Approximated Phong Shading by using the Euler Method

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
After almost three decades and several improvements, Gouraud shading is still more often used for interactive computer graphics than Phong shading. One of the main reasons for this is of course that Phong shading is computationally more expensive. Quadratic polynomial approximation techniques like Bishop's method could reduce the amount of computation in the inner loop to just the double of what is done in Gouraud shading. By using Euler's method we get another quadratic polynomial approximation technique which is just as fast in the inner loop, but it will also give correct intensities on the edges, which we will not get with Bishop's method. By computing the maximum difference over a scan line between Gouraud shading and the proposed method, we could decide if Gouraud will suffice. It is also shown that linearly interpolated normals are normalized by a symmetric function. This means that we could reduce the amount of square roots by the half in Phong shading.

Keywords: Shading, Illumination, Polynomial approximation, Euler's method

1. Introduction

Since curved surfaces are commonly represented by a mesh of polygons, a shading algorithm is used to prevent the surface from appearing faceted. The fastest shading technique is known as Gouraud shading [5]. In this algorithm , the intensities at the vertices of the polygon are calculated first. Then the intensity of interior pixels are calculated by using bilinear interpolation. This can be done by a forward difference or recurrence sequence in just 1 addition in the inner loop for the diffuse intensity. For larger polygons Gouraud shading is of limited use for calculating the specular intensity, since it tends to smear out the highlight over a polygon. If the highlight falls on to the interior of a polygon, it may not be visible at all. Another disadvantage with Gouraud shading is that it tends to produce more mach bands than Phong shading. Another undesirable effect is produced when the intensity at the vertices happen to be the same. Then the intensity will be the same over the whole polygon.

Phong shading [8] overcomes these problems by interpolating the normals at the vertices over the edges. Then these normals are again linearly interpolated over the scan line. Hence, we will have a normal at each pixel, which can be used in the lighting calculation, and therefore, we will get more accurate diffuse as well as more accurate specular light. This takes, of course, significantly more time than Gouraud shading, not at least depending on the square root needed in the normalization process.

A method which is a compromise is known as fence [1] shading. Normals are linearly interpolated at the edges and are then used for Gouraud shading of the scan lines. Hence, more correct intensities for the edges are obtained and high-

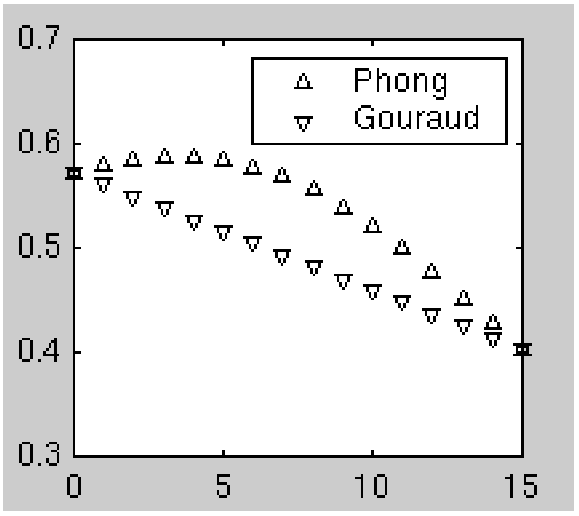

Figure 1: *Difference in intensity between Phong and Gouraud shading along a scan line*

lights will be modeled better in that they are smeared out for each scan line rather than on the whole polygon.

The difference between Phong and Gouraud shading along a scan line is shown in Figure 1 where we clearly see that Gouraud produces a linearly interpolated intensity, whereas Phong will produce a curve. In this and the following diagrams we use normals that have an angle of $60°$ between them. This is used here to make the illustrated differences more clear.

In Phong's illumination model, the intensity at each pixel on the polygons can be modeled by summing the intensities of the diffuse and specular components, as follows:

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

where K_d and K_s are material constants for the diffuse and specular property of the surface, respectively, $\mathbf{N}$ is the normal vector for the surface, $\mathbf{L}$ is a vector pointing in the direction to the light source, $\mathbf{H}$ is the halfway vector introduced by Blinn [3], which is the direction of the normal that would give maximum highlight, 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. Phong used $I_s = (\mathbf{R}\cdot\mathbf{V})^n$ for the specular intensity instead of $(\mathbf{N}\cdot\mathbf{H})^n$. But since Blinn's equation is a dot product between the normal and a constant vector it is similar to the diffuse intensity which also is a dot product between the normal and a constant. Therefore we will focus on the latter since we will discuss how we can use our proposed algorithm for computing the specular intensity as well as the diffuse intensity.

2. Previous Work

Duff [4] showed that the dot products in $\frac{\mathbf{N}\cdot\mathbf{L}}{\sqrt{\mathbf{N}\cdot\mathbf{N}}}$ could be evaluated more efficiently by using finite differences. For successive values of x it could be evaluated with 3 additions, 1 division and one square root per pixel.

Bishop and Weimer [2] took advantage of the fact that many times an approximation is sufficient. They derived a Taylor series from Duff's equation. This quadratic polynomial could be evaluated with only 2 additions per pixel. The advantage with this method is that we do not need to compute the square root at each pixel. The disadvantage is that we get an approximation that will be correct for the pixel at the middle of the polygon, but will get further away from the correct intensity value as we interpolate the intensity towards the edges. Therefore, the intensity at the edge could be significantly different at the two polygons sharing that edge. It should also be noted that the intensity is not calculated on a scan line basis. Their method has a large set up before the intensities are calculated, but then we can move, both along a scan line and between scan lines, with just 2 additions for each step.

Another quadratic interpolation method was introduced by Seiler [9], which also requires 2 additions per pixel. In this approach a shading function is defined that evaluates to the specified intensities at six sample points (the three vertexes and each edge midpoint).

Other approaches have been introduced by Kuijk[6] and by Ouyang and Maynard[7]. The first approach is based on angular interpolation and will give approximations to the Phong curve. The latter will produce correct Phong shading but by using a clever trick the number of square roots will be less, and therefore it is faster than Phong shading.

2.1. Fast Phong Shading

Let us first examine how Duff's method could be implemented for each scan line, rather than for a whole polygon. We want to compute $\frac{\mathbf{N}\cdot\mathbf{L}}{\|\mathbf{N}\|}$, where $\mathbf{L}$ is normalized and $\mathbf{N}$ is not, between the normals $\mathbf{N}_a$ and $\mathbf{N}_b$ on the edges, by using a recurrence. Let $\mathbf{N} = \mathbf{k}x + \mathbf{m}$ along the scan line from x_1 to x_2, where $n = x_2 - x_1$, $\mathbf{k} = (\mathbf{N}_b - \mathbf{N}_a)/n$ and $\mathbf{m} = \mathbf{N}_a$. Now, let:

$$\mathbf{N}\cdot\mathbf{L} = Ax + B = p, \tag{2}$$

where $A = \mathbf{k}\cdot\mathbf{L}$ and $B = \mathbf{m}\cdot\mathbf{L}$. And for the normalization, let

$$\mathbf{N}\cdot\mathbf{N} = Cx^2 + Dx + E = q, \tag{3}$$

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

$$\frac{\mathbf{N}\cdot\mathbf{L}}{\|\mathbf{N}\|} = \frac{Ax+B}{\sqrt{Cx^2+Dx+E}} = \frac{p}{\sqrt{q}}. \tag{4}$$

Now we can set up the following recurrence:

$$p_{i+1} = p_i + dp_i \tag{5}$$
$$q_{i+1} = q_i + dq_i \tag{6}$$
$$dq_{i+1} = dq_i + d^2q \tag{7}$$

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.

3. Euler Shading

If we have normalized normals at the edges we will get correct intensities on the edges, and then we can use a quadratic polynomial to get approximations of the intensity for the interior of the polygon. To achieve good approximations we can borrow a method originally used for solving differential equations, known as Euler's method. It can be used to approximate a curve with a known start point and a given derivative at each point. We begin with the intensity at the start point and as we proceed in the x direction we add the derivative to this value. The smaller the step we choose the better will the approximation be, but we will step one pixel at a time. The recurrence we will use is:

$$y_{i+1} = y_i + dy_i \tag{8}$$

The derivative of Eq. (4) is:

$$di = \frac{d}{dx}(p/\sqrt{q}) = \frac{1}{\sqrt{q}}\left(\frac{dp}{dx} - \frac{p}{2q}\frac{dq}{dx}\right). \tag{9}$$

This equation still contains a square root so it will not be efficient to compute it for each step. We will therefore use an approximation for the derivative. We shall simply use the linear interpolation of the derivatives at the edges. Note that $q = 1$ on the edges, therefore the derivatives at the edges are:

$$di_a = \left(\frac{dp_a}{dx} - \frac{p_a}{2}\frac{dq_a}{dx}\right) \tag{10}$$
$$di_b = \left(\frac{dp_b}{dx} - \frac{p_b}{2}\frac{dq_b}{dx}\right), \tag{11}$$

where $p_a = \mathbf{N}_a \cdot \mathbf{L}$, $p_b = \mathbf{N}_b \cdot \mathbf{L}$, $dp_a = dp_b = A$ and $dq_a = D$. At the other edge we will have $dq_b = 2nC + D$. Then we use the recurrence $s_{i+1} = s_i + ds$ to evaluate the derivative with initial values:$s_0 = di_a$ and $ds = (di_b - di_a)/n$. Finally the Euler method for approximating the Phong shading curve will be

$$p_{i+1} = p_i + s_i \tag{12}$$
$$s_{i+1} = s_i + ds. \tag{13}$$

This approach requires that the normals at the edges are normalized. So, in some sense, this is an improved version of fence shading .

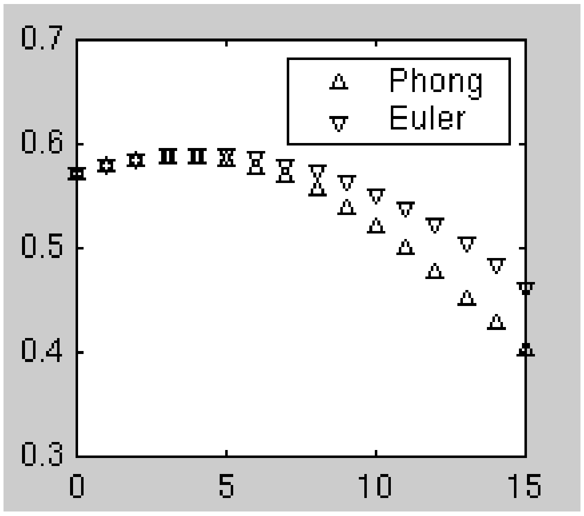

Figure 2: *Difference in intensity between Phong and Euler shading along a scan line*

3.1. Compensating for the Error

Since the derivative we use is only an approximation and the Euler method itself produces an approximation, we will end up with another value for p_n than p_b on the other edge. This is shown in Figure 2 where we can observe the deviation at the right edge.

If we could compute the error in advance, then we could linearly interpolate from 0 at the first edge to the pre calculated error at the other edge and subtract this value from our approximation so that we will have correct values at the edges. Since p_i is calculated as a recurrence we know that we will end up with the following value

$$p_n = p_a + ndi_a + \frac{n^2 - n}{2}ds. \tag{14}$$

Note that, as we add ds to p we also add ds to s, yielding the series $(1 + 2 + 3 + ... + (n-1))ds = \frac{n^2-n}{2}ds$. Hence, we know that the error will be:

$$\varepsilon = p_n - p_b. \tag{15}$$

And then we have to compute I_d as $p_i - \varepsilon_i$ as well as ε_i itself. Here we can do a simplification by setting up this recurrence:

$$p_{i+1} = p_i + t_i \tag{16}$$
$$t_{i+1} = t_i + ds, \tag{17}$$

where $d\varepsilon = \varepsilon/N$ and $t_0 = di_a - d\varepsilon$.

This will make sure that we will have correct values at the edges as shown in Figure 3. Note, that we still get a quadratic approximation. The reason for this is that if we

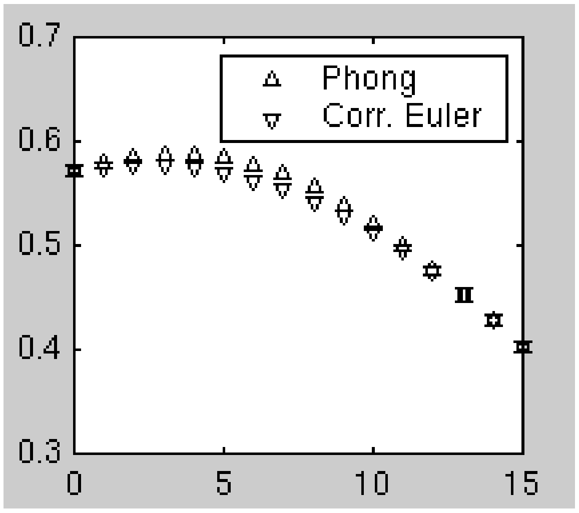

Figure 3: *Difference in intensity between Phong and corrected Euler shading along a scan line*

subtract a linear function from a quadratic function we will still have a quadratic function as the result.

3.2. Faster Computation

There are some possible improvements that will make this algorithm faster. We must compute $dq_b = 2nC + D$ for each scan line, and we will prove that $dq_b = -dq_a$ which is faster to compute since we only have to invert the sign of dq_a.

Proof We have $dq_a = D = 2\mathbf{k}\cdot\mathbf{N}_a$ and $dq_b = 2nC + D = 2n\mathbf{k}^2 + 2\mathbf{k}\cdot\mathbf{N}_a$. We also have $\mathbf{k} = (\mathbf{N}_b - \mathbf{N}_a)/n$, clearly:

$$\mathbf{N}_a \cdot \mathbf{k} = -\mathbf{N}_b \cdot \mathbf{k}. \tag{18}$$

Use this fact:

$$dq_b = 2n\mathbf{k}^2 - 2\mathbf{k}\cdot\mathbf{N}_b \tag{19}$$

Expand $n\mathbf{k}^2$:

$$\begin{aligned} n\mathbf{k}^2 &= n\frac{(\mathbf{N}_b - \mathbf{N}_a)(\mathbf{N}_b - \mathbf{N}_a)}{n^2} \\ &= 2\frac{1 - \mathbf{N}_a\cdot\mathbf{N}_b}{n} = 2\mathbf{k}\cdot\mathbf{N}_b \end{aligned} \tag{20}$$

Substitute for this in Eq.(19), then:

$$dq_b = 4\mathbf{k}\cdot\mathbf{N}_b - 2\mathbf{k}\cdot\mathbf{N}_b = 2\mathbf{k}\cdot\mathbf{N}_b \tag{21}$$

Finally Eq.(18) gives:

$$dq_b = -2\mathbf{k}\cdot\mathbf{N}_a \tag{22}$$

□

We must also keep in mind that we have to compute the normalized normals at the edges. Therefore, we do not have to use this method for scan lines with only one or two pixels, since we can use these normals directly.

We could also easily tabulate $\frac{n^2-n}{2}$ so that we do not need to compute it for each scan line.

4. Symmetry of the Normalization

We have proven that $dq_b = -dq_a$. This means that $dq = 0$ in the middle of the scan line since the derivative of q is a straight line. But $q_a = q_b = 1$, and q is a quadratic curve. So we have a quadratic curve over the scan line with same values at the edges and with the extreme point in the middle of the scan line. Since such a curve must be symmetric, we do not have to compute q for the whole scan line since we can reuse these values for the second part of the scan line. Moreover this means that in Phong shading we only need to compute $\sqrt{q}$ for the first half of the scan line and then use the same values for the second part. We could do that by shading the scan line both from left to the middle and at the same time form right to the middle. In this way we also need only half the amount of loop increments. This must speed up Phong shading significantly.

In our approach we could make use of the symmetry when we compute the normalized normals at the edges.

5. Adaptive shading

An interesting fact is that sometimes the Phong method will produce the same discrete color value as the Gouraud method over the whole scan line. An empirical test was done where the difference of Phong and Gouraud intensities was multiplied with 255 and then truncated, and the number of scan lines which had the same color along the scan line was counted. We used a torus which was modeled with a different number of polygons. Each torus was rendered in 200 frames and rotated in the x, y and z directions. Table 1 shows in percent how many of the scan lines could have been computed with Gouraud instead of corrected Euler, with the same result.

Polygons	Scan lines
288	54.9%
512	69.3 %
800	80.9 %
1152	89.0 %

Table 1: *Percentage of scan lines that could have been computed with Gouraud instead of corrected Euler*

The maximum difference between corrected Euler and Gouraud along a scan line could be found by differentiating the difference between their functions. Where this difference is zero we have our extreme point. The derivative for Gouraud is d_p and the derivative for corrected Euler is $t_0 + nd_s$. Solving for n gives:

$$n = \frac{dp - t_0}{ds} \tag{23}$$

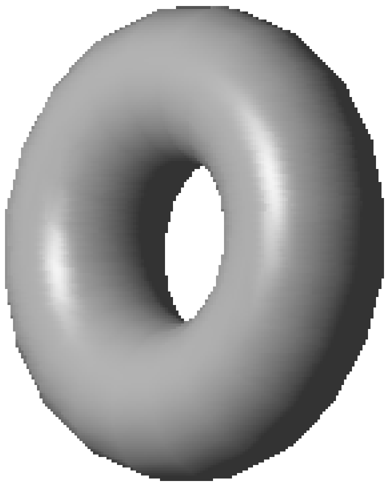

Figure 4: *Torus with 800 polygons shaded with the corrected Euler method*

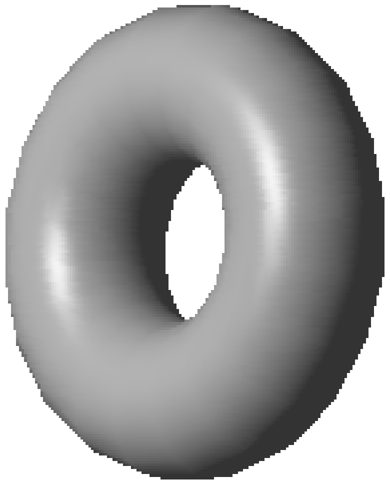

Figure 5: *Phong shaded torus with 800 polygons*

Since we have a recursion we need a n which is an integer. Therefore we need to truncate it an add 1 to make sure that the whole 'step' is done. Use this n to find the difference between the functions:

$$\delta = nd_p - (nt_0 + \frac{n^2 - n}{2} ds) \tag{24}$$

If $\mid \delta * 255 * K_d \mid < 1.0$ then the diffuse intensity could be computed just as well with Gouraud instead of corrected Euler, since they would give the same result.

6. Evaluation of the Proposed Method

Since we will get an approximation of the Phong curve we would like to know how close this approximation is to the correct Phong value. Since we will transform this value into a discrete color value in the frame buffer, we could multiply our intensity value with 255 and then truncate it to get an eight bit color depth. In Table 2 we have computed the average and maximum deviation for Gouraud, Euler and corrected Euler compared to the correct Phong value. Once again, each torus was rendered in 200 frames and rotated in x, y and z directions. About 4 million pixels where rendered for each object.

It is obvious that the corrected Euler method not only perform much better than Gouraud, it also come very close to the Phong value. An maximum deviation of 1 is not much. We could even get that from rounding errors between different implementations of the same method. And a maximum deviation of 3 is still not much, but in this case the torus will not look good anyway, since it has too few polygons. And this is true both for Phong shading and corrected Euler shading. The table also shows that the correction is necessary to produce a good approximation.

	Gouraud		Euler		Corr. Euler	
Polygons	dev.	max	dev.	max	dev.	max
288	1.08	15	0.75	21	0.081	3
512	0.60	9	0.42	10	0.033	1
800	0.38	6	0.28	6	0.016	1
1152	0.26	4	0.21	5	0.009	1

Table 2: *Average and maximum deviation between the Phong curve and other methods*

6.1. Highlights

Is the proposed method also suitable for computing $\mathbf{N} \cdot \mathbf{H}$? In Figure 4 we have used the corrected Euler method for both the diffuse and specular intensity. In Figure 5 we have used Duff's method for both the diffuse and specular intensity. Since the corrected Euler method will produce intensities that lie very close to what Duff's method produces, it should work well for specular computations as well. And the figures shows that this is true.

7. Conclusions

By using the proposed method we could evaluate a quadratic polynomial approximation for the Phong curve in 2 additions per pixel for both diffuse and specular light.

The method will produce results that are within a single intensity level from Phong shading unless the difference in angle for the normals at the edges is large. The set up for each scan line in the proposed method requires the normals at the edges to be normalized. The computational cost for this can be decreased by half by using the symmetry of the interpolating normalizing function. As a side comment it has been pointed out that this symmetry also can be used to decrease the computational cost for conventional Phong shading.

7.1. Future work

This method should be more extensively experimentally compared to other methods such as those mentioned in the introduction. Is it really faster and will it produce better results? For an example, Bishop's method will not give correct values at the edges and have a large setup for the polygon.

Since we have the derivative for the specular light, maybe we could use this to make some decision whether we have to compute the specular intensity for a scan line at all.

Maybe we also could gain something if we computed $(\mathbf{N}\cdot\mathbf{H})^2$ in order to get rid of the square root. The derivative will then be simpler.

References

1. U. Behrens. *Graphics Gems IV*, pp. 404-409 (1994, Boston). Academic Press. Edited by Paul Heckbert. 1
2. G. Bishop, D. M. Weimer, *Fast Phong Shading* Computer Graphics vol. 20, No 4, 1986. 2
3. J. F. Blinn, *Models of Light Reflection for Computer Synthesized Pictures* Computer Graphics, 11 (2) 192-8, 1977. 2
4. T. Duff, *Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays* ACM, Computer Graphics, Vol. 13, 1979, 270-275. 2
5. H. Gouraud, *Continuos Shading of Curved Surfaces*, IEEE transactions on computers vol. c-20, No 6, June 1971. 1
6. A. A. M. Kuijk, E. H. Blake, *Faster Phong Shading via Angular Interpolation* Computer Graphics Forum, 8 315-324 1989. 2
7. S. Ouyang, D. E. Maynard, *Phong Shading by Binary Interpolation* Comput. & Graphics vol. 20, No 6, 1996, pp.839-848. 2
8. B. T. Phong, *Illumination for Computer Generated Pictures* Communications of the ACM, Vol. 18, No 6, June 1975. 1
9. L. Seiler, *Quadratic Interpolation for Near-Phong Quality Shading* Proceedings of the conference on SIGGRAPH 98: conference abstracts and applications, 1998, Page 268.

2