IMPROVED SHADING PERFORMANCE BY AVOIDING VECTOR NORMALIZATION

Anders Hast

Creative Media Lab
University of Gävle, Kungsbäcksvägen 47, S-801 76 Gävle, Sweden. aht@hig.se

Tony Barrera

Cycore AB
Dragarbrunnsgatan 35, P.O. Box 1401, S-751 44 Uppsala, Sweden

Ewert Bengtsson

Centre for Image Analysis
University of Uppsala, Lägerhyddsvägen 17. S-752 37, Uppsala, Sweden. ewert@cb.uu.se

ABSTRACT

Phongs illumination model requires unit length vectors. The surface normal has to be normalized due to the linear interpolation, and if we use single point light sources or a fixed view point, we have to normalize the vectors pointing to the light source and to the viewer. Unfortunately, normalization is a relatively costly operation. One of the main reasons for this is the square root involved. But when we calculate the reflection vector, we actually do not need a normalized normal. This fact can be used in order to get an approximation for the vector we want when we interpolate between normals. The result is faster Phong shading and faster lighting calculations when we are using a single point light source or having a viewer which is not placed at infinity.

Keywords: Shading, Illumination, Normalization

1 INTRODUCTION

Shading makes facetted objects appear smoothly curved. An illumination model is applied in the shading process in order to achieve this effect. When realism is important, we use a global illumination model like ray tracing or radiosity, which take interreflections between facets into account. But when speed is important, a local illumination model is used. We shall see how the latter could be optimized for different shading techniques. In Phongs illumination model [Phong75], the intensity at a point on the surface of an object can be modeled by summing intensities of the diffuse and specular components, respectively:

$$I = K_d(\mathbf{N} \cdot \mathbf{L}) + K_s(\mathbf{R} \cdot \mathbf{V})^n, \tag{1}$$

where K_d is a material constant for the diffuse property of the surface, K_s is a material constant for the specular property of the surface, $\mathbf{N}$ is the normal vector for the surface, $\mathbf{L}$ is a vector pointing in the direction to the light source, $\mathbf{R}$ is the vector in the direction of the perfect reflection from the light source, $\mathbf{V}$ is the vector pointing to the viewer, and finally n is the shininess value which affects the size of the highlighted area.

Similar models can be used, and some include ambient light and the intensity of the light source. Sometimes a distance term is used, since the intensity of light becomes smaller with increased distance. We shall look closer at how the diffuse term $I_d = \mathbf{N} \cdot \mathbf{L}$ and the specular term $I_s = \mathbf{R} \cdot \mathbf{V}$ could be efficiently computed. It should be pointed out that we then have to raise the specular term to the power n which is a costly operation. We also have to multiply the terms with their material constants, and finally add them together. In subsequent sections, when we will count operations in the inner shading loop, we will not take into account these operations since they have to be done, regardless of whatever technique we choose in order to compute I_d and I_s.

When we use this illumination model to shade pixels, there are basically two techniques we can use. They are known as Gouraud- [Goura71] and Phong shading [Phong75], named after their inventors. In Gouraud shading, the intensities at the vertices of the polygon are calculated first. Then bilinear interpolation is used to calculate the intensity of interior pixels. This method is fast, but has its limitations. One examples is when the intensity at the vertices happen to be the same. Then the intensity will be the same over the whole polygon. Phong shading overcomes this problem by interpolating the normals at the vertices. Hence, we will have a linearly interpolated normal at each pixel, which we can use to calculate the intensity by using the illumination model. The drawback is that this takes more time, and one reason for this is the normalization process and the square root involved. Another reason why we prefer Phong shading is that we get more accurate highlights on a surface. Gouraud shading tend to smear out the highlight over a polygon because the intensity is interpolated over the polygon. That is, if the highlight falls close enough to a vertex. But if it falls on to the interior of a polygon, it will not be visible at all, if the vertexes are too far away from that point. Another disadvantage with Gouraud shading is that it tend to produce more mach bands than Phong shading.

1.1 Normalization

Normalization is a 'necessary evil' in many situations. One example is Phong shading, where we have to normalize the linearly interpolated normal, otherwise it will be too short and that will affect the lighting calculations. The interpolation of $\mathbf{N} = (1-\alpha)\mathbf{N_1} + \alpha\mathbf{N_2}$ is often done as a recurrence sequence:

```
N=N_1
dN=(N_2-N_1)/n
for i=1:n
  N=N+dN
end
```

The drawback is that normalization is a costly operation, not at least depending on the square root operation needed, as seen in the following equation, where $\mathbf{N}$ is an unnormalized vector in 3D space.

$$\mathbf{N}' = \frac{\mathbf{N}}{\parallel \mathbf{N} \parallel} = \frac{\mathbf{N}}{\sqrt{\mathbf{N}\cdot\mathbf{N}}}. \qquad (2)$$

1.2 Fast Phong Shading

Duff [Duff79] showed that $\frac{\mathbf{N}\cdot\mathbf{L}}{\|\mathbf{N}\|}$, where $\mathbf{L}$ is normalized and $\mathbf{N}$ is not, could be evaluated for successive values of x with 3 additions, 1 division and one square root per pixel. We will show why this is so and later compare it to our new algorithm, which we will implement using this scheme. It should be pointed out that there exists a number of other techniques to do fast Phong shading, like the ones introduced by Bishop and Weimer [Bisho86] and Ouyang and Maynard [Ouyan96], but in this paper we shall limit our selves to compare it to Duff's method only.

As we saw earlier we can interpolate from normal $\mathbf{N_1}$ to $\mathbf{N_2}$ by using a recurrence. Let $\mathbf{N} = \mathbf{k}x + \mathbf{m}$ along the scan line from x_1 to x_2, where $n = x_2 - x_1$, $\mathbf{k} = (\mathbf{N_2} - \mathbf{N_1})/n$ and $\mathbf{m} = \mathbf{N_1}$. If we evaluate $\mathbf{N}\cdot\mathbf{L}$ instead, as Duff did, then we have:

$$\mathbf{N}\cdot\mathbf{L} = Ax + B = p, \qquad (3)$$

where $A = \mathbf{k}\cdot\mathbf{L}$ and $B = \mathbf{m}\cdot\mathbf{L}$. We must not forget the normalization. Let

$$\mathbf{N}\cdot\mathbf{N} = Cx^2 + Dx + E = q, \qquad (4)$$

where $C = \mathbf{k}^2$, $D = 2(\mathbf{k}\cdot\mathbf{m})$ and $E = \mathbf{m}^2$. Note that $E = 1$, since $\mathbf{m} = \mathbf{N_1}$ has unit length. Then we have:

$$\frac{\mathbf{N}\cdot\mathbf{L}}{\parallel \mathbf{N} \parallel} = \frac{Ax + B}{\sqrt{Cx^2 + Dx + E}} = \frac{p}{\sqrt{q}}. \qquad (5)$$

Now we can set up the following recurrence:

$$p_{i+1} = p_i + dp_i \qquad (6)$$
$$q_{i+1} = q_i + dq_i \qquad (7)$$
$$dq_{i+1} = dq_i + d^2q \qquad (8)$$

where $p_0 = B$, $dp = A$, $q_0 = E = 1$, $dq_0 = C + D$ and $d^2q = 2C$.

This recurrence is evaluated in the inner loop for the scan line, along with $I_d = p/\sqrt{q}$, which is the diffuse intensity for the pixel. For each new scan line we must recalculate A, B, C, D, p_0, q_0, dq_0 and d^2q.

As we can see, this recurrence could be evaluated for successive values of x along a scan line with 3 additions, 1 division and one square root per pixel.

2 THE REFLECTION VECTOR

In many papers and computer graphics text books the following equation is used for calculating the reflection vector $\mathbf{R}$:

$$\mathbf{R} = \mathbf{2N'}(\mathbf{L} \cdot \mathbf{N'}) - \mathbf{L}. \tag{9}$$

Note that $\mathbf{N'}$ must have unit length. This equation is often derived by using the equation for projecting a vector onto another, as in [Foley97] and [Hearn97]. But they use the form where the vector which is project onto is normalized. We shall remake their derivation by using the form where it is not normalized. Hence, we do not need to use a normalized $\mathbf{N}$. This fact is rarely mentioned in computer graphics text books, but it is nothing new. As an example it could be mentioned that this fact is used by Voorhies and Foran [Voorh94] in their environment mapping technique.

Proposition 2.1. *If we have a normal* $\mathbf{N}$ *of arbitrary length, and a light source in the direction of unit vector* $\mathbf{L}$*, then the unit vector* $\mathbf{R}$ *in the direction of a perfect reflection is*

$$\mathbf{R} = 2\mathrm{N}\frac{\mathbf{N} \cdot \mathbf{L}}{\mathbf{N} \cdot \mathbf{N}} - \mathbf{L}. \tag{10}$$

Proof. Let $\mathbf{P}$ be the projection of $\mathbf{L}$ onto $\mathbf{N}$. Then

$$\mathbf{P} = \frac{\mathbf{N} \cdot \mathbf{L}}{\| \mathbf{N} \|^2}\mathbf{N}. \tag{11}$$

Let $\mathbf{K}$ be the vector from $\mathbf{P}$ to $\mathbf{L}$, then

$$\mathbf{K} = \mathbf{L} - \mathbf{P}. \tag{12}$$

We know that

$$\mathbf{R} = \mathbf{P} - \mathbf{K}, \tag{13}$$

thus

$$\mathbf{R} = \mathbf{P} - (\mathbf{L} - \mathbf{P}) \Rightarrow \tag{14}$$
$$\mathbf{R} = 2\mathbf{P} - \mathbf{L} \Rightarrow \tag{15}$$
$$\mathbf{R} = 2\frac{\mathbf{N} \cdot \mathbf{L}}{\| \mathbf{N} \|^2}\mathbf{N} - \mathbf{L}. \tag{16}$$

Finally Eq.(2) gives:

$$\mathbf{R} = 2\mathrm{N}\frac{\mathbf{N} \cdot \mathbf{L}}{\mathbf{N} \cdot \mathbf{N}} - \mathbf{L}.$$

$\square$

3 HYBRID SHADING

Proposition (2.1) says that we can get $\mathbf{R}$ without normalizing the normal. This means that we can get a faster method for calculating highlights, that is, if we do not calculate the diffuse light by using these normals, since it would mean that we had to normalize them anyway. Instead we use Gouraud shading for the diffuse light only. Our new hybrid algorithm would thus use the $K_d(\mathbf{N} \cdot \mathbf{L})$ for the diffuse light at the vertexes which we then bilinearly interpolate over the polygon. Then we use $K_s(\mathbf{R} \cdot \mathbf{V})^{\mathbf{n}}$ to get the specular light, where $\mathbf{R}$ is obtained by linearly interpolating the normals, and using proposition (2.1). This implies that we have to compute all normals at the edges by interpolation, but we do not have to normalize them.

If we assume that $\mathbf{V} = [\mathbf{0}, \mathbf{0}, \mathbf{-1}]$ and remembering that $\mathbf{N}$ is interpolated and thus not normalized, then

$$\mathbf{R} \cdot \mathbf{V} = L_z - 2N_z\frac{\mathbf{N} \cdot \mathbf{L}}{\mathbf{N} \cdot \mathbf{N}}. \tag{17}$$

We can rewrite this equation by using Duff's scheme. Let $A_1 = \mathbf{k} \cdot \mathbf{L}$, $B_1 = \mathbf{m} \cdot \mathbf{L}$, $A_2 = 2k_z$, and $B_2 = 2m_z$, then

$$\mathbf{N} \cdot \mathbf{L} = A_1x + B_1, \tag{18}$$

and

$$2N_z = A_2x + B_2, \tag{19}$$

thus

$$\begin{aligned} &2N_z(\mathbf{N} \cdot \mathbf{L}) = \\ &(A_1x + B_1)(A_2x + B_2) = \\ &A_1A_2x^2 + (A_1B_2 + A_2B_1)x + B_1B_2. \end{aligned} \tag{20}$$

We can multiply L_Z with the denominator to get rid of the subtraction. Let the denominator be:

$$\mathbf{N} \cdot \mathbf{N} = Dx^2 + Ex + F = q. \tag{21}$$

Then, the numerator can be rewritten as:

$$Ax^2 + Bx + C, \tag{22}$$

where $A = L_zD - A_1A_2$, $B = L_zE - A_1B_2 - A_2B_1$ and $C = L_zF - B_1B_2$. Now we can set up the following recurrence:

$$p_{i+1} = p_i + dp_i \tag{23}$$
$$dp_{i+1} = dp_i + d^2p \tag{24}$$
$$q_{i+1} = q_i + dq_i \tag{25}$$
$$dq_{i+1} = dq_i + d^2q \tag{26}$$

where $p_0 = C$, $dp_0 = A + B$, $d^2p = 2A$, $q_0 = F = \mathbf{m} \cdot \mathbf{m}$, $dq_0 = D + E$ and $d^2q = 2D$. If we

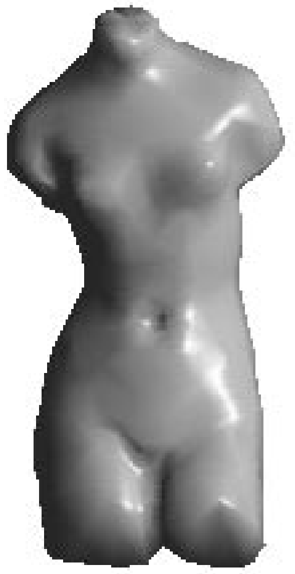

Figure 1: The Venus de Milo statue, hybrid shaded

have normalized normals on the edges we could set $F = 1$.

This recurrence is evaluated in the inner loop for the scan line, along with $I_s = p/q$, which is the specular intensity for the pixel. As we can see this recurrence could be evaluated for successive values of x along a scan line with 4 additions and 1 division per pixel for the specular light and 1 addition for linearly interpolating the intensity of the diffuse light.

Fig.1 shows the famous Venus de Milo statue which has been modeled with 1416 triangles. Here it is shaded using the hybrid shading technique.

4 REFLECTION SHADING

Here we will introduce another way to calculate $\mathbf{N} \cdot \mathbf{L}$ without having to Normalize $\mathbf{N}$. The idea is that if we could find a vector $\mathbf{H}$ which is exactly halfway between our interpolated, and unnormalized normal $\mathbf{N}$ and a vector $\mathbf{n}$ which is normalized, then we could use proposition (2.1) to wrap $\mathbf{n}$ around $\mathbf{H}$ to get $\mathbf{N}'$ which will be in the same directions as $\mathbf{N}$ but will be normalized. The nice thing is that we do not have to use the square root in this method. Even though the equation will be longer, we can reduce it, as we shall see later. The problem is that we will not find $\mathbf{H}$ without using a square root. But we could obtain approximations of $\mathbf{H}$ in several ways. One very easy and many times sufficiently good way is to let $\mathbf{n}$ be the normal of the polygon, and to let $\mathbf{H} = \mathbf{n} + \mathbf{N}$. Remember that we do not have to normalize $\mathbf{H}$:

$$\mathbf{N}' = 2\mathbf{H}\frac{\mathbf{H} \cdot \mathbf{n}}{\mathbf{H} \cdot \mathbf{H}} - \mathbf{n}. \tag{27}$$

One nice thing about computing $\mathbf{H}$ this way is that we can interpolate $\mathbf{H}$ instead of $\mathbf{N}$. We can easily prove that

$$\begin{aligned}&(1-\alpha)\mathbf{N_1} + \alpha\mathbf{N_2} + \mathbf{n} = \\ &(1-\alpha)(\mathbf{N_1} + \mathbf{n}) + \alpha(\mathbf{N_2} + \mathbf{n}).\end{aligned} \tag{28}$$

Now, let us see what happens with Eq.(27) when we use the same scheme to evaluate:

$$\mathbf{N}' \cdot \mathbf{L} = \frac{2(\mathbf{H} \cdot \mathbf{L})(\mathbf{H} \cdot \mathbf{n})}{\mathbf{H} \cdot \mathbf{H}} - \mathbf{n} \cdot \mathbf{L}. \tag{29}$$

We could evaluate each of the two factors in the numerator for them selves, which will require 2 additions and 1 multiplication. But we could also evaluate the product itself:

$$\begin{aligned}&2(\mathbf{H} \cdot \mathbf{L})(\mathbf{H} \cdot \mathbf{n}) = \\ &2(A_1x + B_1)(A_2x + B_2) = \\ &Ax^2 + Bx + C = p,\end{aligned} \tag{30}$$

where $A = 2A_1A_2$, $B = 2(A_1B_2 + A_2B_1)$ and $C = 2B_1B_2$. Then we can set up similar expressions for A_1, A_2, B_1 and B_2 as we did earlier in the previous section.

The point is that the numerator could be evaluated in 2 additions. The denominator is basically the same as the last time so it will again be evaluated in 2 additions. But we must also evaluate $\mathbf{n} \cdot \mathbf{L}$. This could be done in 1 addition as we saw earlier. And we shall not forget that we have to subtract this value from the result of the division. We could even get rid of this subtraction by multiplying $\mathbf{n} \cdot \mathbf{L}$ by the denominator and subtract the result from the numerator. We will show the principle. Let the denominator be:

$$\mathbf{H} \cdot \mathbf{H} = Dx^2 + Ex + F = q, \tag{31}$$

and

$$\mathbf{n} \cdot \mathbf{L} = G. \tag{32}$$

Then, the numerator can be rewritten as: $(A - GD)x^2 + (B - GE)x + (C - GF)$. If we use this scheme, we could evaluate this expression in 4 additions and 1 division. Compare this to the 3 additions, 1 division and 1 square root of Duff's fast Phong shading. Hence, we substituted the square root by 1 addition in the inner loop. This method is faster in the inner loop, but requires a bit more computation in the setup for each scan line.

4.1 Quality of the new Method

What about the error introduced by choosing a vector which is not halfway between the two vectors? Fig.2 and 3 shows that it is very hard for the human eye to distinguish any differences between reflection shading and Phong shading. The interpolation of the normal between $\mathbf{N_1}$ and $\mathbf{N_2}$ gives us an error itself, since the angle between the normals will not be the same on a scan line. The point is, linear interpolation does not give us the correct normal, neither does reflection shading.

But why is it a good idea to choose $\mathbf{n}$ as the normal of the polygon? The reason for this is that $\mathbf{n} + \mathbf{N}$ should not be $\mathbf{0}$ (the zero vector) or too close. There is of course no guarantee that $\mathbf{n} + \mathbf{N}$ will not be $\mathbf{0}$, but if the surface is not too curved we can use this scheme. Choosing, for example, $\mathbf{L}$ or $\mathbf{V}$ as $\mathbf{n}$ will produce bad results when $\mathbf{n} + \mathbf{N}$ comes close to $\mathbf{0}$. Of course this means that we must have the normal of the polygon available. If we are rotating our object, this means that we have to rotate this 'extra' normal, which obviously is a disadvantage. We can get around this problem by choosing the previous $\mathbf{N}'$ as our new $\mathbf{n}$ so that $\mathbf{H} = \mathbf{N}'_{\mathbf{n-1}} + \mathbf{N_n}$.

4.2 Highlights

If we want to add highlights to Duff's method we could use the expression introduced by Blinn [Blinn78] $\mathbf{H} = \frac{\mathbf{L}+\mathbf{V}}{\|\mathbf{L}+\mathbf{V}\|}$ and then evaluate the specular intensity $I_s = \mathbf{N} \cdot \mathbf{H}$ for each pixel. Since I_s is a dot product just as I_d is, we realize that we could compute the specular component with the same amount of work as for the diffuse component. But we could exploit the fact that we have already computed $\sqrt{q}$ which we will use to normalize both $\mathbf{N} \cdot \mathbf{L}$ and $\mathbf{N} \cdot \mathbf{H}$. Since a division takes much longer time than a multiplication we can calculate $1/\sqrt{(q)}$ first and then multiply this quote with p and r, where r is the recurrence variable for the specular reflection, and p is the recurrence variable for the diffuse reflection. Hence, our total computation for both diffuse and specular light will be 4 additions, 2 multiplications 1 division and 1 square root.

What if we will use Eq.(17) instead? If we would use the fast form for calculating I_s that we developed in the 'Hybrid Shading' section, then we have to calculate the numerator for I_d separately, yielding a total of 5 additions, 2 divisions and 1 square root.

We could also exploit the fact that the numerator and the denominator is the same as in Duff's method, except that we have no square root:

$$\mathbf{R} \cdot \mathbf{V} = L_z - 2N_z \frac{\mathbf{N} \cdot \mathbf{L}}{\mathbf{N} \cdot \mathbf{N}} = L_z - 2N_z \frac{p}{q}. \tag{33}$$

We can make the expression shorter if we use the fact that $2N'_z = 2(\mathbf{k}x + \mathbf{m})$. Then we could use the recurrence $r_{i+1} = r_i + dr_i$, where $r_0 = \mathbf{2m}'$, $dr_i = 2\mathbf{k}$. The quote p/q could, as we saw earlier, be evaluated in 3 additions and 1 division. Our new recurrence can be evaluated in 1 addition and we also have 1 multiplication and 1 subtraction in the original expression. So the total cost of the calculations of I_d and I_s will be 4 additions, 1 subtraction, 1 multiplication, 2 divisions and 1 square root.

What is the difference between using Eq.(17) and Eq.(9)? Actually, as we will see, we will end up with the same equation. Let us use Eq.(9), then we can use the same initial values for r as above. Precalculate $s = 1/\sqrt{(q)}$ in order to get rid of one division, but we get two extra multiplications instead

$$I_s = L_z - rsps. \tag{34}$$

But $ss = (1/\sqrt{q})^2 = 1/q$, so we have the same expression as in Eq.(17), as we said earlier. The total cost of calculating I_d and I_s will be 4 additions, 1 subtraction, 3 multiplications, 1 division and 1 square root. This version is probably faster than the previous one since a multiplication is many times faster than a division on most computers. By using this method, we only need 1 subtraction and 1 multiplication more, in order to calculate $\mathbf{R} \cdot \mathbf{V}$, than we would use if we were about to calculate $\mathbf{N} \cdot \mathbf{H}$. So the advantage of using the latter is not that great. Especially since the latter will produce a value that is greater than the first one, and thus, have to be raised to a higher power to produce the same result. This is the case, since the angle between $\mathbf{N}$ and $\mathbf{H}$ is always smaller than the angle between $\mathbf{R}$ and $\mathbf{V}$.

Fig.2 shows the Venus de Milo statue again. This time it is Phong shaded and Eq.(34) has been used to get the highlights.

4.3 Adding Highlights to the Reflection Shading

If we would like to use Blinn's method in order to add highlights to our reflection shading technique, then we could easily use equation (29) to evaluate

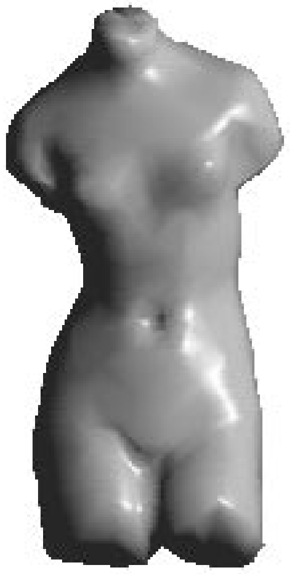

Figure 2: The Venus de Milo statue, Phong shaded with highlights

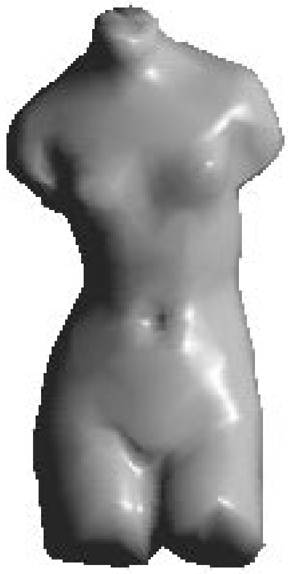

Figure 3: The Venus de Milo statue, reflection shaded with highlights

$\mathbf{N}\cdot\mathbf{H}$ instead of $\mathbf{N}\cdot\mathbf{L}$, where $\mathbf{H}$ is the halfway vector. The denominator will be the same as when we compute the diffuse intensity so the total cost will thus be 6 additions 2 multiplications and 1 division. If we on the other hand want to compute $\mathbf{R}\cdot\mathbf{V}$ instead, we could once again use the fast version from the 'Hybrid Shading' section with the reflection shading method for the diffuse intensity. Hence, we will have a total of 8 additions and 2 divisions. This time we can not use any of the values from the other calculations to make the computation faster. But if we choose to use Eq.(34) we can evaluate it in 6 additions 1 subtraction, 3 multiplications and 1 division. Note that r must be computed with Eq.(27), hence the extra additions. Fig.3 shows the Venus de Milo statue reflection shaded, using this technique for the highlights. The normals along the edges and along the scan lines are interpolated using the 'reflection' technique.

5 POINT LIGHT SOURCES

A simplification often used in applications where speed is crucial, like in games, is to use a parallel light vector $\mathbf{L}$, which means that we can use the same vector over the polygon since it does not change. But if we want to use a single point light source like a spotlight, then we must obtain the vector between the point light source and the pixel that we are currently shading. Once again we need to normalize it, using Eq.(2). If our light source is placed at point F, and our pixel is at point P, then

$$\mathbf{L} = \frac{F-P}{\parallel F-P \parallel}. \tag{35}$$

If our light source is a spotlight, then it has a main direction $\mathbf{S}$, then the intensity, $I_d + I_s$, at pixel P can be multiplied with $(\mathbf{S}\cdot\mathbf{L})^n$, where n will affect the width of the spot which the spot light generates when it shines on an object. This is basically the model introduced by Warn [Warn83]. If our single point light source is not a spotlight, then it shines equally bright in all directions.

As shown in Eq.(28) we could interpolate $F - P$ directly, which is faster, then interpolating P and then subtracting F afterwards. This vector must then be normalized. Of course, we will use a recurrence to do the computations.

5.1 Faster Computation

The trick introduced earlier in the 'Reflection Shading' section could also be used for single point light sources. Once again we must choose a vector that we will wrap around our vector which will lie almost halfway between this vector and the unnormalized vector $\mathbf{L} = F - P$. The obvious choice is $\mathbf{S}$. Therefore, we set $\mathbf{H} = \mathbf{S} + \mathbf{L}$, and equation (27) gives

$$\mathbf{L}' = 2\mathbf{H}\frac{\mathbf{H}\cdot\mathbf{S}}{\mathbf{H}\cdot\mathbf{H}} - \mathbf{S}. \tag{36}$$

Once again we will be in trouble whenever $\mathbf{S}+\mathbf{L}$ is close to $\mathbf{0}$. One way to get around this problem is to choose the previous $\mathbf{L}$ instead, as we have discussed before. Thus $\mathbf{H} = \mathbf{L}'_{\mathbf{n-1}} + \mathbf{L_n}$.

Eq.(36) should be used when we calculate I_d and I_s, but then we will get quite complicated expressions. If we choose to model I_d and I_s with a constant $\mathbf{L}$, then we could still produce a spot by multiplying the intensity with $(\mathbf{S}\cdot\mathbf{L}')^n$. Then we can use Eq.(36) to get:

$$\mathbf{S}\cdot\mathbf{L}' = 2\frac{(\mathbf{H}\cdot\mathbf{S})^2}{\mathbf{H}\cdot\mathbf{H}} - 1. \tag{37}$$

Similar recurrences could be set up for evaluating this equation as we have done before. One of the consequences of using this method is that the width of the spot generated by a spotlight will be smaller than we would get by using ordinary normalization. This is due to the fact that $\mathbf{H}$ is not halfway between $\mathbf{L}$ and $\mathbf{S}$. This is an advantage since we thus could use a smaller n.

6 FIXED VIEW POINT

Another simplification that is often used is having the view vector $\mathbf{V}$ at infinity. If we want the view vector to be at a fixed point, we must calculate the vector from this point to each pixel we are shading. Hence, we must normalize this vector too. If we have a fixed view point at F with the view vector $\mathbf{V}$ pointing at the viewer, we could use the same scheme introduced in the previous section. We set the vector to the viewer from the pixel P on the polygon vie are shading as $\mathbf{V} = F - P$. Once again we can use our method to normalize $\mathbf{V}$. Which vector should we choose to wrap around? The best choice is probably $\mathbf{v} = [0, 0, -1]$ since it is in the middle of our view field, pointing straight into the scene. We set $\mathbf{H} = \mathbf{v} + \mathbf{V}$:

$$\mathbf{V}' = 2\mathbf{H}\frac{\mathbf{H}\cdot\mathbf{v}}{\mathbf{H}\cdot\mathbf{H}} - \mathbf{v}. \tag{38}$$

7 CONCLUSIONS

We have shown that the equation for prefect reflection could be rewritten in a form that does not need an unit length normal. When we implement both forms of the reflection equations, by using Duff's method, we actually get the same expression. A very fast shading method is to use Gouraud shading for the diffuse intensity and compute the specular intensity by using the equation that does not need a unit length normal. Thus, it is a hybrid between Gouraud and Phong shading.

Reflection shading was introduced that utilizes the fact that we do not need a unit length normal in the reflection equation. When we fold a unit length vector around a vector halfway between this vector and the unnormalized vector we get the same vector but with unit length. The problem is to find that vector halfway in between. But we showed that we could easily get an approximation that will produce satisfactory results.

This trick could also be used to obtain unit length vectors when we are using single point light sources or a fixed point view vector.

REFERENCES

[Duff79] T. Duff, *Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays* ACM, Computer Graphics, Vol. 13, 1979, 270-275.

[Bisho86] G. Bishop, D. M. Weimer, *Fast Phong Shading* Computer Graphics vol. 20, No 4, 1986.

[Blinn78] J. F. Blinn, *Models of Light Reflection for Computer Synthesized Pictures* Proc. 5th Conference on Computer Graphics and Interactive Techniques, 1978.

[Foley97] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics - Principles and Practice* Addison-Wesley, 1997.

[Goura71] H. Gouraud, *Continuos Shading of Curved Surfaces*, IEEE transactions on computers vol. c-20, No 6, June 1971.

[Hearn97] D. Hearn, M. P. Baker, *Computer Graphics* Prentice Hall, 1997.

[Ouyan96] S. Ouyang, D. E. Maynard, *Phong Shading by Binary Interpolation* Comput. & Graphics vol. 20, No 6, 1996, pp.839-848.

[Phong75] B. T. Phong, *Illumination for Computer Generated Pictures* Communications of the ACM, Vol. 18, No 6, June 1975.

[Voorh94] D. Voorhies, J. Foran, *Reflection Vector Shading Hardware* Proceedings of SIGGraph, 1994, pp. 163-166.

[Warn83] D. R. Warn, *Lighting Controls for Synthetic Images* ACM, Computer Graphics, Vol. 17, No. 3, 1983, 13-21.