

Modeling Specular Highlights Using Bézier Triangles

Russ Brown

© 1999 Sun Microsystems, Inc. All rights reserved. The SML Technical Report Series is published by Sun Microsystems Laboratories, of Sun Microsystems, Inc. Printed in U.S.A.

Unlimited copying without fee is permitted provided that the copies are not made nor distributed for direct commercial advantage, and credit to the source is given. Otherwise, no part of this work covered by copyright hereon may be reproduced in any form or by any means graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

TRADEMARKS

Sun, Sun Microsystems, and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

For information regarding the SML Technical Report Series, contact Jeanie Treichel, Editor-in-Chief <jeanie.treichel@eng.sun.com>.

Modeling Specular Highlights Using Bézier Triangles

Russ Brown

SMLI TR-99-75

September 1999

Abstract:

Rendering specular highlights typically requires interpolation and renormalization of a unit surface normal vector at each pixel of a three-dimensional graphics image. This report discusses a technique for modeling specular highlights using Bézier triangles. This approach avoids computation by identifying those polygons for which it is not necessary to calculate a specular highlight. The use of Bézier triangles to model specular highlights suggests a general technique for modeling, clipping and interpolation of shading functions.



M/S MTV29-01
901 San Antonio Road
Palo Alto, CA 94303-4900

email address:
russ.brown@sun.com

Modeling Specular Highlights Using Bézier Triangles

Russ Brown

Sun Microsystems Laboratories
901 San Antonio Road
Palo Alto, CA 94303

Introduction

Gouraud shading was developed to enable piecewise linear interpolation of curved surfaces [1]; however, linear interpolation is not appropriate for some shading functions. For example, linear interpolation of specular highlights does not produce realistic results. This report discusses polynomial interpolation of specular highlights that produces realistic results for only a modest increase in computational complexity relative to linear interpolation.

Phong [2] proposed calculating specular highlights proportional to $\cos^n \theta = (\mathbf{V} \cdot \mathbf{R})^n$ (see Figure 1). In this formulation, \mathbf{V} represents a vector in the viewing direction and \mathbf{R} represents a reflected vector calculated by reflecting the light source vector \mathbf{L} about a surface normal vector \mathbf{N} .

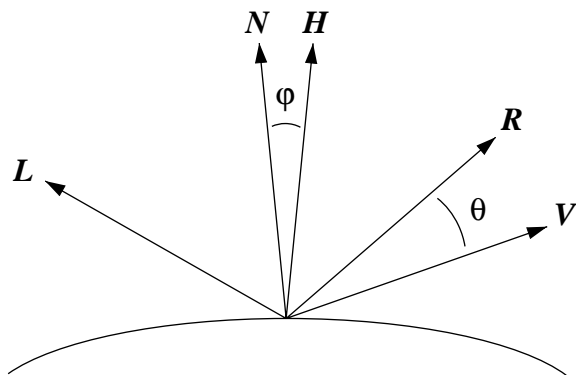


Figure 1: The normal vector is interpolated

The $\cos \theta$ function is raised to an arbitrary power of n so as to narrow this function in order that specular highlights occur only for small values of θ . A significant computational cost of Phong shading is interpolating the normal \mathbf{N} at each pixel because normalizing \mathbf{N} requires calculating a reciprocal square root. In addition, if the viewer and light source lie at finite distances from the shaded surface, it is also necessary to compute and normalize \mathbf{L} and \mathbf{V} at each pixel, which requires calculating

two more reciprocal square roots. Raising $\cos \theta$ to the n th power can also involve considerable computation expense.

Because of these and other computational difficulties, several investigators have proposed various ways to reduce the calculation required to model specular highlights. Blinn [3] computed specular highlights proportional to $\cos^n \phi = (\mathbf{N} \cdot \mathbf{H})^n$, where \mathbf{H} represents a vector in the direction of maximum highlights computed as the normalized bisector of vectors \mathbf{L} and \mathbf{V}

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \quad (1)$$

This approach avoids the difficulties involved in calculating the reflected vector \mathbf{R} , but requires normalization of \mathbf{H} at each pixel in cases where the viewer and light source lie at finite distances from the shaded surface. Duff [4] employed forward differencing to evaluate the Phong shading equation. This technique simplifies the shading calculations but does not eliminate vector normalization at each pixel. Bishop and Weimer [5] used a two-dimensional Taylor expansion to approximate the Phong shading equation and thereby eliminated normalization at each pixel.

Shantz and Lien [6] eliminated normalization by interpolating $\cos \phi$ using a bicubic parametric surface. Kirk and Voorhies [7] eliminated normalization by interpolating $\cos \phi$ quadratically via forward differences. Seiler [8] has recently suggested a similar approach. Kuijk and Blake [9] eliminated normalization by linearly interpolating the angle ϕ between \mathbf{N} and \mathbf{H} , then by calculating $\cos^n \phi$ via piecewise quadratic approximation.

Bergman et al., [10] limited Phong shading calculations to those polygons on which a specular highlight is predicted to occur. The prediction involved comparing the direction of the vector of maximum highlights \mathbf{H} to the direction of a highlight vector computed at each polygon vertex. This comparison failed to detect cases where a

specular highlight occurred on the polygon at places other than a vertex. Harrison et al., [11] proposed a more general technique that tested whether the highlight function $N \cdot H$ would exceed a threshold value: (1) at a polygon vertex, (2) along a polygon edge, or (3) on the interior of the polygon. This test failed to detect some cases of a specular highlight on the interior of the polygon.

These approaches suggest the following approach to reducing the computational complexity of calculating specular highlights: (1) limiting the calculation of specular highlights to those polygons on which the highlights are predicted to occur; (2) calculating the specular highlights in a manner that avoids normalizing vectors at each pixel; and (3) approximating the power function with a simpler function. Requirements 1 and 2 may be met by interpolating a highlight function using a quadratic Bézier triangle [12]. Requirement 3 may be met by using a cubic polynomial in lieu of a power function.

Quadratic Bézier Triangle

Figure 2 illustrates a quadratic Bézier triangle. As its name suggests, it has triangular topology and quadratic

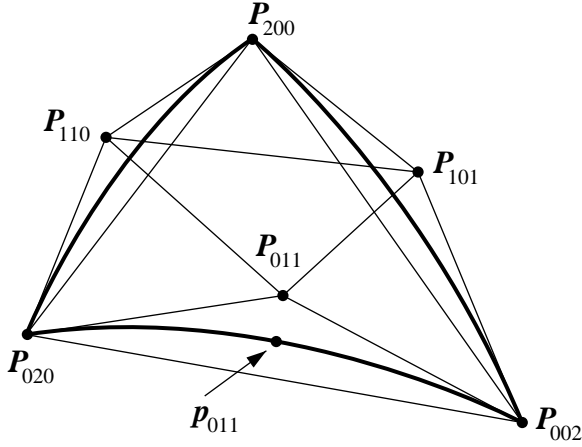


Figure 2: A quadratic Bézier triangle

geometry. Each of its three curved edges $P_{200}P_{020}$, $P_{020}P_{002}$ and $P_{002}P_{200}$ is a quadratic Bézier curve. These three curves enclose a quadratic surface. This parametric surface may be evaluated via the vector-valued function

$$p(b_0, b_1, b_2) = P_{200}b_0^2 + P_{020}b_1^2 + P_{002}b_2^2 + 2P_{110}b_0b_1 + 2P_{011}b_1b_2 + 2P_{101}b_2b_0 \quad (2)$$

to produce a point p on the surface, where P_{200} , P_{020} , P_{002} , P_{110} , P_{011} and P_{101} are the control points for the surface,

and b_0 , b_1 and b_2 are a set of interpolants known as barycentric coordinates [13]. The point p_{011} is calculated in this manner.

An important property of the Bézier triangle is known as the convex hull property [14]. This property guarantees the parametric surface to be enclosed by the convex polyhedron defined by the control points. In Figure 2, the convex hull is constructed by connecting the six control points with line segments.

Barycentric Coordinates

Figure 3 depicts the planar triangle $P_{200}P_{020}P_{002}$ from Figure 2 where points P_{200} , P_{020} and P_{002} have been relabeled P_0 , P_1 and P_2 , respectively, in order to underscore its planar nature. Triangle $P_{200}P_{020}P_{002}$ is a

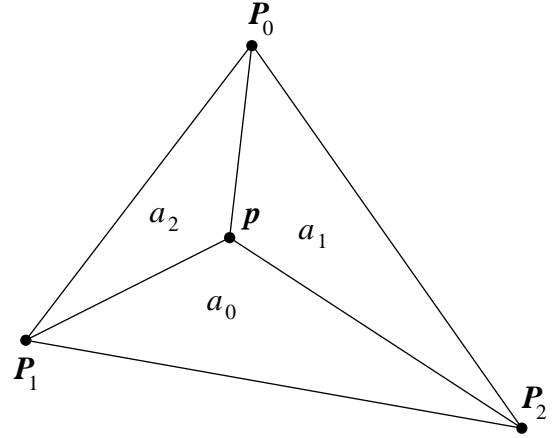


Figure 3: Barycentric coordinates

linear Bézier triangle. Its three linear edges enclose a planar surface which may be evaluated via the vector-valued function

$$p(b_0, b_1, b_2) = P_0b_0 + P_1b_1 + P_2b_2 \quad (3)$$

The importance of eq. 3 extends beyond the calculation of the point p via linear interpolation of points P_0 , P_1 and P_2 using the barycentric coordinates (b_0, b_1, b_2) . This equation also provides a simple method for calculating the barycentric coordinates from the cartesian coordinates of points p , P_0 , P_1 and P_2 . For example, assume that the cartesian (x, y) coordinates of points P_0 , P_1 and P_2 are defined in screen space. Further assume that the (x, y) coordinates of the point p define a pixel at which interpolation is required to accomplish shading

calculations. It is possible to solve for the barycentric coordinates (b_0, b_1, b_2) of the point \mathbf{p} by constructing three simultaneous linear equations. The first equation is simply $\sum b_i = 1$ which expresses the fact that the sum of the barycentric coordinates must equal one. The two other equations are generated from eq. 3 by explicitly writing the linear interpolation that computes the (x, y) coordinates of the point \mathbf{p} from the (x, y) coordinates of points $\mathbf{P}_0, \mathbf{P}_1$ and \mathbf{P}_2 . Then the three simultaneous linear equations are

$$\begin{aligned} b_0 + b_1 + b_2 &= 1 \\ x_0 b_0 + x_1 b_1 + x_2 b_2 &= x_p \\ y_0 b_0 + y_1 b_1 + y_2 b_2 &= y_p \end{aligned} \quad (4)$$

Solving these three equations for b_0, b_1 and b_2 yields the barycentric coordinates of the point \mathbf{p}

$$\begin{aligned} b_0 &= \frac{a_0}{a_0 + a_1 + a_2} \\ b_1 &= \frac{a_1}{a_0 + a_1 + a_2} \\ b_2 &= \frac{a_2}{a_0 + a_1 + a_2} \end{aligned} \quad (5)$$

where a_0, a_1 and a_2 are the screen-space areas of triangles $\mathbf{pP}_1\mathbf{P}_2, \mathbf{pP}_2\mathbf{P}_0$ and $\mathbf{pP}_0\mathbf{P}_1$, respectively, from Figure 3 [13].

Once the barycentric coordinates have been calculated using eq. 5, they may be employed to interpolate other coordinates of the point \mathbf{p} , such as the color coordinates (r, g, b) or texture coordinates (u, v) . This interpolation is accomplished via substitution of the barycentric coordinates into eq. 3 to perform linear interpolation, or into eq. 2 to perform quadratic interpolation. However, because the perspective projection introduces a nonlinear distortion into the screen coordinates, it is necessary that the interpolation correct for this distortion. Interpolation that corrects for this distortion has been described previously [15], and is known as *rational linear interpolation* [16] or *hyperbolic interpolation* [17].

Hyperbolic Interpolation

Hyperbolic interpolation requires the screen-space w coordinates of points $\mathbf{P}_0, \mathbf{P}_1$ and \mathbf{P}_2 . (Screen-space w is proportional to eye-space z , as discussed in [18].) The

hyperbolic interpolation of texture coordinates (u, v) will be considered in order to illustrate this form of interpolation. It is accomplished via three operations: (1) projection of a homogeneous texture vector $(u, v, 1)$, defined at each vertex, from eye space into screen space via division by w to form $(u/w, v/w, 1/w)$; (2) interpolation of $(u/w), (v/w)$ and $(1/w)$ in screen space using eq. 3; and (3) projection of the interpolated vector $(u/w, v/w, 1/w)$ back into eye space via division by $(1/w)$ to recover $(u, v, 1)$. For the u -coordinate, these three operations may be represented as the ratio of two linear interpolations

$$u(b_0, b_1, b_2) = \frac{\left(\frac{u_0}{w_0}\right)b_0 + \left(\frac{u_1}{w_1}\right)b_1 + \left(\frac{u_2}{w_2}\right)b_2}{\left(\frac{1}{w_0}\right)b_0 + \left(\frac{1}{w_1}\right)b_1 + \left(\frac{1}{w_2}\right)b_2} \quad (6)$$

A useful relation may be derived from eq. 6. The first step of this derivation is multiplication of the right-hand side of eq. 6 by $w_0 w_1 w_2 / w_0 w_1 w_2$ to yield

$$u(b_0, b_1, b_2) = \frac{u_0 w_1 w_2 b_0 + u_1 w_2 w_0 b_1 + u_2 w_0 w_1 b_2}{w_1 w_2 b_0 + w_2 w_0 b_1 + w_0 w_1 b_2} \quad (7)$$

Substituting the definitions of b_0, b_1 and b_2 from eq. 5 into eq. 7 and eliminating $a_0 + a_1 + a_2$ gives

$$u(b_0, b_1, b_2) = \frac{u_0 w_1 w_2 a_0 + u_1 w_2 w_0 a_1 + u_2 w_0 w_1 a_2}{w_1 w_2 a_0 + w_2 w_0 a_1 + w_0 w_1 a_2} \quad (8)$$

Equating the coefficients of u_0, u_1 and u_2 in eq. 8 to the coefficients of $\mathbf{P}_0, \mathbf{P}_1$ and \mathbf{P}_2 , respectively, in eq. 3 yields the following definitions for the barycentric coordinates

$$\begin{aligned} b_0 &= \frac{w_1 w_2 a_0}{w_1 w_2 a_0 + w_2 w_0 a_1 + w_0 w_1 a_2} \\ b_1 &= \frac{w_2 w_0 a_1}{w_1 w_2 a_0 + w_2 w_0 a_1 + w_0 w_1 a_2} \\ b_2 &= \frac{w_0 w_1 a_2}{w_1 w_2 a_0 + w_2 w_0 a_1 + w_0 w_1 a_2} \end{aligned} \quad (9)$$

It is possible to derive eq. 9 by projecting the barycentric coordinates from screen space into eye space [19].

Although eq. 6 expresses hyperbolic interpolation in terms of a rational interpolation formula, eq. 9 expresses

hyperbolic interpolation in terms of barycentric coordinate interpolants. The substitution of these barycentric coordinates into eq. 3 or eq. 2 accomplishes linear or quadratic interpolation, respectively, while correcting for the nonlinear distortion introduced by the perspective projection. Linear interpolation may be used to interpolate color coordinates (r,g,b) , whereas quadratic interpolation is useful for interpolating a highlight function.

Highlight Function

Earlier research into the interpolation of a highlight function [6],[7] suggests that an appropriate highlight function ought to be $\cos\phi = \mathbf{N} \cdot \mathbf{H}$. This highlight function may be approximated using a quadratic Bézier surface. The surface is defined by calculating values for the six control points C_{200} , C_{020} , C_{002} , C_{110} , C_{011} and C_{101} , which are analogous to the control points P_{200} , P_{020} , P_{002} , P_{110} , P_{011} and P_{101} of Figure 2, except that C_{200} , C_{020} , C_{002} , C_{110} , C_{011} and C_{101} are scalars instead of vectors because they represent cosines.

The first step of calculating these six cosines is the creation of the six unit normal vectors N_{200} , N_{020} , N_{002} , N_{110} , N_{011} and N_{101} , as shown in Figure 4.

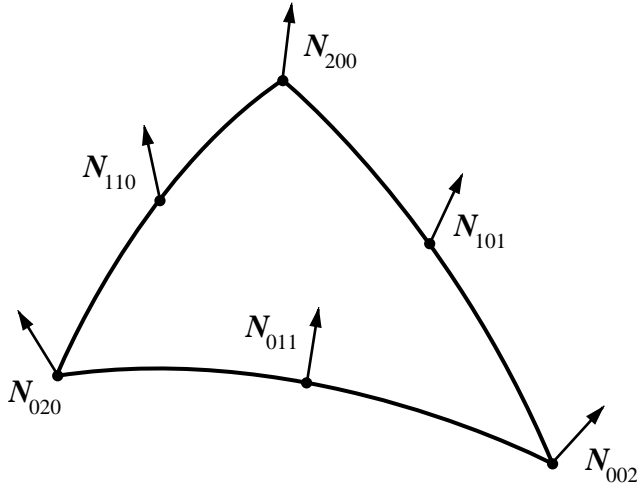


Figure 4: Normal vectors

Three of these vectors, N_{200} , N_{020} and N_{002} , are simply the unit normal vectors associated with the three vertices of a triangle. Linear interpolation between these normal vectors, followed by renormalization, creates the other three unit normal vectors, N_{110} , N_{011} and N_{101} , associated with the midpoints of the triangle edges. For example, N_{011} is calculated as

$$N_{011} = \frac{N_{020} + N_{002}}{|N_{020} + N_{002}|} \quad (10)$$

The other two vectors, N_{110} and N_{101} , are created in an analogous manner.

The next step of calculating the six cosines is the computation of dot products between the vector of maximum highlights \mathbf{H} and the six unit normal vectors N_{200} , N_{020} , N_{002} , N_{110} , N_{011} and N_{101} . If the viewer and light source lie at an infinite distance from the shaded surface, the vector \mathbf{H} may be assumed to be constant. However, if the viewer and light source lie at finite distances from the shaded surface, it is necessary to create a different vector \mathbf{H} for each of the six unit normal vectors. Independent of whether one or six vectors \mathbf{H} are required, a dot product $\mathbf{N} \cdot \mathbf{H}$ is calculated using each of the six unit normal vectors. These six dot products create the six cosines c_{200} , c_{020} , c_{002} , c_{110} , c_{011} and c_{101} . These cosines are not control points, but rather points lying on the quadratic surface (see, for example, the point p_{011} of Figure 2.)

Creation of the control points from these cosines is straightforward. The vertex control points C_{200} , C_{020} and C_{002} equal the vertex cosines c_{200} , c_{020} and c_{002} , respectively. The other three control points C_{110} , C_{011} and C_{101} are created from the subdivision formula for a quadratic Bézier triangle. For example, the following expression results when the barycentric coordinates $(0, 1/2, 1/2)$ are substituted into eq. 2

$$c_{011} = \frac{C_{020} + 2C_{011} + C_{002}}{4} \quad (11)$$

Solving this expression for C_{011} yields

$$C_{011} = \frac{4c_{011} - C_{020} - C_{002}}{2} \quad (12)$$

The other two control points, C_{110} and C_{101} , are created in an analogous manner.

The six control points C_{200} , C_{020} , C_{002} , C_{110} , C_{011} and C_{101} , computed as described above, define a quadratic highlight function useful for calculating specular highlights at each pixel within a triangle. The first step of calculating a specular highlight is to evaluate the quadratic highlight function via eq. 2 to produce a cosine c . The next step of calculating a specular highlight has historically been to raise this cosine to an arbitrary power of n [2],[3]. Raising

this cosine to an arbitrary power of n serves to narrow the highlight function such that specular highlights occur only for small angles. A similar narrowing effect may be achieved more easily by using the cosine c to evaluate a cubic polynomial. The cosine is first reparameterized so that specular highlights occur only for cosines lying in the narrow range $c_{lo} \leq c \leq c_{hi}$ (typically, $c_{hi} = \cos(0) = 1$ and $c_{lo} = \cos(\varphi_{lo})$, where φ_{lo} is the largest angle for which a specular reflection is visible.) This reparameterization is calculated as

$$t = \frac{c - c_{lo}}{c_{hi} - c_{lo}} \quad (13)$$

and produces the parameter t in the interval $0 \leq t \leq 1$ for values of c in the interval $c_{lo} \leq c \leq c_{hi}$. These values of t are used in the cubic polynomial

$$s = 3t^2 - 2t^3 = t^2(3 - 2t) \quad (14)$$

in order to calculate the specular parameter s . The combination of eq. 13 and eq. 14 produces a highlight function that transitions smoothly from 0 to 1 for values of the cosine c lying in the interval $c_{lo} \leq c \leq c_{hi}$.

The final step in the creation of a specular highlight is to sum the (r, g, b) components of the specular, diffuse and ambient colors. One approach to this summation is to use the specular parameter s as a linear interpolant to blend between the specular color and the combined diffuse and ambient colors. For example, if g_s represents the green component of the specular color, and g_{da} represents the green component of the combined diffuse and ambient colors, then the blended green component g is

$$g = sg_s + (1 - s)g_{da} \quad (15)$$

Figure 5a shows an example of a specular highlight blended in this manner. This highlight is contained within the edges of a single triangular polygon whose limits extend beyond the rectangular region displayed.

Precision Requirements

It is possible to predict the bit precision p required for interpolation using eq. 2. This precision is determined in part by the number of bits b necessary to represent the interval $c_{lo} \dots c_{hi}$. When $b < 6$, noise becomes visually

apparent. For example, Figure 5b calculated with $b = 5$ demonstrates noise at the margins of the oval specular reflection. In contrast, Figure 5a calculated with $b = 6$ does not demonstrate this noise.

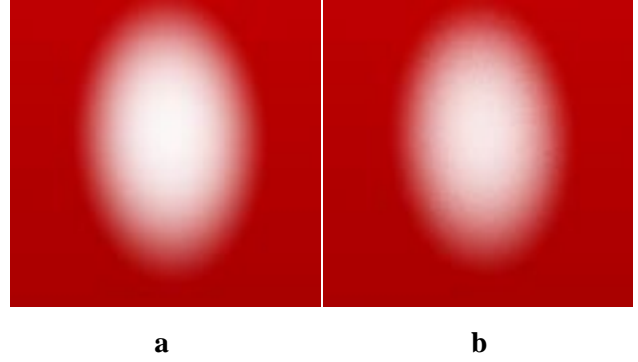


Figure 5: Specular highlights

The parameters b and p are related in a simple way. Because interpolation of eq. 2 must span ± 90 degrees,

$$\frac{c_{hi} - c_{lo}}{\cos(0) - \cos(90)} = \frac{2^b}{2^p} \quad (16)$$

Substituting $c_{hi} = 1$ and $c_{lo} = \cos(\varphi_{lo})$ into eq. 16 and solving for p gives

$$p = \log_2 \left[\frac{2^b}{1 - \cos(\varphi_{lo})} \right] \quad (17)$$

Using $\varphi_{lo} = 3^\circ$ and $b = 6$ in eq. 17 gives $p = 15.51$. Hence 16 bits would be required for interpolation using eq. 2 if a visually noiseless specular highlight were desired for $\varphi \leq 3^\circ$.

Predicting Specular Highlights

The convex hull property of the Bézier triangle may be used to separate the polygons on which specular highlights might occur from the polygons on which no specular highlight could possibly occur. Such a culling is possible because the convex hull generated from the control points is guaranteed to enclose the parametric surface defined by those control points. This property suggests that in order to determine whether a specular highlight might occur on a polygon, it is necessary to compare c_{lo} to the six control points C_{200} , C_{020} , C_{002} , C_{110} , C_{011} and C_{101} which define

the highlight function for that polygon. If any one of these control points is greater than c_{lo} , then a specular highlight might occur on the polygon. In this case, it is necessary to perform quadratic interpolation at each pixel enclosed by the polygon in order to calculate the intensity of the highlight at that pixel. However, if all of the control points are less than c_{lo} , then no specular highlight could possibly occur on the polygon, and it is not necessary to perform quadratic interpolation.

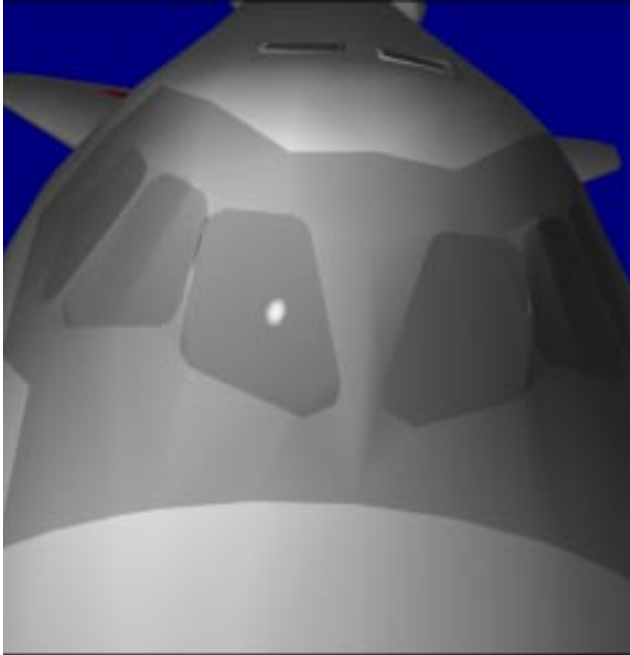


Figure 6: Space shuttle highlight

When the convex hull property is used to identify those polygons on which no specular highlight could possibly occur, 90 percent of pixels are eliminated from quadratic interpolation in the typical case. For example, Figure 6 depicts a specular highlight on the left windscreen of the space shuttle [20]. Of the 1,064,480 pixels shaded to create this 1024 by 1024 pixel image, only 106,115 pixels required quadratic interpolation.

The convex hull property reduces highlight computation to a fraction of the total cost of the shading calculation. Table 1 indicates the arithmetic operations required to evaluate various equations (multiplication by 2 is equivalent to an addition.) Linear interpolation of the color coordinates (r,g,b) requires three executions of eq. 3, and hence requires 9 multiplications and 6 additions. Quadratic interpolation may be accomplished via eq. 2. However, this equation can be rearranged into the following, slightly more efficient form that requires 9 multiplications and 8 additions

$$p(b_0, b_1, b_2) = (\mathbf{P}_{200}b_0 + 2\mathbf{P}_{110}b_1)b_0 + (\mathbf{P}_{020}b_1 + 2\mathbf{P}_{011}b_2)b_1 + (\mathbf{P}_{002}b_2 + 2\mathbf{P}_{101}b_0)b_2 \quad (18)$$

One highlight calculation requires execution of eq. 13, eq. 14, eq. 15, and eq. 18. Thus, this calculation requires 15 multiplications and 13 additions. However, because highlight calculations are required only 10 percent of the time, the average cost of a highlight calculation is only 1.5 multiplications and 1.3 additions. Thus the average cost of including a highlight calculation with an (r,g,b) interpolation is 10.5 multiplications and 7.3 additions. Compared to the cost of an isolated (r,g,b) interpolation, the inclusion of the highlight calculation increases the multiplications and additions by only 17 and 22 percent, respectively.

Table 1

Equation	Multiplications	Additions
eq. 3	3	2
eq. 13	1	1
eq. 14	3	2
eq. 15	2	2
eq. 18	9	8

It is instructive to compare the cost of quadratic interpolation computed via eq. 18 to the cost of Phong shading (linear interpolation of N , computation of $N \cdot H$ for constant H , and renormalization of $N \cdot H$ by the magnitude of N .) Linear interpolation of N and the computation of $N \cdot H$ require four executions of eq. 3. Renormalization requires computing the dot product of N with itself (another execution of eq. 3), computation of a reciprocal square root, and multiplication by the reciprocal square root. The reciprocal square root may be calculated via Newton-Raphson iteration as

$$x_{i+1} = \frac{x_i}{2}(x_i^2 y - 1) \quad (19)$$

where x_i and x_{i+1} are successive approximations to the reciprocal square root of y . Assuming a 4-bit initial approximation, it is possible to calculate a 16-bit result via two iterations of eq. 19. Hence, the cost of Phong shading is 22 multiplications and 14 additions per pixel.

Implicit to the foregoing discussion are the assumptions that the viewer and light source lie at an infinite distance from the shaded surface, and that therefore \mathbf{H} is constant. In the case where the viewer and light source lie at finite distances from the shaded surface, \mathbf{H} is not constant, and the cost of Phong shading is considerably higher. For this case, \mathbf{L} and \mathbf{V} must be constructed and normalized prior to constructing \mathbf{H} . Then $\mathbf{N} \cdot \mathbf{H}$ must be normalized by the magnitudes of \mathbf{N} and \mathbf{H} . The cost of Phong shading that includes all these operations is 56 multiplications and 41 additions per pixel. For comparison, the cost of quadratic interpolation computed via eq. 18 is on average 0.9 multiplications and 0.8 additions per pixel, because this interpolation is only performed for 10 percent of the pixels.

Limits of Applicability

It is important to understand the conditions under which quadratic interpolation can accurately model a specular highlight. In order to be useful, the accuracy should be similar to that obtained via Phong interpolation. In Phong interpolation, the renormalization at each pixel guarantees that the endpoints of the interpolated vectors lie on the surface of a sphere. For this reason, the accuracy with which quadratic interpolation approximates the surface of a sphere indicates the accuracy with which quadratic interpolation can approximate Phong interpolation.

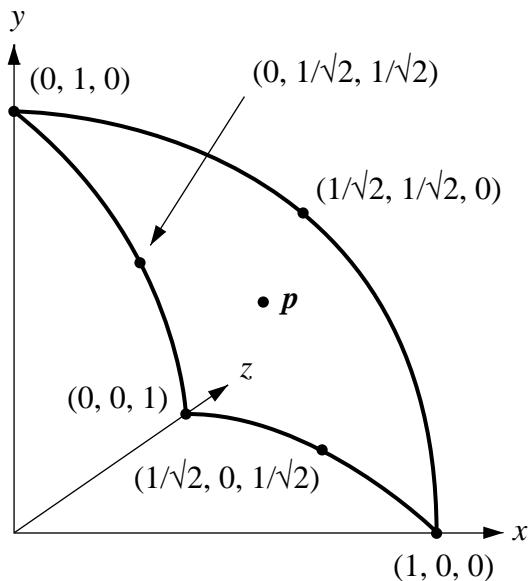


Figure 7: Approximating a sphere

Figure 7 shows a quadratic Bézier triangle that approximates the surface of a unit sphere [12]. The six labeled points are located at a unit radius from the origin.

Three of these points form the vertices of the Bézier triangle, and may be used as control points. The other three are located at the midpoints of the triangle edges, and may be used to generate control points via eq. 12. If the Bézier triangle were to coincide exactly with the surface of a unit sphere, each point \mathbf{p} on the surface of the Bézier triangle would lie at unit radius from the origin. The amount by which the Bézier triangle deviates from the surface of a unit sphere may be calculated by integrating the radius from the origin to the point \mathbf{p} over the extent of the Bézier triangle. Because the (x,y,z) coordinates of the point \mathbf{p} are functions of the barycentric coordinates as indicated by eq. 2, it is possible to define the square of the radius from the origin to the point \mathbf{p} as a function of the barycentric coordinates (b_0, b_1, b_2)

$$r^2(b_0, b_1, b_2) = x^2(b_0, b_1, b_2) + y^2(b_0, b_1, b_2) + z^2(b_0, b_1, b_2) \quad (20)$$

Expanding the functions for x , y and z using eq. 2, and integrating the resulting equation over the extent of the Bézier surface, then taking the square root of the result yields a lengthy expression for the average radius \bar{r} in terms of the control points of the Bézier surface. Because a quadratic Bézier triangle deviates from the surface of a unit sphere, this radius is not unity. Instead, the radius increases as the angle subtended by the edges of the Bézier triangle decreases. Table 2 shows the dependence of this radius upon the angle subtended by the edges of both quadratic and cubic Bézier triangles. For comparison, each edge of the Bézier triangle shown in Figure 7 subtends 90 degrees.

Table 2

angle	\bar{r} quadratic	\bar{r} cubic
90.0	0.949	0.974
45.0	0.992	0.996
22.5	1.0004	1.0002
11.25	0.99990	0.99995
5.625	1.000009	1.000005
2.8125	0.9999987	0.9999993

Table 2 demonstrates that when the edges of the Bézier triangle subtend 22 degrees or less, the quadratic and cubic Bézier triangles approximate the surface of a sphere with

high accuracy. The table also shows that the cubic Bézier triangle approximates a unit sphere only marginally better than does a quadratic Bézier triangle.

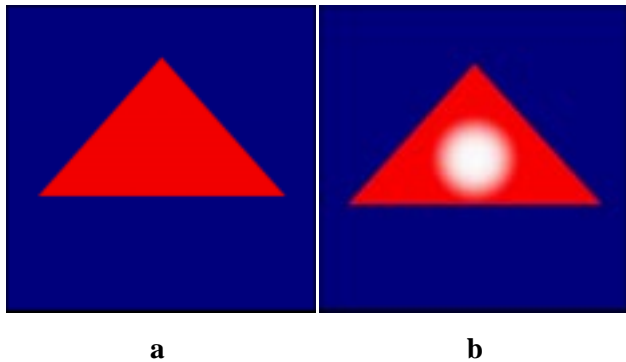


Figure 8: Highlights on approximated spheres

Figure 8 demonstrates how the deviation of the average radius from a unit radius affects the specular highlight calculated for a quadratic Bézier triangle. In Figure 8a each edge of the triangle subtends 45 degrees, whereas in Figure 8b each edge subtends 22.5 degrees. Both figures were calculated with $\phi_{lo} = 1.79^\circ$, and so specular highlights are visible when $\cos \phi \geq 0.9995$. For Figure 8a, the average radius is 0.992 (see Table 2), and therefore the average cosine is at most 0.992, which is less than the lower limit required to produce a specular highlight. For Figure 8b, the average radius is 1.0004 (see Table 2), and so the average cosine is greater than the lower limit required to produce a specular highlight.

Clipping the Highlight Function

Figure 9 illustrates clipping triangle $P_0P_1P_2$ to a rectangular viewport. Associated with triangle $P_0P_1P_2$ is the quadratic highlight function enclosed by curves P_0P_1 , P_1P_2 and P_2P_0 . This highlight function is defined by control points P_{200} , P_{020} , P_{002} , P_{110} , P_{011} and P_{101} (see Figure 2.) Clipping triangle $P_0P_1P_2$ produces triangle $T_0T_1P_2$. In order to compute specular highlights for triangle $T_0T_1P_2$, it is necessary to create a new highlight function enclosed by curves $T_{200}T_{020}$, $T_{020}P_2$ and P_2T_{200} . Creation of this new highlight function is accomplished by reparameterization of the highlight function associated with triangle $P_0P_1P_2$ to produce the new control points T_{200} , T_{020} , T_{002} , T_{110} , T_{011} and T_{101} , as demonstrated in Figure 10.

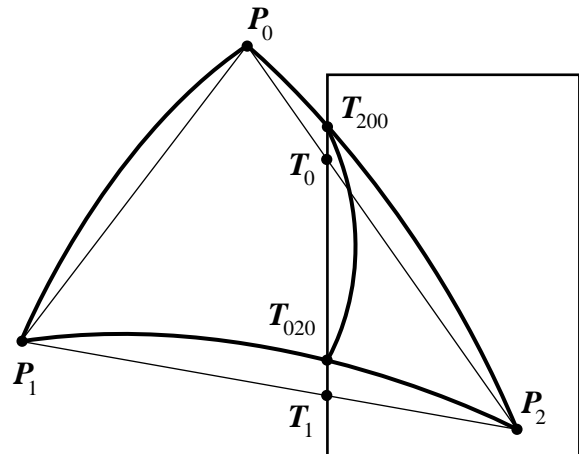


Figure 9: Polygon and highlight clipping

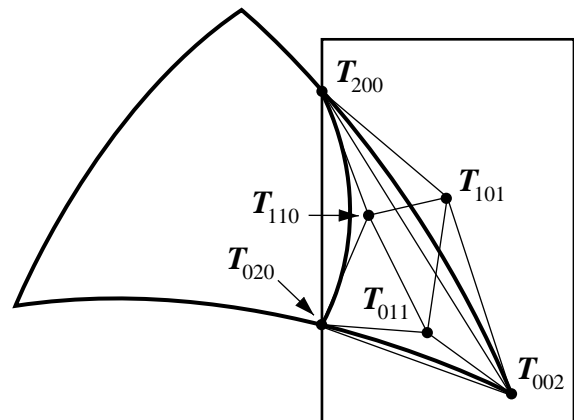


Figure 10: Reparameterized highlight function

Reparameterization of the highlight function is a three step process. First, the barycentric coordinates $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ are associated with the triangle vertices P_0 , P_1 and P_2 , respectively. Next, these barycentric coordinates are clipped to the viewport via linear interpolation contemporaneous with clipping of the vertex (x,y,z,w) coordinates [21]. Finally, these clipped barycentric coordinates are used to reparameterize the highlight function via a blossoming algorithm [12].

The first step in deriving the blossoming equation is to express eq. 2 in the recursive de Casteljaou formulation

$$\begin{aligned}
 p(b_0, b_1, b_2) = & (P_{200}b_0 + P_{110}b_1 + P_{101}b_2)b_0 \quad (21) \\
 & + (P_{110}b_0 + P_{020}b_1 + P_{011}b_2)b_1 \\
 & + (P_{101}b_0 + P_{011}b_1 + P_{002}b_2)b_2
 \end{aligned}$$

Next, (d_0, d_1, d_2) are substituted for the barycentric coordinates outside of the parentheses in eq. 21 to obtain T as a function of two sets of barycentric coordinates

$$\begin{aligned} T[(b_0, b_1, b_2), (d_0, d_1, d_2)] & \quad (22) \\ &= (\mathbf{P}_{200}b_0 + \mathbf{P}_{110}b_1 + \mathbf{P}_{101}b_2)d_0 \\ &+ (\mathbf{P}_{110}b_0 + \mathbf{P}_{020}b_1 + \mathbf{P}_{011}b_2)d_1 \\ &+ (\mathbf{P}_{101}b_0 + \mathbf{P}_{011}b_1 + \mathbf{P}_{002}b_2)d_2 \end{aligned}$$

The blossoming algorithm is executed by substituting the appropriate pairs of barycentric coordinates into eq. 22 to obtain the reparameterized control points T_{200} , T_{020} , T_{002} , T_{110} , T_{011} and T_{101} . The barycentric coordinates to be substituted into this equation are produced by clipping the barycentric coordinates $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ that were associated with the triangle vertices P_0 , P_1 and P_2 , respectively, prior to clipping. The barycentric coordinates produced by clipping are associated with vertices T_0 , T_1 and P_2 , and are designated $(b_0, b_1, b_2)_0$, $(b_0, b_1, b_2)_1$ and $(b_0, b_1, b_2)_2$, respectively. In order to understand the execution of the blossoming algorithm, consider the calculation of T_{110} . This control point is calculated by substituting $(b_0, b_1, b_2)_0$ for (b_0, b_1, b_2) , and by substituting $(b_0, b_1, b_2)_1$ for (d_0, d_1, d_2) in eq. 22. In other words,

$$T_{110} = T[(b_0, b_1, b_2)_0, (b_0, b_1, b_2)_1] \quad (23)$$

Table 3 shows the barycentric coordinate substitutions required to calculate each of the reparameterized control points T_{200} , T_{020} , T_{002} , T_{110} , T_{011} and T_{101} .

Table 3

Control Point	(b_0, b_1, b_2)	(d_0, d_1, d_2)
T_{200}	$(b_0, b_1, b_2)_0$	$(b_0, b_1, b_2)_0$
T_{020}	$(b_0, b_1, b_2)_1$	$(b_0, b_1, b_2)_1$
T_{002}	$(b_0, b_1, b_2)_2$	$(b_0, b_1, b_2)_2$
T_{110}	$(b_0, b_1, b_2)_0$	$(b_0, b_1, b_2)_1$
T_{011}	$(b_0, b_1, b_2)_1$	$(b_0, b_1, b_2)_2$

Table 3

Control Point	(b_0, b_1, b_2)	(d_0, d_1, d_2)
T_{101}	$(b_0, b_1, b_2)_2$	$(b_0, b_1, b_2)_0$

A final comment regarding clipping deals with the problem that arises when clipping a triangular polygon creates a polygon with more than three vertices. Because all of the interpolation equations, including eq. 2, eq. 3, eq. 9 and eq. 22 apply to triangles only, any polygon having more than three vertices must be decomposed into a set of triangles prior to applying these equations.

Conclusion

Bézier triangles provide an effective method for interpolating specular highlights. Compared to Phong shading, interpolation of Bézier triangles results in dramatic increases in efficiency, due in large part to the convex hull property of Bézier triangles. The convex hull allows a simple test that eliminates the need to interpolate specular highlights for 90 percent of pixels in the typical case. This paper discusses specular highlights modeled using quadratic Bézier triangles. For a selection of images, results obtained using cubic Bézier triangles are indistinguishable from the quadratic case.

Bézier triangles suggest a general approach to defining polynomial shading functions for planar triangles. The planar nature of these triangles permits rapid calculation of barycentric coordinates that may be used as interpolants in the polynomial shading functions. These shading functions may be clipped to the visible viewport by using clipped barycentric coordinates in a blossoming algorithm.

References

- [1] H. Gouraud, "Computer Display of Curved Surfaces," *IEEE Trans. C-20*, June 1971, pp. 623-629.
- [2] B.T. Phong, "Illumination for Computer Generated Pictures," *Comm. ACM*, Vol. 18, No. 6, June 1975, pp. 311-317.
- [3] J.F. Blinn, "Models of Light Reflection for Computer-Synthesized Pictures," *Proc. Siggraph 77*, Vol. 11, No. 2, ACM Press, New York, 1977, pp. 192-198.
- [4] T. Duff, "Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays," *Proc. Siggraph 79*, Vol. 13, No. 2, ACM Press, New York, 1979, pp. 270-275.
- [5] G. Bishop and D.M. Weimer, "Fast Phong Shading," *Proc. Siggraph 86*, Vol. 20, No. 4, ACM Press, New York, 1986, pp. 103-106.
- [6] M. Shantz and S.L. Lien, "Shading Bicubic Patches," *Proc. Siggraph 87*, Vol. 21, No. 4, ACM Press, New York, 1987, pp. 189-196.
- [7] D. Kirk and D. Voorhies, "The Rendering Architecture of the DN10000VS," *Proc. Siggraph 90*, Vol. 24, No. 4, ACM Press, New York, 1990, pp. 299-307.
- [8] L. Seiler, "Quadratic Interpolation for Near-Phong Quality Shading," *Siggraph 98 technical sketch*, 1998.
- [9] A.A.M. Kuijk and E.H. Blake, "Faster Phong Shading via Angular Interpolation," *Computer Graphics Forum 8*, North-Holland, the Netherlands, 1989, pp. 315-324.
- [10] L. Bergman, H. Fuchs, and E. Grant, "Image Rendering by Adaptive Refinement," *Proc. Siggraph 86*, Vol. 20, No. 4, ACM Press, New York, 1986, pp. 29-37.
- [11] K. Harrison, D. Mitchell and A. Watt, "The H Test - a Method of High Speed Interpolative Shading," *Proc. CG International '88*, Springer, Berlin, 1988, pp. 106-116.
- [12] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Fourth Ed., Academic Press, San Diego, 1988, pp. 279-307.
- [13] R. Barnhill, "Representation and Approximation of Surfaces," In *Mathematical Software III*, J.R. Rice (ed.), Academic Press, New York, pp. 69-120, 1977.
- [14] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Fourth Ed., Academic Press, San Diego, 1988, p. 38.
- [15] H. Fuchs, J. Goldfeather, J.P. Hultquist, S. Spach, J.D. Austin, F.P. Brooks II, J.G. Eyles and J. Poulton, "Fast Spheres, Shadow, Textures, Transparencies and Image Enhancements in Pixel-Planes," *Proc. Siggraph 85*, Vol. 19, No. 3, ACM Press, New York, 1985, pp. 111-120.
- [16] P.S. Heckbert and H.P. Moreton, "Interpolation for Polygon Texture Mapping and Shading," In *State of the Art in Computer Graphics: Visualization and Modeling*, D.F. Rogers and R.A. Earnshaw (eds.), Springer-Verlag, New York, 1991, pp. 101-111.
- [17] J.F. Blinn, "Hyperbolic Interpolation," *IEEE CG&A*, Vol. 12, No. 4, 1992, pp. 89-94.
- [18] W.M. Newman and R.F. Sproull, "Principles of Interactive Computer Graphics," 2nd Ed., McGraw-Hill, Tokyo, 1979, pp. 333-366.
- [19] G.S. Watkins and R.A. Brown, "System for Polygon Interpolation using Instantaneous Values in a Variable," U.S. Patent No. 5,361,386, 1994, pp. 11-13.
- [20] Space shuttle model was obtained from Viewpoint Datalabs, Orem, Utah.
- [21] I. E. Sutherland and G. W. Hodgman, "Reentrant Polygon Clipping," *Comm. ACM*, Vol. 17, No. 1, January 1974, pp. 32-42.

About the Author



Russ Brown is a principal investigator in the Scalable Systems Center at Sun Microsystems Laboratories. His research interests include algorithm development and system architecture for computer graphics and image processing. Prior to joining Sun, he worked for Evans and Sutherland Computer

Corporation. He holds MD and PhD degrees from the University of Utah.