Faster Computer Graphics - by Reformulation and Simplification of Mathematical Formulas and Algorithms

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

Current computer graphics research tend to be more concerned with applications than image production techniques. Most of the latter research is concerned more with global illumination models than local illumination models. Does this mean that everything that could be said about fundamental algorithms have already been said? Examples of how reformulations of mathematical formulas have led to faster algorithms especially for shading will be given in this paper. A comparison of three different techniques for bivariate shading will exemplify that better formulations of old problems are still possible and necessary.
Keywords: Algorithms, Shading

1 Introduction

Photo realism is one of the prime goals for computer animated movies today, and there is no reason to believe that computer graphics research will not provide new and better algorithms for this purpose in the future. For computer games on the other hand, speed has been a much more important issue, since they are interactive. However, as hardware becomes faster, we can expect games to include more photo realism. If fundamental algorithms can be simplified, and thus faster, then this will also give room for more sophisticated graphics in computer games. This will also have an impact on the time consuming rendering of digital films. If the time spent on rendering can be decreased, then money will be saved, and this speaks for itself.

Although there still is research on fundamental algorithms, much of the current research in computer graphics is more concerned with applications, such as animation, visualization and virtual reality. Watt [Watt00] gives an example from SIGGRAPH, comparing the number of papers concerned with production techniques of images, to the number of papers about applications. In 1985 there was a total of 22 papers concerned with the first category and 13 with the latter. Ten years later in 1995, it had turned so that there was 37 papers about applications, but only 19 on image production techniques. It seems to be a common belief that the fundamentals of computer graphics are more or less well investigated, and there is not much to do in this area anymore. At least, local illumination models do not get much attention anymore. Although, global illumination models like radiosity and Ray tracing still get a lot of attention.

It is the purpose of this paper to give some examples of how reformulations and simplifications yielded faster fundamental algorithms. One reason for the belief that much of the things needed to be done in fundamental algorithms have already been done, is maybe due to the fact that many computer graphics text books do not cover new techniques. For an example, how many computer graphics text books explains how bump mapping is used in graphics cards today? Interesting advances have been made lately by [Peercy97] and nVIDIA [Kilg00]. However, computer graphics text books will in most cases, in detail explain Blinn's [Blinn78] method from 1978. A text book intended for beginners can of course not cover all fundamental algorithms, but it is seems like the advances in this area do not reach the authors of computer graphics text books to the extent that one could expect.

2 Reformulation of Shading

One example of how an algorithm can be made faster is how the equation for the reflection vector is formulated. The reflection vector is computed as a part of the Phong [Phong75] illumination equation:

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

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 at the light source, $\mathbf{R}$ is the vector in the direction of the perfect reflection from the light source, $\mathbf{V}$ is the vector

Figure 1: The Venus de Milo statue, hybrid shaded

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

pointing at the viewer, and finally n is the shininess value which affects the size of the highlighted area.

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

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

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]. However, they use the form where $\mathbf{N'}$ have unit length. Although, it has been noticed by some [Voorh94], [Hast01a], [Hill01] that this equation can be formulated in another way:

$$\mathbf{R} = 2\mathbf{N}\frac{\mathbf{N} \cdot \mathbf{L}}{\mathbf{N} \cdot \mathbf{N}} - \mathbf{L}, \tag{3}$$

where $\mathbf{N}$ has arbitrary non zero length. This did not make the equation shorter, instead it made it longer which is probably the reason why it is seldom mentioned. However, there are some interesting conclusions, that could be drawn from this result. First of all it is possible to make the computation of Phong's formulation of the specular light, faster for a parallel light source and the viewer placed at infinity. This is a simplification often used for computer games. Thus, the vector in the direction of the viewer $\mathbf{V} = (0, 0, -1)$ and the vector to the light source is $\mathbf{L}$. Then:

$$\mathbf{R} \cdot \mathbf{V} = L_z - 2N_z\frac{\mathbf{N} \cdot \mathbf{L}}{\mathbf{N} \cdot \mathbf{N}}. \tag{4}$$

It is shown by [Hast01a] how this equation could be simplified even further so it could compete with Blinn's [Blinn77] formulation of the specular light. Nonetheless, this formulation shows that it is possible to compute the specular light without having a normalized normal. For what purpose could this be used? It could be used for a hybrid type of shading, where the diffuse light is computed by Gouraud shading [Goura71] which is based on bilinear interpolation of the colors at the vertices, and the specular light is computed by using equation (4). 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. Fig.2 shows the Venus de Milo statue with Phong shading for both diffuse and specular light. The total cost of computing the intensity for the latter is 4 additions, 1 subtraction, 3 multiplications, 1 division and 1 square root. This can be compared with the cost of computing the intensity for Hybrid shading which is 4 additions and 1 division per pixel for the specular light and 1 addition for linearly interpolating the intensity of the diffuse light.

Fig.3 shows the space station Mir, with another hybrid shading approach. Gouraud shading is used for the diffuse light and bivariate quadratic interpolation is used for the highlights. The textures are super sampled. Bivariate quadratic shading is explained in the next section.

Figure 3: The space station Mir with Gouraud shading, super sampled textures and quadratic interpolation of highlights

It is also shown in [Hast01a] that equation (3) could be used in order to obtain an approximation of a normalized normal. Why this is important, is because the normalization process is computationally costly, not at least depending on the square root involved. Another approximation has been proposed by Cutler as shown by Wynn [Wynn]:

$$\mathbf{N}' \approx \frac{\mathbf{N}}{2}(3 - \mathbf{N} \cdot \mathbf{N}) \tag{5}$$

If this approximation is used for a shading algorithm then the angles between the normals at the edges should not be larger than 40 degrees. Note that, there is no square root in the equation, and the division can be substituted by a multiplication by 0.5. This will make this approximation very fast, since divisions are much more computationally costly than multiplications.

Another example of a reformulation of a problem is the method proposed by Ouyang and Maynard [Ouyan96]. They take advantage of the fact that some part of the computation is the same over the scan line, if the scan line has a length that is a power of two. This implies that the scan line has to be extrapolated in order to use this faster formulation. The number of square roots are heavily reduced, by using this method.

Hast et al. [Hast01b] showed that it is also possible to reduce the number of square roots by exploiting the fact that the normalization function is symmetric over the scan line. This is true only if the normals on the edges are normalized. Hence, the normalization function has only to be computed for the first half of the scan line and can then be reused for the second half. Duff [Duff79] improved phong shading by reformulating the computation of a dot product by:

$$\frac{\mathbf{N} \cdot \mathbf{L}}{\| \mathbf{N} \|} = \frac{Ax + B}{\sqrt{Cx^2 + Dx + E}} = \frac{p}{\sqrt{q}}. \tag{6}$$

Here, $\sqrt{q}$ is the normalization function and it is shown in [Hast01b] that $q = 1$ on the edges and the derivative at the edges have the relation $dq_a = -dq_b$. Hence, q must be symmetric, and therefore the number of square roots is reduced to the half. It is also shown that the dot product itself could be approximated by a quadratic function.

3 Faster Bivariate Quadratic Shading

The shading methods mentioned so far, use a scan line set up which is relatively computationally expensive. Phong shading can also use a polygon set up which is costly, on the other hand, the gain is a much more efficient scan line set up. Bishop and Weimer [Bishop86] use a Taylor series expansion of Duff's equation to obtain a second order bivariate polynomial. It could be evaluated by only two additions per pixel along a scan line and only three additions and two moves per scan line. The draw back is a large set up for the polygon. Other quadratic

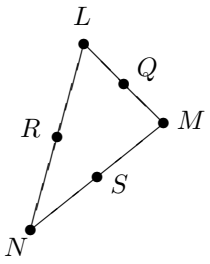

Figure 4: Sample points

shading approaches have been proposed by Kappel [Kapp95] and Kirk et al. [Kirk92], who try to diminish the mach band effect.

Kirk and Voorhies [Kirk90] shows how quadratic interpolation could be set up by fitting a second order surface to six sample points, yielding a polynomial with six unknown coefficients which must be determined. They do not show how these coefficients are computed, neither do Saxe et al. [Saxe96] who also use this type of quadratic interpolation.

The sample points are shown in Fig 4. A quadratic shading function is defined by:

$$\Phi = Ax^2 + By^2 + Cxy + Dx + Ey + F \tag{7}$$

Seiler [Seiler98] proposes a simple and fast way to set up the coefficients for this type of quadratic shading. This is also done by Abbas et al. [Abbas01a], [Abbas01b] but they do not try to make the computation as efficient as Seiler does. These methods have one major disadvantage, which is a special case where division by zero will occur. However, this problem, is solved by Lee and Jen [Lee01] who have simplified Seiler's approach.

The following sub sections shows how these coefficients can be computed in different ways. Thus showing, that one solution of a problem is not necessary the best one. It is also shown that complicated equations sometimes can be simplified by some simple substitutions, as in the case of the method by Abbas et al.

3.1 Seiler's Method

None of the mentioned papers state how the proposed computation is derived, but 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:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2y_2 & x_2 & y_2 & 1 \\ \left(\frac{x_1}{2}\right)^2 & \left(\frac{y_1}{2}\right)^2 & \frac{x_1}{2}\frac{y_1}{2} & \frac{x_1}{2} & \frac{y_1}{2} & 1 \\ \left(\frac{x_2}{2}\right)^2 & \left(\frac{y_2}{2}\right)^2 & \frac{x_2}{2}\frac{y_2}{2} & \frac{x_2}{2} & \frac{y_2}{2} & 1 \\ \left(\frac{x_1+x_2}{2}\right)^2 & \left(\frac{y_1+y_2}{2}\right)^2 & \frac{x_1+x_2}{2}\frac{y_1+y_2}{2} & \frac{x_1+x_2}{2} & \frac{y_1+y_2}{2} & 1 \end{bmatrix} \tag{8}$$

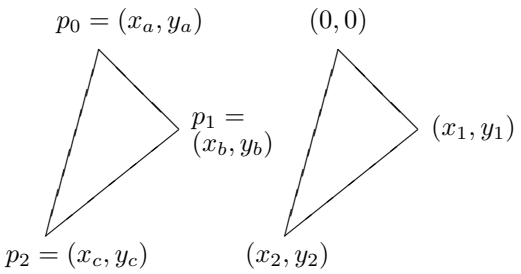

Figure 5: A polygon with sorted vertices at the left and the same polygon with shifted vertices at the right

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). This implies that the vertices has 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 transformation is shown in Fig. 5.

The following substitutions is used in order to shift the triangle and to obtain the relative coordinates:

$$x_1 = x_b - x_a, \tag{9}$$
$$y_1 = y_b - y_a, \tag{10}$$
$$x_2 = x_c - x_a, \tag{11}$$
$$y_2 = y_c - y_a. \tag{12}$$

These relative coordinates was also used in the subsequent sections for both the method by Abbas et al. and Lee and Jen.

Finally, the system of equations to solve is:

$$M \cdot K = \Phi, \tag{13}$$

where

$$\Phi = [L, M, N, Q, R, S]^T, \tag{14}$$
$$K = [A, B, C, D, E, F]^T. \tag{15}$$

In order to speed up the computation of coefficients the following intermediate values are used by Seiler:

$$T = L + M - 2Q, \tag{16}$$
$$U = L + N - 2R, \tag{17}$$
$$V = L + S - R - Q, \tag{18}$$
$$G = 2Q - 2L - T, \tag{19}$$
$$H = 2R - 2L - U, \tag{20}$$
$$i = x_2/x_1, \tag{21}$$
$$j = y_1/y_2, \tag{22}$$
$$k = 1 - ij. \tag{23}$$

Then the coefficients are computed as follows:

$$A = 2(T + Uj^2 - 2Vj)/(k^2 x_1 x_1), \tag{24}$$
$$B = 2(U + Ti^2 - 2Vi)/(k^2 y_2 y_2), \tag{25}$$
$$C = 4(2V - Ti - Uj - Vk)/(k^2 x_1 y_2), \tag{26}$$
$$D = (G - Hj)/(kx_1), \tag{27}$$
$$E = (H - Gi)/(ky_2), \tag{28}$$
$$F = L. \tag{29}$$

It should be pointed out that x_1 could be zero. Then, the correct solution for that case must be used in order to compute the coefficients correctly. Also, note that it is possible to break out the constant 2 from equation 19 and equation 20, which will save two multiplications. Nothing else was done in order to simplify Seiler's method.

3.2 The method by Abbas et al.

If the same notation used by Seiler for the vertices is also used for the equations proposed by Abbas et. el. then their equations becomes:

$$T = M + L - 2Q, \tag{30}$$
$$U = N + L - 2R, \tag{31}$$
$$V = 4(L + S - Q - R), \tag{32}$$
$$G = 4Q - 3L - M, \tag{33}$$
$$H = 4R - 3L - N. \tag{34}$$

Note that if (16) is substituted into (19) and (17) is substituted into (20) as proposed by Seiler, then (33) and (34) are obtained. This is computationally more efficient, and was therefore used later in the comparison of methods. The coefficients according to Abbas et al. are:

$$A = \frac{(4Ty_2 - Vy_1)E_{32} - (4Uy_1 - Vy_2)E_{23}}{E_{32}D_{23} - E_{32}D_{32}}, \tag{35}$$
$$C = \frac{(4Uy_1 - Vy_2)D_{23} - (4Ty_2 - Vy_1)D_{32}}{E_{32}D_{23} - E_{32}D_{32}}, \tag{36}$$
$$B = \frac{2T - Ax_1^2 - Cx_1y_1}{y_1^2}, \tag{37}$$
$$D = \frac{Gy_2 - Hy_1}{x_1y_2 - y_1x_2}, \tag{38}$$
$$E = \frac{Hx_1 - Gx_2}{x_1y_2 - y_1x_2}, \tag{39}$$
$$F = L, \tag{40}$$

where they use the following notation for:

$$E_{23} = x_1y_1y_2 - y_1^2x_2, \tag{41}$$
$$E_{32} = y_1x_2y_2 - x_1y_2^2, \tag{42}$$
$$D_{23} = 2x_1^2y_2 - 2x_1y_1x_2, \tag{43}$$
$$D_{32} = 2y_1x_2^2 - 2x_1x_2y_2. \tag{44}$$

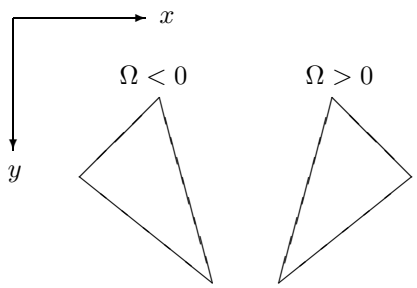

Figure 6: A left and right oriented triangle

Some simplifications are possible. First note, that the denominators in (35) and (36) are equal, and the denominators in (38) and (39) are equal as well. Let:

$$F_1 = \frac{1}{x_1 y_2 - x_2 y_1}, \tag{45}$$

$$F_2 = \frac{1}{E_{32} D_{23} - E_{32} D_{32}}. \tag{46}$$

Then it can be shown that $F_2 = -1/2F_1^3$. Also note, that the factors $(4Ty_2 - Vy_1)$ and $(4Uy_1 - Vy_2)$ in (35) also appears in (36). There are, however, still many multiplications that could be saved by the following substitution:

$$\Omega = x_1 y_2 - x_2 y_1. \tag{47}$$

This equation has got an interesting interpretation. It is essential to determine the orientation of the triangle in order to be able to rasterize it properly. The orientation of a polygon could be determined by computing the cross product. A better way to do it, is to compute the double area of the polygon:

$$\Omega = (x_a - x_c)(y_b - y_c) - (y_a - y_c)(x_b - x_c). \tag{48}$$

Using the substitutions in (9) through (12) yields equation (47). Hence, the sign of Ω will tell how the triangle is oriented as shown in Fig. 6. If $\Omega = 0$ then the vertices are on a straight line, and the area is equal to zero.

The equations for the coefficients, can by using this substitution, be rewritten in the following simplified form:

$$A = (C_1 E_{32} - C_2 E_{23}) F_2 \tag{49}$$

$$C = (C_2 D_{23} - C_1 D_{32}) F_2 \tag{50}$$

$$B = (2T - Ax_1^2 - Cx_1 y_1)/y_1^2, \tag{51}$$

$$D = (Gy_2 - Hy_1) F_1, \tag{52}$$

$$E = (Hx_1 - Gx_2) F_1, \tag{53}$$

$$F = L, \tag{54}$$

where:

$$C_1 = 4Ty_2 - Vy_1, \tag{55}$$
$$C_2 = 4Uy_1 - Vy_2, \tag{56}$$
$$D_{23} = 2x_1\Omega, \tag{57}$$
$$D_{32} = -2x_2\Omega, \tag{58}$$
$$E_{23} = y_1\Omega, \tag{59}$$
$$E_{32} = -y_2\Omega, \tag{60}$$
$$F_1 = \frac{1}{\Omega}, \tag{61}$$
$$F_2 = -1/2F_1^3. \tag{62}$$

Also here it should be noted that this method also suffers from the problem of division by zero when $y_1 = 0$.

3.3 The method by Lee and Jen

Lee and Jen has modified Seiler's method slightly by modifying the denominators, and the problem of division by zero is hereby removed. The intermediate terms are as follows:

$$w_1 = x_1y_2, \tag{63}$$
$$w_2 = x_2y_1, \tag{64}$$
$$w_3 = w_1 - w_2, \tag{65}$$
$$T = L + M - 2Q, \tag{66}$$
$$U = L + N - 2R, \tag{67}$$
$$V = L + S - R - Q, \tag{68}$$
$$G = 2Q - 2L - T, \tag{69}$$
$$H = 2R - 2L - U. \tag{70}$$

Note that, $w_3 = \Omega$. Once again, the constant term in the last two equations should only appear once in each equation, in order to save multiplications. The coefficients are then computed as follows:

$$A = 2(Ty_2^2 + Uy_1^2 - 2Vy_1y_2)/w_3^2, \tag{71}$$
$$B = 2(Ux_1^2 + Tx_2^2 - 2Vx_1x_2)/w_3^2, \tag{72}$$
$$C = 4(V(w_1 + w_2) - Ux_1y_1 - Tx_2y_2)/w_3^2, \tag{73}$$
$$D = (Gy_2 - Hy_1)/w_3, \tag{74}$$
$$E = (Hx_1 - Gx_2)/w_3, \tag{75}$$
$$F = L. \tag{76}$$

Besides breaking out the constant, the only simplifications made in the comparison was that the divisors $1/w_3$ and $1/w_3^2$ were pre computed in order to save divisions.

Table 1: Timing of one million triangles, comparing the computation of the coefficients

Seiler	Abbas et al.	Lee and Jen
9.17	8.24	7.24

3.4 Comparison

Table 1 shows the time for the computation of the coefficients of equation (7) for one million triangles. It took about eight years from what seems to be the first appearance of quadratic shading by Kirk et al. until Seiler explained how the coefficients actually can be computed. Another three years passed until Abbas et al. came with the same idea, and finally Lee and Jen came with the best and fastest method.

4 Conclusions

It has been shown that a reformulation of an equation or an approximation can give faster algorithms. There is still research in fundamental algorithms that gives interesting and useful results. Some of the examples from other papers were that it is possible to compute accurate specular reflection for Gouraud shading, without having to normalize the normal. There also exists methods for computing approximations for the normalization of vectors under certain conditions. Another useful fact is that the normalization function for linear interpolation between two normalized normals is symmetric. This will almost reduce the computations to half.

Three different methods for computing the coefficients of a bivariate polynomial, that is used as a shading, was investigated. It was shown that sometimes it is possible to make simplifications that are not that apparent at first sight. The method by Abbas et al. were a lot more computationally costly, than the method by Seiler, but after simplification it became faster. The method by Lee and Jen is a simplification of Seiler's method and it was not only faster than the other two methods, it was also better in the sense that it did not suffer from the problem of division by zero for certain special cases.

References

[Abbas01a] A. M. Abbas, L. Szirmay-Kalos, G. Szijarto, T. Horvath, T. Foris *Quadratic Interpolation in Hardware Rendering* Spring Conference of Computer Graphics, 2001.

[Abbas01b] 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.

[Bishop86] G. Bishop, D. M. Weimer, *Fast Phong Shading* Computer Graphics, vol. 20, No 4, pp. 103-106, 1986.

[Blinn77] J. F. Blinn, *Models of Light Reflection for Computer Synthesized Pictures*, In Proceedings SIGGRAPH, pp. 192-198, 1977.

[Blinn78] J. F. Blinn, *Simulation of Wrinkled Surfaces*, In Proceedings SIGGRAPH 78, pp. 286-292, 1978.

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

[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.

[Hast01a] A. Hast, T. Barrera, E. Bengtsson, *Improved Shading Performance by avoiding Vector Normalization*, WSCG'01, short paper 1-8. 2001.

[Hast01b] A. Hast, T. Barrera, E. Bengtsson, *Approximated Phong Shading by using the Euler Method*, Eurographics01, short paper pp. 43-48, 2001.

[Hill01] F. S. Hill, JR. *Computer Graphics using OpenGL* Prentice-Hall pp. 417, 2001.

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

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

[Kilg00] M. J. Kilgard *A Practical and Robust Bump-mapping Technique for Today s GPUs*, Game Developers Conference, Advanced OpenGL Game Development, 2000.

[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 Normal Vector Interpolation for Realistic Shading* The Visual Computer, 17, pp. 337-352, 2001.

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

[Peercy97] M. Peercy, A. Airey, B. Cabral, *Efficient Bump Mapping Hardware*, In proceedings of SIGGRAPH 97, August 3-8, pp. 303-306, 1997.

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

[Saxe96] E. Saxe, A. A. Lastra, M. Hughes, *Higher-Order color Interpolation for Real-Time Radiosity Display* UNC-CH Department of Computer Science Technical Report TR96-023, 1996.

[Seiler98] L. Seiler, *Quadratic Interpolation for Near-Phong Quality Shading* Proceedings of the conference on SIGGRAPH 98: conference abstracts and applications, Page 268, 1998.

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

[Watt00] A. Watt, *3D Computer Graphics,* Addison-Wesley, Third edition, pp. xviii, 2000.

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