A SUBDIVISION ALGORITHM FOR COMPUTER DISPLAY OF CURVED SURFACES

by

Edwin Earl Catmull

A dissertation submitted to the faculty of the
University of Utah in Partial Fulfillment of the requirements
for the degree of

Doctor of Philosophy

Department of Computer Science

University of Utah

December 1974

UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Edwin Earl Catmull

I have read this dissertation and have found it to be of satisfactory quality for a doctoral degree.

25 September 1974 — [illegible]

Date — Robert E. Stephenson
Chairman, Supervisory Committee

I have read this dissertation and have found it to be of satisfactory quality for a doctoral degree.

26 SEPT. 1974 — Steven Anson Coons

Date — Steven A. Coons
Member, Supervisory Committee

I have read this dissertation and have found it to be of satisfactory quality for a doctoral degree.

24 Sept. '74 — [illegible]

Date — David C. Evans
Member, Supervisory Committee

I have read this dissertation and have found it to be of satisfactory quality for a doctoral degree.

20 SEPT 74 — [illegible]

Date — Ivan E. Sutherland
Member, Supervisory Committee

UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Edwin E. Catmull in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to the Graduate School.

26 September 1974
Date

Robert E Stephenson
Robert E. Stephenson
Member, Supervisory Committee

Approved for the Major Department

Anthony C. Hearn
Anthony C. Hearn
Chairman/Dean

Approved for the Graduate Council

Sterling M. McMurrin
Sterling M. McMurrin
Dean of the Graduate School

ACKNOWLEDGMENTS

I express my appreciation to Steve Coons who gave me considerable help in writing this thesis.

I am indebted to fellow students and co-workers who aided with ideas and criticism. Ephraim Cohen stimulated a better mathematical form. Lance Williams suggested numerous ideas including "mapping" and helped correct my English. Martin Newell was always willing to discuss new ideas. Robert McDermott provided critical reading. Frank Crow stimulated ideas about frame buffers and shadows.

My thanks to Ivan Sutherland who stimulated me by telling me that I needed to find a faster way of subdividing.

I am grateful to Rich Riesenfeld who help stimulate a good atmosphere of interaction at the University of Utah. My thanks also to the members of my committee, Robert Stephenson, Dave Evans, Ivan Sutherland, and Steve Coons who insisted that this thesis be somewhat readable..

Thanks to Mike Milochik for photographic aid and in making prints of the pictures.

I give special thanks to my patient wife who will be glad to see me when this is all over.

TABLE OF CONTENTS

ABSTRACT

This thesis presents a method for producing computer shaded pictures of curved surfaces. Three-dimensional curved patches are used, as contrasted with conventional methods using polygons. The method subdivides a patch into successively smaller subpatches until a subpatch is as small as a raster-element, at which time it can be displayed. In general this method could be very time consuming because of the great number of subdivisions that must take place; however, there is at least one very useful class of patches -- the bicubic patch -- that can be subdivided very quickly. Pictures produced with the method accurately portray the shading and silhouette of curved surfaces. In addition, photographs can be "mapped" onto patches thus providing a means for putting texture on computer-generated pictures.

CHAPTER ONE

INTRODUCTION

A method for creating shaded pictures of curved surfaces is presented in this thesis. A motivation for the method is that we wish to produce high quality computer-generated images of surfaces and curved solid objects on a raster-scan output device. We would not only like the images to accurately represent the surfaces we choose but in addition we would like control over shading and texture. There has already been significant research directed toward these ends, especially on the hidden-surface [1,2] and shading [3,4] aspects of the problem. All such methods must must address the questions of how to model objects and then how to render them.

Polygons, and sometimes quadric patches, are used to model objects in current shaded-picture methods . There are some difficulties with using these simple pieces to model or approximate free-form curved surfaces. Approximation with polygons gives a faceted effect and a silhouette made up of straight-line segments. Quadric patches [5,6], while smooth in appearance, are not suitable for modelling arbitrary forms, since they don't provide enough degrees of freedom to satisfy slope continuity between patches.

There are two significant methods used for reducing or eliminating the undesirable visual effects that occur when polygons are used to approximate curved surfaces. The first method for getting rid of the faceted effect is that of Henri Gouraud [3]. With this method a scalar light intensity value is associated with each vertex of a polygon. Gouraud does linear interpolation of the intensity value between vertices and then subsequently across scan-lines. If adjoining polygons have the same intensities at the common vertices then this method yields continuous shading across the surface; however, the first derivative of the shading is discontinuous. Gouraud's method has been implemented by different groups making shaded-pictures. It is a simple and successful method but has a few shortcomings: the discontinuity of the derivative is noticable (the "Mach band effect"), it is difficult to do highlights, the shading is affected by the orientation of the polygon in the picture, and the silhouette is still made up of straight-line segments.

The second method developed to improve the appearance of the polygon approximation is that of Phong [4]. Since current methods of generating intensities for polygon surfaces include calculating a surface normal at the vertices, Phong decided to interpolate the entire surface normal vector between vertices and edges instead of the scalar intensity values that Gouraud used. This yields a normal at every display point which can be used to calculate the intensity. Although this normal may not be the mathematically correct one, it is close enough to use for intensity and highlight calculations. As Phong has noted, although there is still a discontinuity in the first derivative of the shading, the discontinuity is smaller than for Gouraud's method and hence less noticeable. Phong's method has been used to make some visually attractive photographs, but the problem of straight-line segments at the silhouette still remains.

Curved surface segments or "patches" can be used instead of polygons to model free-form curved surfaces. If such patches can be joined together with slope continuity across the boundaries then a picture of a surface can be made to appear "smooth" both in shading and at the silhouette. For patches to be useful in modelling a curved surface, techniques must be found for describing and manipulating the patches and for connecting them together with slope continuity across boundaries. One such patch is the bicubic patch, which is widely used (see Appendix A). Most of the ideas in this thesis will be applied to the bicubic patch, but this is not intended to imply a limitation on generality.

Generating pictures of curved patches requires techniques for

1) establishing a correspondence between points on the surface and the elements of the display raster,
2) removing hidden or, more generally, the "not seen" parts of patches, and
3) calculating light intensities to be displayed on the raster.

Chapter two will deal with the first item: it will present a technique for establishing the correspondence between points on the surface and the raster elements, Chapters three and four will describe a specific method for quickly making the correspondence when bicubic patches are used. Chapter five will deal with item two: it will discuss the "hidden-surface" problem for patches. Item three -- calculating light intensities -- will be discussed in chapters six and seven.

CHAPTER TWO

A GENERAL ALGORITHM FOR DISPLAYING CURVED PATCHES

An algorithm for establishing a correspondence between points on a patch and raster elements is described in this chapter. It applies to patches and surface sections in general, hence the algorithm presented will not be specific at the outset. Later on, when a specific kind of patch is used, more detail will be given. Before presenting that algorithm, however, some terms must be defined.

DEFINITIONS

A "raster-scan device" or "raster-display" is the device that we will consider for final output of an image. The rectangular array of "dots" that is produced on a raster-display is called the "raster." Each dot will usually be called a "raster-element." The raster element covers a very small area of the raster; however, it should not be thought of as a point. A row of raster-elements is a "scan-line." Scan-lines are usually produced in sequential order, termed "scan-line-order." Each raster-element has a brightness that is determined by the intensity value for that raster-element. The process of taking the intensity values and putting the dots on the raster with the corresponding intensities is called "displaying."

A "frame-buffer" is a memory large enough to store all of the intensity values prior to displaying. An intensity value in the frame-buffer can be addressed in a way that corresponds to the position where the value will be displayed on the raster. Locations in the frame-buffer will also be called "raster-elements" since there is a strong one-to-one correspondence between those locations and the geometric locations of the raster-elements and because the distinction between the two is not important here. For our purposes, the frame-buffer is made with random-access memory so that values can be written into it in any order, as opposed to scan-line order only. The size of the frame-buffer is determined by the resolution of the raster-display and the number of "bits" used to store intensity values. For example, if the raster has 512 scan-lines and 512 raster-elements per line and each element has 8 bits for the intensity value, then the frame-buffer requires a storage capacity of 512x512x8 bits. For the most part we will ignore the raster-display and address ourselves to the issue of putting the right intensity values in the raster-elements of the frame-buffer.

The terms relating the original description of an object to its image will now be defined. "Object-space" is the three-dimensional space in which objects will ordinarily be described. In order to generate realistic pictures of objects we make a perspective transformation [1,7,8] of the object from object-space to "image-space." Image-space is also three-dimensional but the objects have undergone a perspective distortion so that an orthogonal projection of the object onto the x-y plane would result in the expected perspective image. We want the image-space to be three-dimensional in order to preserve depth information which will later be used to solve the

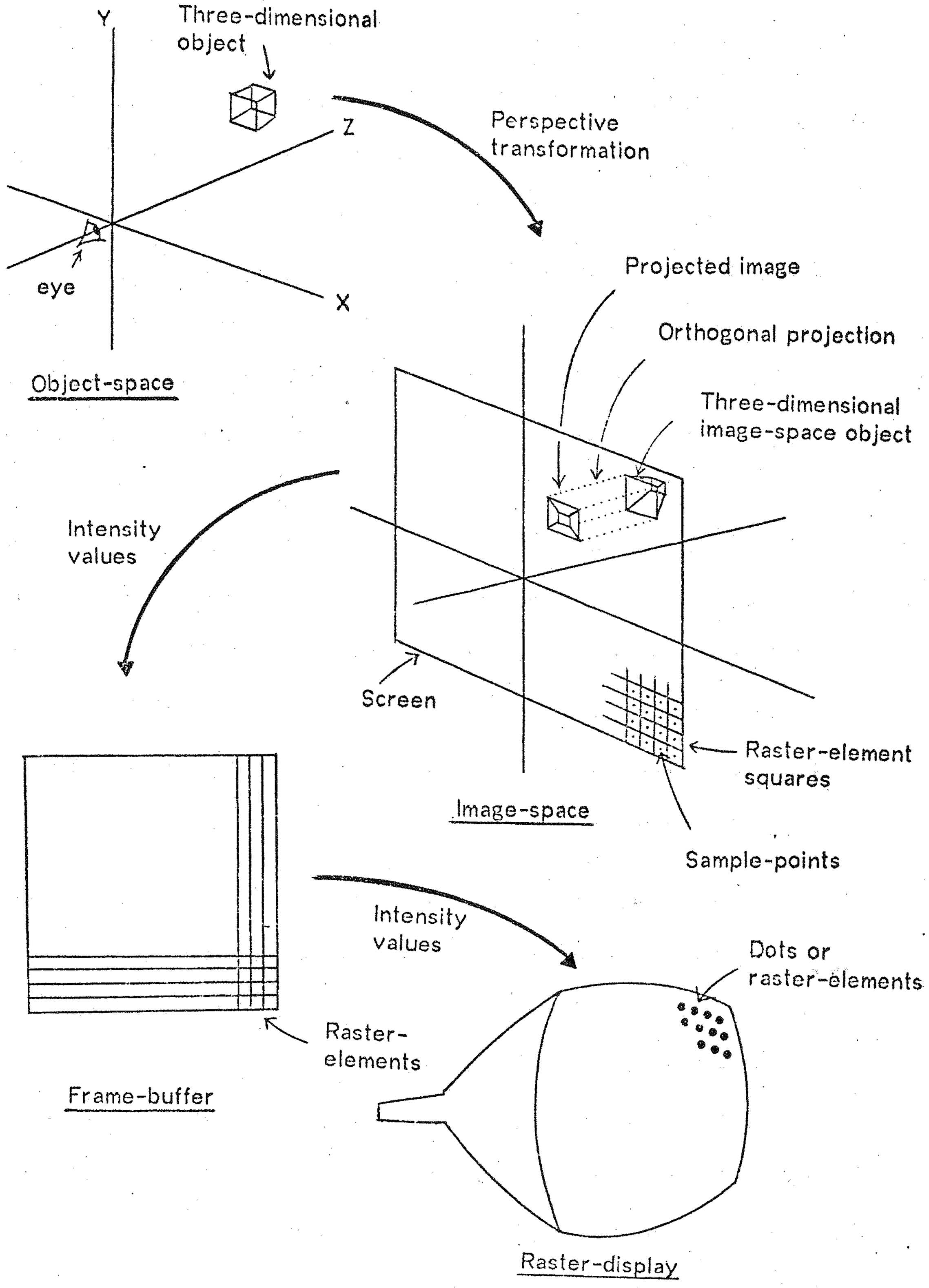

Y
Three-dimensional object
Z
Perspective transformation
eye
X
Object-space
Projected image
Orthogonal projection
Three-dimensional image-space object
Intensity values
Screen
Raster-element squares
Image-space
Sample-points
Intensity values
Dots or raster-elements
Raster-elements
Frame-buffer
Raster-display

Figure 2-1

hidden-surface problem. The orthogonal projection of the image-space object onto the x-y plane is called the "projected image." That part of the x-y plane which will be associated with the raster is called the "screen."

We must define the relationship between the image-space and the raster in order to transfer information from the projected image to the raster. Recall that the screen is the portion of the x-y plane of the image-space that corresponds to the raster. The area of the screen is divided into small squares called "raster-element squares." There is, of course, a one-to-one correspondence between raster-element squares and raster elements. The center of each raster-element square will be called a "sample-point." A diagram depicting the relationships of the above terms is shown in figure 2-1.

THE SUBDIVISION ALGORITHM

The algorithm for establishing the correspondence between a patch and the raster-elements will now be presented. The algorithm, hereafter called the "subdivision algorithm," works for either patches or segments of patches, called "subpatches." Figure 2-2 illustrates a portion of the screen where the dots represent the sample-points. (The outlines of the raster-element squares are not shown.) The curved lines represent the edges of a projected patch. Even though only the projection is shown, we assume that enough information about the patch is maintained so that the light intensity for any location on the patch can be calculated.

A statement of the algorithm is:

If the patch (subpatch) is small enough so that its projection covers only one sample-point, then compute the intensity of the patch and write it into the corresponding element of the frame-buffer; otherwise, subdivide the patch into smaller subpatches; and repeat the process for each subpatch.

Figure 2-3 shows a patch subdivided into four subpatches where most of the subpatches still cover more than one sample-point. In figure 2-4 the subpatches that are too large are again subdivided. Subdivision continues until no subpatch covers more than one sample-point.

Readers familiar with other computer-generated shaded-picture efforts will recognize a similarity between the method presented here and Warnock's hidden surface algorithm [9]. Warnock solved the hidden surface problem for polygons by recursively subdividing the screen space into successively smaller sections until all questions about the ordering of polygons left in a section were easy to answer. Warnock's algorithm differs from the one presented here in that the former subdivides the screen, while the latter subdivides the surface being rendered.

The patch subdivision algorithm as stated is very simple but some questions remain: How is the subdivision process terminated? What if a patch covers no sample-points? What if part of the patch intersects the edge of the screen or is behind the eye? How many times must a patch be subdivided? Finally, what kinds of problems does the discrete sampling introduce? Each of these issues will be discussed in turn.

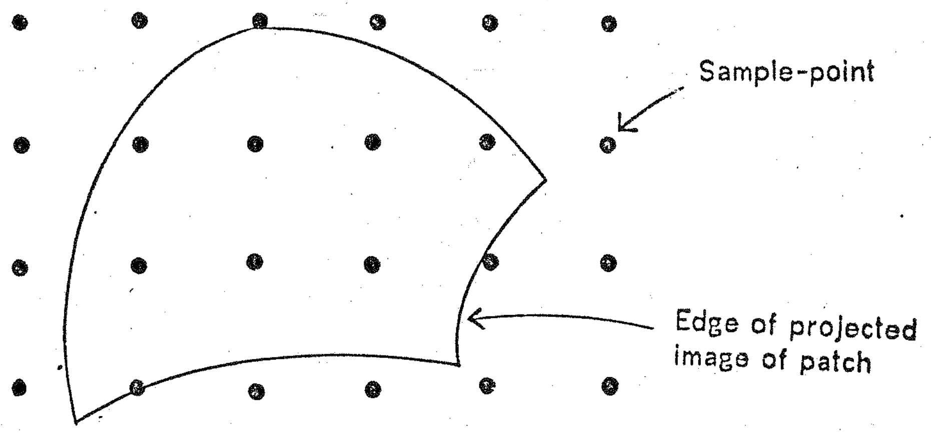


Figure 2-2

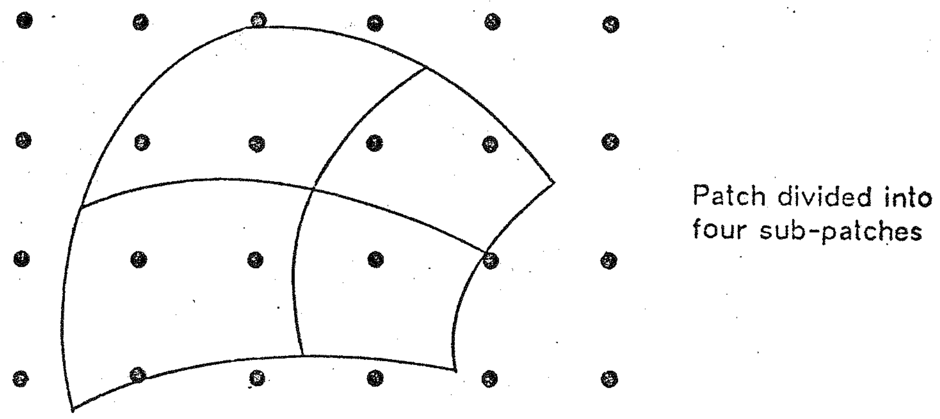


Figure 2-3

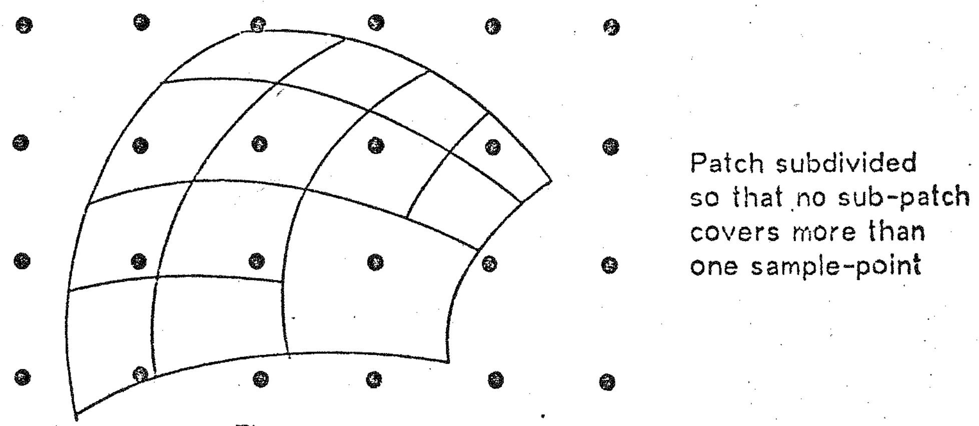

Patch subdivided so that no sub-patch covers more than one sample-point

Figure 2-4

TERMINATION

The decision as to whether or not a subpatch should be subdivided is based on termination conditions. Two termination conditions will be discussed -- size and clipping. For the purpose of this discussion we note that the terms "patch" and "subpatch" can be used interchangeably, hence we will usually use the word "patch."

As specified in the algorithm, subdivision terminates when a patch covers only one sample-point. Since the edges of a patch are curved, the test as to whether or not a

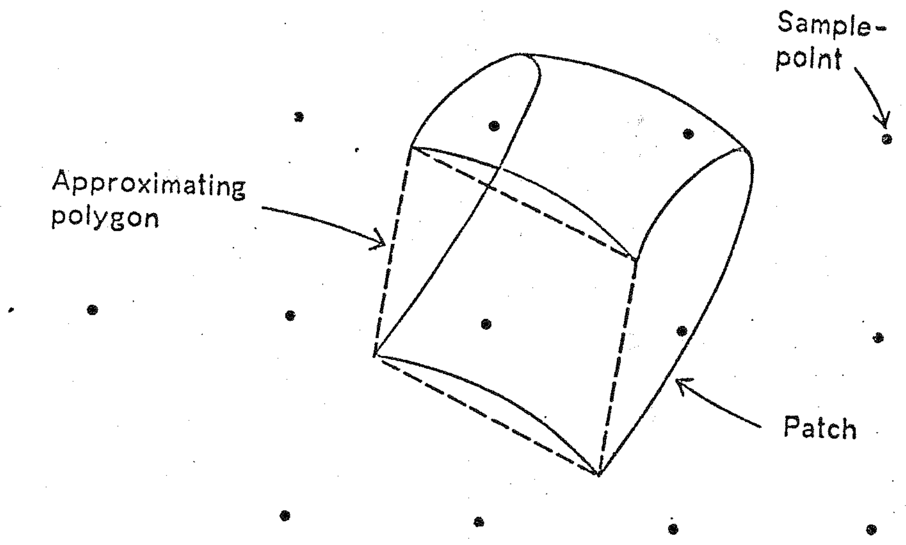


Figure 2-5

patch covers only one sample-point may be time consuming. However, for the purpose of this test, a patch can be approximated by a polygon formed by connecting the four corners of the patch with straight line segments. The size of that polygon can then be checked to determine whether or not it covers at most one sample-point. This approximation should usually be adequate for patches that are approaching the size of the raster-elements. It may not be adequate if the patch is very curved (see figure 2-5). If this case can be detected because of special characteristics of the patch geometry, then the patch can be subdivided again. If it cannot be detected then a local error may occur.

CLIPPING

A second termination condition might be a check to see if the patch is on the screen. If part of the projection of a patch in image-space onto the x-y plane lies off the screen or the patch is behind the eye then that part of the projection should not be displayed. The process of eliminating the portion of the projection that should not be on the screen is called clipping[7,8]. A clipping termination condition requires that there be some method for determining if a patch is totally on or totally off the screen. If the patch is totally on the screen then subdivision may proceed for that patch with no further need of clipping checks for the subpatches generated from that patch. If the patch is totally off the screen then that patch may be discarded. If it cannot be determined that the patch is totally on or totally off the screen then that patch should be subdivided and the clipping check should be made for each new patch resulting from the subdivision.

NUMBER OF SUBDIVISIONS

The number of times a patch nust be subdivided to get down to the size of a raster-element is proportional to the area of the patch on the screen. Consider the best case: a square two-by-two raster-elements needs only one subdivision, or 4^0; a square 2^2 by 2^2 needs 4^1+4^0 subdivisions; a square 2^n by 2^n needs $\sum_{i=0}^{n-1}4^i$ subdivisions. This is a geometric series equivalent to $(4^n-1)/3$ which is approximately $4^n/3$. The area of the square is 2^{2n} or 4^n. Therefore, the ratio of number of subdivisions to area is about 1/3. This analysis is most accurate for nearly square patches. For curved patches and skewed orientations the ratio may be somewhat larger.

THE SAMPLING PROBLEM

There are some problems encountered when using sample points. The most obvious is the "staircase-effect" or "jaggies" seen on the silhouettes of objects. In addition, a patch might be so small that it doesn't cover any sample-point, causing it to disappear. The latter problem can be solved by assigning a patch to the nearest sample-point if it doesn't cover any sample-point. The problems of sampling are inherent with the use of a raster display. Chapter seven will discuss the problems further as well as a means to alleviate them.

APPLICATION

The subdivision algorithm presented above was first applied to bicubic patches. Bicubic patches are convenient on several counts: they are widely used, they can be compactly specified in several different ways (see Appendix A), they can be easily joined with first derivative continuity at the boundaries and they can be subdivided very easily. The next two chapters will present a method for fast subdivision of such patches. It should be emphasized at this point however that the subdivision algorithm is by no means limited to bicubic patches but can be applied to other kinds of surfaces.

CHAPTER THREE

SUBDIVIDING A CUBIC CURVE

A method for quickly subdividing a cubic curve is presented in this chapter; the extension to patches is developed in the next chapter. The method uses a new kind of difference equation for obtaining the midpoint of a curve segment. The resulting ability to quickly subdivide a curve makes the application of the subdivision algorithm practical.

SUBDIVIDING THE CUBIC CURVE

Subdivision is easy because, as we shall see, the midpoint of a cubic curve is the average of its two endpoints minus a correction term. One result of this is that the cubic can be subdivided with only three adds. A similar method can be used to find the derivative at the midpoint.

Consider the cubic:

$$f(t) = at^3 + bt^2 + ct + d.$$

The problem is to find $f(t)$ when $f(t+h)$ and $f(t-h)$ are already known. Note first that:

$$f(t \pm h) = a(t \pm h)^3 + b(t \pm h)^2 + c(t \pm h) + d$$

$$= a(t^3 \pm 3ht^2 + 3h^2t \pm h^2) + b(t^2 \pm 2th + h^2) + c(t \pm h) + d.$$

If the points $f(t+h)$ and $f(t-h)$ are added then:

$$f(t+h) + f(t-h) = 2a(t^3 + 3h^2t) + 2b(t^2 + h^2) + 2ct + 2d$$

$$= 2f(t) + 2h^2(3at + b);$$

therefore $f(t)=[f(t+h) + f(t-h)]/2 - h^2(3at + b)$.

The midpoint then is the average of the two endpoints minus the correction term, $h^2(3at+b)$. The correction term is a linear function of t and h. If $h=1/2^n$, then since h is a power of two it can be calculated on a computer with a simple binary shift.

The correction term at t can similarly be found from the correction terms at t+h and t-h. If $g(t) = h^2(3at + b)$ then $g(t \pm h) = h^2(3a(t \pm h) + b)$. Again by adding:
$g(t+h) + g(t-h) = 2h^2(3at) + 2bh^2 = 2g(t)$
and so

(3-1) $\quad g(t) = [g(t+h) + g(t-h)]/2.$

Let $h_n = 1/2^n$ where n can be considered a level of subdivision. Then $h_{n+1} = h_n/2$ and $h^2_{n+1} = h^2_n/4$ and since $g(t) = h^2(3at + b)$ then

(3-2) $\quad g_{n+1}(t) = g_n(t)/4.$

and (3-1) can be rewritten as

(3-3) $g_n(t) = [g_n(t+h_n) + g_n(t-h_n)]/2$

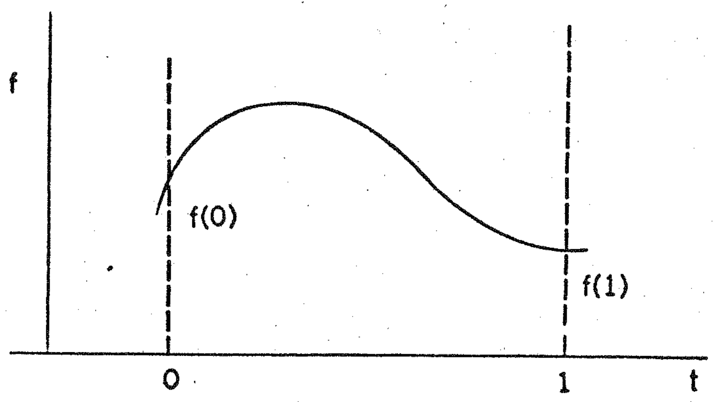
Therefore:

(3-4) $f(t) = [f(t+h) + f(t-h)]/2 - [g_n(t+h) + g_n(t-h)]/2.$

Equation (3-4) is the subdividing difference equation for a cubic and equations (3-2) and (3-3) are used to get the right correction term as h_n is made smaller by powers of two.

Equations 3-2, 3-3, and 3-4 can be expressed diagrammatically as as shown in figure 3-5. At each end point there are two values -- the values of the function and the correction term. Those values can be put into two registers. The contents of the registers for the midpoint can be found by the indicated combination of the registers at the endpoints. In order to subdivide one of the new halves it is necessary to update the correction term at the end points since h_n will be half as big and the correction terms are functions of h_n. In terms of the diagram in figure 3-1, the subdivision process cascades downward. The correction terms are functions of the level of subdivision. The initial values in the registers can be found by solving $f(t)$ and $g_0(t)$. Since $n=0$ then $h^2=1$ and $f(0)=d$, $g_0(0)=b$, $f(1)=a+b+c+d$, and $g_0(1)=3a+b$.

It may be useful sometimes to compute the derivative. The derivative can be found as a simple function of the endpoints and a correction term that is dependent only upon the depth of subdivision. Instead of adding $f(t+h)$ and $f(t-h)$, subtract them:



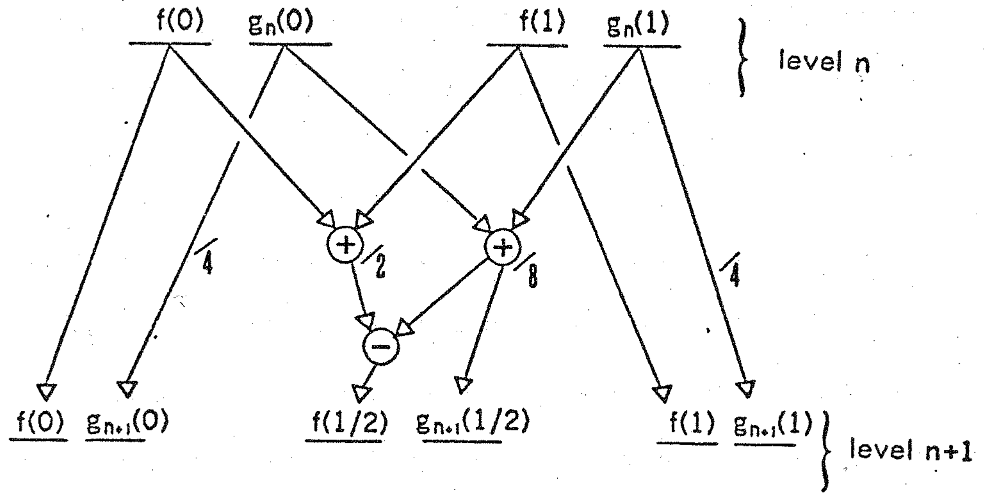


Figure 3-1

$$f(t+h) - f(t-h) = 2a(3ht^2 + h^3) + 2b(2th) + 2ch$$

$$= 2h3at^2 + 2ah^3 + 3h2bt + 2hc$$

Note that the derivative is: $f'(t) = 3at^2 + 2bt + c$

therefore $f(t+h) - f(t-h) = 2hf'(t) + 2ah^3$ so

(3-5) $f'(t) = [f(t+h) - f(t-h)]/2h - ah^2$

Note that ah^2 is a function only of the level of subdivision.

A MATRIX REPRESENTATION

The subdivision method can be put in matrix form and hence related to the matrix methods for generating bicubic patches presented in Appendix A. The matrix form of a simple cubic is:

$$f(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

The correction terms and function values for the simple cubic can also be put in matrix form. Let that matrix be called the correction matrix C and it contents be:

$$C = \begin{bmatrix} f(t-h) \\ g_n(t-h) \\ f(t+h) \\ g_n(t+h) \end{bmatrix}$$

Recall that the correction factor is $g_n(t)=h^2(3at+b)$. At the zeroth level of subdivision $h^2=1/2^{2n}=1$. So $f(0)=d$, $g_0(0)=b$, $f(1)=a+b+c+d$, and $g_0(1)=3a+b$. If we put these values that fit in C then

$$C = \begin{bmatrix} d \\ b \\ a+b+c+d \\ 3a+b \end{bmatrix}$$

Next let

$$A = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

We can get the values in C by using the matrix:

$$S=\begin{bmatrix}0&0&0&1\\0&1&0&0\\1&1&1&1\\3&1&0&0\end{bmatrix}$$

The relation is

(3-5) $C = SA$

The object of subdivision is to find the C matrix for each half of a segment. Let those two matrices be C_L and C_R for C left and C right. There are matrices L and R such that $C_L = LC$ and $C_R = RC$. The operation on the values of C have already been defined. They require that:

$$L=\begin{bmatrix}1&0&0&0\\0&1/4&0&0\\1/2&-1/8&1/2&-1/8\\0&1/8&0&1/8\end{bmatrix}$$

$$R=\begin{bmatrix}1/2&-1/8&1/2&-1/8\\0&1/8&0&1/8\\0&0&1&0\\0&0&0&1/4\end{bmatrix}$$

As an example, the second quarter C' of a segment can be found by $C' = RLC$. Note that all entries in the L and R matrices are powers of two. The bicubic subdivision method is merely a fast way of doing a matrix multiply taking advantage of the values in L and R.

SUBDIVISION APPLIED TO POLYNOMIALS

The subdivision notion can be extended to polynomials in general. A polynomial of degree n can be written as:

$$f(t) = \sum_{i=0}^{n} a_i t^i$$

therefore

$$f(t\pm h) = \sum_{i=0}^{n} a_i (t\pm h)^i$$

The binomial expansion for $(t\pm h)^i$ is

$$(t\pm h) = \sum_{k=0}^{i} \binom{i}{k} t^{i-k} (\pm h)^k$$

Again, as in the cubic case, add f(t+h) and f(t-h). Consider just one term $(t\pm h)^i$

$$(t+h)^i + (t-h)^i = 2\sum_{k=0}^{i} \binom{i}{k} t^{i-k} h^k \quad (k \text{ even})$$

Since a_i are only coefficients,

$$f(t+h) + f(t-h) = 2\sum_{i=0}^{n} a_i \sum_{k=0}^{i} \binom{i}{k} t^{i-k} h^k \quad (k \text{ even})$$

but $\sum_{i=0}^{n} a_i t^i = f(t)$

so we can take the first element out of the series:

$$f(t+h) + f(t-h) = 2f(t) + 2\sum_{i=2}^{n} a_i \sum_{k=2}^{i} \binom{i}{k} t^{i-k} h^k \quad (k \text{ even})$$

and finally

$$f(t) = [f(t+h) + f(t-h)]/2 - \sum_{i=2}^{n} a_i \sum_{k=2}^{i} \binom{i}{k} t^{i-k} h^k \quad (k \text{ even})$$

The correction term is a polynomial of degree n-2. One can apply the same method to the correction term to reduce it to a function of the endpoints and their separation, h.

TAYLOR SERIES

A further extension of the subdivision concept applies to Taylor series. This last discussion should point the way to finding appropriate solutions for functions other than simple polynomials. Recall that the Taylor series is:

$$f(x) = f(a) + (x-a)f'(a) + (x-a)^2 f''(a)/2! + \ldots + (x-a)^n f^{(n)}/n! + R_n$$

and if $R_n \to 0$ as $n \to \infty$ then

$$f(x) = \sum f^{(n)}(a)(x-a)^n/n!$$

Let $a = (x \pm h)$

then

$$f(x) = \sum(\pm h)^n f^{(n)}(x \pm h)/n!$$

Again h_k can be of the form $1/2^k$. If for some k the truncated series is a good approximation to f(x) in the interval of h_k, then the function can be found in any subinterval of h_k. This differs from the polynomial case in that information for a segment can be thought of as being at one end rather that at both ends.

CHAPTER FOUR

EXTENSION OF CUBIC SUBDIVISION TO SURFACES

The method of subdividing cubic curves can be extended to bicubic surfaces. With a cubic curve there is a value and a correction term at each end; with a bicubic patch there is a value and three correction terms at each corner. Subdivision of the patch into four pieces means finding the midpoint of each of the sides and the midpoint of the patch.

There may be several components to the vector that describes a three dimensional patch. The surface has three purely geometric components X(u,v), Y(u,v), and Z(u,v). There may be additional components for other information such as shading and color. Each component is treated the same so we need only consider one component of the patch here.

Since we are considering only one component of the surface let that component be:

$$f(u,v) = [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

If we multiply the u matrix by the coefficient matrix this equation becomes

$$F(u,v) = [F_1 \;\; F_2 \;\; F_3 \;\; F_4] \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where

$F_1 = a_1u^3 + b_1u^2 + c_1u + d_1$

$F_2 = a_2u^3 + b_2u^2 + c_2u + d_2$

$F_3 = a_3u^3 + b_3u^2 + c_3u + d_3$

$F_4 = a_4u^3 + b_4u^2 + c_4u + d_4$

Since each F_n is a cubic we observe that there is a correction term for each F_n. Call this correction term G_n.

The final value of the component is

$f(u,v) = v^3 \cdot F_1 + v^2 \cdot F_2 + v \cdot F_3 + F_4$.

Consider $v^3 \cdot F_1$:

$v^3 \cdot F_1 = (a_1v^3)u^3 + (b_1v^3)u^2 + (c_1v^3)u + (d_1v^3)$.

So v^3 can be considered as a coefficient in the u equation. In that case $v^3 \cdot G_1$ is a correction term for $v^3 \cdot F_1$. Similarly, $V^2 \cdot G_2$ is the correction term for $V^2 \cdot F_2$, etc. If we sum the F_n and G_n

$f = v^3 \cdot F_1 + v^2 \cdot F_2 + v \cdot F_3 + F_4$.

$g = v^3 \cdot G_1 + v^2 \cdot G_2 + v \cdot G_3 + G_4$.

Now g is the correction term for f along constant v. This reduction to two numbers when v is constant is exactly as expected since the curve along constant v is simple cubic. Therefore, for any v, the function and its correction terms along u can be found.

Next suppose v changes while u is constant. In this case F_n and G_n are constants and can be thought of as just coefficients in the above equations. Let the correction term for f be c_f and the correction term for g be c_g. Since g is a correction term for f, then c_g is a correction term for c_f.

These four numbers can be arranged in a square as shown in figure 4-1. This reresentation will be called a "register-square."

f	g
c_f	c_g

Figure 4-1

In the register-square, f is the value of the function at u,v, and g, c_f, and c_g are correction terms. If we move in the v direction then c_f corrects f and c_g corrects g. If we move in the u direction, g corrects f and c_g corrects c_f. Inserting u, v, and the coefficients yields:

$v^3(u^3a_1 + u^2b_1 + uc_1 + d_1) + v^2(u^3a_2 + u^2b_2 + uc_2 + d_2) + v(u^3a_3 + u^2b_3 + uc_3 + d_3) + (u^3a_4 + u^2b_4 + uc_4 + d_4)$	$h^2[v^3(3a_1u + b_1) + v^2(3a_2u + b_2) + v(3a_3u + b_3) + ((3a_4u + b_4)]$
$k^2[3v(u^3a_1 + u^2b_1 + uc_1 + d_1) + (u^3a_2 + u^2b_2 + uc_2 + d_2)]$	$h^2k^2[3v(3a_1u + b_1) + (3a_2u + b_2)]$

where h^2 and k^2 apply to the u and v directions respectively and have the same meaning as h in chapter three. As in the cubic polynomial case they can be calculated on a computer with a shift.

A register-square makes it easy to think about an algorithm for subdividing a patch. A register-square can be associated with each corner of a patch (see figure 4-2).

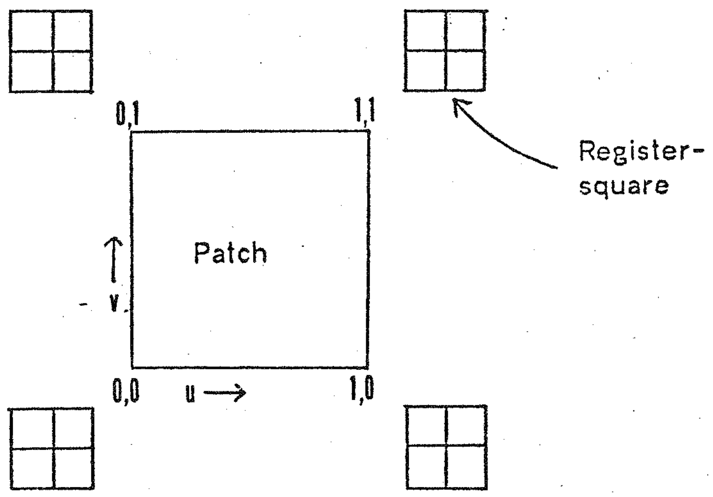


Figure 4-2

The subdivision algorithm can be applied to the register squares either vertically or horizontally depending on whether u or v is constant. Figure 4-3 shows a notation for horizontal subdivision. The top two values of the left and right register-squares are used to create the top two values of the middle square using the same subdivision algorithm presented in chapter three. The same applies to the bottom two values of each square. Vertical subdivision works in a similar manner. The notation of figure 4-3 can be used for the entire patch as shown in figure 4-4. The center square can be derived from two of the newly created edge squares. There are now four squares for each quarter of the patch so subdivision can again take place for each quarter.

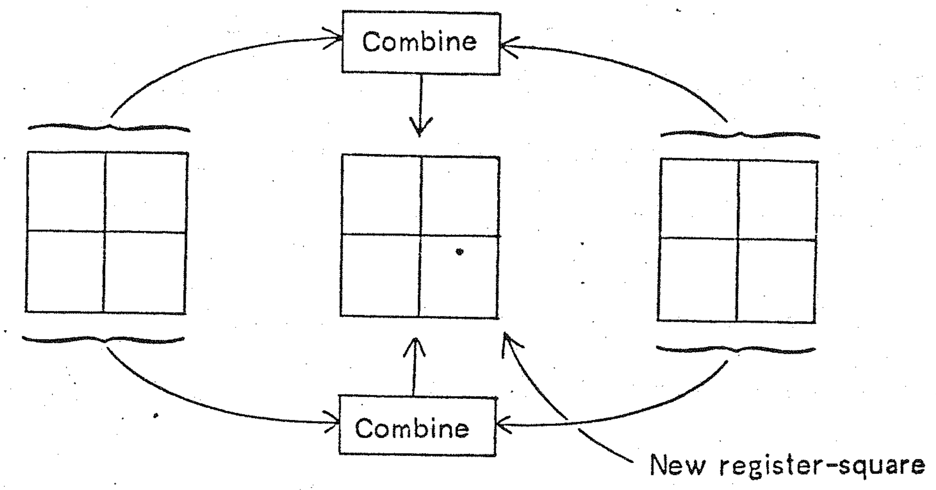


Figure 4-3

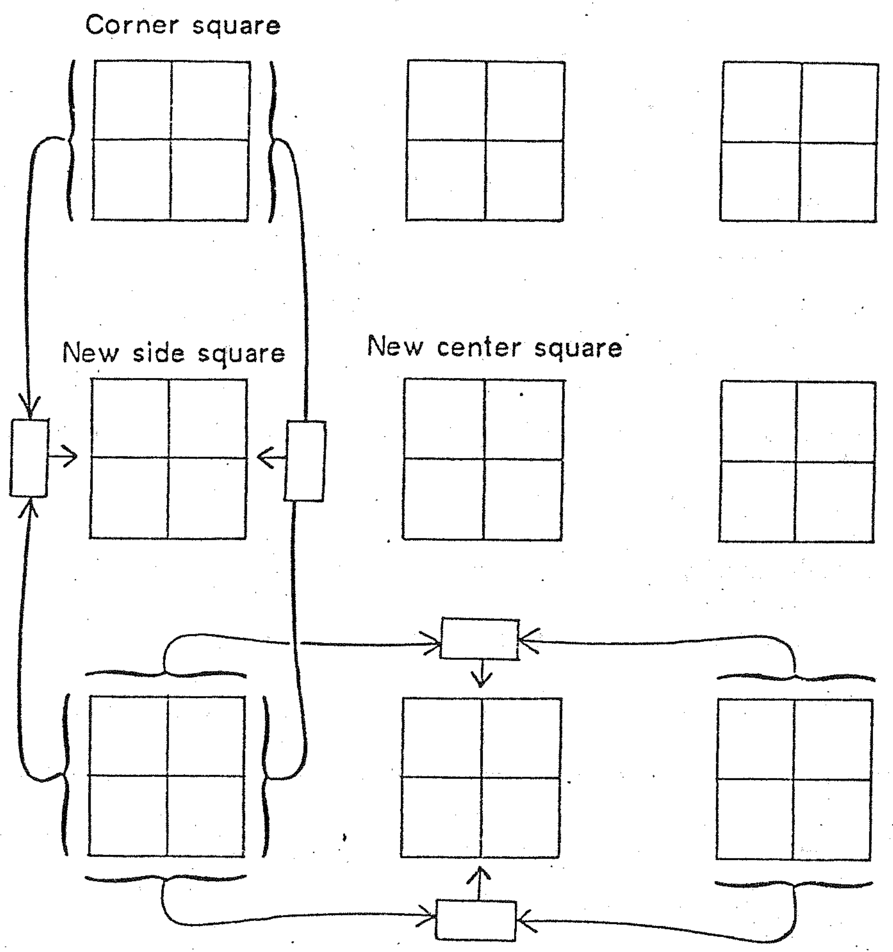


Figure 4-4

It is important to note that the concept of level of subdivision still applies. This means that the correction terms must be adjusted each subdivision. One could think of each square as extending in two directions. When two squares are combined to create a new one then its correction terms in that direction are divided by four as required by the subdivision algorithm. The extension in the other direction is the same for the new square as for the two end ones and is unaffected by subdivision. This depth correction will be called "reduction."

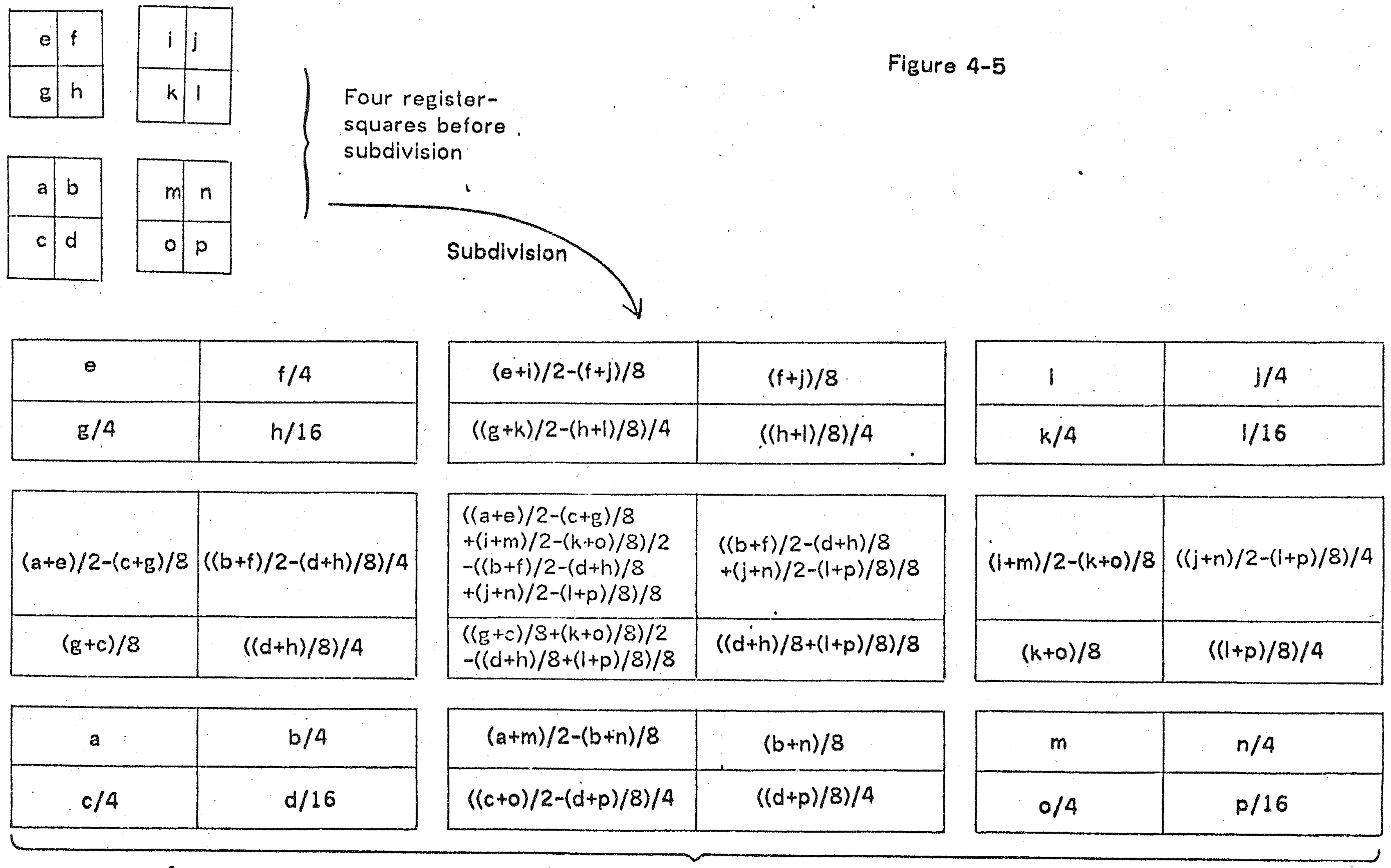
A full patch subdivision can be clarified with figure 4-5. The letters in the four small boxes represent the initial values in the register-squares. The next nine boxes depict the subsequent values in each register-square after subdivision.

If the initial values of u and v are (0,0), (1,0), (1,1), and (0,1) then the initial square values are as shown in figure 4-6.

PERSPECTIVE

Perspective presents a problem for patch subdivision since the above method works only for components that are simple bicubics and the perspective transformation results in rational bicubics. In order to display a perspective view of a surface the mathematical definition of a patch must go through a perspective transformation which results in a surface equation of $F(u,v) = [X(u,v)\ Y(u,v)\ Z(u,v)\ W(u,v)]$. $W(u,v)$ is called the homogeneous coordinate [7,8] and is generated by the perspective transformation. Three ways of displaying a perspective surface are:

Figure 4-5



e f g h
i j k l
a b c d
m n o p
Four register-squares before subdivision
Subdivision
e
f/4
g/4
h/16
(e+i)/2-(f+j)/8
(f+j)/8
((g+k)/2-(h+l)/8)/4
((h+l)/8)/4
i
j/4
k/4
l/16
(a+e)/2-(c+g)/8
((b+f)/2-(d+h)/8)/4
(g+c)/8
((d+h)/8)/4
((a+e)/2-(c+g)/8 +(i+m)/2-(k+o)/8)/2 -((b+f)/2-(d+h)/8 +(j+n)/2-(l+p)/8)/8
((b+f)/2-(d+h)/8 +(j+n)/2-(l+p)/8)/8
((g+c)/8+(k+o)/8)/2 -((d+h)/8+(l+p)/8)/8
((d+h)/8+(l+p)/8)/8
(i+m)/2-(k+o)/8
((j+n)/2-(l+p)/8)/4
(k+o)/8
((l+p)/8)/4
a
b/4
c/4
d/16
(a+m)/2-(b+n)/8
(b+n)/8
((c+o)/2-(d+p)/8)/4
((d+p)/8)/4
m
n/4
o/4
p/16
Nine register-squares after subdivision

u=0 v=0

d_4	b_4
d_2	b_2

u=0 v=1

$d_1+d_2+d_3+d_4$	$b_1+b_2+b_3+b_4$
$3d_1+d_2$	$3b_1+b_2$

u=1 v=1

(sum of all coefficients)	$3(a_1+a_2+a_3+a_4)$ $+(b_1+b_2+b_3+b_4)$
$3(a_1+b_1+c_1+d_1)$ $+(a_2+b_2+c_2+d_2)$	$3(3a_1+b_1)$ $+3a_2+b_2$

u=1 v=0

$a_4+b_4+c_4+d_4$	$3a_4+b_4$
$a_2+b_2+c_2+d_2$	$3a_2+b_2$

Figure 4-6

1. Get the equation of the perspective surface by dividing by the homogeneous coordinate. This results in a rational cubic wich does not fit into the subdividing scheme.
2. Subdivide X, Y, Z, and W and do the perspective division at every point. This requires extra space for subdividing W and time to do the subdivision and perspective division.
3. Take only the defining points of the patch (See appendix A) through the perspective transformation and recreate the cubic in perspective space. The defining points are correctly recreated, although the surface they now define is

not the "correct" surface (as defined in (1)) but, in the subjective opinion of the author, is a very close approximation. The pictures in this thesis were made using this method.

CHAPTER FIVE

THE HIDDEN SURFACE PROBLEM

In order to display surface patches it is necessary to determine which surfaces are visible. Two methods that can be used to solve the hidden surface problem for bicubic patches are the "modified Newell algorithm" and the "z-buffer algorithm."

THE MODIFIED NEWELL ALGORITHM

Newell, Newell, and Sancha [10] have devised an algorithm for displaying polygons that sorts the polygons in z order and paints the polygons in that order into a frame buffer; the polygon farthest away from the eye is written first. Subsequent polygons may be written over those already in the buffer thus eliminating obscured polygons. If two polygons intersect or are situated so that it is not easy to sort them in z order, they are split into smaller pieces until they can be correctly sorted.

There are two parts to the z sort in the Newell algorithm. The first is a simple, quick z sort of all the polygons based on their farthest vertex. It does not guarentee that the polygons are in the correct order to be written into the buffer. The second is a time-consuming sort that guarentees that the polygons are in the right order.

Martin Newell of the University of Utah has noted in private discussion that that algorithm can be extended to patches and that the Beziér control points (see appendix

A) can be used for ordering. Since a patch is constrained to lie within the convex hull of its defining points, the defining points can be used to sort the patches. If the order between two patches can not be determined then the patches can be subdivided until the correct sort can be done. With the fast subdivision of bicubic patches one can keep subdividing the patches until the z order is resolved and then render the curved pieces as shown earlier. The relationship between Beziér control points and the correction factors is shown in Appendix B.

THE Z-BUFFER

The z-buffer is an extension of the frame-buffer idea in that the z value from the image-space of the visible object is stored at every raster-element as well as the intensity. The z value of any new point to be written into the buffer is compared with the z value of the point already there. If the new point is behind, it is discarded. If it is in front it replaces the old value.

There are several advantages to using the z-buffer. Hidden surface problems and intersection of arbitrary surfaces are handled trivially. Pictures can be of any complexity. Except as noted below, surfaces may be written into the buffer in any order, thus saving the time-consuming sorting of highly complex surfaces.

There are of course some disadvantages to the z-buffer. A 512 by 512 buffer with 8 bits of intensity and 20 bits of z uses a quarter of a million 28 bit words. At the current cost of memory this means an expensive implementation. A more serious problem is that of "anti-aliasing," or getting rid of the "staircase effect" (see Chapter

7). Any algorithm for getting rid of the staircase effect requires that on the silhouette of objects the intensity at the corresponding raster-elements will be some combination of intensities from at least two objects -- namely, the object being displayed and the object being partially obscured, which may of course be simply background. If all of the objects have been rendered in random order then it is possible that the intensities from the wrong objects will be combined, giving a local error. This means that it may sometimes be necessary to sort the objects to eliminate the staircase effect.

The author implemented the z-buffer algorithm by paging the z-buffer onto disk. Thirty-two pages could be resident in core where each page contained a 16 by 16 square section of the raster. The time needed for swapping was small compared to the time spent by the software implementation of the subdivision algorithm. All of the pictures in this thesis were made using the z-buffer.

A combination z-buffer-Newell algorithm could be developed where a simple z sort puts the patches in approximately the right order and the z-buffer guarentees that they are in the right order. The only error that would occur would be a local "staircase error" on an edge if the associated patch were written in the wrong order. We have traded off the time-consuming sort for the increased memory and the possibility of a small error.

CHAPTER SIX

INTENSITY

When a patch has been subdivided into subpatches small enough to cover only one sample point it is necessary to associate an intensity with the corresponding point. There are several ways of getting the intensity at each point.

1. Use the normal to the surface to calculate intensity.
2. Use some intensity function of u and v.
3. Map the intensities from some picture.
4. Modify existing intensities for shadows or transparency.

There are good examples where each of the above might be applicable, so they will each be discussed.

USING SURFACE NORMALS

The normal to a surface is frequently needed to calculate the intensity. Phong has already shown [4] several ways of calculating intensity if the surface normal and the light sources are known. A typical way of doing it would be to use as the intensity the dot product of a light vector and the surface normal. One needs to use the normal from the object-space surface before the perspective transformation is performed instead of the image-space surface because perspective distorts the surface and hence falsifies the intensity. Unfortunately, finding the normal is complicated by the fact that the equation of the normal to a bicubic patch is a fifth degree polynomial.

Three ways of finding the normal are:

1. Use a fifth degree subdivision equation to solve the normal surface equation. This seems impractical because of the increased space, time, and complexity required.
2. Approximate the normal equation with a cubic equation and then subdivide the components of that equation just as the surface equation is subdivided. Appendix C explains how to approximate the normal equation. This method was used to make the pictures for this thesis. Six components of the patch were subdivided to make the pictures -- the three components of the surface and the three components of the normal.
3. Take the cross product of the tangents at every point to get the surface normal. We have already shown in chapter three that the tangent at the midpoint of a line can readily be found. Therefore the three components of the object-space patch can be subdivided (in addition to the perspective patch) and the normal can be found by taking the cross product of the u direction tangent and the v direction tangent at each sample point. This method requires a little extra information in order to get the tangents and, of course, it requires the extra work involved in taking a cross product at every point.

USING AN INTENSITY FUNCTION

The intensity at a raster element is represented by a number and any useful way of deriving that number is legitimate. Instead of being a function of the orientation of the surface, the intensity might be a function of pressure, strain, height, density, artistic whim, etc. If these can be expressed in a bicubic equation then they fit into the subdividing scheme. Color components could also be calculated as bicubic equations.

One must use care to ensure that the calculated intensity values stay within required bounds for the display. Three ways of doing this are:

1. Check each calculated value and clip it if too large or too small.
2. If using normals, renormalize at every point.
3. Solve for the Beziér control points of the patch (see Appendix A) and normalize those points so that none of them are out of range, then recalculate the patch. Since the patch is contrained to lie within the convex hull of the points they will be in the required bounds. First derivative continuity across patch boundaries may be lost with this method.

MAPPING

Photographs, drawings, or any picture can be mapped onto bivariate patches. This is one of the most interesting consequences of the patch splitting algorithm. It gives a method for putting texture, drawings, or photographs onto surfaces. It also allows one to have reflections in pictures, as in flat or curved mirrors.

One can make a correspondence between any point on a patch and an intensity on a picture. If a photograph is scanned in at a resolution of x times y then every element can be referenced by $u \cdot x$ and $v \cdot y$ where $0 \leq u,v \leq 1$. In general, one could think of the intensity as a function I(u,v) where I references a picture. In keeping with the bicubic method, the picture does not need to be rectangular but can have edges that are cubic curves.

In practice the above method for getting intensities from pictures can fall afoul of sampling errors. This will occur when the number of points to be displayed on a patch is less than the number of elements in the stored picture, resulting in less information being put on the patch than is in the picture.

One way to alleviate this is to map areas onto areas rather than points onto points. Every time the patch is subdivided, the picture is also subdivided. When the algorithm determines that a subpatch is to be displayed, the corresponding area on the picture is known. The average intensity of that area can be found and used as the intensity of the piece. While this reduces considerably the sampling problem it does not completely solve it.

The sampling problem can be better understood by considering figure 6-1. Suppose that the algorithm subdivides the patch up as shown and that the squares in the figure represent raster-element squares. Since in general the pieces of the patch do not mesh well with the raster grid there will be times when more than one piece of the patch logically belongs to one display element, ie., pieces a, b, and c would be painted in element one. However, a, b, and c are not usually created in time sequential order so combining them would be difficult. If only one of the pieces is chosen for display then some information would be lost. A solution to the problem is presented in chapter Seven.

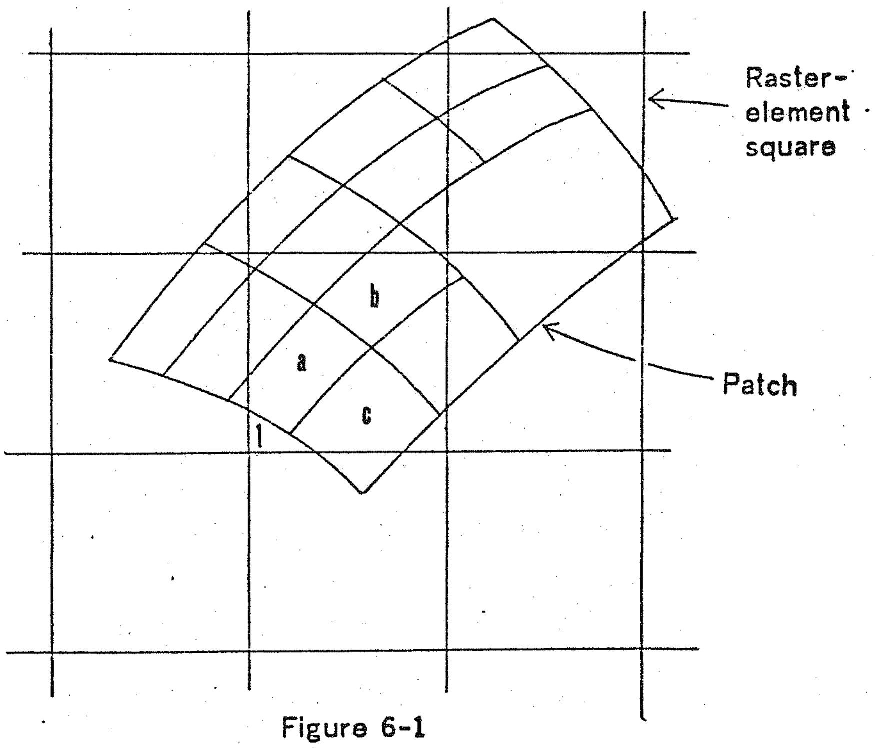


Figure 6-1

INTENSITY MODIFICATION

Once an intensity is in the buffer there may be several reasons to modify it; for example, transparency and shadows. If a new surface is transparent [10] then the intensity to be put into the buffer is some combination of the new intensity and the one already in the buffer. A typical formula might be New + (Old - New) * T where T is the transmittance which ranges from 0 for opaque to 1 for tranparent. Ad hoc variations on this formula can be made to get acceptable looking transparency. Transparent objects must be written into the buffer in the correct order, ie., close objects are written last.

Shadows can be made with the z buffer using "shadow-patches." A shadow-patch can be made by finding the silhouette of an object from the point of view of the light.

(See figure 6-2) The silhouette can be used to create shadow-patches that extend from

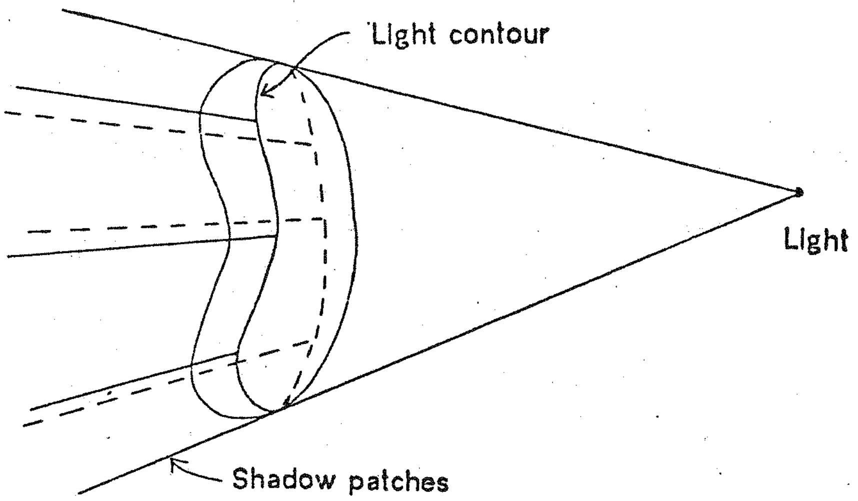


Figure 6-2

the silhouette away from the light. Front and back shadow-patches can then be paired up. Any object that lies between the two shadow-patches is in the shadow of the object. After the picture has been created the shadow-patch pairs can be split as bicubic patches with x, y, z-front, and z-back components. If the visible element in the z buffer lies in the shadow range then its intensity can be attenuated. The difficulties with this method are that one must find the silhouette, that the front and back shadow-patches must be matched up, and that diminishing the intensity does not correctly eliminate a highlight that should not appear in a shadow. It should be clear that although shadows can be made, it is not an easy problem.

CHAPTER SEVEN

SAMPLING, RASTERING, AND ALIASING

There are some inherent limitations with using a raster-display. The raster display cannot produce images with clean sharp edges or small (compared to the raster-element size) detail. Unfortunately, these limitations frequently lead to disturbing visual effects. We shall try to explain here the nature of these limitations and show steps that can be taken to alleviate the undesirable effects, especially with regard to the subdivision algorithm.

ALIASING

There are two different kinds of unwanted visual effects that result when using a raster-display -- "aliasing" and "rastering." The first -- aliasing -- is used to denote effects that result from sampling. Five manifestations of aliasing are

1. A "staircase effect" appears at the silhouettes of objects.
2. Small objects fall between the sample points and disappear.
3. In a motion picture, the slow smooth movement of an object appears as discrete jumps.
4. An image of a picket fence or similar regular pattern causes a moiré pattern to appear.
5. If a picture is mapped onto a surface then all of the above occur over the entire

surface.

As noted previously, a raster display cannot produce images of sharp edges or small detail. Aliasing occurs because we are sampling an image[1] which has information that the raster-display cannot possibly reproduce.

The phenomenon of aliasing can be better understood by considering a stagecoach movie . Note first that a movie camera can sample the real world 24 times a second. Suppose the camera views a stage coach as it starts and accelerates; the wheels moving faster and faster. Most readers will have witnessed that when the coach begins to move, the wheel appears to rotate in the right direction but as the wheel rotates faster it appears to go backwards, then stop, and finally to rotate forwards again even though the coach is always moving forwards. It is easy to understand that the wheel has a frequency of rotation. The movie film can accurately reproduce a rotational frequency of not more than twelve spokes per second. As the wheel rotates faster than that, the higher frequency is "aliased" as a low frequency which can be reproduced. The analog with sampled images is that an image may have intensity undulations that vary faster than the sampling rate and hence alias themselves as undulations that can be reproduced.

The field of signal-processing helps us understand aliasing even better. If a two-dimensional fourier transform of an image is taken prior to sampling, the result is a "picture" of the frequencies present in the image. Sharp edges and small objects

1. This image we are sampling exists only as a high resolution description in the computer, as contrasted with an actual photograph.

result in high frequencies. The raster display can reproduce only low frequencies; the upper limit on the frequency is determined by the resolution of the raster display. During the process of sampling, frequencies that are higher than those that can be reproduced are "folded" back onto those that can be and become indistinguishable from them; hence the term "aliasing."

"Anti-aliasing" will be used here to denote the process of reducing or eliminating the aliasing effects. An effective method for anti-aliasing is to eliminate from the image, prior to sampling, those components that cannot be reproduced or, in terms of signal processing, to filter out the high frequencies with a "low-pass filter." This filtering of an image could be thought of as a "smearing" operation. Sharp edges are smeared so that they are no longer sharp and therefore won't cause severe aliasing problems. The filtered image can then be sampled. The filtering and sampling process can be expressed in a diagram (see figure 7-1).

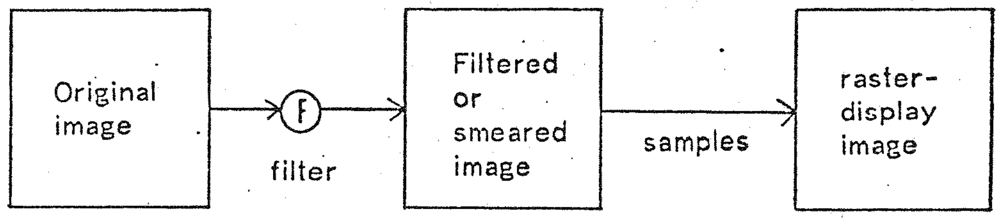


Figure 7-1

One method of filtering is to "convolve" the original image with a "two-dimensional fourier window" or "box window." With this method we in effect take a "box" that can cover one raster-element square and is one unit high and put the box on the original image. The box is multiplied by the intensities in the image -- which results in zero's everywhere but at the box -- and the resulting values are then integrated. This yields one value for that position of the box. The value in effect is the average of the intensities under the box. The box can be moved and a value calculated for some other point. As the box is moved over the entire image a new filtered image is created. The process of moving the box window over the image, multiplying, and integrating to form a new image is called "convolution" and can also be used with windows other than a box. The filtered image that results from using a box window no longer has sharp clean lines; much, but not all of the high frequency information is gone. Even though some of the high frequency information remains, a box filter is still good enough for most computer graphics purposes.

AREA SAMPLING

Since the filtered image will be sampled only at discrete points corresponding to the raster-elements it is necessary to calculate the filtered image only at those points. In other words, we can think of a raster-element as corresponding to some small square area of the original image and we only need to find the average intensity of the visible surfaces in that square. We shall call this particular form of filtering and sampling "area-sampling."

Area-sampling is the technique usually used in computer graphics to do anti-aliasing. Typically, when an edge of a polygon passes through a raster-element square, the intensity for the corresponding raster-element is some average of the polygon intensity and the intensity of polygon behind, weighted by their respective visible areas in the square. Most methods for anti-aliasing have been applied at the edges of polygons since the aliasing effects in the center of a polygon have usually been negligible.

Since there are several ways of filtering it is natural to ask "what is the best achievable anti-aliasing?" One should not be misled into thinking that area-sampling is the best anti-aliasing possible even though it is a considerable improvement over point-sampling of an unfiltered image. A better method, for example (although how much better is not known) would be to use a pyramid with a base that could cover four raster-element squares as a window for convolution instead of a box window. Unfortunately, "perfect" anti-aliasing is also undesirable because the filter necessary to make this possible also modifies the image in an undesirable way. The reasons for this and the answer to the above question are beyond the scope of this thesis. Methods for anti-aliasing are part of on-going research at the University of Utah.

RASTERING

Rastering occurs during the process of display regardless of the intensity values at each raster-element.[2] Rastering occurs when we can see the individual dots or scan-lines on the raster display. An example of rastering occurs in television where we frequently can see the scan-lines. If we can see the dots or scan-lines, then we are seeing something that is an artifact of the raster display and is undesirable information, thus the name "rastering."

---

2. There are actually two kinds of rastering -- "static" and "dynamic." The distinction between the two is beyond the scope of this thesis.

The meaning of the word "rastering" used here is not universal and in computer graphics it is frequently used to denote what we call here "aliasing." However in order to be consistent with the use of the word by the signal-processing research group at the University of Utah we shall take it to mean the effect that occurs in the process of actually displaying the raster.

"Anti-rastering" is the process of reducing or eliminating rastering. The practical method for anti-rastering is to defocus the CRT beam enough so that adjacent dots on the raster-display just merge. A picture of a "flat-field" on a raster-display should appear to be of uniform intensity with no dot or line structure.

ANTI-ALIASING FOR THE SUBDIVISION ALGORITHM

The subdivision algorithm can be modified to allow for area-sampling. Such a modification requires techniques for determining what is visible in each raster-element square and some method for storing and combining intensity values at each square to get the average. The modified algorithm has some drawbacks which will be discussed at the end of the chapter. Before presenting the modification, some groundwork needs to be laid and an "area-averaging algorithm" must be described.

One of the termination conditions described in chapter two required that a patch be approximated by a polygon to see if it was small enough. This same polygon can be used to do the area-sampling. After the finest subdivision, the polygon will be very small. We will require that no polygon cover more than four raster-element squares (see figure 7-2). In each square then, there will be some "piece" of the polygon.

The average intensity of all pieces visible in a square is needed to do area-sampling. Unfortunately, the pieces that logically belong to a square are not derived in immediate sequential order; that is, after one piece is found for a square, other areas of the screen may be worked on before finding another piece for that square. Some mechanism must be found for storing the piece intensities so that the average intensity can be found.

The problem is simplified if we make use of the following observations. In the large majority of raster-element squares all visible pieces come from the same patch. In a smaller, but still significant, number of squares, the pieces come from two patches -- namely at silhouettes and patch boundaries. A very small number have three or more patches visible in a single square. The method to be presented will do area-sampling for the first two cases correctly but is not guarenteed to be correct if more than two patches are visible in a single square.

The above implies that each piece must be identified with some patch. A patch code will be introduced for this purpose. The problem of identification is complicated by the fact that a patch may obscure itself; and in general it will, in regions near the silhouette. We can, however, differentiate between front and back-facing pieces by using an area-calculation method that gives negative or positive area depending on which way the piece faces. A bit can be set for a piece which indicates its facing direction.

The area-averaging algorithm requires that pieces be processed in z order. That requirement holds even within patches, ie., the four subpatches of a patch are sorted so that subdivision continues first with the more distant subpatch. If the z-buffer is used and the order is wrong, then the error will show up only at the silhouette where a staircase effect might become visible.

A large frame-buffer will be needed for the area-averaging algorithm. At every raster-element, storage will be needed for two intensities, I_1 and I_2, area, a facing direction bit, and a patch code. If the algorithm is used in conjunction with the z-buffer then storage for z is also needed. We shall describe values already in the buffer as "old" and the new values to be written as "new." The area of a single raster-element square will be taken to be unity so that is the largest value that can be stored in the area part of the raster-element. Initially I_1 will have the background intensity, the area bits will contain one's and the patch code will be zero.

The area-averaging algorithm is: A new piece is found with its area, code, direction bit, and intensity value which is weighted by the area. Then its corresponding raster-element is retrieved. The following are the possibilities.

1. If the new code is the same as the old and the direction bits are the same then the pieces come from the same patch. Add the areas and the new weighted intensity to the old value in I_1.
2. If the two codes are the same and the direction bits are different then the silhouette has been encountered and the accumulated area from that patch is about to be obscured by the new and subsequent pieces. Set the area to be the new area, the intensity I_1 to be the new weighted intensity, and the direction bit to the new direction value.
3. If the codes are different then a new piece or pieces will partially or completely

obscure the old pieces. Put the value of I_1 divided by area (to unweight it) into I_2, set I_1 to the new weighted intensity, set the area value, and set the patch code and direction bit. For displaying, the intensity will be $I_1 - I_2 * (1. - \text{area})$.

This algorithm has solved two problems: all of the pieces have been put together to allow mapping and at the silhouettes and boundaries the intensity is a combination from the two visible objects.

The subdivision algorithm of chapter two can now be modified to allow for area-sampling instead of point-sampling. Consider figure 7-2. Each square

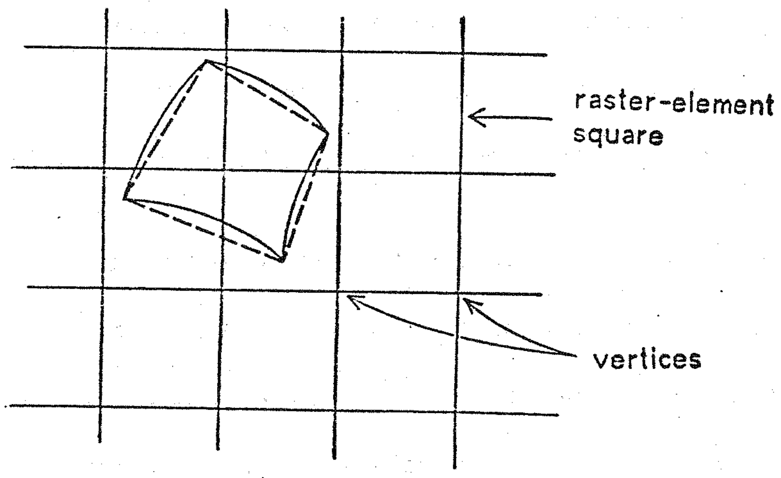


Figure 7-2

represents a raster-element square. Recall that a raster-element square is the area on the screen corresponding to one raster-element. The crossings of the horizontal and vertical lines which bound the squares will be called "vertices." The modification to the algorithm is that the patches will be subdivided until they cover at most one vertex (as opposed to a sample-point). An additional constraint on the termination condition is that the approximating polygon (the dotted lines in figure 7-2) lie within the area of the four squares adjoining the vertex.

The polygon that approximates the patch (the dotted lines in figure 7-2) will be used for the area calculations. The polygon must be divided into pieces that belong to each of the four squares. Each piece then is used with the area-averaging algorithm.

The algorithm presented is unsatisfactory in some ways: it requires a lot of memory, there is a lot of computation required, it is applied at every point instead of just where needed, it does not work with transparency, and there are several cases where it fails. On the other hand, mapping requires the ability to area-sample over the entire surface. It is not clear at this time just how much or how little is required to do acceptable anti-aliasing. Hopefully, the above discussion will lead to some cheaper or better methods for doing area-sampling.

CHAPTER EIGHT

CONCLUSION

The subdivision algorithm has been implemented in software on a PDP-10 at the University of Utah. Several pictures generated by the program are included in this thesis in Appendix D.

Table 7-1 lists some timing information about the generation of a few of the pictures. The initialization of the frame-buffer took about 7 seconds and displaying the frame buffer took about 28 seconds. The times listed below do not include initialization and display time.

OBJECT	PICTURE #	TIME (minutes:seconds)
single patch	2	1:17
glass	1	1:55
bottle	1	4:15
klein bottle	14	15:00

TABLE 7-1

It is natural to consider a hardware implementation because of the simplicity of the algorithm and the tremendous number of times those simple steps must be performed. The four components of such an implementation are:

1. The subdivider.
2. The stack.
3. The tester.
4. The shader.

The subdivider can split the patch into four pieces. Since subdividing a cubic takes three adds, the number of adds to subdivide a bicubic component is 30. The values must travel through the edges to the center so the values must pass through four adders. The fastest possible implementation would have a subdivider for each component.

There are several ways of trading off speed with cost. One subdivider could be used to subdivide each component sequentially. The system would just run slower. In addition, since each subdivider can be broken up into modules that combine register-squares, one could use just one module and give it two register-squares at a time to get a new square. Then the system would run even slower.

A stack would be needed to push the new squares onto. It needs to be large enough to handle the maximum level of subdivision, probably no greater than 15.

The tester must decide whether to display the patch or subdivide. It would check the x and y values at the corners. In addition, it may sort the four new patches if necessary either for the Newell algorithm or to do transparency. It is possible at some level of recursion to determine that no more sorting needs to be done.

The shader picks out the surface normal components, normalizes them, takes some dot products, and calculates the intensity for each raster-element. If the area-sampling method is used then the area of the patch in each raster-element square must also be calculated and the results merged with the information in the frame-buffer.

PROBLEMS

The most immediate problem is that of aliasing. One would hope that there is a cheaper or faster solution than the one presented here. For example, one might detect the silhouette by using the tangents and then area sample only at the silhouette of objects to calculate the right combination of intensities. An advantage of the subdivision algorithm is that a lot of information about the patch is available. The problem is to find a way to use that information to solve the aliasing problem.

Another problem is that bicubic patches may not adequately fill the needs of some people working with curved surfaces. It seems likely that the notion of subdividing can be applied to other curved surface schemes.

APPENDIX A

THE BICUBIC EQUATION

There are several different methods for generating bicubic patches. Each method is useful on different occasions. Bicubic equations are widely used in computer aided geometric design. Some good references are [11,12,13,14] with the article by George Peters in [11] being specially devoted to the bicubic patch.

Consider the simple cubic:

$x(t) = at^3 + bt^2 + ct + d$

This can be expressed in matrix notation:

$$x(t) = [t^3 \ \ t^2 \ \ t \ \ 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

A curve in space can be represented by the parametric vector equation $F(t) = [x(t) \ y(t) \ z(t)]$. Since each component is a parametric function of t and is treated the same as the other components, it is only necessary for us to consider one component x(t).

A patch is a function of two variables, u and v. $F(u,v) = [x(u,v) \ y(u,v) \ z(u,v)]$. Again only one component needs to be considered. The matrix notation for x(u,v) is:

$$x(u,v) = [u^3 \ \ u^2 \ \ u \ \ 1] \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where the a_{ij} are the coefficients of the equation just as a, b, c, and d were coefficients in the univariate case.

The problem then is to find the coefficients. There are many ways of doing this. We shall consider here only those ways that are local, that is, the changing of data only affects the coefficients of nearby patches. In order to find the coefficients of the simple cubic it is necessary to have four items of information. We can then transform that information into the coefficients by some four-by-four matrix M.

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = M \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} .$$

therefore

$$x(t) = [t^3 \quad t^2 \quad t \quad 1]\, M \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

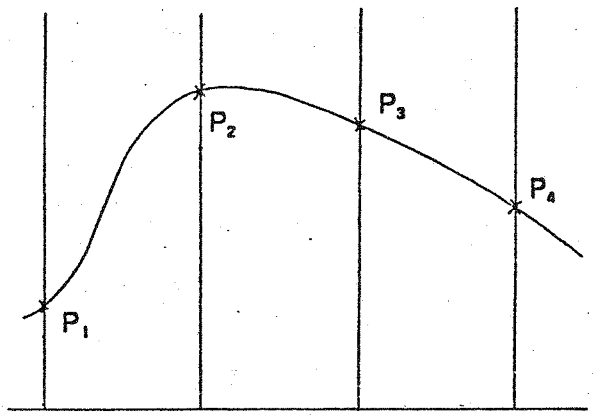
The P_i's can be some physically relevant items of information such as points or slopes. The matrix M is a constant matrix that corresponds to the particular kind of information chosen as the P_i's. It is important to note that this concept can be trivially extended to the bivariate case:

$$x(u,v) = [u^3 \quad u^2 \quad u \quad 1]\, M \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where the P_{ij} are relevant data such as points or slopes. For example the P_{ij} might be a four-by-four grid of points.

The balance of this appendix shall be devoted to showing what the M matrices are for different kinds of P's. When the P's are points they may be referred to as "control points." The examples shall be given using the univariate case so recall that the extension to bivariate patches is shown above.

1. SIMPLE CUBIC THROUGH 4 POINTS



Consider the four points P_1, P_2, P_3, and P_4. The cubic will pass through each point and $x(0)=P_1$, $x(1/3)=P_2$, $x(2/3)=P_3$, and $x(1)=P_4$. Then:

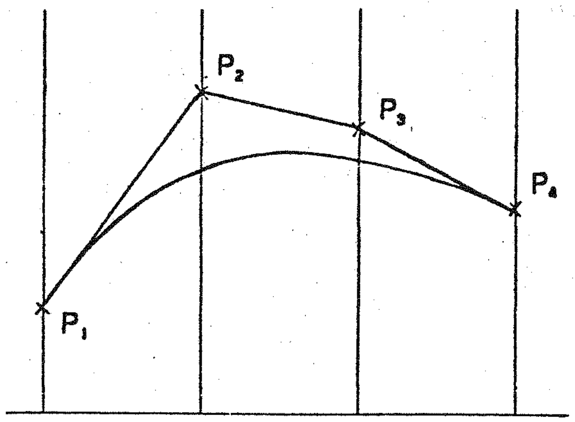
$$x(t) = [t^3 \;\; t^2 \;\; t \;\; 1] M_1 \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

and for this particular choice of the values of the independent variable,

$$M_1 = (1/2) \begin{bmatrix} -9 & 27 & -27 & 9 \\ 18 & -45 & 36 & -9 \\ -11 & 18 & -9 & 2 \\ 2 & 0 & 0 & 0 \end{bmatrix};$$

It is difficult with this scheme to connect two cubics at some point with c^1 continuity.

2. THE BEZIER OR BERNSTEIN CUBIC



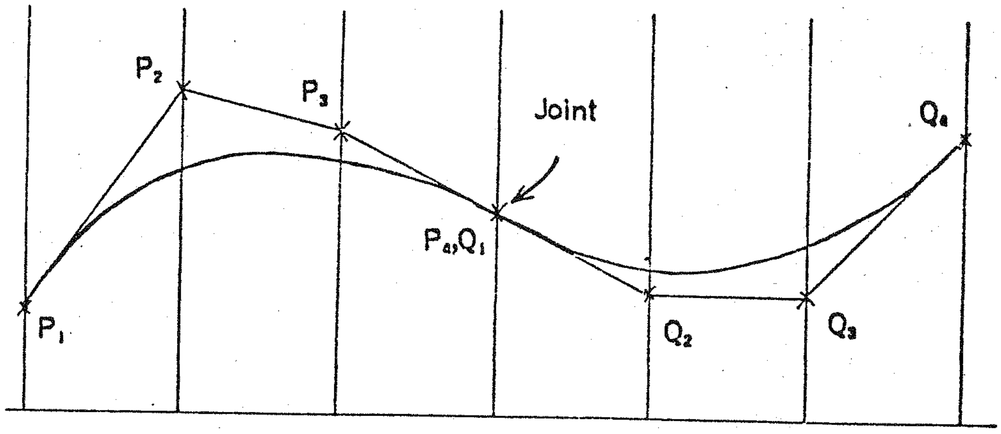
Consider the four points P_1, P_2, P_3, and P_4. The curve will pass through P_1 and P_4. The line from P_1 to P_2 is tangent to the curve at P_1 and the line from P_3 to P_4 is tangent at P_4. The length of the tangent vector at P_1 is three times the length of the line from P_1 to P_2. Similarly the length of the tangent vector at P_4 is three time the length of the line from P_3 to P_4. The curve is constrained to lie with the convex hull of the defining points.

$$x(t) = [t^3 \ t^2 \ t \ 1] M_2 \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

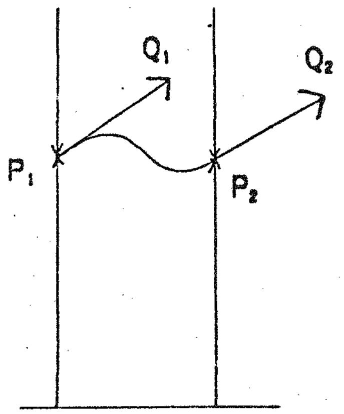
where

$$M_2 = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Two cubics can be joined with C_1 continuity if the control points at the joint are the same (quite obviously) and the two control points of both connecting ends are all colinear, ie., in the following diagram P_3, P_4=Q_1, and Q_2 are colinear.



3. THE HERMITE INTERPOLANT



Consider two points and two tangents, then:

$$x(t) = [t^3 \quad t^2 \quad t \quad 1]\, M_3 \begin{bmatrix} P_1 \\ P_2 \\ Q_1 \\ Q_2 \end{bmatrix}$$

where

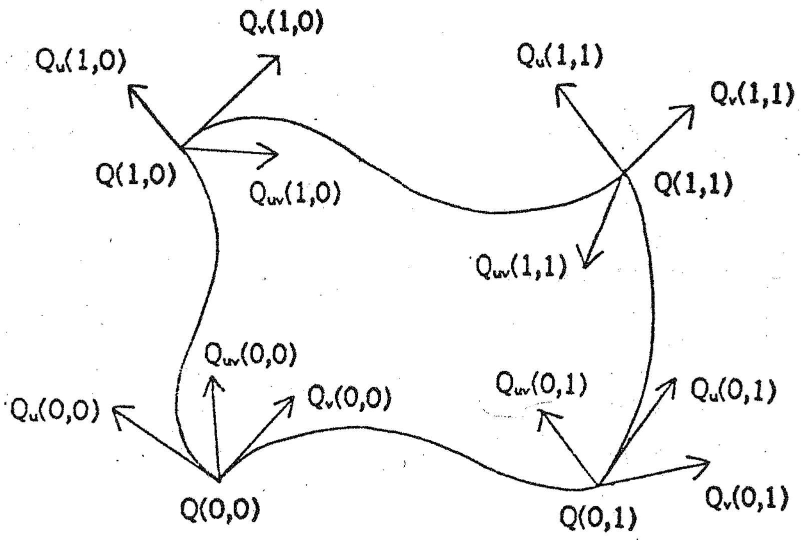
$$M_3 = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Two cubics can be connected easily with C^1 continuity if the tangents at the connecting points are the same.

The extension to a bivariate patch is not as straightforward as in the other cases. The bivariate patch is frequently called a bicubic Coons patch and sometimes the "hermite tensor-product bicubic surface." The elements of the P matrix are:

$$P = \left[\begin{array}{cc|cc} Q(0,0) & Q(0,1) & Q_v(0,0) & Q_v(0,1) \\ Q(1,0) & Q(1,1) & Q_v(1,0) & Q_v(1,1) \\ \hline Q_u(0,0) & Q_u(0,1) & Q_{uv}(0,0) & Q_{uv}(0,1) \\ Q_u(1,0) & Q_u(1,1) & Q_{uv}(1,0) & Q_{uv}(1,1) \end{array}\right]$$

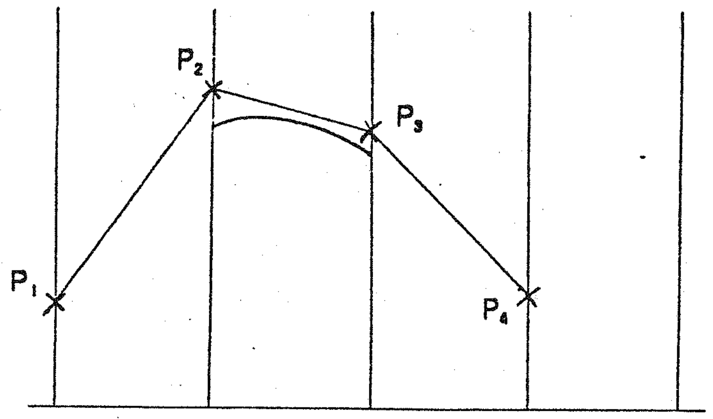
which corresponds to the patch



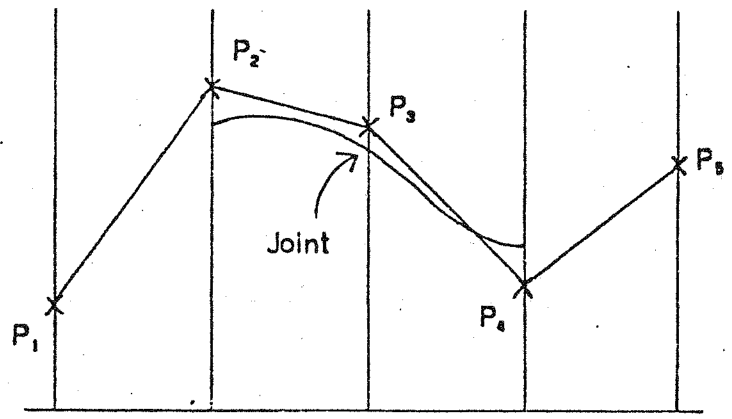
The Q are the corner points, the Q_u are the tangent vectors in the u direction, the Q_v are the tangents in the v direction, and the Q_{uv} are the cross derivatives which are frequently called the twist vectors. The twist vectors are sometimes set to zero which may cause "pseudo-flats" at the corners.

4. THE B-SPLINE

The cubic B-spline gives very nice looking curves and provides continuity of the second derivative. In general it does not interpolate its control points, but rather approximates them. The generated cubic is also constrained to lie within the convex hull of its defining points. Consider the four points P_1, P_2, P_3, and P_4:



A cubic curve can be generated that in general does not pass through any of its four control points. Now consider a fifth point P_5.



Another section of curve can be generated using points P_2, P_3, P_4, and P_5. The two curved pieces will be connected with c^2 continuity at the joint. The equation to generate a section is

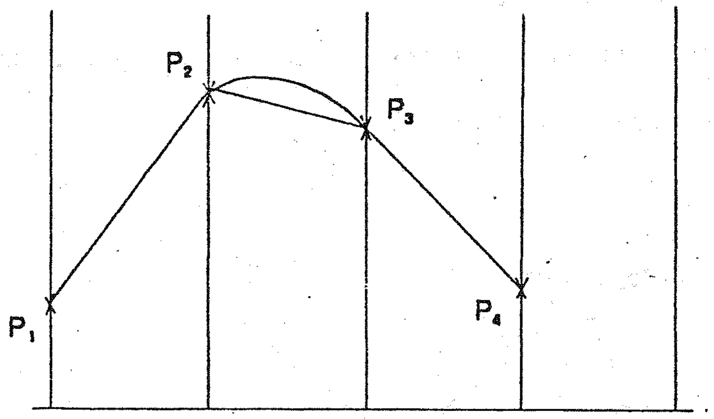
$$x(t) = [t^3 \;\; t^2 \;\; t \;\; 1]\, M_4 \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

where

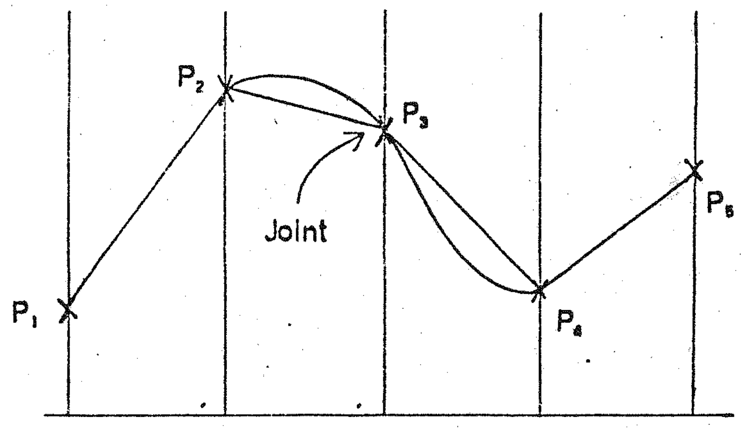
$$M_4 = (1/6) \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

5. THE CATMULL-ROM CUBIC SPLINE

This spline interpolates its control points and has continuity of the first derivative. Consider the four points P_1, P_2, P_3, and P_4.



A cubic can be generated that passes from point P_2 to P_3. Now consider a fifth point P_5.



Another piece of curve can be generated using points P_2, P_3, P_4, and P_5. The two sections will be connected with c^1 continuity. The equation to generate a section is:

$$x(t) = [t^3 \ \ t^2 \ \ t \ \ 1] \, M_5 \cdot \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

where

$$M_5 = (1/2) \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

APPENDIX B

RELATIONSHIP OF CORRECTION FACTORS TO BEZIER CONTROL POINTS

We can find the Beziér control points for a patch since the corner values and correction terms can be expressed in matrix form as shown in chapter three. Recall that the patch generated by the 16 control points is constrained to lie within the convex hull of those points. This is useful for clipping and determining when two patches might intersect.

Recall equation (3-5) C = SA where A is the matrix of coefficients. Of course we can go the other way by noting $A = S^{-1}C$. If we have four points d_1, d_2, d_3, and d_4 and

$$D = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

then the coefficients for the Beziér cubic for those points are A=BD where B is the four-by-four matrix given in Appendix A in the section on Beziér cubics. If we put the relationship into equation 3-5 then C=SA=SBD. Therefore:

$$\text{(B-2)} \qquad D = B^{-1}S^{-1}C$$

giving the control points D as a function of the correction matrix C.

This analysis can be extended to surfaces. Let C be the four-by-four correction matrix for a patch. We expect it to contain the same values listed in figure 4-6. Let M be the four-by-four matrix of coefficients for the bicubic patch. Then

(B-3) $C = SMS^T.$

Let P be the four-by-four matrix of the 16 Bezier control points for the patch. Then $M = BPB^T$ and it follows that $C = SBPB^TS^T$. Therefore:

(B-4) $P = B^{-1}S^{-1}C(S^T)^{-1}(B^T)^{-1}$

where

$$B^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

$$B^{-1}S^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2/3 & -2/9 & 1/3 & -1/9 \\ 1/3 & -1/9 & 2/3 & -2/9 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

APPENDIX C

APPROXIMATING THE BICUBIC NORMAL EQUATION

The normal vector to a bicubic patch can be found by taking the cross product of the tangent vector in the the u direction and the tangent vector in the v direction and can be shown to be quintic. It is desirable to approximate the quintic normal equation with a bicubic equation because a bicubic equation is easier to work with.

The x component of the surface vector is:

$$x = [u^3 \;\; u^2 \;\; u \;\; 1]\, M_x \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where M_x is the matrix of coefficients for x. The derivative in the u direction is:

$$x_u = [3u^3 \;\; 2u \;\; 1 \;\; 0]\, M_x \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

and the derivative in the v direction is:

$$x_v = [u^3 \;\; u^2 \;\; u \;\; 1]\, M_x \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

For simplicity we shall define:

$U = [u^3 \; u^2 \; u \; 1]$

$U' = [3u^2 \; 2u \; 1 \; 0]$

$U'' = [6u \; 2 \; 0 \; 0]$

$$V = \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

$$V' = \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

$$V'' = \begin{bmatrix} 6v \\ 2 \\ 0 \\ 0 \end{bmatrix}$$

Therefore $X_u = U'M_xV$ and $X_v = UM_xV'$. The y and z components are treated similarly of course. The tangent in the u direction is $[x_u\ y_u\ z_u]$ and the tangent in the v direction is $[x_v\ y_v\ z_v]$.

We need to find the normal vector $[x_n\ y_n\ z_n]$. The normal vector can be found by taking the cross product:

$x_n(u,v) = y_uz_v - y_vz_u$

$y_n(u,v) = z_ux_v - z_vx_u$

$z_n(u,v) = x_uy_v - x_vy_u$

but since $x_u = U'M_xV$, $x_v = M_xV'$, $y_u = U'M_yV$, etc., we can write:

(A-1) $x_n(u,v) = U'M_yVUM_zV' - UM_yV'U'M_zV$

(A-2) $y_n(u,v) = U'M_zVUM_xV' - UM_zV'U'M_xV$

(A-3) $z_n(u,v) = U'M_xVUM_yV' - UM_xV'U'M_yV$

It should be apparent on close examination of the equations A1-A3 that each component is, as asserted, a fifth degree polynomial in u and v. Let us consider only the x component of the normal. In order to approximate the normal vector equation with a bicubic normal vector equation we require that the bicubic normal have the same:

1. values at the corners, $x_n(u,v)$
2. derivatives in the u direction at the corners, $dx_n(u,v)/du$
3. derivatives in the v direction at the corners, $dx_n(u,v)/dv$
4. cross derivatives at the corners, $d^2x_n(u,v)/dudv$.

If we group this data in a matrix we have:

$$P_x = \left[\begin{array}{cc|cc} x_n(0,0) & x_n(0,1) & \frac{dx_n(0,0)}{dv} & \frac{dx_n(0,1)}{dv} \\ x_n(1,0) & x_n(1,1) & \frac{dx_n(1,0)}{dv} & \frac{dx_n(1,1)}{dv} \\ \hline \frac{dx_n(0,0)}{du} & \frac{dx_n(0,1)}{dv} & \frac{d^2x_n(0,0)}{dudv} & \frac{d^2x_n(0,1)}{dudv} \\ \frac{dx_n(1,0)}{du} & \frac{dx_n(1,1)}{du} & \frac{d^2x_n(1,0)}{dudv} & \frac{d^2x_n(1,1)}{dudv} \end{array}\right]$$

The form of this matrix is the same form as the data matrix for a bicubic Coons patch. Therefore we can use Coons magic matrix:

$$C = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

So the x component of the bicubic normal is:

(A-5) $x = CP_xC^T$

The quintic function and its derivatives can be written more explicitly as:

(A-6) $x_n(u,v) = U'M_yVUM_zV' - UM_yV'U'M_zV$

(A-7) $dx_n(u,v)/du = U''M_yVUM_zV' + U'M_yVU'M_zV'$
$-U'M_yV'U'M_zV - UM_yV'U''M_zV$

(A-8) $dx(u,v)/dv = U'M_yV'UM_zV' + U'M_yVUM_zV''$
$- UM_yV''U'M_zV - UM_yV'U'M_zV'$

(A-9) $d^2x(u,v)/dudv = U''M_yV'UM_zV' + U''M_yVUM_zV''$
$+ U'M_yV'U'M_zV' + U'M_yVU'M_zV''$
$- U'M_yV''U'M_zV - U'M_yV'U'M_zV'$
$- UM_yV''U''M_zV - UM_yV'U''M_zV'$

The values of these equations at u=0,1 and v=0,1 can then be substituted into the appropriate places in equation A-4.

Rather than rewrite equations A-4 through A-9 for the y and z components just note that for y we can use the substitutions

1. y replaces x
2. z replaces y
3. x replaces z

and for z

1. z replaces x

2. x replaces y

3. y replaces z

APPENDIX D

PICTURES

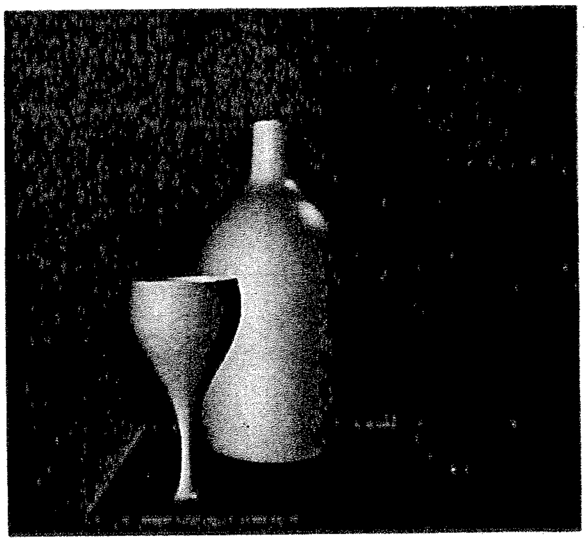
The pictures in this appendix were made on the high-precision CRT at the University of Utah. All pictures were made at 512 resolution. The beam was slightly overfocused.

The discontinuities in the shading on the everting spheres are caused by first derivative discontinuities in the surface description and not by the algorithm. The roughness at the intersections are a result of insufficient z resolution. The front clipping plane was much too close to the eye.

The area-sampled klein bottle clearly illustrates deficiencies in the area-sampling algorithm presented in chapter seven. However, the algorithm works very well for mapping.

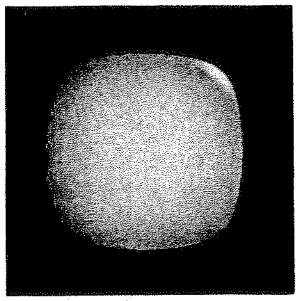
The photographs used for mapping were scanned into the computer with a scanning device at the University of Utah. Only lack of time prevented a more elaborate demonstration of the power of mapping.

The shading discontinuities in the brick cylinder occur because the original brick wall was not evenly lit.



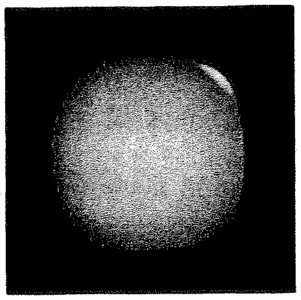
Picture 1

A bottle and glass. The bottle has 32 patches.



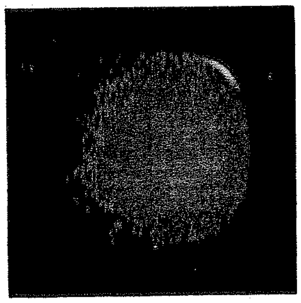
Picture 2

A single patch demonstrating the aliasing that results from point-sampling. Observe the edges.



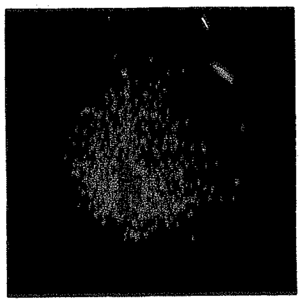
Picture 3

A patch demonstrating area-sampling. Again observe the edges.



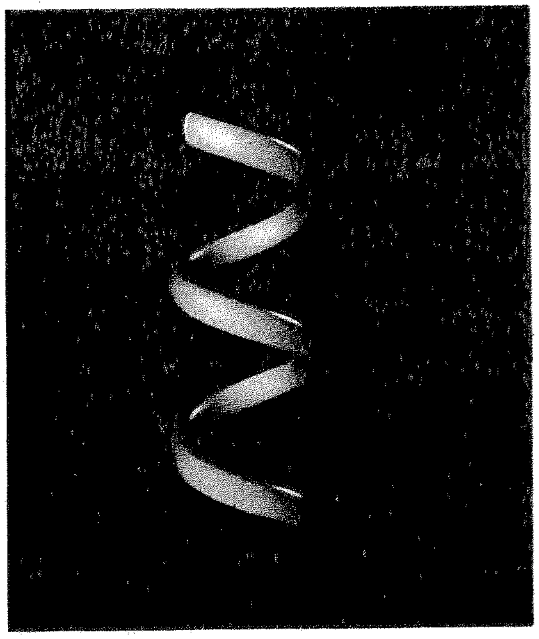
Picture 4

Point-sampled and over-focused.



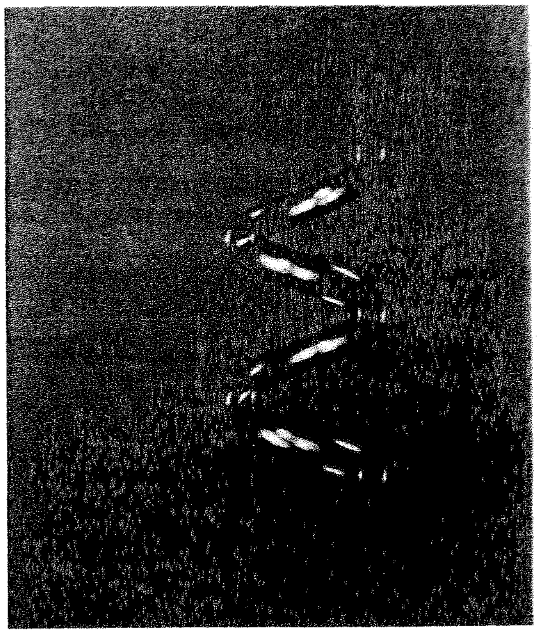
Picture 5

Area-sampled and over-focused.



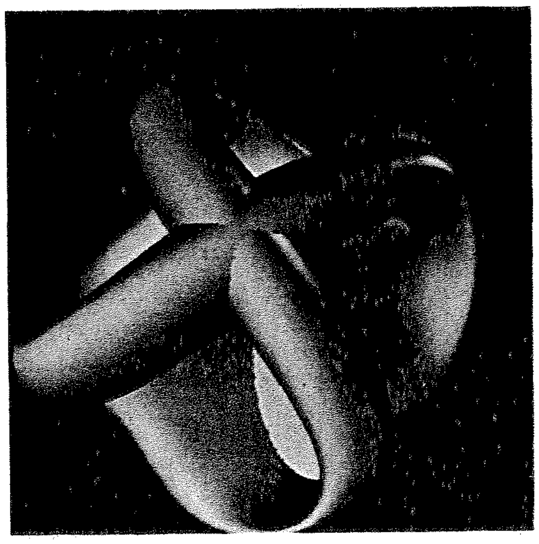
Picture 6

A spiral tube.



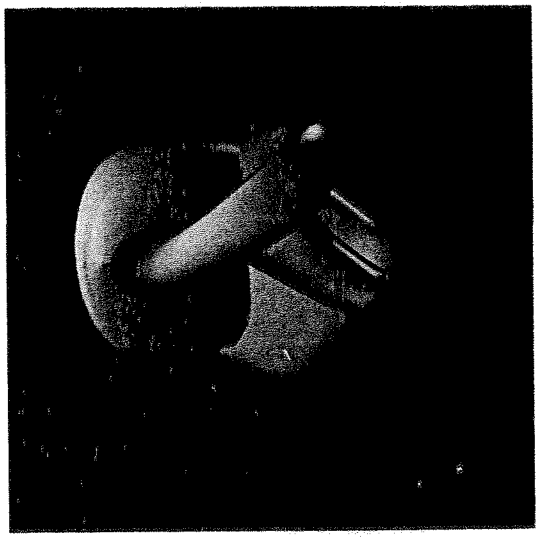
Picture 7

A transparent spiral tube.



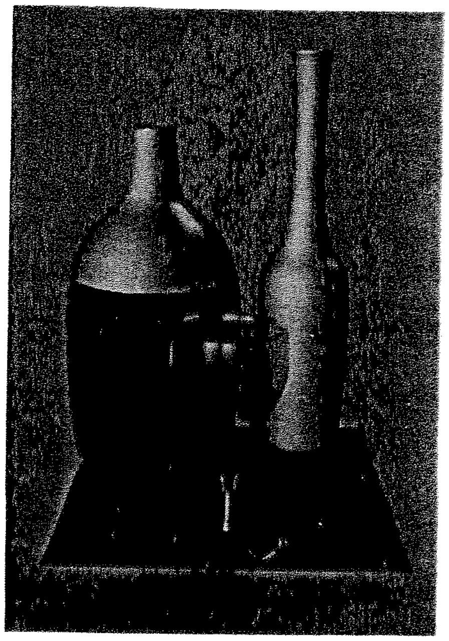
Picture 8

A sphere midway through its eversion. Designed by Dr. Nelson Max at Carnegie-Mellon University using bicubic Coons patches.



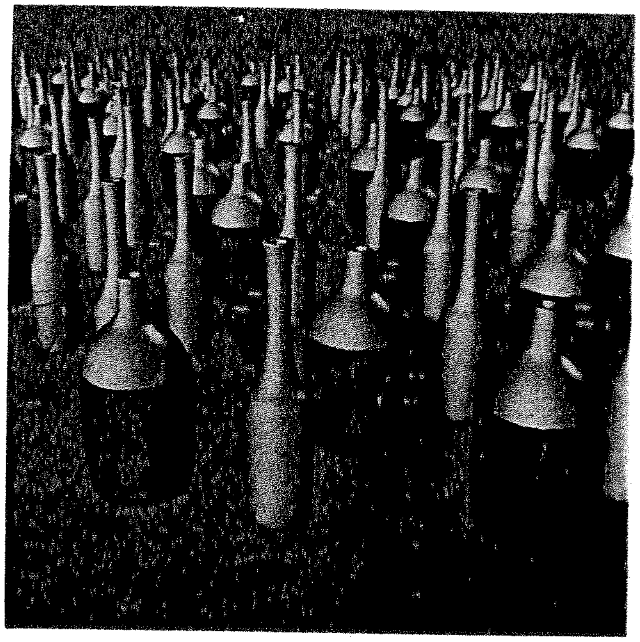
Picture 9

Another view of an everting sphere.



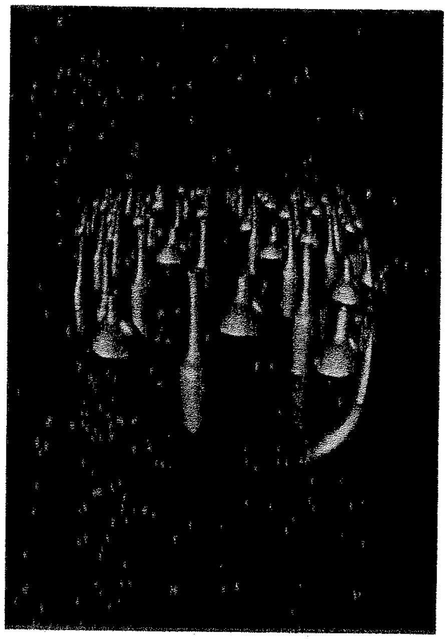
Picture 10

Two bottles and a glass.



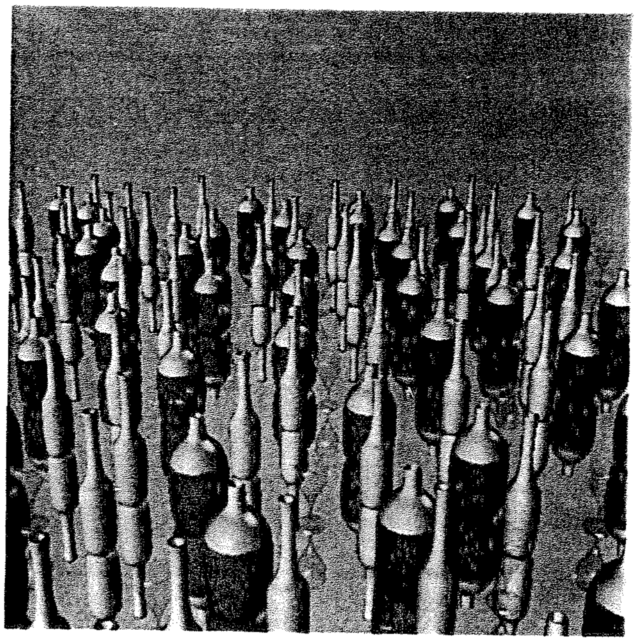
Picture 11

142 bottles and glasses.



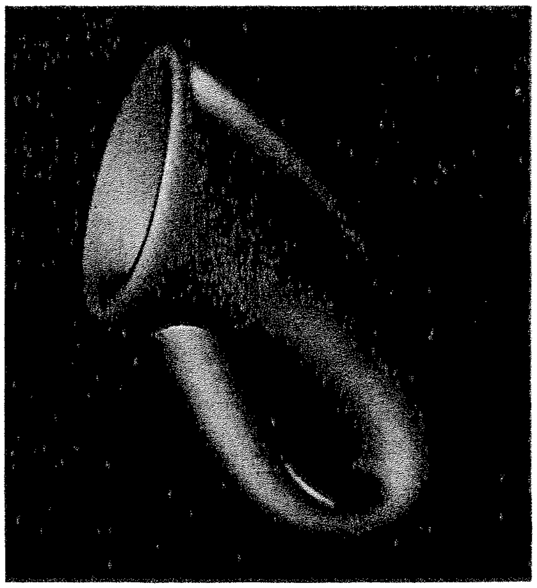
Picture 12

The bottles scene mapped onto a curved patch.



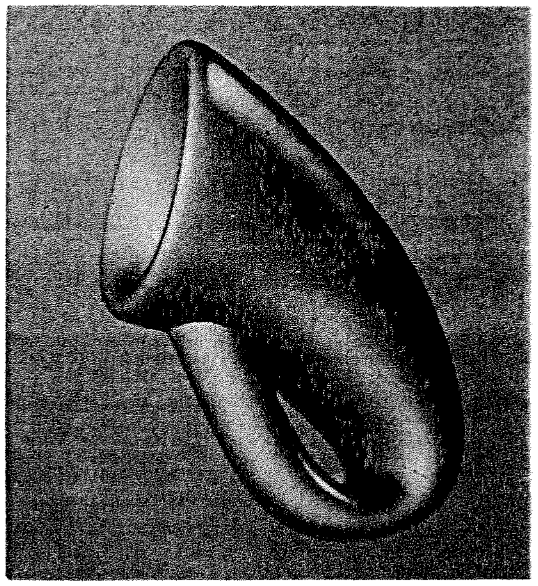
Picture 13

The bottles with simulated reflection.



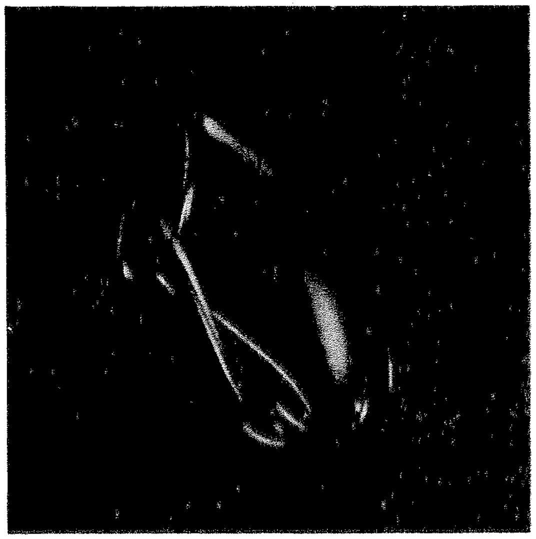
Picture 14

Klein bottle. Designed by Dr. James Clark using B-Splines.



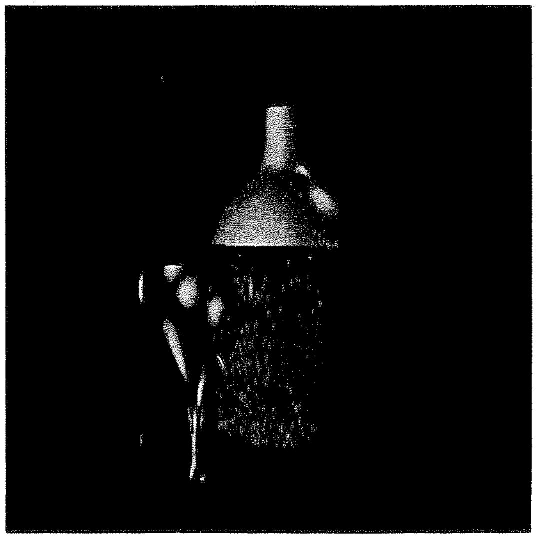
Picture 15

Klein bottle with area-sampling used. Notice the occasional failure at the silhouette to do the anti-aliasing correctly.



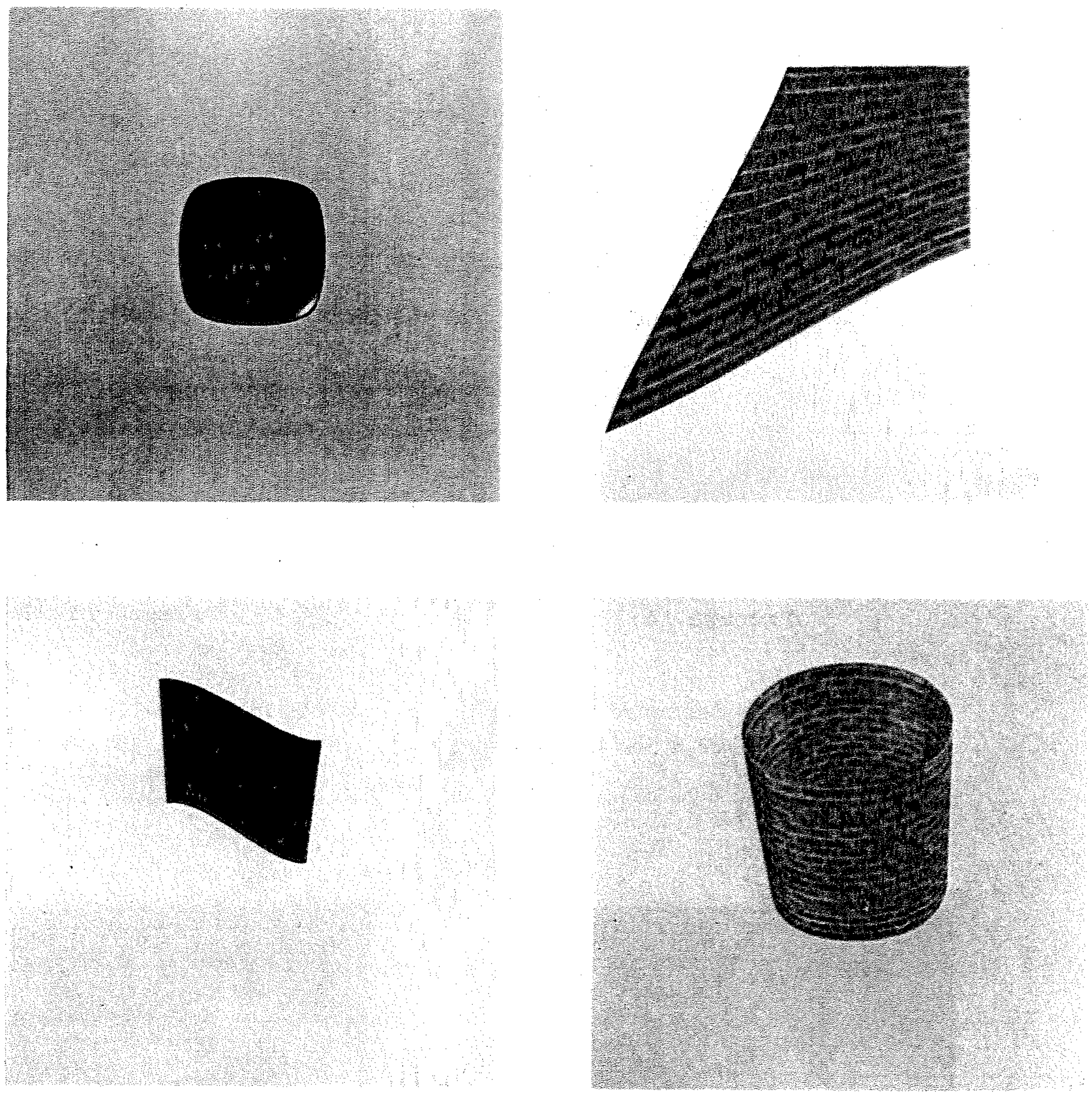
Picture 16

Klein bottle with transparency.



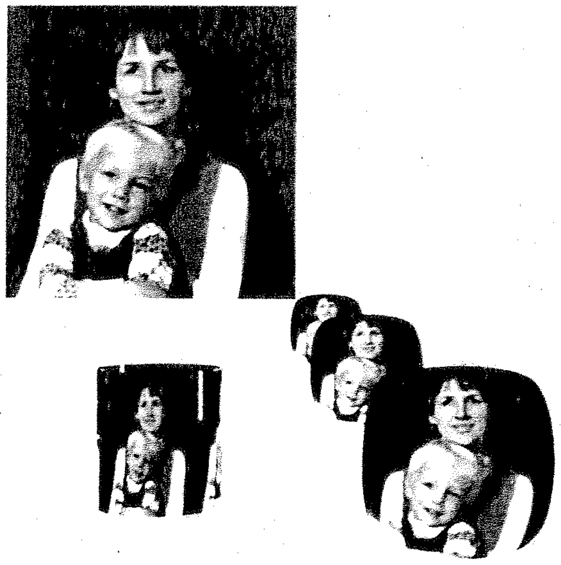
Picture 17

The bottle and glass with transparency added to the glass and "color" to the bottle.



Picture 18

The brick image mapped
respectively onto a single
rounded patch, a stretched patch,
an S curved patch, and a cylinder
of four patches.



Picture 19

A picture of the author's family mapped onto several patches.

Picture 20

A picture of the author's wife mapped onto a cylinder.

Picture 21

A photograph of a hill mapped onto a curved patch.

Picture 22

Winnie the Poo and Tigger on a curved patch.

LIST OF REFERENCES

[1] I.E. Sutherland, R.F. Sproull, R.A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," ACM Computing Surveys, Volume 6, number 1, March 1974.

[3] H. Gouraud, "Computer Display of Curved Surfaces," Department of Computer Science, University of Utah, UTEC-CSc-71-113, June 1971. Also in IEEE, TC-20 June 1971, page 623.

[4] Bui Tuong-Phong, "Illumination for Computer-Generated Images," Department of Computer Science, University of Utah, UTEC-CSs-73-129, July 1973.

[5] MAGI, Mathematical Applications Group Inc., "3-D Simulated Graphics," Datamation, February 14, 1968, p. 69.

[6] R. Mahl, "Visible Surface Algorithm for quadric patches," IEEE, TC-21, p. 1, January, 1972.

[7] W.M. Newman and R.F. Sproull, Principles of Interactive Graphics, Chapter 12, McGraw-Hill, 1973.

[8] I.E. Sutherland and G.W. Hodgman, "Reentrant Polygon Clipping," CACM, page 43, January 1974.

[9] J.E. Warnock, "A Hidden-Line Algorithm for Halftone Picture Representation," Department of Computer Science, University of Utah, TR 4-15, 1969.

[10] M.E. Newell, R.G. Newell, T.L. Sancha, "A New Approach to the Shaded Picture Problem," Proceedings of the ACM, 1973 National Conference.

[11] R.E. Barnhill and R.F. Riesenfeld, Computer Aided Geometric Design, Academic Press, 1975.

[12] A.R. Forrest, "On Coons and Other Methods for the Representation of Curved Surfaces," Computer Graphics and Image Processing, page 341, 1972. (Contains an extensive bibliography.)

[13] A.R. Forrest, "Appendix 2 -- Coons' Surfaces," Numerical Control -- Mathematics and Applications, P.E. Bézier, London: John Wiley and Sons, 1972.

[14] J.H. Clark, "B-spline Surface Design," Dissertation, Computer Science Department, University of Utah, December 1974.

VITA

Name	Edwin Earl Catmull
Birthplace	Parkersburg, West Virginia
Birthdate	March 31, 1945
High School	Granite High School Salt Lake City, Utah
Degrees	B.A., Physics University of Utah 1969
	B.A., Computer Science University of Utah 1969