1985

Survey of Texture and Shading Techniques for Visual Flight Simulation

Walter R. Steiner
University of Central Florida
SURVEY OF TEXTURE AND SHADING TECHNIQUES FOR VISUAL FLIGHT SIMULATION

BY

WALTER R. STEINER
B.S., Pennsylvania State University, 1978

RESEARCH REPORT

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering University of Central Florida Orlando, Florida

Spring Term 1985
ABSTRACT

Shading and texturing are two techniques that flight simulators can take advantage of to increase scene realism. Shading imitates the effects of light reflecting from a surface. Texture refers to superimposing a pattern on a surface to give the illusion of extra detail and realism.

In this report, several techniques for shading and texturing are evaluated with respect to their applicability to visual flight simulators.

The image quality produced by shading and texturing is a function of computational cost. The Phong model is found to produce the most realistic shading, but is too computationally expensive. The Gouraud shading model improves upon the Phong model in that realistic shading is produced with less computational effort. The table look up technique was found to be the most flexible and realistic way to produce texture on the surface of a polygon. It is shown that true perspective shading is cost effective when texture is required because the hardware needed to produce texturing and true perspective shading are very similar.
TABLE OF CONTENTS

LIST OF FIGURES .................................................. iv
GLOSSARY ............................................................ v
INTRODUCTION ......................................................... 1

Chapter
I SHADING TECHNIQUES ............................................ 4
   Polyhedron Shading .............................................. 14
   The Phong Shading Model ....................................... 17
   The Gouraud Shading Model .................................... 26
   Shading Using Gradients ....................................... 30
   True Perspective Shading ...................................... 33

II TEXTURING ......................................................... 42
   The Texture Model ............................................... 43
   The Texture Mapping Function ............................... 46
   The Texture Pattern Generator .............................. 49
   Texture Blending ............................................... 54
   Texture Anti-Aliasing Techniques ........................... 57

III CONCLUSION ..................................................... 61

REFERENCES ........................................................ 66
LIST OF FIGURES

1. The Eye and the World Coordinate systems . 11
2. Examples of Polyhedron Shading .............. 16
3. Average Vertex Normal derivation ............ 19
4. Interpolating to Find surface normal vectors .................................. 19
5. Example of Gouraud Shading ................... 28
6. Interpolation with a Concave Polygon ....... 29
7. The functional Elements Needed for Texture .................................. 45
8. Textured Scene ................................. 64
9. Textured Scene ................................. 65
GLOSSARY

Aliasing - Stair step-like effect which occur along boundaries of discontinuous color due to the quantization effects of pixels.

Edge - A line segment formed between two vertices.

Pixel - The smallest unit of video or color of which the view screen is divided into.

Polygon - A two-dimensional surface defined by vertices that are coplanar.

Surface - The exterior of an object or a plane or curved two-dimensional locus of points. In flight simulation, a surface is usually a polygon.
INTRODUCTION

The term "visual flight simulation" refers to training pilots, navigators, or anyone operating a vehicle on a simulator in which a computer generates the images that the trainee would otherwise see in a real aircraft. The trainee sits in a mock cockpit. But, instead of seeing the real scenery out of the cockpit, the trainee views a simulated image generated by a computer that, as closely as possible, mocks the real scenery.

To accomplish this, the computer must execute graphics algorithms that transform a geometric description of the scenery into a realistic visual image of what the trainee would see. The image must be updated at a rate such that continuous, jitter-free motion is achieved. This rate is usually greater than 30 frames or updates per second (Schachter, 1983).

Unfortunately, the amount of computer processing needed to execute these graphics algorithms at a thirty hertz rate or greater is extremely expensive. Special purpose computers must be designed to handle the large computational requirements and high data rates needed.
Even then, a typical simulator can cost anywhere from $1 million to $100 million. To simplify the algorithms as much as possible, surfaces are modeled with polygons. Higher order parametric equations which describe a surface are much more expensive in terms of computer time than linear equations that describe polygons.

Modeling surfaces with polygons tends to make the surface look faceted. The more polygons that are used to model the surface, the more realistic the surface looks. As the number of polygons increases, however, the amount of processing needed to compute an image also increases. As a result, in order to cut costs, compromises are made to remove as much detail as possible yet keep the scene as realistic as possible.

By applying shading and texturing techniques, the realism of an image can be greatly improved without adding more polygons. Shading techniques can be used not only to give realistic shading effects caused by light sources, but can also give a smooth continuous appearance to a surface modeled with polygons. Texture techniques can be applied to surfaces to give a textured appearance such as a plowed field or ripples of sand in a desert. Photographs of the actual terrain can be digitized and "painted" over a surface to give an image
of photographic-like quality.

Various shading and texturing techniques that can be applied to visual flight simulators are presented in the following chapters.
A shading model simulates, as closely as possible, the behavior of light on a surface. A shading model has two main properties: properties of the surface and properties of the illumination falling on the surface. The principal property of the surface is its reflectance, which indicates how much of the light which strikes the surface is reflected. If a surface has a different reflectance for light of different wavelengths, the surface will appear to be colored. If a surface is textured, the reflectance will vary with position on the surface. Another property of a surface that plays a role in shaded images is transparency. A surface may allow some light to be transmitted through it from behind (Newman and Sproull, 1979, p. 392).

Illumination plays an important role in the shading model. A scene may have some illumination that is uniform in all directions, called diffuse illumination (Newman and Sproull, 1979, p. 390). In addition, there may be point sources of light in the scene which cause
another type of reflection (called specular reflection) to appear on surfaces. Specular reflection occurs at the surface of a mirror. All light striking a surface is reflected, and in nearly a single direction (Harrington, 1983, p. 360). For example, specular reflection may appear as the glare from a shiny surface.

The objective of a shading model is to determine the energy radiating from a surface. A shading model uses the properties of the surface, and the properties of the illumination falling on the surface to approximate the energy radiating from a surface. The shading model can be decomposed into the sum of three parts: one part from diffuse illumination; one part from specific light sources; and one part from transparency (Newman and Sproull, 1979, p. 390). The sum of these parts give the total energy radiating from a surface.

The energy reflected from diffuse illumination, defined as diffuse reflection, can be represented by the product of the reflection coefficient of the surface and the diffuse illumination falling on the surface. The reflection coefficient is the ratio of the amount of light striking a surface to the amount of light reflected from a surface.
Transparency adds an energy contribution which is related to the energy arriving from behind a point on the polygon. The energy due to transparency can be represented by the product of the transmission coefficient and the energy arriving behind the polygon. The transmission coefficient indicates the amount of energy that the surface allows to pass through it (Newman and Sproull, 1979, p. 393).

Shading contributions from specific light sources will cause the energy radiated from a surface to vary as the surface orientation with respect to the light sources changes, and will also include specular reflection effects. Lambert's law describes how the energy radiated from a surface varies as the orientation to the surface changes. Lambert's law states that the energy falling on a surface varies as the cosine of the angle of incidence of the light (Newman and Sproull, 1979, p. 391). The angle of incidence is defined as the angle between a vector normal to the surface at a point p, and a vector that indicates the direction of the light ray which strikes the surface at a point p.

Specular reflection occurs when a surface acts as a mirror, that is nearly all the light striking the surface is reflected, and nearly all the light is
reflected in a single direction. The principal reflected ray is defined to be a unit vector that indicates the direction of nearly all the reflected light. The principal reflected ray leaves the point on the surface with an angle of reflection equal to the angle of incidence. The angle of reflection is defined as the angle between the vector normal to the surface at point p, and the principal reflected ray which is reflected from point p.

The total energy arriving at the eye from point p on a surface is the sum of the individual energies stated in Equation 1 (Newman and Sproull, 1979, p. 393).

\[
Et = (Tc)(Eb) + (R)(Is)\cos(i) + (R)(Id) + \sum_{n}^{(W(i))}(Is)[\cos(s)]
\]

where

- \(Et\) is the total energy radiating from point p.
- \(W(i)\) is the specular reflection coefficient at point p and is a function of the angle of incidence.
- \(Tc\) is the transmission coefficient at point p.
- \(R\) is the reflection coefficient at point p.
- \(Eb\) is the energy arriving from behind the polygon at point p.
- \(Id\) is the diffuse illumination in the scene.
- \(i\) is the angle of incidence at point p.
- \(s\) is the angle between the principal reflected ray at point p and the observer.
- \(Is\) is the illumination at point p due to a specific light source.
- \(n\) is a number indicating how shiny the surface is at point p.
The exponent $n$ ranges from about 1 to 10 and indicates how shiny the surface is. Shiny surfaces have a larger $n$ than dull surfaces. The function $W(i)$ is the specular reflection coefficient which is a function of the angle of incidence. Light striking a surface at a larger angle of incidence will reflect more light (Newman and Sproull, 1979, p. 392). The function $W(i)$ ranges between 10 and 80 percent and is larger for highly reflective materials.

The product of $Tc$ and $Eb$ represents the energy due to transparency. The product of $R$ and $Id$ represents the energy from diffuse reflection. The product of $R$, $Is$, and $\cos(i)$ represents the energy contribution of a specific light source. The product of $W(i)$, $Is$, and $\cos(s)$ to the $n$th power is the energy contribution due to specular reflection from a specific light source. It should be noted that the color of specular reflection is the color of the light source and not the color of the surface (Newman and Sproull, 1979, p. 392). This is certainly true if the quality of the mirror is very good. However, the author notes that if the quality of the mirror is not very good, it would seem likely that the color of reflection be a combination of the color of the incident light, and the color of the surface.
The cosine of the angle $s$ for specular reflection can be found by a method that Phong derived. Phong states that the three components of the unit vector that indicates the direction of the principal reflected ray can be found as shown in Equation 2 (1975).

$$V_{pox} = 2(N_z)(N_x)$$

$$V_{poy} = 2(N_z)(N_y)$$

$$V_{poz} = 2(N_z) - 1$$

where

$N_x$, $N_y$, and $N_z$ are the components of the surface normal.

$V_{pox}$, $V_{poy}$, $V_{poz}$ are the components of the unit vector from point $p$ to the observer.

The three components are derived assuming that the $z$ axis points in the direction of the light source. The cosine of the angle $s$ is the inner product of the surface normal and the vector $V_{po}$. Neither Phong or Newman and Sproull explain what happens when the angle $s$ is greater than 90 degrees and the cosine of $s$ is negative. The derivation of Equation 2 can be found in Newman and Sproull, page 394.

Equation 1 is used to find the energy radiating from point $p$ on a surface. However, realistically the energy is calculated for a small area of the surface referred to as an elemental area rather than a point.
Equation 1 can be used to calculate the energy radiating from each elemental area of the surface. Equation 1 can be used to calculate the energy of an elemental area illuminated by more than one light source. The energy contributions due to each specific light source can be summed together.

The energy found by Equation 1 can be used to indicate the intensity of each elemental area of a surface. Technically, the intensity of a wave is defined as the amount of energy falling on an area per unit time (Halliday and Resnick, 1974, p 308). However, in this report, intensity will refer to the relative brightness of an elemental area, and Equation 1 can be used directly to find the intensity or relative brightness of an elemental area.

For purposes of explanation, it is helpful to define a coordinate system as shown in Figure 1. Figure 1 shows a coordinate system with the observer at the origin, a view screen, and a polygon lying in the field of view. The z axis intersects the view screen at the center of the view screen, and is perpendicular to the view screen. The observer's line of sight, or boresight is along the z axis. Points that lie in three-dimensional x, y, and z space must be projected
Figure 1. The Eye and World Coordinate Systems
onto the two-dimensional view screen. The coordinates of the view screen will be designated by $y'$ and $x'$. Any point in the field of view can be projected onto the view screen by Equation 3. (Newman and Sproull, 1979, p. 357).

$$x' = \frac{x}{z}(z')$$

$$y' = \frac{y}{z}(z')$$

where

- $x'$ is the $x$ coordinate of point $p$ projected onto the view screen.
- $y'$ is the $y$ coordinate of point $p$ projected onto the view screen.
- $z'$ is the $z$ coordinate of the view screen.
- $x$ is the $x$ coordinate of point $p$.
- $y$ is the $y$ coordinate of point $p$.

The coordinate system at the observer's eye will be referred to as the eye coordinate system. The coordinate system in which the scene is modeled will be referred to as the world coordinate system. A transformation matrix can be used to transform all points defined in the world coordinate system to points defined in the eye coordinate system (Newman and Sproull, 1979, p. 333). As the observer and correspondingly the eye coordinate system change position relative to the world coordinate system, the transformation matrix is calculated and used to transform all points (as well as normal vectors and
gradients) from the world coordinate system into the eye coordinate system. It will be assumed that all points considered in the following discussions have been transformed into the eye coordinate system and lie inside the field of view, unless otherwise specified.

Since most visual flight simulators use polygons to model a surface, a shading model that applies to surfaces which are modeled with polygons is needed (Schachter, 1983). Since the surface is composed of polygons, the shading model should both reproduce realistic shading effects and give the illusion of a smooth continuous surface. The model must also be computationally efficient for real time requirements needed in flight simulation. The design of the shading model is a compromise between fidelity and computing cost. Exact replication of the effects of light on a surface can be very expensive in terms of computations. Ray tracing algorithms which produce very realistic shading can be used to project shadows or calculate the lighting effects caused by light reflecting off of nearby objects. Such algorithms can take from 44 to 122 minutes of VAX 11/780 time which make them prohibitive for real time use (Whitted, 1980).
Fortunately, such elaborate shading models are not necessary for visual flight simulation. Approximations can be used wherever possible to reduce the number of computations to a reasonable level. Trade-offs in a shading model are especially difficult because the properties of the human visual system influence the perception of realism. Approximations in the shading model that lead to unrealistic shading effects to the viewer must be avoided (Newman and Sproull, 1979, p. 389).

**Polyhedron Shading**

The polyhedron shading model assumes that the light rays from a light source are parallel, uniform over the surface, and constant in intensity over the entire scene (a reasonable assumption in flight simulation since the light source is usually the sun). This means that the illumination from a light source is uniform for each point on the surface.

The elemental area for which the shading intensity is to be found is the entire polygon. This means the intensity is uniform over the entire polygon. The effects of transparency and specular reflection are ignored. The intensity of the polygon can be found from
Equation 1, ignoring the effects of transparency and specular reflection. The normal vector needed to find the cosine of the angle of incidence is a unit vector normal to the polygon. The cosine of the angle of incidence is found by the inner product of the polygon unit normal and the light direction vector LV.

Polyhedron shading is the simplest of all shading models and offers only a very rough approximation of the true effects of light reflecting from a surface. Polyhedron shading is not useful for shading polygons that are used to model curved surfaces. Since the intensity of the polygon remains uniform over the entire polygon, individual polygons that make up the surface will be distinctly noticeable, giving the surface a faceted appearance. Polyhedron shading is useful for shading a surface that is intended to be flat, e.g., the sun shining on a planar surface such as a roof. The General Electric Company uses polyhedron shading techniques for this purpose (Bunker, 1984). Figure 2 shows a human face and a car modeled with polygons and shaded using the polyhedron shading model. Notice the polygons which make up the face are uniform in color which gives the surface a faceted appearance.
Figure 3. A Volkswagen and a Human Face Modeled with Polygons and Shaded with the Polyhedron Shading Model.
The Phong Shading Model

The Phong shading model, unlike the polyhedron shading model, considers computing a new shading intensity for many elemental areas on the polygon. In the case of flight simulators, curved surfaces are modeled with polygons, that is, a group of polygons may be intended to represent a curved surface. When shading techniques are applied to a group of polygons intended to represent a curved surface, the elemental areas of each polygon can be treated as if they lie on the actual curved surface instead of the polygons used to model the surface. By treating the elemental areas as though they lie on the actual curved surface, the shading effects will give the polygons the appearance of a curved surface.

The primary difference between an elemental area that lies on a polygon, and an elemental area that lies on a curved surface is the orientation of the surface normal vector. By using the vector normal to the curved surface rather than the vector normal to the polygon used to model the curved surface in Equation 1, the shading intensity can be found for the curved surface. The normal to any point on a surface can be found by using the derivative of the equation of the surface if
known. However, there is a much simpler approximation that can be used. When polygons are used to model a surface, averaged vertex normals defined at each vertex of a polygon can be used to express the curvature of the surface modeled with polygons. An averaged vertex normal is a unit normal vector found by averaging all the normal vectors of each polygon that share a vertex. Figure 3 is an illustration of four polygons each with a normal vector that is used to find the average vertex normal vector at a vertex (Newman and Sproull, 1979, p. 401).

The surface normal vector at any elemental area on the interior of a polygon can be found by interpolating between the averaged vertex normals. The normal vector for any elemental area on the interior of the polygon will be referred to as surface normal vectors. The interpolation procedure is shown in Figure 4. Points A, B, C, and D are vertices of a polygon. Averaged vertex normals are defined at each vertex. The surface normal at point L is found by Equation 4.

\[
\begin{align*}
L_n(x) &= \frac{[L(x) - A(x)]/[A(x) - B(x)]}{[A(x) - B(x)]} [A_n(x) - B_n(x)] \\
L_n(y) &= \frac{[L(y) - A(y)]/[A(y) - B(y)]}{[A(y) - B(y)]} [A_n(y) - B_n(y)] \\
L_n(z) &= \frac{[L(z) - A(z)]/[A(z) - B(z)]}{[A(z) - B(z)]} [A_n(z) - B_n(z)]
\end{align*}
\]

where
Figure 3. Averaged Vertex Normal Vectors are Derived by Averaging Polygon Normals of Neighboring Surfaces.

Figure 4. Point P is Found by Interpolating Points A, B, C, and D.
Ln(x), Ln(y), and Ln(z) are the x, y, and z components of the surface normal at point L.

An(x), An(y), and An(z) are the x, y, and z components of the average vertex normal at point A.

Bn(x), Bn(y), and Bn(z) are the x, y, and z components of the average vertex normal at point B.

A(x), A(y), and A(z) are the x, y, and z components of the point A.

B(x), B(y), and B(z) are the x, y, and z components of the point B.

L(x), L(y), and L(z) are the x, y, and z components of the point L.

The surface normals at all elemental areas must be normalized. The surface normal at point R can be found in a similar manner. When surface normals at points L and R are found, the surface normal for any elemental area along the line segment LR can be found.

So far, only the shading intensity of elemental areas that lie on a curved surface has been considered. However, the objective is to find the shading intensity of elemental areas that lie on a view screen. The view screen is divided up into elemental areas. The finer the resolution of the picture, the more elemental areas there are. The elemental areas which divide up the view screen are called pixels. When a polygon is projected onto a view screen, the polygon will lie on a number of pixels. The shading intensity of each one of these
pixels which lies on the polygon, must be computed because it is the shading intensity of the pixel that is sent to the display device, and ultimately viewed by an observer. To find the shading intensity of a pixel, the polygon can be projected using Equation 3 before interpolation. After the polygon has been projected, the pixels that the polygon lies on can be found because the polygon is in terms of view screen coordinates \( x' \) and \( y' \). Then the interpolation procedure can be executed as shown in Equation 4, but instead of using \( x \) and \( y \), \( x' \) and \( y' \) are used. Using \( x' \) and \( y' \) in Equation 4 will give the shading intensity of a pixel but will introduce an error in the shading intensity. By projecting the polygon, the shape of the polygon will change, and, as a result, the interpolation results will be different. Since the interpolation process is an approximation anyway, and if the polygons used to model an object are relatively small in size (projection does not change the shape of the polygon drastically), the error will be small.

The cosine of the angle of incidence will be equal to the inner product of the surface normal and the unit vector indicating the direction of the light source. When the cosine of the angle \( i \) and the cosine of the
angle \( s \) are found, Equation 1 can be used to find the intensity of a pixel. The cosine of the angle \( s \) is the inner product of the vector from the observer to the point on the polygon and the direction of the reflected light ray. Since Equation 1 must be calculated for every pixel which lies on a polygon, the number of calculations can become prohibitive. If the intensity were to be calculated for every pixel of a screen, and the screen had \( 1024 \times 1024 \) pixels, Equation 1 would have to be calculated more than one million times per image update.

The effect of treating an elemental area as though it lies on the actual curved surface rather than a polygon will vary the intensity across a polygon as if the polygon were curved, and if properly sampled the intensity will be continuous across the polygon and polygon boundaries, giving the illusion of a smooth continuous surface; that is, the intensity will vary gradually across the polygon and the polygon boundaries in such a way that the eye would not perceive a noticeable change. If not sampled properly, a phenomenon known as Mach bands may occur. Mach bands were investigated by Mach who described how the retina performs a two dimensional filtering on a shaded object.
A Mach band occurs when the incremental change in intensity across an organized boundary is great enough for the eye to perceive a noticeable change. Each neuron, depending on the intensity of the light it receives, interacts with its neighbors and modifies their performances. The result of this interaction will be an attenuation of the low spatial frequencies present in the shading (Gouraud, 1971). A Mach effect of the first order will appear to enhance the discontinuity in the shading intensity by brightening the brighter side of the discontinuity, and darkening the darker side. A Mach effect of the second order will appear when the rate of change of intensity across a boundary is discontinuous and will appear as a bright band at the discontinuity. Phong states:

The linear interpolation scheme used here to approximate the orientation of the normal does not guarantee a continuous first derivative of the shading function across an edge of a polygonal model. In the extreme cases where there is an abrupt change in the orientation of two adjacent polygons along a common edge, the subjective brightness due to the Mach Band effect will be visible along this edge. However this effect is much less visible in the described model than in the Gouraud smooth shading model.

Phong also observes:
Also, an interesting fact discussed previously on Mach effect shows that this effect is visible whenever there is a great change in the slope of the intensity distribution curve, even if the curve has a continuous first derivative.

Gouraud states:

From the explanation of the Mach band distortion it appears that in order to represent correctly the smooth aspect of a curved surface, the shading rule on this surface has to be continuous in value and, if possible, in derivative. One way to achieve this would be to increase the number of polygons approximating the surface, but this would be impractical for storage and time reasons.

From these quotes it seems that there is always a possibility for Mach Bands to occur but by using the Phong Shading Model and modeling the surface with an adequate number of polygons, the Mach effect can be minimized.

Simplifications can be made to the Phong shading model. For example, specular reflection and shading effects due to transparency can be ignored. Since the light source is usually the sun, the light rays from the sun can be assumed to be parallel, uniform over the entire scene, and constant in intensity over the entire scene. Also the effects of light which has been reflected off of other objects, as well as shadows can be ignored. The reflectance coefficient for a surface
can also be assumed to be constant. Gouraud proposed a further simplification by interpolating between shading intensities at each vertex rather than averaged vertex normals at each vertex. Interpolating between shading intensities involves calculating the shading intensities at each vertex using Equation 1, then interpolating these shading intensities at each vertex as discussed previously to find the shading intensities of each elemental area.

Phong stresses that finding interior surface normals by interpolating between averaged vertex normals given at each vertex, gives better shading results than simply interpolating shading intensities calculated at each vertex of a polygon. Care must be taken to normalize the interior surface normals when the surface normals are calculated. Then, when the interior surface normals are found, the shading normals can be used to find the shading intensities. The reason interpolating shading normals gives better results can be seen in the case of a polygon lying perpendicular to the light ray. In the case of interpolating average vertex normals, a surface normal is found by interpolating, normalizing the surface normal, then computing the shading intensity by using Equation 1. For the case of interpolating
between shading intensities however, the two shading intensities that are used for interpolation may have the same value, and therefore the shading intensity will be uniform across the polygon, giving a faceted effect. When interpolating shading normals, it may be tempting to omit the normalizing step, with the intent of reducing the number of calculations. However, if this is done the results will be exactly the same as interpolating between shading intensities.

The Gouraud Shading Model

Henri Gouraud proposed a shading model that uses less computations than the Phong model by calculating shading intensities at each vertex using Equation 1, then using interpolation techniques to find the interior shading intensities of each pixel. For the Gouraud model, Equation 1 need only be calculated for each vertex as opposed to each pixel for the Phong model. Also, interpolating shading intensities takes less computation than interpolating normals.

If the modeler models a curved surface with a sufficient number of polygons, the Gouraud model will achieve very good results. The object will look smooth and very realistic. Singer Link, General Electric and Evans and Sutherland use the Gouraud shading model in
their flight simulators (Schachter, 1983). Figure 5 is a human face and a car modeled with polygons and shaded with the Gouraud shading model. Notice that the individual polygons used to model the face are invisible and the face looks as though it were modeled with a smooth continuous curved surface.

The Gouraud approach is not without its deficiencies. If the modeler is not careful, Mach bands can appear at the edges between adjacent polygons. Concave shaped polygons may introduce shading anomalies with the Gouraud and the Phong model (Newman and Sproull, 1979, p. 402). A convex-shaped polygon is a polygon such that for any two points inside the polygon, all points on the line segment connecting them are also inside the polygon. A concave polygon is one that is not convex (Harrington, 1983, p. 60). The reason for these anomalies is that the interpolation method used to calculate shading intensities on the polygon can give discontinuities as shown in Figure 6. The shading intensity at point p will be the shading intensity produced by the averaged vertex normal defined at point p. However, the shading intensity at a point that is an incremental distance from p, point p1 for example, will be found from the shading intensity using interpolation
Figure 5. A Volkswagen and a Human Face Modeled with Polygons and Shaded with the Gouraud Shading Model.
between points A and B. Since points A and B have no relationship to point p, the shading intensity at point p may be noticeably different from point p1.

![Figure 6. Concave Shaped Polygon](image)

Interpolation is only an approximation and may shade the object unrealistically. But for flight simulation it gives the trainee a sense of realism and depth. Evans and Sutherland enhance the Gouraud shading model by interpolating the actual red, green, and blue components of the color that are specified at each vertex. Previously only the shading intensity was considered. The shading intensity found from the shading model is used to modify the color of a polygon. For example, the shading intensity may be used to vary the color of a polygon from bright green to dark green. For shading due to lighting effects, this is the desired effect. With the Evans and Sutherland approach, the
color of a polygon can change from red at a vertex to orange at another vertex. This enhancement is not used to represent accurate shading but for special effects such as a red-orange sunset, or exhaust flames from the output of a jet engine.

Shading Using Gradients

The General Electric Company used a reduced version of Gouraud shading for a system at Williams Air Force Base (Schachter, 1983). In this system, a simplification was added in the interpolation of shading intensities. Instead of interpolating to find the interior shading intensities as was described in the previous section on the Gouraud shading model, an intensity gradient which described the rate of change, and direction of change of the shading intensity across a polygon was calculated.

Like the Gouraud model, averaged vertex normals are defined at each vertex, and shading intensities at each vertex are calculated. An intensity gradient of the form shown in Equation 5 can be found from the intensity of any three vertices on the polygon. The coefficients, C1, C2, and C3 of the intensity gradient can be found by solving the system of equations shown in Equation 6.
Gradient = C1(x) + C2(y) + C3

where

C1, C2, C3 are the coefficients of the gradient.

I1 = C1(x'1) + C2(y'1) + C3
I2 = C1(x'2) + C2(y'2) + C3
I3 = C1(x'3) + C2(y'3) + C3

where

I1, I2, I3 are the shading intensity values at vertex 1, 2, and 3 respectively.
x'1, y'1 are the coordinates of projected vertex 1.
x'2, y'2 are the coordinates of projected vertex 2.
x'3, y'3 are the coordinates of projected vertex 3.

The intensity gradient is calculated as a function of the intensity change in the x' and y' directions. The shading intensity of any adjacent point can now be found by incrementing the shading intensity by the intensity gradient in the x' or y' direction respectively. Finding interior shading intensities by this incremental approach rather than by interpolation simplifies the hardware considerably.

The rate of change of intensities is linear across the polygon (Bunker, 1984). The restriction when using this type of shading is to guarantee continuous shading at all vertices and edges, the polygons must all be
triangles. For example, if a four-sided polygon were used, the shading intensity at vertex W which was not used to calculate the intensity gradient would not necessarily match the value predicted by the intensity gradient. However, for the adjacent polygon which shares vertex W, the shading intensity at vertex W will match the value predicted by the intensity gradient for the adjacent polygon if vertex W is used to derive the intensity gradient for the adjacent polygon. The result will be a discontinuity at the vertex W. Polygons that have more than three sides must be broken up into two or more polygons, each with three sides. Since clipping can produce non-triangles from triangles, the resulting non-triangle must also be broken up in triangles and the intensity gradient calculated for the triangles. Clipping is the process of truncating polygons at the view screen boundaries. More information on clipping can be found in Newman and Sproull.

The author notes that although hardware is reduced, the number of polygons and edges due to restricting polygons to triangles increases, and therefore the processing time and storage also increase.
True Perspective Shading

The shading models discussed previously did not take into the account the effects of projecting a polygon lying in three-dimensional space onto a two-dimensional viewing screen. Shading intensities were calculated at each vertex in three-dimensional space, then the polygon was projected onto a viewing screen, then the interior shading intensities for each pixel of the polygon were found by linear interpolation or incremental techniques using the projected vertices. However, three-dimensional space does not map into two-dimensional space linearly with the distance from the observer as seen from Equation 3. An error is introduced into the shading model by using linear interpolation techniques. When a polygon is projected, the polygon may change shape due to the viewpoint. For example, an observer views a runway from overhead. When the runway is projected the runway will appear as a rectangle. Now suppose the observer's position changes, and the observer is standing at the beginning of the runway looking down the runway. Due to projection, the runway looks to the observer as if the runway vanishes into the distance, and appears shaped as a triangle.
The shading intensity of a point p on the runway shaped as a rectangle should be the same as the shading intensity of point p on the runway shaped as a triangle. However, because the linear interpolation is done after projection, the shading intensities of pixel p on the rectangle and on the triangle-shaped runway will not necessarily be the same. This effect may produce an undesirable shift in intensity on the polygon as the viewpoint changes.

The General Electric Company uses a shading model that correctly maps shading intensity in three dimensions to a shading intensity in two dimensions. Like the previous section, the rate of change of the shading intensity is assumed to be linear in the plane of the polygon. A shading intensity for any point on a polygon can be expressed in the form given by Equation 7.

\[ I = C_1(x) + C_2(y) + C_3(z) + C_4 \]  

(7)

where

I is the intensity.
x is the x coordinate of the polygon.
y is the y coordinate of the polygon.
z is the z coordinate of the polygon.
C1, C2, C3, and C4 are the coefficients which define the intensity I of the polygon.
Because all points \( x, y, \) and \( z \) lie on a plane, \( z \) can be solved for in terms of \( x \) and \( y \) from the equation of the plane shown in Equation 8 (Newman and Sproull, 1979, p. 356). The value \( d \) is found by taking the inner product of any point on the polygon with the polygon normal vector.

\[
z = \frac{-Nx(x) - Ny(y) - d}{Nz}
\]  

(8)

where

- \( Nx \) is the \( x \) component of the normal vector of the polygon.
- \( Ny \) is the \( y \) component of the normal vector of the polygon.
- \( Nz \) is the \( z \) component of the normal vector of the polygon.
- \( d \) is the length of the vector which is perpendicular to the polygon from the origin to a point on the polygon.

Now Equation 8 can be substituted for \( z \) in Equation 7, and the coefficients rearranged and solved from a system of equations similar to Equation 6.

The system of equations solution requires many calculations. There is a better method which reduces the number of calculations. Suppose instead of defining an intensity gradient as shown in Equation 7, a gradient is defined as in Equation 9.
\[ \text{Grad}(i) = P_0 + \text{inner product}(\text{Grad}, \text{Vci}) \] (9)

where

\text{Grad}(i) \text{ is the value of the Gradient at point } i \text{ on the polygon.}

\text{P}_0 \text{ is the intensity at the center of the polygon.}

\text{Vci} \text{ is a vector from the center of the polygon to point } i \text{ on the polygon.}

\text{Grad} \text{ is a gradient defined by coefficients } C_1, C_2, \text{ and } C_3.

It is important to note that the gradient, \text{Grad}, is defined relative to the center of the polygon. The reason for defining the gradient this way is that now a universal gradient can be defined for the polygon, no matter where the observer is positioned relative to the polygon. If the intensity gradient is found from a system of equations, each time the observer changes position a new intensity gradient must be calculated because the \( x, y, \) and \( z \) position of the polygon changes.

By defining the gradient relative to the center of the polygon, only the \text{P}_0 \text{ term need be updated. This can be shown by the following derivation. The vector } \text{Vci} \text{ is equal to the difference of } \text{Voi} \text{ and } \text{Voc}, \text{ where } \text{Voi} \text{ is the vector from the observer to point } i \text{ on the polygon, and } \text{Voc} \text{ is the vector from the observer to the center of the polygon. Substituting this for } \text{Vci} \text{ in Equation 9 yields:}
Grad(I) = Po - inner product(grad, Voe) +
+ inner product(grad, Voi) \tag{10}

The inner product of grad and Voi is simply:

\[ IP = C_1(x) + C_2(y) + C_3(z) \tag{11} \]

where

$C_1, C_2, C_3$ are the coefficients of grad.

$x, y, z$ are components of the vector Voi.

Equation 11 is then expressed in terms of $y'$ and $x'$, the coordinates of the view screen. This, in effect, projects Equation 11 onto the view screen. Solving Equation 3 for $y$ and $x$ and substituting into Equation 5 yields the following.

\[ Ip = C_1[(x')(z)/(z')] + \tag{12} \]
\[ + C_2[(y')(z)/(z')] + C_3(z) \]

The equation of the plane solved for $z$ in terms of $x'$ and $y'$ is shown in Equation 13.

\[ z = d/[N_x(x')/(z') + N_y(y')/(z') + N_z] \tag{13} \]

Let:

\[ C_1 = C_1/z' \tag{14} \]
\[ C_2 = C_2/z' \tag{15} \]
\[ C_4 = (N_x)/[(-d)(z')] \tag{16} \]
Substituting Equations 13 into 12 and then Equations 12, 14, 15, 16, 17, and 18 into Equation 10 yields the final equation for intensity.

\[ I = P_c + \frac{[C_1(x') + C_2(y') + C_3]}{[C_4(x') + C_5(y') + C_6]} \]  

where

\[ P_c = P_0 - \text{inner product(\text{grad}, V_oe)} \]

Equation 19 gives the intensity of any point on a polygon in terms of the view screen coordinates \( x' \) and \( y' \). Equation 19 correctly maps the intensities of any point on a polygon from three dimensions to two dimensions, but with many more computations than the Gouraud shading model. The normal vector, \( V_oe \), \( \text{grad} \), and \( P_0 \) are all computed off-line and stored.

Equation 19 will produce the value of a gradient defined in the plane of a polygon lying in the field of view, in terms of the coordinates of the view screen, \( x' \) and \( y' \). The effects of the light source and diffuse illumination in the scene must now be incorporated into Equation 19. A gradient is defined which expresses the rate of change of each component of the surface normal with respect to the coordinate system.
\[ Nx\text{grad} = P_{ox} + C_{1x}(x) + C_{2x}(y) + C_{3x}(z) \]  
\[ Ny\text{grad} = P_{oy} + C_{1y}(x) + C_{2y}(y) + C_{3y}(z) \]  
\[ Nz\text{grad} = P_{oz} + C_{1z}(x) + C_{2z}(y) + C_{3z}(z) \]

Equation 20 expresses the rate of change of the x component of the normal with respect to the world coordinate system. Equation 21 expresses the rate of change of the y component of the normal with respect to the world coordinate system, and Equation 22 expresses the rate of change of the z component of the normal with respect to the world coordinate system. The surface normal vector components of any interior point x, y, and z on the polygon are \( Nx\text{grad}, Ny\text{grad}, \) and \( Nz\text{grad}. \) It is important to note that when Equations 20, 21, and 22 are rotated into the eye coordinate system (\( P_{ox}, P_{oy}, \) and \( P_{oz} \) are scalars and are not rotated), the coefficients express the rate of change of each component in the eye coordinate system, but with respect to the world coordinate system. The shading intensity of any pixel is the diffuse illumination plus the inner product of the surface normal vector and the light direction vector, \( LV. \) Since the surface normal vector is expressed as the rate of change of the normal with respect to the world coordinate system, the light vector must be expressed in terms of the world coordinate system.
system. The final shading intensity value of a pixel on a polygon expressed in terms of the viewing screen $x'$ and $y'$ is shown in Equation 23.

$$I = P'c + \left[ C_1'(x') + C_2'(y') + C_3' \right]$$

$$\left[ C_4(x') + C_5(y') + C_6 \right]$$

(23)

and

$$P'c = Po + Id + LVx(Pox) + LVy(Poy) + LVz(Poz)$$

$$C_1' = LVx(C_1 + C_2 + C_3)$$

$$C_2' = LVy(C_1y + C_2y + C_3y)$$

$$C_2' = LVy(C_1y + C_2y + C_3y)$$

where

LVx, LVy, LVz are the x, y, and z components of the light direction vector in world coordinates.

Id is the diffuse illumination.

The coefficients of Equations 20, 21, and 22 that describe the rate of change of the normal vector, and Pox, Poy, and Poz, are found for a polygon by solving a system of equations off line. By using the x components of any three averaged vertex normals corresponding to three vertices on the polygon, and the equation of the plane that the polygon lies in, the coefficients and Pox can be found using a system of equations. Likewise, the coefficient and scalars for Nygrad and Nzgrad can be found in a similar manner.
Usually true perspective shading is not necessary since surfaces are usually modeled with an appropriate number of polygons, the polygons are relatively small in size, and the error in intensity is small enough that the error is not noticeable to the observer. However, this is not true for texture as will be seen in the next chapter. The texture mapping function equation is of the same form as Equation 19. Therefore, much of the same hardware that is used to produce texture can be used to produce true perspective shading.
Natural scenes are rich in texture. The human visual system relies heavily on texture cues to perceive the structure in a scene. Texture can give the perception of air speed, height above the ground, tilt of the ground, and location in the environment. Texture also provides additional cues for the relative size of objects and their relationships to one another (Schachter, 1979). The appearance of texture on a surface will be referred to as a texture pattern. Patterns on a surface as seen from the air or at a distance, such as a dense cluster of trees or the detailed surface of a mountainside, are extremely useful in creating realistic scenery. It is very desirable to be able to create a texture pattern that exactly replicates the actual landscape as it exists in nature.
The Texture Model

Texture can be thought of as "painting" a pattern over a surface. As the viewpoint changes, the surface (polygons in this case) will change in orientation and size due to the perspective nature of the image. The texture pattern must be mapped onto the polygon in such a way that as the polygon changes in orientation and size, the texture pattern on the polygon changes correspondingly. This mapping function will be referred to as the texture mapping function.

The texture pattern that is to be used on the polygon must be generated. The algorithm or hardware that generates the texture pattern will be referred to as the texture pattern generator. Visual flight simulators require that the texture pattern generator must be capable of producing a wide variety of realistic texture patterns.

The distance of a polygon from the viewer can range from many miles to a few feet. This requires control of the level of detail of the texture pattern. As a polygon recedes into the distance, the level of detail of the texture pattern must decrease. If the texture pattern generator generates a continuum of levels of detail based upon the distance of the polygon from the
viewer, the level of detail problem is solved. Some texture pattern generators store texture patterns as a collection of texture patterns in a table look up, each texture pattern being at a different level of detail. When a table look up is used, the transition from one level of detail to another will be noticeable. A function that will provide a smooth transition from one level of detail to another is needed. This function will be referred to as the texture blending function.

Texture patterns can be very detailed. This can cause an aliasing problem. Aliasing occurs due to the quantization effects of composing an image with pixels. Aliasing becomes even more of a problem when motion is involved. Aliasing causes straight lines to appear to look like staircases. When the polygon moves, these staircase-like patterns will appear to ripple along the edge of the polygon. Anti-aliasing techniques can be used to correct the aliasing problem. The texture mapping function, texture pattern generator, texture blending, and texture anti-aliasing are the functions required for texture in a visual flight simulator. Figure 7 shows the functional elements needed for a typical texture system. The texture system requires position and orientation of the polygon as inputs. The
Figure 7. The Functional Elements Needed for Texture.
output of the texture system is information on what color or how to modify the color of the pixels that the polygon lies on so as to produce a texture pattern on the surface of a polygon. The texture mapping function, texture pattern generator, texture blending, and texture anti-aliasing techniques along with various approaches to the problems are discussed in detail in the next sections.

The Texture Mapping Function

The texture pattern must be mapped onto a polygon. In order to map a texture pattern on a polygon, one must know how the polygon is oriented relative to the viewer. The mapping function is determined by the orientation of the polygon. For example, if the polygon is oriented far away from the viewer, the polygon will decrease in size, and correspondingly, the texture pattern on the surface will squeeze together. This property is called compression (Schweitzer, 1983). A polygon that lies in a plane almost parallel to the observer will appear to vanish to a point. Likewise, the texture pattern must appear to vanish to a point. This property is called convergence (Schweitzer, 1983). The properties of compression are found by finding the depth of the polygon. The properties of convergence are found based
on the orientation of the polygon normal (Schweitzer, 1983).

The General Electric Company uses Equation 19, as previously discussed, as a texture mapping function (Chandler, 1984). However, for texture the coefficients C1, C2, and C3 are derived differently than for true perspective shading.

A texture gradient is defined which expresses the rate of change of the texture pattern and the direction of the texture pattern. This texture gradient is defined by three coefficients labeled C1, C2, and C3 (Chandler, 1984). A ground plane is defined as a large polygon that represents the surface of the earth. An initial value of the gradient at a point on the ground plane is labeled Po. The texture gradient is defined to lie in the ground plane. Polygons used to model the terrain of the image are placed on top of the ground plane. The texture gradient in the ground plane is then projected up into the polygons that are used to model the terrain as shown in Equation 24.
Grad(x proj) = \text{Grad}(z \text{ gnd}) + \\
- \text{Nx}[\text{inner product}(\text{Grad(gnd)}, N)]

Grad(y proj) = \text{Grad}(y \text{ gnd}) + \\
- \text{Ny}[\text{inner product}(\text{Grad(gnd)}, N)]

Grad(z proj) = \text{Grad}(z \text{ gnd}) + \\
- \text{Nz}[\text{inner product}(\text{Grad(gnd)}, N)]

\text{Pc} = \text{Po} + \text{inner product}(\text{Grad(gnd)}, V)

where

- Grad(x proj) is the x coefficient of the gradient projected into the polygon.
- Grad(y proj) is the y coefficient of the gradient projected into the polygon.
- Grad(z proj) is the z coefficient of the gradient projected into the polygon.
- N is the polygon normal.
- Nx, Ny, Nz are the x, y, and z components of the polygon normal.
- Grad(gnd) is the gradient in the ground plane.
- Grad(x gnd) is the x component of the gradient in the ground plane.
- Grad(y gnd) is the y component of the gradient in the ground plane.
- Grad(z gnd) is the z component of the gradient in the ground plane.
- V is a vector from the point where \text{Po} is defined on the ground plane to a point at the center of the polygon.

This process produces a projected gradient and initial value, for each polygon, that is continuous from one polygon to another.

Equation 19 is then used to solve for the texture value at any point on the polygon in terms of the view screen coordinates x' and y'. The texture value found from Equation 19 is then used as an address for the
final texture intensity stored in a table look up which is discussed in the next section.

**The Texture Pattern Generator**

The mapping function is used as an input to a texture pattern generating function. The pattern generating function must be capable of stretching or squeezing the pattern as the orientation and distance of the polygon changes. The pattern generating function must be capable of generating a very wide range of patterns that look realistic and yet are not too expensive in terms of computations. The pattern generator function must also be capable of generating patterns at different levels of detail. In visual flight simulation, the distance of a polygon can range from several feet to many miles away from the viewer. When the polygon is close to the viewer, the texture pattern should be very detailed. But as the polygon moves away, the polygon decreases in size and the texture pattern on the polygon becomes squeezed together to the point of being unrecognizable. The level of detail of the texture pattern must decrease as the polygon moves away from the viewer.
Several authors have described different methods of realizing the texture pattern generator function. Dino Schwietzer describes the use of texels as a means to generating texture patterns (1983). A texel is described as an individual texture element, in particular a circle. As the normal of the polygon changes the circle becomes distorted into an ellipse that reflects the change in orientation of the polygon. As the polygon recedes into the distance, the circles become smaller and move closer together. The texels can be put together to form a particular pattern on a polygon. Schweitzer makes no mention of how the level of detail will be implemented, or how texels will produce texture patterns that are very detailed.

Another way of producing a texture pattern is by modifying the reflection coefficient of the surface, then applying shading techniques so that as light reflects off of the surface, a texture pattern appears (Blinn, 1976). The problem of what function to use to calculate the reflection coefficient still exists. Blinn only considers texture on a surface described mathematically as a bivariate patch (Blinn, 1976). James Blinn extends this work further by perturbing surface normals both in magnitude and direction, and
then applying shading techniques to produce the appearance of wrinkled surfaces (Blinn, 1978). The merit of the method proposed by Blinn is that shading as well as texturing effects are produced in one operation. The problem is, once again, how to perturb the surface normals in such a way that yields a realistic texture pattern. These methods described by Blinn are mainly for use in computer graphics applications and do not take into account the problems encountered in flight simulation such as real time requirements, level of detail adjustment, and the effects of compression and convergence on the texture pattern mapped onto a polygon.

Fournier, Fussel, and Carpenter describe a method of producing texture patterns called fractal texture synthesis (Fournier, Fussel, and Carpenter, 1982). Fractal texture handles level of detail control very naturally. The fractal generating algorithm increases and decreases level of detail continuously as a function of distance. Fractal texture has an infinite level of detail, the closer the distance to the polygon, the more level of detail the fractal generator generates. Unfortunately, generating fractals requires quite a bit of calculation. However, the real problem is that the
generation technique is essentially a random one. The fractal generator will produce beautifully detailed patterns but it is very difficult to use a fractal generator to generate a very structured pattern which resembles realistic scenery in a visual flight simulator.

The most flexible technique and the one used in practice by the General Electric Company is the table look up technique (Chandler, 1984). The table look up technique stores a texture pattern in a table look up. The table look up contains texture intensity values. The output of the mapping function is used to address the table look up to access the correct texture intensity. Because a table look up is used, any conceivable texture pattern can be stored (Chandler, 1984).

A two-dimensional table look up can be used to produce two-dimensional texture patterns. Two texture gradients as described in the previous section are used. The texture values found from the two texture gradients are concatenated together to form an address for the texture map (Chandler, 1984). The two texture gradients are defined such that they are perpendicular to each other. The General Electric Company uses this approach
and calls it cell texturing (Bunker, 1984). Using this cell texturing approach, a digitized photograph can be decomposed into a two-dimensional table look up. The picture is quantized into a number of cells, each cell an entry in the texture table, hence the name cell texture. Cell texturing provides a method of generating any texture pattern simply by digitizing a photograph.

Dungan, Stenger, and Sutty propose a similar approach to cell texture called a texture tile (1978). Dungan, Stenger, and Sutty define a texture tile as a digital array of stored texture information. Texture tiles were developed specifically to add realism to the computer image displays used for pilot training. Like the General Electric approach, the mapping function is done by calculating the distance from the eye to the point on the surface. The texture pattern on the polygon is made up of texture tiles, each tile being one unit square.

The merits of a table look up to generate texture patterns are unlimited flexibility in variation of the texture patterns, and ease of generation of texture patterns by digitizing a photograph. The calculation time involves only a memory access per sample point. The disadvantages are that an extremely large amount of
fast memory may be needed, and level of detail must be stored in multiple level of detail tables which adds more memory and requires more computational capability to implement texture blending.

**Texture Blending**

When a table look up is used, texture patterns at different levels of detail can be stored to provide level of detail. Williams calls this idea "mip mapping." Since a finite number of levels of detail are stored, the transition from one level of detail to another is not continuous. Discontinuity between levels of detail will be noticeable whenever the texture mapping function determines that a transition from one level of detail to another is needed. Smoothing the transition between two levels of detail is called texture blending.

To produce multiple texture tables at various levels of detail, a square area of texture from the highest level of detail texture pattern can be averaged down to a lower level of detail (Williams, 1983). Early General Electric systems added the texture intensities of four neighboring cells of the texture table then divided by four to find the texture intensity of the cell at the next lower level of detail (Chandler, 1984).
The next lower level of detail contains half as many cells as the table at the next higher level of detail. The later General Electric systems use a method of level of detail that both improved blending as well as anti-aliasing but costs more memory. The number of cells in each texture table for each level of detail are the same. A cell of the next lower level of detail is derived from the weighted average of the corresponding cell of the next higher level of detail, and the eight neighboring cells of the corresponding cell of the next higher level of detail.

The final texture intensity is found by interpolating between the texture intensity of the cell at the next higher level of detail and the texture intensity of the cell at the next lower level of detail. Crow extends this idea and proposes a summed area table (1984). Crow states that:

Where the surface curves away from the viewer, texture may be compressed along only one dimension. Since table addressing must be based on the axis of maximum compression, mip mapped texture may appear fuzzier than would otherwise be necessary. A more accurate mapping may be obtained by allowing the corners of a rectangular region to lie between texture pixels. Therefore, the summed area at each corner of the rectangle must be calculated by interpolating from four values in the table.
Crow observes that the compression of the texture pattern may be great in only one direction. For example the x direction, and no compression in the y direction. However, the lower level of detail texture tables are derived assuming equal compression in both directions. Crow contends that by using a table look up with only one level of detail and then deriving the lower level of detail texture values using this table, clearer texture patterns are produced and blending is eliminated. However, the summed area table technique costs more in terms of calculations and memory than mip mapping (Crow, 1983).

General Electric also uses the mip map concept and texture blending to solve the level of detail problem (Chandler, 1984). When a polygon is between two levels of detail, the texture intensity from the table look up at the next higher and the next lower level of detail are accessed simultaneously. Then the final texture intensity value is found by interpolating between the two levels of detail. The intensity of the texture patterns is not decreased with distance.

The proper level of detail is dependent on the area on the surface projected by the pixel, and is a function of range, orientation and the sampling interval (Dungan,
Stenger, and Sutty, 1978). The area on the surface projected by a pixel is called a "footprint." The average footprint size, that is the width plus the length divided by two, is the proper level of detail. Dungan, Stenger, and Sutty used eight levels of detail (eight tables). Each level of detail table was assigned a footprint size interval. The General Electric company uses a very similar method. General Electric determines the change in texture value from one pixel to another to determine the proper level of detail. If the change in texture value given by Equation 19 is relatively large, the level of detail is reduced, and if the change in intensity is relatively small, a high level of detail is used.

Textured Anti-Aliasing Techniques

An image on the view screen is made up of pixels. The larger the number of pixels, the higher the resolution of the image. A problem that occurs from representing an image with pixels is called aliasing. Aliasing refers to the stairstep-like effect which occur from quantization effects due to representing a straight edge with pixels. For example, a diagonal line will not appear to be smooth but will appear to be a jagged stairstep-like line (Bloomenthal, 1981). Shading a
surface adds very little increase in scene detail. The rate of change of the shading intensity is very gradual across the polygon. Because there is very little increase in detail, and the shading intensity varies gradually from pixel to pixel, aliasing is not a problem. However, for textured polygons aliasing becomes an immediate problem (Crow, 1984). Anti-aliasing techniques must be used for textured surfaces because texturing can add almost unlimited detail to a polygon surface. For shading, the contrast in color between two pixels is very subtle and gradual. But for texturing, the colors between two pixels can change from black to white.

General Electric uses a bilinear interpolation approach as an anti-aliasing method (Bunker, 1984). Using Equation 19, two texture values are calculated from two perpendicular texture gradients. The two texture values are concatenated together to address four adjacent cells. Also, the four adjacent cells of the next lower level of detail are accessed. Each cell of the higher level of detail is blended with its corresponding cell at the next lower level of detail to produce four texture intensities. The final texture intensity of the pixel is a function of the position of
the pixel on the four cells. For example, if the pixel lies half way between cells 1, 2, 3, and 4, the final texture intensity of the pixel will be one fourth the intensity of cell 1 plus one fourth the intensity of cell 2 plus one fourth the intensity of cell 3 plus one fourth the intensity of cell 4. The position of the pixel will determine which weight to give the intensities from each of the cells when summing up the intensities to find the texture intensity of the pixel.

Dungan, Stenger, and Sutty use a method to anti-alias texture patterns that involves sampling the texture pattern at four different points on a pixel. A pyramid shaped filter is then applied to the four different points to find the average texture intensity of the pixel. Dungan, Stenger, and Sutty claim that the pyramid shaped filter reduces the effects of aliasing as well as the effects of moire patterns (1978).

Filter techniques can be used to reduce the effects of aliasing. A particular filter will average the color of neighboring pixels with the color of the pixel being considered. For example, a pixel P may have a neighboring pixel to the right, left, top and bottom. A particular filter may average the color of the right, left, top, and bottom pixel and pixel P. The final
color value of pixel P is the average of the colors of pixel P and the neighboring pixels. The shape of the filter, for example a pyramid, describes the weight given to the color of the neighboring pixels. Consider the case of the pyramid shaped filter. One can imagine a pyramid placed over a number of pixels. The length of the base of the pyramid will determine how many pixels the filter will use in the averaging process. The height of the pyramid directly over the pixel will determine the weight of the pixel. The base of the pyramid may cover any number of pixels, and different shapes such as cones, cylinders or rectangles may be used (Bunker, 1984).
CHAPTER III

CONCLUSION

In this paper the Phong shading model, the Gouraud shading model, the polyhedron shading model, and true perspective shading were discussed. The accuracy of the shading model is a function of cost in terms of computations. Polyhedron shading is the least costly but the least accurate and is useful for illuminating flat surfaces but does not work well when shading curved surfaces because it leaves the surface looking faceted.

The Gouraud shading model used by Evans and Sutherland, Singer Link, and early General Electric systems, produced good shading effects at a reasonable cost. The reduced version of Gouraud shading simplifies the interpolation section of the Gouraud shading model but restricts all polygons to triangles.

True perspective shading produces more accurate results than the Gouraud shading model because the shading intensity values are correctly projected from three dimensions to two dimensions. True perspective shading is more computationally expensive than the
previous shading models mentioned and restricts polygons
to triangles. However, the calculations for true
perspective shading and texture are very similar and,
therefore, much of the same hardware used for true
perspective shading can be used for texture.

The Phong model produces very realistic effects
because averaged vertex normals are interpolated as
opposed to shading intensities, but requires more
calculations on a per pixel basis which is very costly.
For this reason Phong shading has not been used by any
major visual flight simulator companies to this date.
Shading effects due to specular reflections, shadows, or
transparency are also not used because of the
computational expense.

Texturing requires that a texture pattern be mapped
onto a polygon. The mapping function must take into
account the position and orientation of the polygon.
The position and orientation of the polygon can be
expressed in terms of compression and convergence. The
texture pattern itself must be generated by a texture
mapping function. The table look up approach seems to
be the best suited to flight simulator applications
because a wide variety of texture patterns can be easily
calculated from a digitized photograph. The level of
detail of the texture patterns can be implemented by using the mip map technique and texture blending. Finally, anti-aliasing techniques must be used because of the amount of detail that texture produces. Figures 8 and 9 show textured images generated by a General Electric simulator using actual digitized data from photographs. The sky, ground, mountains, trees, and all the landscape have digitized texture patterns on them. The scenes were generated using less than 100 polygons even though there is almost unlimited detail. Notice how the abrupt changes in texture intensities are smoothed, thus avoiding aliasing effects. The trees in the scene all use the same texture pattern. Notice how the trees in the foreground are more detailed than the ones in the background. This is attributed to level of detail.

In conclusion, this report surveys several candidate shading and texturing techniques for visual flight simulators. Each candidate is described, and, strengths and weaknesses are explained. The results of this report show that the Gouraud shading model and the table look up texture technique gives the best cost performance trade-off, and as a result is widely used in industry.
Figure 8. A Textured Scene From a General Electric Simulator.
Figure 9. A Textured Scene From a General Electric Simulator.
REFERENCES


