An Improved Illumination Model for Shaded Display

Turner Whitted

Bell Laboratories
Holmdel, New Jersey 07733

ABSTRACT

To accurately render a scene, global illumination information that affects the intensity of each pixel of the image must be known at the time the intensity is calculated. In a simplified form, this information is stored in a tree of "rays" extending from the viewer to the first surface encountered and from there to other surfaces and to the light sources. The visible surface algorithm creates this tree for each pixel of the display and passes it to the shader. The shader then traverses the tree to determine the intensity of the light received by the viewer. Consideration of all of these factors allows the shader to accurately simulate true reflection, shadows, and refraction as well as the effects simulated by conventional shaders. Anti-aliasing is included as an integral part of the visibility calculations. Surfaces displayed include curved as well as polygonal surfaces.

KEY WORDS AND PHRASES: computer graphics, computer animation, visible surface algorithms, shading, raster displays

CR CATEGORIES: 8.2

Introduction

Since its beginnings, shaded computer graphics has progressed toward greater realism. Even the earliest visible surface algorithms included shaders that simulated such effects as specular reflection [2], shadows [1,3], and transparency [4]. The importance of illumination models is most vividly demonstrated by the realism produced with newly developed techniques [7,9,12,131.

The role of the illumination model is to determine how much light is reflected to the viewer from a visible point on a surface as a function of light source direction and strength, viewer position, surface orientation, and surface properties. The shading calculations can be performed on three scales: microscopic, local, and global. Although the exact nature of reflection from surfaces is best explained in terms of microscopic interactions between light rays and the surface [8], most shaders produce excellent results using aggregate local surface data. Unfortunately, these models are usually limited in scope, i.e. they look only at light source and surface orientations while ignoring the overall setting in which the surface is placed. The reason that shaders tend to operate on local data is that traditional visible surface algorithms cannot provide the necessary global data.

A shading model is presented here that uses global information to calculate intensities. Then, to support this shader, extensions to a ray tracing visible surface algorithm are presented.

Conventional Models

The simplest visible surface algorithms use shaders based on Lambert's cosine law. The intensity of the reflected light is proportional to the dot product of the surface normal and the light source direction, simulating a perfect diffuser and yielding a reasonable looking approximation to a dull, matte surface. A more sophisticated model is the one devised by Bui-Tuong Phong [6]. Intensity from Phong's model is given by

$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}'_j)^n \qquad (1)$$

where

- I = the reflected intensity
- I_a = reflection due to ambient light
- k_d = diffuse reflection constant
- $\bar{N}$ = unit surface normal
- $\bar{L}_j$ = the vector in the direction of the jth light source
- k_s = the specular reflection coefficient
- $\bar{L}'_j$ = the vector in the direction halfway between the viewer and the jth light source, and
- n = an exponent that depends on the glossiness of the surface.

Phong's model assumes that each light source is located at a point infinitely distant from the objects in the scene. The model does not account for objects within a scene acting as light sources or for light reflected from object to object. As noted in [10], this drawback doesn't affect the realism of diffuse reflection components very much,

but it seriously hurts the quality of specular reflections. A method developed by Blinn and Newell [7] partially solves the problem by modelling an object's environment and mapping it onto a sphere of infinite radius. The technique yields some of the most realistic computer generated pictures ever made, but its limitations preclude its use in the general case.

In addition to the specular reflection, the simulation of shadows is one of the more desireable features of an illumination model. A point on a surface lies in shadow if it is visible to the viewer but not visible to the light source. Some methods [12,13] invoke the visible surface algorithm twice, once for the light source and once for the viewer. Others [1,3,14] use a simplified calculation to determine whether the point is visible to the light source.

Transmission of light through transparent objects has been simulated in algorithms that paint surfaces in reverse depth order [4]. When painting a transparent surface the background is only partially overwritten, allowing previously painted portions of the image to show through. While the technique has produced some impressive pictures, it does not simulate refraction.

Improved Model

A simple model for reflection of light from perfectly smooth surfaces is provided by classical ray optics. As shown in Figure 1, the light intensity, I, passed to the viewer from a point on the surface consists primarily of the specular reflection, S, and transmission, T, components. These intensities represent light propagated along the $\bar{V}$, $\bar{R}$, and $\bar{P}$ directions respectively. Since surfaces displayed are not always perfectly glossy, a term must be added to model the diffuse component as well. Ideally the diffuse reflection should contain components due to reflection of nearby objects as well as predefined light sources, but the computation required to model a distributed light source is overwhelming. Instead, the diffuse term from (1) is retained in the new model. Then the new model is

$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s S + k_t T \qquad (2)$$

where

S = the intensity of light incident from the $\bar{R}$ direction
k_t = the transmission coefficient, and
T = the intensity of light from the $\bar{P}$ direction.

The coefficients k_s and k_t are held constant for the model used to make pictures in this report, but for the best accuracy they should be functions that incorporate an approximation of the Fresnel reflection law (i.e. the coefficients should vary as a function of incidence angle in a manner that depends on the material's surface properties). In addition, these coefficients must be carefully chosen to correspond to physically reasonable values if realistic pictures are to be generated. The $\bar{R}$ direction is determined by the simple rule that the angle

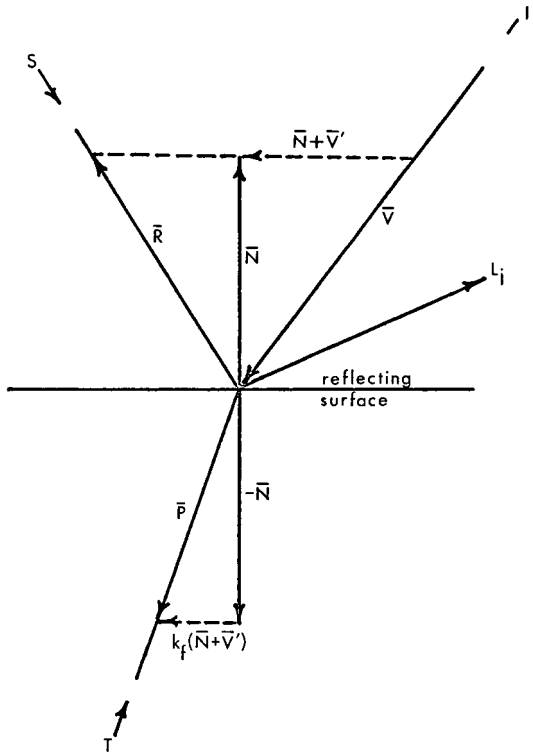

Figure 1

of reflection must equal the angle of incidence. Similarly, the $\bar{P}$ direction of transmitted light must obey Snell's law. Then, $\bar{R}$ and $\bar{P}$ are functions of $\bar{N}$ and $\bar{V}$ given by

$$\bar{V}' = \frac{\bar{V}}{|\bar{V} \cdot \bar{N}|}$$

$$\bar{R} = \bar{V}' + 2\bar{N}$$

$$\bar{P} = k_f(\bar{N} + \bar{V}') - \bar{N}$$

where

$$k_f = (k_n^2 |\bar{V}'|^2 - |\bar{V}' + \bar{N}|^2)^{-1/2}$$

and k_n = the index of refraction.

Since these equations assume that $\bar{V} \cdot \bar{N}$ is less than zero, the intersection processor must adjust the sign of N so that it points to the side of the surface from which the intersecting ray is incident. It must likewise adjust the index of refraction to account for the sign change. If the denominator of the expression for k_f is imaginary, T is assumed to be zero because of total internal reflection.

By making k_s smaller and k_d larger, the surface can be made to look less glossy. However, the simple

model will not spread the specular term as Phong's model does by reducing the specular exponent *n*. As pointed out in [8], the specular reflection from a roughened surface is produced by microscopic mirror-like facets. The intensity of the specular reflection is proportional to the number of these microscopic facets whose normal vector is aligned with the mean surface normal value at the region being sampled. To generate the proper looking specular reflection a random perturbation is added to the surface normal to simulate the randomly oriented micro-facets. (A similar normal perturbation technique is used by Blinn [9] to model texture on curved surfaces.) For a glossy surface, this perturbation has a small variance; with greater variances, the surface will begin to look less glossy. This same perturbation will cause a transparent object to look progressively more frosted as the variance is increased. While providing a good model for microscopic surface roughness, this scheme relies on sampled surface normals and will show the effects of aliasing for larger variances. Since this scheme also requires entirely too much additional computing, it is avoided whenever possible. For instance in the case of specular reflections caused directly by a point light source, Phong's model is used at the point of reflection instead of the perturbation scheme.

The simple model approximates the reflection from a single surface. In a scene of even moderate complexity, light will often be reflected from several surfaces before reaching the viewer. For one such case, shown in Figure 2, the components of the light reaching the viewer from point A are represented by the tree in Figure 3. Creating this tree requires calculating the point of intersection of each component ray with the surfaces in the scene. The calculations require that the visible surface algorithm (described in the next section) be called recursively until all branches of the tree are terminated. For the case of surfaces aligned in such a way that a branch of the tree has infinite depth, the branch is truncated at the point where it exceeds the allotted storage. Degradation of the image from this truncation is not noticeable.

In addition to rays in the $\bar{R}$ and $\bar{P}$ direction, rays corresponding to the $\bar{L}_j$ terms in (2) are associated with each node. If one of these rays intersects some surface in the scene before it reaches the light source, the point of intersection represented by the node lies in shadow with respect to that light source. That light source's contribution to the diffuse reflection from the point is then attenuated.

After the tree is created, the shader traverses the tree, applying equation (2) at each node to calculate intensity. The intensity at each node is then attenuated by a linear function of the distance between intersection points on the ray represented by the node's parent before it is used as an input to the intensity calculation of the parent. (Since one cannot always assume that all the surfaces are planar and all the light sources are point sources, square law attenuation is not always appropriate. Instead of modelling each unique situation, linear attenuation with distance is used as an approximation.)

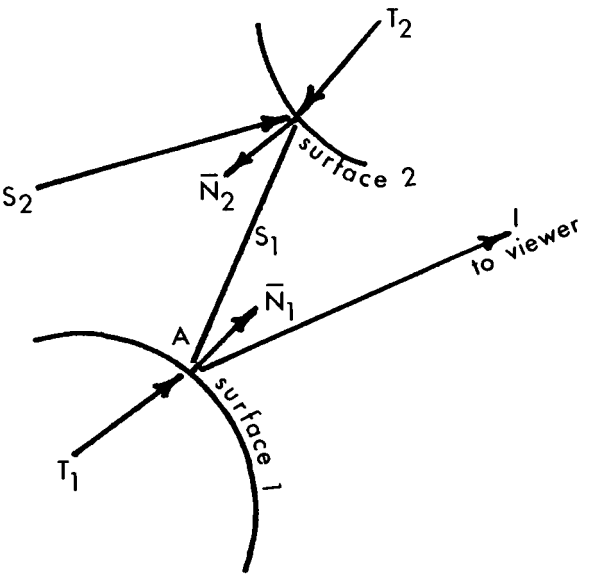

Figure 2

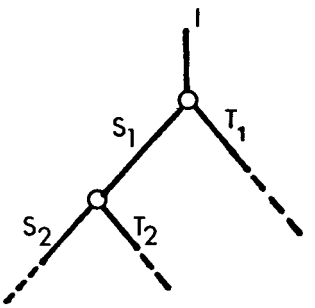

Figure 3

Visible Surface Processor

Since illumination returned to the viewer is determined by a tree of "rays", a ray tracing algorithm is ideally suited to this model. In an obvious approach to ray tracing, light rays eminating from a source are traced through their paths until they strike the viewer. Since only a few will reach the viewer, this approach is wasteful. In a second approach suggested by Appel [1] and used successfully by MAGI [5], rays are traced in the opposite direction - from the viewer to the objects in the scene.

Unlike previous ray tracing algorithms, the visibility calculations do not end when the nearest intersection of a ray with objects in the scene is found. Instead, each visible intersection of a ray with a surface produces more rays in the *R* direction, the *P* direction, and in the direction of each light source. The intersection process is repeated for each ray until none of the new rays intersects any object.

Because of the nature of the illumination model, some traditional notions must be discarded. Since objects may be visible to the viewer through reflections in other objects, even though some other object lies between it and the viewer, the measure of visible complexity in an image is larger than for a conventionally generated image of the same scene. For the same reason, clipping and eliminating backfacing surface elements are not applicable with this algorithm. Because these normal preprocessor stages that simplify most visible surface algorithms cannot be used, a different approach is taken. Using a technique similar to one described by Clark [11], the object description includes a bounding volume for each item in the scene. If a ray does not intersect the bounding volume of an object, then the object can be eliminated from further processing for that ray. For simplicity of representation and ease of performing the intersection calculation, spheres are used as the bounding volumes.

Since a sphere can serve as its own bounding volume, initial experiments with the shading processor used spheres as test objects. For non-spherical objects, additional intersection processors must be specified whenever a ray does intersect the bounding sphere for that object. For polygonal surfaces, the algorithm solves for the point of intersection of the ray and the plane of the polygon and then checks to see if the point is on the interior of the polygon. If the surface consists of bicubic patches, bounding spheres are generated for each patch. If the bounding sphere is pierced by the ray, then the patch is subdivided using a method described by Catmull and Clark [17] and bounding spheres are produced for each subpatch. The subdivision process is repeated until either no bounding spheres are intersected (i.e. the patch is not intersected by the ray) or the intersected bounding sphere is smaller than a predetermined minimum. This scheme is suggested for simplicity rather than efficiency; it is not implemented in the current program.

The visible surface algorithm also contains the mechanism to perform anti-aliasing. Since aliasing is the result of undersampling during the display process, the most straightforward cure is to low pass filter the entire image before sampling for display [15]. A considerable amount of computing can be saved, however, if a more economical approach is taken. Aliasing in computer generated images is most apparent to the viewer in three cases: 1) at regions of abrupt change in intensity such as the silhouette of a surface, 2) at locations where small objects fall between sampling points and disappear, and 3) whenever a sampled function (such as texture) is mapped onto the surface. The visible surface algorithm looks for these cases, and performes the filtering function only in these regions.

For this visible surface algorithm, a pixel is defined in the manner described in [16] as the rectangular region whose corners are four sample points as shown in Figure 4a. If the intensities calculated at the four points have nearly equal values, and no small object lies in the region between them, the algorithm assumes that the average of the four values is a good approximation of the intensity over the entire region. If the intensity values are not nearly equal (figure 4b), the algorithm subdivides the sample square and starts over again. This process runs recursively until the computer runs out of resolution or until an adequate amount of information about the detail within the sample square is recovered. The contribution of each single subregion is weighted by its area, and all such weighted intensities are summed to determine the intensity of the pixel. This approach amounts to performing a Warnock [2] type visibility process for each pixel. In the limit it is equivalent to area sampling, yet it remains a point sampling technique. A better method, currently being investigated, considers volumes defined by each set of four corner rays and applies a containment test for each volume.

To insure that small objects are not lost, a minimum radius (based on distance from the viewer) is allowed for bounding spheres of objects. This minimum is chosen so that no matter how small the object, its bounding sphere will always be intersected by at least one ray. If a ray passes within a minimum radius of a bounding sphere, but does not intersect the object, the algorithm will know to subdivide each of the four sample squares that share the ray until the missing object is found. Although adequate for rays that reach theviewer directly, this scheme will not always work fo rays being reflected from curved surfaces.

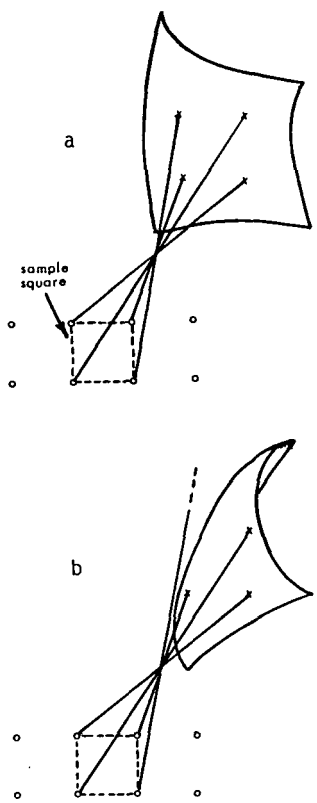

Figure 4

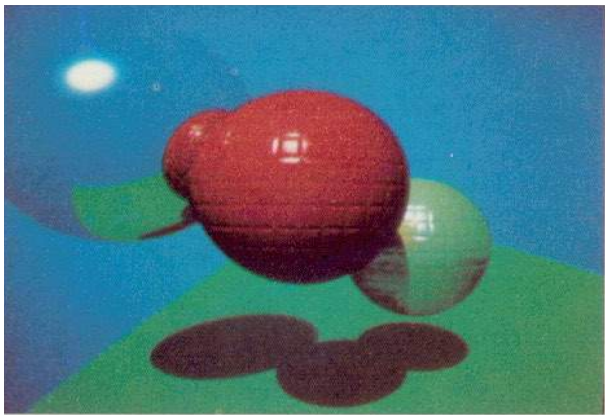

Figure 5

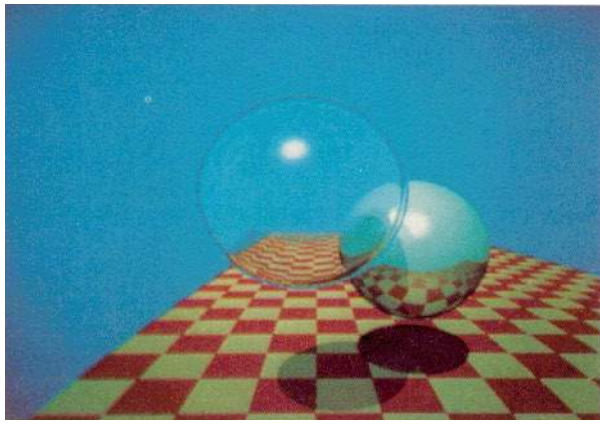

Figure 6

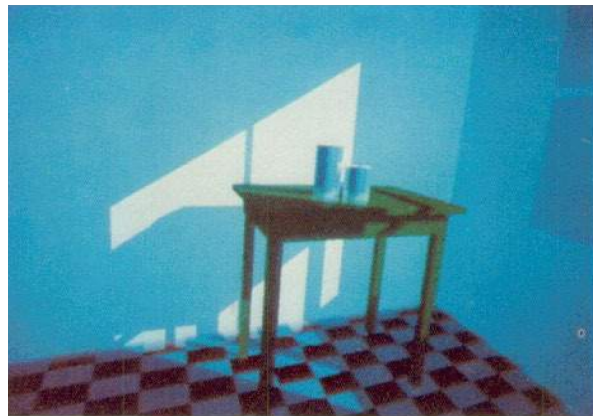

Figure 7

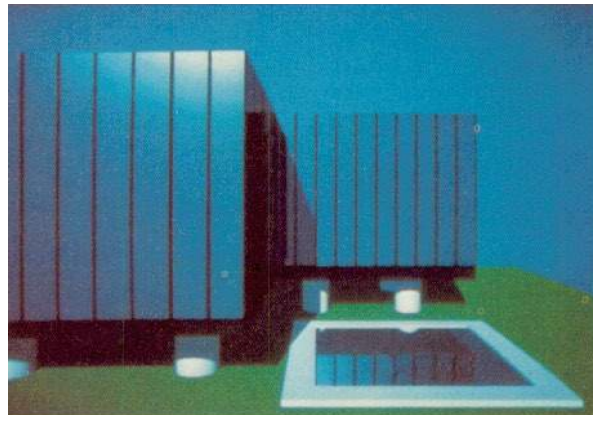

Figure 8

Results

A version of this algorithm has been programmed in C, running under UNIX™ on both a PDP- 11/45 and a VAX/11-780. To simplify the programming, all calculations are performed in floating point (at a considerable speed penalty). The pictures are displayed at a resolution of 480 by 640 pixels with 9-bits per pixel. Since only 3 bits of intensity resolution are available for each of the three primary colors, ordered dither [18] is applied to the color pictures to produce 111 effective levels per primary. Consequently, the pictures shown here are degraded by the dither pattern.

Table 1 lists user times for various pictures shown in this report. All times given are for the VAX which is nearly three times faster than the PDP-11/45 for this application. The image of figure 5 shows three glossy objects with shadows and object-to-object reflections. The texturing is added using Blinn's wrinkling technique. Figure 6 illustrates the effect of refraction through a transparent object. The algorithm has also been used to produce a short animated sequence. The enhancements provided by this illumination model are more readily apparent in the animated sequence than in the still photographs. A breakdown of where the program spends its time for simple scenes is:

Overhead	13%
Intersection	75%
Shading	12%

For more complex scenes the percentage of time required to compute the intersections of rays and surfaces increases to over 95%. Since the program makes almost no use of image coherence, these figures are actually quite promising. They indicate that a more efficient intersection processor will greatly improve the algorithm's performance. This distribution of processing times also suggests that a reasonable division of tasks between processors in a multiprocessor system is

TM UNIX is a trademark of Bell Laboratories.

Picture times	
Fig.	time(min)
5	44
6	74
7	122

Table 1.

to have one or more processors dedicated to intersection calculations with ray generation and shading operations performed by the host.

Summary

This illumination model draws heavily on techniques derived previously by Phong [6] and Blinn [7,8,9], but it operates recursively to allow the use of global illumination information. It is implemented through a visible surface algorithm that is very slow, but which shows some promise of becoming more efficient. When better ways of using picture coherence to speed the display process are found, this algorithm may find use in the generation of realistic animated sequences.

References

[1] Appel, A. Some techniques for shading machine renderings of solids. AFIPS 1968 SJCC, 37-45.

[2] Warnock, J.E. A hidden line algorithm for halftone picture representation. Technical Report TR 4-15, Computer Science Dept., University of Utah, 1969.

[3] Bouknight,W.K. and Kelley, K.C. An algorithm for producing half-tone computer graphics presentations with shadows and movable light sources. AFIPS 1970 SJCC,I-10.

[4] Newell, M.E., Newell,R.G., and Sancha, T.L., A solution to the hidden surface problem. Proc. ACM Annual Conf., 1972, 443-450.

[5] Goldstein, R.A. and Nagel, R. 3-D visual simulation. *Simulation,* January 1971, 25-31.

[6] Bui-Tuong Phong. Illumination for computer generated images. *Comm. ACM* 18,6(June 1975),311-317.

[7] Blinn, J.F. and Newell, M.E. Texture and reflection in computer generated images. *Comm. ACM* 19,10(October 1976), 542-547.

[8] Blinn, J.F. Models of light reflection for computer synthesized pictures. Proceedings of the 4th Annual Conf. on Computer Graphics and Interactive Techniques, 1977.

[9] Blinn, J.F. Simulation of wrinkled surfaces. Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques, 1978.

[10] Blinn, J.F. and Newell, M.E. The progression of realism in computer generated images. Proc. of the ACM Annual Conf., 1977, 444-448.

[11] Clark, J.H. Hierarchical geometric models for visible surface algorithms. *Comm. ACM* 19,10(October 1976), 547-554.

[12] Atherton, P., Weiler, K., and Greenburg, D. Polygon shadow generation. Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques, 1978.

[13] Williams, L. Casting curved shadows on curved surfaces. Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques, 1978.

[14] Crow, F.C. Shadow algorithms for computer graphics. Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques, 1977.

[15] Crow, F.C. The aliasing problem in computer-generated shaded images. *Comm. ACM20,11* (Nov. 1977), 799-805.

[16] Catmull, E. A subdivision algorithm for computer display of curved surfaces. UTEC CSc-74-133, Computer Science Dept., University of Utah, 1974.

[17] Catmull, E. and Clark, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design.* 10 6 (Nov 1978) pp 350-355.

[18] Jarvis, J.F., Judice, C.N., and Ninke, W.H. A survey of techniques for the display of continuous tone pictures on bilevel displays. *Comp. Graphics and Image Proc.,* 5, pp. 13-40, 1976.