TEXTURE MAPPING ON GEOMETRICAL MODELS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND
INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING and SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT of the REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Oktay Aydın AKTIMUR
July, 1989

TEXTURE MAPPING ON GEOMETRICAL MODELS

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

Oktay Aydın Açıkgöz

July 1989

Oktay Aydın Açıkgöz tarafından bağışlanmıştır.

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. Cevdet Aykanat

Approved for the Institute of Engineering and Sciences:

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

ABSTRACT

TEXTURE MAPPING ON GEOMETRICAL MODELS

Oktay Aydın Açıkgöz
M.S. in Computer Engineering and
Information Sciences
Supervisor: Prof. Dr. Bülent ÖZGÜÇ
July 1989

The contribution of the visual effects of textures is an important aspect in generating images of real objects. Texture mapping is a very successful technique in this respect. Texture mapping can be subdivided into two fundamental topics: the geometric mapping and the filtering. The texture mapping system developed in this study is adaptable to different types of geometric models. Superquadric, Bezier or b-spline surfaces can be mapped with textures. The geometric modeling and the texture synthesis subsystems were also implemented for this purpose. The system works in an interactive manner, the user describes the geometric model and the texture and gets the result in a reasonable amount of time. The speed and the usability of the system by a naive user are the keypoints of implementation.

Keywords : Textures, texture mapping, antialiasing, shading, image synthesis, convolution, color, user interface design, hidden-surface elimination, computer graphics.

ÖZET

GEOMETRİK MODELLERİN DOKULANDIRILMASI

Oktay Aydın Açıkgöz
Bilgisayar Mühendisliği ve Enformatik Bilimleri Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Bülent ÖZGÜÇ
Temmuz 1989

Gerçeğe uygun görüntü elde edilmesinde, dokuların görsel etkilerinin verilmesinin önemi büyüktür. Doku eşlemesi bu bağlamda oldukça başarılı bir tekniktir. Doku eşlemesi iki temel başlık altında incelenebilir: geometrik eşleme ve filtreleme. Geliştirilen sistem değişik geometrik modellere uygulanabilmektedir. Buna örnek olarak "superquadric" veya Bezier yüzeylerini verebiliriz. Geometrik modelleme ve doku sentezi alt sistemleri de bu amaç için geliştirilmiştir. Uygulamanının hızı ve kolay kullanılırlığı önemli noktalardır. Sistem karşılıklı etkileşimli olarak çalışmaktadır. Kullanıcı dokuyu ve geometrik modeli tanımlamakta ve sonucu kısa bir süre içinde alabilmektedir.

Anahtar Kelimeler : Dokular, doku eşleme, karşıeşgörüngeleme, tarama, görüntü bireşimi, katlanma, renk, etkileşim sistemleri, görünmeyen yüzeyleri yoketme, bilgisayarlı çizim.

ACKNOWLEDGMENTS

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

I appreciate my colleagues Uğur Güdükbay, Aydın Kaya, Cemil Türün, Ahmet Coşar, and Veysi İşler for their valuable discussions and comments.

Special thanks to Prof. Dr. Mehmet Baray and Asst. Prof. Dr. Cevdet Aykanat for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Realistic computer-synthesized raster images created by conventional techniques such as geometrical modeling and ray tracing cannot be successful enough since they do not model minute surface details such as bumps, dirts or real textures in a reasonable amount of time. It is very time consuming to model every minute surface detail a real world object has mathematically. To create attractive pictures artificially, real life details must somehow be generated. Otherwise, created pictures are too smooth and lack the small imperfections that nature has.

Surface properties can be described in different ways: texture, geometry, roughness, shininess, finish, opacity, transparency, etc. It is possible to categorize these properties in two groups:

- geometric surface properties
- color surface properties

It is also possible to further subdivide geometric surface properties:

- macroscopic surface geometry
- microscopic surface geometry

Bumps, cracks, wrinkles, surface curvature are types of macroscopic geometry features that are larger than the wavelengths of visible light. Microscopic geometry describes the roughness of a surface.

Bumps that are macrogeometry features are simulated by applying a geometric perturbation function on the surfaces. Blinn [3] was the first to use

this method to produce images of "bumpy" surfaces. In this method the visible surface calculation is performed on the unperturbed surface, while the shading calculations are performed on the perturbed surface. The problem with this method is that, the silhouettes of bumpy surfaces remain smooth.

Microscopic geometry is implicit in the reflection model used. The amount of light reflected specularly and diffusely is dependent on the roughness of the surface. Surely, rough surfaces reflect the light diffusely, rather than specularly. Some parameters are used to adjust the reflections respectively.

Color surface properties can be described as the texture of the surface. There are many definitions of texture. Pickett [32] states that "texture is used to describe 2-dimensional arrays of variations ... The elements and rules of spacing or arrangement may be arbitrarily manipulated, provided a characteristic repetitiveness remains." Hawkins [15] has provided a more detailed description of texture: "The notion of texture appears to depend upon three ingredients: (1) some local 'order' is repeated over a region which is large in comparison to the order's size, (2) the order consists in the nonrandom arrangement of elementary parts, and (3) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region." Here, the word texture is attached a more general meaning, it is a multidimensional image mapped to a multidimensional space. This definition covers nonrepetitive images such as paintings.

Texture can be defined in one, two or three dimensions. In this work 2-dimensional textures are used, and textures are assumed to be in 2-dimensions, unless the contrary is stated.

Texture may be classified as being artificial or natural. Artificial textures are made up of some symbols and figures arranged by human being. Natural textures are images of natural scenes.

One possible way of simulating color surface properties is called texture mapping. The idea is that instead of modeling every minute detail, first model the object geometrically than map onto it a texture that the object might have in reality. Of course by using this technique, it is possible to create images not necessarily realistic but also artistic.

Texture mapping system implementation presented in this work is adaptable to many geometrical modeling techniques, since a planar subdivision is applied to the surfaces. Obviously as the number of planes increase, curved surfaces can be approximated better.

One basic need in this implementation is a tool to synthesize textures to be mapped. This includes images taken by photographic scanners. The tool implemented can combine textures created by different techniques and through different media. For example a black and white picture taken by the photographic scanner can be painted.

In the following chapters, representation of color images, geometric modeling, geometric mapping, and filtering techniques are discussed and some implementation details are given.

2. COLOR TEXTURE REPRESENTATION

The study of color is important in the design and development of color vision systems. The perceptual attributes of color are brightness, hue, and saturation. Brightness represents the perceived luminance. The hue of a color refers to its "redness", "greenness", and so on. Saturation refers to purity, that is, how little the color is diluted by white light. In other words, saturation determines how pastel or strong a color appears.

Color representation is based on the classical theory of Thomas Young [33], who stated that the eye possesses three types of sensors, each sensitive over a different wavelength band. Subsequent findings, starting from those of Maxwell [33] and more recent ones have established that there are three different types of cones in the human retina with absorption spectra $s_1(\lambda), s_2(\lambda)$, and $s_3(\lambda)$ where $\lambda_{min} \leq \lambda \leq \lambda_{max}$, and $\lambda_{min} \cong 380nm$, $\lambda_{max} \cong 780nm$. These responses are peak in the yellow-green, green, and blue regions of the visible spectrum.

Frei proposed a color vision model [33]. In his model three receptors with spectral sensitivities $s_1(\lambda)$, $s_2(\lambda)$, and $s_3(\lambda)$, which represent the absorption pigments of the retina, produce signals:

$$\begin{aligned} e_1 &= \int C(\lambda) s_1(\lambda) d\lambda \\ e_2 &= \int C(\lambda) s_2(\lambda) d\lambda \\ e_3 &= \int C(\lambda) s_3(\lambda) d\lambda \end{aligned} \tag{1}$$

where $C(\lambda)$ is the spectral energy distribution of the incident light source. The three signals e_1, e_2, e_3 are then subjected to a logarithmic transfer function and combined to produce the outputs

$$\begin{aligned} d_1 &= log(e_1) \\ d_2 &= log(e_2) - log(e_1) = log(e_2/e_1) \\ d_3 &= log(e_3) - log(e_1) = log(e_3/e_1) \end{aligned} \tag{2}$$

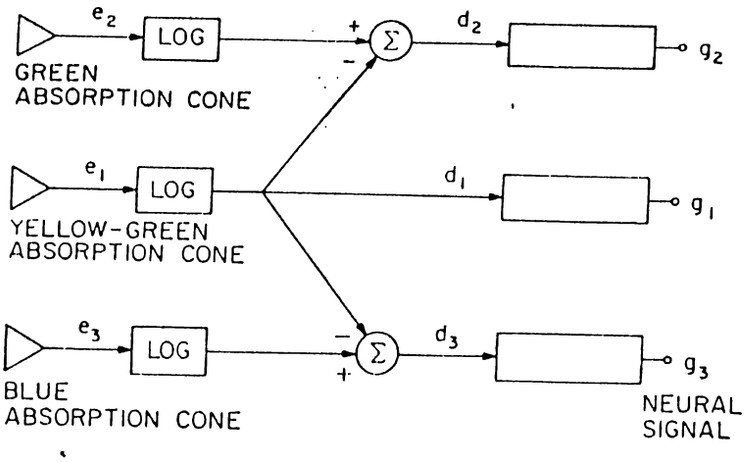

Figure 2.1: Perception of color by the brain

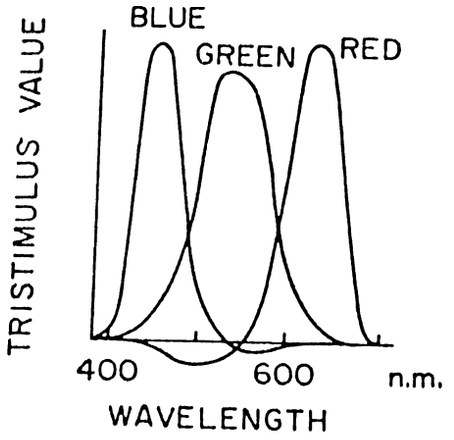

Figure 2.2: Spectral sensitivities of three types of retinal cones.

The signal d_1, d_2, d_3 pass through linear systems to produce output signals g_1, g_2, g_3 that provide the basis for perception of color by the brain, as seen in the Figure 2.1.

In this model the signals d_2, d_3 are related with the chromacity, while the signal d_1 represents the luminance. This model satisfies the basic laws of calorimetry, for example if the spectral energy of a light changes by a constant multiplicative factor then the signals d_2, d_3 representing the chromacity of the light do not change, only d_1 that represents the luminance changes in a logarithmic manner.

Figure 2.2 shows the spectral sensitivities of $s_i(\lambda)$ of the three types of retinal cones obtained by spectral absorption measurements of cone pigments.

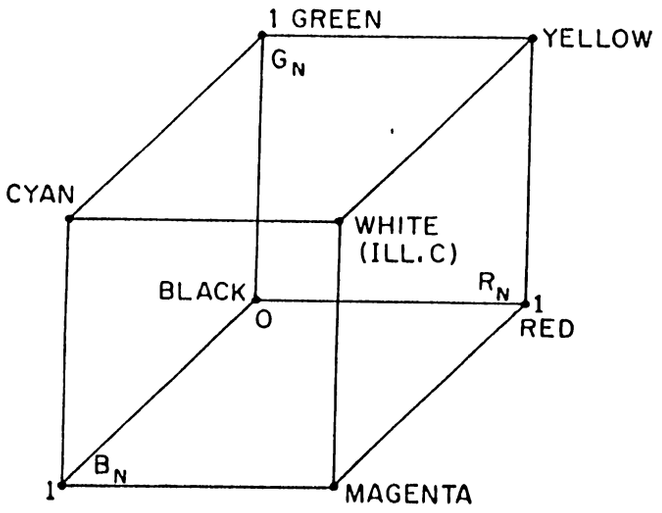

Figure 2.3: The color solid for NTSC Receiver Primary Color Coordinate System.

There are many color coordinate systems employed for the specification of color. These systems have been defined experimentally for the applications requiring different descriptions of the color. Unfortunately, there appears to be no technique for determining an "optimum" coordinate system for most applications. The representation of natural colors is very difficult in color imaging systems. Since physical primaries can only emit positive amounts of light, the colors that require a negative color value cannot be displayed.

The color coordinate system employed in the target machine is NTSC Receiver Primary Color Coordinate System. In this system there are three phosphor primaries that glow in the red, green, and blue regions of the visible spectrum. The color solid for this system is shown in Figure 2.3. In the target machine color images are stored in three parts. In the first part information related to the size of the image is stored. The second part is the color lookup table holding 256 combinations of red, green, and blue colors each having a possibility of 256 intensities. The rest of the image file is a 2 dimensional array of indices to the color lookup table.

2.1 Color Lookup Table Problem

The machine on which the implementation has been carried out has a limited size of colormap (color lookup table), 256. Since the space for the colors to be used simultaneously is restricted by this number it becomes a problem to

find a place for those colors obtained from filtering and shading. Since each of red, green, and blue may yield 256 different intensities for a pixel in the screen space, a total of 16 billion different colors are possible for each pixel and they cannot be predicted before filtering and shading. The only solution to this problem is an approximation method: if there are not sufficient entries in the colormap for a calculated pixel color then an approximate one is chosen. There can be many different strategies with different time consumptions. One possible way is to start with an empty colormap and fill it as the color intensities for pixels are calculated, when no more space is left then the remaining pixels are given approximate values from the filled colormap. Another strategy is to load the colormap homogeneously by selecting some colors as the representatives of all colors. While the first performs better when the number of calculated colors do not exceed 256 much, the second has the advantage of limiting the difference between the calculated colors and the assigned ones. The filtering work based on the bilevel and gray level machines does not have problems like this. Some other and more general techniques can be developed in order to minimize the difference between the actual colors calculated and colors displayed. Unless the size of the reference (pixel depth) and the colormap can be increased, realization of an optimum colormap usage seems extremely hard, and may be solved by using operations research techniques. In order find the most approximate color, some table accesses should be done. Each time searching the whole table is the waste of time. Therefore, a hashing function is used to save time by minimizing the number of these accesses.

2.2 Paint Brush

In a texture mapping system, the need to create textures (2-dimensional images) is very obvious. As previously stated, textures may be defined as functions that have two parameters x and y and a vector value for red, green, and blue intensities. However, it is not very easy to recognize such mathematically defined textures and to perceive a surface mapped with such a texture. Therefore, the most reasonable way is to give the user capability of creating texture images he/she imagines by a simple tool. Such a tool should be able to process pictures taken by photographic scanners, create new texture images, load and modify previously prepared images, and convert them into a form to be used by the texture mapping program.

As a part of the system, the paint brush subsystem was implemented on

top of a user interface toolkit, namely SunView[1] [36]. This is either used for generating texture patterns or editing existing ones that have been previously generated by this system or inputted via a video camera.

A paint brush system requires a high level of user interface. Therefore, the most suitable interface tool seems self explanatory icons. For example, a naive user can easily understand what a pen, a duster or scissors mean and can use them effectively to create the pictures he wants to draw.

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

3. GEOMETRICAL MODELING

For the creation of realistic or interesting images of objects, first it is necessary to construct surfaces of those objects. Surfaces should be defined in such a way that their properties such as visibility, color, normal, etc. can be computed in a convenient way for algorithms. Objects may be described by the surfaces that bound them or by the volume they occupy. Generally attractive surfaces are created by surface oriented techniques, whereas volume oriented techniques are useful for computer aided shape design. Traditionally the synthetic imagery is generated from polygonal models, recently, parametric or implicit surfaces have become popular. A parametric surface is defined by a form such as $F_x(s,t)$, $F_y(s,t)$, $F_z(s,t)$ while an implicit surface is defined as a function $F(x,y,z)$. Quadric and cubic patches are defined by parametric descriptions. Implicit descriptions are usually used for solid modeling systems.

Polygonal surfaces have the advantage of linearity of all the elements. As a result, calculations such as intersections or transformations can be performed in a quick and simple way. Sometimes an enormous number of polygons is needed to approximate a complicated curved surface, however any shape can be approximated upto an arbitrary precision with an arbitrarily large collection of polygons.

Although the higher-order surface descriptions are more compact, intersection or transformation computations for them are more complex and time consuming.

3.1 Data Structures

Even though a polygonal mesh may be described by listing the coordinates of vertices of each polygon in order, this description wastes memory by not

considering vertices shared by more than one polygon. Therefore, it is a reasonable way to store all vertices in a list and then to refer to the vertex list for the polygon coordinates. This structure can be enhanced by storing information such as surface or vertex normals, common edges between the polygons. The data structure in this implementation using C language notation is as follows:

```
struct coordinate {
    int x;
    int y;
    int z;
}

struct vertex {
   struct coordinate location;
   struct coordinate normal;
}

struct vertex *vlist;

struct polygon {
    struct vertex *v1;
    struct vertex *v2;
    struct vertex *v3;
}

struct polygon *polylist;
```

As it is seen, this is a very simple structure. A normal vector is stored with each vertex, since Gouraud [21] polygon shading technique is used. The "polygon" structure has only three vertices, that is, polygons are triangles. The quadrilateral patches are subdivided into two triangles, in order to get linear surfaces.

Since the surfaces are also shaded while being textured, the normal of each vertex is computed during the modeling. Each vertex normal is the average of normals of the planes sharing that vertex. The intensity values at vertices are calculated and used to calculate the intensity values at the edges connecting two vertices. The same interpolation is used for the scan lines

connecting two points on the edges. This technique is known as Gouraud shading and explained in [21].

Complex objects that consist of many surfaces require a more complicated structure. For objects assembled as a collection of intersecting surfaces, the surface for each subobject may be listed separately. Surface characteristics such as color, glosiness can belong to an entire subobject.

Vertices or control points may be grouped into independent movable parts. This allows coordinate transformations to be applied to selected parts of the coordinate data, causing some parts to move independently of others.

Similarly, nonlinear surface patches can be defined as a list of coordinates of control points. Some nonlinear parametric surfaces that have been implemented are explained in the following subsection.

3.2 Parametric Surfaces

Parametric surfaces describe the shape of an object by some parameters. One way of representing curved surfaces is by parametric equations. 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

$$P(u,v) = (x(u,v), y(u,v), z(u,v)) \tag{1}$$

Usually parameters u and v are defined within the range 0.0 to 1.0.

Parametric surfaces can be specified using a set of control points. Bezier [29] surfaces are defined by the formula

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} B_{j,m}(u) B_{k,n}(u) \tag{2}$$

with $p_{j,k}$ specifying the location of the $(m+1)$ by $(n+1)$ control points, $B_{j,m}(u)$ and $B_{k,n}(v)$ are polynomial functions defined as

$$B_{i,j} = C(j,i) u^i (1-u)^{j-1} \tag{3}$$

and the $C(j,i)$ represent the binomial coefficients

$$C(n,k) = \frac{n!}{k!(n-k)!} \tag{4}$$

b-spline [29] surfaces are similar to Bezier surfaces and defined as

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} N_{j,s}(u) N_{k,t}(v) \tag{5}$$

As before, vector values for $p_{j,k}$ specify the $(m+1)$ by $(n+1)$ control points. The parameters s and t control the order of continuity of the surface. The most important feature of the b-spline blending functions is that they are nonzero in only a portion of the range of the parameter. The b-spline blending functions of degree $k-1$ may be defined recursively as follows:

$$N_{k,1}(u) = \begin{cases} 1 & \text{if } u_k \le u \le u_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{k,t}(u) = \frac{u-u_k}{u_{k+t-1}-u_k} N_{k,t-1}(u) + \frac{u_{k+t}-u}{u_{k+t}-u_{k+1}} N_{k+1,t-1}$$

where

(6)

$$u_j = \begin{cases} 0 & \text{if } j < t \\ j-t+1 & \text{if } t \le j \le n \\ n-t+2 & \text{if } j \ge n \end{cases}$$

for values of j ranging from 0 to $n+1$.

Because the denominators can become zero, this formulation adopts the convention $0/0 = 0$.

Second order surfaces are also called quadric surfaces. They are defined by an expression with each term having coordinates to the second power.

$$Ax^2+2Bxy+2Cxz+2Dxw+Ey^2+2Fyz+2Gyw+Hz^2+2Izw+Jw^2 = 0 \tag{7}$$

This can be rewritten in matrix notation as:

$$\begin{bmatrix} x & y & z & w \end{bmatrix} \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0 \tag{8}$$

The algebraic properties of the symmetric matrix Q determines the shape of the surface. The various shapes are first distinguished by examining the signs of the four eigenvalues of Q.

Barr [1] has introduced superquadrics. These are different from the corresponding quadrics in the exponents of their terms. The superquadrics used in this work are defined using trigonometric parameterization:

Superellipsoid:

$$\begin{aligned} &x = cos^m(u)cos^n(v),\ \ y = sin^m(u)cos^n(v),\ \ z = tan^n(v) \\ &\text{so that } (x^{2/m} + y^{2/m})^{m/n} + z^{2/n} = 1 \end{aligned} \tag{9}$$

Superhyperboloid of one piece:

$$\begin{aligned} &x = cos^m(u)sec^n(v),\ \ y = sin^m(u)sec^n(v),\ \ z = tan^n(v) \\ &\text{so that } (x^{2/m} + y^{2/m})^{m/n} = 1 \end{aligned} \tag{10}$$

Supertoroid:

$$\begin{aligned} &x = cos^m(u)(k + cos^n(v)),\ \ y = sin^m(u)(k + cos^n(v)),\ \ z = sin^n(v) \\ &\text{so that } ((x^{2/m} + y^{2/m})^{m/2} - k)^{2/n} + z^{2/n} = 1 \end{aligned} \tag{11}$$

3.3 Geometric Calculations

In order to render images of surfaces it is necessary to:

- reposition and reorient them using linear transformations,
- clip them to the limits of view,
- find their representations in perspective
- calculate intersections between them
- determine whether parts of the surface are inherently hidden just by their orientation

For polygonal surfaces all of the above are straight forward. Surfaces may be repositioned and reoriented by applying shape-preserving transformations (rotations, translations, scaling, and mirroring) to the vertex coordinates. Clipping algorithms are also quite straight forward with algorithms available in literature [14,29] although variations keep on coming [25,34].

Perspective representations for polygons generally are found by transforming to a space where the view point is at the origin and the direction

of view lines lie along the z-axis then dividing x and y coordinates by the z coordinate for each vertex.

Calculating intersection between polygons is also simple. First, the plane equation for one of the polygons is found then the edges of the other polygon may be clipped against the plane of the first.

The plane equation of a polygon can be found by taking the cross product of vectors formed by any three non-colinear vertices of the polygon

$$[a\ b\ c] \leftarrow [p2 - p1] * [p3 - p1] \tag{12}$$

Using the normal vector, the plane equation is given by

$$a * x + b * y + c * z + d = 0 \tag{13}$$

Substituting values a, b, c and the coordinates of one of the vertices of the polygon, it is easy to find d.

If any coordinate that is not on the plane is substituted into the plane equation, a value proportional to the distance of that point from the plane is obtained. This fact is used to find the intersection point of an edge that pierces the plane.

Given two points, $p1$ and $p2$:

$$\begin{aligned} d1 &\leftarrow a * p1[x] + b * p1[y] + c * p1[z] + d \\ d2 &\leftarrow a * p2[x] + b * p2[y] + c * p2[z] + d \end{aligned} \tag{14}$$

If $d1$ and $d2$ have opposite signs then the intersection point q is given by:

$$\begin{aligned} \alpha &\leftarrow d1/(d1 - d2) \\ q[x] &\leftarrow p1[x] * (1.0 - \alpha) + p2[x] * \alpha \\ q[y] &\leftarrow p1[y] * (1.0 - \alpha) + p2[y] * \alpha \\ q[z] &\leftarrow p1[z] * (1.0 - \alpha) + p2[z] * \alpha \end{aligned} \tag{15}$$

If polygonal data is taken consistently so that the vertices of a polygon appear in clockwise order, then it is possible to determine that some faces are hidden from view just by their orientation. In particular, the z-coordinate of the vector normal to the plane can be used to determine hidden planes.

Since polygons can be non-planar, sometimes it is impossible to calculate the normals. A simple solution to this problem is to subdivide polygons into triangles. Scan conversion of non-planar polygons is also more difficult.

For nonlinear surfaces the above calculations are more complex. In order to reposition and reorient the same transformations can be applied. However, clipping algorithms are problematic. Nonlinear surface clipping algorithms work in two phases. First, a bounding volume is found. This bounding volume is used to detect whether the entire surface is inside, or outside the field of view. If neither of these conditions is true then, during the scan conversion, each pixel is checked individually whether it is inside the view or not. Another solution is to subdivide the surface until it is detected that the entire fragment is inside or outside.

For the perspective representation of nonlinear surfaces the surface can be evaluated at intervals and interpolated between the intervals.

Finding intersections between nonlinear surfaces pose problems similar to those of clipping. Therefore, when only the visual representation is important it is possible to scan convert two surfaces and compare pixel by pixel to determine the frontmost one. In the case that the edge of intersection is needed, it may be necessary to solve a systems of equations, or a subdivision process may be applied.

It is very difficult to determine if a nonlinear surface is visible or not. This subject has been generally ignored in the literature. Due to this and other reasons mentioned above, our system models objects by their closest approximation of planar triangles. The idea of using triangle approximations has also been suggested by Deering [9] and a specific VLSI device has been developed that can render one million triangles per second. Since this device was developed for the Sun Workstation environment, in the future our system can utilize its potentials.

3.4 Geometric Modeling Utility

In this system an interactive modeling utility was implemented to create geometric models. This utility is not very much elaborated, but a naive user can easily connect one triangle to others in 3-dimensions, so that a complicated surface can be formed.

Rubber band and dragging techniques are used for the user interface. The user may select an edge from the current set of triangles and add a new triangle connected to that edge. It is also possible to discard triangles from the scene. While the mouse is traced for x and y coordinates, z value can

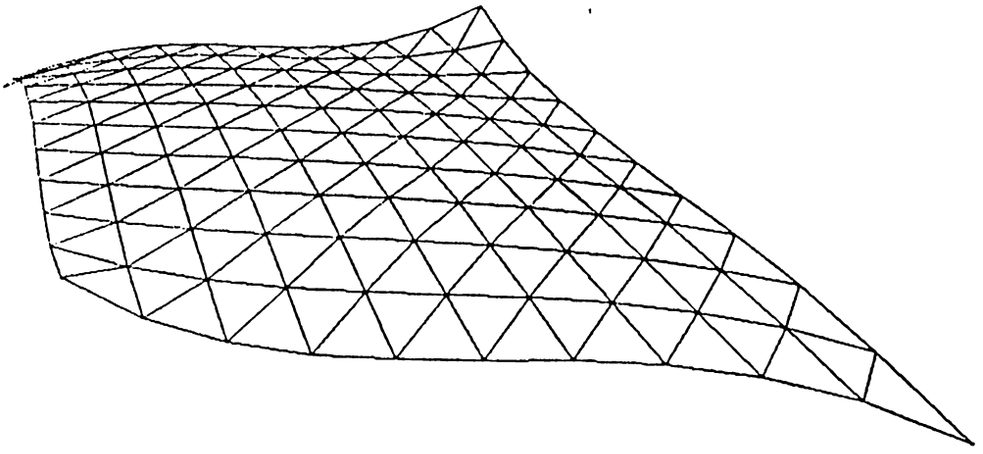

Figure 3.1: A Bezier surface.

be adjusted using a z-depth slider. The viewing point can be changed and the user can take different looks from different angles in order to visualize the scene properly. The viewing point is automatically adjusted when the depth is changed, that is, when we increase the depth we come closer to the objects found in deeper z-coordinates. The projection method can be chosen as either perspective or axonometric by the user.

Bezier or b-spline surfaces can also be generated by this utility by entering coordinates of their control points using the mouse or superquadric surfaces can be created by entering their related parameters via a pop-up window. Since a conversion to planar polygons takes place internally, it is possible to modify surfaces created by Bezier, b-spline or superquadric techniques manually. Wireframe drawings of a Bezier surface and a b-spline surface are shown in 3.1 and 3.2.

It is possible to see the hidden surface eliminated wireframe of the model before texturing begins. Depth sorting method is used for hidden surface elimination. Therefore, polygons are sorted with respect to their highest depth. Since all the polygons are triangles and all triangles are edge connected to each other, intersection calculations done for depth sorting hidden surface elimination algorithm are greately simplified. Wireframe drawings of a superellipsoid, a one piece superhyperboloid and a supertoroid are shown in Figures 3.3, 3.4, and 3.5.

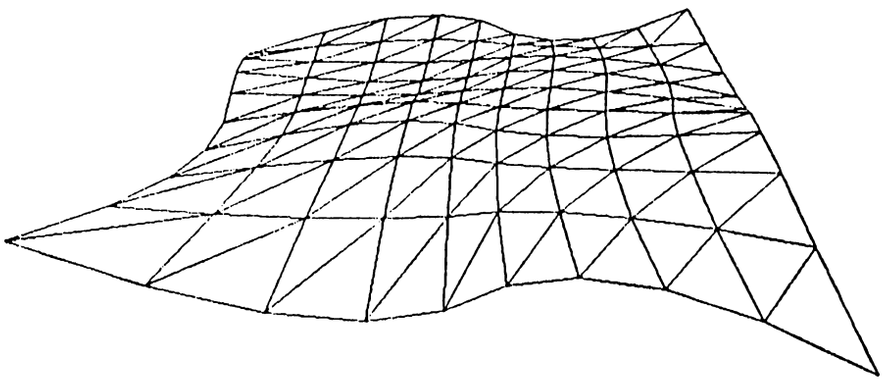

Figure 3.2: A b-spline surface.

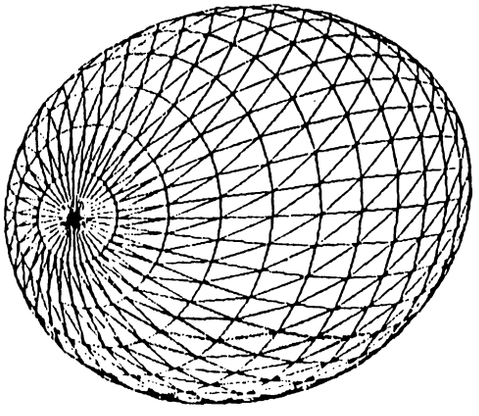

Figure 3.3: A hidden surface eliminated wireframe drawing of an superellipsoid.

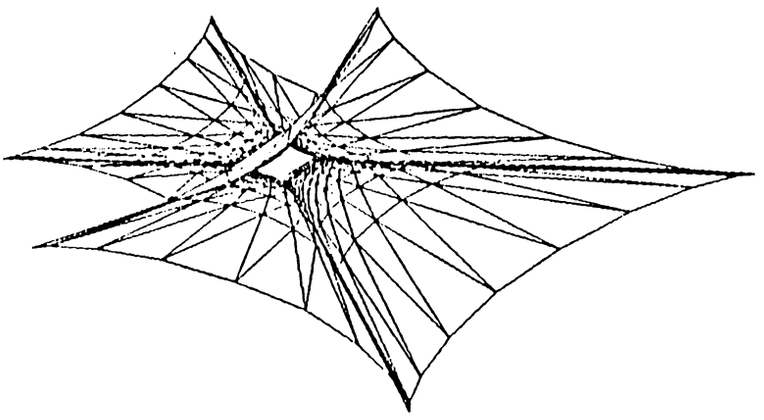

Figure 3.4: A hidden surface eliminated wireframe drawing of a one piece superhyperboloid.

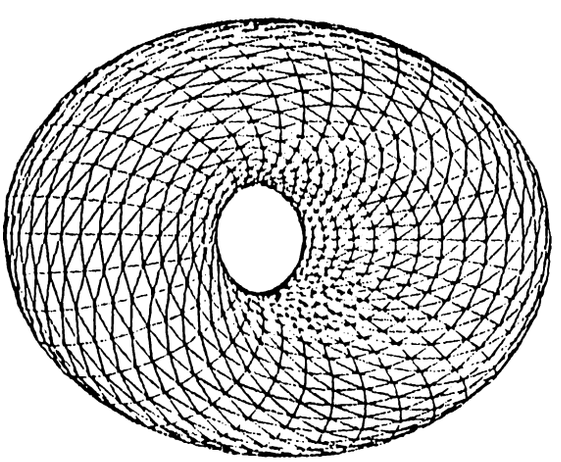

Figure 3.5: A hidden surface eliminated wireframe drawing of a supertoroid.

4. GEOMETRIC MAPPING

Texture mapping is a combination of geometric mapping and filtering. First, a procedure to calculate the corresponding points in the texture space and the object space is needed. Since the transformation from the object space to the screen space is usually performed within this procedure one mapping is defined from the texture to the screen. This calculation can be done in different orders: screen order, texture order and the two-pass methods. In the screen order, as each picture element in the screen is scanned, the texture coordinates to be mapped on that picture element are calculated. The texture order is just the opposite: the texture is scanned and the screen coordinates are calculated. The two-pass methods decompose one 2-dimensional to 2-dimensional mapping to two 1-dimensional to 1-dimensional mapping [35].

To map a texture onto a surface, different techniques can be employed. The first is the parameterization of the surface as it is shown in Figure 4.1, in terms of u and v. Parameters u and v are then used either as the input variables into some texture generating function, or as the coordinates of a value within a texture image . The surface parameterization may cause the arbitrary positioning of the texture map. The parameterization is highly dependent on the way the surfaces are defined. For example, it can be done very naturally for parametrically defined surfaces, whereas it is not so for other types of surfaces such as quadrics.

In planar polygons parameterization is linear, whereas for nonplanar polygons it is not. Since the solution of nonlinear equations is harder and more time consuming than that of linear equations, planar polygons have been preferred. Details of parameterization are explained in [17].

One problem with parameterization is to ensure apparent continuity of two dimensional textures applied to complex surfaces. The texture areas assigned to the neighbors at opposite sides of a polygon may be radically

different. Mapping a texture to an arbitrarily complex shape without discontinuities is a very hard problem and it has been solved by ad-hoc approaches up to now.

Some solution methods [5] were proposed, but they are not generally applicable to arbitrary shapes and to arbitrary geometric models. Therefore, surfaces textured by parameterization do not contain very much complexities. Indeed mapping a 2-dimensional texture to an arbitrarily convex 2-dimensional polygon is itself a problem to be dealt. In [13] a solution was proposed. They developed a technique to construct a continuous bijective map from a polygonal texture space to an arbitrary convex polygon.

In this work, any arbitrarily shaped geometric model can be textured without discontinuities between the adjacent polygons. However it is not claimed that this solution is perfect and has no internal flaws in it. The mapped texture is compressed and stretched at appropriate locations in order to cover the surface. Algorithmically there is no exact way of formulating these distortions. Particular formulas for a sphere or for a cube can be applied. However, for a shape that has many holes, it is not so obvious.

Similarly, when one covers an object with a piece of paper, in order to cover the object properly, he folds and wrinkles the paper in the way he wishes. When mapping texture onto an object, it is not folded nor wrinkled, instead it is compressed or stretched, in order to make it take the shape of the object and again this is done arbitrarily provided that the texture is continuously mapped all over the surface.

The visual appearance is important rather than some physical or mathematical laws in texture mapping. If an ordinary observer perceives the shape behind the texture then the solution can be accepted as satisfactory. The shading effects are used to increase the visual realism.

Normal vector intersection is another technique for the geometric mapping. A map template is suspended above the surface to be mapped. The template may be a surface such as a rectangle or a sphere. The texture value for a particular point on the surface is determined by intersecting the normal vector at that point with the map template, such as in Figure 4.2. The point of intersection provides a u, v pair then to be used in finding the texture value. This indirection sometimes may distort the map in unpleasant fashions.

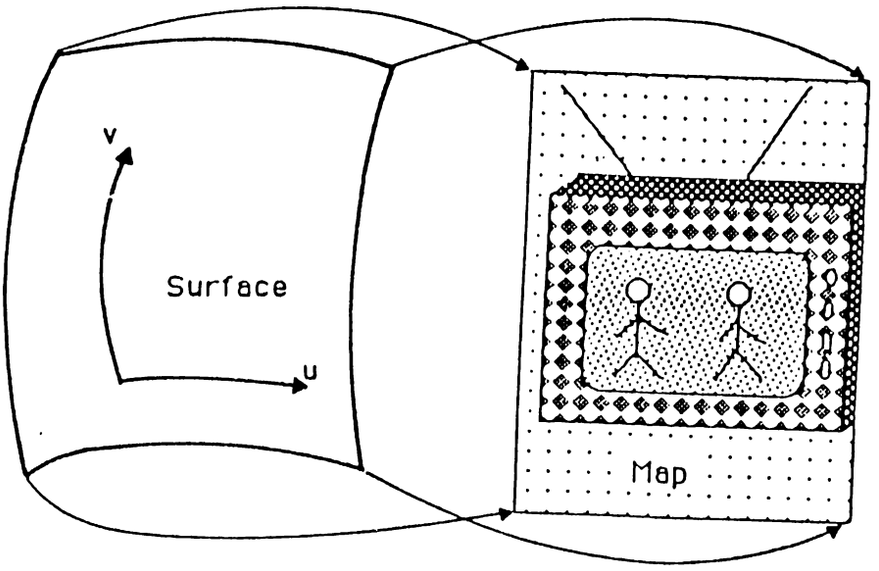

Figure 4.1: Geometric mapping by surface parameterization.

Texture mapping by surface normals for planar polygons is not applicable, since slopes between the polygons change sharply. A way to overcome this difficulty may be surface normal interpolation as in the case of Phong [29] shading. Mapping by surface normals is a good way for environment mapping. Environment mapping is a form of texture mapping wherein the texture applied to 3-dimensional surfaces is represented in an environment map [23]. The diffuse and specular illumination impinging on a region of a surface can be found by texture filtering regions of the environment map with a space-variant filter.

Environment mapping is superior to ray-tracing, in the sense that, ray tracing requires integration over parts of the 3-dimensional environment, while environment mapping simplifies the problem by treating the environment as a 2-dimensional projection. However, this simplification prevents the simulation of phonemena such as light effects created locally like shadows. Therefore, environment mapping is more effective when the local environment does not affect surface shading very much.

Diffuse illumination at a surface point comes from the hemisphere of the world centered on the surface normal, and it can be found by filtering the region of the environment map corresponding to this hemisphere. Filtering should be done according to Lambert's Law, which states that the illumination coming from a point on the hemisphere should be weighted by the cosine of the angle between the direction of that point and the surface normal [29].

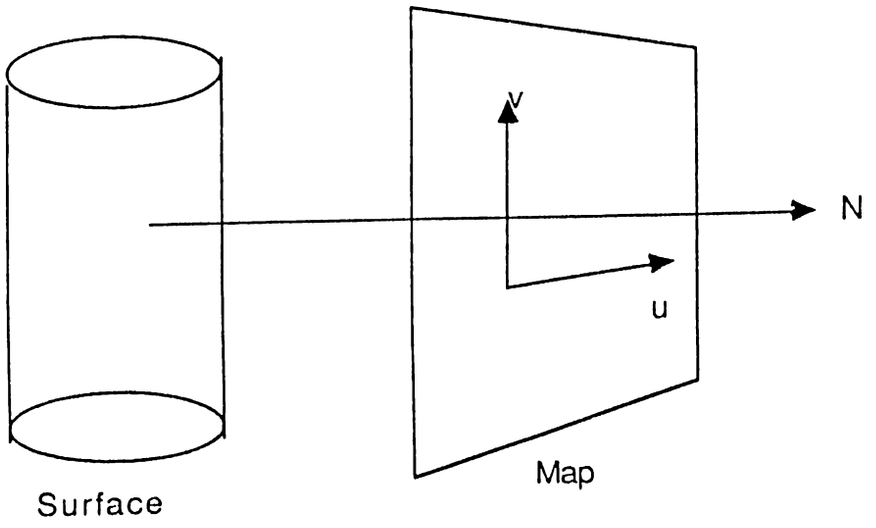

Figure 4.2: Geometric mapping by normal vector intersection.

Specular illumination can also be computed by filtering an appropriate region of the environment. This region is dependent on the viewpoint, that is, it is defined by the rays emanating from the viewpoint, reflected on the surface with equal coming and going angles with respect to the surface normal, and impinging on the environment map. This region is a quadrilateral if a pixel is assumed to be square, otherwise if a pixel is assumed to be a circle the filtered area is an ellipse. Assuming the pixel as a circle gives better results, but it is more costly.

Another technique of solid-texturing utilize the concept of a mapping template, but in a slightly different manner. The 2-dimensional mapping template is extruded through space in a direction normal to the map, thus producing a 3-dimensional mapped volume. The texture value at a point on a surface is determined by finding the position of that point within the solid map extrusion. This position can be represented by u, v and w, where w is the distance along the axis of extrusion, as in Figure 4.3.

A solid texture function for a color parameter ρ is simply a texture function defined at the points on a surface in terms of their 3-space coordinates [30]: $\rho(x, y, z)$.

This definition makes it unnecessary to be concerned about the shape of the surface being textured. Solid texture functions can be defined periodically like 2-dimensional texture functions, therefore the location of the object that is textured is not important.

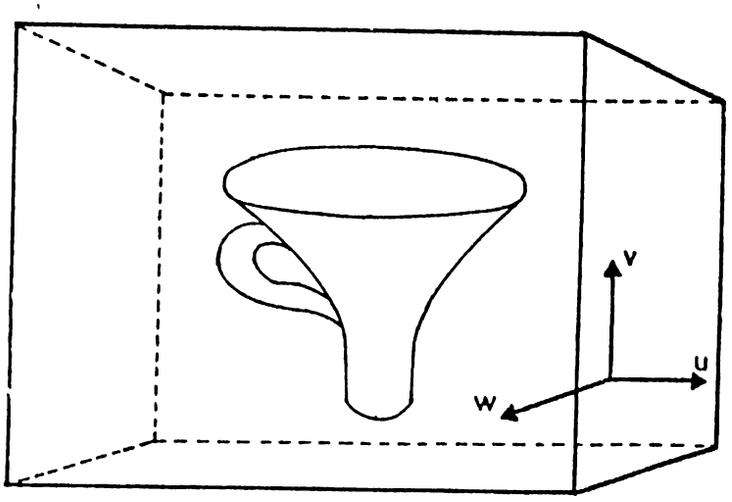

Figure 4.3: Solid texturing.

Solid texturing has advantages in rendering objects whose surface texture arises from their internal structure. Since the texture is defined in 3-dimensions, the texture covers the surface more realistically than mapped 2-dimensional textures. Solid texturing eliminates the aliasing problems that arise from the highly compressed surface coordinate near the poles of a sphere or in regions of tight curvature on some parametric surfaces.

Solid texturing can be easily applied to arbitrarily complex surfaces. Using 2-dimensional texturing techniques, each patch of a complex surface can be textured easily. However, it is very hard to map a single texture over the entire complex surface in a coherent fashion without introducing discontinuities. In 2-dimensional texture mapping texture space is partitioned into regions to be applied to the various patches that make the surface. It is not an easy task to prevent texture discontinuities between the adjacent patches. As the number of patches grows and their arrangement becomes less regular, 2-dimensional texture mapping becomes more awkward. Solid texturing can be applied to every kind of surface without dealing with individual patches.

Another advantage of solid texturing is that it is much more applicable to soft objects that are defined by a number of key points in space. Soft objects have been described in [38]. The key points define a skeleton. Each point has an effect range and the surface of an object is determined by this range. Therefore, a single point represents a sphere and collection of these produces a shape that is a blending of the spheres. When the key points move independently the object changes its shape and topology. For such

objects using 2 dimensional maps seem to be inapplicable. The texture space is connected to a coordinate system the object is defined, since the texture of an object moving through space may change inconsistently.

The most striking problem with solid texturing is the generation of textures. Digitizing solid textures is not simple and the storage to be allocated for such a texture is tremendously large, e.g, for a $512 * 512 * 512$ resolution 134MB is necessary, assuming one byte per texel(texture element). The only possible way is to use synthetic textures defined procedurally. A function of three variables that return a color value can be used to define interesting textures. However, it is very hard to define many artistic and natural textures procedurally. Due to these problems, 2-dimensional texture mapping has been adopted for this work and the remaining parts will discuss how the problems associated with 2-dimensional textures are solved.

5. FILTERING

5.1 Aliasing

If each point on the screen is mapped through pure geometrical calculations some problems occur. First, two neighbor points on the screen can be mapped by two widely apart points on the texture. The reverse is also true, that is, two neighbor points in the texture can be mapped to distant points on the object. This is a common case in warped surfaces resulting in the effect called aliasing with some unrealistically sharp transitions and stair cases in the image.

Taking discrete measurements of a signal at an inadequate number of regular intervals causes the effect called "aliasing". An inadequate sampling interval when synthesizing digital images causes small errors in representing the positions of the edges which characterize the image. The inadequate sampling is mostly caused by the equipment we have. In other words, the positions of details in an image are forced to coincide exactly with the positions of the individual pixels.

As it is seen in Figure 5.1, the set of samples from the high-frequency signal is the same as the set from the much lower frequency signal. Here the two different signals are called aliases of each other.

A fundamental signal-processing theorem, the Sampling Theorem, states that the frequency at which uniformly spaced samples of a continuous one-dimensional signal are taken should be greater than twice the maximum frequency present within the signal. If this rule is not obeyed then, it is impossible to reconstruct unambiguously the original signal from the samples. Signal frequencies that are greater than half the sampling frequency cannot be distinguished from lower alias frequencies.

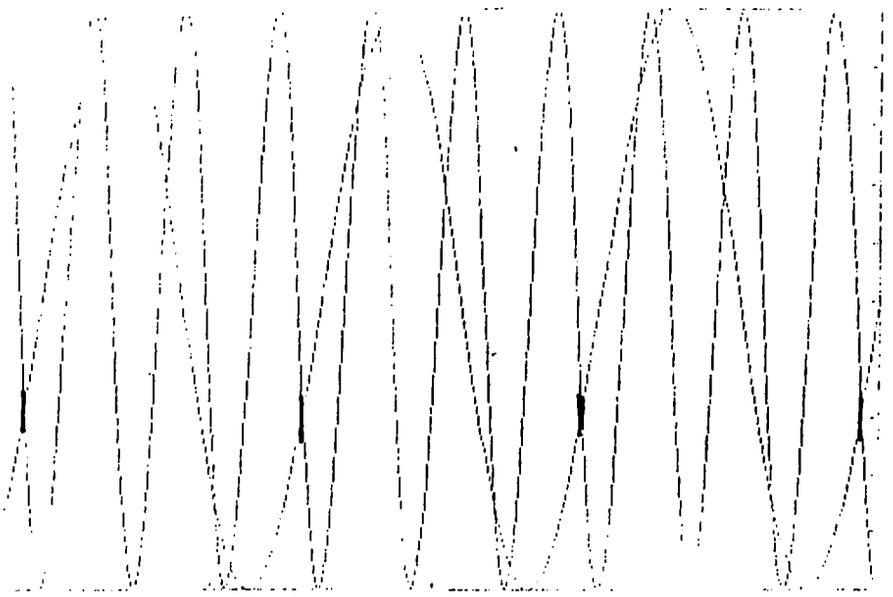

Figure 5.1: Aliasing of sampled signals.

The extension of the one-dimensional sampling theorem into two dimensions is straight forward: the x and y sampling frequencies should be greater than twice the maximum x and y spatial frequencies present in the picture being sampled [20].

It is possible to avoid artifacts in two ways:

(1) Take samples at a non-uniform spacing. This approach called stochastic sampling, is one of the recent issues receiving considerable attention [10,24]. However, it is not yet shown that it is an effective technique for other than ray-tracing applications.

(2) Take samples at a regular spacing, but obey the Sampling Theorem. This is the traditional approach.

In order to obey the Sampling Theorem one of the following conditions should be satisfied:

(a) Increase the sampling frequency to greater than twice the maximum frequency present within the signal ,

(b) Filter the signal before sampling to remove frequency components greater than one half the sampling frequency .

Supersampling does not provide a general solution to aliasing because there is no restriction on the frequency of some signals and it is more expensive

than prefiltering.

5.2 Filtering Techniques

Convolution is the fundamental filtering operation. A weighting function or kernel is passed over the input signal and a weighting average is computed for each output sample.

Direct convolution which is the most straight forward filtering method is very expensive for wide kernels. In this method a weighted average is computed anew for each output sample. When the shape of the kernel filter does not change as it moves across the signal, the filter is called space invariant. For space invariant filtering, the signal and kernel are transformed to the frequency domain using a FFT, these are multiplied together, and an inverse FFT is computed [4].

Fourier series filtering has a restricted use since it is applicable when the texture is represented as a Fourier series. The low-pass filtering is applied to its spectrum. Otherwise it is first necessary to convert the texture into frequency space causing an overhead.

Catmull developed a filter that computes the unweighted average of the texture space elements in the quadrilateral that is the preimage of a single pixel assumed to be a square [16].

The filter by Blinn and Newell are implemented via a weighting function that takes the form of a square pyramid with a base width of 2x2 picture elements. The 2x2 region surrounding the given picture element is inverse mapped to the corresponding quadrilateral in the texture space element. The values in the texture pattern within the quadrilateral are weighted by a pyramid distorted to fit the quadrilateral and summed [2].

The filter developed by Feibush, Levoy, and Cook is more elaborate. First the filter function is centered on the pixel, then the corresponding quadrilateral region of texture space is found, and a weighted average of texture pixels is formed [12].

The texture filter proposed by Gangnet, Perny, and Coueignoux is quite similar to the method of Feibush et al. However pixels are assumed to be circular and their preimages are ellipses. The texture values are weighted by

a truncated *sinc* two pixels wide in screen space and summed [18].

The elliptical weighted average filter by Greene and Heckbert is similar to Gangnet's method in that it assumes circular pixels that map to arbitrarily oriented ellipses, and it is like Feibush's method because the filter is stored in a lookup table [22].

Many applications demand a space variant filter, the kernel of which changes with the position. Texture mapping and nonlinear image warps are such applications. For proper antialiasing it is necessary to filter the texture area corresponding to each screen pixel [12]. Since these texture areas may be arbitrarily large, using direct convolution accurate filtering of such pixels can be prohibitively expensive. In order to reduce the cost to a reasonable level some different techniques are necessary.

The generally accepted solution is signal prefiltering. Two different structures have been proposed: pyramids and integrated arrays.

Pyramid methods are common in image synthesis for texture filtering [11,37]. This method restricts the filtered areas to be squares, otherwise filtering of rectangular areas is inconvenient.

The integrated array prefiltering is appropriate for filtering of rectangular areas [6,16,31].

In pyramid data structures texture is stored in lower resolutions. A pyramid is formed with 1 by 1, 2 by 2, 4 by 4 ... squares with powers of two. Any shape in the texture area can be subdivided into squares and the values of these squares are used to speed up the process, since they were calculated before the actual mapping. This technique was first proposed by Catmull [17].

In this study the summed area tables proposed in [6] are used. In this method the texture array is preintegrated in such a way that each entry in the summed area table gives the sum of all texture samples contained in the rectangle defined by itself and the lower left corner of the texture array. Hence, the sum of texture samples in any rectangle is the result of only three additions. The following formulation, and Figure 5.2 show this computation:

$$T[x_r, y_t] - T[x_r, y_b] - T[x_l, y_t] + T[x_l, y_b]$$

where

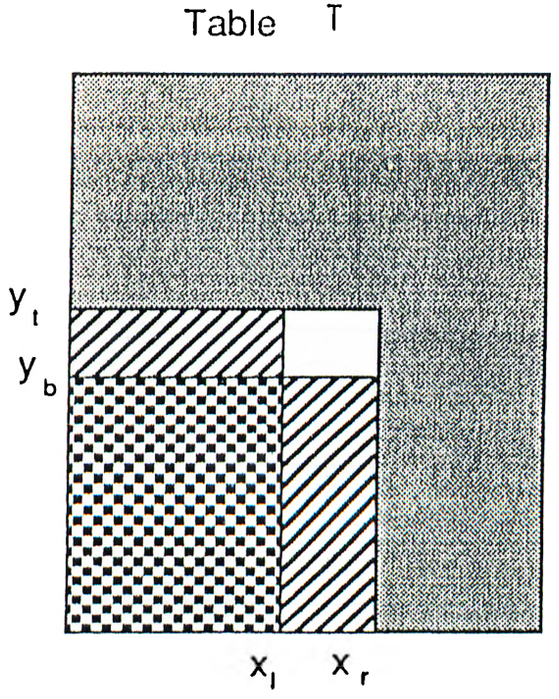

Figure 5.2: Calculation by a summed area table.

x_l : the left x_coordinate of the rectangle
x_r : the right x_coordinate of the rectangle
y_b : the bottom y_coordinate of the rectangle
y_t : the top y_coordinate of the rectangle
$T[x, y]$: the value of the summed area table at location (x, y)

This technique is generalized by increasing the number of preintegrations and filters of better quality can be achieved. The drawback of this is the abundance of storage allocated. For an image size of 512 by 512 and 256 levels of intensity at a machine using a color lookup table , we have to allocate $(9 + 9 + 8 = 26$ bits$) = 4$bytes for each summed area entry and therefore 4 times more than the original image size. Since we should store 3 different tables for an RGB machine the storage allocated for the tables is 12 times of the original one. If the number of integrations is increased than it is not possible to use 32 bit integers. The use of floating numbers decrease the performance relatively.

Filtering by repeated integration which is a generalization of Crow's summed area table is a space variant filtering technique providing constant cost. Figure 5.3 shows the shapes of some low order repeated integration filters. Theoretical bases of this method are given in [16].

Since the areas to be filtered are not necessarily rectangles, sometimes we include unrelated areas. Approximation of a quadrilateral by a rectangle is shown in Figure 5.4. This has surely a negative effect on the performance

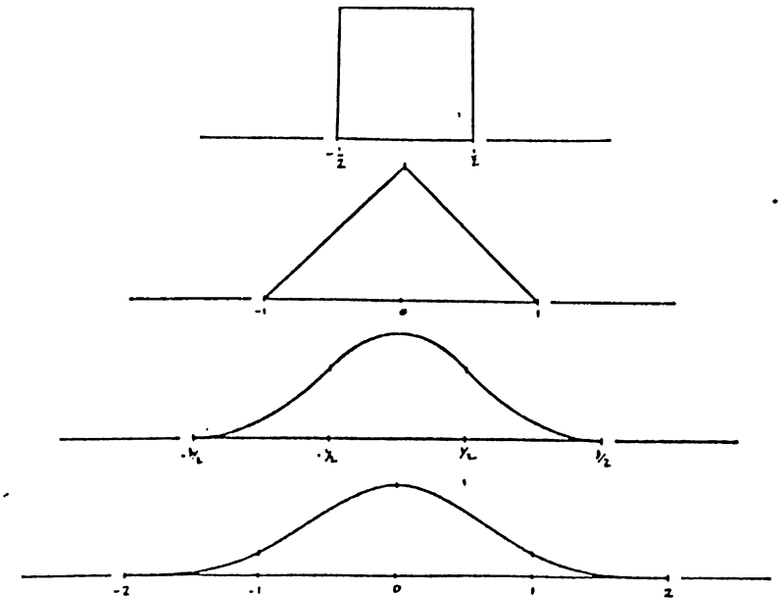

Figure 5.3: Repeated integration filters of order 1-4.

of this technique. In order to increase the performance some approximation techniques are described by Glassner in [19]. He proposed to add or subtract rectangles of appropriate sizes to increase the quality of filtering.

5.3 Chromatic Image Filtering

Filtering of chromatic images has some problems that do not exist in achromatic image filtering. RGB images can be filtered by taking each component of the color vector independently. However, if this way is chosen then, some important aspects of the colors are not taken into account. One problem is that any unit change in the amount of one component is not perceived as equivalently noticeable color shift to an observer. Some experimental results indicate that a human observer is most sensitive to color shifts in the blue, moderately sensitive to color shifts in the red, and least sensitive to color shifts in the green. Therefore, it may be a good solution to use a color coordinate system that has the property of giving equivalently noticeable color shifts for the unit changes in the coordinate system. In 1960 the CIE (Commission Internationale de l'Eclairage- the International Commission on Illumination) adopted a coordinate system, called the Uniform Chromacity Scale (UCS), in which equal changes in the chromacity coordinates result in just noticeable changes in hue and saturation to a good approximation. The UCS color coordinate system is linear transformation of the RGB coordinate

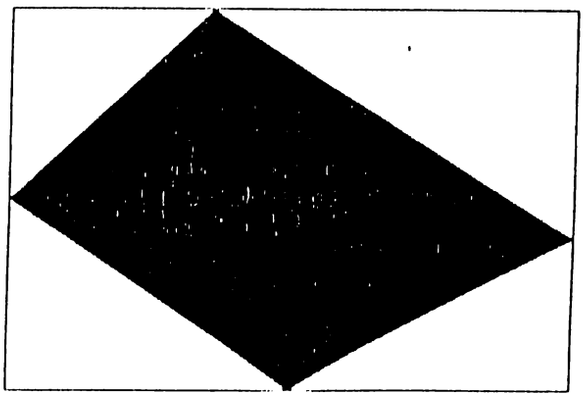

Figure 5.4: Approximating a quadrilateral by a rectangle.

system. The chromacity coordinates of two systems are related by

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} 0.405 & 0.116 & 0.133 \\ 0.299 & 0.587 & 0.114 \\ 0.145 & 0.827 & 0.627 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

The $U^*V^*W^*$ coordinate system is an extension of the UVW coordinate system in an attempt to obtain a color solid for which unit shifts in luminance and chrominance are uniformly perceptible. The $U^*V^*W^*$ coordinates are defined as

$$\begin{aligned} U^* &= 13W^*(u - u_0) \\ V^* &= 13W^*(v - v_0) \\ W^* &= 25(100V)^{1/3} - 17 \end{aligned} \tag{2}$$

where

$$u = \frac{U}{U + V + W}, \quad v = \frac{V}{U + V + W}$$

and u_0, v_0 chromacities of reference white ($u_0 = 0.201$, $v_0 = 0.307$).

Conversion from one color coordinate system to another adds some extra computation, while reducing deficiencies emanating from the abnormal transitions on the color edges, by adjusting the hue, saturation and luminance contributions realistically. Since such a conversion in the target machine prevents interactive usage, the chromatic image filtering problem is solved without a coordinate conversion. However, as the algorithm is improved or the speed of the target machine increases adding such a conversion to the code can be feasible.

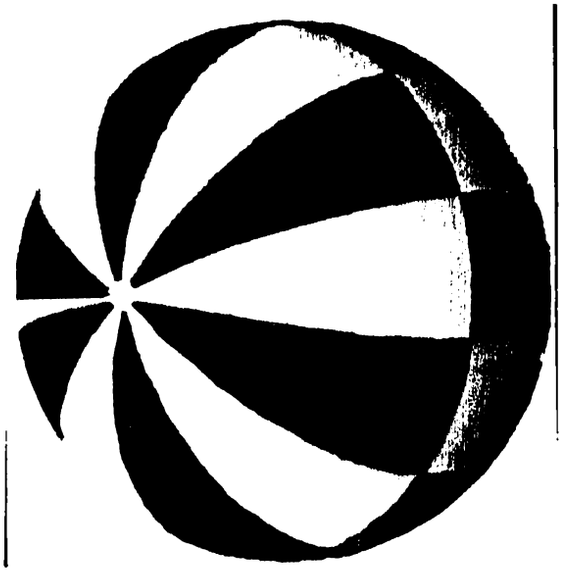

Figure 5.5: A checker-board mapped superellipsoid.

5.4 Implementation

In this implementation the screen scanning scheme is used, that is, as the screen is scanned line by line the corresponding preimages are found, filtered and, given as the color value for the pixel scanned. For each (x, y) pixel scanned a (u, v) coordinate that defines a corner of the preimage rectangle coordinate is calculated. The other corners are defined by the texture space coordinates mapped by screen coordinates $(x-1, y)$, $(x, y-1)$, and $(x-1, y-1)$. These four coordinates denote a quadrilateral. This quadrilateral is approximated by a rectangle and the average value of the rectangle is calculated by the summed area table method as explained above. The sizes of rectangles may change considerably depending on the surface slope. Since surfaces are always restricted to be planar polygons in the implementation the textured area sizes are same through each polygon scanned; some minute differences come as a result of truncations. The checker-board mapped superellipsoid, and a supertoroid are shown in Figures 5.5 and 5.6. For the wireframe representations of those surfaces refer to Chapter 3.

Besides the textures, real objects have some fluctuations on their surfaces such as bumps. It is also possible to map such bumps onto the smooth surfaces. This is done by changing intensity values artificially, that is intensity values for scanned points on a surface are altered periodically or randomly. In this implementation it is altered periodically and the sizes of the bumps are given by the user. A bump mapped supertoroid, the real texture mapped

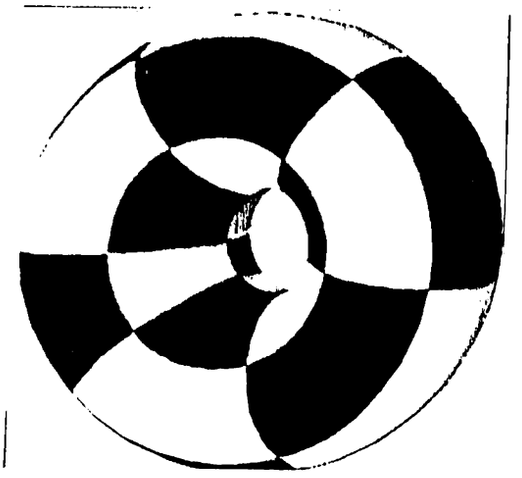

Figure 5.6: A checker-board mapped supertoroid.

onto it and the wireframe of the supertoroid are shown in Figure 5.7.

In order to get some interesting effects, more than one texture can be mapped on a surface by giving some transparency values between 0.0 and 1.0 to the textures. Apparently, if the transparency value for a texture is 0.0, previously mapped texture is not seen under the currently mapped texture, and if the transparency value is 1.0 the previous texture stays without being altered, any value between 0.0 and 1.0 blends the previously mapped color value and the current one accordingly. A transparently texture mapped object and the two textures are shown in Figure 5.8.

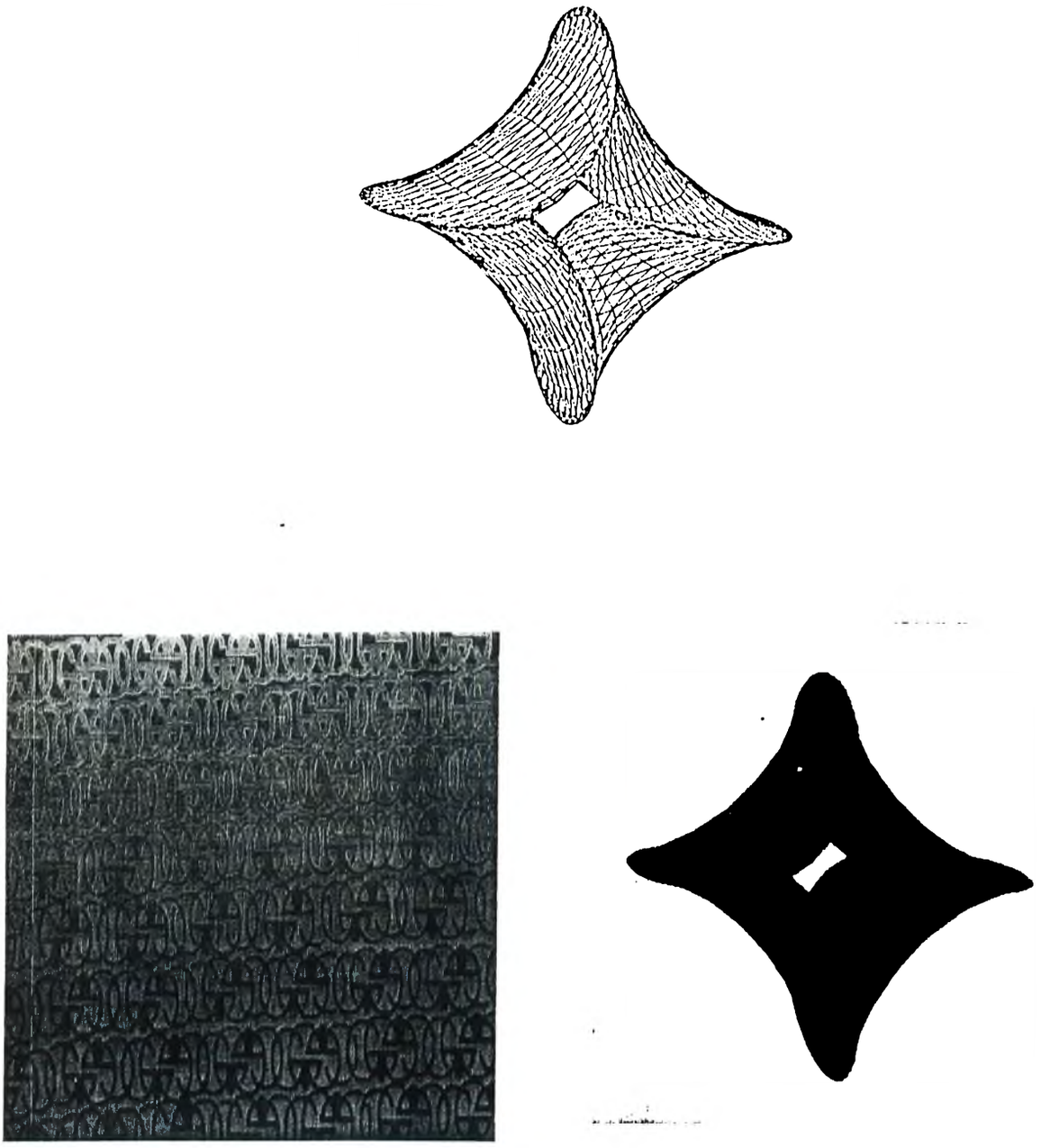

Figure 5.7: The texture, the geometric model and the bumpy texture mapped object.

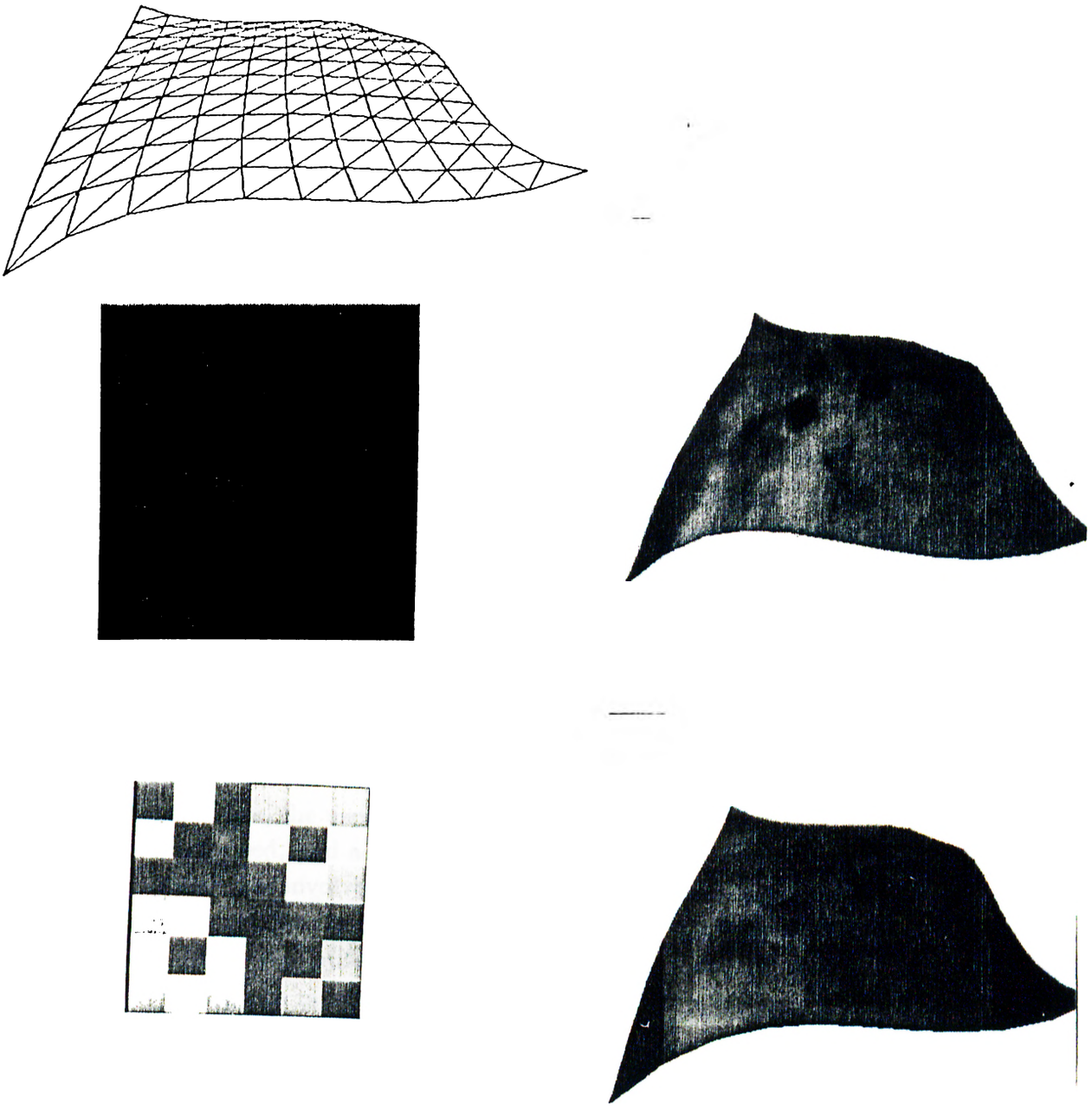

Figure 5.8: Three stages in transparently mapping textures to an object, and the textures mapped.

6. CONCLUSION

The texture mapping practice we had has shown us that it is a very effective way for creating realistic and attractive computer images. It has research potentials to improve the quality while keeping the costs at a reasonable level. First of them is surely filtering. Especially filtering of chromatic images needs attention. Indeed some experimental study seems necessary, besides the CIE standards and knowledge about color developed upto now.

The other issue is to make this technique more usable in our daily life applications such as CAD, CAM, computer art or education with visual simulations. The user interface part becomes very important in this respect. For example, the user may be given the chance to select between the quality and the speed, and adjust them accordingly depending on the nature of application he is involved. The paint brush system can also be enhanced to include a facility for the creation of textures mathematically. Such attempts together with an elaborated geometric modeling tool make our system an effective tool for texture mapping.

Besides the surface properties of objects that are modeled by using computers, the effect of light on these surfaces can be simulated properly and interestingly by a cost effective technique. Environment mapping which is a natural extension of texture mapping seems applicable for this purpose.

REFERENCES

[1] Barr, A. H., Superquadrics and Angle-Preserving Transformations *IEEE CG&A*, Vol. 1, No. 1, January 1981, pp 11-23.

[2] Blinn, J. F., and Newell, M.E. Texture and Reflection in Computer Generated Images, *Comm. ACM*, Vol. 19, No. 10, October 1976, pp 542- 547.

[3] Blinn, J. F., Simulation of Wrinkled Surfaces, *Computer Graphics (Proc. SIGGRAPH 78)*, Vol. 12, No. 3, pp 286-292.

[4] Brigham, E. O., *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[5] Crow, F. C., A More Flexible Image Generation Environment, *Computer Graphics (Proc. SIGGRAPH 82)*, Vol. 16, No. 3, pp 9-18.

[6] Crow, F. C., Summed-Area Tables for Texture Mapping, *Computer Graphics (Proc. SIGGRAPH 84)*, Vol. 18, No. 3, pp 207-212.

[7] Crow, F. C., Advanced Image Synthesis - Anti-Aliasing, *Advances In Computer Graphics*, Enderle, G., Grave, M., Lillehagen, F., Eds., Springer-Verlag, Air-la-Ville, 1986, pp 419-440.

[8] Crow, F. C., Advanced Image Synthesis - Surfaces, *Advances In Computer Graphics*, Enderle, G., Grave, M., Lillehagen, F., Eds., Springer-Verlag, Air-la-Ville, 1986, pp 457-467.

[9] Deering, M., Winner, S., Schediwy, B., Duffy, C., Hunt, N., The Triangle Processor and Normal Vector Shader: a VLSI System for High Performance Graphics, *Computer Graphics (Proc. SIGGRAPH 88)*, Vol. 22, No. 4, pp 21-30.

[10] Dippé, M. A. Z., Wold, E. H., Antialiasing Through Stochastic Sampling. *Computer Graphics (Proc. SIGGRAPH 85)*, Vol. 19, No. 3, pp 69-78.

[11] Dungan, W. Jr., Stenger, A., and Sutty, G., Texture tile considerations for raster graphics, *Computer Graphics (Proc. SIGGRAPH 78)* Vol. 12, No. 3, pp. 130-134.

[12] Feibush, E. A., Levoy, M., and Cook, R. L., Synthetic Texturing Using Digital Filters, *Computer Graphics (Proc. SIGGRAPH 80)*, Vol. 14, No. 3, pp. 294-301.

[13] Fiume, E., Fournier, A., Canale, V., Comformal Texture Mapping, *Eurographics (Proc. EG 87)*, pp 53-64.

[14] Foley, J. D., and van Dam, A., *Principles of Interactive Computer Graphics*, Addison-Wesley, Reading, Mass., 1982.

[15] Hawkins, J. K., Textural Properties for Pattern Recognition of Objects, *Picture Processing and Psychpictorics.*, B. C. Lipkin and A.Rosenfeld, Eds., Academic Press, New York, 1970, pp 347-370.

[16] Heckbert, P. S., Filtering by Repeated Integration, *Computer Graphics (Proc. SIGGRAPH 86)*, Vol. 20, No. 4, pp. 315-321.

[17] Heckbert, P. S., Survey of Texture Mapping, *IEEE CG&A* Vol. 6, No. 11, November 1986, pp 56-67.

[18] Gangnet, M., Perny, D., and Coueignoux, P., Perspective Mapping of Planar Textures, *Eurographics 82*, pp 57-71.

[19] Glassner, A. S., Adaptive Precision in Texture Mapping, *Computer Graphics (Proc. SIGGRAPH 86)*, Vol. 20, No. 4, pp 297-306.

[20] Gonzales, R. C., *Digital Image Processing*, Addison-Wesley, Reading, Mass., 1977.

[21] Gouraud, H., Continuous Shading of Curved Surfaces, *IEEE Transactions on Computers*, Vol. 20, No. 6, June 1971, pp 623-628.

[22] Greene, N., and Heckbert, P. S., Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter, *IEEE CG&A*, Vol. 6, No. 6, June 1986, pp.21-27.

[23] Greene, N., Environment Mapping and Other Applications of World Projections, *IEEE CG&A*, Vol. 6, No. 11, November, 1986, pp.21-29.

[24] Lee, M. E., Redner, R. A., Uselton S. P., Statistically Optimized Sampling for Distributed Ray Tracing, *Computer Graphics (Proc. SIGGRAPH 85)*, Vol. 19, No. 3, pp 61-65.

[25] Liang, Y., Barsky, B., An Analysis and Algorithm for Polygon Clipping, *Comm. ACM 26*, Vol. 11, November 1983, pp 868-877.

[26] Lobb, R. J., Antialiasing of Polygons with a Weighted Filter, *Computer Graphics 1987*, Tosiyasu, L. K., Ed., Springer Verlag, Tokyo, 1987, pp 107-127.

[27] Lorig, G., Advanced Image Synthesis - Shading, *Advances In Computer Graphics.*, Enderle, G., Grave, M., Lillehagen, F., Eds., Springer-Verlag, Air-la-Ville, 1986, pp 441-456.

[28] Miller, G. S., Hoffman, C. R., Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments, *SIGGRAPH 84: Advanced Computer Graphics Animation Seminar Notes*, July 1984.

[29] Newman, W. M., Sproull, R. F., Principles of Interactive Computer Graphics, McGraw-Hill, New-York, 1973.

[30] Peachey, D. R., Solid Texturing of Complex Surfaces, *Computer Graphics (Proc. SIGGRAPH 85)*, Vol. 19, No. 3, pp 279-286.

[31] Perlin, K., Course Notes, *SIGGRAPH 85: State of the Art in Image Synthesis seminar notes*, July 1985.

[32] Pickett, R. M., Visual Analysis of Texture in Detection and Recognition of Objects, *Picture Processing and Psychpictorics.*, B. C. Lipkin and A.Rosenfeld, Eds., Academic Press, New York, 1970, pp 289-308.

[33] Pratt, W. K., *Digital Image Processing*, Wiley-Interscience , New York, 1978.

[34] Rogers, D. F., Ryback, L. M., On an Efficient General Line-Clipping Algorithm, *IEEE CG&A 5*, Vol.5, No. 1, January 1985, pp 82-86.

[35] Smith, A. R., Planar 2-Pass Texture Mapping and Warping, *Computer Graphics (Proc. SIGGRAPH 87)*, Vol. 21, No. 4, pp 263-272.

[36] Sun Microsystems, *Sun View Programmer's Guide*, Mountain View, CA 1986.

[37] Williams, L., Pyramidial Parametrics, *Computer Graphics (Proc. SIGGRAPH 83)*, Vol. 17, No. 3, July 1983, pp 1-11.

[38] Wyvill, G., McPheteers, C., Wyvill, B. L. M., Data Structure For Soft Objects, *The Visual Computer*, Vol. 2, No. 4., pp 227-234.

A. THE USER'S MANUAL

The implementation of the system was written to run in SunView environment of Sun Workstations[1]. It has the name "txtmp". The code was written in C. A user friendly interface is provided. Both the geometric model and the texture can be created by the user interactively.

A.1 Panel Item Descriptions

As it is seen in Figure A.1, there is a set of panel items to enter necessary parameters related to geometric modeling, texture mapping and texture preparation.

The **REDISPLAY** button is useful especially when the model held is not displayed properly. This is the case when z-coordinate is changed, the screen is automatically cleared and it may be redisplayed according to new z coordinate.

The **SAVE** button saves the current geometric model as a file on the disk.

The **LOAD** button loads a geometric model that has been saved as a file on the disk.

The **CLEAR** button clears the data structure holding the geometric model and clears the screen. When the current geometric model is to be discarded and a new model is to be developed, this button is pressed.

The **T-MAP** button maps the texture that has been stored as an integrated table to the current geometric model.

[1]SunView and Sun Workstation are registered trademarks of Sun Microsystems.

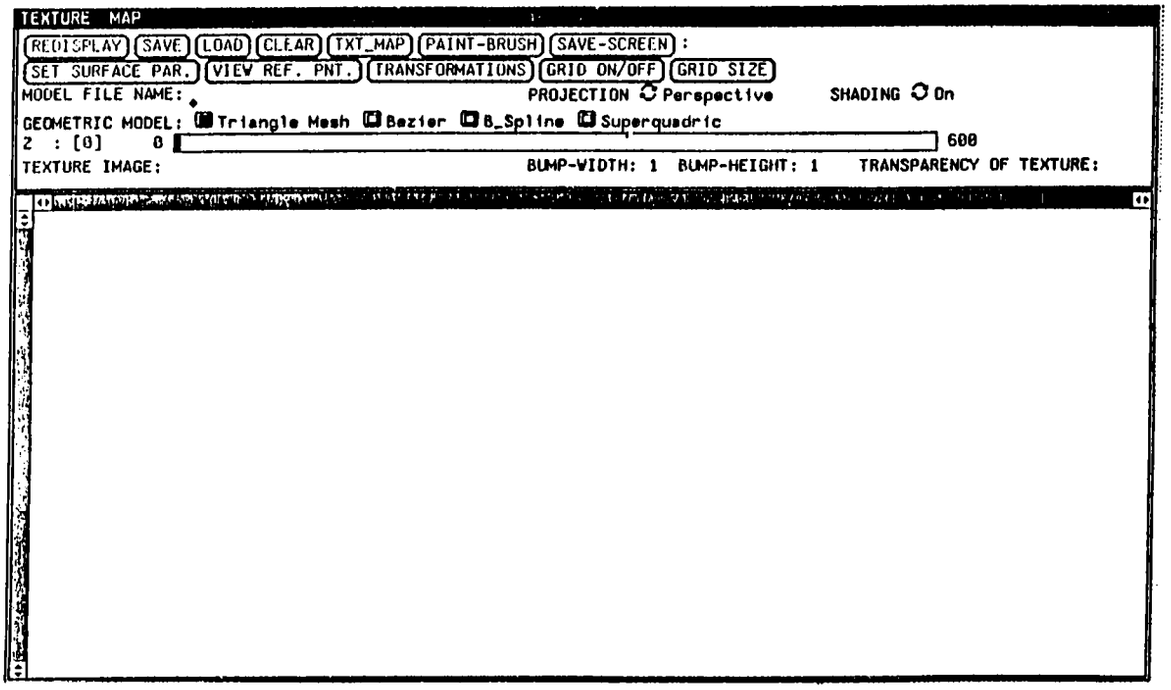

Figure A.1: The user interface of the texture mapping system.

The **SAVE-SCREEN** button saves the canvas on the disk, for the archive of texture mapped surfaces.

The **SET S-PARS** button pops up a window for the entry of Bezier, b-spline or superquadric surface parameters as shown in Figure A.2.

The first five parameters are valid if the superquadric geometric model is selected. The first two of these are exponent parameters for superquadric surfaces. These are north to south and west to east squareness parameters. The next three are coefficients of the x, y, z parametric equations for superquadric surfaces.

The following four parameters are valid for both Bezier and b-spline surfaces. The first two of these are number of control points to be given in rowwise and columnwise order. The next two points are the number of points required to be generated in rowwise and columnwise order.

Following two parameters are valid only for b-spline surfaces and they denote the order of continuity in rowwise and columnwise order respectively.

The last two parameters are valid for all the surface types and denote the initial location the created object is to be placed on the screen.

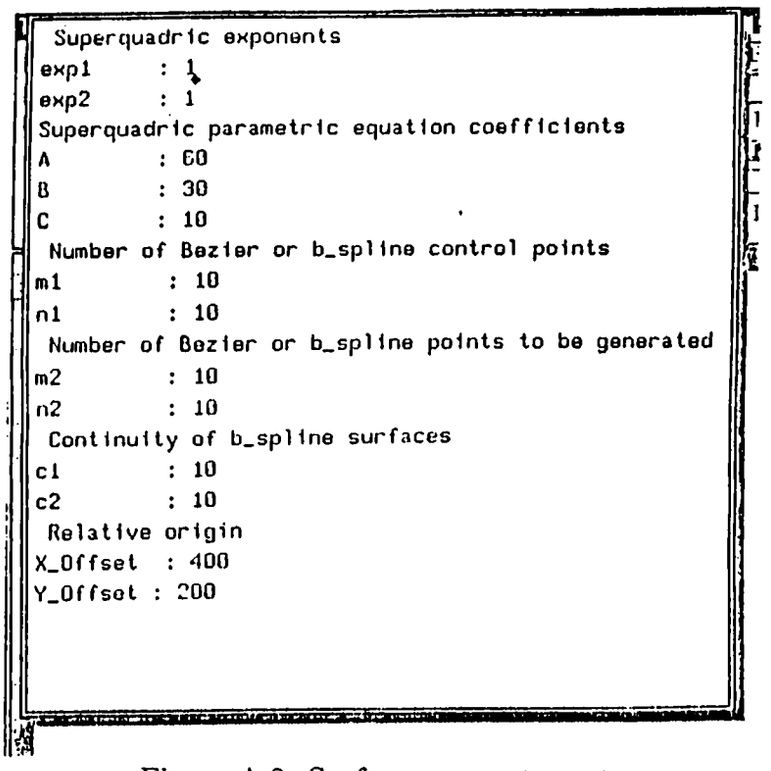

Figure A.2: Surface parameter entry

The **VIEW-REF. PNT** button pops up a small window having x and y coordinates of view reference point. The user is able to look from different angles by adjusting the view reference point. The z coordinate is received from the z-slider.

The **GRID ON/OFF** button works like toggle, that is, when the grid lines are not seen, they can be made visible by pressing this button and if they are already visible they become invisible when this button is pressed. They are helpful when constructing a geometric model with the required precision.

The **GRID SIZE** button is for adjusting the grid size. The possible values are 2, 4, 8, 16, and 32.

The text item **MODEL FILE NAME** is used to give the name of the disk file where a current geometric model is to be saved, or to load a geometric model from the disk that has been saved previously.

The **TEXTURE FILE NAME** is the name of the texture image file to be mapped to the geometric model.

The **Z** slider determines the current z coordinate, when modeling the z coordinates are received from this slider and the model is projected as if the

user were at this z coordinate.

The **TRANSFORMATIONS** button pops up a window for entering transformation parameters and selecting a specific transformation. Transformations available are translation, rotation, and scaling. The first three parameters are valid for all transformations whereas the last one which is the angle is valid only for rotation. The first three parameters are interpreted according to the transformation selected, for example if translation is selected then those values are added to the present (x, y, z) coordinate values of the object in the scene, if scaling is selected those values are multiplied with the present (x, y, z) values, if rotation is selected then given (x, y, z) values denote the vector around which the object is rotated.

The **TRANSPARENCY** coefficient determines how transparent a mapped texture is. If a texture is highly transparent then the previous texture of the surface can be seen under the current one.

The **PROJECTION** toggle item can be used to select either perspective or axonometric projections.

The **SHADING** toggle item is to turn shading on or off.

The **BUMP-HEIGHT** and **BUMP-WIDTH** are parameters to specify the height, and width of bumps. If the default value 1 is given then no bumps are mapped. If a greater integer number is given than bumps of that size are mapped together with the texture.

The **PAINT** button pops up the paint-brush subsystem for synthesizing new textures. Details of the paint brush is explained in Appendix A.3.

The **GEOMETRIC MODEL** item is to select a particular modeling type from the currently available ones. The available types are a manually designed triangular mesh, Bezier, b-spline and Superquadrics. Appropriate parameters for these modeling utilities are given using "SET SURFACE PAR." as described previously. The way the canvas events are interpreted is dependent on the geometrical model selected and it is explained in the following section.

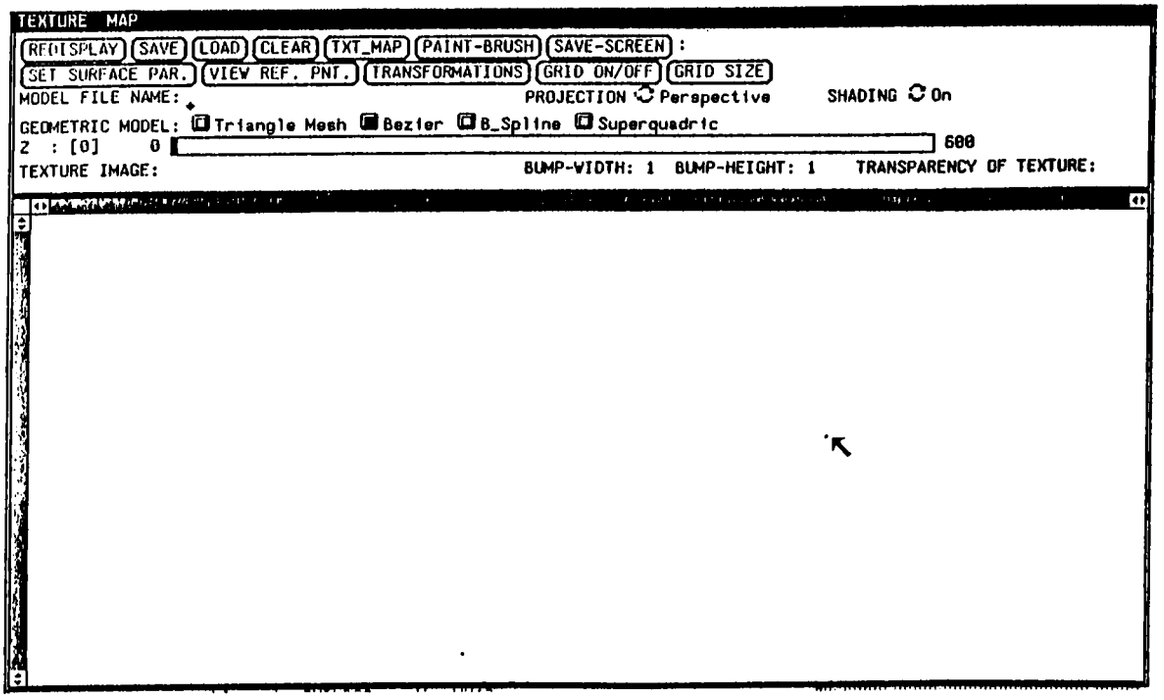

Figure A.3: Entering control points.

A.2 Canvas Events

If the selected geometric model is Bezier or b-spline then, the control points are given by pressing the *right mouse button* in the canvas. When the pressed button is released, current cursor locations is accepted as a new control point coordinate as shown in Figure A.3. Dots are placed on these coordinates. When all the points are given the surface is drawn as a wireframe on the screen.

A mesh of triangles can be constructed by adding one triangle to others at a time. The *left mouse button* is pressed and moved for this purpose. As the mouse is dragged, a triangle attached to it follows. When the button is released the last state of the triangle is added to the current set of triangles. Triangles are formed in such a way that the current mouse position is connected to the closest edge of triangles that have already been created as shown in Figure A.4. The triangle that has been added to the set can be discarded by pressing the *middle mouse button.*

If the selected geometric model is superquadrics, a menu for superquadric surfaces is displayed when the *right mouse button* is pressed in the canvas.

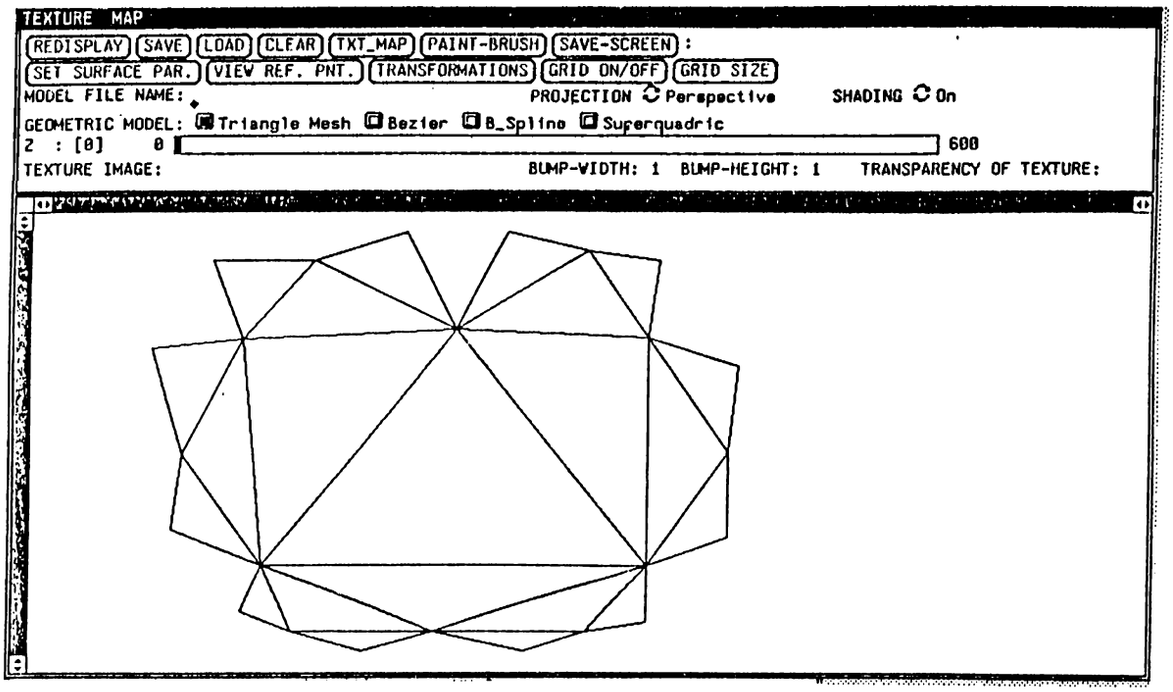

Figure A.4: Constructing a triangle mesh.

One of the possible superquadric surfaces, that is, superellipsoids, superhyperboloids or supertoroids can be chosen.

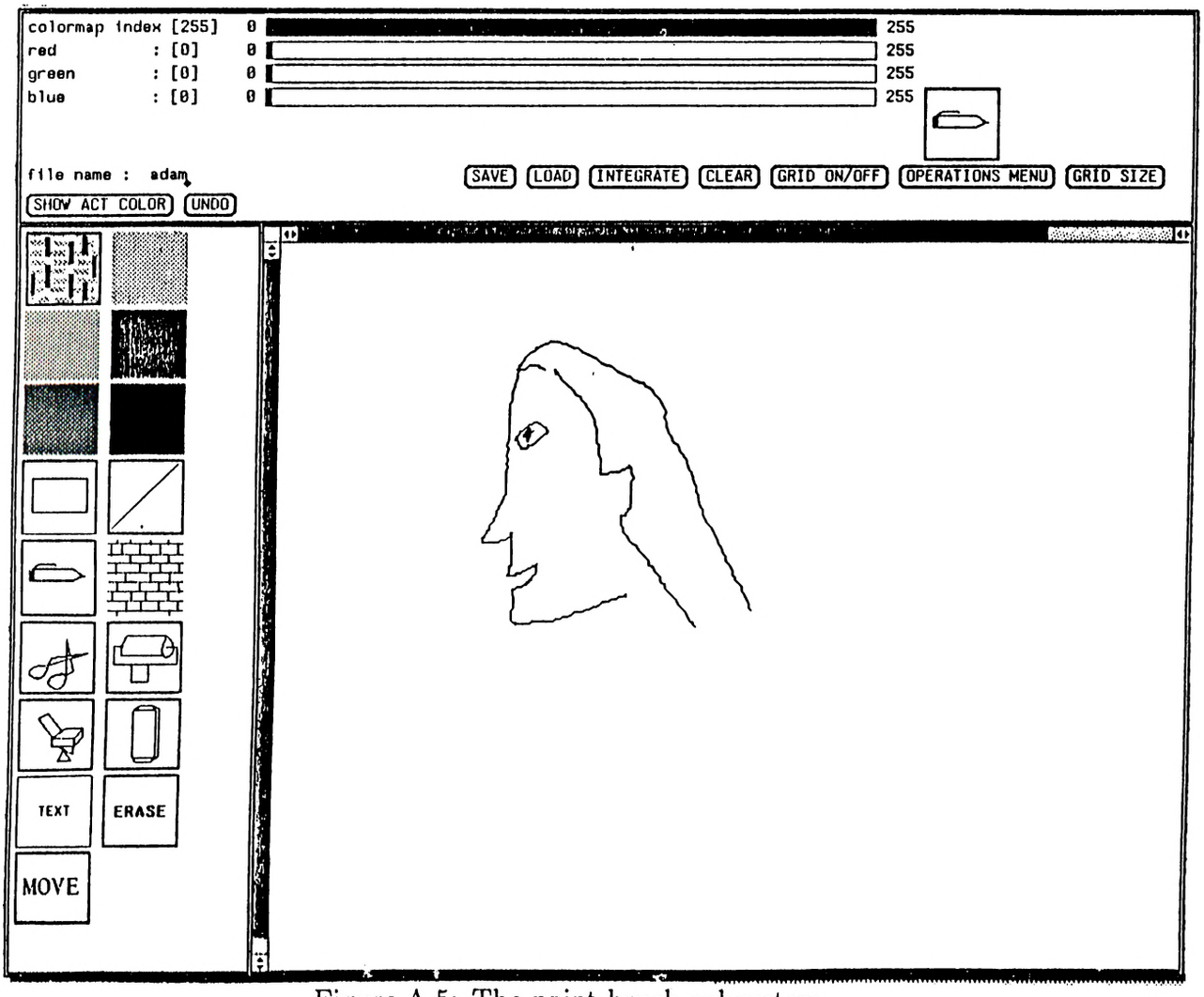

Figure A.5: The paint brush subsystem.

A.3 Paint Brush

The paint-brush subsystem can run on either black and white or color machines. It consists of three subwindows: a horizontal panel, a vertical panel and a canvas. The horizontal panel contains sliders, buttons and an icon representing the current painting pattern selected from the vertical control panel. The user interface of the paint-brush is shown in Figure A.5.

The vertical panel contains items for cutting and pasting besides painting patterns. Therefore, any part of the entire canvas can be cut, erased, or moved to another place making the replication of a single figure possible.

The first three sliders are used to select the intensities of red, green, and blue and the fourth is used to put the selected color in one of the 256 color lookup table locations. One of the buttons on this panel can be used to select the type of operation e.g, AND, OR, XOR of the source color with the destination. Another button is used to adjust the grid size according to

required thickness of the brush. The last executed action can be canceled by an undo button.

The images created can be saved as a file on disk and reloaded giving a file name and pressing the appropriate button. A preintegration button constructs and saves the summed area table of a given portion of the canvas. This table is necessary while mapping that image on a given surface.

The vertical panel has some icons representing the painting pattern or some operations such as cutting and pasting. A given region on the canvas can be automatically painted by a flood fill procedure.

These icons are self explanatory, for example the icon showing scissors represents a cutting operation, while the duster figure is used for cleaning.