Faster Phong Shading via Angular Interpolation

A.A.M. Kuijk and E.H. Blake*

Abstract

One of the most successful algorithms that brought realism to the world of 3D image generation is Phong shading. It is an algorithm for smooth shading meshes of planar polygons used to represent curved surfaces. The level of realism and depth perception that can be obtained by Phong shading is attractive for 3D CAD applications and related areas. However, per pixel computation costs which were too high and/or artifacts, introduced by some of the more efficient evaluation methods and apparent only when displaying moving objects, are major factors that blocked the common usage of Phong shading in highly interactive applications.

In this paper we present angular interpolation for Phong shading planar polygons. Angular interpolation was a method especially designed to meet requirements as imposed by special purpose hardware we developed[1], but turned out to be generally applicable. The angular interpolation method appears to be very efficient and reduces artifacts when displaying moving objects. Ideally a shading algorithm imposes no need for subdivision of patches as presented by the solid modelling system. Shading calculation via angular interpolation yields such an ideal algorithm. We will describe two alternative evaluation methods that trade off evaluation cost against level of accuracy. They both can handle light source and view point at arbitrary distances, but Mer in level of accuracy. As a consequence these alternative evaluation methods do impose restrictions on the topology of patches and light sources. However, generally, the limitations imposed by these alternative shading methods are much more liberal than the limitations on patch size imposed by the geometry.

The most economic evaluation method we present can incrementally compute the colour intensity along a scanline by two additions per pixel. The methods presented are generally applicable and can easily be implemented in hardware.

Key Words & Phrases: image synthesis, shading, angular interpolation, spherical geometry, quadratic approximation, quaternions.

*Centre for Mathematics and Computer Science (CWI)
Department of Interactive Systems
Kruislaan 413
1098 SJ Amsterdam
The Netherlauds
Email: fons@cwi.nl edwin@cwi.nl

North-Holland
Computer Graphics Forum 8 (1989) 315-324

Introduction

Phong shading is one of the most successful algorithms for obtaining a high degree of realism in computer generated images. This shading model is often used to shade planar polygonal approximated surfaces smoothly. It not only has ambient and Muse intensity components but incorporates a specular reflection component that produces a highlight as caused by reflection of shiny surfaces[2,3]. The surface reflectance dependent approximation of this specular component as proposed by Phong is based on empirical observation. The Phong shading model implies that at every pixel the diffuse component cos a and the reflection component $\cos^n\beta$ has to be evaluated. In general both a and β vary across the surface to be shaded.

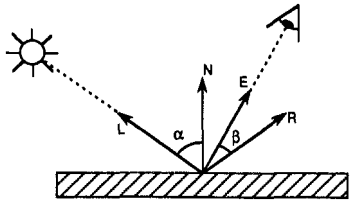

Figure 1. The Phong shading model comprises a diffuse component $\cos\alpha$ and a specular component $\cos^n\beta$. α is the angle between the surface normal $\vec{N}$ and the direction of the light source L, β is the angle between the direction of reflection R and the direction of the viewpoint $\vec{E}$ and n is a coefficient depending on the reflectivity of the surface.

Phong shading would be very attractive for interactive 3D CAD applications. The level of realism that can be obtained with it will give a better depth perception and may give a better idea of the final result of the design during the design process. In spite of this, common introduction of Phong shading in highly interactive CAD applications for smooth shading of planar polygon meshes did not occur, mainly because of the computation intensiveness of the method. Another deficiency are artifacts, introduced by some of the more efficient evaluation methods[4]. These artifacts may be almost invisible for static images, but become disturbingly apparent when displaying moving objects.

This may be reduced by reducing the polygon size, but this strategy would increase the computational demands and data transportation. As a result, the Phong shading model has so far been a widely accepted shading model for post design visualisation; the interactive design session -supported by a less demanding shading model- is followed by a request to show "what it really looks like".

In order to make Phong shading applicable for highly interactive applications, the basic problems that have to be solved are: "how do the angles α and β vary across the polygon?" and "given this variation how to evaluate the intensity components as economically as possible?"

In this paper we will show an efficient and high quality method for Phong shading planar polygonal meshes. In the first section, we will focus on answering the question raised above: "how do the angles α and β vary across the polygon?" We will discuss equi-angular interpolation for which no normalisation per pixel is needed as opposed to the traditional vector interpolation. Two alternative interpolation methods will be presented, both capable of handling light source and/or viewpoint at finite or infinite distance. The result of these methods will be an expression relating the cosines of the Phong model to one or two (depending on the method) angles incremented linearly along a scanline.

In the second section, the question: "how to evaluate the diffuse and specular components as economically as possible?" will be answered.

1. Interpolation Across Polygons

For curve approximation by planar polygon meshes, information of the curvature the planar polygon has to a proximate is represented by a surface normal vector specified at each vertex of the polygon[5], as shown in Figure 2. From this, the normal along the edges and at each point inside the polygon has to be interpolated. In general, not only the normal $\vec{N}$, but also the direction of the light source $\vec{L}$, the direction of the eye $\vec{E}$ and the direction of reflected light $\vec{R}$ is dependent on the position. Since $\vec{N}$ is obtained by interpolation, these vectors usually also are found by interpolation from the values calculated at each vertex, rather than by exact calculation at each pixel.

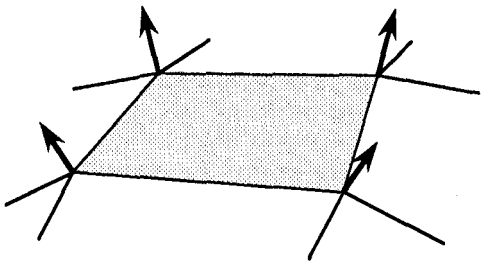

Figure 2. Approximation of curved surfaces by planar polygon meshes. For the purpose of shading calculations, the surface normal is specified at each vertex.

1.1. Vecctor Interpolation

The traditional vector interpolation method is based on the assumption that along a path from A to B a vector $\vec{N}_t$ at $P = A*(1-t) + B*t$ can be calculated by $\vec{N}_t = \vec{N}_A*(1-t) + \vec{N}_B*t$ with $0 \leq t \leq 1$ and $\vec{N}_A$ and $\vec{N}_B$ being the normals at A and B respectively. This interpolation is performed along the edges first. Using the thus obtained $\vec{N}_P$ and $\vec{N}_Q$ the normal along a scanline can be similarly interpolated, as shown in Figure 3.

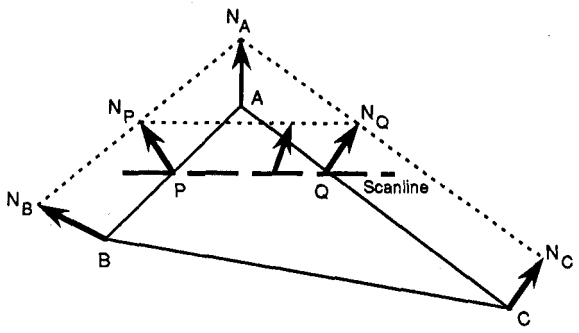

Figure 3. Interpolation of a vector across the polygon in two steps: first the vector is interpolated along the edges AB and AC, next the interpolated vectors $\vec{N}_P$ and $\vec{N}_Q$ thus obtained are used for interpolation along the scanline through PQ.

To have an efficient method, it has to be possible to relate the increments directly to the evaluation of the intensity components $\cos\alpha$ and $\cos^n\beta$. This is why vector interpolation became so widespread, the spatial increments dx, dy and dz of the vectors along the scanline are closely related to the vector coordinates needed for evaluation of the scalar product, which makes it possible to incrementally calculate the intensity components.

Figure4 however, shows that the resulting interpolated vector has a varying length and varying angular increment. This is a result of making ual steps along a chord joining the two vectors $\vec{N}_A$ and Due to the varying length, renormalisation of the interpolated vector has to be done. The varying angular increment

$\dagger\cos\alpha = (\vec{N}\cdot\vec{L})$ and $\cos^n\beta = (\vec{R}\cdot\vec{E})^n$, provided the vectors are normalised.

introduces an orientation dependency of the highlight. This results in a jitter of the highlight when rotating, for example, a cylindrical object around its main axes. It also causes Mach band effects because the intensity variation is not smooth. Generally these deficiencies are avoided by putting restrictions on the maximum angular variation of the vectors across the polygon; in other words by reducing the polygon size. It may be clear that this restriction is not without cost.

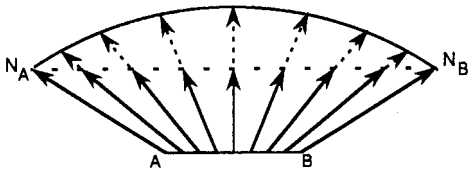

Figure 4. As a result of vector interpolation, successive vectors vary both in length and in angular increments. Renormalisation is needed when the interpolated vector is used in a scalar product.

A common assumption of most of the more efficient approaches known so far is that either the light source or the viewpoint is taken to be at infinity (sometimes even both). This means that $\vec{L}$ and/or $\vec{E}$ is taken to be a constant. This may simplify the calculations involved, but reduces the realism of the picture (e.g. if light source and eye are at infinity, planar surfaces are shaded with a constant intensity).

Bishop and Weimar6 published a method for Phong shading which requires a quadratic polynomial for calculating $(\vec{N}\cdot\vec{H})$ †, provided the curvature is less than 60 degrees. The $\vec{L}$ is taken to be a constant, but $\vec{E}$ is allowed to vary. Exponentiation of $(\vec{N}\cdot\vec{H})$ is done via table lookup.

An approach which uses bicubic approximations for the scalar products was published by Shantz and Sheue-Ling Lien8. They assumed L fixed and used lookup tables for the exponentiation.

A method also assuming $\vec{L}$ fixed was published by Deering[9]. A highly pipelined architecture interpolates and normalises $\vec{N}$ and $\vec{E}$ lookup tables are used for square root and exponentiation functions. Of particular interest in this publication was that the approach Bishop proposed was rejected because the break-even point of the method at 10 pixels was considered too high!

1.2. Angular Interpolation

The basic idea of angular interpolation is that the angular rotation of a directional vector ($\vec{N}$, $\vec{L}$ or $\vec{H}$) is linearly related with the position along a straight line across the polygon (see Figure 5). Vectors interpolated according to this assumption have a constant length and are all in one plane; the plane spanned by the start and end vector. An elegant way of calculating these angularly incremented vectors using quaternions is described elsewhere10,11. Analogous to the vector interpolation method, interpolation will be done in the two steps shown in Figure 3; first the vector is interpolated along the edges of the polygon, next the resulting vectors are used for interpolation along the scanline.

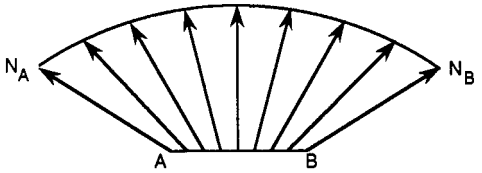

Figure 5. Angular interpolation. Successive vectors are found by qui-angular rotation using quaternions. As a consequence their length remains constant.

When observing the variation of one of the vector couples along a path across a polygon you can imagine a "dance" these two vectors perform. Since both vectors vary independently, the result is that they may circle around each other, get closer or move away. This "dance" along a straight path is completely determined by two linearly varying angles, one for each vector.

We have to find a relation between these angles incremented along the scanline and the evaluation of the intensity component $\cos\alpha$ ‡. We will show how to find this relation based on the following example of interpolation across a triangle.

The normal vector and the direction of the light source at the vertices of a triangle are shown in Figure 6. For each of the two vectors there is a mapping of the polygon on the unisphere indicating the range of that vector across the polygon. If for instance the light source is at infinity the spherical triangle $L_A L_B L_C$ is reduced to a point.

† $\vec{H}$ is the vector in the direction of the highlight, that is the vector halfway between $\vec{L}$ and $\vec{E}$. This vector is often introduced to replace $(\vec{E}\cdot\vec{R})$ by $(\vec{N}\cdot\vec{H})^n$. Note however, that the an e between $\vec{N}$ and $\vec{H}$ is about half the angle between $\vec{R}$ and $\vec{E}$, a difference discussed in Hall7 that can more or less be corrected by adjusting n.

‡In the following we will discuss the vector couple $\vec{N}$ and $\vec{L}$ needed for the diffuse term, only; interpolation of the vector couple that produces the specular term, i.e. $\vec{N}$ and $\vec{H}$, is identical.

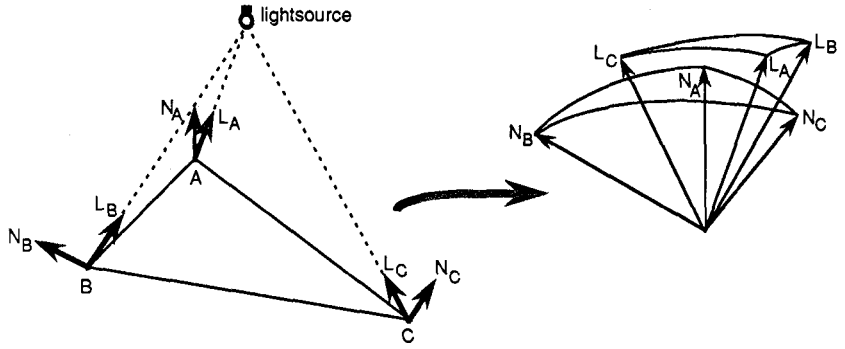

Figure 6. The range of the directional vectors $\vec{N}$ and $\vec{L}$ across the triangle A B C is indicated by two spherical triangles $N_A N_B N_C$ and $L_A L_B L_C$ on the unisphere.

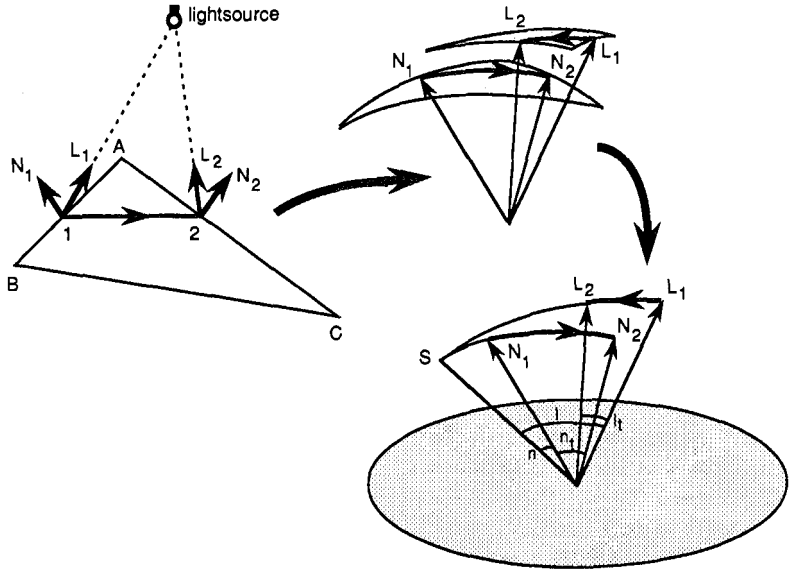

Figure 7. Interpolating $\vec{N}$ and $\vec{L}$ along a scanline is related to two paths along two great-circles. The intersection point S of these great-circles forms one point of a spherical triangle S N L. This triangle relates the two interpolated angles n_t and I, with the intensity component cos a along the scanline.

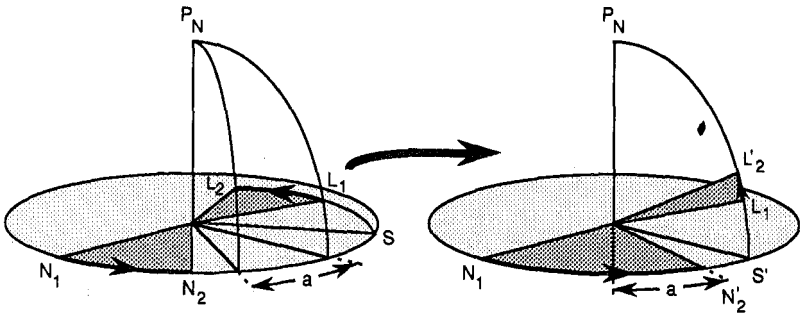

Figure 8. Decomposition of the rotation of one of the two independent rotating vectors results in two perpendicular rotated vectors.

A scanline across the triangle is mapped on two circular paths, indicating the variation of the vectors $\vec{N}$ and L along this scanline (see Figure 7). These paths, from $\vec{N}_1$ to $\vec{N}_2$ and from L_1 to L_2 are each part of a great-circle. These two great-circles intersect at S. Let γ be the angle between the two great-circles, n be the angle between S and $\vec{N}_1$, I be the angle between S and $\vec{L}_1$. Then having t linearly varying along the scanline from t = 0 at startpoint 1 and t = 1 at endpoint 2, we define n_t to be t*(the angle between $\vec{N}_1$ and N_2) and l_t to be t*(the angle between L_1 and $\vec{L}_2$). $\vec{N}_t$ and $\vec{L}_t$ are the vectors $\vec{N}$ and L, interpolated along the scanline.

With this we have a spherical triangle $S N_t L_t$, dependent on t. For this triangle a standard formula, given by spherical trigonometry, leads us to the following relation between a, (i.e. the angle between $\vec{N}_t$ and $\vec{L}_t$) and the linearly incremented angles n_t and l_t:

$$\cos\alpha_t = \cos(n+n_t)\cos(l+l_t) + \sin(n+n_t)\sin(l+l_t)\cos\gamma \quad (1)$$

Note that n, l and γ are constant along the scanline. Here we have an expression that directly produces the diffuse intensity component. The specular component can be found similarly, but needs raising to the power of the specular exponent as well. This expression already can be calculated more efficiently than calculation of the intensity by vector interpolation, which inherently involves renormalisation. However, for our particular application it was still too complicated for efficient evaluation. We developed two alternative approximation methods for this which both result in a simpler expression, and produce good results under mild restrictions. In one method the rotation of one vector is decomposed in two perpendicular components. The other method combines the two varying vectors in one.

Decomposition Method

Expression (1) will reduce to a product of two cosines, when the two great-circles are perpendicular; in that case cosy is zero. The idea is to decompose the rotation of one vector into two rotation components, one parallel and one perpendicular to the rotation of the other vector.

Let a be the angle between the two great-circles through L_1 and L_2 both perpendicular to the plane in which $\vec{N}$ rotates as shown in Figure 8. Rotating $\vec{N}_2$ as well as $\vec{L}_2$ with angle a around the axis through P_N (the pole of the plane through $\vec{N}_1$ and $\vec{N}_2$), will produce the two vectors $\vec{N}'_2$ and $\vec{L}'_2$. Interpolation of $\vec{N}$ can now be done from $\vec{N}_1$ to $\vec{N}'_2$ and interpolation of $\vec{L}$ from L_1 to L'_2 as indicated in the figure. In doing so, we added the rotation component of $\vec{L}$ around the axis P_N to the rotation of $\vec{N}$ around that axis. As a result, for $\vec{L}$ only a rotation component perpendicular to the plane through $\vec{N}_1$ and $\vec{N}_2$ remains; and $\vec{L}'_2$ are both on the same great-circle throu P_N that intersects the great-circle through $\vec{N}_1$ and $\vec{N}_2$ at S'.

Let n_1 be the angle between S' and $\vec{N}_1$, n_2 the angle between S' and $\vec{N}'_2$ and let l_1 be the angle between S' and $\vec{L}_1$ and finally l_2 the angle between S' and $\vec{L}'_2$. Then for the right angled spherical triangle $N_t S' L_t$ we can write the following equation:

$$\cos\alpha_t = \cos(n_1+t(n_2-n_1))\cos(l_1+t(l_2-l_1)) \quad (2)$$

The spherical triangle shown in Figure 9 will clarify the implicit assumption we made using this method. For this spherical triangle the following equations hold:

$$\cos l_2 = \frac{\cos(l_t+l)}{\cos(a+b)} \quad (3.a)$$

$$\cos\gamma = \frac{\tan(a+b)}{\tan(l_t+l)} \quad (3.b)$$

$$\sin\gamma = \frac{\sin l_2}{\sin(l_t+l)} \quad (3.c)$$

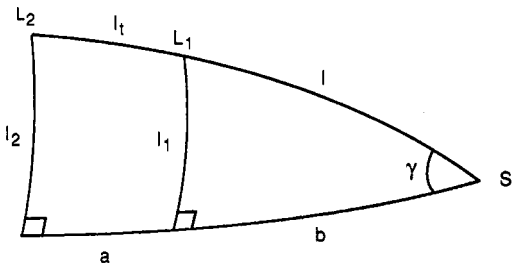

Figure 9. From this spherical triangle it can be seen that in general, linearly incrementing both a and l_2 is not equivalent to linearly incrementing I_t.

Using (3.a) and (3.b), we can prove that for t = 0 and t = 1, (2) will produce the same result as (1). Of more interest however, are the relations between a and l_t and between l_2 and l_t as given by (3.b) and (3.c). These relations show that in general linearly incrementing the angles $n_2 - n_1$ (with a hidden in it) and $l_2 - l_1$ is not equivalent with linearly incrementing n_t and l_t. This however is an acceptable first order approximation. Only extreme situations such as an extremely nearby light source may affect the result. We could not produce a visible difference in a realistic situation up to now.

Reducing to One Vector

Instead of inte lating the two independently varying vectors $\vec{N}$ and $\vec{L}$ we would like to interpolate only one vector, without loosing. the generality of the method, that is, without assuming one of the vectors fixed. Realising that only the relative position of both vectors $\vec{N}$ and $\vec{L}$ is of interest, not their absolute position, we define a vector $\vec{V}$ at each vertex of the polygon that is found by rotating $\vec{L}$ around the same axis and with the same angle as needed to rotate $\vec{N}$ to be aligned with a fixed vector $\vec{O}$(e.g. $\vec{o} = \begin{bmatrix}0\\0\\1\end{bmatrix}$).

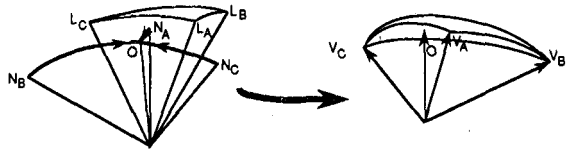

Figure 10. Interpolation of two independently varying vectors $\vec{N}$ and $\vec{L}$ can be replaced by interpolation of one vector V. For each vertex, $\vec{V}$ can be found by applying the same rotation on the vector L as needed to align the corresponding $\vec{N}$ with the reference vector $\vec{O}$.

Using this method, we have specified a vector $\vec{V}$ which is the only vector that remains to be interpolated across the polygon and the quest for the angle between $\vec{N}$ and $\vec{L}$ can be replaced by the search for the angle between $\vec{V}$ and $\vec{O}$.

We do this by constructing a right angled spherical triangle $V_t O S$. This spherical triangle (illustrated in Figure 11) has one vertex at $\vec{O}$ and one vertex at $\vec{V}_t$ ($\vec{V}_1 \leq \vec{V}_t \leq \vec{V}_2$). The third vertex $\vec{S}$ lies on the great-circle through $\vec{V}_1$ and $\vec{V}_1$ such that $V_t S$ is perpendicular to SO †. $\vec{S}$ defines two constant angles; s the angle between $\vec{S}$ and $\vec{O}$ and v, the angle between $\vec{S}$ and $\vec{V}_1$

Let v_t be the angle between $\vec{V}_1$ and $\vec{V}_t$, linearly incremented alon the scanline. Then, α_t, the angle between $\vec{V}_t$ and $\vec{O}$ can be expressed in terms of the angle v, with the formula:

$$\cos \alpha_t = \cos(v + v_t) \cos s \qquad (4)$$

†We do not have to worry about the two degenerate situations; 1) $\vec{O}$ on the great-circle through $\vec{V}_1$ and $\vec{V}_1$ and 2) $\vec{O}$ perpendicular to the plane through the great-circle because these are covered by the resulting expression equally well.

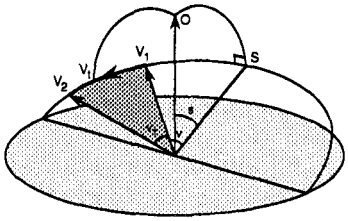

Figure 11. A right angled spherical triangle S O V relates the linear1 incremented angle v_t with the angle between $\vec{V}_t$ and $\vec{O}$.

With this expression we succeeded in relating the intensity component cos α to one angle v_t, linearly v along a scanline. By introducing the vector $\vec{V}$ we simplified the interpolation and at the same time eliminated the difference between directional and positional light sources[5].

How does this method relate to the previous one? In the previous method, only the parallel component of the rotation of one vector was added to the rotation of the other vector, whereas this method adds the full rotation of one vector to the rotation of the other. As a consequence this methods imposes more restrictions on the maximum acceptable variation of the vectors across the polygon. In practice however, these limitations are easily met, Differences between the methods became visible when vector rotations on the polygon were about 90°. This is much more than what is to be expected with curved surface approximation by planar polygon meshes.

Since especially for highly reflective surfaces (i.e. a high reflectivity exponent for the specular term) the specular highlight will have a limited range, checking the range of $\vec{V}$ with respect to $\vec{O}$, can give at an early stage insight whether or not the specular term will contribute at all. This check can be performed at polygon level as well as at scanline level.

2. Evaluation of the Intensity Components

In the previous sections we presented angular interpolation of vector couples which yielded the linear expressions (1), (2) and (4) for the cosine of the angle between them along a scanline. Choosing one of these expressions, we can evaluate the diffuse component. For the specular intensity component it takes raising to the power of the specular exponent as well. Evaluation of the cosine as well as exponentiation can be done using standard function libraries or lookup tables. The latter may be the fastest solution for software implemented evaluation, but neither of these are attractive for highly parallel VLSI implementation. Lookup tables in particular would require large on-chip storage (> 8K bytes for exponentiation[6]).

We had to find a more acceptable approximation to evaluate the general term $\cos^n\theta$ which is present in both (2) and (4). The approximation should meet the following criteria:

- easy to evaluate incrementally by forward differencing
- high quality that is: no discontinuity in magnitude or slope of intensity.

An approximation by means of a second order Taylor series would be easy to evaluate. This may produce a quadratic function that correctly describes the behaviour near the centre, but the grave discontinuity in slope of the intensity where the quadratic function gets to zero will cause an unacceptable Mach band effect. Given this, we tried to find a better evaluation method.

Evaluation of $\cos^n\theta$ by Quadratic Curve Approximation

Due to hardware limitations imposed by the technology[1], we could not allow for a higher than second order polynomial. With this in mind and looking at the shape of $\cos^n\theta$ we found that a combination of three successive quadratic curves would closely fit that shape (see Figure 12). The condition that no discontinuity in magnitude or slope of intensity is allowed is met by the following function:

$$f(\theta) = \begin{cases} 0 & \text{if } \theta \leqslant -b \text{ or } b \leqslant \theta \\ \dfrac{(\theta + b)^2}{b(b - a)} & \text{if } -b < \theta < -a \\ 1 - \dfrac{\theta^2}{ab} & \text{if } -a \leqslant \theta \leqslant a \\ \dfrac{(\theta - b)^2}{b(b - a)} & \text{if } a < \theta < b \end{cases}$$

With the two parameters a and b, this function has sufficient freedom to produce a best fit for the function $\cos^n\theta$. As can be seen in Figure 12, $-$a and a are the points where the quadratic function changes its second derivative and $-$band b limit the region for which the function is nonzero. This latter quantity is very useful; it is nice to know the range where $f(\theta) \neq 0$.

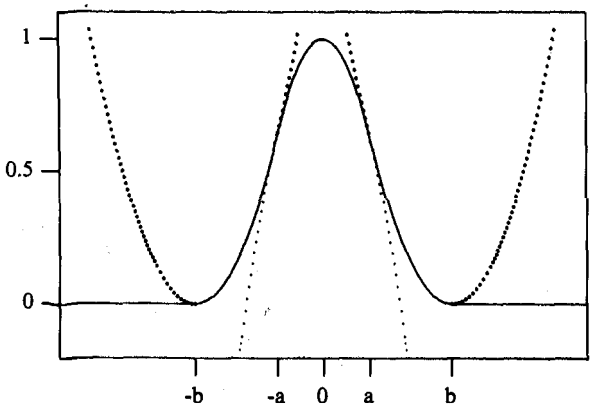

Figure 12. Three successive quadratic curves fitted on $\cos^3\theta$.

By least square fitting, for each n $(1 \leqslant n \leqslant 125)$ a pair (a, b) was found. Because there are only a limited number of pairs, these values can easily be put in a small lookup table. However, looking at the values of a and b as a function of n it was clear that their relation could be described functionally as well. We found the following relations:

$$a = \frac{n + 65.0}{5.0n + 31.7}$$

$$b = \frac{n + 5.6}{n(0.09n + 5.2)}$$

Note that the values a and b are dependent on n only, so they can be considered as an attribute of the polygon, replacing the specular exponent n.

Results of this evaluation method for several values of n can be seen in Figure 13. Given the fact that the specular component as proposed by Phong is based on empirical observation, it is clear that we can be satisfied with the result. Even more so because evaluating the diffuse component with this method (n = 1) has the pleasant side effect that it removes Mach banding. This Mach banding is caused by the sharp discontinuity in slope when the cosine reaches zero. At that point the slope changes from its maximum to zero[7]. The method proposed here does not have such a discontinuity for n = 1, as can be seen in Figure 13. The smooth transition of this evaluation method agrees with reality where diffraction and a finite sized light source cause a smooth transition as well.

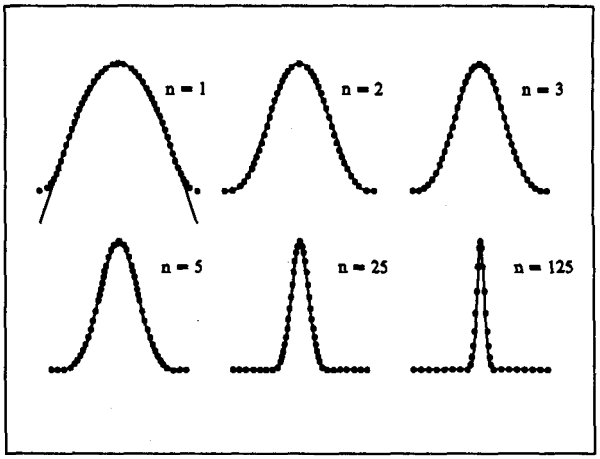

Figure 13. Results of the quadratic approximation of $\cos^n\theta$ for several values of n. The solid lines represent $\cos^n\theta$ whereas the boxes show the approximated values.

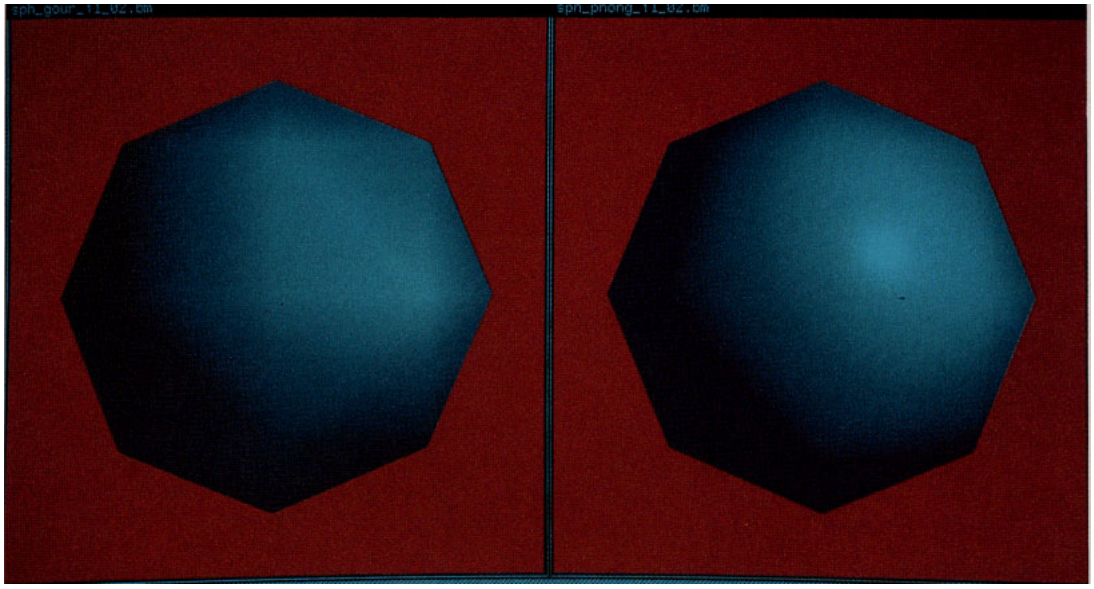

Figure 16.

On the left a Gouraud shaded and on the right a Phong shaded image of a "sphere" approximated by 32 triangle patches. The light source is at a distance of four times the radius of the sphere. This and all following images have a parallel projection.
Obvious differences between these two images is 1) the amount of Mach banding, due to which the triangle patches are clearly visible for the Gouraud shaded image. This is not the case in the Phong shaded image. 2) the specular highlight is absent in the Gouroud shaded image.
These differences become less when the patch size is reduced, but at the cost of having more patches.

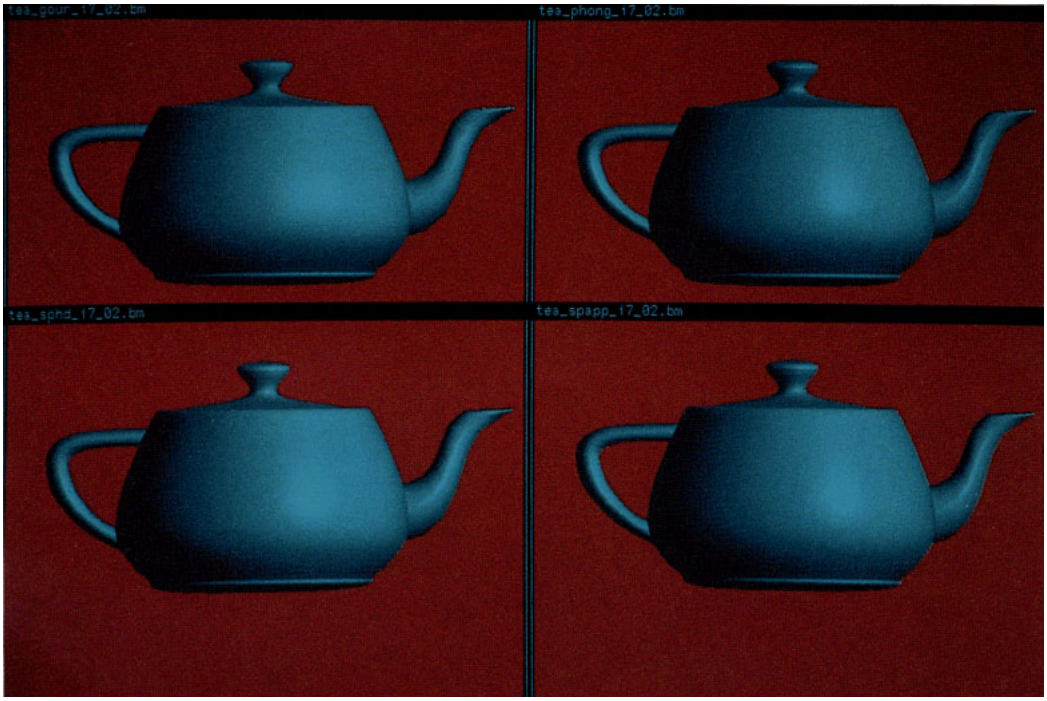

Figure 16.

Diffuse shaded images of the same teapot lit by two light sources. On top we see the result of calculation of the intensity by a cosine function. It shows Mach banding due to a sharp discontinuity in slope of intensity where the cosine reaches zero. For the bottom image the Muse term is evaluated using the quadratic approximation method presented in this paper, resulting in a smooth transition to zero. This agrees with reality where natural effects cause a smooth transition as well.

<Figure 15.

Four images of the Utah teapot (3136 triangular facets). The top left is a Gouraud shaded image, the top right a Phong shaded image generated by the standard vector interpolation method. For each pixel this method involves. interpolation and normalisation of four vectors (two for the diffuse and two for the specular contribution) and evaluation of two scalar products. The diference in quality between these two images is obvious.
At the bottom, the leftmost image is generated by angular interpolation of the vectors, which involves interpolation of four angles (two for the Muse and two for the specular term) and evaluation of a trigonometric function per pixel. Finally, the rightmost image is generated using the single vector approximation method as described in this paper. This involves interpolation of two angles (one for the Muse and one for the specular term) and evaluation of a very simple trigonometric function. Although the computational costs of the latter two methods is less, there is no difference in image quality when compared with the image generated by the standard vector interpolation method.

Evaluation of (4) can be done directly with this method. Per scanline and per light source for the diffuse and the specular term the constant angles s and v have to be calculated. For each of these, along the scanline the quadratic function $f(\theta)$ can be evaluated using forward differencing. This costs two additions per pixel. At the points where the transition from one quadratic section to the next has to be made, the second order derivative has to be changed; the function value and the first order derivative are both continuous at those points.

Instead of evaluating the quadratic functions per light source and for the diffuse and specular components individually it can be noted that adding these quadratic functions before performing the forward differencing along the scanline, would still result in a quadratic function, although composed of more sections. It still holds that the function value and the first order derivative are both continuous, so that only the second order derivative has to be changed at transition from one quadratic section to the next. This can be cheaper than evaluation of the diffuse and specular terms per light source individually.

Evaluation of (2) needs evaluation of two individual cosines which have to be multiplied. Dependent on the situation, in particular on the hardware available, multiplication can be done per pixel or per scanline, the latter resulting in a fourth order polynomial.

3. Conclusions & Future Work

The angular interpolation presented has shown to result in a very efficient high quality evaluation of the Phong shading algorithm. Although the exact method can already be considered to be efficient as compared to vertex interpolation methods, we presented two even more efficient evaluation methods that operate satisfactorily without putting Severe restrictions on viewpoint or light source distance. They differ in level of accuracy traded off against corresponding costs. The limitations on patch size and light source distance imposed by the more efficient evaluation methods due to the approximations made, generally are much more liberal than the limitations imposed by the geometry.

The quadratic evaluation presented can easily be implemented in hardware. Forward differencing would reduce the evaluation cost to two additions per pixel. Using this method for the diffuse term has the side effect of removing the Mach banding on the edge of the diffuse area as normally present when using a cosine or scalar product.

On future work: More use can be made of coherence between scanlines. This would produce a more efficient way of evaluating the angles which are constant along a scanline.

We have a satisfactory method to evaluate a cosine raised to a power, but this is a particular case. We are currently studying how to extend this to more general situations such as the product of a linear function and $\cos^n$ and the product $\cos^n\theta \cos^m\phi$. These generalisations are needed when implementing spot light sources or anti-aliasing.

4. Acknowledgements

We acknowledge the national foundation STW who is funding a project that as side effect faced us with this topic. We acknowledge Kapila Jayasinghe (University of Twente) for the fruitful discussions that led to the quadratic evaluation method. Furthermore, discussions with Ute Claussen (Universität Tubingen) made us more aware of the problems involved in angular interpolation.

References

1. J.A.K.S. Jayasinghe, A.A.M. Kuijk, and L. Spaanenburg, "A Display Controller for a Structured Frame Store System," in Advances in Graphics Hardware III, ed. A.A.M. Kuijk, Springer-Verlag (to appear in 1989).
2. B.T. Phong, "Illumination for Computer Generated Images," Communications of the ACM 18(6), pp. 311-317 (1975).
3. J. Foley and A. van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley, Reading, Mass. (1982).
4. U. Claussen, "On Reducing the Phong Shading Method," pp. 333-344 in Proceedings EUROGRAPHICS '89, Hamburg (Sept 1989).
5. A. v. Dam, "PHIGS+ Functional Description, Revision 3.0," ACM Computer Graphics 22(3) (July 1988).
6. G. Bishop and D.M. Weimer, "Fast Phong Shading," ACM Computer Graphics 20(4), pp. 103-106 (1986).
7. R. Hall, Illumination and Color in Computer Generated Imagery, Springer-Verlag (1988).
8. M. Shantz and Sheue-Ling Lien, "Shading Bicubic Patches,'' ACM Computer Graphics 21(4), pp. 189-196 (1987).
9. M. Deerin, S. Winner, B. Schediwy, C. Duffy, and N. Hunt, "The Triangle Processor and Normal Vector Shader," ACM Computer Graphics 22(4), pp. 21-30 (1988).
10. W.R. Hamilton, Elements of Quaternions, Chelsea, New York (1969).
11. E. Pervin and J.A. Webb, "Quaternions in Computer Vision and Robotics," CMU-CS82-150, Carnegie Mellon University (1982).