Fast Near Phong-Quality Software Shading

Tony Barrera
Barrera Kristiansen AB
Uppsala,Sweden
tony.barrera@spray.se

Anders Hast
Creative Media Lab
University of Gävle, Sweden
aht@hig.se

Ewert Bengtsson
Centre For Image Analysis
Uppsala University, Sweden
ewert@cb.uu.se

ABSTRACT

Quadratic shading has been proposed as a technique giving better results than Gouraud shading, but which is substantially faster than Phong shading. Several techniques for fitting a second order surface to six points have been proposed. We show in this paper how an approximation of the mid-edge samples can be done in a very efficient way. An approximation of the mid-edge vectors are derived. Several advantages are apparent when these vectors are put into the original formulation. First of all it will only depend on the vertex vectors. Moreover, it will simplify the setup and no extra square roots are necessary for normalizing the mid-edge vectors. The setup will be about three times faster than previous approaches. This makes quadratic shading very fast for interpolation of both diffuse and specular light, which will make it suitable for near Phong quality software renderings.

Keywords

Quadratic shading, Phong shading.

1. INTRODUCTION

Quadratic shading has been proposed as a technique giving better results than Gouraud shading [Gour71], but which is substantially faster than Phong shading [Phon75]. Quadratic interpolation could be setup by fitting a second order surface to six points. Both the diffuse and specular light must be computed at these points and the best quality is obtained if these are interpolated separately. Furthermore, the power in the specular computation must be computed per pixel. Besides this computation, quadratic shading, including both diffuse and specular light, can be performed using four additions per pixel instead of four additions and a reciprocal square root for Phong shading.

The main drawback using quadratic shading has been the rather complex rasterization setup. This paper proposes a solution to this problem and the simplified setup presented in this paper will be about three times faster than previous setup approaches. Hence, quadratic shading will be suitable for software rendering and could therefore be implemented on hand held devices like cell phones etc.

Conference proceedings ISBN 80-86943-03-8
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

2. PREVIOUS WORK

A 3D object represented by polygons will appear faceted when rendered, unless some kind of shading technique is used that interpolates intensities or colors over the polygons. Gouraud uses bilinear interpolation of the colors at the vertices. If the intensities are interpolated, then only one addition per pixel is necessary to achieve this type of shading.

The intensity over the polygon in screen space can be considered as a plane in (x, y, Φ)-space, where x and y are the screen coordinates, and Φ is the intensity. Phong uses bilinear interpolation of the normals at the edges, to obtain intermediate normals for each pixel. These normals are then used in the illumination computation. Duff [Duff79] shows how the computation can be implemented in an efficient way, requiring only three additions, one division and one square root per pixel for Phong shading. The intensity will form a smooth surface in (x, y, Φ)-space. Phong shading produces better results than Gouraud. Nonetheless, Phong does not produce correct shading, in the sense that Phong shading is also an approximation of the 'real' shading curve. This can easily be proved by the fact that the Phong intensity curve is not necessarily C^1 continuous over polygon edges.

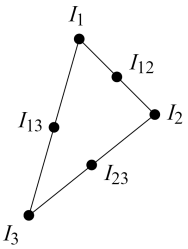

Figure 1: *Sample points for quadratic shading, where the mid edge samples are I_{12}, I_{13} and I_{23}.*

Therefore, an approximation of the intensity curve will suffice. Bishop and Weimer [Bish86] use a Taylor series expansion of Duff's equation to obtain a second order bi-variate polynomial. It could be evaluated by only two additions per pixel along a scanline. This quadratic interpolation scheme will produce a second order polynomial surface in (x, y, Φ)-space.

Kappel [Kapp95] uses a surface fitting technique based on Powell-Sabin [Powe77] quadratic interpolation using the Cendes-Wong [Cend87] formulae. The purpose of that approach is to eliminate Mach bands by generating a C^1 continuous surface over the polygon. This is possible by subdividing the triangle into smaller triangles. Kirk et al. [Kirk92] solves this problem by fitting a surface to a triangle, where the intensity of the vertices of adjacent triangles are the same, and the first derivative of the intensity at the edges tend toward being continuous.

Quadratic shading

Kirk and Voorhies [Kirk90] adopt another approach to quadratic shading which is somewhat different from previous approaches. They show that quadratic interpolation could be setup by fitting a second order surface to six points, yielding a polynomial with six unknown coefficients, which must be determined. The sample points are the vertices and edge mid points of a triangle, as shown in Fig 1.

We will throughout this paper assume that a parallel light source is used, as well as a constant view vector. However, it will be discussed how a point light source can be used and what complications it will give for the algorithm. A parallel light source is often used when speed is crucial, as it usually is for software renderings on hand held devices. The light vector must be interpolated over the polygon using Phong shading when point light sources are used. However, for quadratic shading it is possible to use the light source vectors at the six sample points instead. Furthermore, the view vector should be interpolated over the polygon for Phong shading. However, a simplification often made when speed is crucial is to use a constant view vector, i.e. the viewer is assumed to be at infinity. With a distant light source the six samples are computed as

$$I_1 = \mathbf{L} \cdot \mathbf{N}_1 \tag{1}$$

$$I_2 = \mathbf{L} \cdot \mathbf{N}_2 \tag{2}$$

$$I_3 = \mathbf{L} \cdot \mathbf{N}_3 \tag{3}$$

$$I_{12} = \mathbf{L} \cdot \frac{\mathbf{N}_1 + \mathbf{N}_2}{\| \mathbf{N}_1 + \mathbf{N}_2 \|} = \frac{I_1 + I_2}{\| \mathbf{N}_1 + \mathbf{N}_2 \|} \tag{4}$$

$$I_{13} = \mathbf{L} \cdot \frac{\mathbf{N}_1 + \mathbf{N}_3}{\| \mathbf{N}_1 + \mathbf{N}_3 \|} = \frac{I_1 + I_3}{\| \mathbf{N}_1 + \mathbf{N}_3 \|} \tag{5}$$

$$I_{23} = \mathbf{L} \cdot \frac{\mathbf{N}_2 + \mathbf{N}_3}{\| \mathbf{N}_2 + \mathbf{N}_3 \|} = \frac{I_2 + I_3}{\| \mathbf{N}_2 + \mathbf{N}_3 \|} \tag{6}$$

A second order polynomial is constructed using these six samples and the coordinates of the six point. Such a quadratic shading function is defined by

$$\Phi(x,y) = Ax^2 + By^2 + Cxy + Dx + Ey + F \tag{7}$$

Kirk and Voorhies do not show how these coefficients are computed, neither do Saxe et al. [Saxe96] who use this type of quadratic interpolation in Pixel-Planes 5 to display radiosity illuminated models [Fuch89].

Seiler [Seil98] proposes a simple and fast way to setup the coefficients for this type of quadratic shading. How the computation can be done is also shown by Abbas et al. [Abba01a, Abba01b] but they do not try to make the computation as efficient as Seiler does. Both methods do not cover special cases which will lead to division by zero. This problem however, is solved by Lee and Jen [Lee01] who have simplified Seiler's approach. In [Hast01] it is shown how these methods can be simplified further in order to make the computation of the coefficients as efficient as possible.

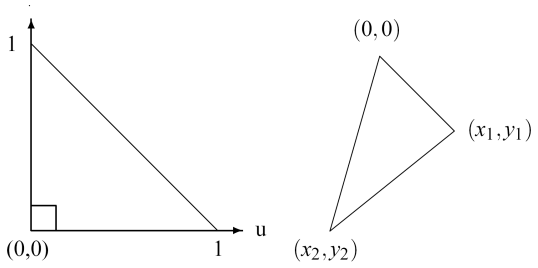

Figure 2: *To the left: Ortho-normalised triangle. To the right: a polygon with relative vertices*

In general the coefficients can be obtained by setting up a system of equations using equation (7) and the relative coordinates of the sample points. Relative coordinates are preferable since it will yield a system which is easier to solve, as it will contain more zeroes. The relative coordinates are obtained by

shifting the triangle so that the top vertex has coordinate (0,0). The middle vertex will have coordinate (x_1, y_1), and the bottom most vertex will be (x_2, y_2), as shown to the right in figure 2. This implies that the vertices have to be sorted in the y direction. Note that, these operations are done in screen space and it is here presumed that the top left corner is coordinate (0,0). The system of equations is

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ x_1^2 & y_1^2 & x_1 y_1 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2 y_2 & x_2 & y_2 & 1 \\ \left(\frac{x_1}{2}\right)^2 & \left(\frac{y_1}{2}\right)^2 & \frac{x_1}{2}\frac{y_1}{2} & \frac{x_1}{2} & \frac{y_1}{2} & 1 \\ \left(\frac{x_2}{2}\right)^2 & \left(\frac{y_2}{2}\right)^2 & \frac{x_2}{2}\frac{y_2}{2} & \frac{x_2}{2} & \frac{y_2}{2} & 1 \\ \left(\frac{x_1+x_2}{2}\right)^2 & \left(\frac{y_1+y_2}{2}\right)^2 & \frac{x_1+x_2}{2}\frac{y_1+y_2}{2} & \frac{x_1+x_2}{2} & \frac{y_1+y_2}{2} & 1 \end{bmatrix} \quad (8)$$

And the problem is to solve this equation

$$M\mathbf{c} = \Phi \quad (9)$$

where

$$\Phi = [I_1, I_2, I_3, I_{12}, I_{13}, I_{23}]^T \quad (10)$$

$$\mathbf{c} = [A, B, C, D, E, F]^T \quad (11)$$

2.1.1 Alternative derivation

An alternative and ultimately slightly faster approach is to use an ortho-normalized triangle with p_0 = (0,0), p_1 =(1,0) and p_2 = (0,1), as shown to the left in figure 2. It has been obtained by first sorting the vertices, and then shifting them as shown in the same figure to the right. The coefficients for a bi-variate polynomial over this triangle is first computed by setting up the following system of equations

$$\Phi = M\mathbf{c} \quad (12)$$

where

$$\Phi = [I_1, I_2, I_3, I_{12}, I_{13}, I_{23}]^T \quad (13)$$

$$\mathbf{c} = [a, b, c, d, e, f]^T \quad (14)$$

And

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ \frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{2} & 1 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \quad (15)$$

The solution for $\mathbf{c} = M^{-1}\Phi$ is

$$\begin{aligned} a &= 2(I_1 + I_2 - 2I_{12}) \\ b &= 2(I_1 + I_3 - 2I_{13}) \\ c &= 4(I_1 + I_{23} - I_{13} - I_{12}) \\ d &= I_2 - I_1 - a \\ e &= I_3 - I_1 - b \\ f &= I_1 \end{aligned} \quad (16)$$

where

$$\Phi(u, v) = au^2 + bv^2 + cuv + du + ev + f \quad (17)$$

It is shown in [Hast03a] and [Hast02b] how the setup variables needed for incrementally stepping along the edges as well as along the scanlines, can be computed directly using equation 17 instead of using equation 7. This approach is actually slightly faster. It was also shown in [Hast02a] how this could be used for bump mapping. The reason why the ortho-normalized triangle is used is that the coordinates for the vertices are (0,0), (1,0) and (0,1) and this yields a much more simple matrix inverse to solve, i.e the inverse of equation (15). The three different solutions discussed in [Hast01] and originally developed by Lee & Jen [Lee01], Seiler [Seil98] and Abbas et al. [Abba01a, Abba01b] are actually three approaches for solving this inverse. The inverse of this new matrix formulation becomes much more straight-forward to solve. Next a transformation is needed from the actual triangle (the triangle to the right in figure 2) into the so called ortho-normalized triangle. For any point (x, y) on the actual triangle, a corresponding point (u, v) on the normalized triangle can be obtained by the following transformation

$$u = x\alpha_u + y\beta_u \quad (18)$$

$$v = x\alpha_v + y\beta_v \quad (19)$$

where

$$\begin{aligned} \alpha_u &= y_2\omega \\ \beta_u &= -x_2\omega \\ \alpha_v &= -y_1\omega \\ \beta_v &= x_1\omega \end{aligned} \quad (20)$$

and

$$\omega = \frac{1}{x_1 y_2 - x_2 y_1} \quad (21)$$

Finally equations (18) and (19) are put into equation (17) and the terms are rearranged so that we obtain the coefficients for (7). This is done by extracting the terms multiplied by x^2, x and so forth. The coefficients are

$$
\begin{aligned}
A &= a\alpha_u^2 + b\alpha_v^2 + c\alpha_u\alpha_v \\
B &= a\beta_u^2 + b\beta_v^2 + c\beta_u\beta_v \\
C &= 2a\alpha_u\beta_u + 2b\alpha_v\beta_v + c(\alpha_u\beta_v + \alpha_v\beta_u) \\
D &= d\alpha_u + ea\alpha_v \\
E &= d\beta_u + e\beta_v \\
F &= f
\end{aligned} \qquad (22)
$$

The time needed for computing these coefficients as well as the computation time for computing them using the previously explained approaches will be shown later in this paper.

3. APPROXIMATION OF MID-EDGE VECTORS

One of the drawbacks with the quadratic setup is the computation and normalization of the mid-edge vectors that are needed in order to compute the intensities I_{12}, I_{13} and I_{23} as equations (4) through (6) show. We will now show that these computations can be avoided and the setup will end up being even simpler but also faster. Our working name for this approach has been X-shading and we will therefore refer to it as that in the subsequent sections. An approximation based on one step of the Newton-Raphson method was used by Wynn [Wynn01] for normalization of an interpolated normal

$$\mathbf{N}' = \frac{\mathbf{N}}{2}(3 - \mathbf{N}\cdot\mathbf{N}) \qquad (23)$$

We shall derive an equation for computing the approximation of a normalized vector that is halfway in between two vectors. The vector in between is computed as

$$\mathbf{N}_{1/2} = \frac{\mathbf{N}_1 + \mathbf{N}_2}{2} \qquad (24)$$

Use equation (23) to normalize equation (24), by letting $\mathbf{N} = \mathbf{N}_{1/2}$, in the following way

$$\mathbf{N}'_{1/2} = \frac{\mathbf{N}_1 + \mathbf{N}_2}{4}\left(3 - \frac{\mathbf{N}_1 + \mathbf{N}_2}{2}\cdot\frac{\mathbf{N}_1 + \mathbf{N}_2}{2}\right) \qquad (25)$$

Expand the equation and note that the normals are normalized and therefore $\mathbf{N}_1 \cdot \mathbf{N}_1 = 1$ and $\mathbf{N}_2 \cdot \mathbf{N}_2 = 1$. Hence

$$\mathbf{N}'_{1/2} = \frac{\mathbf{N}_1 + \mathbf{N}_2}{4}\left(3 - \frac{2(\mathbf{N}_1\cdot\mathbf{N}_2) + 2}{4}\right) \qquad (26)$$

And after rearranging the terms

$$\mathbf{N}'_{1/2} = (\mathbf{N}_1 + \mathbf{N}_2)\frac{5 - \mathbf{N}_1\cdot\mathbf{N}_2}{8} \qquad (27)$$

Accuracy of the proposed approximation

This approximation of the mid-edge vectors works quite well for normals with an angle that are less than 45° as shown in figure 3 where the vector norm of the mid-edge vector is computed for different angles. The norm is very close to one for angles less than 45°. Then the norm starts to decrease rather rapidly, as shown in the figure, but it should be noted that the scale goes from 0.88 to 1.0. Hence, the approximation is still not so bad for angles over 45°.

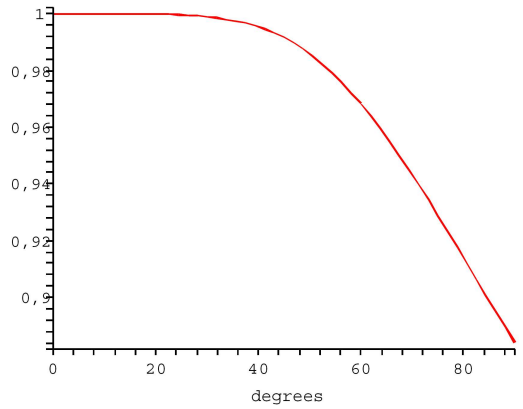

Figure 3: *The vector norm of the approximation for different angles between the vectors.*

The approximation will always do better than a linear approximation, which assures that the quadratic approach will always be better than Gouraud shading. However, the quality of the highlight will be affected for larger angles. It could be argued though, that such large angles means that the polygons should be subdivided further.

4. FASTER QUADRATIC SETUP

In the next step we use this approximation for computing the mid-edge vectors and then these are substituted into equations (16).
The intensities are computed as

$$I_{12} = \mathbf{L}\cdot\mathbf{N}'_{1/2} = \mathbf{L}\cdot(\mathbf{N}_1 + \mathbf{N}_2)\frac{5 - \mathbf{N}_1\cdot\mathbf{N}_2}{8} \qquad (28)$$

Hence

$$I_{12} = (I_1 + I_2)\frac{5 - \mathbf{N}_1\cdot\mathbf{N}_2}{8} \qquad (29)$$

Similarly

$$I_{13} = (I_1 + I_3)\frac{5 - \mathbf{N}_1\cdot\mathbf{N}_3}{8} \qquad (30)$$

$$I_{23} = (I_2 + I_3)\frac{5 - \mathbf{N}_2\cdot\mathbf{N}_3}{8} \qquad (31)$$

Use equations (29) through (31) and substitute them into equations (16). After simplification, the solution for X-shading now becomes

$$
\begin{aligned}
a &= (I_1 + I_2)(\mathbf{N}_1 \cdot \mathbf{N}_2 - 1)/2 \\
b &= (I_1 + I_3)(\mathbf{N}_1 \cdot \mathbf{N}_3 - 1)/2 \\
c &= a + b - (I_2 + I_3)(\mathbf{N}_2 \cdot \mathbf{N}_3 - 1)/2 \\
d &= I_2 - I_1 - a \\
e &= I_3 - I_1 - b \\
f &= I_1
\end{aligned}
\tag{32}
$$

It should be mentioned that if a point light source was used instead of a parallel light source, then we for an example would obtain

$$
I_{12} = \mathbf{L}_{12} \cdot (\mathbf{N}_1 + \mathbf{N}_2) \frac{5 - \mathbf{N}_1 \cdot \mathbf{N}_2}{8} \tag{33}
$$

where $\mathbf{L}_{12}$ is the light source vector in the direction to the point light source at the sample point in question. This will make the setup a bit more complicated. Nonetheless, the normalization of the mid-edge vector is still avoided.

Note that this solution only depends on the vertex normals and the reason for this is of course that the approximation of the mid-edge vectors only depends on the vertex normals. The effect of this is that the extra normalization of the mid-edge vectors, which are shown in (4) through (6), disappears. Hence, a substantial amount of computation time is saved by using the proposed method, especially on systems where there is no hardware for computing divisions and square roots.

Lee & Jen	Abbas et al.	Seiler	Hast et al.	X-shading
40.6	42.0	49.4	40.3	12.7

Table 1: *Timing in seconds of one hundred million computations of the coefficients on an 1.8 GHz Pentium 4.*

Table 1 shows the timing for the computation of the coefficients for equation (7). The earlier mentioned methods proposed by Lee and Jen, Siler and Abbas et al, and the fast version proposed by Hast et al. in [Hast02b, Hast03a] and the method proposed in this paper for one hundred million computations.

The new method is about three times faster than previous methods. It should be mentioned that the approximation of the mid-edge vectors could be used for all the other previously mentioned approaches as well. However, we chose to use it for the fastest approach in this paper. Another important thing to mention is that the dot products in equations (32) could be precomputed and stored since the angle between the vertex normals are invariant to affine transformations such as translation and rotation. In fact ($\mathbf{N}1 \cdot \mathbf{N}2 - 1)/2$ and so forth could be precomputed and stored. This will make the proposed approach even faster.

Figure 4: *A Phong shaded Venus de Milo statue.*

Figure 5: *A shaded Venus de Milo statue using the proposed approach*

5. HIGH QUALITY HIGHLIGHTS

Both the diffuse and specular light can be interpolated using the proposed X-shading approach. When near Phong quality highlights are required for large polygons, it is necessary to do one interpolation for each type of light. The diffuse light is generally computed as

$$I_d = \mathbf{N} \cdot \mathbf{L} \quad (34)$$

where **N** is the normal and **L** is the vector in the direction to the light source. We omit the color of the surface and the color of the light source for simplicity. The intensities computed for each vertex using this equation can be interpolated by the proposed approach. Likewise, the specular intensity can be interpolated separately by computing the specular intensity for each vertex by

$$I_s = \mathbf{N} \cdot \mathbf{H} \quad (35)$$

where **H** is the halfway vector introduced by Blinn [Blin77]. Note, that this is not the full specular computation. For each pixel it is necessary to compute $(I_s)^s$ where s is the so called shininess. A faster technique is proposed in [Hast03b] for computing the specular light, which will be very well suited for software rendering. The fact that two different interpolations are used does not make the whole setup twice as large. Several intermediate variables needed in the first setup for the diffuse light can be reused for the second setup for the specular light. Anyhow, it will be faster than a true Phong interpolation, which will need two bilinear interpolations, one bi-quadratic interpolation, as well as one division and one square root per pixel, in order to compute the diffuse and specular light. Figure 4 shows the back of the famous Venus de Milo statue that has been shaded by Phong shading using a polygon setup approach. This image can be compared to the same object in figure 5 shaded by the proposed X-shading approach. The difference is hardly noticeable.

A torus with only 288 triangles where every two adjacent triangles constitute a planar surface, was shaded in figure 6 and 7. The maximum angle between two normals are as high as 41.4° and obviously it should have been subdivided further since the contour is not looking good. Anyway, the shading and especially the highlights are indistinguishable from each other. The conclusion is that X-shading, even though an approximation is used for the mid-edge vectors, still will produce satisfactory results. Moreover, it is much faster than Phong shading and even faster than previous quadratic shading approaches. This will make X-shading attractive for software rendering. Especially since several divisions and all the square roots needed for computing the coefficients in the bi-variate quadratic equation have been removed from the setup.

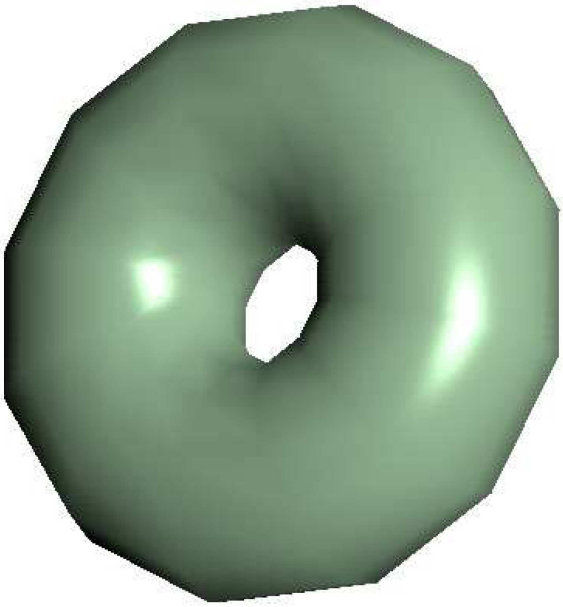

Figure 6: *A Phong shaded torus with 288 polygons.*

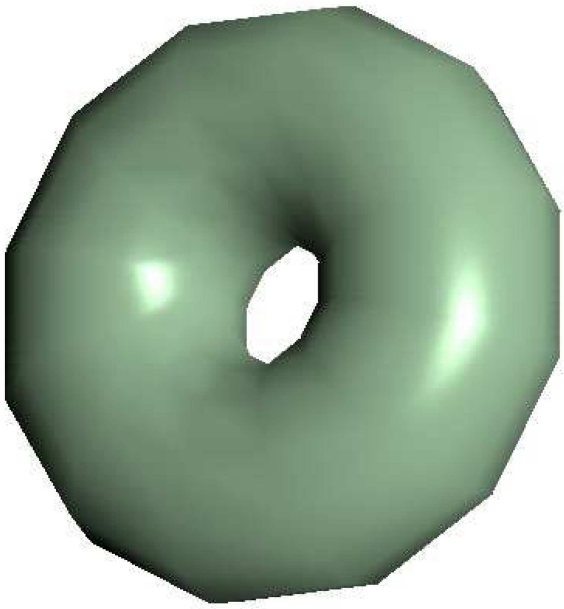

Figure 7: *A shaded 288 polygon torus using the proposed approach*

6. CONCLUSIONS

The proposed setup approach for quadratic shading is about three times faster than previous approaches. The approximation is accurate enough for normals where the angle is less than 45°, and will still do better than Gouraud shading for larger angles. Since the angles are usually smaller than that, otherwise the contour of the object will look far from smooth, the proposed method will produce near Phong quality highlights without any square roots and only one

division needed for computing the coefficients in the bi-variate quadratic equation. This can be compared with the other approaches that need at least three more divisions and square roots as implied by equations (4) through (6). Thus, the proposed approach is very well suited for software rendering where speed is crucial.

Today, graphics hardware makes Phong shading useful for 3D applications like games and real-time visualizations. However, it is still out of reach for many types of hand held devices. It is reasonable to presume that such devices will include more and more applications using graphics. Any graphics card on such a device will be much simpler than the card used for gaming on a home computer since graphics cards are big energy consumers and energy is crucial for these types of devices. Even though some attempts to construct energy saving graphics cards have been done [Kame03], it is reasonable to presume that software rendering will be used for 3D graphics on many of these devices for a long time ahead.

As future work we are planning to investigate how X-shading would compare to Gouraud shading on hand held devices such as those on a cell phone. The small display area versus the limited processing power leads to some interesting trade-offs.

7. REFERENCES

[Abba01a] A. M. Abbas, L. Szirmay-Kalos, G. Szijarto, T. Horvath, T. Foris *Quadratic Interpolation in Hardware Rendering,* Spring Conference of Computer Graphics, 2001.

[Abba01b] A. M. Abbas, L. Szirmay-Kalos, T. Horvath, T. Foris *Quadratic Shading and its Hardware Implementation*, Machine Graphics and Vision, Vol. 9, No. 4, pp. 825-804, 2001.

[Bish86] G. Bishop, D. M. Weimer, *Fast Phong Shading* Computer Graphics, vol. 20, No 4, pp. 103-106, 1986.

[Blin77] J. F. Blinn, *Models of Light Reflection for Computer Synthesized Pictures,* In Proceedings SIGGRAPH, pp. 192-198. 1977.

[Cend87] Z. J. Cendes, S. H. Wong, C^1 *Quadratic Interpolation over Arbitrary point sets,* Computer Graphics & Applications, Vol. 7, No. 11, pp. 8-16, 1987.

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

[Fuch89] H. Fuchs, J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, L. Israel, *A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories* Computer Graphics (Proc. of SIGGRAPH '89), Vol. 23, No. 3, pp 79- 88.

[Gour71] H. Gouraud, *Continuous Shading of Curved Surfaces,* IEEE transactions on computers vol. c-20, No 6, June 1971.

[Hast01] A. Hast, T. Barrera, E. Bengtsson, *Faster Computer Graphics - by Reformulation and Simplification of Mathematical Formulas and Algorithms*, IMAGINE2001/F.E.S.T, 2001.

[Hast02a] A. Hast, T. Barrera, E. Bengtsson, *Improved Bump Mapping by using Quadratic Vector Interpolation*, Eurographics02, short paper/Poster 2002.

[Hast02b] Anders Hast, *Licentiate Thesis: Improved Fundamental Algorithms for fast Computer Graphics*, 2002.

[Hast03a] A. Hast, T. Barrera, E. Bengtsson *Fast Setup for Bilinear and Biquadratic Interpolation over Triangles,* Graphics Programming Methods, Charles River Media, pp. 299-314, 2003.

[Hast03b] A. Hast, T. Barrera, E. Bengtsson, *Fast Specular Highlights by modifying the Phong-Blinn Model,* A. Hast,T. Barrera, E. Bengtsson, Siggraph, Sketches and applications 2003.

[Kapp95] M. R. Kappel, *Shading: Fitting a Smooth Intensity Surface* Computer-Aided Design, Vol. 27, No. 8, pp. 595-603, 1995.

[Kame03] M. Kameyama, Y. Kato, H. Fujimoto, H. Negishi, Y.Kodama, Y. Inoue, H. Kawa, *3D Graphics LSI Core for Mobile Phone Z3D,* Eurographics/siggraph Graphics Hardware 2003.

[Kirk90] D. Kirk, D. Voorhies, *The Rendering Architecture of the DN10000VS* Computer Graphics vol. 24, pp. 299-307, August 1990.

[Kirk92] D. Kirk, O. Lathrop, D. Voorhies, *Quadratic Interpolation for Shaded Image Generation* Patent Nr: US5109481, 1992.

[Lee01] Y. C. Lee, C. W. Jen, *Improved Quadratic NormalVector Interpolation for Realistic Shading* The Visual Computer, 17, pp. 337-352, 2001.

[Phon75] B. T. Phong, *Illumination for Computer Generated Pictures* Communications of the ACM, Vol. 18, No 6, June 1975.

[Powe77] M. J. D. Powell, M. A. Sabin, *Piecewise Quadratic Approximations on Triangles* ACM Trans. Math. Software Vol. 3, pp 316-325, Dec. 1977.

[Saxe96] E. Saxe, A. A. Lastra, M. Hughes, *Higher-Order color Interpolation for Real-Time Radiosity Displa,y* UNCCH Dep. of Computer Science Technical Report TR96-023, 1996.

[Seil98] L. Seiler, *Quadratic Interpolation for Near-Phong Quality Shading* Proceedings of SIGGRAPH 98: conference abstracts and applications, Page 268, 1998.

[Wynn01] C. Wynn, *Implementing Bump-Mapping using Register Combiners,* Newton-Raphson fast combiner normalization technique. http://developer.nvidia.com, 2001.