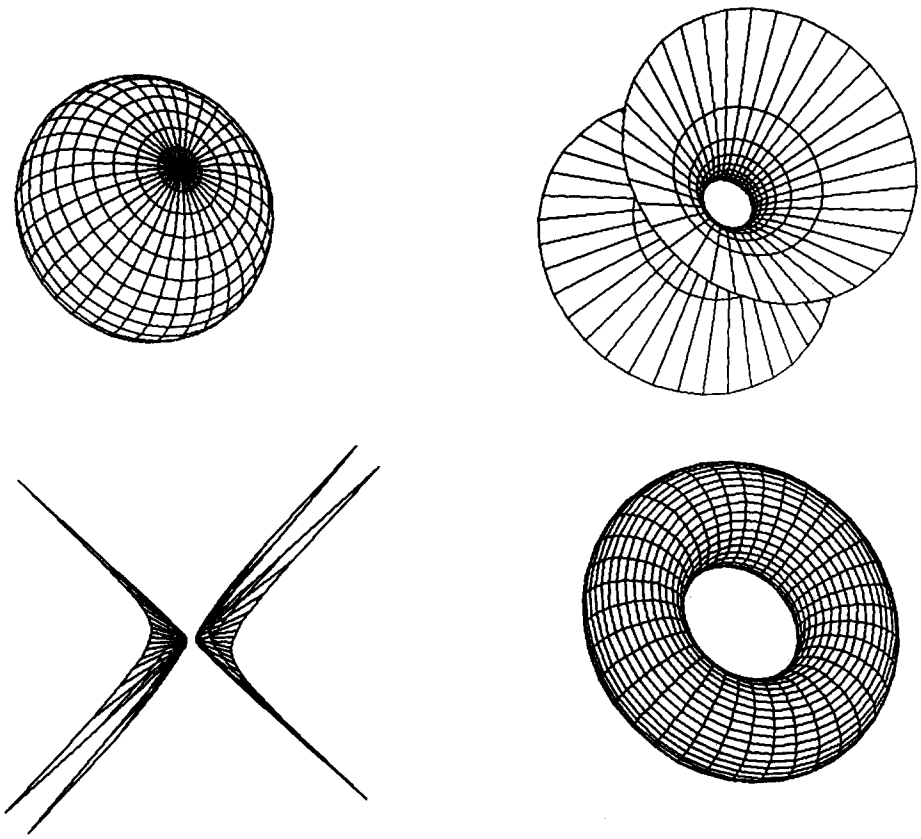

Fig. 1. Superquadric ellipsoid, hyperboloid of one piece, hyperboloid of two pieces, and toroid.

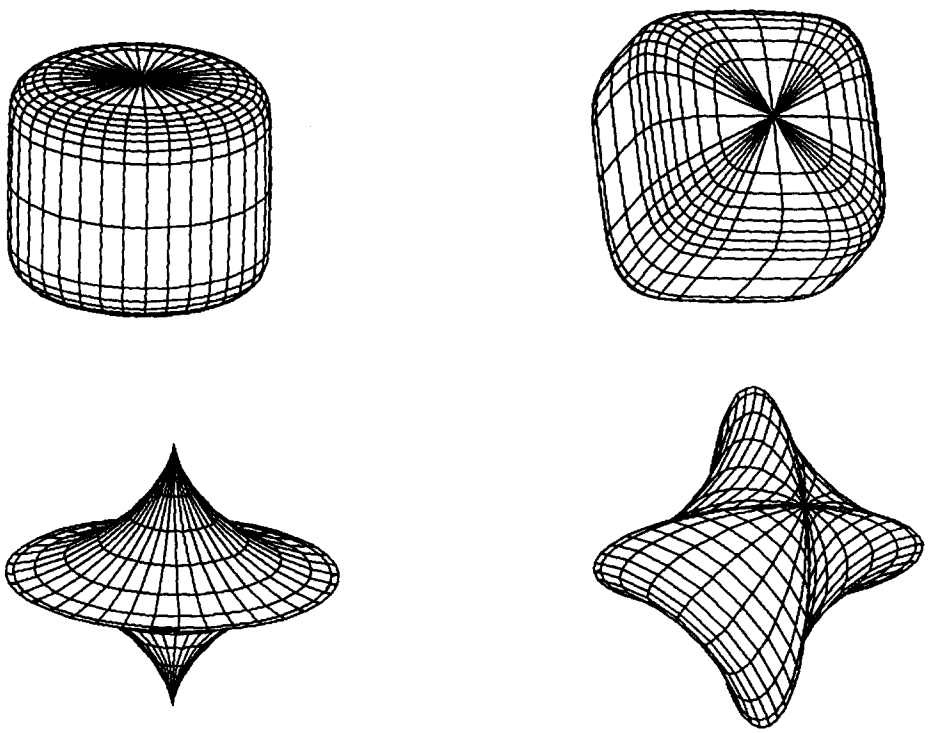

Fig. 2. Superquadric ellipsoids with different exponents.

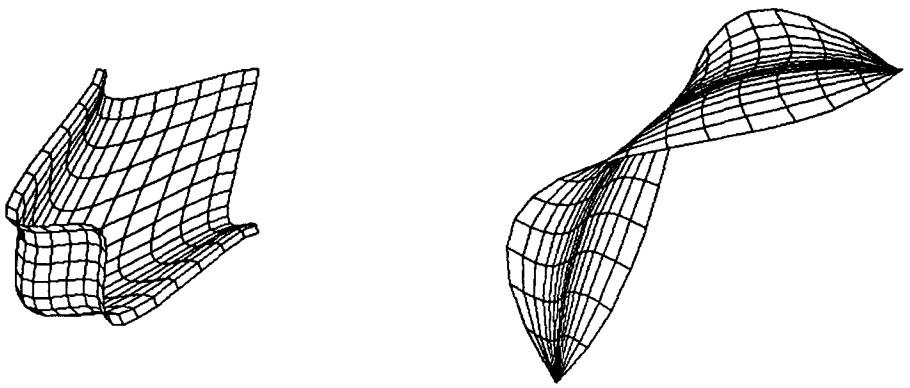

Fig. 3. Two Bézier surfaces.

niques are used in the implementation. Regular deformations[2] simulate twisting, bending, tapering, or similar transformations of geometric objects. Free-form deformation technique[10] is mainly developed to define the solid geometric model of an object bounded by free-form surfaces, and to sculpt it further with flexibility and freedom. Both of these techniques have advantages and disadvantages.

Although results of FFD can be guessed according to the movement of control points, desired objects can be obtained by trial and error. However, regular deformations are well defined and their results are straightforward. On the other hand, using FFD as a free-form modeling technique is better than using regular deformations due to the generality of it.

A very important disadvantage of FFD is the speed of the deformations since operations on trivariate Bernstein polynomials are very costly. It can be made faster by converting trivariate Bernstein polynomials to standard power basis polynomials. However, this operation also takes a fair amount of time.

The speed of deforming an object with FFD technique also depends on the number of control points. A deformation defined by larger number of control points can cause the deformed shape to follow the control points more closely than for lower degree deformations[11]. This produces better results, but at the expense of slowing down the operations. Regular deformations are very fast compared to FFD technique.

These two approaches have some common properties.

- Both approaches can be applied hierarchically to create complex objects from simpler ones.
- Both approaches can be applied to any solid modeling scheme.
- Both approaches compute the new x, y, z coordinates of a point as polynomial functions of the original x, y, z coordinates of that point.

Another important issue is that regular deformations can also be performed using FFD since it is a method for deforming three-dimensional objects in a free-form manner. However, using FFD to make twisting, bending, and tapering operations brings some undesired effects. For example, when an object is bent using FFD technique, the result will not be as predictable as when the same object is bent using regular deformations.

Due to the reasons stated above, our system combines the two approaches to alleviate the problems of them, and to offer the benefits to the user. When regular deformations are suitable for modeling an object, he or she may use them to gain in speed. When they are not sufficient for modeling an object, either FFD technique can be used, or both of the techniques can be applied hierarchically and interchangeably. The two deformation techniques are explained briefly with some examples in the following sections for completeness.

3.1. *Regular deformations*

A globally specified deformation of a three-dimensional solid is a mathematical function $\underline{F}$ which explicitly modifies the global coordinates of points in space. Mathematically, it can be represented by the equation $\underline{X} = \underline{F}(\underline{x})$ where $\underline{x}$ represents the point in the undeformed solid, and $\underline{X}$ represents the points in the deformed solid[2]. Each of the regular deformations are explained in detail in the following sections.

3.1.1. *Tapering.* Tapering is making an object become gradually narrower toward one end of it. Mathematically, it differentially changes the length of the two global components without changing the length of the third.

To do a tapering operation along the z-axis, one should choose a tapering function depending on the z-coordinates of the points. When the tapering function $f(z) = 1$, the portion of the deformed object is unchanged; the object increases in size as a function of z when $f'(z) > 0$ and decreases in size when $f'(z) < 0$. The object passes through a singularity at $f(z) = 0$ and becomes everted when $f(z) < 0$. Global tapering along the z-axis is given by the following equations:

$$r = f(z)$$
$$X = rx$$
$$Y = ry$$
$$Z = z.$$

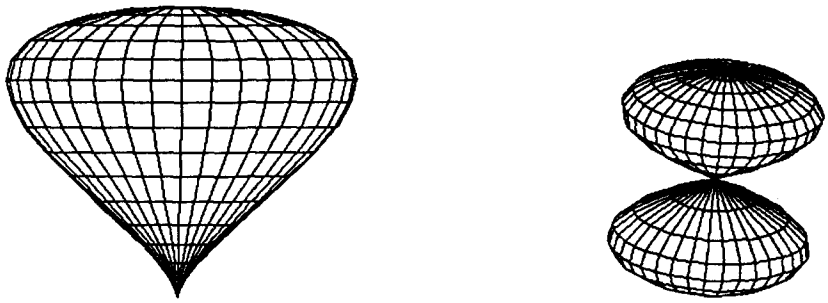

Fig. 4. Tapered superquadric ellipsoids.

Examples of tapering are shown in Fig. 4 where a superellipse is tapered using different tapering functions. The first object is obtained using the tapering function $f(z) = z\frac{\pi}{180}$ and the second one is obtained using the tapering function $f(z) = \cos\left(z\frac{\pi}{180}\right)$.

3.1.2. *Twisting.* A twist operation can be approximated as differential rotation, just as tapering is a differential scaling of the global basis vectors. We rotate one pair of global basis vectors as a function of height, without altering the third global basis vector. An example of twisting operation is the twisting of a deck of cards, by which each card is rotated somewhat more than the card beneath it. Twisting operation preserves the volume of the original solid.

To do a twisting operation along the z-axis, twisting angle θ should be a function of the z-coordinate of the point to be deformed. Global twisting along the z-axis is given by the following equations:

$$\theta = f(z)$$

$$C_\theta = \cos(\theta)$$

$$S_\theta = \sin(\theta)$$

$$X = xC_\theta - yS_\theta$$

$$Y = xS_\theta + yC_\theta$$

$$Z = z.$$

The twist proceeds along the z-axis at a rate of $f'(z)$ radians per unit length in the z direction.

Examples of twisting are shown in Fig. 5 where a superellipse is twisted using different twisting functions. The first object is obtained using the twisting function $\theta = z\frac{\pi}{180}$ and the second one is obtained using the twisting function $\theta = \sin\left(z\frac{\pi}{180}\right)$.

3.1.3. *Bending.* Bending simulates an important manufacturing process for fabricating objects. An example of this operation is the bending of a bar stock or sheet metal.

To make a bending along the y-axis, one has to specify a bent region along the y-axis. The range of the bending deformation is controlled by $y_{\min}$, and $y_{\max}$, with the bent region corresponding to values of y such that $y_{\min} \le y \le y_{\max}$. The axis of the bend is located along $\left[s, y_0, \frac{1}{k}\right]^T$, where s is the parameter of the line. The center of the bend occurs at $y = y_0$. The radius of curvature of the bend is $\frac{1}{k}$. The bending angle θ is constant outside the bent region, changes linearly in the central region. In the bent region, bending rate k, measured in radians per unit length, is constant. Outside the bent region, the deformation consists of a rigid body rotation and translation. The length of the centerline passing through the object along the y-axis does not change during the bending process.

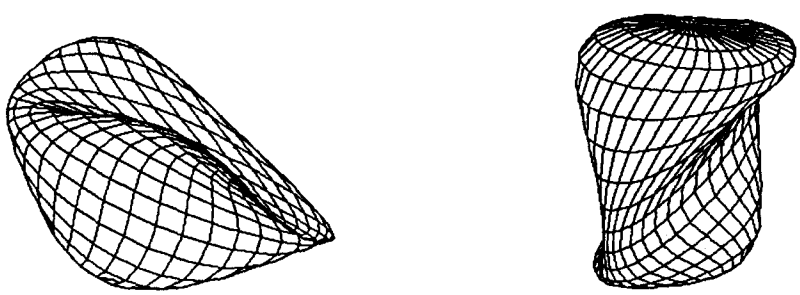

Fig. 5. Twisted superquadric ellipsoids.

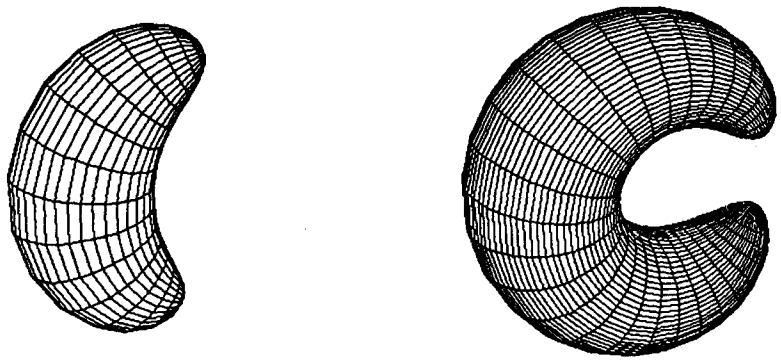

Fig. 6. Bent superquadric ellipsoids.

The bending angle θ is given by:

$$\theta = k(\hat{y} - y_0)$$

$$C_\theta = \cos(\theta)$$

$$S_\theta = \sin(\theta)$$

where

$$\hat{y} = \begin{cases} y_{\min}, & y \le y_{\min} \\ y, & y_{\min} < y < y_{\max} \\ y_{\max}, & y \ge y_{\max}. \end{cases}$$

The formula for bending along the y-axis centerline is given by the following equations:

$$X = x$$

$$Y = \begin{cases} -S_\theta\left(z - \frac{1}{k}\right) + y_0, & y_{\min} \le y \le y_{\max} \\ -S_\theta\left(z - \frac{1}{k}\right) + y_0 + C_\theta(y - y_{\min}), & y < y_{\min} \\ -S_\theta\left(z - \frac{1}{k}\right) + y_0 + C_\theta(y - y_{\max}), & y > y_{\max} \end{cases}$$

$$Z = \begin{cases} C_\theta\left(z - \frac{1}{k}\right) + \frac{1}{k}, & y_{\min} \le y \le y_{\max} \\ C_\theta\left(z - \frac{1}{k}\right) + \frac{1}{k} + S_\theta(y - y_{\min}), & y < y_{\min} \\ C_\theta\left(z - \frac{1}{k}\right) + \frac{1}{k} + S_\theta(y - y_{\max}), & y > y_{\max}. \end{cases}$$

Examples of bending operation are shown in Fig. 6. In these examples, the bent region includes the whole object. In the first figure, an ellipse is bent 90°, and in the second one an ellipse is bent 360°.

Regular deformations explained above can be combined with rotation around some axes so that these operations can be performed around other axes than the ones specified. The mathematical details of the regular deformations can be found in [2]. Results obtained by applying regular deformations hierarchically are shown in Fig. 7.

3.2. *Free-form deformations*

Free-form deformation can be thought of as a method for sculpturing solid models in a free-form manner. FFD can sculpt solids bounded by any analytical surface; planes, quadrics, parametric surface patches, or implicit surfaces. Its application is not re-

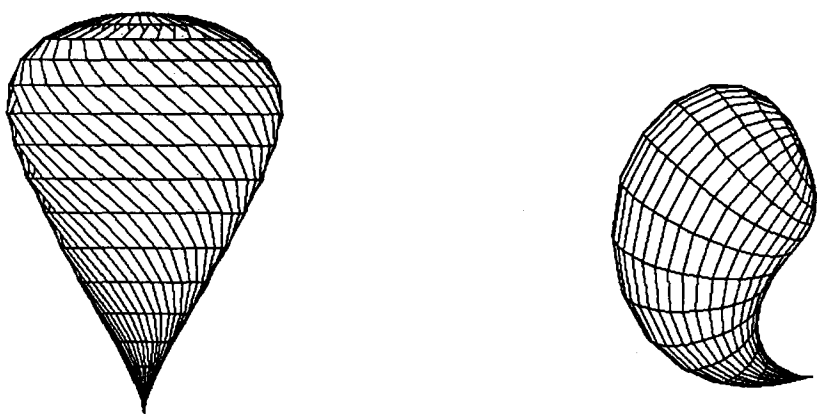

Fig. 7. A twisted, tapered superquadric ellipsoid, and a tapered, bent superquadric ellipsoid.

stricted to solid models, but it can also sculpt surfaces or polygonal data.

In our system, two kinds of parametric surface patches, namely Bézier surfaces and superquadrics, are deformed in a free-form manner using FFD techniques. In fact, the objects are approximated using small polygons. Thus, the deformed data are actually polygonal data.

The free-form deformation is initiated by defining a three-dimensional grid of control points about the region to be deformed. The objects to be deformed are embedded in the grid of control points that can be interactively deformed as if it is made of a flexible material. The objects themselves can also be regarded as if they are made of a flexible material, so that when the whole grid is deformed, the objects inside them are also deformed respectively.

The deformation function for an FFD operation is defined by a trivariate tensor product Bernstein polynomial as specified in [10]. Since it is very time consuming to evaluate a trivariate Bernstein polynomial for lots of polygon vertices that are to be transformed, it is wise to convert the Bernstein polynomial to standard power polynomial basis, which can then be evaluated using Horner's method.

Steps of deforming an object using FFD technique can be summarized as follows:

1. Move control points from their undisplaced, latticial positions to their new positions.
2. Convert the Bernstein polynomial whose coefficients are the displaced control points to a standard power basis polynomial.
3. Calculate free-form deformation position of each point on the object by evaluating standard power basis polynomial found in step 2.

Our system uses the algorithms in [11] to deform objects in a free-form manner. The mathematical details of FFD technique can be found in [10]. Examples of FFD technique are shown in Fig. 8.

3.3. *Combination of regular deformations and FFD technique*

We have applied the two deformation techniques hierarchically to regular classes of objects by using our implementation. Results are shown in Fig. 9. The first object is obtained by tapering a superquadric hyperboloid of one piece and applying an FFD to the tapered object. The second object is also obtained in the same way, but the initial object is an ellipse. The third object is obtained by applying an FFD to a superquadric toroid to compress it and bending the compressed object.

4. THE USER INTERFACE

Since our system is interactive, we have to provide the user some facilities for defining parameters in creating objects and for defining deformation parameters so that the user can use it in an efficient manner. The facilities provided by the system are explained in detail in the following sections.

4.1. *Facilities for creating Bézier surfaces and superquadrics*

In our system, the user can create the desired objects with the help of the object menu (Fig. 11). The user can either select Bézier surfaces or one of the superquadrics to create a three-dimensional object.

To create Bézier surfaces, number of control points and number of curve points on each curve of the surface can be specified by the user, and control points for a Bézier surface can be interactively entered with the help of a mouse.

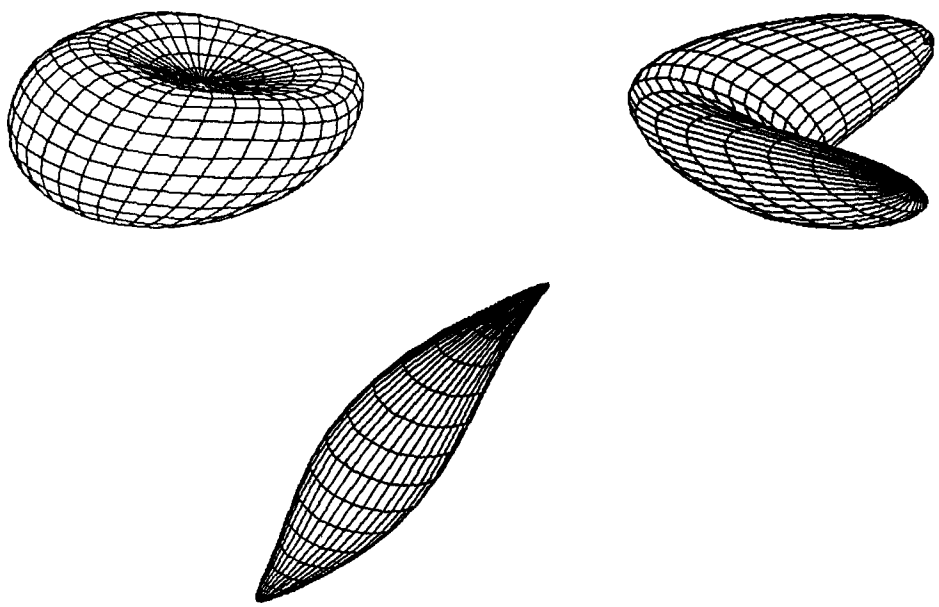

Fig. 8. An ellipse deformed using different FFDs.

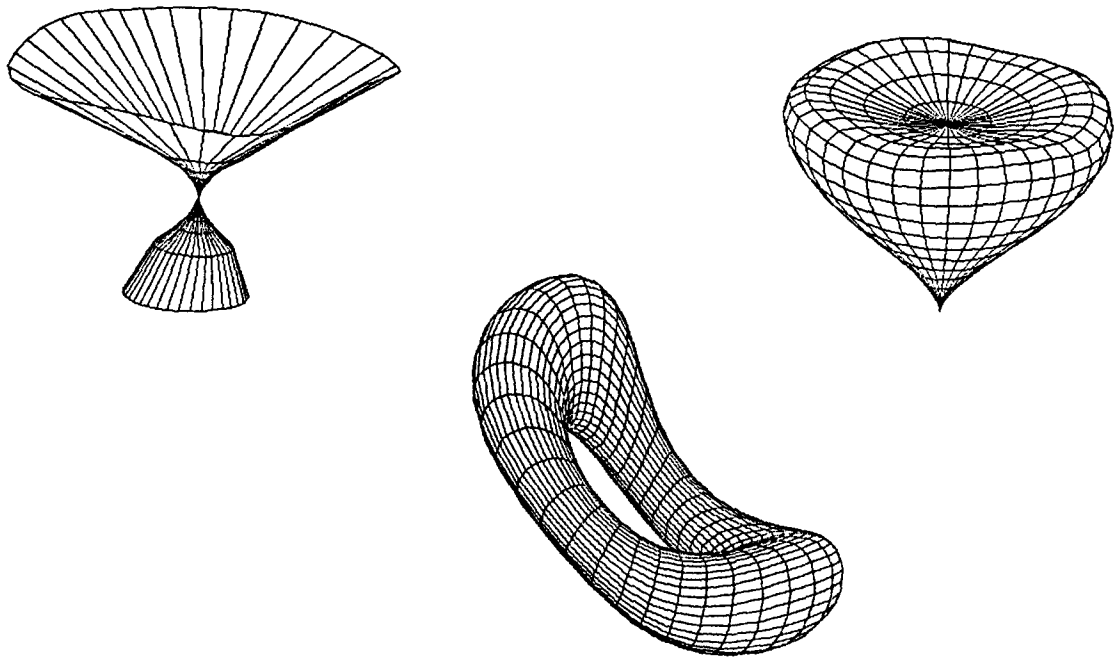

Fig. 9. Results of applying combination of regular deformations and FFDs.

To create superquadrics, scaling factors and exponents are interactively input from the user and the user is presented with a rough sketch, namely the bounding box, for the object that will be created. Then the user may want the system to create the object specified by the parameters or discard it and specify different values for the parameters. The layout of the screen while parameters of superquadrics are being specified is shown

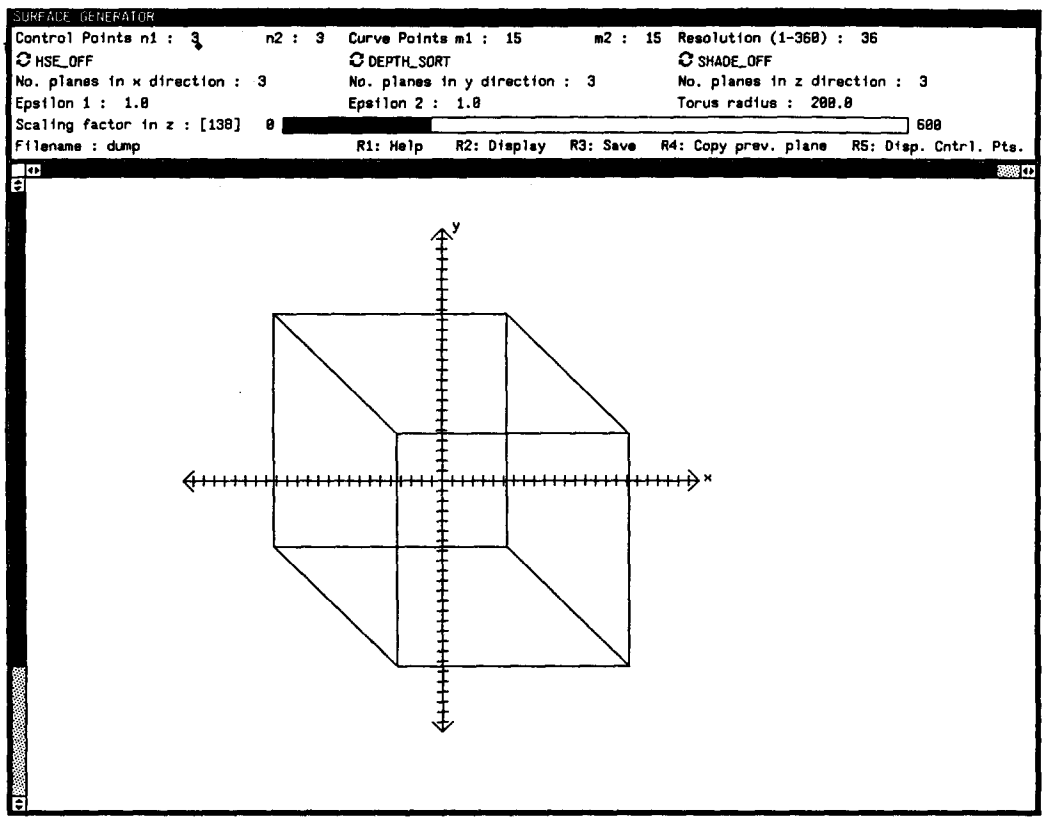

Fig. 10. The layout of the screen while parameters of superquadrics are given.

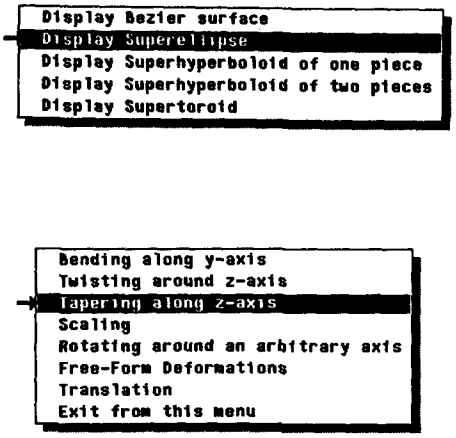

Fig. 11. The object menu and the operations menu of the system.

in Fig. 10. The rectangular prism gives an idea to the user about the size of the object. Resolution of the superquadric objects can also be specified. In order to obtain very good polygonal approximations of superquadric objects, a high resolution may be specified. However, using a high resolution will produce better results at the expense of slowing down the operations.

4.2. *Facilities for deforming objects*

The implementation also provides facilities for deforming the created objects. The operations that will be performed on the objects can be selected using the operations menu (Fig. 11). Different twisting and tapering functions can be selected using menus. The user may select currently available twisting and tapering functions for these operations or select the option for specifying a new twisting or tapering function. If the user selects this option, a new window appears on the screen in which he or she can enter the function for twisting or tapering operation in infix notation using the keyboard. Also, bending region and center of bend can be specified interactively for bending operation.

To do an FFD, the user is presented with a regular lattice of control points. The number of planes in x, y, and z directions can be specified interactively. In this way, the user may select between high quality and speed of the operations. If the number of control points in each direction is high, the deformation specified will be better, but operations will be slower. On the other hand, if the number of control points in each direction is low, operations will be faster, but deformations will not be high quality.

The user may take each plane parallel to the xy-plane and change the coordinates of points interactively with the help of a mouse. He or she may see the lattice of control points any time during that process. After deforming lattice of control points, the user can get the deformed object according to the specified lattice. Fig. 12 shows a lattice of control points generated by our system.

Display facilities such as shading and hidden-surface elimination are also provided by the system so that

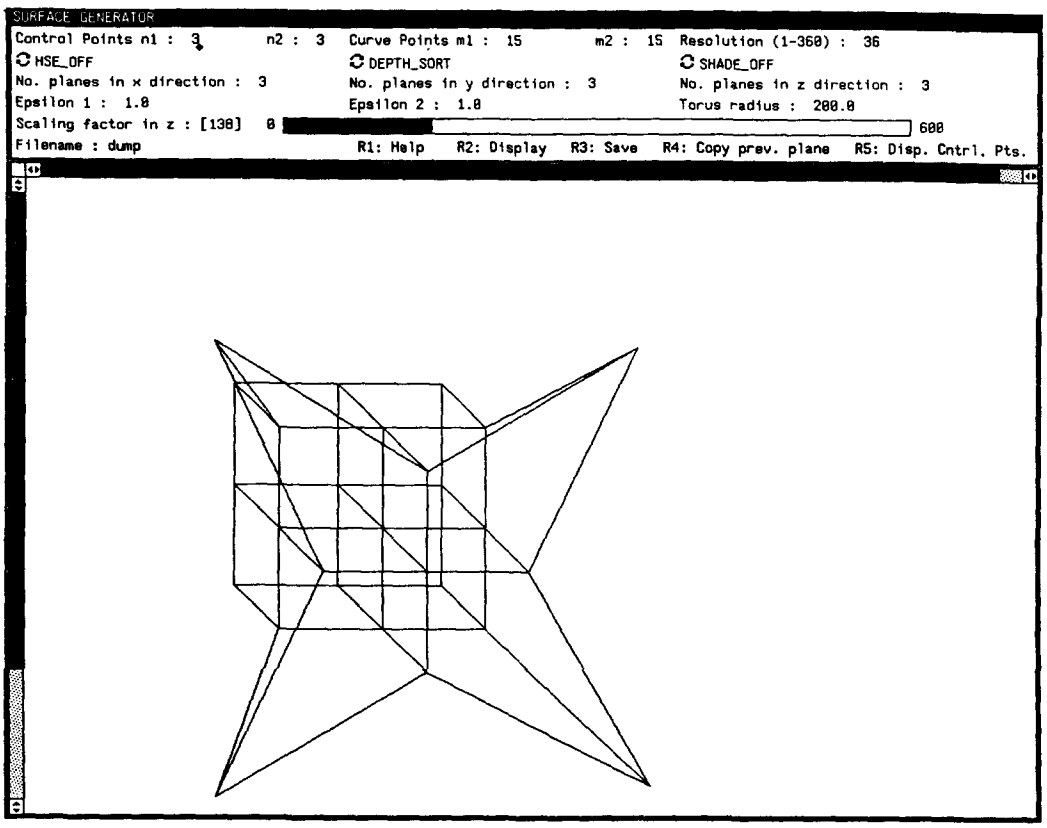

Fig. 12. A lattice of control points created and deformed by our system.

they can be used to obtain realistic scenes. The implementation uses the facilities provided by Sun View† system such as windows, panels, and menus[12].

5. CONCLUSIONS

A system is developed to create the objects and scenes that the user desires through the use of a set of primitive objects and a set of operations to deform these primitives. Two different deformation techniques are used in the implementation of the system. Since both deformation techniques have advantages and disadvantages, we have combined them.

Our system can be used to model objects with free-form surfaces or to sculpt objects. Both of the deformation techniques can be applied hierarchically and interchangeably through a set of user interface facilities provided by the implementation. By combining the two approaches, some of the problems peculiar to each method disappear and the advantages of both approaches are utilized.

REFERENCES

1. A. H. Barr, Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications* **1**(1), 11–23 (January 1981).
2. A. H. Barr, Global and local deformations of solid primitives. *Computer Graphics* **18**(3), 21–30 (July 1984).
3. P. Bézier, *Numerical Control—Mathematics and Applications* John Wiley and Sons, London (1972).
4. M. S. Casale, Free-form solid modeling with trimmed surface patches. *IEEE Computer Graphics and Applications* **7**(1), 33–43 (January 1987).
5. H. Chiyokura, An extended rounding operation for modeling solids with free-form surfaces. *IEEE Computer Graphics and Applications* **4**(7), 27–35 (July 1984).
6. R. T. Farouki and J. K. Hinds, A hierarchy of geometric forms. *IEEE Computer Graphics and Applications* **5**(5), 51–78 (May 1985).
7. U. Güdükbay and B. Özgüç, Deformation of solid models. *Proceedings of the Fourth International Symposium on Computer and Information Sciences,* Çeşme, Turkey, 525–534 (1989, October).
8. B. W. Kernighan and D. M. Ritchie, *The C Programming Language* Prentice Hall, Inc., Englewood Cliffs, NJ (1978).
9. F. Kimura, Geomap—III. Designing solids with free-form surfaces. *IEEE Computer Graphics and Applications* **4**(6), 58–72 (June 1984).
10. T. W. Sederberg and S. R. Parry, Free-form deformation of solid geometric models. *ACM Computer Graphics (Proc. SIGGRAPH)* **20**(4), 151–160 (August 1986).
11. T. W. Sederberg and S. R. Parry, Free-form deformation of polygonal data. In *Proceedings of the International Electronic Image Week,* Nice, France, 633–639 (April 1986).
12. Sun Microsystems, *Sun View Programmer's Guide,* Mountain View, CA (1986).

† Sun View is a registered trademark of Sun Microsystems.