Situation Normal

Andrew Glassner

Microsoft Research

In the pantheon of science, having your name attached to something is a pretty high honor. Graphics isn't terribly big on this practice of eponymy—most of our techniques bear descriptive names, such as *z*-buffering, ray tracing, and the RGB color space. But there are exceptions. Two of the most famous graphics techniques, derived from shading algorithms by Henri Gouraud and Bui-Tuong Phong, have been implemented in hardware and software worldwide.

Both methods use various hacks to smooth out the shading of a polygonal surface. So rather than looking like an assembly of flat slabs, a smooth-shaded model seems, well, smoother. The sharp creases between polygons are gone, replaced by a continuous change in tone or color.

But if the original polygons aren't being rendered directly, then the shading doesn't correspond to the original model. What model does it correspond to? In other words, what is the smooth surface which, when rendered accurately, has the same appearance as a Phong-shaded polygonal model? That's the surface Phong shading pretends actually lies underneath.

Often we know what we want that surface to be, because we started there. For example, if we begin with a cylinder, chop it up into polygons, and render the polygons with Gouraud or Phong shading, we would like the final rendered picture to have the same intensities that we'd get from an accurate point-by-point rendering of the original cylinder. Even when we don't start with a smooth mathematical object, we often imagine that the polygons form a framework over which an elastic sheet stretches; this sheet is conceptually the smooth surface we want to represent. How close do these shading methods come to these preferences?

Before we plunge in, I'd like to make the standard distinction between the two very separate ideas often lumped together as "Phong shading." The first—Phong normal interpolation—is the process of computing a point's surface normal by linear interpolation of the components of two normals at either end of a line containing that point. This normal may then be used as part of a shading equation that takes into account specular highlights in an empirical manner—Phong illumination. Throughout this article, I will deal only with perfectly diffuse surfaces lit by a single light source, so Phong illumination isn't part of the discussion. And we will assume that the light source is conveniently located at infinity (so the direction in which we look at that light doesn't change from point to point over the surface).

So here, "Gouraud shading" means the process of interpolating a color component to find intermediate color values across a polygon. And "Phong shading" means interpolating surface normals to find intermediate normals that we then evaluate with respect to the light source to find a color for that point.

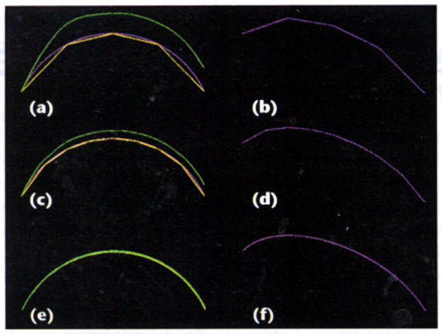
Under the surface

Let's look first at Gouraud shading. To keep this discussion simple, we're going to do everything in 2D. Suppose we're rendering the top half of a cylinder. Figure 1a shows the basic idea: The purple curve is the cylinder, and the four yellow lines delineate four polygonal facets that approximate the cylinder. There is one light source, directly to the northwest. Figure 1b shows

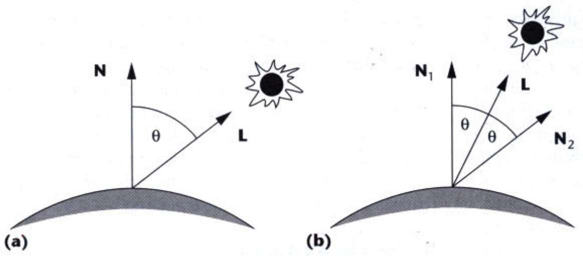
Further Reading

You can find lots of interesting papers on Phong shading; here I've listed just a few. A nice discussion of Phong shading appears in Tom Duff's paper, "Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays," pages 270-275 of the Siggraph 79 conference proceedings. Gary Bishop and David Weimer offer a way to speed up the computation in "Fast Phong Shading," pages 103-106 of the Siggraph 86 conference proceedings. A famous paper by Bui-Tuong Phong and Frank Crow on smoothing out Phong-shaded images is sadly all but unavailable; the official citation lists "Improved Rendition of Polygonal Models of Curved Surfaces" in *Proceedings of the 2nd USA-Japan Computer Conference*, 1975. Nelson Max has investigated smoothing out models in a variety of ways, so that silhouettes and intersections are rounded as well, in "Smooth Appearance For Polygonal Surfaces," published in the June 1989 issue of *The Visual Computer*, pages 160-173.

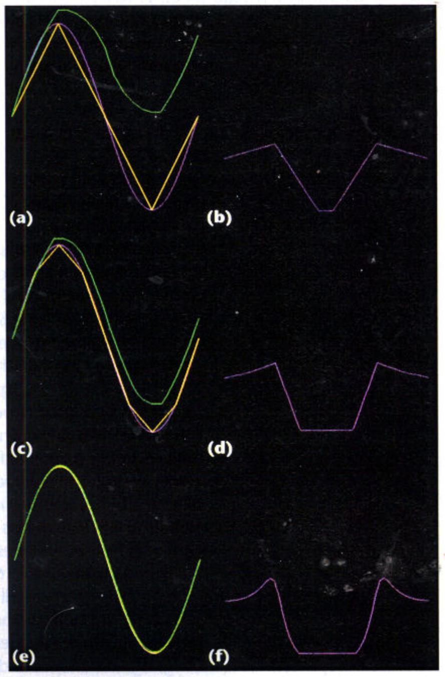
Recently, C.W.A.M. van Overveld and Brian Wyvill described a clever method for changing the normal-interpolation scheme for Phong illumination with highlights. Rather than interpolate the normal, they interpolate the highlight vector, then derive the shade from a table lookup. They described their work in "Hi-speed, Hi-fi, Hi-lights: A Fast Algorithm for the Specular Term in the Phong Illumination Model," which appears in the *Journal of Graphics Tools*, Vol. 1, No. 2, pages 25-30.



1 Gouraud shading applied to a cylinder. The purple curve is the original cylinder, approximated by yellow facets. The single light source is directly to the northwest. The green curve is shape derived from the shading. (a) A four-facet approximation and (b) its intensity profile. (c) An eight-facet approximation and (d) its intensity profile. (e) A 50-facet approximation and (f) its intensity profile.



2 The geometry behind diffuse shading: (a) The normal and light make an angle θ. (b) Only two candidate normal vectors make an angle θ with the light.



3 Gouraud shading applied to a sine curve. The images are arranged as in Figure 1.

the intensity profile created by Gouraud-shading the yellow facets. We want to find out what surface this intensity profile represents.

Because everything is so simple, we can use a very primitive shape-from-shading algorithm. The first step is to recall the basic equation of Gouraud shading that relates surface normals and illumination geometry. For a perfectly diffuse, grayscale world,

$$I = k_d(\mathbf{N} \cdot \mathbf{L}) = k_d \cos\theta$$

where I is the resulting intensity, k_d is a scalar that controls diffuse reflectivity, **N** is the (unit-length) surface normal, and **L** is the (unit-length) vector to the light source. Figure 2a shows the geometry of the situation.

In Gouraud shading, we evaluate this equation only at the facets' vertices, and then interpolate the value of I across the facet. To find **N** at a vertex, we often average the values of the facets that share that vertex. For this column, I had access to the underlying curved shapes, so I used the actual surface normal computed from the surface at that vertex. If the normals at the endpoints are **P** and **R**, then we compute $I_\mathbf{P}$ and $I_\mathbf{R}$ and linearly interpolate them to find the Gouraud intensity I_G (for utter simplicity, let's assume that $k_d = 1$):

$$I_G = \alpha I_\mathbf{P} + (1-\alpha)I_\mathbf{R} = \alpha(\mathbf{P}\cdot\mathbf{L}) + (1-\alpha)(\mathbf{R}\cdot\mathbf{L})$$

In shape-from-shading, we need to invert this equation: we're given I (the shade), and we want to find **N** (which reveals the shape). Then we easily find that $\theta = \cos^{-1}(I)$. Given the light source direction **L**, only two vectors in the plane make an angle θ with **L**, as Figure 2b shows. The quick-and-dirty approach (which works well here) picks the choice closest to the neighboring surface normals. To start this process, you guess (or cheat) to get one normal somewhere on the surface, then work outward from there.

In Figure 1a, I knew that the normal at the far left pointed almost due west, so that was my starting normal. Then I marched across the intensity graph in Figure 1b, looking up I, computing θ, and choosing the closest **N**. To plot the surface, I drew a little green line perpendicular to **N** that had a horizontal span of one pixel starting where the last bit of surface ended. Then I just marched on to the next pixel, got the next I and θ, and continued the process.

Notice that the curve is, in fact, smooth. That's why Gouraud shading looks smooth—it's faking this smooth surface. But there's something wrong, because the curve seems to rise too high over the underlying faceted approximation. Remember that this green curve represents the shape of the underlying surface. Well, is it really such a problem? Notice that the rightmost three-fourths of the curve tracks the facets (and the cylinder) pretty well, but it's just too high. Since the light's at infinity, and we're not using the Gouraud values to compute depth information, this isn't really a problem—but it is worth thinking about. Why does it go wrong?

The Gouraud surface rises up because the interpolated shade values in the leftmost segment don't do a good job of tracking the actual shade values of a smooth cylinder. Each of those little errors accumulates along the facet. The problem here is not too few facets, but that the illumination over the facets doesn't match the illumination that would derive from the actual cylinder.

Figures 1c and 1d show the same process repeated for eight facets. The situation looks quite improved. More facets means more places where the Gouraud shading locks to the actual computed illumination values. Figures 1e and 1f pump up the subdivision to 50 facets, and now the surface implied by Gouraud shading matches the underlying cylinder very well.

Let's try the same process with a sine curve. Figures 3a and 3b show the four-facet approximation (the middle two facets are colinear and look like one long straight line, but they actually have a shared vertex where they cross the purple sine curve). The intensity profile is five facets, not four, because we've clipped it at 0. (Otherwise we'd have to suck the photons out of your eyes when you look at the rendered image, and while not particularly painful, this can be dangerous.) It's reassuring to see that crinkly little sine-wave-ish curve in the center, but it's not really correct for the surface. Of course, we're asking way too much of Gouraud shading to try to handle this much surface variation with only four facets, but it's interesting to see that it still gets things vaguely right.

Figures 3b through 3e show the 8- and 50-facet versions, respectively. As you would expect, things get much better with improved subdivision.

Approximately normal

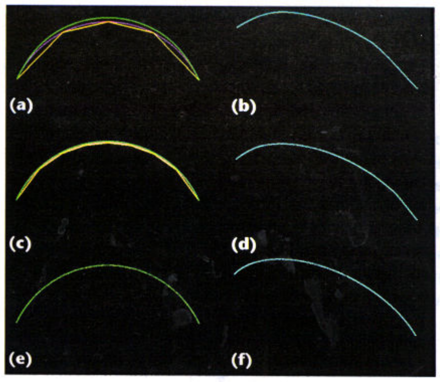
Let's now turn to Phong normal interpolation. We will use exactly the same shading equation presented above, but we'll change how we find the normal. Just as in Gouraud shading, Phong shading finds the normals at the vertices of the polygon. But in Phong shading we next interpolate the normals across the polygon's face, then recompute the illumination value at each point.

Just how to interpolate the normals has been a subject of much debate, and we'll return to it below. For now, though, we'll use the method originally proposed by Phong and most widely implemented. If **P** and **R** are the normals at the two ends of a line, then the normal **Q** at a point between them that cuts the line in the ratio α can be found from

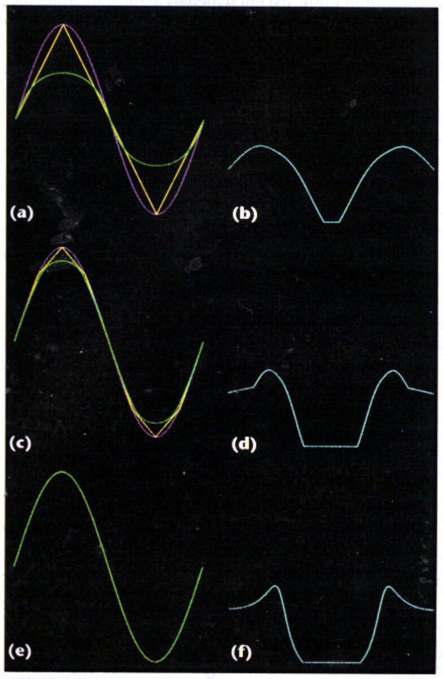
$$\mathbf{Q} = \frac{\alpha\mathbf{P} + (1-\alpha)\mathbf{R}}{|\alpha\mathbf{P} + (1-\alpha)\mathbf{R}|}$$

So, we just linearly interpolate each of the components, then normalize the result. Then we plug it into the shading formula to find the Phong intensity I_P:

$$I_P = \mathbf{Q} \cdot \mathbf{L}$$



4 **Phong shading applied to a cylinder. Compare to Figure 1.**



5 **Phong shading applied to a sine curve. Compare to Figure 3.**

Figures 4a and 4b show the same 4-faceted cylinder that we used in Figure 1, only now we've used Phong shading. It's easy to see how much closer the derived curve follows the underlying surface. It also links up with the far point of the cylinder. Figures 4c through 4f show the 8- and 50-faceted cylinders; the match in Figure 4e looks about perfect.

Figure 5 shows the same sine wave as before. The green curve almost looks like a B-spline built on the control polygon of the yellow facets (but it doesn't stay within the convex hull they define). Notice how much more symmetrical the figure is compared to the Gouraud-shaded version.

Bright answers

Let's compare the Gouraud and Phong illuminations. Tom Duff expanded out the dot product in his 1979 paper to find

$$\begin{aligned} I_{\mathrm{P}} &= \frac{\alpha\mathbf{P}+(1-\alpha)\mathbf{R}}{\left|\alpha\mathbf{P}+(1-\alpha)\mathbf{R}\right|}\cdot\mathbf{L} \\ &= \frac{\alpha\mathbf{P}\cdot\mathbf{L}+(1-\alpha)\mathbf{R}\cdot\mathbf{L}}{\left|\alpha\mathbf{P}+(1-\alpha)\mathbf{R}\right|} \\ &= \frac{I_{\mathrm{G}}}{\left|\alpha\mathbf{P}+(1-\alpha)\mathbf{R}\right|} \end{aligned}$$

Thus, Phong shading produces the same values as Gouraud shading, except scaled by a normalization factor that is always in the range 0 to 1. So in equivalent situations, Phong shading will always produce a brighter image than Gouraud shading. Remember that this has nothing to do with highlights—we're simply looking at the purely diffuse component here.

Tom observed that sometimes people talk about speeding up Phong shading by skipping the renormalization step—that is, you interpolate the normal components, but you don't bother scaling the result to give it unit length. In other words, the denominator in the above equation is assumed to be 1, which means that this procedure would compute exactly the same values as Gouraud shading. Sure, it's slower, but it's more expensive. Figure 6 shows the intensity profiles for Gouraud and Phong shading overlaid for the cylinder and the sine wave. You can see that the Phong values are always the larger of the two.

What is the nature of this normalization factor? Figure 7 shows the geometry of the situation; we're basically sweeping through a triangle to find the interpolated normal, then extending it to reach the unit circle. The catch here is that we're not stepping by equal angles. Because we're interpolating components by equal amounts, we're stepping by equal lengths along the chord. Figure 8 shows a plot of the length of the interpolated normal as a function of α for various values of θ.

Who's to say what's normal?

This whole process of interpolating normal components seems pretty suspect to a lot of people, myself included. After all, why should that be the best way to interpolate normals?

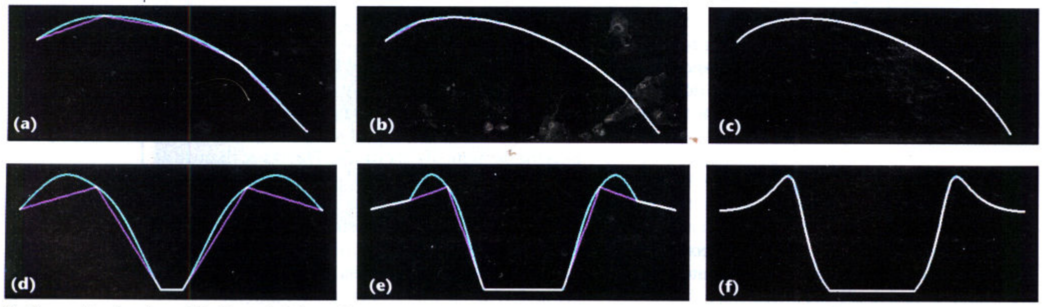
The answer, of course, is that it's not the best way—but there is no better way. It's an arbitrary hack based on the assumption that the default shape resulting from this interpolation is a reasonable match for the underlying shapes we're modeling with polygons. Other interpolation methods will produce other types of curves. As with any approximation, sometimes one type of guess will be better than another; there's no "right" way to interpolate normals unless you know what kind of surface you're trying to match.

Still, it's interesting to look at alternatives. One of the most popular approaches interpolates the normal in equal angular steps rather than equal steps along the chord. So we want a formula to find the interpolated normal $\mathbf{Q}$ between $\mathbf{P}$ and $\mathbf{R}$ as a function of α, which is the percentage of the angle between the extremes. This is the equal-angle interpolation formula.

My favorite geometric derivation of this formula was developed by Frits Post. Take a look at Figure 9. The point C lies at the center of a circle of unit radius. I've marked off the two extreme vectors $\mathbf{P}$ and $\mathbf{R}$, which both have unit length. We're going to write $\mathbf{Q}$ as a linear sum of $\mathbf{P}$ and $\mathbf{R}$, or $\mathbf{Q} = \beta_1\mathbf{P} + \beta_2\mathbf{R}$. What are these two scaling factors?

We'll start by labeling a bunch of angles and lengths, which makes finding the unknowns pretty simple. For convenience, I'll write the point at the tip of any vector as that vector's name without boldface; that is, point P is at the tip of vector $\mathbf{P}$. Angles will be identified by the three points that compose them (for example, $\angle QP'C$), though when it's unambiguous I'll just use a single point (such as $\angle P'$).

In Figure 9 I've marked points P' and R', the tips of the scaled vectors $\mathbf{P}'$ and $\mathbf{R}'$. Then we have a parallelogram formed by $CP'QR'$. Note that the line QP is not in general tangent to the circle. $\angle PCR$ is θ, our original angle between the vectors $\mathbf{P}$ and $\mathbf{R}$. And $\angle PCQ$ is the interpolated angle, $\psi = \alpha\theta$. So $\angle QCR$ is $\theta - \psi$. Because CR' is parallel to $P'Q$, $\angle CQP'$ is the same as $\angle QCR'$, or $\theta - \psi$. That just leaves $\angle P'$, which must be $\pi - \theta$. By construction,



6 **Phong shading (cyan) is always brighter than Gouraud shading (purple). (a) The intensity profiles for the 4-facet cylinder, (b) the 8-facet cylinder, (c) the 50-facet cylinder, (d) the 4-facet sine wave, (e) the 8-facet sine wave, and (f) the 50-facet sine wave.**

$|CP'| = \beta_1$ and $|CR'| = \beta_2$

Now we can use the law of sines to write down the relationships between angles and lengths in triangle $P'QC$, and solve:

$$\frac{\sin P'}{|QC|} = \frac{\sin Q}{|P'C|} = \frac{\sin C}{|P'Q|}$$

Recalling that $\sin(\pi - \theta) = \sin(\theta)$, we can replace these distances with the lengths we've assigned them:

$$\frac{\sin\theta}{1} = \frac{\sin(\theta-\psi)}{\beta_1} = \frac{\sin\psi}{\beta_2}$$

Solving for β_1 and β_2 yields

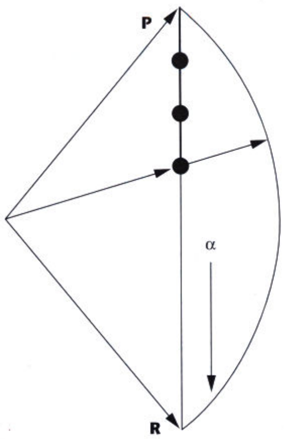
$$\beta_1 = \frac{\sin(\theta-\psi)}{\sin\theta} \qquad \beta_2 = \frac{\sin\psi}{\sin\theta}$$

So our interpolation formula for $\mathbf{Q}(\alpha)$ leads us to

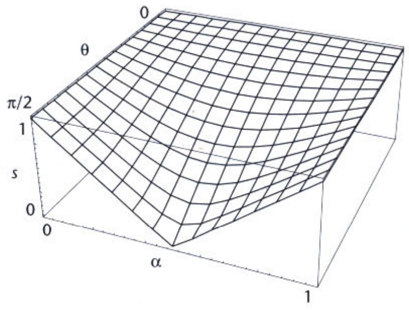
$$\mathbf{Q}(\alpha) = \frac{\sin(\theta-\psi)}{\sin\theta}\mathbf{P} + \frac{\sin\psi}{\sin\theta}\mathbf{R}$$

where $\psi = \alpha\theta$. You can prove to yourself that this doesn't require a normalization step; in other words, the vector $\mathbf{Q}$ computed by this formula always has unit length.

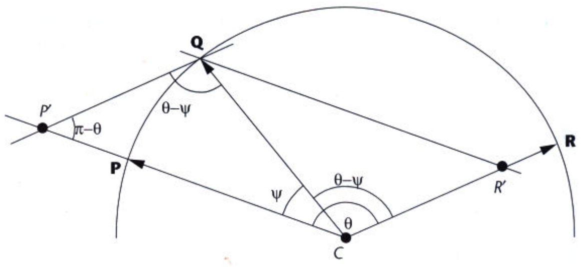
Equal-angle interpolation will yield different results than component interpolation, but how different? Figure 10 shows a comparison of the two techniques on a two-element version of the sine curve. This is about as extreme a test as you can hope for, because the normals are almost antiparallel at the two ends. The two techniques yield pretty much the same intensity profile. Under less severe conditions the two methods are usually very close—much closer than Gouraud and Phong shading. ■



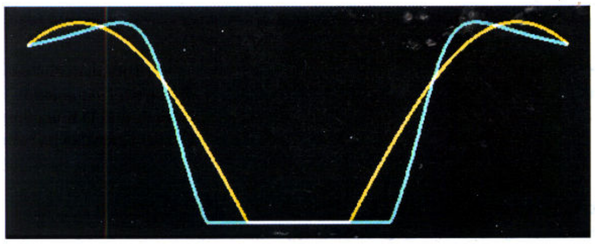
7 When interpolating normals using component interpolation, we effectively move along a straight line between the two endpoints. We step in equal increments along this line, not in equal angular steps between the extremes. The interpolated normal must then be scaled to unit length.



8 The length *s* of the interpolated normal as a function of the angle θ between the extremes and the interpolant α between them. Note that when the angle is small, the interpolated lengths are very close to 1. When θ = π/2, the normal at α = 0.5 has length 0 and is undetermined. We divide by the value in this plot to scale the interpolated normal to unit length.



9 The geometry behind equal-angle interpolation.



10 A comparison of intensity profiles for equal-angle interpolation (in yellow) and Phong shading (in purple) for an extreme case.