Fast Phong Shading by Gary Bishop   **Back**

Fast Phong Shading

Fast Phong Shading. p 103-105
Gary Bishop
Room 4E-626
David M Weimer
Room 4F-637
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733

Abstract:

Computer image generation systems often represent curved surfaces as a mesh
of polygons that are shaded to restore a smooth appearance. Phong shading
is a well known algorithm for producing a realistic shading but it has not
been used by real-time systems because of the 3 additions, 1 division and 1
square root required per pixel for its evaluation. We describe a new
formulation for Phong shading that reduces the amount of computation per
pixel to only 2 additions for simple Lambertian reflection and 5 additions
and 1 memory reference for Phong's complete reflection model. We also show
how to extend our method to compute the specular component with the eye at
a finite distance from the scene rather than at infinity as is usually
assumed. The method can be implemented in hardware for real-time
applications or in software to speed image generation for almost any
system.

Introduction:

Most computer image generation (CIG) systems represent curved surfaces as a
mesh of planar polygons because polygons can be transformed and rendered
quickly with well known algorithms. Since the polygonal representation is
an artifact of the image generation method and is usually of no interest to
viewers (see figure 1), these systems attempt to restore a smooth
appearance to surfaces by varying the intensity across polygons. The
efficiency of this shading operation is critical to the performance of a
CIG system because it must be performed for one million or more pixels per
image. Although algorithms for realistic shading of polygons are well
known, real-time CIG systems have not used them because of the large amount
of computation they require per pixel. This paper describes a new
formulation of Phong shading that drastically reduces the amount of
computation compared to previously known formulations. While the new
formulation is well suited to implementation with special hardware for
real-time display, it is also appropriate for implementation with software
for a variety of display systems. Readers who are unfamiliar with surface

representation or shading are referred to one of the standard graphics texts for more information, [Newman and Sproul, 1979; Foley and Van Dam, 1983].

Background:

Assume, for simplicity, that the input to the shader consists of triangles, in screen coordinates, with normals, N, to the original curved surface specified at each vertex. The formulations can be extended to polygons with four or more sides.

The shading algorithms must determine the intensity of each point within a triangle from the surface normals, and a vector to the light source, L. The light source is assumed to be at infinity so that L is independent of the point on which the light shines. We will start with diffuse reflection,

$$I_{diffuse} = \frac{L * N}{|L| * |N|},$$

and later show how to extend the results to include Phong's highlight model and an extended highlight model.

GOURAUD SHADING: the most commonly used shading method in real-time systems. Gouraud shading [Gouraud, 1971], computes the intensity at each point by linear interpolation of the intensities at the vertices. The intensities are determined using the reflection equation above with normals given at the vertices. The method is popular for real-time systems because it produces images of acceptable quality with only 1 addition per pixel, but the shading has several disturbing characteristics. (1) Moving objects tend to "flatten out" at certain viewing angles, (2) surfaces appear dull or chalky, and (3) images show pronounced Mach bands, exaggerations of intensity change at discontinuities. Figure 2 is a Gouraud shaded chess piece.

PHONG SHADING: Phong shading, [Phong, 1973], eliminates "flattening out" and dull surfaces and reduces Mach bands but has not, to my knowledge, been used in any real-time system because of the large amount of computation required for its usual formulation. Phong's method determines the intensity of each point using an approximate surface normal that is linearly interpolated from the true surface normals specified at the vertices,

  N(x,y) = Ax + By + C

where A, B and C are chosen to interpolate the normal across the polygon. Interpolation for successive values of x and y and evaluating the illumination model requires 7 additions, 6 multiplications, 1 division and 1 square root per pixel. Phong proposed a complicated circuit for evaluating this function but it has not, to my knowledge, been implemented. Figure 3 is a Phong shaded chess piece.

Tom Duff has shown, in [Duff, T.,1979, "Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays", ACM Computer Graphics, vol 13, no 2, pp 270-275]], that Phong shading can be implemented more efficiently by combining the interpolation and reflection equations.

$$I_{diffuse}(x,y) = \frac{L}{|L|} * \frac{Ax + By + C}{|L| \, |Ax + By + C|}$$

Which can be rewritten as

$$L*Ax + L*By + L*C$$

```
       I          (x,y) = ------------------------
        diffuse           |L|  |L*Ax + L*By + L*C|
```

Performing the indicated dot products and expanding the vector magnitude
yields:

```
                             ax + by + c
       I          (x,y) = ---------------------------------------
        diffuse           SQRT( dx^2 + exy + fy^2 + gx + hy + i)
```

with

```
           L * A
     a = ----- ,
            |L|

           L * B
     b = ----- ,
            |L|

           L * C
     c = ----- ,
            |L|

     d = A * A,

     e = 2A * B,

     f = B * B,

     g = 2A * C,

     h = 2B * C, and

     i = C * C.
```

Using forward differences, this form can be evaluated for successive values
of x and y with 3 additions, 1 division and 1 square root per pixel. This
is a substantial savings over Phong's formulation but the division and
square root are still too expensive for real-time use.

A New Approach:

For display purposes we need not evaluate the reflection equation exactly;
a good approximation will suffice. The "error" introduced by the
approximation will be of no consquence since Phong's normal interpolation
and the reflection equation are already approximations. the one-dimensional
form of Taylor's series is widely used for approximation functions, such as
sine and log, with polynomials. The less widely used, two-dimensional form
will serve to approximate the reflection equation.

Taylor's series for a function of two variables is:

f(a+h, b+k) = f(a,b) + (h ... + k ...)*f(a,b) + ... [look it up!]

Shifting the triangle so that (0,0) lies at its center, and expanding in a
Taylor series to the second degree produces

```
       I          (x,y) = T x^2 + T xy + T y^2 + T x + T y + T
        diffuse           5       4       3       2     1     0
```

with

```
       3ig^2 - 4cdi - 4agi
```

```
   T   = ---------------------,
    5       8 * i^2 * SQRT(i)


         3cgh - 2cei - 2bgi - 2ahi
   T   = ----------------------------,
    4            4 * i^2 * SQRT(i)


         3ch^2 - 4cfi - 4bhi
   T   = ---------------------,
    3       8 * i^2 * SQRT(i)


             2ai - cg
   T   = -----------------,
    2      2 * i * SQRT(i)


             2bi - ch
   T   = -----------------, and
    1      2 * i * SQRT(i)


              c
   T   = ---------
    0      SQRT(i)
```

Using this expansion and forward differences, the intensity can be
evaluated with only 2 additions per pixel.

The method can be extended to handle Phong's specular reflection model,

$$I_{specular} = (N * H)^n$$

by using Taylor's series to evaluate the dot product and table lookup to do
the exponentiation (A table of 1K bytes will allow exponentiation to powers
up to 20 with less than 1 percent error and a table of 8K bytes allows
powers up to 164.) H is a vector in the direction of maximum highlight
which is halfway between the light direction, L, and the eye direction, E,

$$H = \frac{E * L}{|E + L|},$$

The eye, like the light source, is assumed for the present to be at
infinity. Phong's reflection equation,

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

can now be evaluated with only 5 additions and 1 memory access per pixel
(the ambient component can be rolled into the series for the diffuse
component and the series for the specular component can be scaled to allow
direct addressing of the exponentiation table). It should be simple to
configure hardware to do these operations in less than 100 nanoseconds per
pixel.

The additional computation required to determine the T_i for our new method
is offset by the greatly reduced computation per pixel for polygons with 10
or more pixels. This estimate is based on the following assumptions: (a)
the algorithms are implemented on sequential processors of conventional
design, (b) with modern hardware, the time for floating point addition,
multiplication, and memory references are all about equal, (c) the
computation of $1 / SQRT(x)$ requires about 10 operations, and (d) common
subexpressions have been removed from the formulas for T_i. Timings and

analysis of the method as implementd in the raster testbed [Whitted and
Weimer, 1982] also support a 10 pixel break-even point. The break-even
point would be much lower if some simple hardware were added to do the
additions and table lookup for our method in parallel. Of course, special
hardware could be built for the other methods but it would be much more
complicated and probably not as fast.

The images we have produced with this approximation are indistinguishable
from those produced with Phong's method. For polygons with more than about
60 degrees of curvature, this new expansion will produce shadings that are
noticeably different from Phong shading but curvatures this large should
never be represented by a single polygon.

Going Further:

Since the form of the polynomial we have to evaluate at each pixel is
independent of the original reflection equation, we can use more elaborate
models. One useful extension is to provide for finite eye distance, thus
requiring that the vector to the eye, E, vary across the scene. The
resulting variation in the specular component produces more natural looking
illumination for scenes rendered with perspective. This is most obvious
when there are planar surfaces in the scene because Phong shading with the
eye at infinity, like flat shading, renders all parallel surfaces with the
same intensity. Figure 4 is a comparision of Phong shading with the eye at
infinity and with the eye at a true perspective distance. The reflection
equation for the specular component with the eye at a finite distance now
becomes

$$I_{specular} = \frac{N(x,y) * H(x,y)}{|N(x,y)| * |H(x,y)|},$$

where $H(x,y) = E(x,y) + L$ and $E(x,y)$ interpolates the eye vector across the
triangle. This can be expanded just as before

$$I_{specular} = \frac{(Ax + By + C) * (Dx + Ey + F)}{|(Ax + By + C)| * |(Dx + Ey + F)|}$$

After performing the dot products and expanding the vector magnitude as
before

$$I_s = \frac{ax^2 + bxy + cy^2 + dx + ey + f}{\text{SQRT}((gx^2 + hxy + iy^2 + jx + ky + l)*(mx^2 + nxy + oy^2 + px + qy + r))}$$

with

  a = A * D,

  b = A * E + B * D,

  c = B * E,

  d = A * F + C * D,

  e = B * F + C * E,

  f = C * F,

  g = A * A,

  h = 2A * B,

```
   i = B * B,

   j = 2A * C,

   k = 2B * C,

   l = C * C,

   m = E * E,

   n = D * D,

   o = 2D * E,

   p = 2D * F,

   q = 2E * F, and

   r = F * F.
```

And expanding in Taylor's series about (0,0).

```
  I(x,y) = T x^2 + T xy + T y^2 + T x + T y + T
            5       4       3       2       1      0
```

with

```
     8al^2r^2 - 4djlr^2 - 4fglr^2 + 3fj^2r^2 - 4dl^2pr + 2fjlpr - 4fl^2mr
T = ---------------------------------------------------------------------
 5                              8l^2r^2 * SQRT(l*r)

        + 3fl^2p^2
    + --------------------,
      8l^2r^2 * SQRT(l*r)

     4bl^2r^2 - 2dklr^2 - 2ejlr^2 - 2fhlr^2 + 3fjkr^2 - 2dl^2qr
T = -----------------------------------------------------------
 4                            4l^2r^2 * SQRT(l*r)

        + fjlqr - 2el^2pr + fklpr - 2fl^2nr + 3fl^2pq
    + ------------------------------------------------,
                       4l^2r^2 * SQRT(l*r)

     8cl^2r^2 - 4eklr^2 - 4filr^2 + 3fk^2r^2 - 4el^2qr + 2fklqr - 4fl^2or
T = ---------------------------------------------------------------------
 3                              8l^2r^2 * SQRT(l*r)

        + 3fl^2q^2
    + --------------------,
      8l^2r^2 * SQRT(l*r)

     2dlr - fjr - flp
T = -------------------,
 2    2lr * SQRT(l*r)

     2elr - fkr - flq
T = -------------------,
 1    2lr * SQRT(l*r)

          f
T = -----------
 0   SQRT(l*r)
```

Summary:

We have shown that computer image generation systems can use Phong shading
with only a little more computation per pixel than is required for Gouraud
shading. This result should allow real-time systems to generate more
realistic scenes and conventional systems to produce images more rapidly.
To test the method, we have implemented it as a shader subroutine for the
raster testbed. The cpu time for rendering figure 4 with 14620 polygons at
2048x2048 resolutions on a DEC microVAX-II are: Shading method CPU time
Notes Gouraud 406s Taylor (infinite) 767s incl 119s for Taylor coeffs
Taylor (finite) 850s incl 202s for Taylor coeffs Duff 2597s incl 1420 in
sqrt Phong 3900s incl 1405 in sqrt

Acknowledgements:

Tom Duff and the reviewers provided many helpful suggestions.

[BACK] Back