This is to certify that the

thesis entitled

COLOR GRAPHICS IN ENGINEERING DESIGN

presented by

Robert Alan Coviak

has been accepted towards fulfillment
of the requirements for

MS degree in ME

[illegible]
Major professor

Date 8-10-81

COLOR GRAPHICS IN ENGINEERING DESIGN

By

Robert Alan Coviak

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

1981

ABSTRACT

COLOR GRAPHICS IN ENGINEERING DESIGN

By

Robert Alan Coviak

The interest in color graphics has been growing rapidly in the past few years. A great deal of effort has been expended finding ways to produce high quality, visually pleasing shaded images. Color graphics can also be viewed as a tool to display engineering information. Shaded color displays of images open up another dimension not possible in traditional line drawing methods.

This thesis reviews the techniques which have been developed to produce visually pleasing shaded images. It is then shown that these techniques may be less than ideal for some important engineering applications and alternatives are suggested which are better suited to engineering use.

To my wife, Becky, whose love, patience, and sacrifice made this work possible, and whose presence gave it meaning.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. James E. Bernard, my major professor, for his friendship and advice througout my graduate program.

Also, I wish to thank Dr. Erik Goodman for his thoughtful comments throughout the preparation of this thesis, and to Dr. Fred Fink for his input regarding its final form.

Many thanks, also, to Jan Swift for her excellent typing and organizational help, and to Judy Duncan for her last minute typing to help me meet my deadline.

Finally, special thanks to my wife, Becky, and to my parents, Norbert and Dolores, for their loving support and guidance.

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

Chapter 1

INTRODUCTION

The interest in color graphics has been growing rapidly in the past few years. A great deal of effort has been expended finding ways to produce high quality, visually pleasing shaded images. There has also been a lot of interest in producing realistic animated films.

We can also view color graphics as a tool to carry and display engineering information. Shaded color displays of images open up another dimension not possible in conventional line drawing methods. This is particularly useful in design problems involving parametric surfaces. We want to be able to view the surfaces from various angles and with various light source orientations. Color can then be used to assess several types of surface conditions that are extremely difficult, if not impossible, to assess using conventional display methods such as flow lines and sections. This color application demands a faithful representation of the characteristics of the image more for utility than for realism.

In Chapter 2 we will review some terminology in the graphics area. Methods of intensity calculation including various shading rules, reflections, shadows, and ray tracing will be discussed in Chapter 3. Chapter 4 will discuss three methods for displaying scenes composed of planar polygons. Three methods for displaying smoothly shaded parametric surfaces will be reviewed in Chapter 5. Chapter 6 presents recent research at the Case Center for Computer-Aided Design involving a point by point display technique for parametric surfaces.

Chapter 2
TERMINOLOGY

In this chapter we will explain some of the terminology that is commonly used in color graphics.

The object space is the coordinate system in which the objects in a scene were originally defined. The image space is the coordinate system related to the display screen. This coordinate system is shown in Figure 2-1. Linear transformations relate object space to image space.

A raster scan device is a digital display device. A television set is an example of this type of device. The smallest element of the raster scan display screen is a pixel. Pixels are often thought of as points, but should more appropriately be thought of as rectangles. The shade or color at each pixel can be set independently of other pixels. A row of pixels across the screen is a scan line. On a raster scan device, the screen is displayed in scan line order. Pixels are displayed starting at the upper left corner of the screen and proceeding a scan line at a time down the screen, as in Figure 2-2a. To reduce flicker, many raster screens use an interlaced scan, shown in Figure 2-2b.

The current color value of each pixel is stored in a frame buffer. This buffer has one entry for each pixel on the screen.

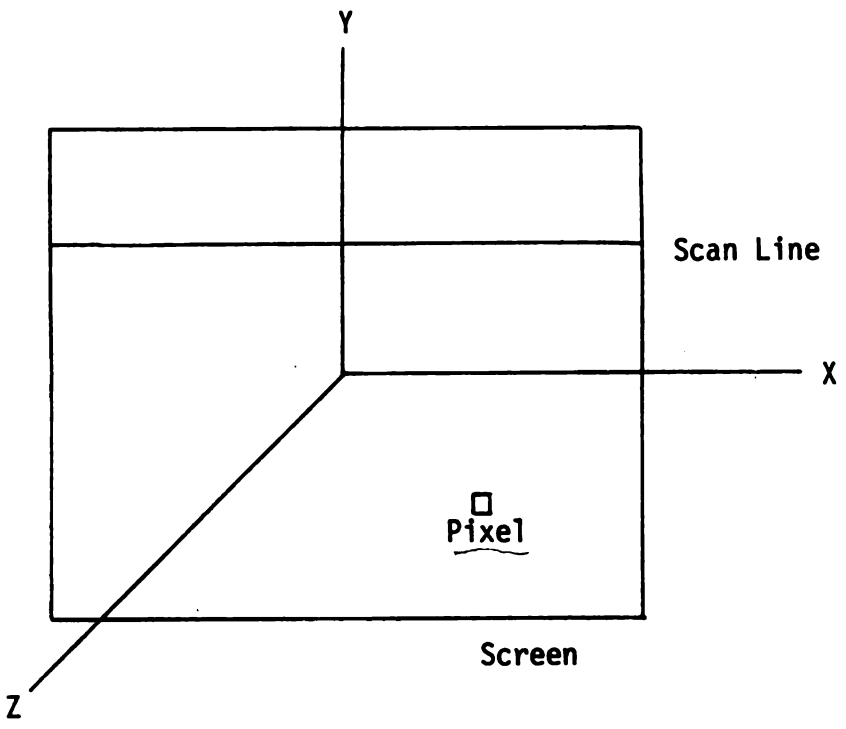


Figure 2-1. Display Screen Environment.

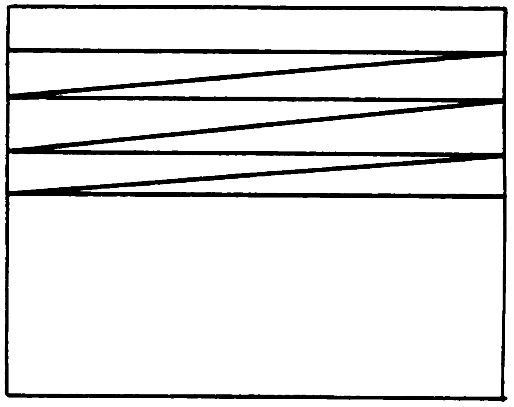


Figure 2-2a. Scan Line Order.

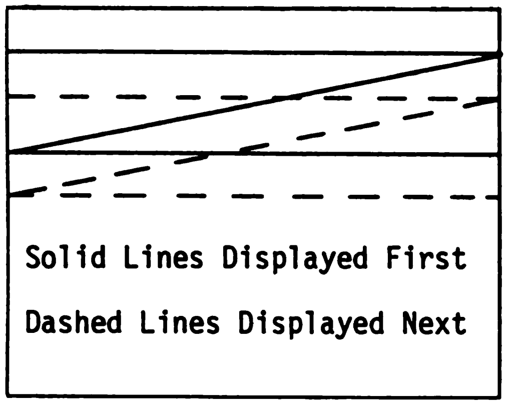


Figure 2-2b. Interlace Scan.

These entries may consist of from one bit for a monochrome display, up to eight or more bits for a color display. The depth buffer is similar to the frame buffer. The screen space Z coordinate of each point being displayed is stored in the depth buffer. This allows the depth buffer to be used for hidden surface removal.

Because raster scan devices are digital devices, we are confronted with aliasing. Aliasing consists of a group of apparent defects including jagged lines and lost detail caused by insufficient sampling of the objects being displayed. Reference [6] presents a comprehensive discussion of this topic.

The image displayed by the raster scan device can be shaded according to any of a wide variety of methods. The next chapter will discuss some of these methods.

Chapter 3

SHADING

This section describes the shading rules that are currently in wide use. These rules can be divided into two major groups: those that use only local information to shade a pixel, and those that use global information.

3.1 Local Methods

In the local category, there are pointwise methods and methods that use interpolation to find the shade at a pixel. We will examine the pointwise methods first.

3.1.1 Pointwise Intensity Functions

The simplest method is

$$I = r \cdot d \qquad (3.1)$$

where d is the diffuse illumination of the scene, and r (≤ 1) is the reflectance of the object. This simple rule yields constant shade over an object, regardless of the position of the viewer, and does not allow for point light sources. To allow for point light sources, we can use:

$$I = r \cdot \cos(\theta) \cdot s \tag{3.2}$$

where θ is the angle between the incoming light ray and the normal to the surface, as in Figure 3-1. The intensity of the light source is s. In this model, the position of the viewer does not affect the shade at a point and hence disallows specular reflection or highlights.

To allow for specular reflection, we use:

$$I = P(\theta)\cos(\beta)^n \tag{3.3}$$

where β is the angle between the reflected light ray and the viewing ray, and $P(\theta)$ is the specular reflectance function of the surface. The use of n allows different surfaces to have different sized highlight areas. The value of n is determined by trial and error, based on how shiny the surface is.

Combining Equations (3.1), (3.2) and (3.3) yields:

$$I = s[P(\theta)\cos(\beta)^n + r \cdot \cos(\theta)] + r \cdot d \tag{3.4}$$

This model allows for diffuse and point light sources, viewpoint dependent shading, and highlights. More detailed relationships are required to account for wavelength dependent effects, colored light sources, or surfaces which have an inherent color.

When the equations above are applied to surface based algorithms that do not use interpolation, very realistic images are possible, limited only by the sophistication of the intensity

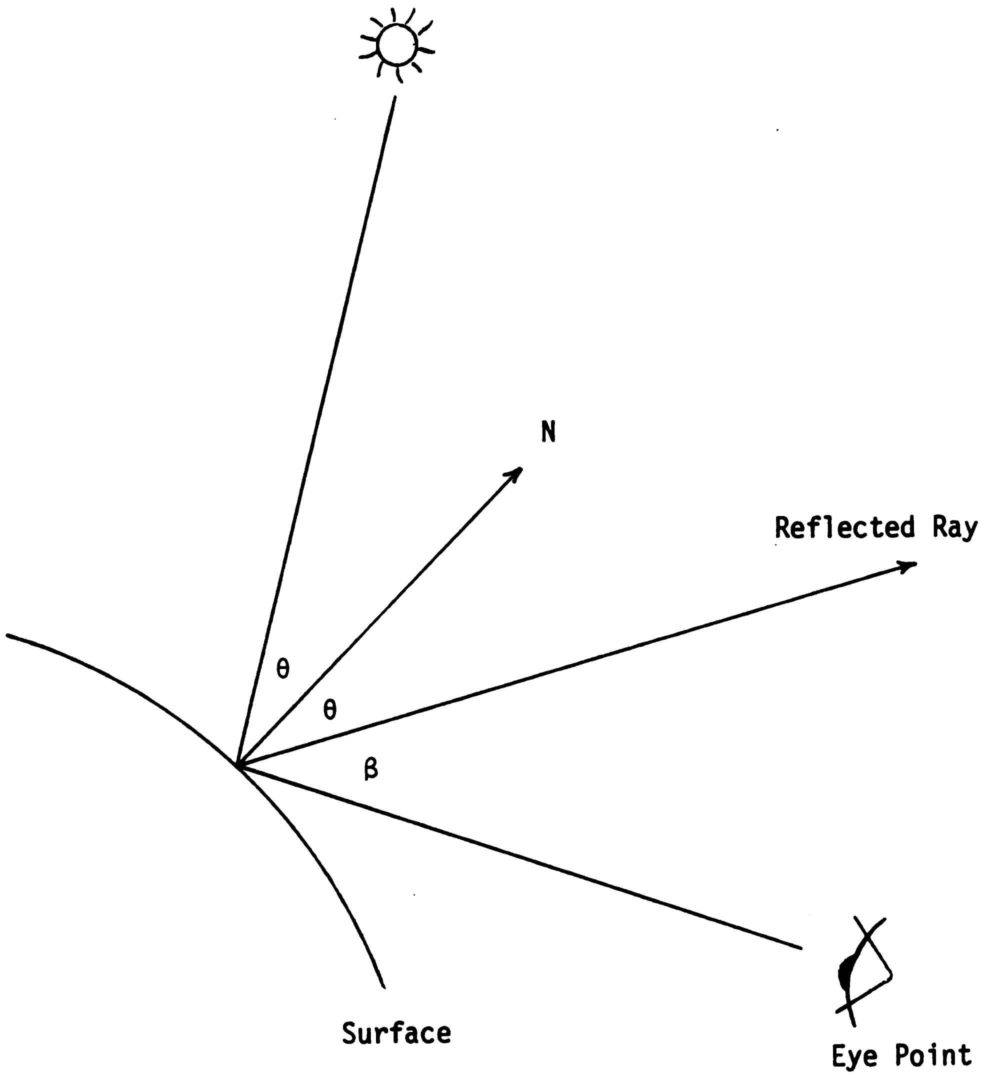


Figure 3-1. Shading Geometry.

function. This is the case because all the information needed to shade a certain point is available at that point. When these intensity functions are applied to the polygon based algorithms, however, they give the objects in the scene a distinctly faceted appearance. This is fine for block type scenes. But for curved surfaces, this is not satisfactory. When displaying curved surfaces, we do not want to inject discontinuities that do not exist in the data base. To remove these facets and achieve smooth shading, more sophisticated methods are necessary.

3.1.2 Gouraud Shading

Gouraud [7] developed a method to produce smoothly shaded polygons. To achieve continuity of shade across polygon boundaries, normals at A, B, C, and D must be determined, as in Figure 3-2. These normals must be determined uniquely for each vertex. The normals can be computed as the average of the normals of the adjacent faces. Or preferably, if the necessary surface descriptions are available, they can be computed directly from the surface.

Referring to Figure 3-2, the quantities S(A), S(B), S(C), and S(D) can be computed using any of the pointwise shading rules. The shade of edge points at E and F are computed by interpolating between the appropriate vertices. Then for a point on a scan line on a visible segment,

$$S(G) = (1-\alpha)S(E) + \alpha S(F) \tag{3.5}$$

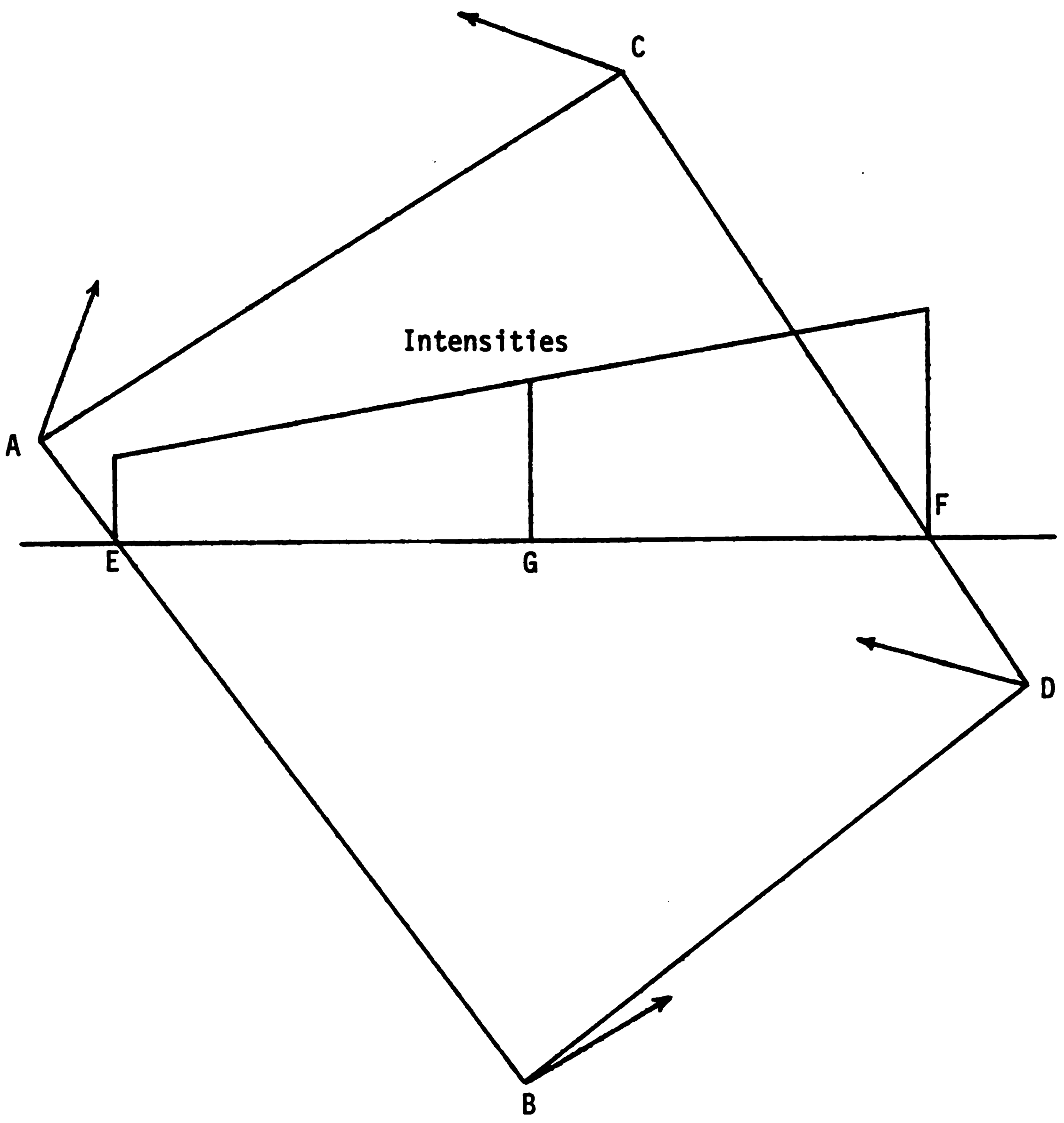


Figure 3-2. Gouraud Shading.

where

$$\alpha = (X(G)-X(E))/(X(P)-X(E)). \tag{3.6}$$

This will yield continuous shading over the surface. The shading has a discontinuous first derivative. This discontinuity causes a visually apparent flaw known as the Mach band effect [11]. Also, special action must be taken to defeat the smooth shading where one needs to display an abrupt change in the surface normal vectors. This could occur, for instance, due to a corner or crease in the scene.

3.1.3 Phong Shading

Phong [2] developed a method to reduce, though not eliminate, the disturbing Mach band effect which is evident in Gouraud shading. Phong's method is similar to Gouraud's, but Phong interpolates normal vectors rather than intensities. In Figure 3-3, the normal vectors at edge points E and F are computed by interpolating between the normals at A, B, C, and D. The normal vector at point G is then computed by interpolating between the normals at E and F. This interpolated normal is then used in a shading rule to calculate the intensity for the pixel at G. This reduces the noticeability of the Mach bands. As with Gouraud shading, special attention must be paid to areas where the user desires discontinuity in shade.

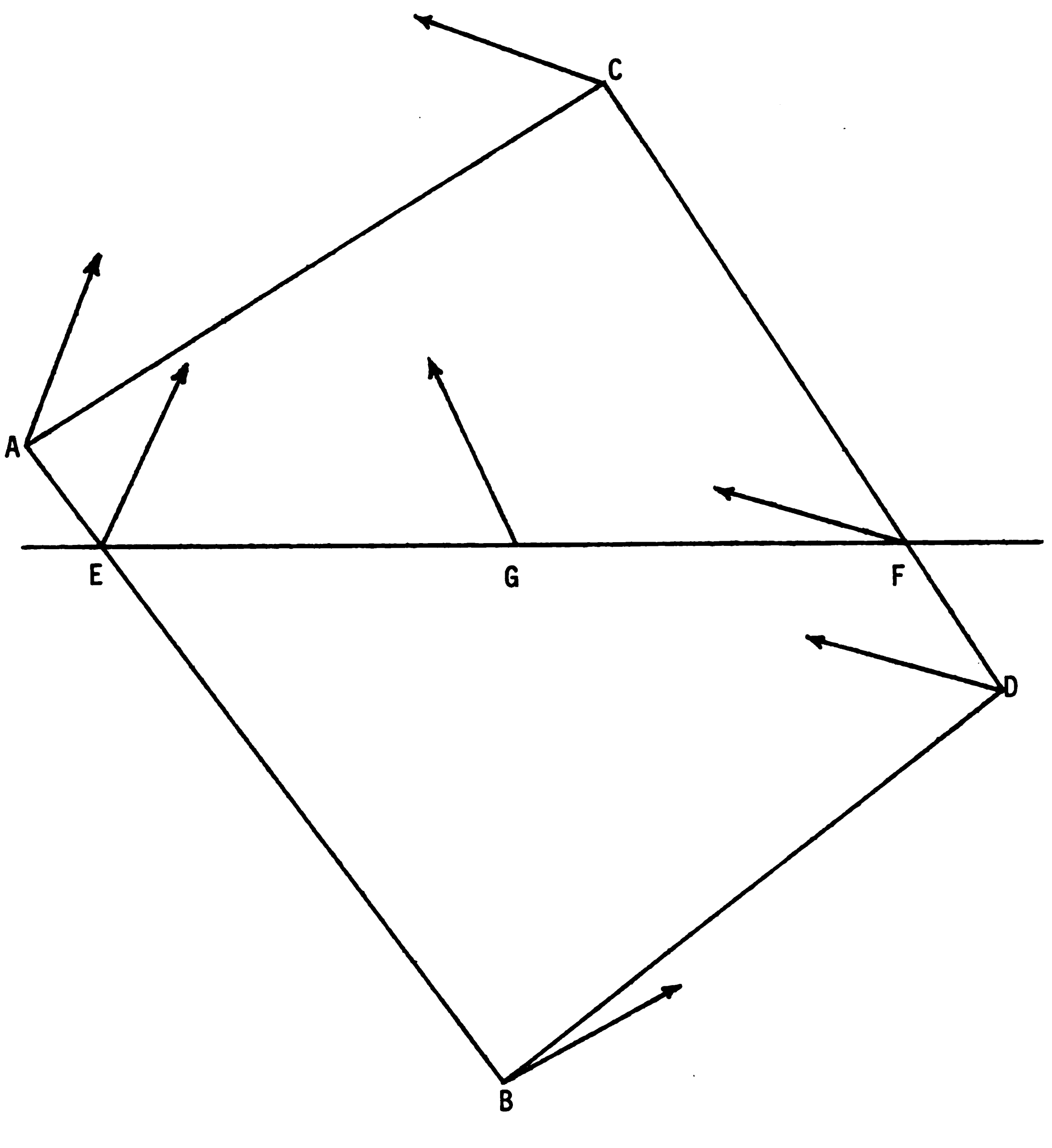


Figure 3-3. Phong Shading.

3.2 Global Shading Methods

The second major class of shading algorithms includes those that use global information to shade a pixel. These algorithms take into account not only the object to be displayed, but also the position of this object relative to other objects in the scene.

A limited amount of interaction between objects in a scene is exploited in the simulation of transparency. In the screen space, points interact to form the displayed image. They are able to interact based on their respective transmission coefficients. The objects must have the same X and Y location on the screen for this to happen. In global techniques, objects need not be near each other in the screen space in order to interact.

3.2.1 The Whitted Display Technique

The global technique described here was developed by Whitted [17]. Whitted's algorithm uses a method known as ray tracing. In ray tracing, rays are followed from the eye point, through each sample point in the screen, to the objects in the scene. The ray does not stop when it hits the first object in the scene. New rays are generated in the reflected and refracted directions at each intersection, as well as in the direction of each light source. This is seen in Figure 3-4. The intensity of each of these rays, as well as the direction of the refracted ray, is determined by the appropriate surface property coefficients. This intersection process continues until none of the generated rays intersect any objects.

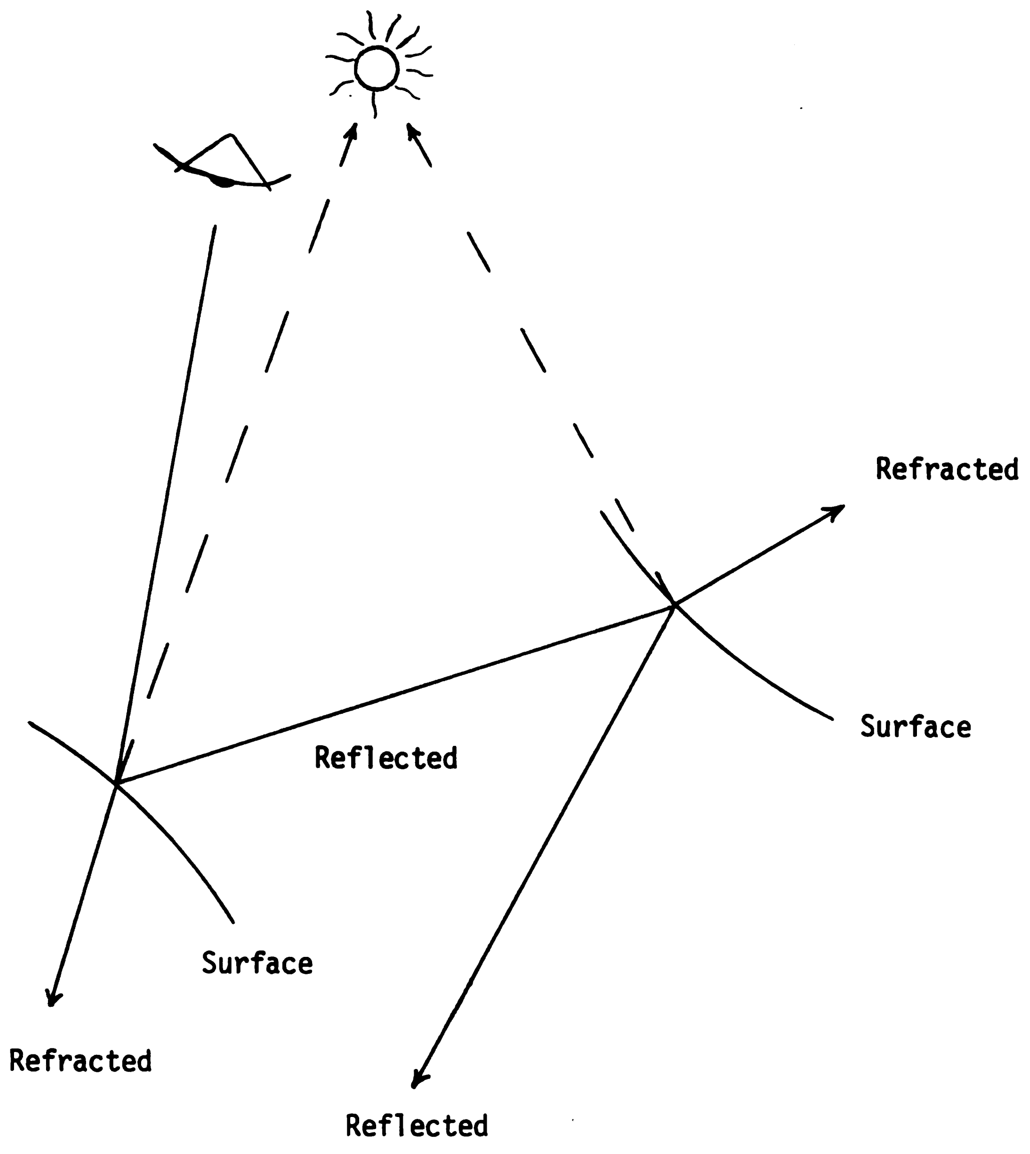


Figure 3-4. Ray Tracing.

Using this technique, an object can be visible as a reflection in an object, even though another object lies between it and the eye point. This adds a great deal of realism to scenes, especially when there are highly reflective or transparent objects in the scene. This would not be possible in a method using only local information.

3.2.2 Shadows

Another shading algorithm of note was developed by Williams [18]. This algorithm does not deal with shading directly, but it deals with casting curved shadows on curved surfaces. It does not deal with how to calculate intensities for objects, but with how to determine if they are in a shadow.

Two complete views of the scene are constructed. The method used for generating these views is not critical, as long as all depth and hidden surface information is retained. The first of the views is constructed from the point of view of the light source. The shading values do not need to be calculated. The second view is constructed from the location of the eyepoint. In the absence of shadowing, this would be the view that would be displayed. All shading information is calculated and retained.

As each visible point in the second view is generated, it is transformed into the space of the first view. The transformed points are then checked, using depth information from both views, to see if it would be visible to an observer located at the light source. If the point is not visible from the light source, it is in a shadow, and its intensity is adjusted from its original unshadowed intensity. This allows one surface to cast a shadow on another surface.

The following two chapters will describe various techniques to solve the hidden surface problem. These methods can be combined with the techniques in this chapter to form a complete shaded image with hidden surfaces removed.

Chapter 4

POLYGON DISPLAY METHODS

This section describes three algorithms for displaying shaded scenes with hidden surface removal: Warnock; Newell, Newell, and Sancha; and Watkins. The data bases for each of these methods are planar polygons. The faces are represented only as a set of ordered vertices, with internal information derived from the equations of the plane.

All three of these methods are classified as image-space methods as opposed to object-space methods. This means that prior to all of these algorithms, the data must be translated and rotated into viewer's coordinate system, it must undergo perspective transformation, and it must be windowed into the screen space.

These methods, plus simple shading rules, lead to faceted approximations to curved scenes. If compatible, more complex shading methods such as Gouraud or Phong yield smooth shading. These methods have been used for years to produce quite high quality, visually pleasing images [10].

4.1 The Warnock Method

The Warnock method [15] can be classified as an image-space depth priority algorithm. It views the screen as being made up of a collection of homogeneous areas, called windows. If a face completely covers a window, and it is the face closest to the viewer, then the window can be displayed. Otherwise the window is divided into four smaller windows and the process is repeated.

There are three possibilities resulting from testing a window, shown in Figure 4-1. First, one or more faces completely cover the window. These faces are called surrounders. The closest of the surrounders is called the critical surrounder. The critical surrounder must be tested at each of the corners of the window against all of the other surrounders. If it is closer to the viewer than all of the surrounders, the window is called homogeneous and it can be displayed.

The second possibility is that one or more faces intersect the window, but do not surround the window. This case calls for further X-Y subdivision of the window. The exact method of subdivision is not critical. Consider splitting it into four equal subwindows. The process is then repeated on each of the four subwindows in turn.

The third possibility is that all faces completely miss the window. This indicates that the window is blank and can be colored the background color.

The algorithm avoids unnecessary calculations by passing relevant faces down to subwindows when they are formed. This means that a subwindow is tested against a face only if the window from which it was made was tested against the same face.

The major drawback of this method is that the windows to be shaded are generated in a random order, and are of varying

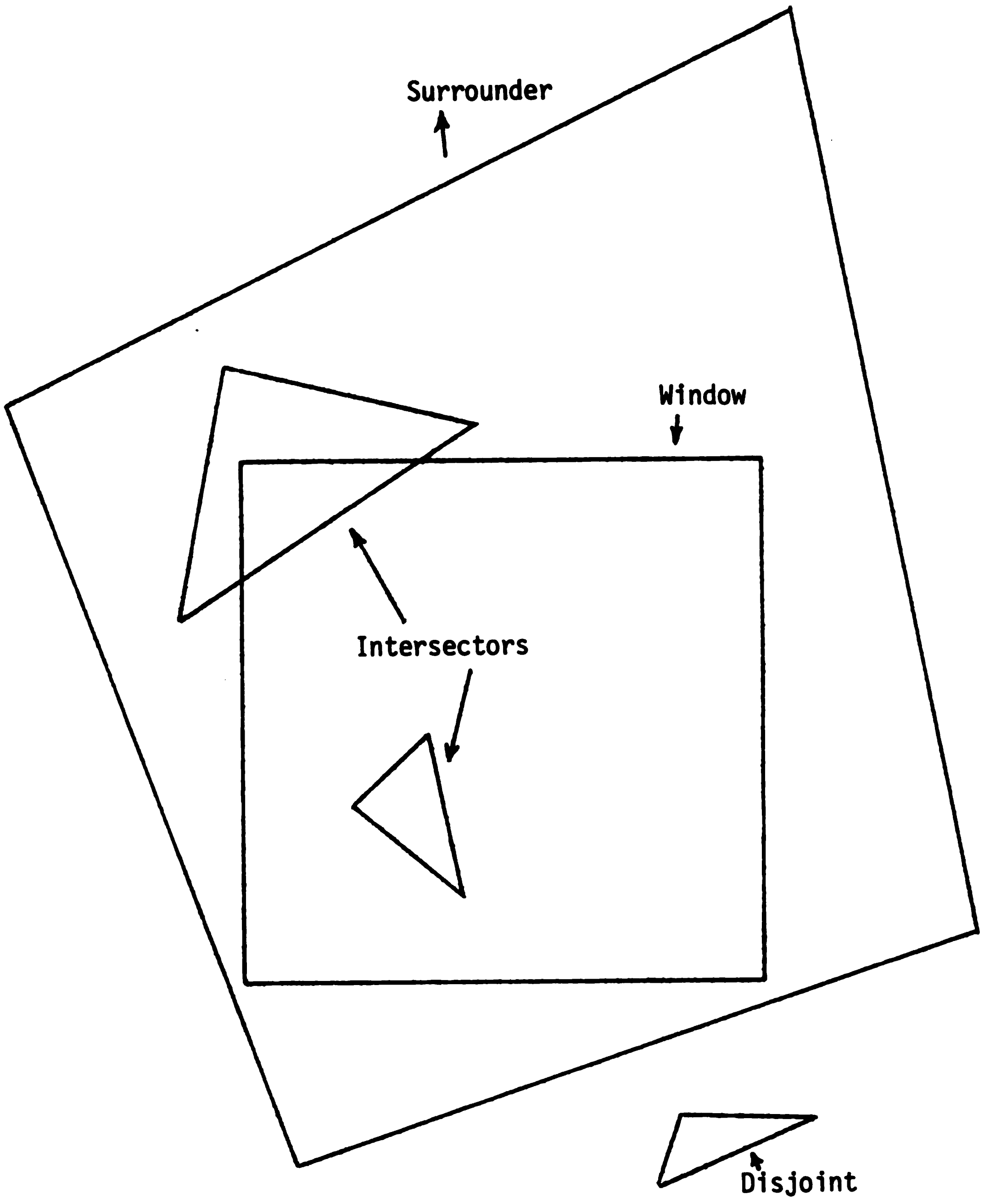


Figure 4-1. Testing Polygon Faces in the Warnock Method.

sizes. This is not a convenient form if the image is to be displayed on a raster scan device. The windows can be manipulated to be compatible with a raster scan device, but a large amount of sorting is required [5]. Also, this random order is not an easy form from which to apply antialiasing techniques.

4.2 The Newell, Newell, and Sancha Method

The Newell, Newell, and Sancha method does a Z sort on each face's centroid to determine Z priority of all faces in the scene [9]. The Z sort orders all faces according to how far they are from the eye in the Z direction. The resultant priority list is used to generate intensities. These intensities are then written into a depth buffer in increasing priority. This will fail in cases involving cyclic overlap, large faces, or areas where penetration occurs. These are shown in Figure 4-2. In both cases, smaller faces are needed because there exists no correct priority with the current face configuration.

The method works well with numerous small faces. This would be the case in a model of a curved surface. It also adapts well to the centroid of each distinct group of faces. Once the priority of the groups is determined, the individual groups can be handled separately.

To handle cyclic overlap, large faces, or areas where penetration occurs, an extended algorithm has been developed. The face furthest from the eye is tested against any face that it could obscure. The first step of the extended algorithm is a

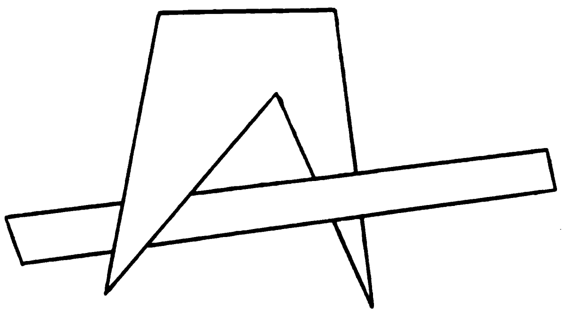


Figure 4-2a. Cyclic Overlap with Polygon Faces.

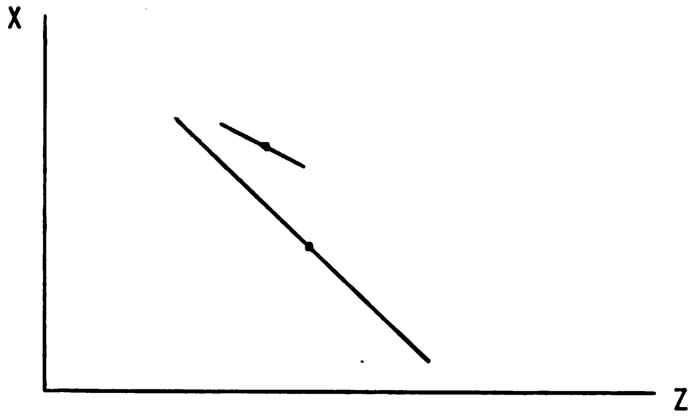


Figure 4-2b. Large Polygon Faces.

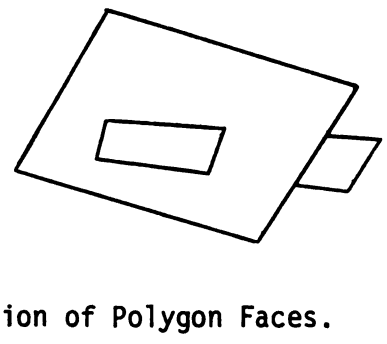


Figure 4-2c. Penetration of Polygon Faces.

test for overlap in Z. This is done by checking whether a situation as in Figure 4-3 occurs. This happens when the vertex of polygon A closest to the viewer is closer to the viewer than the vertex of B which is furthest from the viewer. This is followed by a similar test for overlap in both X and Y. This is necessary because face A can only obscure face B if they overlap each other in both X and Y. The final test is to see whether one of the faces lies completely in the front or back half space created by the plane of the other. This is done by testing whether all of the vertices of one face lie behind the plane created by the other face. This is shown in Figure 4-4. If the priority of the two faces cannot be determined using the above steps, one of the faces is split and the process is repeated using the two fragments.

An advantage of the depth buffer approach is that transparency effects can be simulated. This is done by partially overwriting, using a transmission coefficient, in the depth buffer. If one point is in front of another, it does not necessarily replace it in the depth buffer, but a combination of the two results. The characteristic of the combination depends on the transparencies of the faces involved. To achieve proper rendering of transparency, points must be written into the buffer from back to front.

4.3 The Watkins Method

Watkins also developed a polygon display algorithm [16]. This is the most widely used of the three methods, being available from Evans & Sutherland [14] and with MOVIE.BYU [8]. It is classified

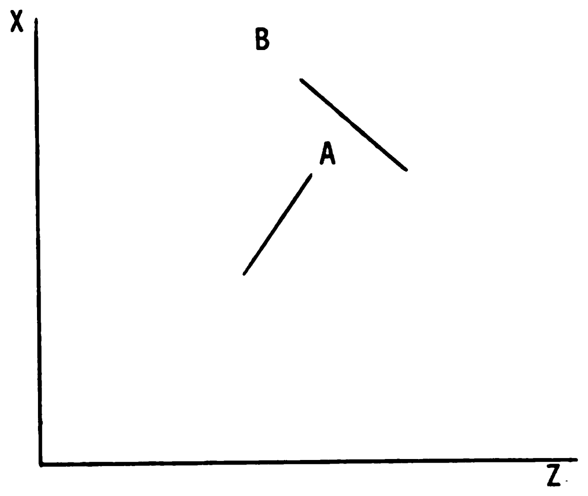


Figure 4-3. Z Overlap Test for Polygon Faces.

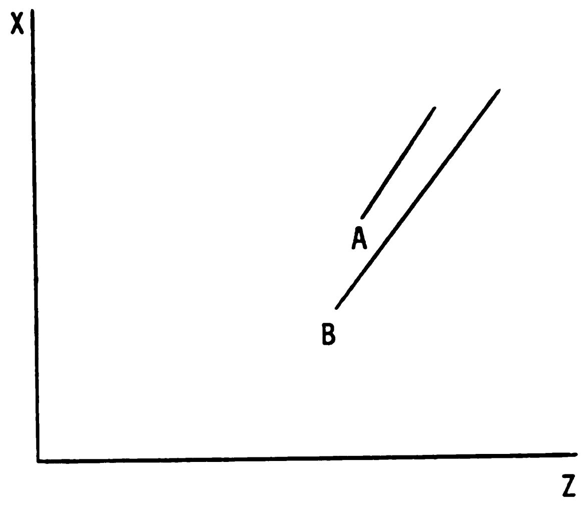


Figure 4-4. Back Half-Space Test for Polygon Faces.

as an image-space scan line algorithm. The polygons are made up of a set of ordered edges. The edges are processed separately, but each one carries a tag indicating which polygon it is a part of.

The algorithm starts with a preprocessor. This is a scan line screening of all edges. This screening determines which edges begin or end on each scan line. These intersections are ordered in X from left to right, establishing when each edge becomes active, and for how many scan lines it is active. The program then saves calculations by working only with the edges that are active on the current scan line.

The next step, the Y scan, computes a set of spans for each scan line, as in Figure 4-5. These spans are sections of the scan line on which the same face is visible. Within each span, all active faces (a face is active whenever one or more of its edges are active in both X and Y) must be tested in Z to determine which one is visible. That face will then stay visible as X increases along the scan line at least until the next edge intersection.

The process has been sped up by assuming that the ordering of spans on one scan line is a good place to start when processing the next scan line. This is called scan line coherence. For example, if no edges enter or leave the active list, and the order of intersections does not change between scan lines, the visibility of all of the spans is the same as in the last line. This indicates that they can be displayed immediately, saving a significant portion of the calculation time for that scan line.

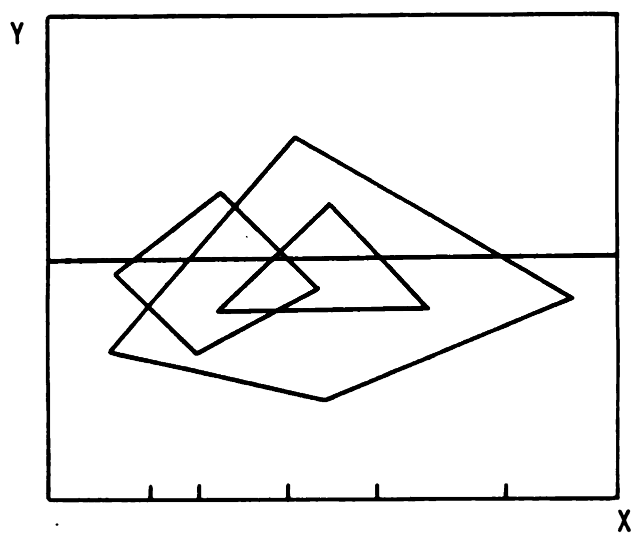


Figure 4-5a. Sample Spans in Watkins Method, X-Y.

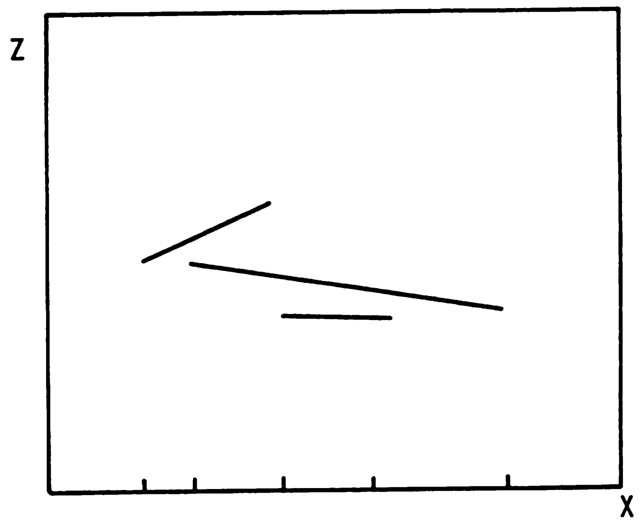


Figure 4-5b. Sample Spans in Watkins Method, X-Z.

Even more aggressive uses of scan line coherence are possible, as indicated in Reference [16].

The last step, the X scan, shades one scan line at a time. The Watkins method can be used with a pointwise intensity function, as well as with either Gouraud or Phong shading. The Watkins algorithm also uses a very efficient method for sorting edges in Z, the so-called logarithmic sort. The major advantages of the Watkin's method are its speed and the fact that it generates intensities in scan line order. This is very convenient when antialiasing techniques are to be employed.

The above methods are capable of producing visually pleasing, smoothly shaded images. This is appropriate for many applications. But if the data base is composed of parametric surfaces, we may want to operate directly on them without first breaking them up into polygons. The following chapter will describe three methods for displaying parametric surfaces directly.

Chapter 5
DISPLAY METHODS FOR PARAMETRIC SURFACES

Most computer-aided design of curved surfaces is done using parametric surfaces. These surfaces are defined by three parametric functions

$$x = x(s,t)$$
$$y = y(s,t)$$
$$z = z(s,t)$$

Most often, the parameters s and t vary from 0 to 1 across a patch on the surface. This type of surface allows the designer a great deal of control over the shape of the surface. It also provides smoothness, well defined derivatives, and continuity everywhere on the surface. Appendix B discusses two types of parametric surfaces. The rest of this chapter discusses various display methods for parametrically defined surfaces.

5.1 The Catmull Algorithm

Catmull presented one of the first algorithms to display parametric curved surfaces directly, without first breaking the surface into polygons [3]. Catmull's algorithm involves subdividing a patch until each of the pieces covers only one pixel.

When this criterion is met, the subpatch can be displayed. This method bears likenesses to two of the polygon methods mentioned earlier. It is similar to Warnock's in that recursive subdivision is used until a termination criterion is met. The main difference between them is that Warnock subdivides the viewing screen while Catmull subdivides the surface which is displayed. It resembles Newell, Newell, and Sancha's method in that both subdivide the object to be displayed until a termination criterion is met. The main difference between the two is that Newell, Newell, and Sancha's subdivisions are motivated by conflicts between objects, while Catmull's are motivated by patch size.

Subdividing all patches in the scene down to pixel size can be very time consuming due to the large number of subdivisions involved. Catmull sped up the process by finding a very fast way of subdividing a specific surface, the parametric bi-cubic.

In order to terminate subdivision, a test is run to see if a subpatch covers more than one pixel, as in Figure 5-1. An approximating polygon is used for this purpose. This polygon connects the four corners of the patch. The polygon is tested to see if it covers more than one pixel. This method should work on patches with little curvature. It would not work with highly curved patches or patches with poor orientation, as in Figure 5-2. This condition would have to be detected from the patch geometric parameters and dealt with more rigorously. If it could not be detected, a local error would occur.

A second method to terminate subdivision is to check whether a subpatch is completely out of the viewing area. If a subpatch

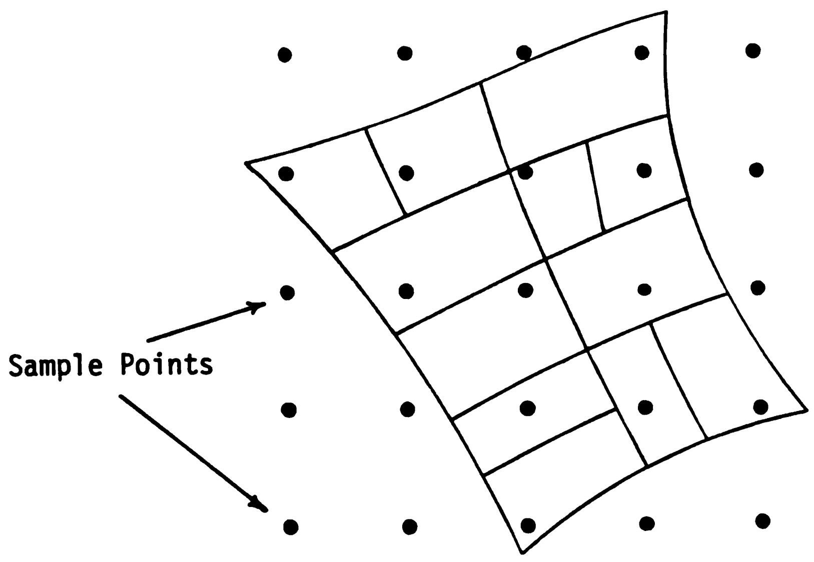


Figure 5-1. Subdivision of a Patch in the Catmull Method.

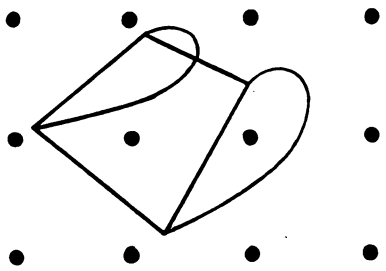


Figure 5-2. Failure of Approximating Polygon.

does fall outside the viewing area, it is dropped from further consideration. The same polygonal approximation could be used when clipping against the edges of the viewing area.

To aid in solving the hidden surface portion of the general display problem, Catmull described a 'Modified Newell Algorithm', in which patches are sorted by their Bezier control points (see Appendix B). This is useful since the Bezier surface is constrained to be in the convex hull of its control points [12]. If the Z priority between two patches cannot be determined, the patches are subdivided. When the Z priority of the subpatches is resolved, they can be displayed from front to back by the method above.

The method could involve a large number of subdivisions, due to the fact that the Bezier control points of two patches might intersect even if the patches do not. The idea of establishing Z priority is the key that links this idea to the method of Newell, Newell, and Sancha.

Alternatively, a Z buffer can be used. This is the method which Catmull implemented. Each patch is subdivided as necessary until it can be displayed. As each point is to be displayed, its Z value is compared to that of the point already in the buffer at the appropriate location. If the new point is behind the one already in the buffer, it is discarded. Otherwise, its intensity is calculated, and it is displayed. The ideas concerning transparency, which were discussed in the Newell, Newell, and Sancha method, are also applicable here. As before, the points on a particular pixel must be displayed from back to front if transparency is desired.

Catmull also described the first method by which shadows could be cast from one curved surface onto another curved surface. This involves finding the silhouette of a surface, from the point of view of a light source. One can then create shadow patches which extend out from the silhouette, away from the light source. Any object which falls in the path of these shadow patches is in the shadow of the surface which generated the silhouette. When the unshadowed picture is completed in the buffer, shadowed elements of the scene must have their intensities modified appropriately.

The major advantage of Catmull's algorithm over polygon based techniques is that it operates directly on parametric surfaces. A drawback is that the points for display are not generated in an order convenient for antialiasing techniques to be applied.

5.2 The Blinn Algorithm

The next surface method was developed by Blinn [1]. This algorithm is similar to Catmull's in that both operate directly on parametric surfaces. It is similar to the Watkins algorithm in that both are scan line methods.

The algorithm has three major parts: preprocessor of patches, Y scan, and X scan. The preprocessor finds and orders a set of local Y maxima for each patch. The maxima are processed in the Y scan to form a set of points, called iteration points, on each scan line. The segments between these iteration points are filled in during the X scan.

The first step in the algorithm, the preprocessor, starts with a prescreen of all patches in the scene. The patches are scanned by their Bezier control points. This creates an active patch list analogous to the active edge list in the Watkins algorithm. Each patch is then scanned using nonlinear optimization to find its local maxima. True local maxima satisfy:

$$dY(s,t)/\underset{ds}{\cancel{dx}} = 0 = dY(s,t)/dt \qquad (5.1)$$

These maxima are bulges in a patch. These can include points on a silhouette curve. The normals to points which are on a silhouette curve lie in the plane of the screen. The Z component of the normal vector to a point on a silhouette curve is zero:

$$Nz(s,t) = 0 \qquad (5.2)$$

Local maxima can occur at an edge of a patch, as in Figure 5-3, where either

$$dY(s,t)/ds \neq 0 \qquad (5.3)$$

or

$$dY(s,t)/dt \neq 0$$

A maximum also can occur at a corner, as in Figure 5-4, where:

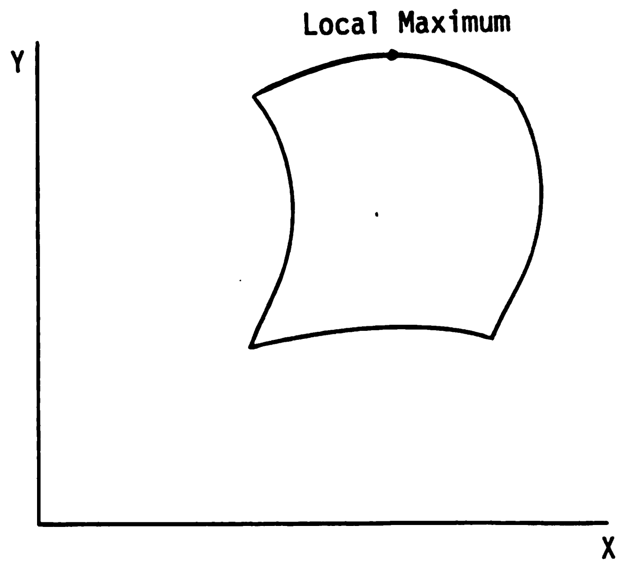


Figure 5-3. Local Maximum on the Edge of a Patch.

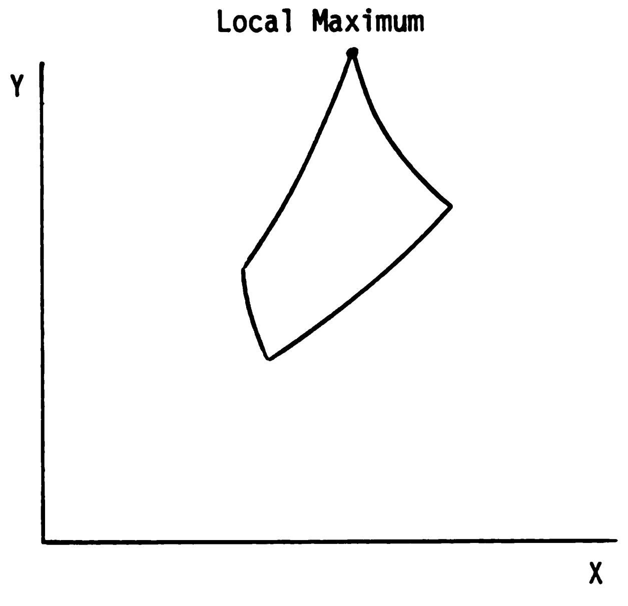


Figure 5-4. Local Maximum at the Corner of a Patch.

$$dY(s,t)/ds \neq 0 \tag{5.4}$$

and

$$dY(s,t)/dt \neq 0$$

These maxima are used in the second step of the program, the Y scan, to produce traces of the boundary edges and silhouettes for each patch. The silhouettes and boundary edges are the minimum information needed to describe the projection of the patch onto the screen. In addition to the boundaries and silhouettes, Blinn finds a number of key visual points on each scan line. He uses these in conjunction with the traces of edges and silhouettes obtained from the maxima, to shade the surface. The key visual points can be identified by any of a number of characteristics, including inflection points in the screen plane, or constant angle increments on successive points along a curve, as in Figure 5-5. A more complete description of these points and their identification can be found in Reference [1].

The second step in the algorithm, the Y scan, cuts constant Y planes through the screen plane. As the scan moves down the screen it has several functions. First, it must process entering local maxima. A local maximum becomes active when the Y scan falls below the Y position of that maximum on the screen. Next, it must update the position of iteration points (key visual points, traces of edges or silhouettes) from one scan line to the next. The process of updating the position or edges of silhouettes means calculating the intersection of edges and silhouettes with the current scan line. For edges, this means solving equations of the form:

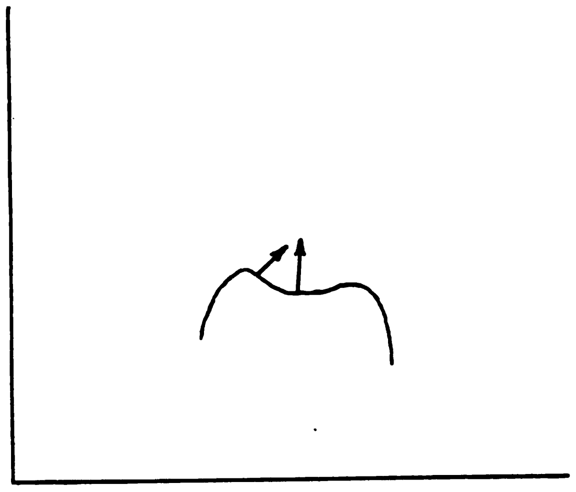


Figure 5-5a. Key Visual Points - Inflection Points.

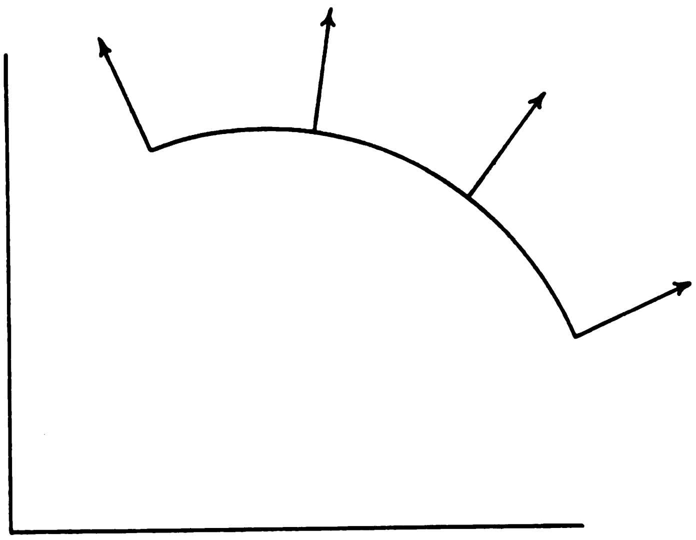


Figure 5-5b. Key Visual Points - Angular Increments.

$$\begin{aligned} & Y(s,0) = Yscan, \\ & Y(s,1) = Yscan, \\ & Y(t,0) = Yscan, \\ \text{or}\quad & Y(t,1) = Yscan. \end{aligned} \tag{5.5}$$

For silhouettes, the equations are:

$$\begin{aligned} & Y(s,t) = Yscan \\ \text{and} \\ & Nz(s,t) = 0 \end{aligned} \tag{5.6}$$

The solutions for the edges are found using univariate iteration, the silhouettes by bivariate iteration.

The Y scan portion must also remove points from consideration as needed. This occurs when a portion of the surface does not extend below the current scan line. Finally, the Y scan must also keep all iteration points properly connected as any maxima enter or leave the chain of points from one scan line to the next. The output of the Y scan is a set of points on each scan line. These points will be used during the X scan to shade the entire scan line.

The last step in the algorithm, the X scan, scans across the screen. It scans from left to right on any given Y scan line, with the purpose of shading all of the pixels on that scan line. To do this, the normal at each iteration point on the scan line is calculated from the surface description. Gouraud or Phong shading can then be used to fill in shade between the iteration points, with the intensities being written in a depth buffer. This approach will yield a smoothly shaded scene with hidden surfaces removed.

The major advantage of the Blinn approach is that it is a surface based algorithm. This means that information needed about the surfaces is available at any point. Another advantage is that it is a scan line algorithm. This means that it can be conveniently displayed on a raster scan device, and that anti-aliasing techniques can be applied.

A disadvantage is that intersections of curved surfaces are represented in the final image using straight lines, as in Figure 5-6. Also, interpolation in shade explicitly uses the surface information at relatively few points in a scene, namely the key visual points and edges. This would seem to offset some of the leverage gained by going to the more exact representation of surfaces rather than using collection of polygons or quadrics.

5.3 The Whitted Algorithm

The next surface-based display algorithm was developed by Whitted [17]. It has the same three major sections as the Blinn algorithm.

The preprocessor of the Whitted algorithm represents the projection of each surface on the screen plane by a collection of cubic curves. These curves are all chosen so that they are functions of one variable.

The first type of curve is the boundary edge curve. These are of the form:

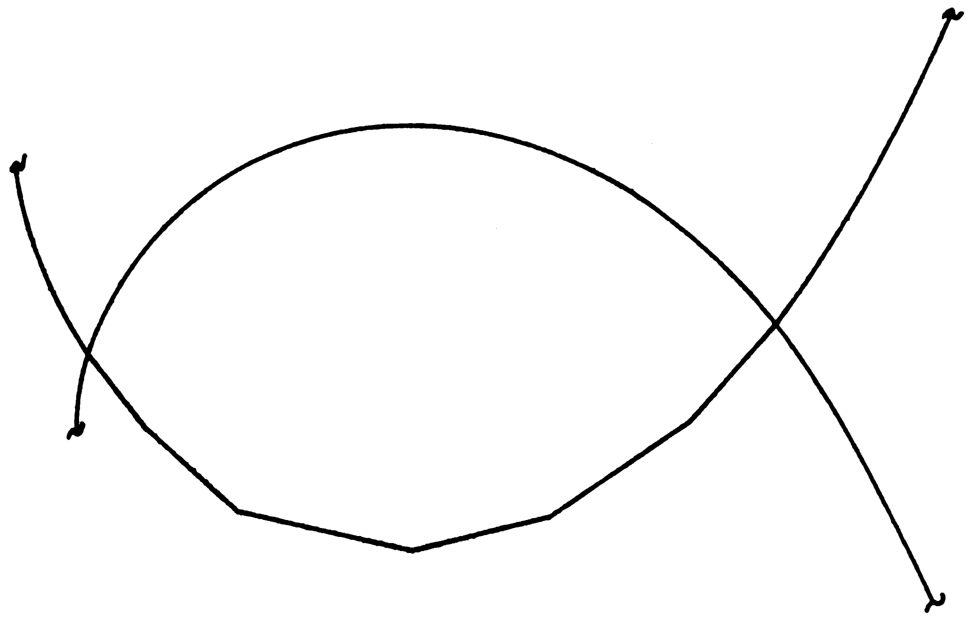


Figure 5-6. Straight Line Intersection.

$$
\begin{aligned}
Q0(s) &= Q(s,0) \\
Q1(s) &= Q(s,1) \\
Q0(t) &= Q(0,t) \\
Q1(t) &= Q(1,t)
\end{aligned}
\tag{5.7}
$$

These are univariate cubic curves by definition.

The second type of curve to be considered is the silhouette, which is characterized by Equation (5.2). A silhouette would normally be of a higher order than cubic. The algorithm approximates the silhouettes by sets of piecewise continuous cubics of one variable. Whitted uses Hermite interpolants to do this (see Appendix A). The number and placement of these interpolants will determine the quality of the approximation of the silhouette.

The addition of internal curves of the form:

$$
Qk(s) = Q(s,k) \quad \text{or} \quad Qk(t) = Q(k,t) \tag{5.8}
$$

where k is constant, has the effect of breaking the surface into quadrilateral surface portions. These quadrilaterals are analogous to polygon representation of the surface, albeit with curved boundaries. The number and spacing of these internal curves control the quality of the representation of the surface in the screen plane.

After the edge curves are defined, silhouettes approximated, and interior curves added, the program works only with cubic equations in one variable. All of these curves, with the exception of the silhouettes, lie completely on the surface. The edges and internal curves are cubics, while the silhouettes are higher order curves being approximated by cubics.

The Y scan portion of the program solves equations of the form:

$$y(*) = Yscan \tag{5.9}$$

for each cubic that intersects the current scan line. Curves which are not monotonic in Y must be treated separately due to multiple intersections with the scan line. After all of the intersections on a scan line have been located, they are paired together to form segments on the scan line, as in Figure 5-7.

The X scan shades the segments between the iteration points on each scan line. Along all of the curves in the scene, Whitted calculates the normal to the surface. This is analogous to calculating the normals along the edges of a polygon. This obtains the normals for the end points of all segments on a scan line. Phong shading is then used to fill in the intensities on a segment. These intensities are written into a Z buffer. The result is a smoothly shaded scene with hidden surfaces removed.

Since the silhouettes are represented by cubics rather than by straight lines, this method will yield smoother silhouettes than polygon routines. But averaging is used to find shading values,

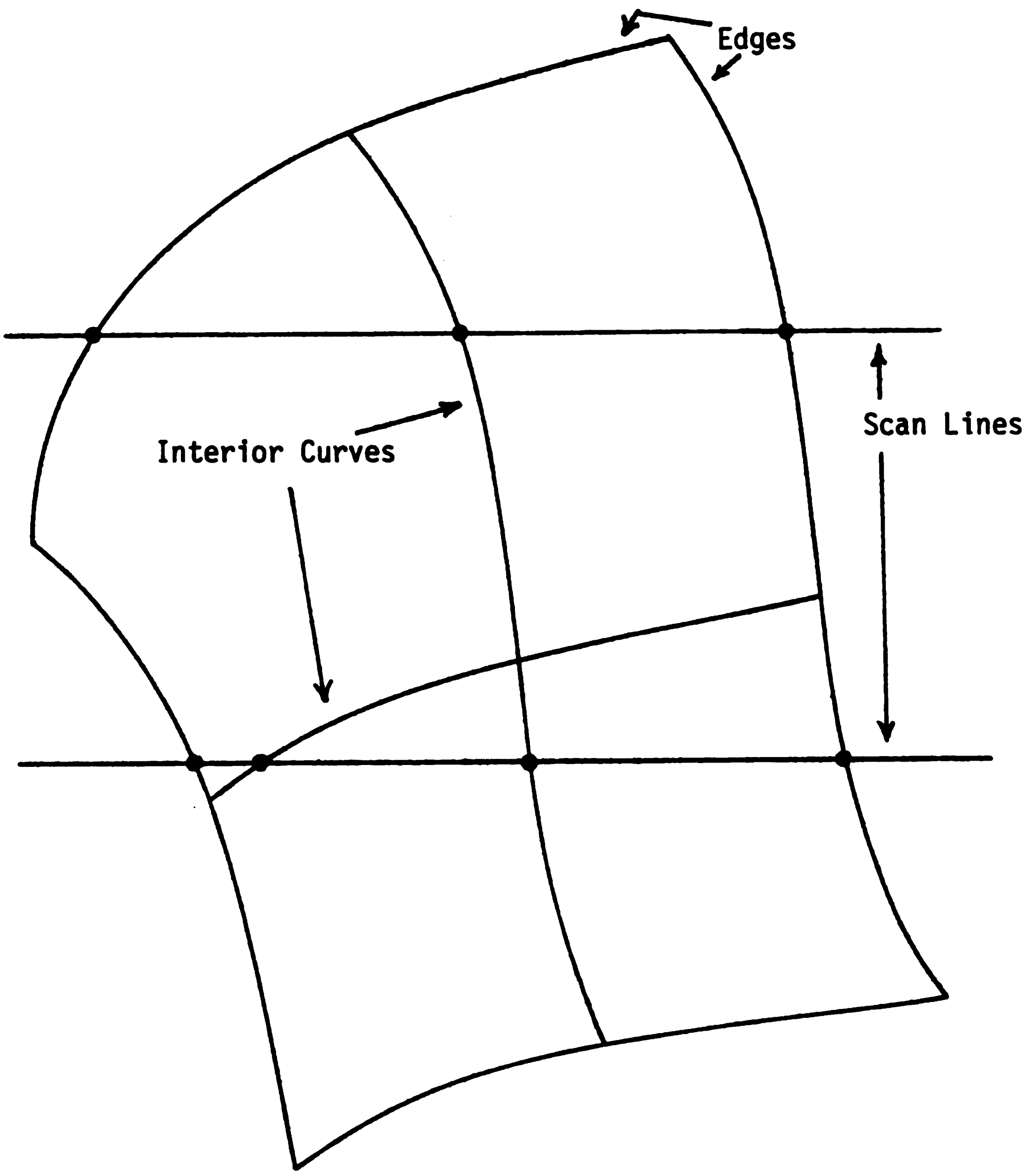


Figure 5-7. Segments in Whitted Method.

giving up some of the advantage gained by using the parametric representation for patches.

Whitted does not present an explicit method to decide how many internal curves to use in representing the surface, nor how many to use to represent a silhouette. But it is clear that the number of curves affects the ratio of points on a scan line whose normals are calculated directly to those whose normals are found by interpolation. This means that the number of curves used to represent a patch will affect the quality of the shaded image.

The following chapter will describe a method to display parametric surfaces directly, a pixel at a time, in scan line order.

Chapter 6

A POINT BY POINT SCAN LINE DISPLAY ALGORITHM FOR PARAMETRIC SURFACES

Using a polygonal approximation to curved surfaces simplifies the surface to a collection of planar faces. This introduces slope discontinuities into the scene. Averaging must then be applied to remove the faceted appearance of the faces. It is also clear that slope discontinuities between patches which actually exist in a data base could be altered or lost in a polygon method.

The methods of Blinn and Whitted lead to similar problems. They break each scan line into segments. They calculate normal vectors at the ends of each segment, ignoring the parametric nature of the surface everywhere else. The segments can be thought of as polygons which are one pixel high. As with any polygon method, interpolation is needed to avoid a faceted or banded appearance in the shaded image.

The objective of a raster display algorithm is to assign an intensity to each pixel on the display screen. The ideal way to do this is to compute the intensity for each pixel directly from the geometry of the objects in the scene. This point by point calculation gives the truest rendering of the geometry of the objects. Since we are dealing with parametric surfaces, we would like to shade pixels directly from the parametric surface definitions.

We will now examine a way to display each pixel based on a separate surface calculation. This will be the most accurate representation. This will avoid introducing slope discontinuities into the scene, while at the same time preserving any discontinuities that are actually represented in the original data base.

6.1 A Pixel Based Display Method

Research is currently underway at the Case Center for Computer-Aided Design which indicates that the shade at each pixel can be efficiently calculated directly from the surface geometry. This is equivalent to dropping a sight line through the screen and displaying the point on the surface which the sight line intersects, as in Figure 6-1.

As in more traditional methods, we must first project the edges and silhouettes of each patch onto the screen plane. This can be done using a preprocessor and Y scan similar to the work of Blinn. The Y scan yields the intersection of each scan line with each edge or silhouette. In the course of the Y scan, we also determine the s and t parameter values of each of these intersections. The intersection points are then paired together to form a set of spans on each scan line. We then need to fill in these spans. This is done in ~~in~~ the X scan.

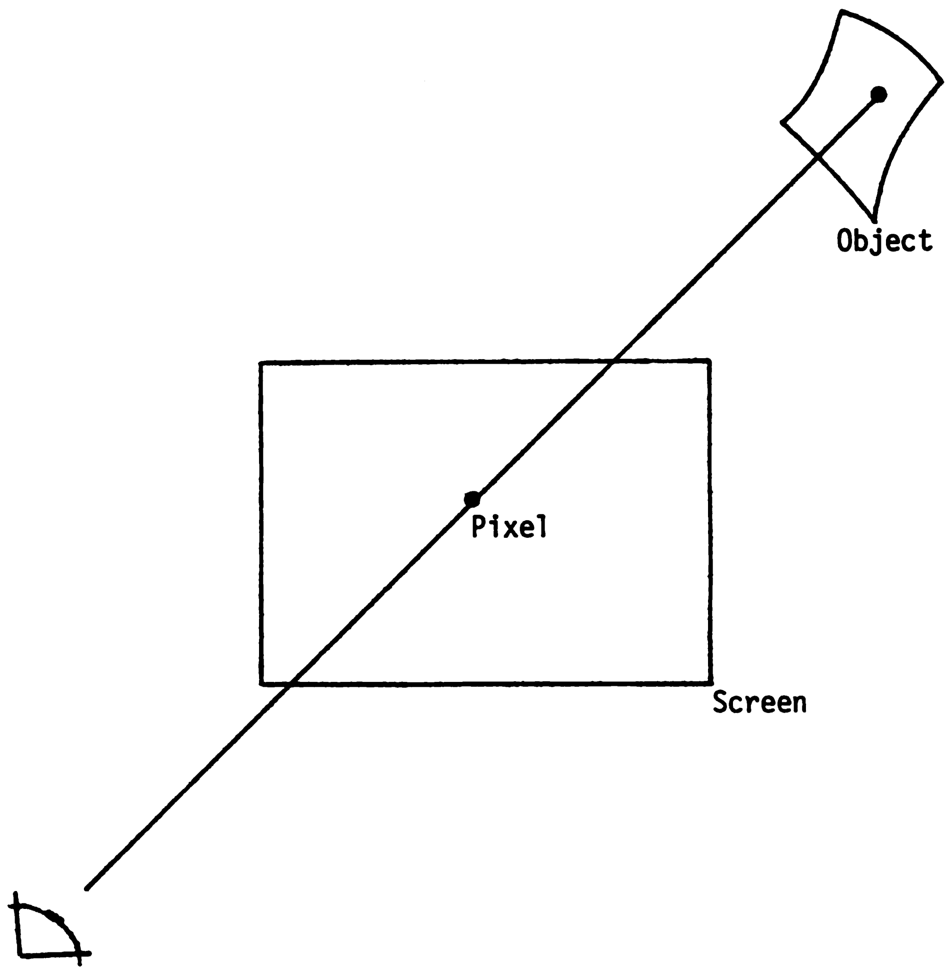


Figure 6-1. A Pixel Based Display Method.

The X scan moves along a constant Y line in the screen plane. The s and t values along the scan line are determined by:

$$\Delta y = \varepsilon = \partial y/\partial s \Delta s + \partial y/\partial t \Delta t$$

$$\Delta x = \delta = \partial x/\partial s \Delta s + \partial x/\partial t \Delta t \qquad (6.1)$$

Equations (6.1) yield a Δs and Δt. The partial derivatives are calculable from the parametric definition of the surface. The values of δ and ε are inputs to the above equations. If δ is set to one pixel spacing in x, and ε is set to zero, the resulting Δs and Δt will be a first order approximation to move one pixel in the X direction on the screen. If the resulting X and Y are within specified bounds of the target pixel, it is written in a depth buffer. Otherwise, as in Figure 6-2, the errors in X and Y are entered into Equations (6.1) through δ and ε, and another iteration takes place. By repeating this process, we are able to step from one end of a span to the other, one pixel at a time. At each pixel, the current s and t are used to calculate the normal vector for that pixel.

6.2 A Display Method using Nth Order Blending Functions

Another approach is to use an equation algebraically equivalent to:

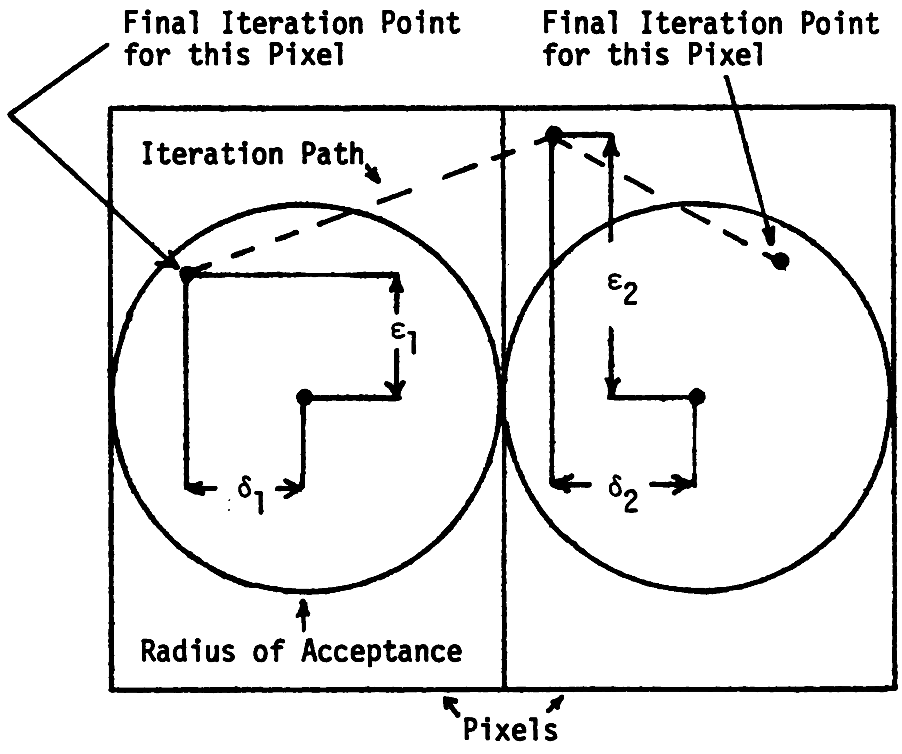


Figure 6-2. Iteration on a Scan Line.

$$N(x) = N_o FN1(x) + N_1 FN2(x) + \frac{dN_o}{dx} FN3(x) + \frac{dN_1}{dx} FN4(x)$$
$$+ \frac{d^2N_o}{dx^2} FN5(x) + \frac{d^2N_1}{dx^2} FN6(x) + \ldots \tag{6.2}$$

between the end points of each span, as in Figure 6-3. The functions FN are blending functions similar in form to those in Appendix A. Thus, to shade a span to any desir~~ed accuracy,~~ we need only calculate the normal ~~vectors and their~~ derivatives with respect to X at the ends of each span.

6.3 A Hybrid Display Methods

A third method to be considered is a hybrid of the first two. We can break each span into subspans using Equation (6.1) with a large δ. Equation (6.2) is then applied on each subspan. We can enlist methods similar to Clark [4] to determine the size of each subspan. In this method, a surface is subdivided until each piece is deemed to be close enough to a planar face. This same idea can be applied along a scan line with an acceptance criterion in which the basis functions are N^{th} order blending functions instead of linear interpolation.

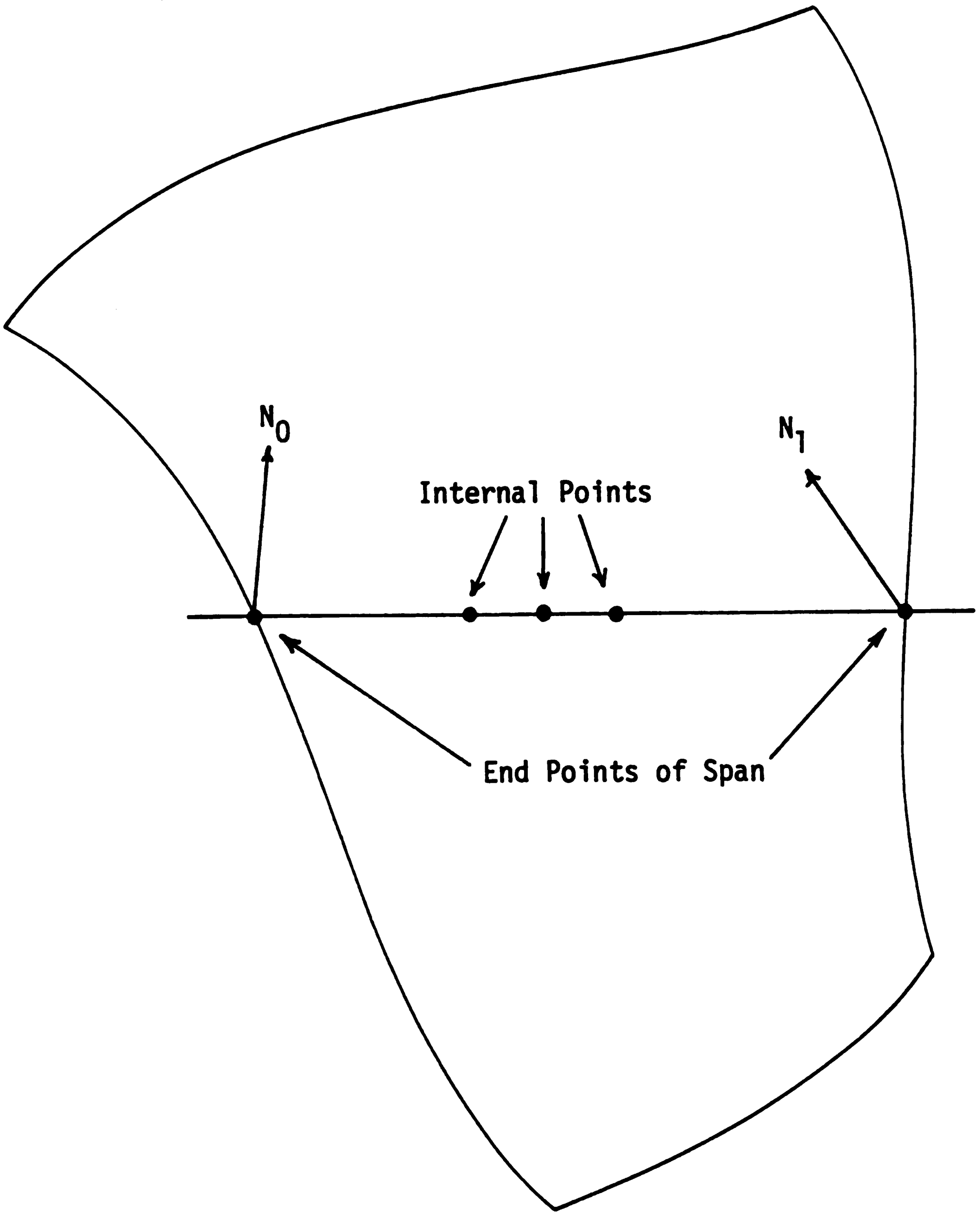


Figure 6-3. N^{th} Order Blending Functions on a Scan Line.

Chapter 7
CONCLUSIONS

Polygon display methods can create smoothly shaded images, but they lose information when breaking a parametric surface into polygons and through interpolation during shading. Traditional scan line parametric surface display algorithms also lose information through interpolation.

Research at Michigan State University indicates the utility of methods which shade each pixel based on surface geometry without linear interpolation. These methods include pixel to pixel iteration, N^{th} order blending functions for surface normals, and a hybrid method which includes both techniques.

Further work is needed to determine what order blending functions are appropriate based on the surface geometry. In the hybrid method, a criterion for the size of interpolation spans and the order of blending functions is needed.

APPENDICES

APPENDIX A

Two types of parametric cubic space curves will be discussed. They are the Bezier or Bernstein cubic, and the Hermite interpolant. These are two ways to generate the same set of curves. They differ in the blending or basis functions that they use. In both cases, the curves have the form:

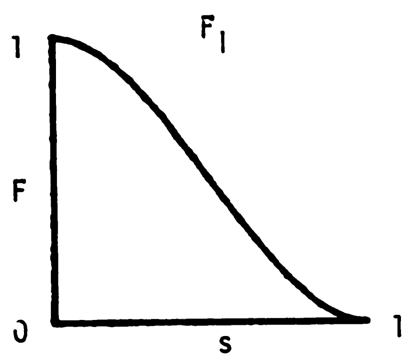
$$Q(s) = \{F(s)\}\{P\} \qquad 0 \leq s \leq 1$$

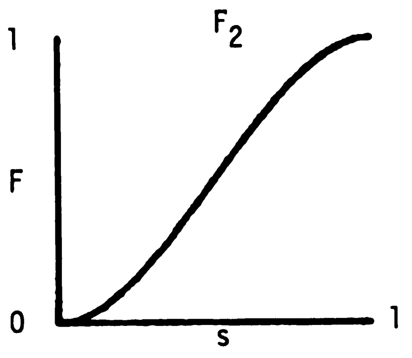
$\{F(s)\}$ is a four element vector containing the blending functions. $\{P\}$ is a vector containing four pieces of information about the curve. Exactly what information is needed depends on the form of the blending functions.

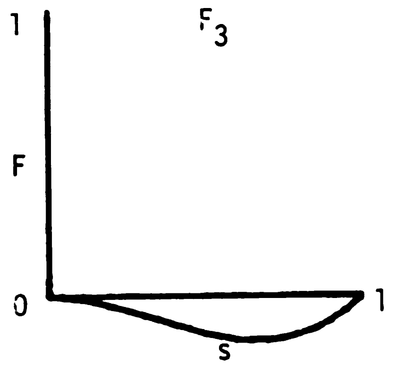
For the Hermite interpolant, these are the blending functions:

$$F1 = 2s^3 - 3s^2 + 1$$
$$F2 = -2s^3 + 3s^2$$
$$F3 = s^3 - 2s^2 + s$$
$$F4 = s^3 - s^2$$

As you can see in Figure A-1, these functions have either a value or slope of one at either end. This allows the user to specify the end coordinates and the end tangent vectors. A sample curve







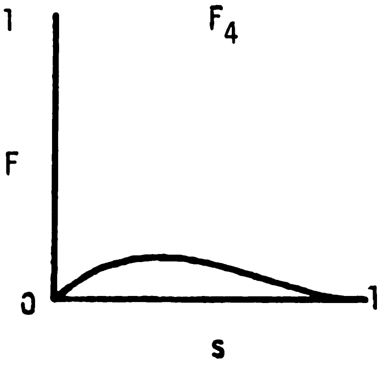


Figure A-1. Hermite Blending Functions.

is shown in Figure A-2. To specify a three-dimensional curve, a {P} vector is needed for each of the three dimensions. This makes a total of twelve pieces for information to construct the curve.

A Bezier or Bernstein cubic uses the following blending functions:

$$F1 = -s^3 + 3s^2 - 3s +1$$
$$F2 = 3s^3 - 6s^2 + 3s$$
$$F3 = -3s^3 + 3s^2$$
$$F4 = s^3$$

The {P} vector contains four coordinates, called control points. A set of control points, and the resulting curve is shown in Figure A-3. As you can see, the curve matches the control points only at the ends. Also, the slopes at the ends are defined by segments AB and CD. Slope continuity across segment boundaries is controlled by making the terminal segments of adjacent curves collinear, as in Figure A-4. As with the Hermite interpolant, three {P} vectors are needed for a three-dimensional curve, resulting in twelve coordinates.

A useful property of Bezier curves is that the curve is constrained to lie in the convex hull of its control points. This means that no part of the curve can extend in any direction further than the set of its control points. This is a convenient

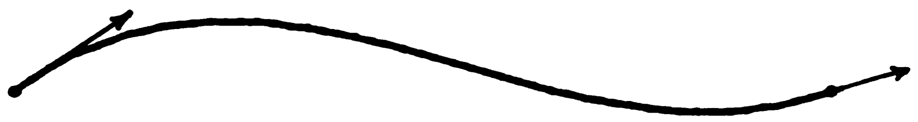


Figure A-2. Hermite Interpolant.

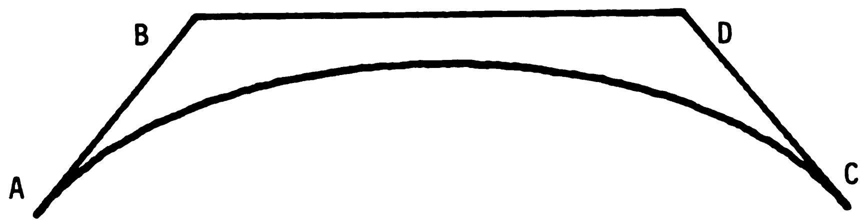


Figure A-3. A Sample Bezier Curve.

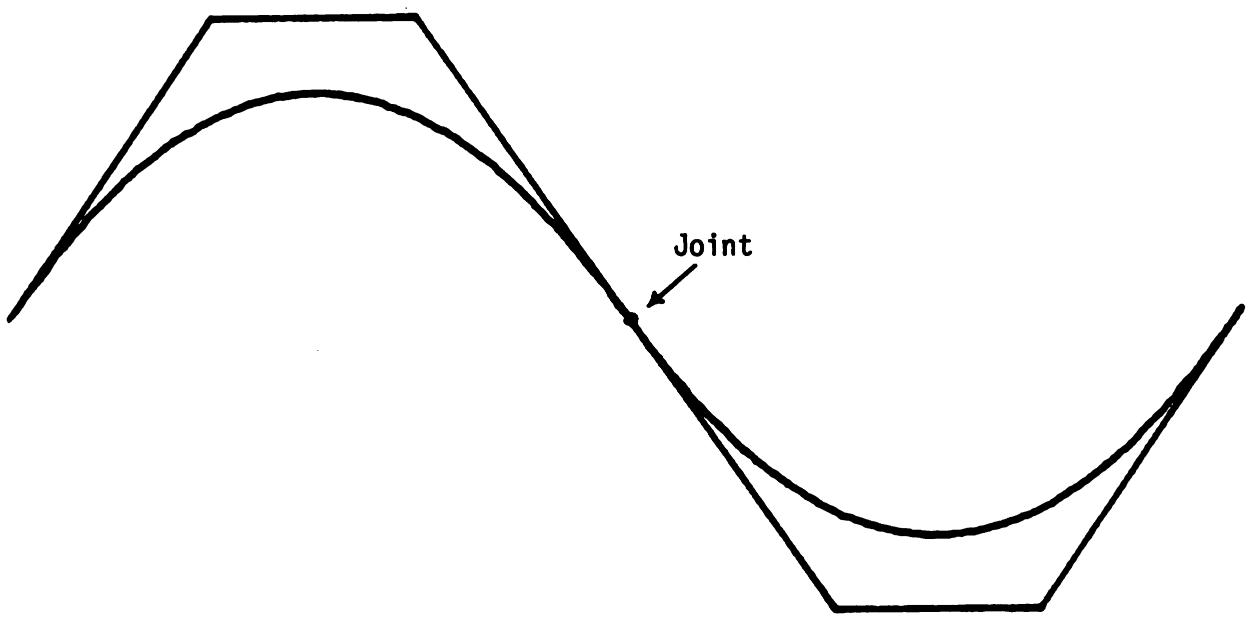


Figure A-4. Slope Continuity in Bezier Curves.

way to put an upper bound on the range of the curve. This property is used for sorting purposes in several algorithms.

APPENDIX B

The Coons and Bezier bi-cubic surfaces will be discussed in this appendix.

B.1 Coons Bi-Cubic Surface

The Coons surface, or Hermite tensor-product surface, has the following form:

$$Q(s,t) = \{FC(s)\}\ [P]\ \{FC(t)\}^T \tag{B.1}$$

{FC(s)} and {FC(t)} are 4-element vectors of the blending functions in Equations (A.1). [P] is a 4x4 matrix:

$$[P] = \left[\begin{array}{cc|cc} Q(0,0) & Q(0,1) & dQ(0,0)/dt & dQ(0,1)/dt \\ Q(1,0) & Q(1,1) & dQ(1,0)/dt & dQ(1,1)/dt \\ \hline dQ(0,0)/ds & dQ(0,1)/ds & d^2Q(0,0)/dsdt & d^2Q(0,1)/dsdt \\ dQ(1,0)/ds & dQ(1,1)/ds & d^2Q(0,0)/dsdt & d^2Q(1,1)/dsdt \end{array}\right]$$

which means

$$[P] = \begin{bmatrix} \text{Corner} & \text{T-Tangent} \\ \text{Coordinates} & \text{Vectors} \\ \text{S-Tangent} & \text{Corner Twist} \\ \text{Vectors} & \text{Vectors} \end{bmatrix}$$

The scalar function Q can be any of the three coordinates x, y, or z. This indicates that the user must specify [PX], [PY], and [PZ].

B.2 Bezier Bi-Cubic Surface

The Bezier bi-cubic surface has the following form:

$$Q(s,t) = \{FB(s)\}\ [B]\ \{FB(t)\} \qquad (B.2)$$

{FB(s)} and {FB(t)} are 4-element vectors of the blending functions in Equations (A.2). [B] is a 4x4 matrix of control points:

$$[B] = \begin{bmatrix} B(1,1) & B(1,2) & B(1,3) & B(1,4) \\ B(2.1) & B(2.2) & B(2.3) & B(2.4) \\ B(3,1) & B(3,2) & B(3,3) & B(3,4) \\ B(4,1) & B(4,2) & B(4,3) & B(4,4) \end{bmatrix}$$

which means

$$[B] = \begin{bmatrix} \text{Corner} & \text{Edge Points} & \text{Corner} \\ \begin{array}{l}\text{Edge}\\\text{Points}\end{array} & \text{Interior Points} & \begin{array}{l}\text{Edge}\\\text{Points}\end{array} \\ \text{Corner} & \text{Edge Points} & \text{Corner} \end{bmatrix}$$

Again, [BX], [BY], and [BZ] must be defined. As with the Bezier curves, the surface only hits the control points at the corners. Also, the surface is constrained to lie within the convex hull of its control points.

In a design situation, one may have Coons patches as the working data base. Bezier control points are convenient for sorting and preprocessing.

Appendix C will present a method to convert back and forth between a Coons and a Bezier definition of a surface.

APPENDIX C

A method to convert back and forth between a Coons and a Bezier definition for a bi-cubic surface will be presented in this appendix.

The Coons and Bezier forms of the bi-cubic surface are two different ways of generating the same class of surfaces. They differ fundamentally in the blending functions they use. These blending functions are actually basis functions. Since they are basis functions, we expect a conversion of the form

$$B = [A]\ [P]\ [A]^T$$

where A is the change of basis matrix [13].

To find A, we rewrite Equations (A.1) as:

$$\{FC(s)\} = \{s^3 s^2 s\ 1\} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \{S\}\ [N] \qquad (C.1)$$

(A similar relationship exists for {FC(t)}.) So, substituting into Equation (B.1), we find

$$Q(s,t) = \{s\}\ [N][P][N]^T\{t\}^T \qquad (C.2)$$

Similarly,

$$Q(s,t) = \{s\}[M][B][M]\{t\}^T \qquad \text{Note: } ([M] = [M]^T) \qquad (C.3)$$

where

$$\{FB(s)\} = \{s^3 s^2 s\ 1\} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \{s\}[M] \qquad (C.4)$$

Equating (C.2) and (C.3)

$$\{S\}[M][B][M]\{t\}^T = \{s\}[N][P][N]^T\{t\}^T \qquad (C.5)$$

or

$$[M][B][M] = [N][P][N]^T \qquad (C.6)$$

which yields

$$B = [M]^{-1}[N][P][N]^T[M]^{-1}$$

Combining $[M]^{-1}[N]$ and $[N]^T[M]^{-1}$

$B = [A][P][A]^T$, which is the desired form. (C.7)

$$[A] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1/3 & 0 \\ 0 & 1 & 0 & -1/3 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Also, manipulating (C.7),

$P = [A]^{-1}[B][A]^{T^{-1}}$. (C.8)

So, given a Coons matrix P, or a Bezier matrix B, we can transform to the other using (C.7) or (C.8).

LIST OF REFERENCES

REFERENCES

1. Blinn, J.F., "Computer Display of Curved Surfaces", Ph.D. Diss., Computer Science Dept., University of Utah, Salt Lake City, Utah, 1978.

2. Bui-Toung, Phong, "Illumination for Computer Generated Pictures", Ph.D. Diss., Computer Science Dept., University of Utah, Salt Lake City, Utah, 1973.

3. Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces", UTEL-CSC-74-133, 1974.

✓

4. Clark, J.H., "A Fast Scan-line Algorithm ~~for Rendering~~ Parametric Surfaces", Proceedings of SIGGRAPH 1979.

5. Cohen, D., "Incremental Methods for Computer Graphics", ESD-TR-69-193, Ph.D. Diss., Harvard Univ., 1969.

6. Crow, F.C., "A Comparison of Anti-Aliasing Techniques", IEEE Computer Graphics, Jan. 1981, pp. 40-48.

7. Gouraud, H., "Continuous Shading of Curved Surfaces", IEEE Transactions on Computers, Vol. C-20, No. 6, June 1971, pp. 623-629.

8. "Graphics Utah Style - 80", Manual, University of Utah, Salt Lake City, Utah, 1980.

9. Newell, M.E., Newell, R.G., and Sancha, T.L., "A Solution to the Hidden Surface Problem".

10. Newman, W.M., Sproull, R.F., "Principles of Interactive Computer Graphics, McGraw-Hill, 1979.

11. Ratliff, F., "Mach Bands: Quantitative Studies on Neural Networks on the Retina", Holden-Day, S.F., 1965.

12. Rogers, D.F., Adams, J.A., "Methematical Elements for Computer Graphics", McGraw-Hill, 1976.

13. Strang, G., "Linear Algebra and its Applications", Academic Press, 1976.

14. Sutherland, I.E., Sproull, R.F., and Schumacher, R.A., "A Characterization of Ten Hidden Surface Algorithms", Computing Surveys, Vol. 6, No. 1 March 1974, pp. 1-55.

15. Warnock, J.E., "A Hidden Surface Algorithm for Computer-Generated Half-Tone Pictures", Computer Science Dept., University of Utah, TR4-15, June 1969.

16. Watkins, G.S., "A Real-Time Visible Surface Algorithm", Computer Science Dept., UTECH-CSC-70-101, June 1970.

17. Whitted, T., "An Improved Illumination Model for Shaded Display", Communications of the A.C.M., Vol. 23, Num. 6, June 1980, pp. 343-349.

18. Williams, L., "Casting Curved Shadows on Curved Surface," Proceedings of SIGGRAPH 1978, Atlanta, GA., pp. 270-274.