FREE-FORM SOLID MODELING USING DEFORMATIONS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND

INFORMATION SCIENCES

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Uğur Güdükbay

June 1989

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Bülent ÖZGÜÇ (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Mehmet Baray

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Varol Akman

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Sciences

ABSTRACT

FREE-FORM SOLID MODELING USING DEFORMATIONS

Uğur Güdükbay

M.S. in Computer Engineering and
Information Sciences
Supervisor: Prof. Dr. Bülent ÖZGÜÇ
June 1989

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 thesis explains a number of geometric modeling techniques with deformations applied to them in attempts to combine various approaches developed so far. Regular deformations, which include twisting, bending, and tapering, and free-form deformation technique are combined as a new deformation method. This eliminates some of the disadvantages peculiar to each method and utilizes the advantages of both.

Keywords: Deformations, geometric modeling of solids, free-form surfaces, user interface design, shading, hidden surface elimination, computer graphics.

ÖZET

DEFORMASYON TEKNİKLERİ KULLANILARAK DÜZENSİZ NESNELERİN MODELLENMESİ

Uğur Güdükbay
Bilgisayar Mühendisliği ve Enformatik Bilimleri Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Bülent ÖZGÜÇ
Haziran 1989

Bugüne kadar yapılmış olan katı modelleme sistemlerinin en önemli sorunlarından birisi de üretilebilen şekillerin kısıtlı olmasıdır. Alışılagelmiş bir katı modelleme sisteminde düzensiz yüzeyleri olan nesnelerin modellenmesi kolay değildir. Böyle nesneler, üzerindeki her nokta verilerek tanımlanabilir, fakat bu yöntem kullanıldığında bu nesneler üzerindeki işlemler belirgin olmamakta ve bazı zorluklar ortaya çıkmaktadır. Bu nesnelerin modellenmesinde etkili bir yöntem de düzenli şekilleri oluşturduktan sonra onlar üzerinde deformasyon tekniklerini uygulamaktır. Bu yolla gerekli nesneler elde edilirken ortaya çıkan zorluklar da önlenmekte ve işlemlerde açıklık sağlanmaktadır. Bu araştırma bazı modelleme yöntemlerine deformasyon tekniklerinin uygulanması ve deformasyon tekniklerinde bugüne kadar kullanılan değişik yaklaşımların birleştirilmesi konuları ile ilgilidir. Kıvırma, bükme ve inceltme gibi düzenli deformasyonlar serbest deformasyon tekniği ile birleştirilerek yeni bir deformasyon yöntemi elde edilmiştir. Böylece bu yöntemlere özgü bazı kısıtlamalar yokedilmiş ve her iki yöntemin yeteneklerinden daha etkin bir şekilde yararlanılmıştır.

Anahtar Kelimeler: Deformasyon, katıların geometrik modellenmesi, düzensiz nesneler, kullanıcı arabirimleri, tarama, görünmeyen yüzeyleri yoketme, bilgisayar grafiği.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Prof. Bülent Özgüç, for his guidance and support during the development of this study.

I appreciate my colleagues, Aydın Açıkgöz, Veysi İşler, Aydın Kaya, Ahmet Coşar, Faruk Polat, İsmail Hakkı Toroslu, Cem Evrendilek, and Ufuk Çelikkan, for their valuable discussions and comments.

I am also grateful to Cemil Türün for his technical help.

Special thanks to Prof. Mehmet Baray and Asst. Prof. Varol Akman for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

A system for deforming three dimensional models to obtain objects with free-form surfaces is explained in this thesis. The system is implemented using C language [15] on a Unix [1] workstation environment. In the implementation of the system, care has been taken to include user interface facilities to simplify the usage [14].

In advanced CAD/CAM applications, designers need to model solid objects with complex surfaces [16]. Objects whose surfaces are free-form defy description in terms of analytical surfaces such as planes, cones, spheres, or toroids [8]. 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 [9]. 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 [10]. This is a unified approach and geometric operations can be performed with equal facility on simple primitives and complex

[1] Unix is a trademark of AT&T Laboratories.

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 [4], are 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.

Our 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 [4].
- Free-form deformation (FFD) technique [21].

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.

The remaining parts of this thesis are organized as follows. Chapter 2 surveys the concepts related to geometric modeling by explaining the history of geometric modeling and shows how objects are modeled in our implementation. Mathematical details of superquadrics and Bézier surfaces are also explained in this part.

Chapter 3 describes different deformation techniques and compares them by giving the advantages and disadvantages of them. It also gives the reasons why these techniques are combined to obtain a different deformation

technique and explains how deformations are implemented in our system by giving examples.

Chapter 4 gives a detailed explanation about the display facilities provided by the system (shading and hidden surface elimination) and describes the methods used for these purposes together with the reasons why they are used.

Chapter 5 contains information about the user interface issues of the implementation. The user interface facilities for creating the objects that the user desires and manipulating them through the set of operations provided by the system are explained in detail.

Appendix A presents a user's manual for those who wish to use the system.

2. MODELING

2.1 History of Modeling

The application of computers to drafting and design started with Sketchpad, which is a remarkable program devised by Ivan Sutherland at MIT in the early 60's. First, the techniques of Sketchpad were applied to circuit design where the connectivity or topological properties of the data, represented as a graph of nodes and links, are important [6].

Later, drafting systems progressed in two ways; firstly, in the amount of structure captured and in the variety of graphics entities represented, and secondly, by moving from two to three dimensions. Wireframe models have emerged as a result of these developments where perspective views and removal of hidden lines have gained importance.

Wireframe models can be defined as a collection of curve segments which represents an object's edges. Wireframe models have some serious deficiencies. These deficiencies can be listed as follows [19]:

- The wireframe may be ambiguous; it may represent more than one object.
- Nonsense objects such as a wireframe with one of the edges missing cannot be detected by the system.
- Representation of lines to depict viewpoint-dependent artifacts in the wireframe creates problems.
- Lots of low level data is stored to represent even trivial objects.

The first two deficiencies limit various automatic processes that can be done by a system using wireframe modeling. For example, calculating volumes of objects represented by a wireframe and automatic sectioning of them are almost impossible to implement automatically.

A further step in the progress of drafting systems was to allow 3-D wireframe models to be *surfaced.* The surfaces are represented by embedding of the graph formed by edges and vertices of a wireframe model. This embedded graph represents the *boundary model* of the solid.

The systems addressing the problem of designing classes of parts comes after drafting systems. These systems are dedicated to a family of special products, such as pumps. They calculate design properties of pumps, generate pictures of them, etc. These systems cover three stages of product definition, namely, design, analysis, and manufacture. Consequently, they reduce the chance of error from stage to stage. Their drawback is that they are expensive to write and develop, and they can act as a barrier to progress since they embody certain design rules which new knowledge may make obsolete.

Later, solid modelers have emerged as a new development in drafting systems. The term *solid modeling* encompasses a body of theory, techniques, and systems focused on *informationally complete* representation of solids — representations that permit (at least in principle) any well-defined geometric property of any represented solid to be calculated automatically [19].

Solid modeling is a powerful and promising concept for computerized product definition [16]. Systems used in the design of products cover not only design but also analysis and manufacture. This means that the same data are repeatedly used at design, analysis and manufacturing steps. For this reason, if the information were captured in a suitable, general form at the design time, it could be used again for analysis and for manufacture. In this way, mistakes could be avoided, and time and money saved. This requires to decide on a product representation that is suitable for all types of computations. The representation must be accurate, must contain enough information for all subsequent inquiries to be made, and should be concise. To achieve this goal, solid modelers have been developed.

2.1.1 Classification and Properties of Solid Modelers

Solid modelers can be classified into two broad categories: *regional (descriptive) modelers*, and *boundary modelers*. Regional modelers can be further classified as subdivisional modelers and cellular modelers.

One type of descriptive modeler is based on Constructive Solid Geometry (CSG). It uses trees (CSG trees) of building block primitives, such as parallelepipeds, spheres, cylinders, etc. combined by geometric transformations and Boolean set operations as a representation of three-dimensional solid objects [20].

Another type of descriptive modeler represents solid objects using half spaces, each described by an equation of the form $f(x, y, z) \geq 0$. Some of the CSG models also decompose primitive solids into subtrees whose leaves are halfspaces so that they combine both pure CSG and halfspaces [20].

Modelers using octree methods are in the group of cellular modelers. They represent solid objects by a binary tree defining recursive subdivision of space and recording which parts are empty and which parts are solid. The trees used to represent two dimensional objects are called octrees and the ones used to represent solid objects are called quadtrees [7].

Subdivisional modelers are similar to cellular modelers, but the subdivision of space into smaller parts is not necessarily regular, as in the case of cellular modelers.

Boundary (or surface-based) representation techniques describe solid volumes in terms of their enclosing surfaces. Such models can be called solid models when they completely describe the form and the extent of the individual surfaces of the objects. They must also contain enough information to determine how the surfaces are joined together to form completely closed and connected volumes [11].

Boundary representation (B-rep) techniques use some low-level operators, called Euler operators, as a convenient and consistent way of creating and modifying the topological data of the solid model. Most common Euler operators are [6]:

1. Create shell (a *shell* of an object S is defined as a maximal connected set of faces of S), face (*face*'s can be defined as contiguous surface areas of the volume enclosed by face boundaries), and vertex.

2. Add edge and vertex.

3. Add edge, face and loop (a *loop* on a face f is defined as a closed chain of edges bounding f).

4. Add genus and loop, remove face and loop.

5. Add loop, remove shell, face and loop.

Boolean operations are not included in the representation of a B-rep model, but they often are employed as one of the means of creating and manipulating the model. Since B-rep systems require an explicit representation of the boundary of the solid, they must evaluate the new boundary that is the result of the Boolean operation applied [8].

Other methods for creating and manipulating geometry in B-rep systems include *sweeping*, which defines the bounding surfaces of the solid by moving a cross-section along a path in space, and *tweaking*, which performs local operations on the geometry, but leaves the topology of the model unchanged. Tapering is an example to tweaking. Some other operations which introduce strain, shear, and torsion into models are *bending*, which changes the topology of a model in a well-defined way, *blending*, which rounds off and chamfer sharp edges, and *stretching*, which occurs in physical bending.

2.1.2 Modeling Free-Form Surfaces

Although there are a number of solid modeling systems that can be used as a basis for product modeling, solid modeling capabilities have not been fully utilized. There are several technical problems remaining, such as dimensioning and tolerancing, user interface, and speed and reliability of processing [16]. Treatment of solid objects with free-form surfaces is another major problem of current solid modeling systems. Objects with free-form surfaces cannot be described in terms of analytical functions. Some examples of objects with free-form surfaces are the fender of a sports car, the transition between the wing and fuselage of an aircraft, the hull of a destroyer, the handle of a hand-held mixer, etc. [8]. Free-form surface design capability should be integrated into solid modeling systems in order to model objects with complex surfaces. The approaches for free-form surface design (explained in Chapter 1) can be used in solid modeling systems for this purpose.

2.2 Modeling in Our System

Since our system uses regular objects to create objects with free-form surfaces, we have to provide methods to model these regular objects. In the implementation of the system, two different surface modeling methods based on parametric polynomial functions, namely superquadrics and Bézier surfaces, are used to model regular classes of objects. They are explained in detail with some examples in the following sections.

2.2.1 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 [10]. Superquadric objects show potential to achieve this goal [3]. 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 mainly consists of 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 exponent of their terms, two for the quadric shapes, is 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. The parametric form is used for calculation, since surface points and normal vectors can be generated more easily with this method than with the implicit form, but the sampled points are sometimes very unevenly spaced. The nonparametric form, which is developed using computational geometry techniques, uses explicit equations or series approximations for generating points on the curve. Advantages of the nonparametric form are [2]:

- It naturally produces equally spaced points.
- It requires much less CPU time for each point.
- It can produce surfaces of great generality.

Our implementation uses parametric equations to generate superquadrics since generation of surface points is easier with this method. Also, we need normal vectors for shading and hidden surface elimination. Calculation of normal vectors at each surface point is much easier using parametric equations. To speed up the generation of points, the values that will be needed to produce the points are calculated once and they are stored in look-up tables so that they will be retrieved when they are needed.

The mathematics used to define superquadrics can be summarized as follows [3].

Given are two two-dimensional curves

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

and

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

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

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

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.

Types of superquadric shapes and the formulation of them are explained in the following sections.

Superellipsoids

Position vector of surface:

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

Normal vector:

$$\underline{n}(\eta,\omega) = \begin{bmatrix} \frac{1}{a_1} C_\eta^{2-\epsilon_1} C_\omega^{2-\epsilon_2} \\ \frac{1}{a_2} C_\eta^{2-\epsilon_1} S_\omega^{2-\epsilon_2} \\ \frac{1}{a_3} S_\eta^{2-\epsilon_1} \end{bmatrix}$$

ϵ_1 is the squareness parameter in the north-south direction and ϵ_2 is the squareness parameter in the east-west direction. Cuboids are produced when both ϵ_1 and ϵ_2 are < 1. Pillow shapes are produced when $\epsilon_1 \approx 1$ and $\epsilon_2 < 1$. Pinched shapes are produced when either ϵ_1 or $\epsilon_2 > 2$. Flat-beveled shapes are produced when either ϵ_1 or $\epsilon_2 = 2$.

Wireframe examples of superquadric ellipsoids with different exponents are shown in Figure 2.1.

Superhyperboloids of one piece

Position vector of surface:

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

Normal vector:

$$\underline{n}(\eta,\omega) = \begin{bmatrix} \frac{1}{a_1} sec^{2-\epsilon_1}\eta C_\omega^{2-\epsilon_2} \\ \frac{1}{a_2} sec^{2-\epsilon_1}\eta S_\omega^{2-\epsilon_2} \\ \frac{1}{a_3} tan^{2-\epsilon_1}\eta \end{bmatrix}$$

Wireframe examples of superquadric hyperboloids of one piece with different exponents are shown in Figure 2.2.

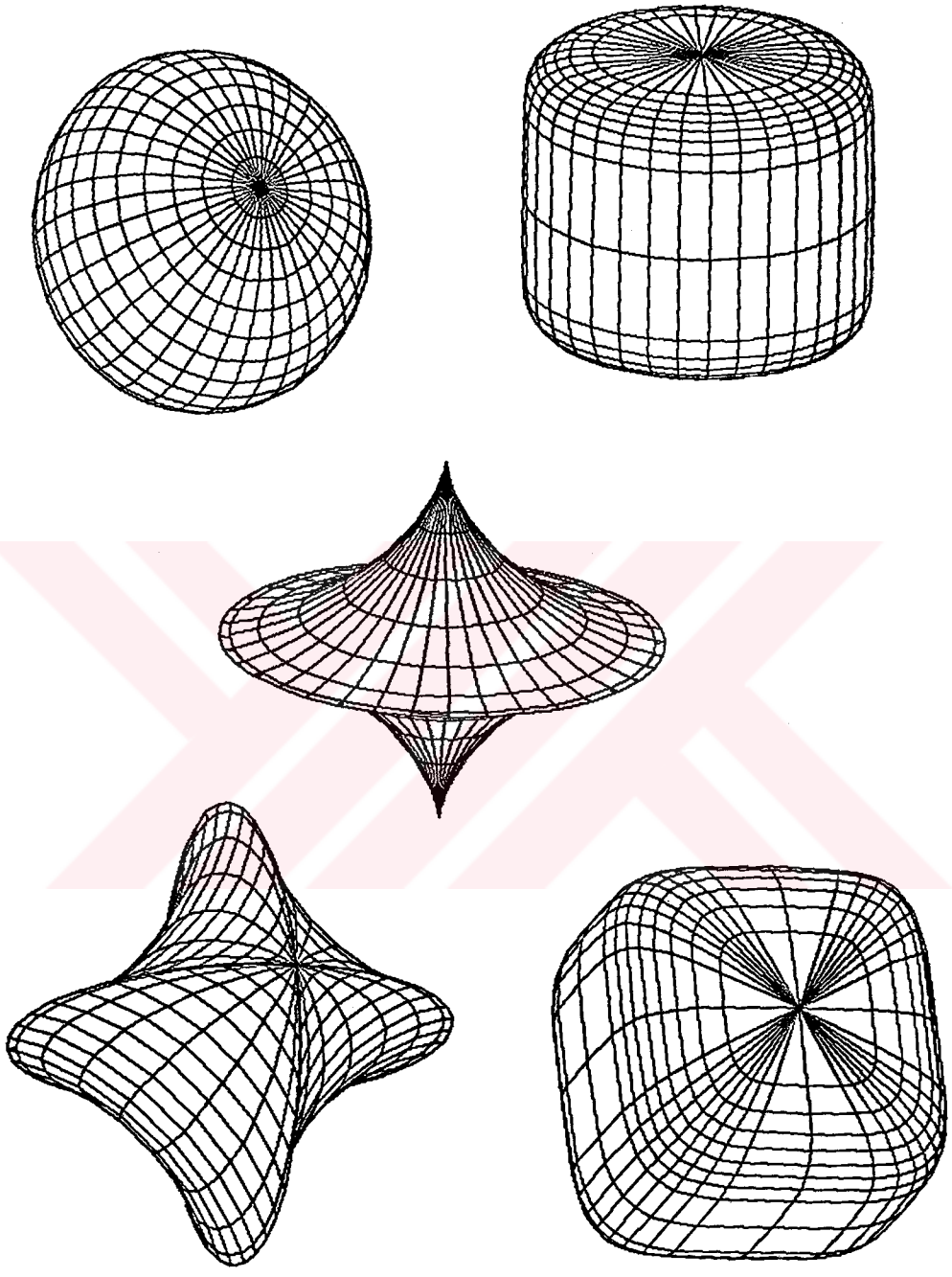

Figure 2.1: Superquadric ellipsoids with different exponents.

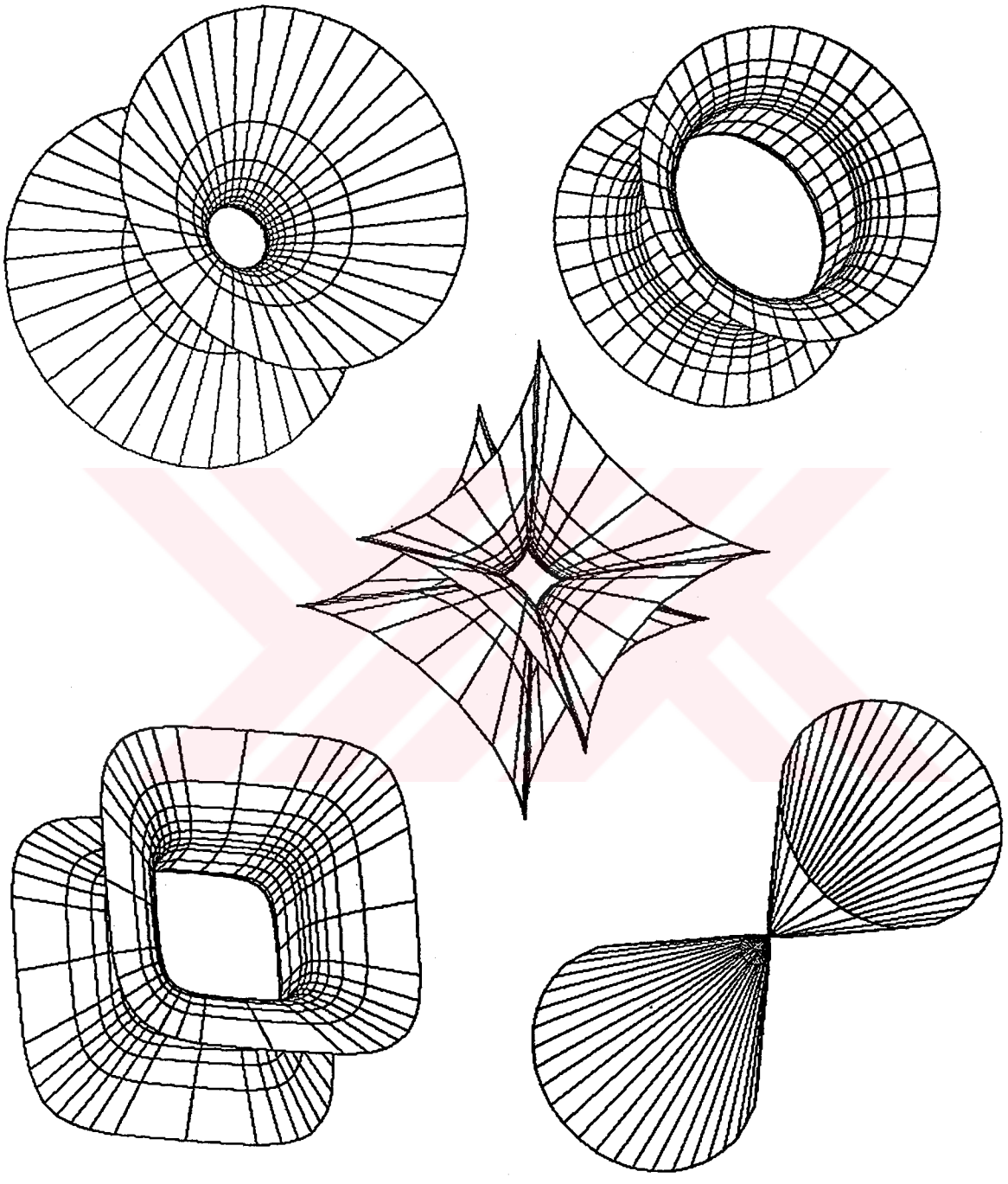

Figure 2.2: Superquadric hyperboloids of one piece with different exponents.

Superhyperboloids of two pieces

Position vector of surface:

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

Normal vector:

$$\underline{n}(\eta,\omega) = \begin{bmatrix} \frac{1}{a_1} sec^{2-\epsilon_1}\eta sec^{2-\epsilon_2}\omega \\ \frac{1}{a_2} sec^{2-\epsilon_1}\eta tan^{2-\epsilon_2}\omega \\ \frac{1}{a_3} tan^{2-\epsilon_1}\eta \end{bmatrix}$$

Wireframe examples of superquadric hyperboloids of two pieces with different exponents are shown in Figure 2.3.

Supertoroids

Position vector of surface:

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

Normal vector:

$$\underline{n}(\eta,\omega) = \begin{bmatrix} \frac{1}{a_1} C_\eta^{2-\epsilon_1} C_\omega^{2-\epsilon_2} \\ \frac{1}{a_2} C_\eta^{2-\epsilon_1} S_\omega^{2-\epsilon_2} \\ \frac{1}{a_3} S_\eta^{2-\epsilon_1} \end{bmatrix}$$

where $a_4 = \frac{\hat{a}}{(a_1^2+a_2^2)}$, and $\hat{a}$ is the radius of the torus.

Wireframe examples of superquadric toroids with different exponents are shown in Figure 2.4.

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 different sets of parametric functions over different

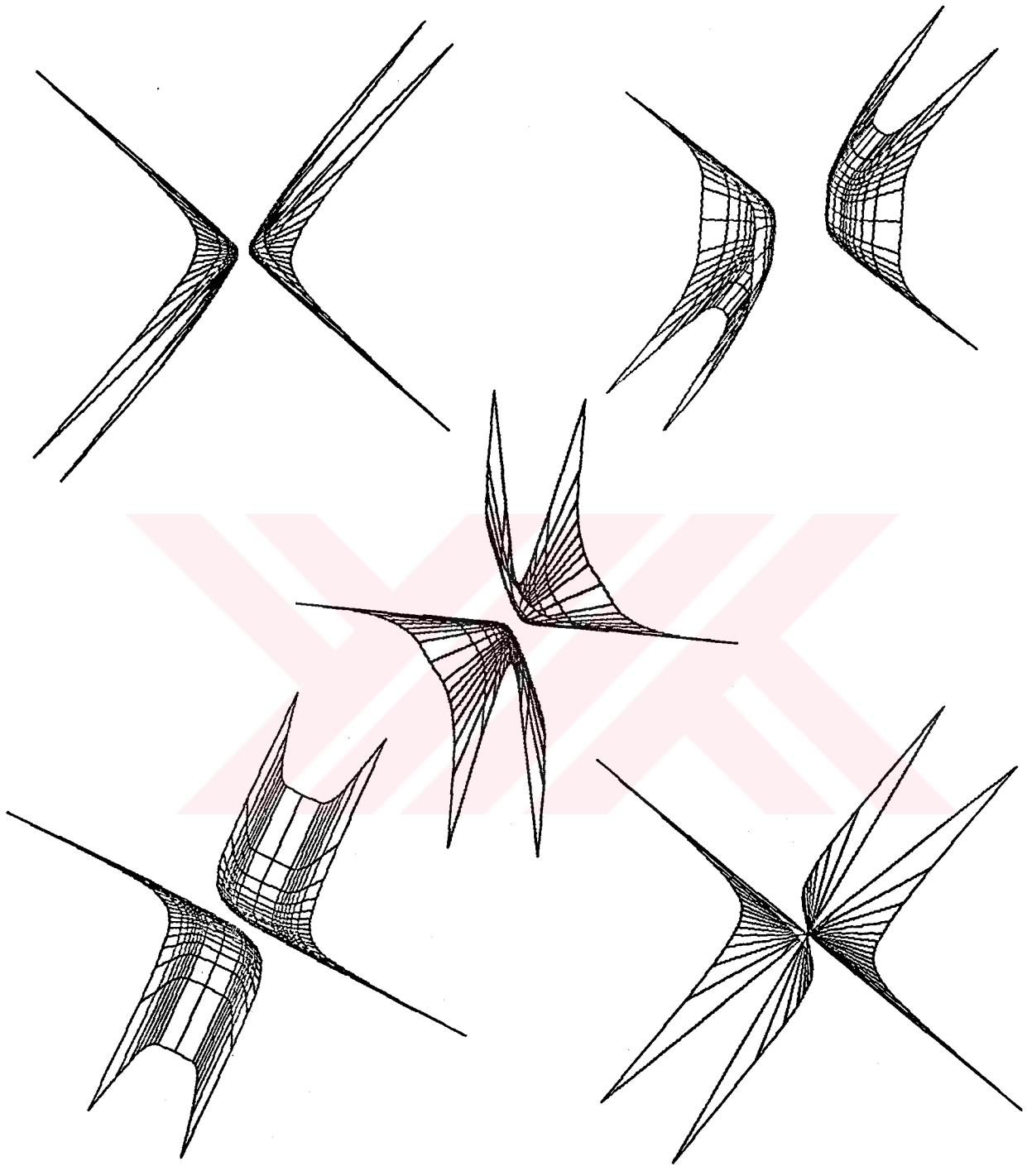

Figure 2.3: Superquadric hyperboloids of two pieces with different exponents.

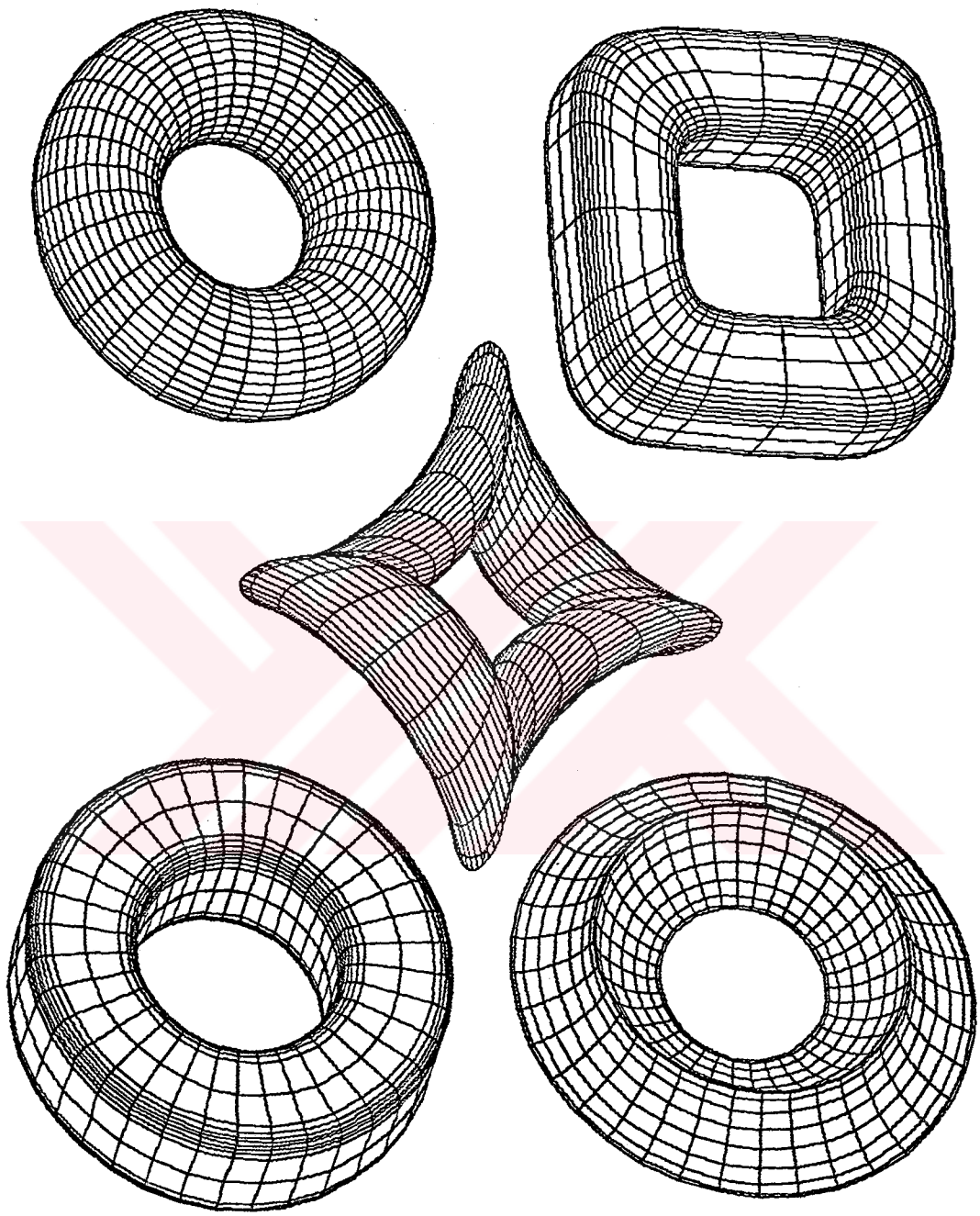

Figure 2.4: Superquadric toroids with different exponents.

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 [5].

Parametric equations for surfaces are formulated with two parameters u and v. A coordinate position on a surface is then represented by the parametric vector function $P(u,v) = (x(u,v), y(u,v), z(u,v))$. The equations for coordinates x, y, and z are often arranged so that parameters u and v are defined within the range 0 to 1.

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 Berstein polynomial basis. A useful feature of Berstein 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 curves can be summarized as follows [5].

Suppose $n+1$ control points are input and designated as the vectors $p_k = (x_k, y_k, z_k), 0 \leq k \leq n$. From these coordinate points, we calculate an approximating Bézier vector function $P(u)$ which represents the three parametric equations for the curve that fits the input control points p_k. It can be calculated as

$$P(u) = \sum_{k=0}^{n} p_k B_{k,n}(u) \qquad (*)$$

Each $B_{k,n}(u)$ is a polynomial function defined as

$$B_{k,n}(u) = C(n,k)u^k(1-u)^{n-k},$$

where $C(n,k)$ represents the binomial coefficients

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Equation (*) can be written as a set of parametric equations:

$$x(u) = \sum_{k=0}^{n} x_k B_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k B_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k B_{k,n}(u)$$

Two sets of Bézier curves can be used to represent surfaces of objects specified by input control points. The parametric function for a Bézier surface is formed as the cartesian product of Bézier blending functions:

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} B_{j,m}(u) B_{k,n}(v)$$

with $p_{j,k}$ specifying the location of $(m+1)$ by $(n+1)$ control points.

Figure 2.5 shows four Bézier surfaces generated by our system.

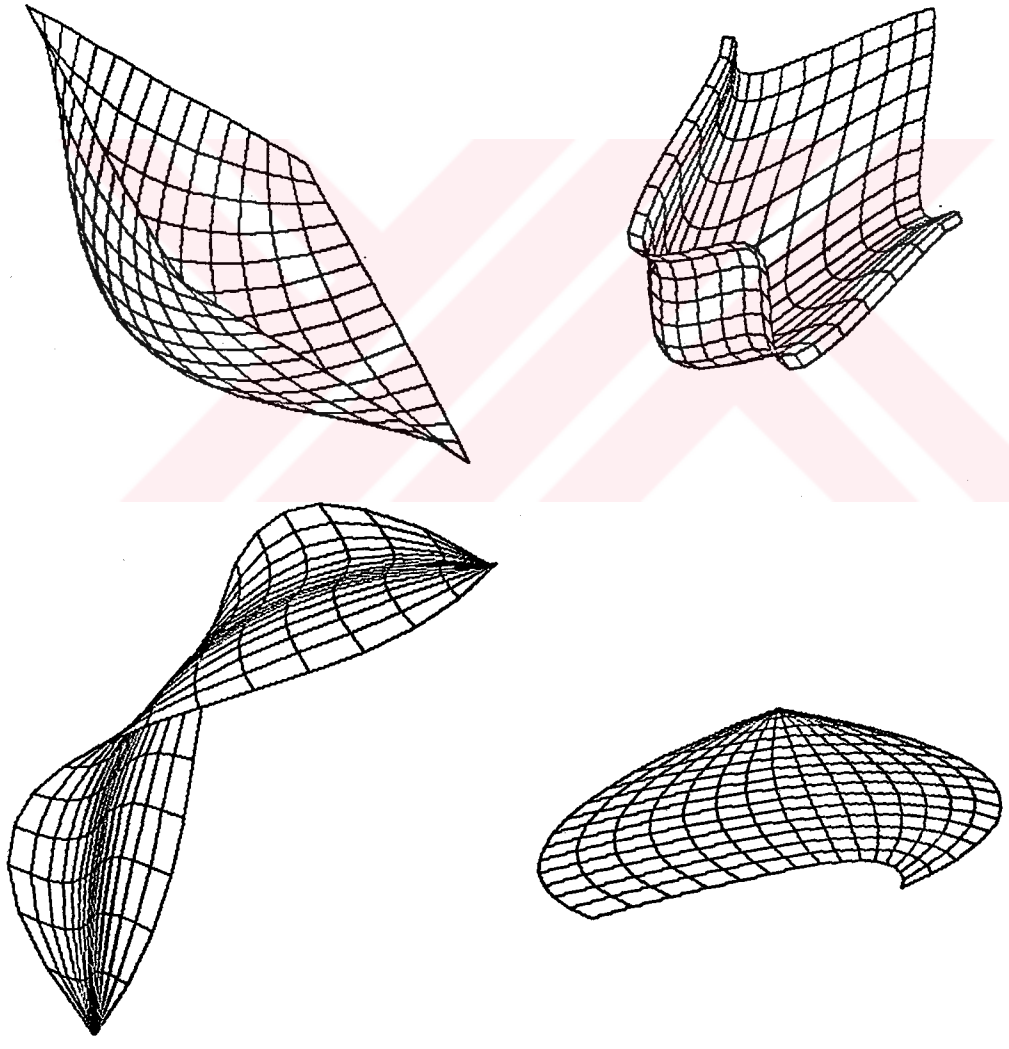

Figure 2.5: Four Bézier surfaces.

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 techniques are used in the implementation. Regular deformations [4] simulate twisting, bending, tapering, or similar transformations of geometric objects. Free-form deformation (FFD) technique [21] is mainly developed to define solid geometric model of an object bounded by free-form surfaces, and further sculpturing it 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 FFD.

A very important disadvantage of FFD is the speed of the deformations since operations on trivariate Berstein polynomials are very costly. It can be made faster by converting trivariate Berstein 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 [22]. This produces better results 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.

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, the user 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 in detail with some examples in the following sections.

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.

A locally specified deformation modifies the tangent space of the solid. It is used to manipulate the tangent vectors which is used for delineating and constructing the local geometry. In other words, it is used to obtain more general shapes.

Tangent and normal vectors on the undeformed surface may be transformed into the tangent and normal vectors on the deformed surface; the algebraic manipulations for the transformation rules involve a single multiplication by the Jacobian matrix $\underline{\underline{J}}$ of the transformation function $\underline{F}$.

The Jacobian matrix $\underline{\underline{J}}$ for the transformation function $\underline{X} = \underline{F}(\underline{x})$ is a function of $\underline{x}$, and is calculated by taking the partial derivatives of $\underline{F}$ with respect to the coordinates x_1, x_2, x_3:

$$\underline{J}_i(\underline{x}) = \frac{\partial \underline{F}(\underline{x})}{\partial x_i}$$

In other words, the i^{th} column of $\underline{\underline{J}}$ is obtained by taking the partial derivative of $\underline{F}(\underline{x})$ with respect to x_i.

New tangent vectors of the deformed surface are calculated by multiplying the tangent vectors of the undeformed surface by the Jacobian $\underline{\underline{J}}$, and new normal vectors of the deformed surface are calculated by multiplying the normal vectors of the undeformed surface by the inverse transpose of the Jacobian matrix.

Each of the regular deformations are explained in detail in the following sections.

3.1.1 Scaling

One of the simplest deformations is a change in the length of the three global components parallel to the coordinate axes. This produces an orthogonal scaling operation which is represented by the following equations:

$$X = a_1 x$$
$$Y = a_2 y$$
$$Z = a_3 z$$

The components of the Jacobian matrix are given by

$$J_{ij} = \frac{\partial X_i}{\partial x_j},$$

so

$$\underline{\underline{J}} = \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix}$$

The volume change of a region scaled by this transformation is obtained from the Jacobian determinant, which is $a_1 a_2 a_3$. The normal transformation matrix is given by:

$$\det J\underline{\underline{J}}^{-1T} = \begin{pmatrix} a_2 a_3 & 0 & 0 \\ 0 & a_1 a_3 & 0 \\ 0 & 0 & a_1 a_2 \end{pmatrix}$$

After converting a point $[x_1, x_2, x_3]^T$ lying on a roughly spherical surface centered at the origin, with normal vector $[n_1, n_2, n_3]^T$, the transformed surface point on the resulting ellipsoidal shape is $[a_1 x_1, a_2 x_2, a_3 x_3]^T$ and the transformed normal vector is parallel to $[\frac{n_1}{a_1}, \frac{n_2}{a_2}, \frac{n_3}{a_3}]^T$. The volume ratio between the shapes is $a_1 a_2 a_3$.

3.1.2 Tapering

Tapering is making an object become gradually narrower towards 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:

$$\begin{aligned} r &= f(z) \\ X &= rx \\ Y &= ry \\ Z &= z \end{aligned}$$

Tangent vector transformation matrix is given by

$$\underline{J} = \begin{pmatrix} r & 0 & f'(z)x \\ 0 & r & f'(z)y \\ 0 & 0 & 1 \end{pmatrix}$$

Normal vector transformation matrix is given by

$$r^2 \underline{J}^{-1T} = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ -rf'(z)x & -rf'(z)y & r^2 \end{pmatrix}$$

Examples of tapering are shown in Figure 3.1 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 $f(z) = cos(z\frac{\pi}{180})$.

3.1.3 Twisting

A twist operation can be approximated as a differential rotation, just as tapering is a differential scaling of the global basis vectors. We rotate one

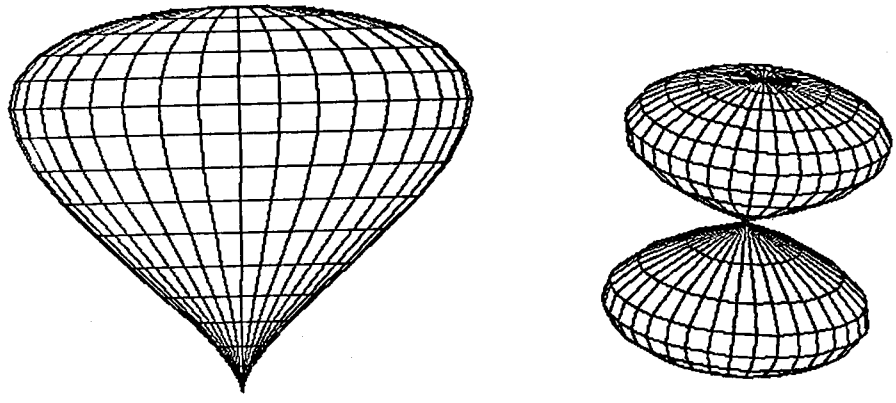

Figure 3.1: Tapered superquadric ellipsoids.

pair of global basis vectors as a function of height, without altering the third global basis vector. An example to 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.

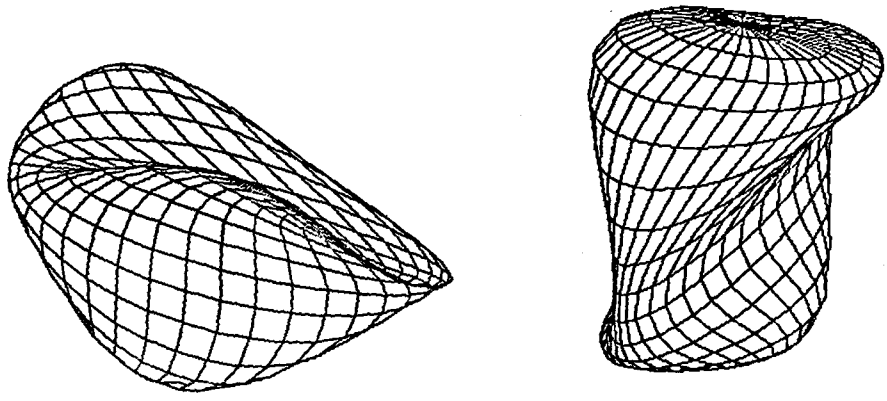

Figure 3.2: Twisted superquadric ellipsoids.

Tangent vector transformation matrix is given by

$$\underline{\underline{J}} = \begin{pmatrix} C_\theta & -S_\theta & -xS_\theta f'(z) - yC_\theta f'(z) \\ S_\theta & C_\theta & xC_\theta f'(z) - yS_\theta f'(z) \\ 0 & 0 & 1 \end{pmatrix}$$

Normal vector transformation matrix is given by

$$\underline{\underline{J}}^{-1T} = \begin{pmatrix} C_\theta & -S_\theta & 0 \\ S_\theta & C_\theta & 0 \\ yf'(z) & -xf'(z) & 1 \end{pmatrix}$$

Examples of twisting are shown in Figure 3.2 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 $\theta = sin(z\frac{\pi}{180})$.

3.1.4 Bending

Bending simulates an important manufacturing process for fabricating objects. An example to 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 $y_{min} \leq y \leq y_{max}$. The axis of the bend is located along $[s, y_0, \frac{1}{k}]^T$, 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.

The bending angle θ is given by:

$$\begin{aligned} \theta &= k(\hat{y} - y_0) \\ C_\theta &= cos(\theta) \\ S_\theta &= sin(\theta) \end{aligned}$$

where

$$\hat{y} = \begin{cases} y_{min}, & y \leq y_{min} \\ y, & y_{min} < y < y_{max} \\ y_{max}, & y \geq y_{max} \end{cases}$$

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

$$X = x$$

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

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

Tangent vector transformation matrix is given by

$$\underline{\underline{J}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_\theta(1-\hat{k}z) & -S_\theta \\ 0 & S_\theta(1-\hat{k}z) & C_\theta \end{pmatrix}$$

where

$$\hat{k} = \begin{cases} k, & \hat{y} = y \\ 0, & \hat{y} \neq y. \end{cases}$$

Normal vector transformation matrix is given by

$$(1-\hat{k}z)\underline{\underline{J}}^{-1T} = \begin{pmatrix} 1-\hat{k}z & 0 & 0 \\ 0 & C_\theta & -S_\theta(1-\hat{k}z) \\ 0 & S_\theta & C_\theta(1-\hat{k}z) \end{pmatrix}$$

Examples to bending operation are shown in Figure 3.3. In these examples, 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°.

The transformations explained above can be combined with rotation around some axes so that these operations can be performed around other axes than the ones explained above.

Results obtained by applying regular deformations hierarchically are shown in Figure 3.4.

3.2 Free-Form Deformations (FFD)

Free-form deformations 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 restricted 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.

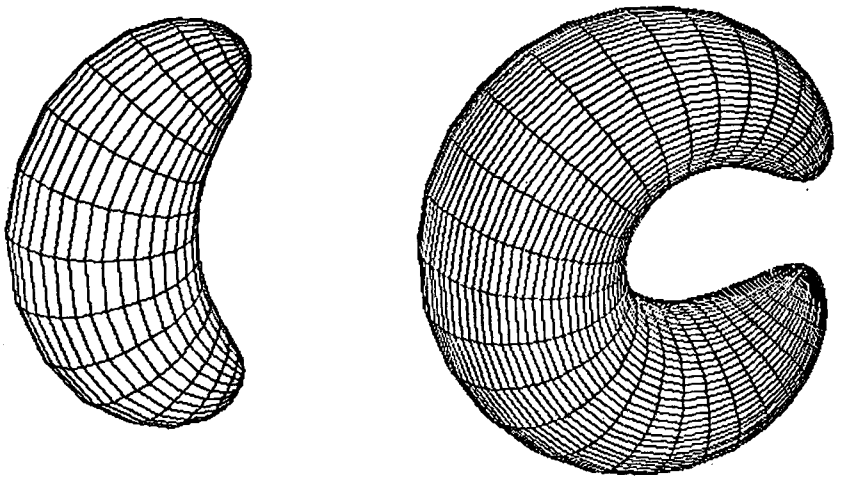

Figure 3.3: Bent superquadric ellipsoids.

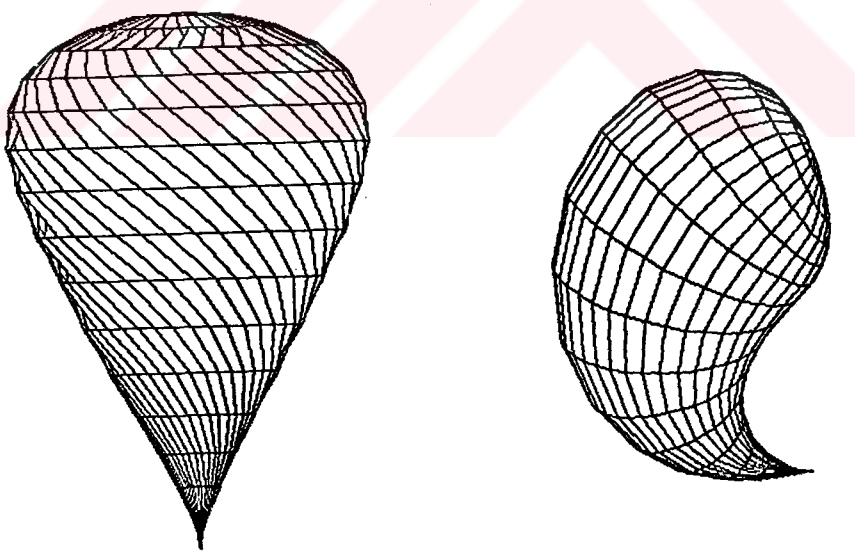

Figure 3.4: A twisted, tapered superquadric ellipsoid, and a tapered, bent superquadric ellipsoid.

3.2.1 Formulation of FFD's

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 out of a flexible material. The objects themselves can also be regarded as if they are made out of a flexible material, so that when the whole grid is deformed, the objects inside them are also deformed respectively.

The control points form a regular lattice which is defined within a parallelepiped generated by three non-coplanar vectors $\mathbf{S}, \mathbf{T}$, and $\mathbf{U}$, and a point $\mathbf{X_0}$. Any point has (s, t, u) coordinates in this system such that

$$\mathbf{X} = \mathbf{X_0} + s\mathbf{S} + t\mathbf{T} + u\mathbf{U}.$$

In our system, $\mathbf{X}$ will typically be a polygon vertex expressed in (x, y, z) coordinates, and $\mathbf{S}, \mathbf{T}$, and $\mathbf{U}$ are analogous to the unit vectors in $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$ direction. The (s, t, u) coordinates of $\mathbf{X}$ can be found as follows:

$$s = \frac{\mathbf{T} \times \mathbf{U} \cdot (\mathbf{X} - \mathbf{X_0})}{\mathbf{T} \times \mathbf{U} \cdot \mathbf{S}}$$

$$t = \frac{\mathbf{S} \times \mathbf{U} \cdot (\mathbf{X} - \mathbf{X_0})}{\mathbf{S} \times \mathbf{U} \cdot \mathbf{T}}$$

$$u = \frac{\mathbf{S} \times \mathbf{T} \cdot (\mathbf{X} - \mathbf{X_0})}{\mathbf{S} \times \mathbf{T} \cdot \mathbf{U}}$$

In our system, the above calculation is not necessary since x, y, and z coordinates of the points are calculated automatically. It can be used when the coordinates of a point are given in vector form.

For any point interior to the parallelepiped, $0 < s < 1$, $0 < t < 1$, and $0 < u < 1$. The grid of control points $\mathbf{P}_{ijk}$ imposed on the parallelepiped forms $l + 1$ planes in the $\mathbf{S}$ direction, $m + 1$ planes in the $\mathbf{T}$ direction, and $n + 1$ planes in the $\mathbf{U}$ direction. So the number of control points is $(l + 1)(m + 1)(n + 1)$ and their undisplaced locations are defined as

$$\mathbf{P}_{ijk} = \mathbf{X_0} + \frac{i}{l}\mathbf{S} + \frac{j}{m}\mathbf{T} + \frac{k}{n}\mathbf{U}$$

The deformation function is defined by a trivariate tensor product Berstein polynomial. The deformation is specified by moving the P_{ijk} from their undisplaced, latticial positions. A point $\mathbf{X}$ is transformed to its free-form deformation position $\mathbf{X}_{ffd}$ by first computing its (s,t,u) coordinates of $\mathbf{X}$ and then evaluating the Berstein polynomial

$$\mathbf{X}_{ffd} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^i$$

$$\times \; [\sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^j$$

$$\times \; [\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^k \mathbf{P}_{ijk} \;] \;]$$

The control points $\mathbf{P}_{ijk}$ are actually the coefficients of the Berstein polynomial. For a polygonal model, each node is passed through the deformation function, and the connectivity is unchanged.

Since it is very time consuming to evaluate a trivariate Berstein polynomial for lots of polygon vertices that are to be transformed, it is wise to convert the Berstein polynomial to standard power polynomial basis which can then be evaluated using Horner's method.

Our system uses the algorithms in [22] to convert a Berstein basis polynomial to a standard basis polynomial and to evaluate the power basis polynomial.

Examples to FFD technique are shown in Figure 3.5 where an ellipsoid is deformed using different FFDs.

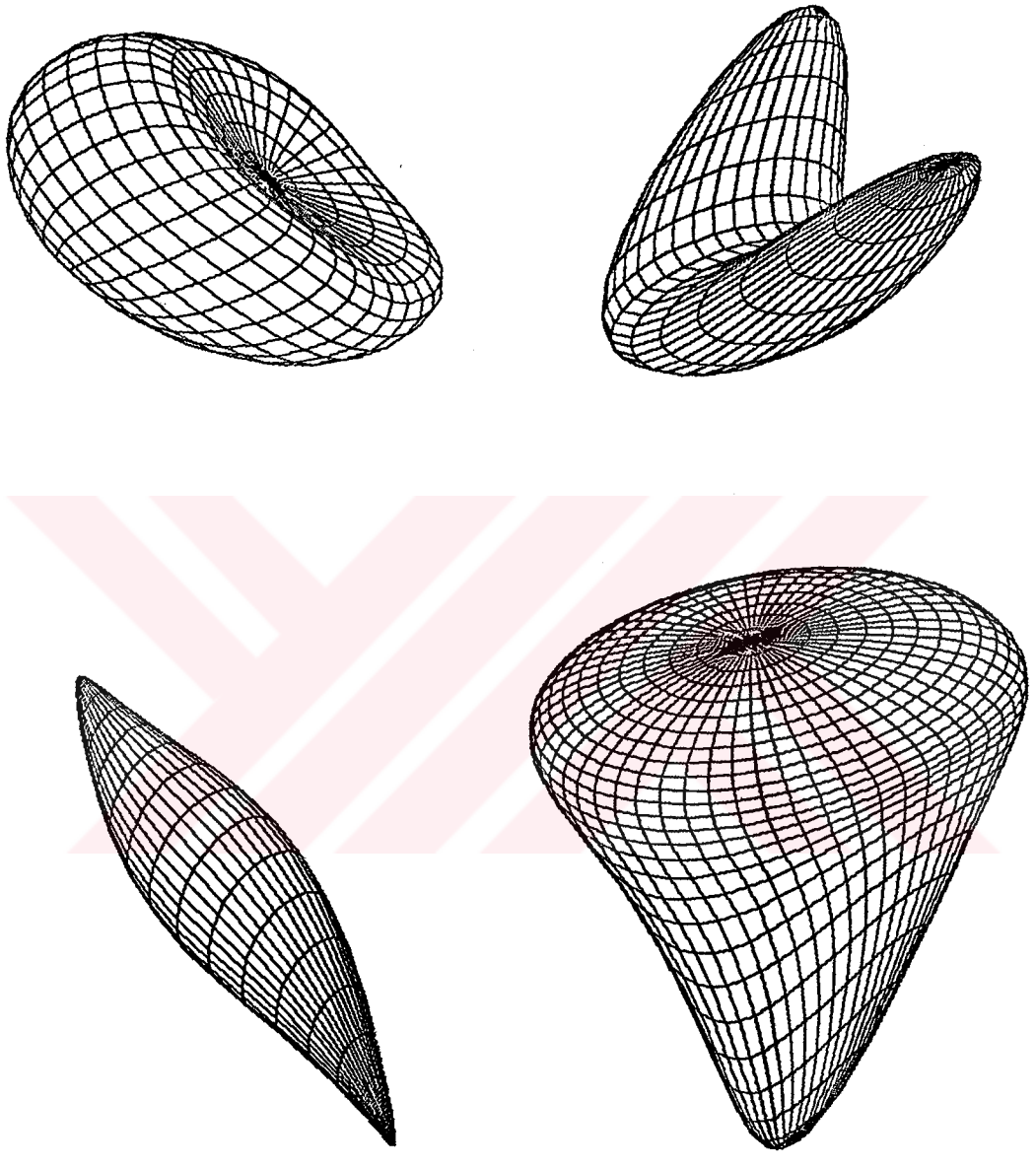

Figure 3.5: An ellipsoid deformed using different FFDs.

3.3 Combination of Regular Deformations and FFD Technique

Since FFD is a method for deforming three-dimensional objects in a free-form manner, regular deformations can also be performed using it. 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. This is also true for other regular deformations. Another reason why using FFD to deform an object is more difficult than using regular deformations is that the user should perform trial and error until the desired object is obtained. FFD technique is slower since using it involves operations with trivariate tensor product Berstein polynomials. Regular deformations are faster since the whole operation is just a single matrix multiplication. Consequently, when regular deformations can perform the desired operation, it is better to use them.

In addition to twisting, tapering, and bending, Barr [4] explains some operations for deforming objects locally when the Jacobian of the deformation function is known. This is another way of obtaining more general shapes. However, the implementation of local deformations is very difficult. FFD technique can be used to achieve the results that can be obtained by using the local operations described by Barr, since it can either be applied locally or globally.

We have applied the two deformation techniques hierarchically to regular classes of objects in our system. Results are shown in Figure 3.6. The first object is obtained by initially tapering a superquadric hyperboloid of one piece and then 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, and the last object is obtained by tapering a superquadric toroid and applying FFD to the tapered object.

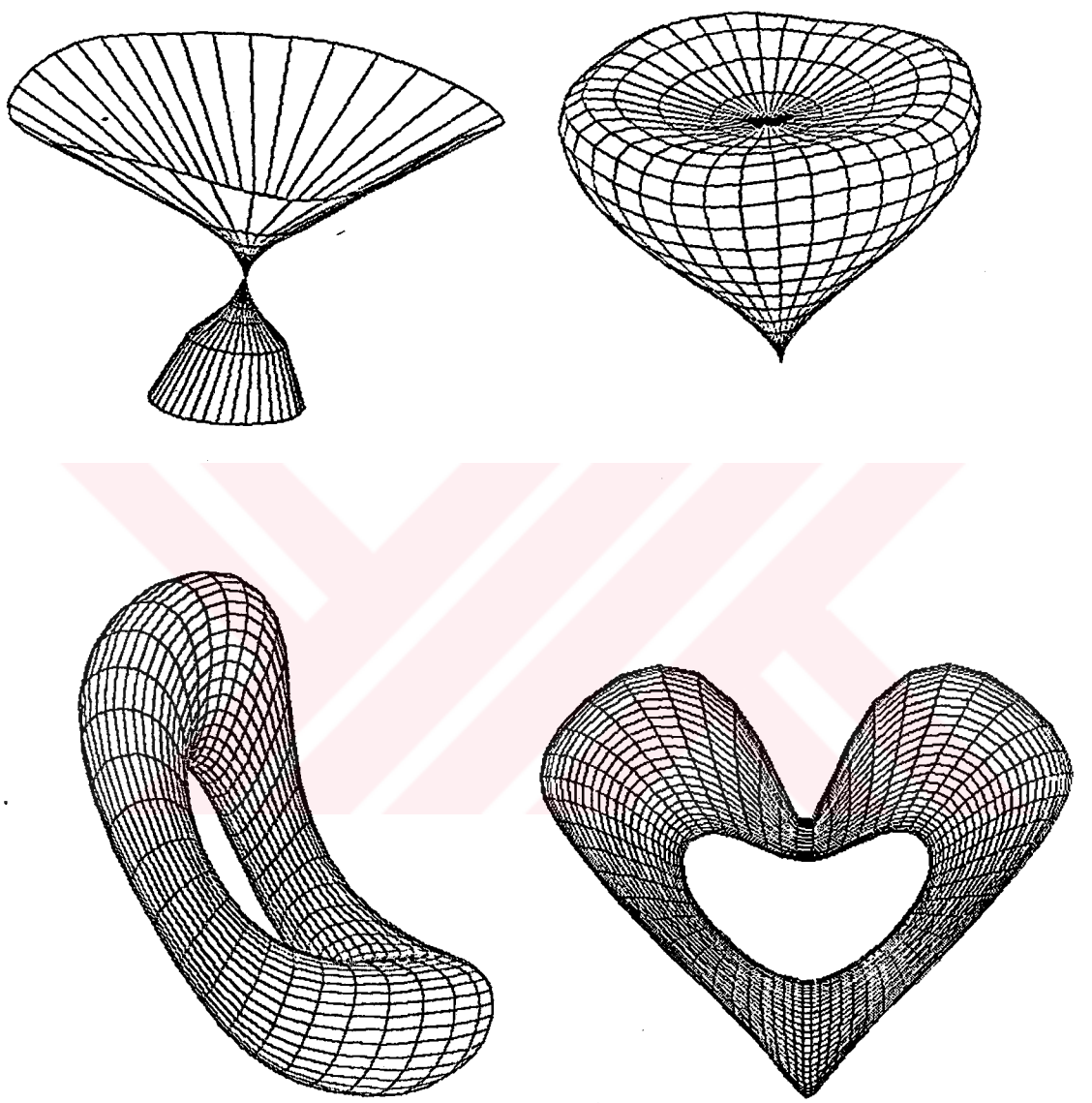

Figure 3.6: Results of applying combination of regular deformations and FFDs.

3.4 An Example of Generating a Composite Shape

In this section, the use of the system is described by an example. In the following example, the system is used to generate a telephone handset from a rounded bar. Steps of the process are as follows :

1. First, it is necessary to generate a rounded bar. This is accomplished by creating a superquadric ellipsoid similar to a rectangular prism. The superquadric ellipsoid in Figure 3.7 is generated by setting the "Resolution" to 72, and giving ϵ_1 and ϵ_2 as 0.52 and 0.52 respectively. Scaling factors are arranged interactively by the user as described in Figure A.2. The superquadric ellipsoid is seen from the top when it is generated. To obtain the shape in the figure, the initial shape is rotated around x axis.

2. FFD technique is hierarchically applied to the rounded bar to obtain the shape in Figure 3.8. The shape is obtained by performing FFD three times. The number of planes in x, y, and z directions of the lattice of control points are (8, 8, 2) in the first and second FFD, and (4, 4, 2) in the third FFD. The way how the coordinates of the control points in the lattice are changed interactively is described in A.9.

3. The shape in Figure 3.8 is applied a bending operation to impart a slight curvature to make it similar to a real handset. To perform bending, the shape should first be positioned in an appropriate way. It is rotated around y axis for this purpose. Then, a 90° bending operation is applied. The way how bending operation is accomplished is described in Figure A.5. The shape is rotated back to its original position to obtain the shape in Figure 3.9.

4. Finally, the hidden surfaces are removed and shading is applied (Figures 3.10, 3.11).

This example also shows how regular deformations and free-form deformation technique can be combined in an appropriate way to utilize the advantages of them.

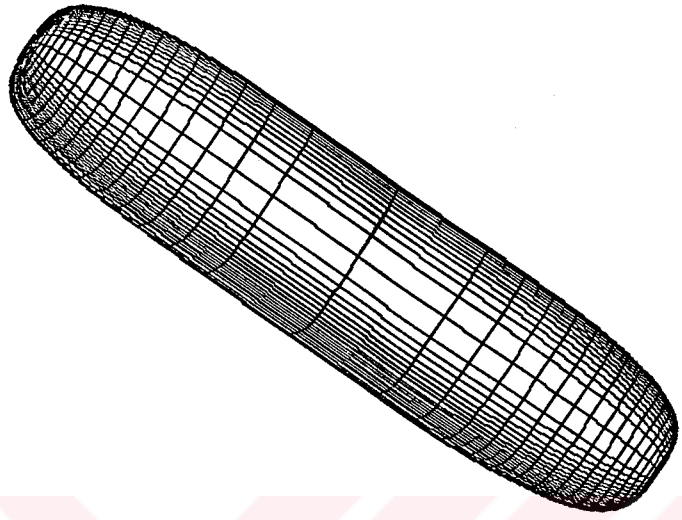

Figure 3.7: The rounded bar initially.

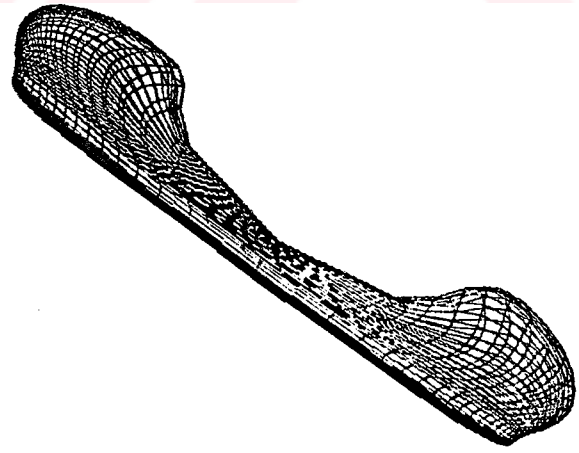

Figure 3.8: The rounded bar after applying FFD three times.

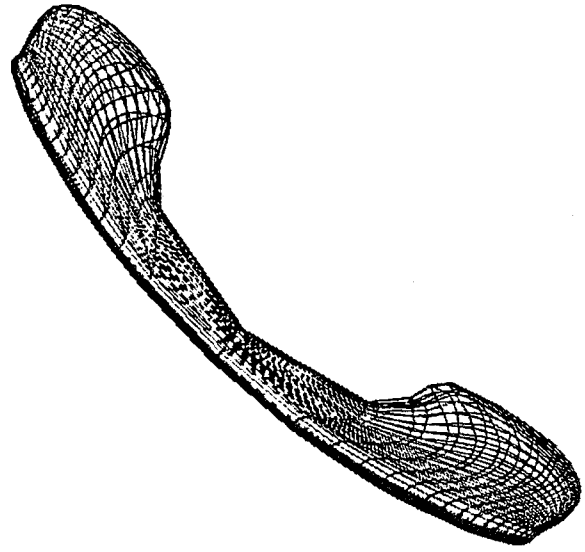

Figure 3.9: Telephone handset generated.

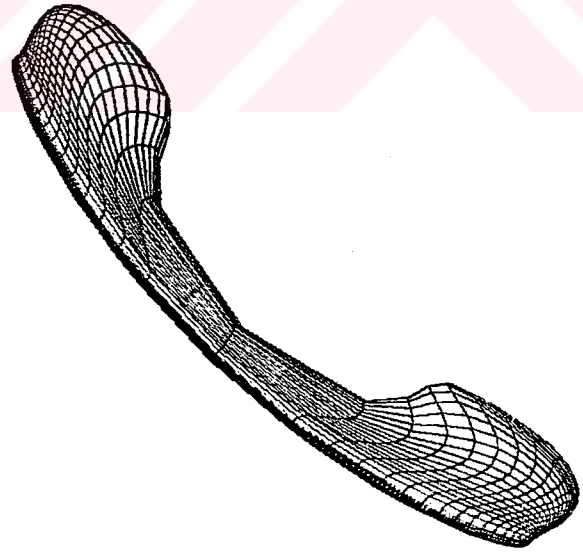

Figure 3.10: Telephone handset (hidden surfaces eliminated).

Figure 3.11: Telephone handset (shading applied).

4. DISPLAY FACILITIES PROVIDED BY THE SYSTEM

Realistic display of objects are obtained by generating perspective projections with hidden surfaces removed and then applying shading and color patterns to the visible surfaces [1].

In our system, curved surfaces, which are superquadric objects and Bézier surfaces, are approximated by small polygons. This helps us to solve hidden surface problem easily. However, it brings some other problems; e.g. to shade an object, we have to use a smooth shading technique, such as Gouraud shading [13], to prevent intensity discontinuities between adjacent polygons of the object. This takes much more time than shading each polygon with a constant intensity value. However, the resolution of objects produced by the system is specified by the user. In other words, the size of the polygons that are used to approximate the objects can be made smaller or larger depending on the quality desired. Consequently, making the polygons smaller and smaller, and applying constant shading on these polygons produces pictures nearly as good as the ones obtained by applying Gouraud shading.

4.1 Hidden Surface Elimination

Since our system's primary goal is the generation of realistic scenes of three-dimensional objects, the identification and removal of the parts of the picture definition that are not visible from a chosen viewing position becomes an important issue. There are many approaches that can be taken to solve this problem. Since the objects are approximated using lots of small polygons, hidden line elimination requires lots of intersection calculations. So we chose to implement hidden surface elimination.

Hidden surface and hidden line algorithms are often classified according to whether they deal with object definitions directly or with their projected images. These two approaches are called object-space and image-space methods, respectively. An object-space method compares objects and parts of objects to each other to determine which surfaces and lines, as a whole, should be labeled as invisible. In an image-space algorithm, visibility is decided point by point at each pixel position on the projection plane [1].

The depth-sorting method, which we chose to solve hidden surface elimination problem is a combination of these two approaches. It performs the following basic operations to eliminate hidden parts:

1. Surfaces, which are small polygons in our system, are sorted in the order of increasing depth.
2. Surfaces are scan-converted in order, starting with the surface of greatest depth.

Sorting operations are carried out in object space, and the scan conversion of the polygon surfaces is performed in image space. Surfaces are sorted according to their distance from the view plane. The intensity values for the farthest surface are then entered into the refresh buffer. Taking each succeeding surface in turn (in decreasing depth order), we *paint* the surface intensities onto the frame buffer over the intensities of the previously processed surfaces.

Steps of painting polygon surfaces onto the frame buffer according to depth can be summarized as follows:

1. Surfaces are ordered according to the largest z value on each surface.
2. The surface with the greatest depth (call it S) is then compared to the other surfaces in the list to determine whether there are any overlaps in the depth. If no depth overlaps occur, S is scan converted.
3. Steps (1) and (2) are repeated for the next surface in the list until all of the surfaces are scan converted (provided that no overlaps occur).

If S overlaps with other surfaces, following tests should be done to determine whether reordering is necessary or not. (The tests are performed in order of increasing difficulty.)

1. The bounding rectangles in the xy-plane for the two surfaces do not overlap.

2. Surface S is on the *outside* of the overlapping surface, relative to the view plane.

3. The overlapping surface is on the *inside* of surface S, relative to the view plane.

4. The projections of the two surfaces onto the view plane do not overlap.

As soon as one test is found to be true for an overlapping surface, we know that the surface is not behind S. Then the same process is repeated for the next surface that overlaps S. If all the surfaces pass at least one of these tests, no reordering is necessary and S can be scan converted. If all of the tests fail with a particular overlapping surface S', S and S' are interchanged in the sorted list.

4.2 Shading

An intensity model can be applied to surface shading in various ways, depending on the type of surface and the requirements of a particular application [1].

Approximation of curved surfaces using a large number of small polygons causes Mach band distortion [18] when each polygon is shaded with a constant intensity value in which each small polygon is distinctly visible. However, in the case where a surface is exposed only to the light from a point source, and the planes subdividing the surface are made small enough Mach band distortion will be very little. It also depends on the surface curvature, the position of the light source, and view position. The curvature of the objects created by the system changes gradually which helps in decreasing the Mach band distortion.

To shade an object approximated by small polygons using constant shading technique, we have to calculate an intensity for each polygon. For superquadric objects, the normals calculated for each surface point are used to calculate the intensity for each polygon. Also, applying regular deformations on these objects does not bring any problem since new surface normals can be calculated using normal vector transformation rules described in previous

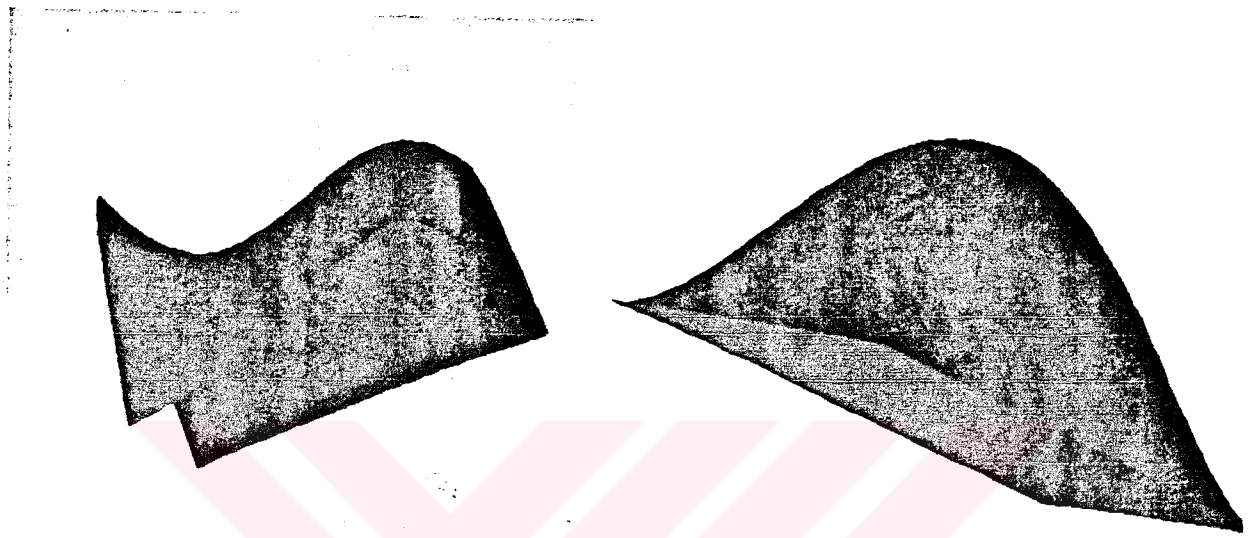

Figure 4.1: Shaded Bézier surfaces.

sections. However, for Bézier Surfaces and for the objects obtained by applying Free-Form Deformations, the system uses analytical methods to find the normals for each polygon.

Figures 4.1, 4.2, and 4.3 show Bézier surfaces, superquadric objects, and objects obtained through deformations with shading applied on them, respectively.

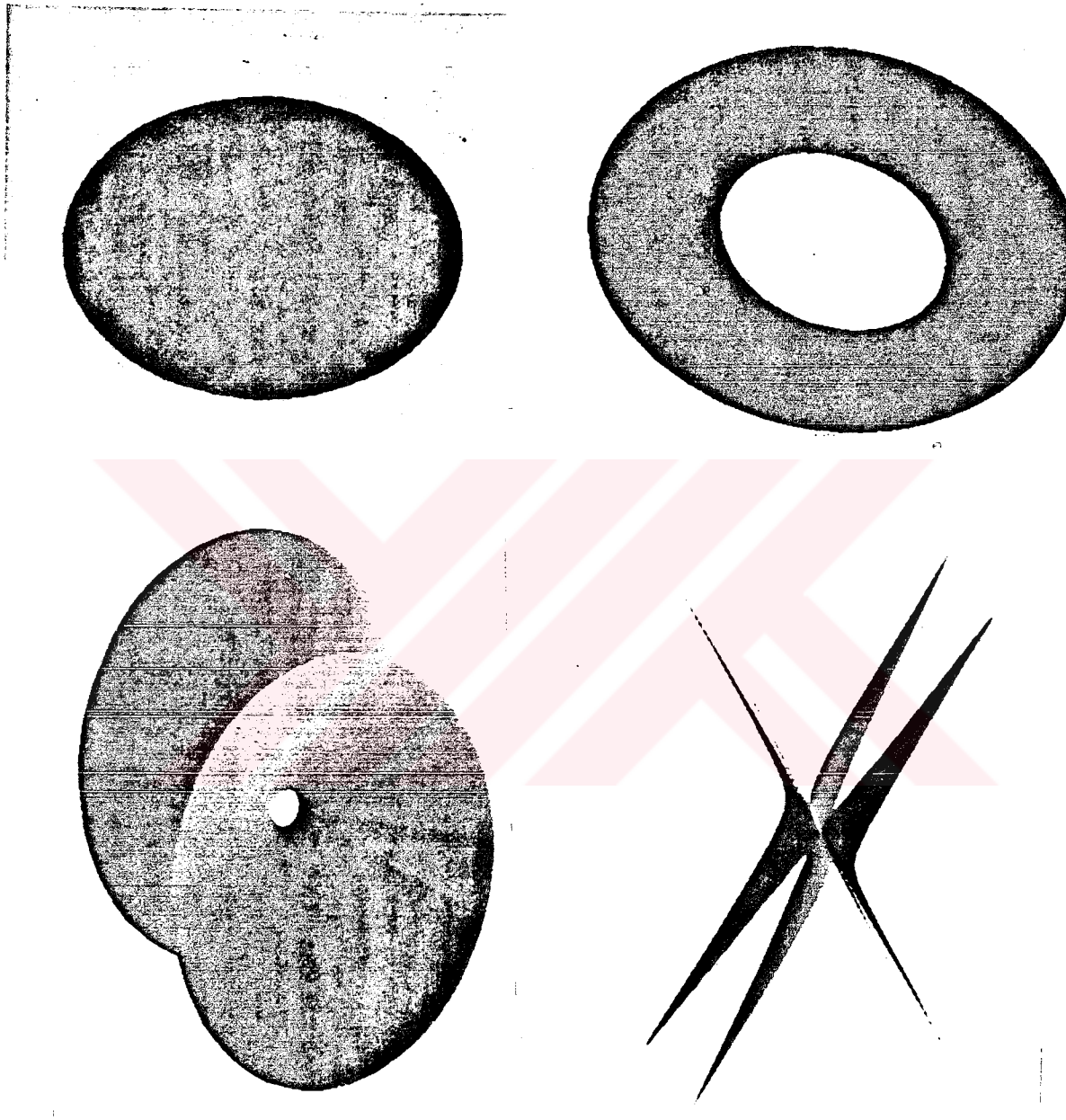

Figure 4.2: Shaded superquadric objects.

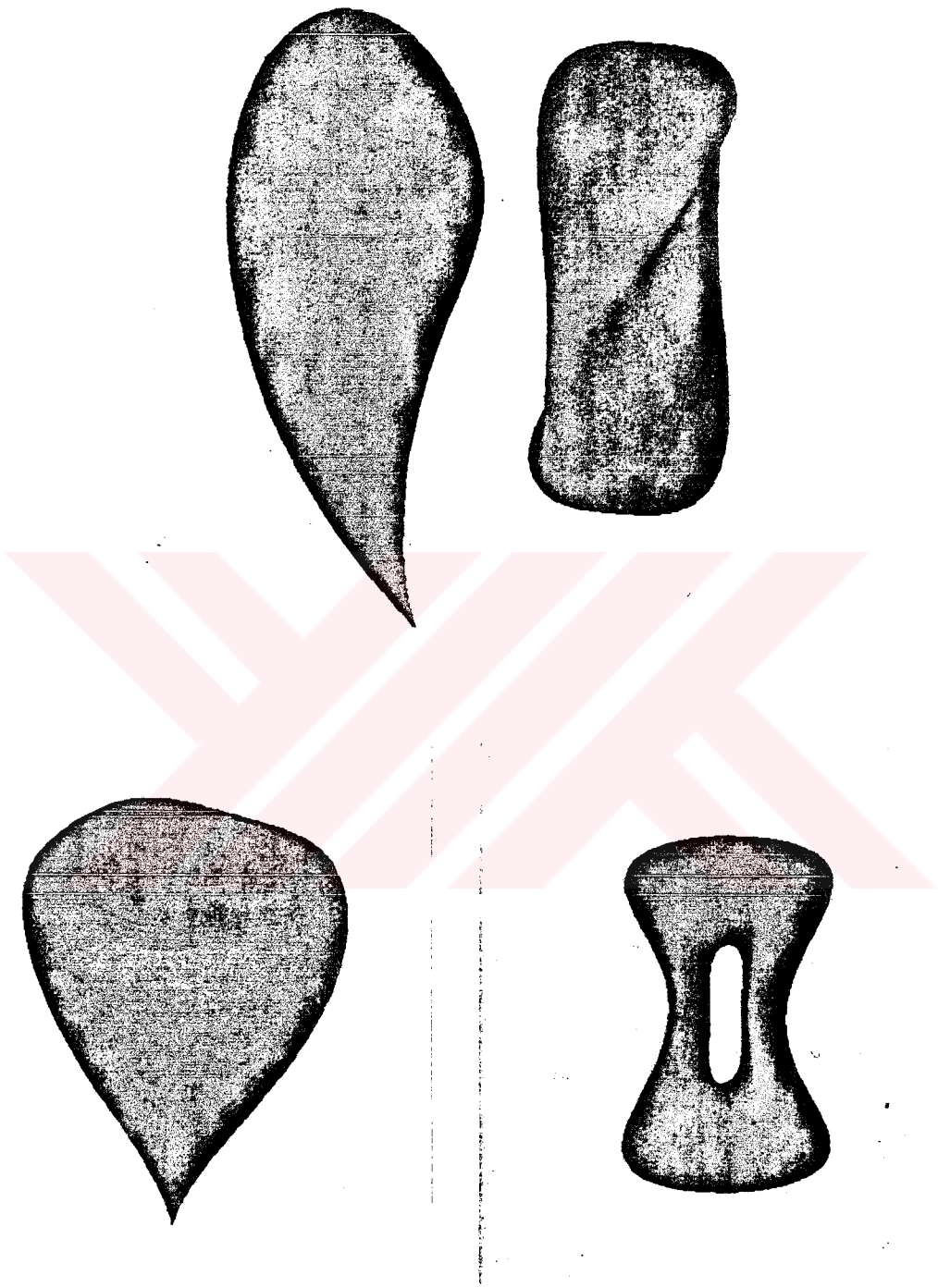

Figure 4.3: Objects obtained through deformations with shading applied.

5. THE USER INTERFACE

It is very important to pay careful attention to the design of interactive user interfaces. Bad user interfaces are not only difficult to learn but they also make a system inefficient to operate even in the hands of an experienced user. In extreme cases an entire system may be invalidated by poor user interface design, that is, it may prove impossible to train users to operate the system, or the user interface may be so inefficient and unreliable that the cost of using the system cannot be justified [17].

Since our system is an interactive one, 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.

5.1 Facilities for Creating Bézier Surfaces and Superquadrics

In our system, the user can create the desired objects with the help of a menu. He 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.

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 he may want

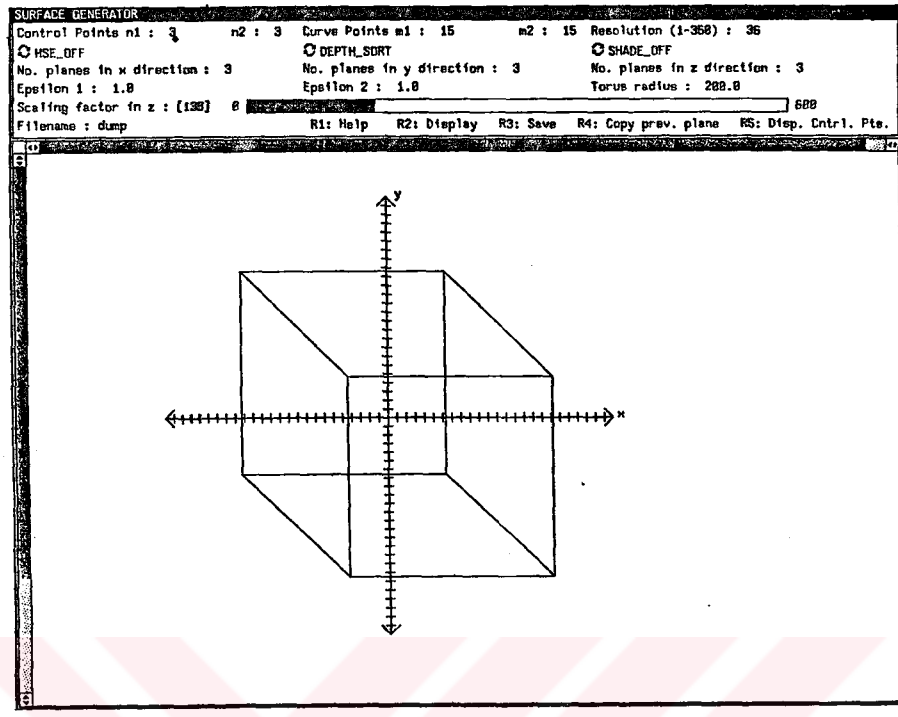

Figure 5.1: The layout of the screen while parameters of superquadrics are given.

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 in Figure 5.1. The items in the panel subwindow are explained in detail in Appendix A. 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.

5.2 Facilities for Deforming Objects

The implementation also provides facilities for deforming the created objects. 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 he selects this option, a new window appears on the screen in which he can enter the function for twisting or tapering operation in infix notation

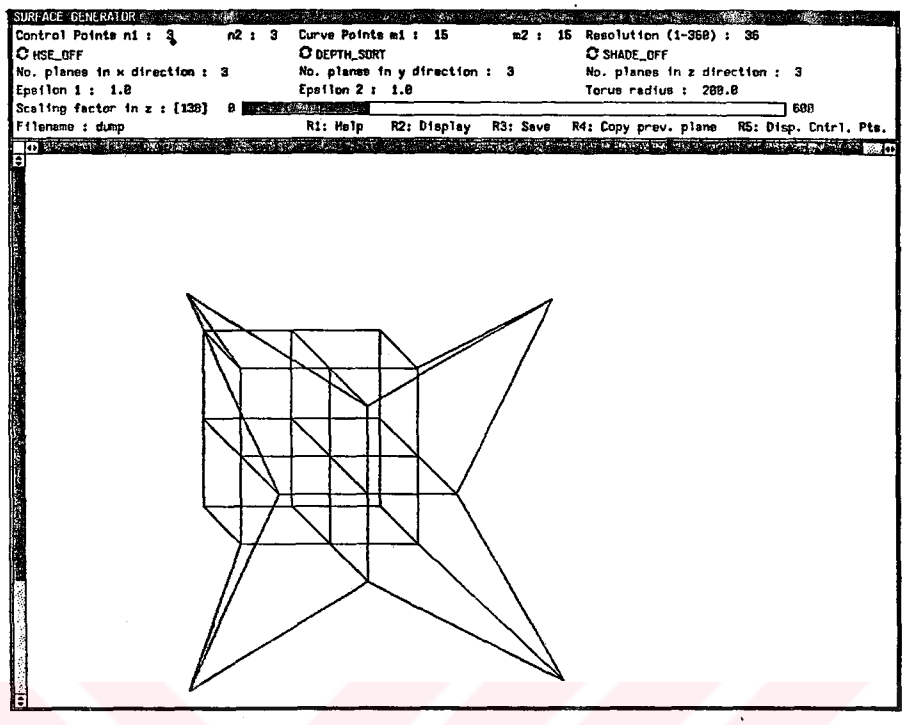

Figure 5.2: A lattice of control points created and deformed by our system.

using 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 may see the lattice of control points any time during that process. After deforming lattice of control points, he can get the deformed object according to the specified lattice. Figure 5.2 shows a lattice of control points generated by our system.

The implementation uses the facilities provided by Sun View[1] system such as windows, panels, and menus [23].

[1]Sun View is a registered trademark of Sun Microsystems.

6. CONCLUSIONS

A system has been 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. Parametric surface modeling methods, namely superquadrics and Bézier surfaces, are used to model undeformed surfaces. Two different deformation techniques, which are regular deformations including twisting, tapering and bending, and free-form deformation technique, are used to deform these objects.

Since both deformation techniques have advantages and disadvantages, we have combined them. FFD technique has two major disadvantages: results are obtained by trial and error and deformations are slow because of operations on trivariate Berstein polynomials. However, FFD is a very good modeling technique because of its generality. Regular deformations are restricted, but their results are straightforward and they are very fast. By combining the two approaches some of the problems peculiar to each method disappear and the advantages of both approaches are utilized.

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.

The user interface of the system is designed to increase the quality of interaction with the system according to the combination of some criteria such as the *time* any user must spend accomplishing a particular project which the system is intended to support, the *accuracy* with which the user can accomplish the project, and the *pleasure* the user derives from the process [12]. Future efforts can be directed towards improving the system in these three aspects.

REFERENCES

[1] Hearn, D. and Baker, M. P., *Computer Graphics*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1986).

[2] Barr, A. H. and Franklin, W. R., "Faster Calculation of Superquadric Shapes," *IEEE Computer Graphics and Applications*, Vol. 1, No. 3 (July 1981), pp. 41–47.

[3] Barr, A. H., "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, Vol. 1, No. 1 (January 1981), pp. 11–23.

[4] Barr, A. H., "Global and Local Deformations of Solid Primitives," *Computer Graphics* Vol. 18, No. 3 (July 1984), pp. 21–30.

[5] Bézier, P., *Numerical Control — Mathematics and Applications*, John Wiley and Sons, London (1972).

[6] Braid, I. C., "Geometric Modeling," *Advances in Computer Graphics I, Eurographics Seminars XII*, Edited by Enderle, G., Grave, M., and Lillehagen, F., Springer Verlag, Berlin (1986), pp. 325–362.

[7] Carlbom, I., Chakravarty, I., and Vanderschel, D., "A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects," *IEEE Computer Graphics and Applications*, Vol. 5, No. 4 (April 1985), pp. 24–31.

[8] Casale, M. S., "Free-Form Solid Modeling with Trimmed Surface Patches," *IEEE Computer Graphics and Applications*, Vol. 7, No. 1 (January 1987), pp. 33–43.

[9] Chiyokura, H., "An Extended Rounding Operation for Modeling Solids with Free-Form Surfaces," *IEEE Computer Graphics and Applications*, Vol. 4, No. 7 (July 1984), pp. 27–35.

[10] Farouki, R. T. and Hinds, J. K., "A Hierarchy of Geometric Forms," *IEEE Computer Graphics and Applications*, Vol. 5, No. 5 (May 1985), pp. 51–78.

[11] Floriani, L. D. and Falcidieno, B., "A Hierarchical Boundary Model for Solid Object Representation," *ACM Transactions on Graphics*, Vol. 7, No. 1 (January 1988), pp. 42–60.

[12] Foley, J. D., Wallace, V. L. and Chan, P., "The Human Factors of Computer Graphics Interaction Techniques," *IEEE Computer Graphics and Applications*, Vol. 4, No. 11 (November 1984), pp. 13–48.

[13] Gouraud, H., "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers*, Vol. C-20, No. 6 (June 1971), pp. 623–629.

[14] Güdükbay, U. and Özgüç, B., "Deformation of Solid Models," *Proceedings of the Fourth International Symposium on Computer and Information Sciences*, Çeşme, Turkey (October 1989, to appear).

[15] Kernighan, B. W., Ritchie, D. M., *The C Programming Language*, Prentice Hall, Inc., Englewood Cliffs, NJ (1978).

[16] Kimura, F., "Geomap - III. Designing Solids with Free-Form Surfaces," *IEEE Computer Graphics and Applications*, Vol. 4, No. 6 (June 1984), pp. 58–72.

[17] Özgüç, B., "Thoughts on User Interface Design for Multi Window Environments," *Proceedings of the Second International Symposium on Computer and Information Sciences*, Istanbul, Turkey (1987), pp. 477–488.

[18] Ratliff, F., *Mach Bands: Quantitative Studies on Neural Networks in the Retina*, Holden-Day, San Francisco, CA (1965).

[19] Requicha, A. A. G. and Voelcker, H. B., "Solid Modeling: Current Status and Research Directions," *IEEE Computer Graphics and Applications*, Vol. 3, No. 10 (October 1983), pp. 25–37.

[20] Samet, H. Tamminen, M., "Bintrees, CSG trees, and Time," *ACM Computer Graphics (Proc. SIGGRAPH)*, Vol. 19, No. 3 (1985), pp. 121–130.

[21] Sederberg, T. W. and Parry, S. R., "Free-Form Deformation of Solid Geometric Models," *ACM Computer Graphics (Proc. SIGGRAPH)*, Vol. 20, No. 4 (August 1986), pp. 151–160.

[22] Sederberg, T. W. and Parry, S. R., "Free-Form Deformation of Polygonal Data," *Proceedings of the International Electronic Image Week*, Nice, France (April 1986), pp. 633–639.

[23] Sun Microsystems, *Sun View Programmer's Guide*, Mountain View, CA (1986).

A. THE USER'S MANUAL

Since the system is implemented using Sun View as described before, it can only be used in this environment. Thus, it is necessary to run *suntools* program to provide the necessary environment. Once the user enters Sun View environment, the system can be initiated by typing *surface* at the Unix prompt. After running the system, the main window will appear on the screen. It mainly consists of a panel subwindow and a canvas. The panel subwindow contains the panel items which are used to handle the user input, and the canvas is used to display the created objects and scenes. In the following sections, the items used to take user input and the help facilities provided are explained in detail.

A.1 Description of the Panel Items

The panel items are mainly used to take text input from the user. The function keys are used for some special operations. All of the panel items and the function keys are listed and described below.

- `Control Points n1: 3  n2: 3`: These text items are used to take the number of control points for a Bézier surface. The number of control points is $(n1+1)$ by $(n2+1)$. The default values of $n1$ and $n2$ are 3.
- `Curve Points m1: 3  m2: 3`: These text items are used to take the number of curve points on a Bézier surface. The number of curve points on a Bézier surface is $(m1+1)$ by $(m2+1)$. The default values of $m1$ and $m2$ are 15.
- `Resolution (1-360): 36`: This panel item is used to change the size of the polygons that are used to approximate superquadric objects. It

represents the increment in the parameters used to generate the surface points in the following way. For example, when "Resolution" is 36, the increment in the parameters will be $\frac{\pi}{36}$. When it is high, the polygons will be smaller and better approximations are obtained. When it is low, the polygons will be larger and rough approximations are obtained. The default value of this panel item is 36.

- `No. planes in x direction: 3`
  `No. planes in y direction: 3`
  `No. planes in z direction: 3:`
  These three panel items are used to take the number of control points of a control lattice which is used for deforming an object using FFD technique. The default values of these panel items are 3.

- `Epsilon 1: 1.0` `Epsilon 2: 1.0`: These panel items are used to take exponents for superquadric objects. The default values for these items are 1; this produces the quadric objects.

- `Torus radius: 200.0`: This panel item determines the radius of superquadric toroids created by the system. The default value of this item is 200.

- `Scaling factor in z: [50]`: This is a slider bar and the values are entered by clicking the left mouse button anywhere in the slider. The values entered are shown in square brackets. This panel item is used to take the z coordinates of points.

- `Filename: dump`: This item is used to take the name of the file that will be used to save a created image.

- `HSE_OFF / HSE_ON`: This is a panel cycle to determine whether hidden surface elimination will be performed or not when an object is displayed. Its purpose is to perform hidden surface elimination only when it is needed since it is a time consuming operation. The default value for this item is `HSE_OFF`.

- `DEPTH_SORTH / BACK_FACE_REMOVAL`: This panel cycle determines the type of hidden surface elimination method that will be used. Since "Back face removal" algorithm works for convex shapes, it can be used for such shapes to gain in speed. For other shapes, "Depth Sorting" method should be used. The default value of this item is `DEPTH_SORT`.

- SHADE_OFF / SHADE_ON: This panel cycle is used to determine whether shading operation will be performed or not. The default value of this item is SHADE_OFF.

- The operations performed using function keys are as follows:

  - R1: When this function key is pressed, the system gives help messages related to the operation performed.

  - R2: This function key is used to display the current object. Its main purpose is to make hidden surface elimination and shading on a created object.

  - R3: This function key is used to save a created image into a file.

  - R4: This function key is used while the user is changing the coordinates of the lattice of control points for an FFD operation. It copies the x and y coordinates of the plane whose coordinates are previously modified to the plane on which the user is currently working on.

  - R5: This function key displays the lattice of control points during an FFD operation.

A.2 Help Facilities Provided by the System

The system has help facilities to make it easily usable by a naive user. Almost all of the operations are performed using mouse buttons unless a text item is required from the user. This means that the mouse buttons have different functions for different operations implemented in the system. Due to this, the user may get confused about which mouse buttons are necessary for an operation that he wants to perform at a particular time.

At each step of a session, user may wish the system to give help that can be achieved by pressing a function key. The system displays appropriate help messages for the operation that the user is currently involved with. The user may either move the help window to a place that he can observe during the operation, or he may destroy it after reading the directions. Help windows provided by the system are reproduced in the following figures.

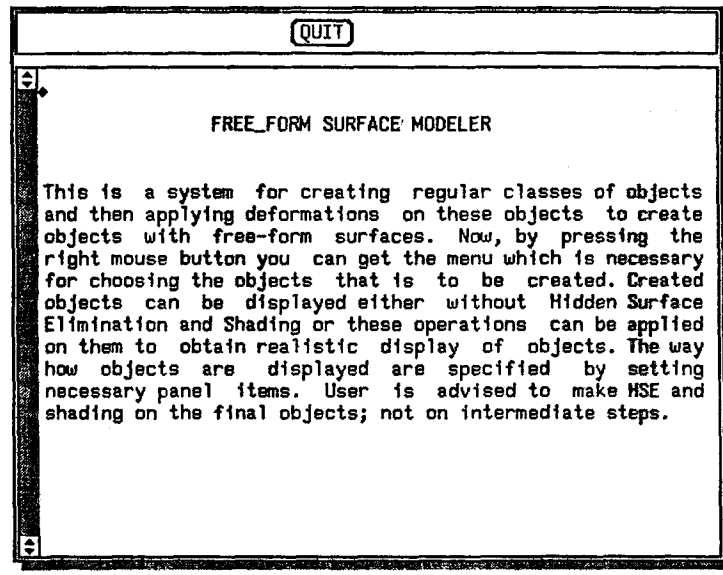

Figure A.1: The help window giving a general idea about the system.

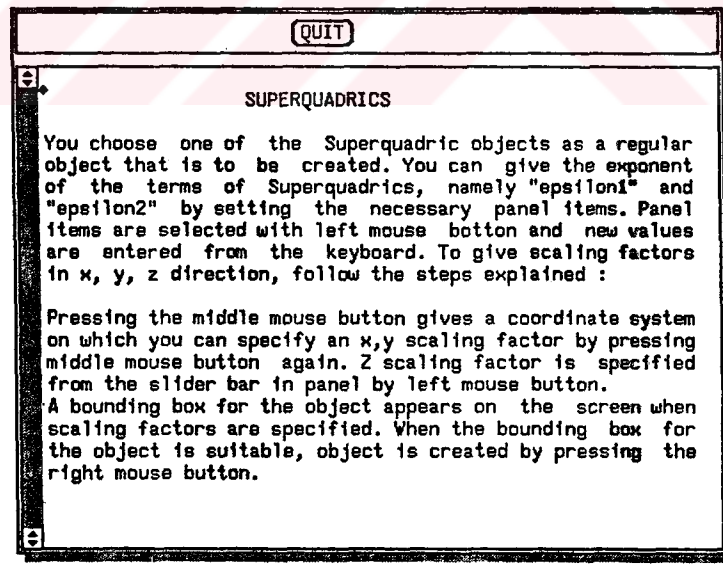

Figure A.2: The help window explaining how scaling factors of superquadrics are entered.

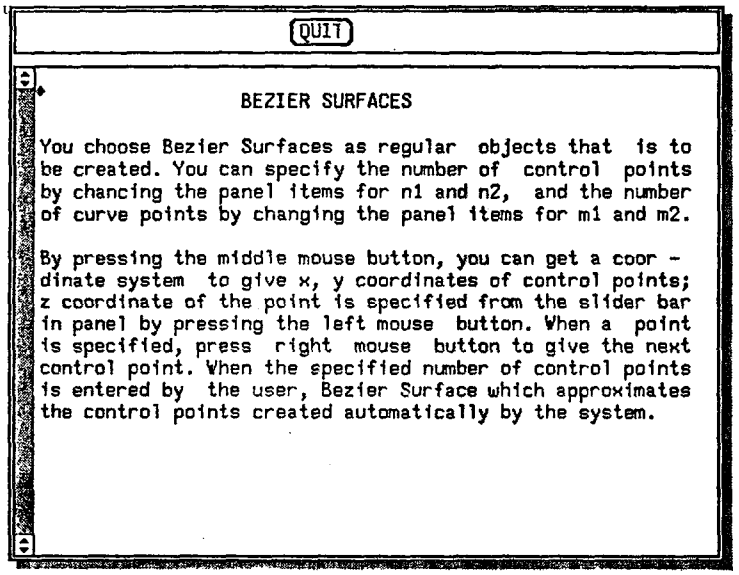

Figure A.3: The help window explaining the entry of control points of Bézier Surfaces.

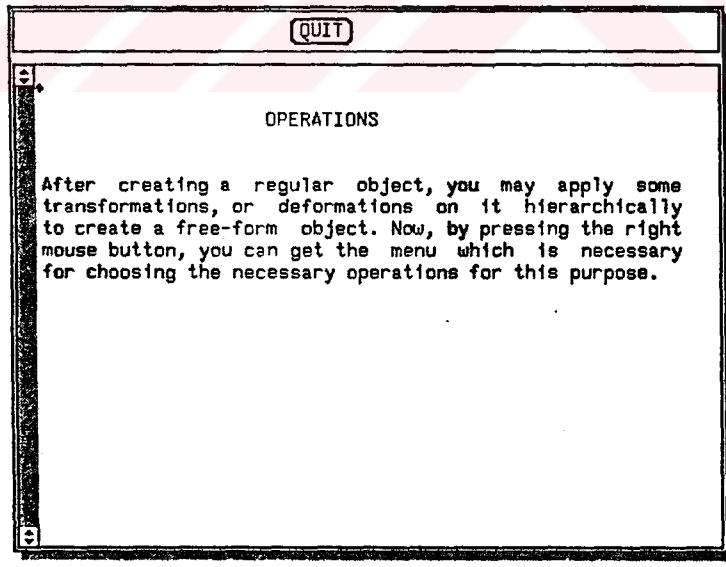

Figure A.4: The help window giving a general idea for the operations that can be done by the system.

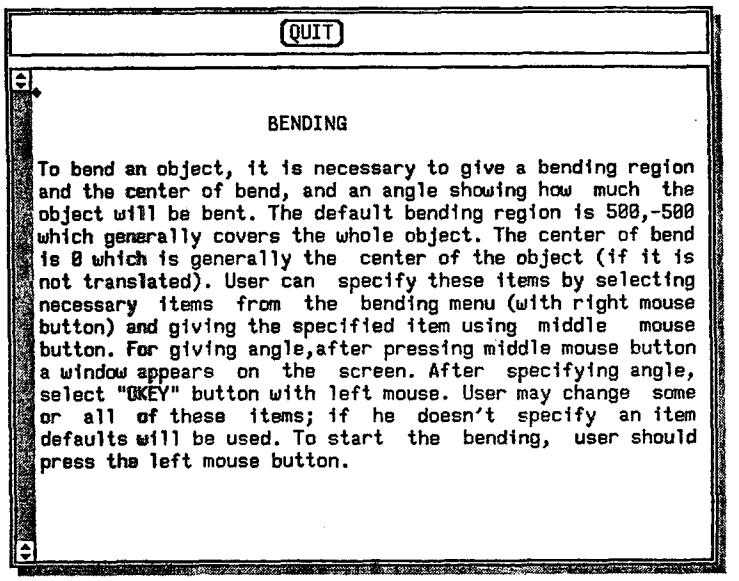

Figure A.5: The help window explaining how parameters are entered for the bending operation.

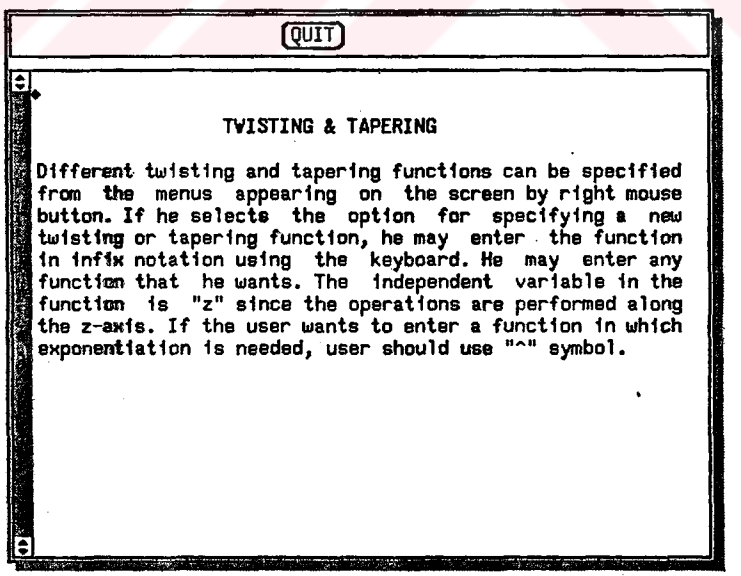

Figure A.6: The help window explaining how different twisting and tapering functions can be entered for these operations.

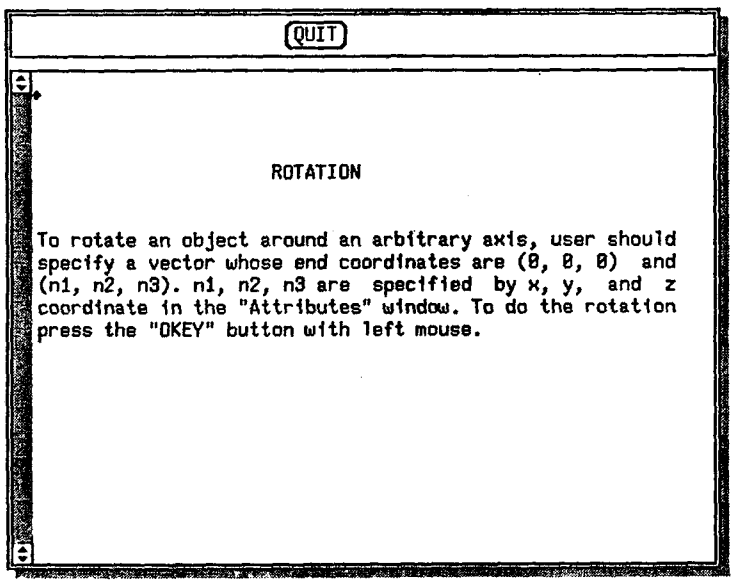

Figure A.7: The help window explaining how rotation parameters are entered.

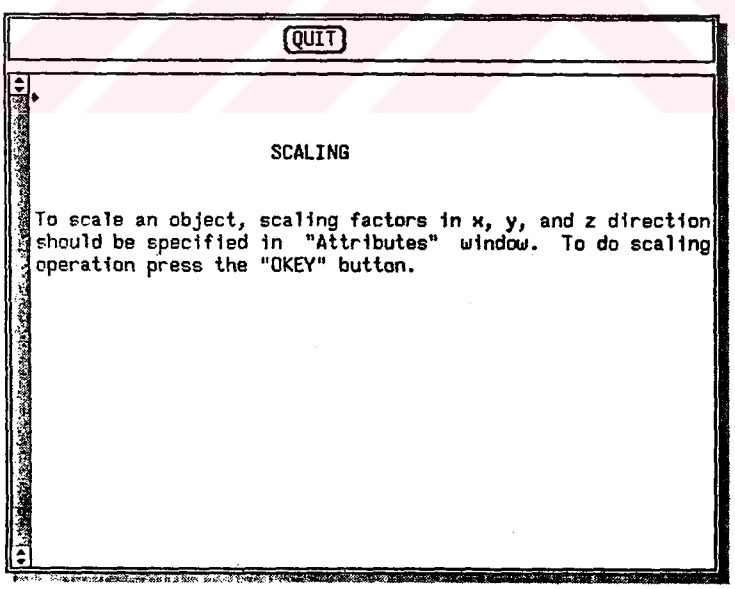

Figure A.8: The help window explaining how scaling parameters are entered.

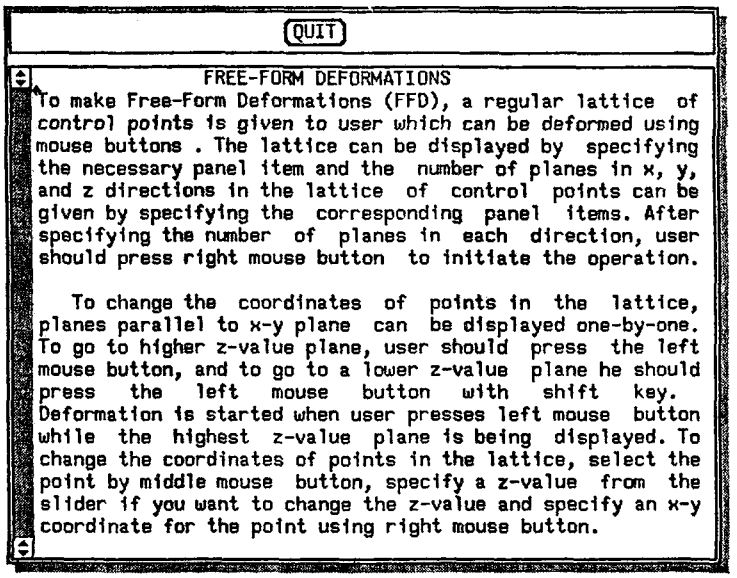

Figure A.9: The help window explaining how the lattice of control points are created and modified for performing an FFD.

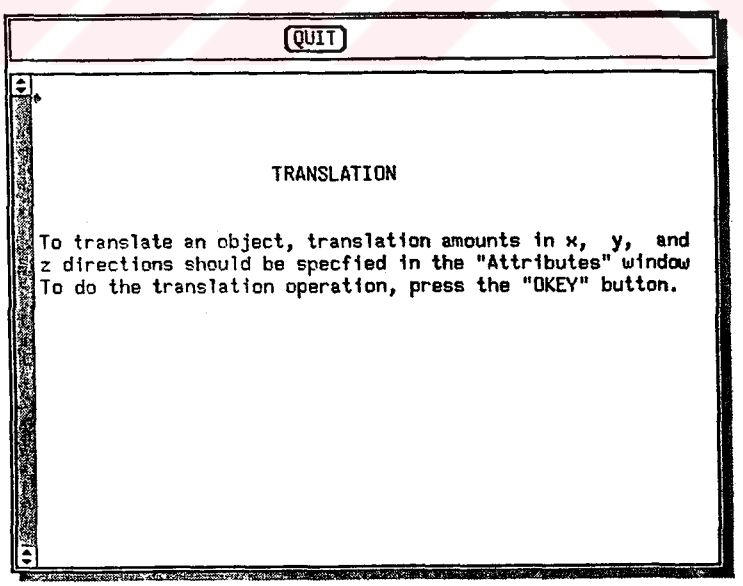

Figure A.10: The help window explaining how translation parameters are entered.