Abstract

The computational effort to render images with light sources and camera at infinity is less than with light sources at finite distance from the illuminated surface. On the other hand, in the case of an infinitely remote light source and camera, planar polygons don’t receive highlights. In this paper, a method is suggested to use the (relatively cheap) infinite-distance model instead of the expensive finite-distance model for the computation of highlights. It works by replacing a light source at finite distance by a light source at infinite distance, and at the same time adjusting the normal vectors in such a way that the resulting illumination pattern stays the same. With these modifications, a simple table look-up comes in the place of an expensive computation to obtain the specular term in the standard illumination model.
Hi-speed, hi-fi, hi-lights: a fast algorithm for the specular term in the Phong illumination model.

C.W.A.M. van Overveld
Department of Mathematics and Computing Science
Eindhoven University of Technology*

B.Wyvill
Department of Computer Science
University of Calgary†

November 18, 1999

Keywords shading, rendering, normal vectors, computer graphics

1. Introduction: the asymptotic illumination model

Phong shading interpolates normal vectors across polygons, using the

*Address: P.O.Box 513, 5600 MB, Eindhoven, The Netherlands. Email: wsinkvo@info.win.tue.nl
†Address: 2500 University Drive N.W., Calgary, Alberta, Canada, T2N 1N4 Email: blob@cpsc.ucalgary.ca
following formula for highlight calculation: ([Foley 90], pag. 731-739)

\[ R(n, h) = R_0(n \cdot h)^\beta. \]  \hspace{1cm} (1)

Here, \( R_0 \) is the initial intensity, \( R \) is the observed intensity, \( n \) is the normal vector at the point of the surface that is being rendered, \( \beta \) is a measure for the specularity of the surface and \( h \) is the normalised so called halfway-vector. This halfway vector is the bisector of the viewing direction and the vector which points to the light source.

In general, \( h \) varies over the surface which makes the computation of 1 quite expensive. Three computations contribute to the cost of the calculation: (a) computing a normalised \( h \); (b) computing the dot product \( (n \cdot h) \) and (c) computing the exponent with power \( \beta \). Even if we replace (c) by a simple approximation, such as Schlick’s proposal: \( t^\beta \rightarrow \frac{t}{\beta \cdot \log(1 + t)} \) ([Schlick 94]), the first two parts still involve some 10 floating point multiplications, a floating point division and a square root for each pixel.

However, cases exist where \( h \) is practically constant: if both the viewer and the light source are at sufficiently large distance from the scene, the variation of \( h \) may be neglected. In the sequel, this will be called 'the asymptotic illumination model'. It is obvious that in the asymptotic illumination model \( h \) doesn’t need normalisation in each pixel; effectively, only
the parts (b) and (c) remain to be done. Moreover, as we will see in section 3, further simplifications then can reduce the cost of (b) and (c).

2. Transforming a finite light source to the asymptotic model

The essential observation here, is that in (1) the normal vector \( n \) in itself is not essential for the reproduction of the highlights, but rather the angle between \( n \) and \( h \). As long as this angle varies over the surface, a varying amount of reflected light is obtained. This means that we can rotate both \( n \) and \( h \) over the same axis and the same angle without affecting the specular reflection.

A cheap way to vary the angle between \( n \) and \( h \) over a scanline is to keep \( h \) constant and vary \( n \). Varying \( n \) can be easily achieved by means of a simple scanline-wise incremental algorithm (see also section 3).

If we move the light source to infinity but still want to have the same result for specular term in the vertices of the polygons, we should first perturb (rotate) the normal vectors in the vertices. Since this rotation is performed at the vertices of the polygons, the resulting normal vector field stays continuous over the polygon mesh if normal vectors in the vertices are shared by the neighbour polygons.

The appropriate way to rotate the normal vectors in the vertices, in such a way that the finite-distance light source in combination with the original
normal vectors yields the same illumination pattern as the asymptotic light source in combination with the perturbed normal vectors, can be seen in figure 1. Here, a polygon is depicted together with a finite-distance light source located at $S$. Consider a vertex $a$ where $n$ is a normalised normal vector. The angle that should be conserved is the angle between $h$ and $n$ where the halfway vector $h$ is the unit vector in the direction half way between $S-a$ and $v$ where $v$ is the viewing direction. Let $k$ be the asymptotic direction of the infinitely remote light source that is to replace the light source at $S$, and let $h'$ be the halfway vector between $k$ and $v$. Then we compute a rotation matrix $M$ for the rotation over an axis $h \times h'$ such that $h$ becomes oriented along $h'$, and we rotate $n$ in the point $a$ with $M$. (See [Rogers 76] for the derivation of a rotation matrix over an arbitrary axis). Similar for the other vertices of the polygon.

Note that rotating the normal vectors only occurs in the vertices of a polygon, so it can conveniently take place as a pre-process before the actual rendering takes place. If the scene is illuminated by several light sources the approach as given above still can be used by using separate passes, one for every light source, and adding the r,g,b colour values of the resulting images per pixel.

The method proposed here can be applied in combination with bump
mapping. In that case, first the normals in the vertices of a polygon are rotated, and interpolation of the normal vector over a scanline takes place.

For every pixel, a perturbation of the interpolated normal vector according to the applied bump map can be performed.

The method also works if the normal vectors are not perpendicular to the polygon, which is the case if the polygon is part of a mesh which approximates a curved surface. However, in this case the obtained effect is not so conspicuous, since for a polygon mesh which approximates a curved surface, highlights already occur with the standard Phong algorithm even in the asymptotic model.

3. Replacing steps (b) and (c) by table look-up.

Now that we have replaced the expression of 1 for a fixed normal and a varying $h$ by an equivalent expression with fixed $h$ and varying normal, we can apply a further computational reduction. Let the normal $n$ have components $n.x$ and $n.y$ in the screen coordinate system. The coordinate $n.z$ perpendicular to the screen is redundant, since $n.z = \sqrt{m^2 - n.x^2 - n.y^2}$ where $m$ is the normalisation length of the normal vectors. We assume that all normals point in the direction of the viewer, in a left handed viewing system with the eye looking along the positive $z$ axis, so all $n.z$ are non-negative. This means that, with $h$ constant, $R(n, h) = R_0(n \cdot h)^\beta$ is only a
function of the two variables \( n \cdot x \) and \( n \cdot y \). In that case, \( R(n, h) = R(n \cdot x, n \cdot y) \) can be easily pre-computed in a two-dimensional table, which reduces the evaluation of (1) for every subsequent pixel along a scanline to an incremental update of \( n \cdot x \) and \( n \cdot y \) using a standard DDA algorithm ([Foley 90], pag. 74) (which costs about 4 additions) plus a table-look-up.

As follows: assume that we use a normalisation length \( m \), say. This means that both \( n \cdot x \) and \( n \cdot y \), that are represented by integers for the DDA algorithm, vary between \(-m\) and \(+m\). There are \((2m+1) \times (2m+1)\) different \((n \cdot x, n \cdot y)\)-pairs, of which only the ones with \( n \cdot x^2 + n \cdot y^2 < m^2 \) correspond to an existing normal vector. So we can define a 2-D table with \((2m+1) \times (2m+1)\) entries, which for each entry \((n \cdot x + m, n \cdot y + m)\) with \( n \cdot x^2 + n \cdot y^2 < m^2 \), contains the value of \( R \) from (1) that belongs to the normal vector \((n \cdot x, n \cdot y, (m^2 - n \cdot x^2 - n \cdot y^2)^{1/2})\). Such a table has to be computed once for each of the light sources in the scene.

4. Limitations, Discussion

Our proposal consists of two steps. First, prior to scan converting a polygon, the normal vectors \( n \) in the vertices are rotated to \( n' \) such that \((n \cdot h) = (n' \cdot h')\) where \( h \) is the halfway vector for the original light source and \( h' \) is the halfway vector for an infinite light source. This replacement gives exactly the same value for 1 in the vertices. Next, during scan conver-
sion, the normal vectors are represented by integers \( n_x \) and \( n_y \) and these are interpolated over a scanline by means of a DDA. This linear interpolation differs from the original computation based on varying the halfway vectors, therefore it may result in a value for \( R(n_x, n_y) \) that differs somewhat from the exact \( R(n, h) \) inside a polygon. As a consequence highlights are faithfully reproduced, even inside polygons, but they may occur in slightly different places compared with the highlights from the Phong algorithm\(^1\). The difference between the Phong algorithm and ours is smaller when the polygons are smaller, since the two algorithms are equivalent at the vertices.

The table lookup part of the proposed algorithm works under the assumption that even for the rotated normal vectors, \( n_z \) is non-negative. For an arbitrary asymptotic light source direction, the required normal vector rotation might cause normal vectors to get \( n_z < 0 \). Therefore some care should be taken that:

- the asymptotic light source direction should not be too different from the average direction \( S - a \) for vertices \( a \) and finite light source position \( S \). For instance, when a scene with centre of gravity \( C \) is illuminated by a light source at position \( S \), the asymptotic light direction could be

\(^1\)Observe that the Phong algorithm itself is totally empirical and is not intended for physically accurate rendering
taken \( k = \frac{s - C}{|s - C|} \). This choice will typically cause only relatively small normal vector rotations.

- rotating a normal vector should not cause a sign change of \( n_z \). If such a sign change is about to happen, the rotation angle should be made smaller. In this way, the rotated normal vectors \( n' \) are forced to stay always within the part of the table with \( n_x^2 + n_y^2 < m^2 \).

In general, to avoid possible difficulties with very shallow normal vector angles (i.e. small \( n_z \)), the rotation angle of section 2 could be made slightly smaller\(^2\) than the angle between \( h \) and \( h' \).

Introducing a table which contains samples of \( R(n_x, n_y) \) for discrete \( n_x \) and \( n_y \) seems liable to produce Mach bands. However, Mach bands typically occur if significant jumps in the intensity occur with sufficient space between adjacent jumps. Due to the nature of the intensity distribution in specular reflection, either the differences between intensity values in adjacent pixels are small (far away from a highlight), or large intensity jumps occur with little room between subsequent jumps (within a highlight). As a result, Mach bands turn out to be not a real problem with our method for values of \( m \) of 128 or larger.

\(^2\)This results in highlights that become somewhat larger. Also, the 100% match between our algorithm and the Phong algorithm in the vertices is lost with this modification.
References

