# Fitting Virtual Lights For Non-Diffuse Walkthroughs

Bruce Walter Gün Alppay Eric Lafortune Sebastian Fernandez Donald P. Greenberg

Cornell Program of Computer Graphics *

## Abstract

This paper describes a technique for using a simple shading method, such as the Phong lighting model, to approximate the appearance calculated by a more accurate method. The results are then suitable for rapid display using existing graphics hardware and portable via standard graphics API's. Interactive walkthroughs of view-independent non-diffuse global illumination solutions are explored as the motivating application.

**CR Categories:** I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Shading

**Keywords:** interactive walkthroughs, non-diffuse appearance, global illumination, Phong shading

## 1 INTRODUCTION

This paper describes a method to take a view-independent non-diffuse global illumination solution and approximate it in a form that is suitable for rapid display and interactive walkthroughs. The method fits "virtual lights" to each object that, when displayed using a simple Phong lighting model, will closely reproduce its correct appearance.

One goal of realistic computer graphics is to let a viewer experience a virtual space as if they were physically present in a real space. There are many possible aspects to this mimicry, but here we will emphasize two facets. We want the viewer to be able to move about and explore the space in a natural and unrestricted way, and we want to match the appearance of the real space as closely as possible.

Real lighting is complex and subtle. Global illumination calculations are necessary if we hope to duplicate its appearance. These calculations are expensive, but if we are willing to restrict ourselves to a static environment, this part of the simulation can done as a pre-process. However, we still need to display the results rapidly if we want interactive walkthroughs. To accomplish this, we would like to leverage the existing 3D graphics hardware/software infrastructure.

Unfortunately, there is no standard format for storing non-diffuse lighting information; previously this has meant displaying a diffuse-only approximation to the actual appearance. While the results can be impressive, the absence of directionally dependent lighting effects, such as glossy highlights, means that important perceptual cues are missing.

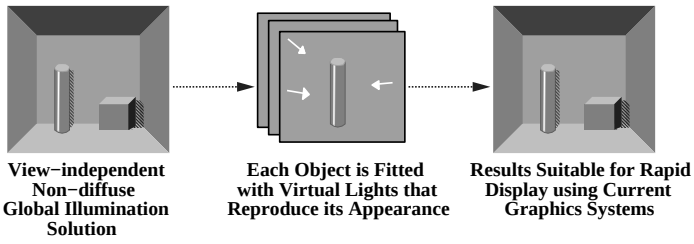


Figure 1: Approximation process.

The continuing popularity of the Phong [10] lighting model[1] is a testament to the importance of including such highlights. Most current graphics API's include a Phong-style lighting model for fast shading. These lighting models are much too simplistic to accurately compute global illumination, but we can still make use of them. Instead of viewing Phong as a lighting model, we can think of it as a set of "appearance basis functions" which can be used to approximately reproduce the results of a more accurate method.

The basic process is outlined in Figure 1. We start from a view-independent non-diffuse global illumination solution. For each non-diffuse object, we fit a set of "virtual lights" that, under the Phong lighting model, will reproduce its computed appearance as closely as possible. By utilizing directionally varying parts of the Phong model, the results will contain non-diffuse aspects of the original solution, although there will also be some loss of directional information due to the limitations of the Phong "basis functions". The results can then be displayed using a standard Phong lighting model.

The translated model can easily be displayed using standard graphics API's (e.g. OpenGL, VRML, or Direct3D) and can even be embedded in display lists. This makes the model portable and suitable for the existing highly optimized 3D graphics display systems. The results are also much more compact than the original global illumination solutions. Most importantly, we apply the lesson of the popular but physically impossible Phong lighting model: even fairly approximate highlights are better than none.

### 1.1 Related Work

Several researchers have proposed methods for generating and displaying view-independent non-diffuse global illumination solutions (e.g. [6, 11]). Practical application of such methods has so far been hampered by their high computational cost, large storage requirements, and slow display

*[bjw, gun, eric, spf, dpg]@graphics.cornell.edu
http://www.graphics.cornell.edu
1997



[1] In this paper we use the term Phong somewhat loosely to mean the Phong model, the Blinn-Phong model [3] or any similar simple direct lighting model.

speeds. We hope the methods presented here may help push them toward greater use.

Image-based techniques represent a very different route to non-diffuse walkthroughs. They store the illumination in a set of images instead of on surfaces. Image rendering algorithms such as [4, 7, 8] are then used to quickly interpolate new viewpoints from the precomputed images for a walkthrough. These methods offer some potential advantages, but it is not yet known how well they will scale to walkthroughs of larger environments. We consider them to be promising, but take a different approach here.

Environment or reflection maps [1, 5, 12] have long been used for the rapid display of directionally dependent effects. Their main difference from our work lies in their application. They are usually used as a small extension to a simplistic direct lighting model, whereas we are fitting our directional effects in order to reproduce the appearance computed by a physically-based method. In the future, environment maps may be used in a manner similar to our virtual lights.

Multi-pass rendering techniques are another way to perform walkthroughs with non-diffuse effects. They can implement a variety of extensions to the standard Phong lighting model such as shadows, mirror reflections, refraction, and translucency [2]. The results can be striking and they can handle dynamic environments, which is a major advantage. The problem is that the number of passes required per image increases rapidly with the number of lights and the number of lighting effects simulated. To keep the frame rate interactive, one is forced to limit the environment and choose a somewhat *ad hoc* lighting model.

## 2 OVERVIEW

Before our technique is used, we assume that a view-independent non-diffuse global illumination solution has been computed for the environment of interest. For each object, this solution will specify its appearance as the amount of light leaving (by emission and/or reflection) every point on the object and in each direction. For simplicity we will assume that this information is specified at a number of selected points which we will refer to as vertices.

An example of a directional light pattern leaving a vertex is shown in 2D at the left in Figure 2. Our goal is to reproduce this pattern using parts of the Phong lighting model. The Phong model allows us two kinds of basis functions: a diffuse or directionally invariant type and the "Phong lobes", or directionally dependent parts, which are caused by specific lights. The diffuse basis is commonly used to encode diffuse global illumination solutions. The new idea of this paper is to also use the "Phong lobes" to approximate non-diffuse appearance as illustrated in Figure 2.

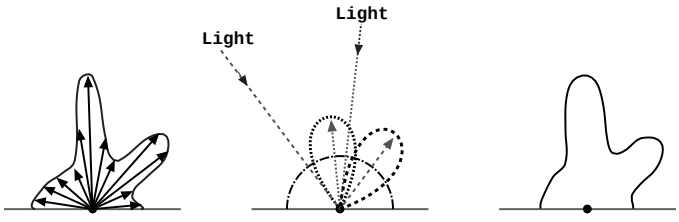


Figure 2: Directional light pattern leaving a vertex. Left: exact or computed pattern, Middle: diffuse basis and two "Phong lobe" basis functions, Right: approximated pattern using the basis functions.

We need to be aware of the many limitations in the Phong model. Some of these make perfect sense (e.g. limit on the number of active lights). Others are somewhat arbitrary and due to the fact that the designers were thinking of Phong as a lighting model rather than as "appearance basis functions". For instance, there is a specular exponent parameter which controls the width of the Phong lobes. We would like to use different exponents for different lights, and thus fit using lobes of several different sizes. We cannot because in the usual Phong lighting model, the exponent is a property of the surface and not a property of the lights.

Given these various restrictions, we must decide which parts and parameters will be the most useful. For each object, we have chosen to use a single set of directional light sources and a single specular exponent. Additionally, at each vertex we set a diffuse coefficient and a specular coefficient. Together, the exponent, light positions, and light intensities determine the shape of the specular basis function at each vertex as shown in Figure 3. The vertex coefficients then specify the mixture of the diffuse and specular basis functions which will serve as our approximation.

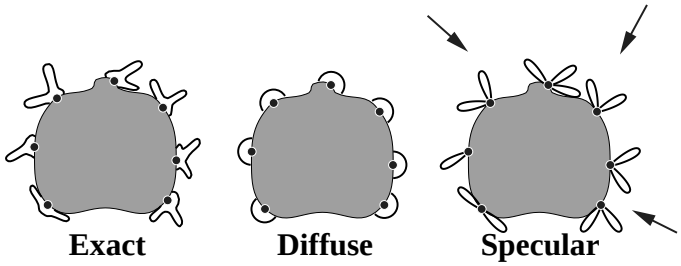


Figure 3: Directional light patterns at selected vertices on an object. Left: exact or computed patterns, Middle: diffuse basis function, Right: specular basis functions induced by three directional lights shown as arrows. Previous methods approximated the exact pattern using only the diffuse basis, while we use both the diffuse and specular.

Setting these parameters is a non-linear optimization problem. At first we tried using a general purpose non-linear optimization procedure, but found that this took a long time and often did not converge. Instead, we have developed a simple set of heuristics for choosing reasonable values. Further optimization could then be done using these values as the initial guess, although we do not currently do this. We iteratively perform a simple three stage fitting process, where a subset of the parameters are set in each stage.

For each object we start by assuming some value for the specular exponent which fixes the shape of the specular lobes, and iteratively fitting a set of lights. We find the brightest value among all vertices and directions on the object, and select the light direction that will create a Phong lobe centered in that direction for that vertex and the light intensity that will reproduce this maximum value (assuming the specular coefficient is 1.0 for now). The effect of this new light is subtracted from each vertex and the process is repeated until some maximum number of lights have been fit.

Once the exponent and lights are chosen, the shape of the specular basis functions is determined. The problem is now a linear optimization, and we set the two coefficients for each vertex using simple least squares fitting. Finally, we repeat this process with different values of the exponent and choose the exponent which gives the best fit in the least squares step.

Figure 4: An environment containing a teapot shown using virtual lights.

We cannot expect to achieve an exact fit, but this procedure guarantees there will be highlights in the places where the object has its brightest highlights. Note that each object gets its own set of "virtual lights" which do not affect other objects. These lights do not cast shadows and need not correspond to real lights in the environment. For example, several lights may be used to better simulate a highlight whose shape is different than that of a Phong lobe, or lights may correspond to an indirect light source such as the ceiling above a halogen light.

# 3 IMPLEMENTATION

For our implementation we have worked with OpenGL's version of the Phong shading model [9].

## 3.1 OpenGL's Lighting Model

OpenGL uses a simple lighting model to approximate the direct illumination of surfaces by light sources. This lighting model consists of four components: emitted, ambient, diffuse, and specular. These are intended to simulate, respectively: light emitted by a surface (glow), multiply reflected indirect lighting, diffusely reflected direct lighting, and specularly reflected glossy highlights from lights. Lights can be ambient, directional, positional, or spotlights and have ambient, diffuse and specular coefficients. OpenGL guarantees that at least eight lights are available. Surfaces have emitted, ambient, diffuse, and specular coefficients, and a shininess parameter that controls the size of the highlights.

In our implementation, we only use the emitted and specular components, along with directional light sources. The emitted, ambient, and diffuse components all produce directionally invariant lighting at a vertex and are thus redundant for our purposes. We use the emitted component to encode the diffuse part of our solution. The specular component is directionally varying and depends on the shininess of the material, the light direction relative to the surface, the surface normal, and the viewing direction relative to the surface.

Using only the emitted and specular components, the OpenGL lighting equation for determining vertex colors becomes:

$$\text{emitted} + \sum_{\text{lights}} \max(\mathbf{s}\cdot\mathbf{n},0)^{\text{shininess}} * \text{specular}_{\text{light}} * \text{specular}_{\text{vertex}}$$

where $\mathbf{n}$ is the vertex normal, and $\mathbf{s}$ is the vector obtained by adding the light direction and the view direction and normalizing.

Figure 5: Comparison of original data for the teapot (left) and our approximation using 8 lights (right) shown from three viewpoints.

## 3.2 The Fitting Process

We compute the initial data by computing a radiosity solution via density estimation [13] and then performing a gather at each vertex and storing the results for a discrete set of outgoing directions. The details are not important and many other methods are possible. For each object, we then need to find the emitted and specular values for each vertex, the directions and intensity values for its lights, and its shininess value. Our algorithm for a single object is:

```
Repeat
  Choose a shininess value
  Repeat
    Subtract effects of existing lights from input light data
    Find the maximum difference
    Add directional light to cause a highlight at this maximum
    Set light intensities to match input data at their maxima
  until all lights have been fit
  For each vertex
    Set emitted and specular values by least squares fitting
until shininess search is done
```

We search for the shininess (i.e. exponent) which minimizes the least squares error. We currently use a golden section search method which eliminates a portion of the search interval on each iteration.

# 4 RESULTS

We computed a global illumination solution for the environment shown in Figure 4 displayed using eight "virtual lights" per specular object. This scene contains the familiar Utah teapot which we use as a an example object to demonstrate our results. A comparison between the computed teapot and our fitted approximation with eight "virtual lights" is shown in Figure 5. The results are perceptually convincing overall although small differences can easily be seen. We also compare results when using fewer fitted lights in Figure 6.

The real test of our techniques is in walkthroughs of non-diffuse environments. We can only show images here, but

Figure 6: Comparison of original data for the teapot (left) and our approximation using 2 (middle) and 8 (right) virtual lights.

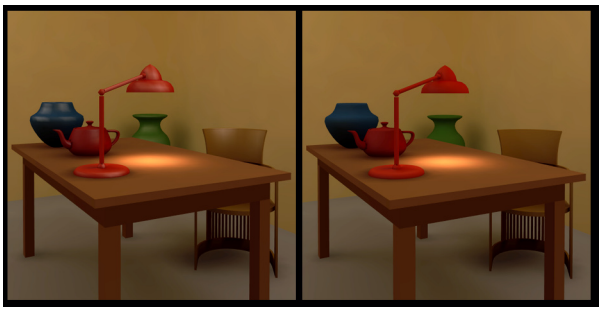

Figure 7: Our environment shown with virtual lights and diffuse only (virtual lights turned off).

we have included a live walkthrough of our environment in the video proceedings. Figure 7 shows images of this environment both with and without the virtual lights to demonstrate how much they contribute perceptually.

### 4.1 Limitations and Open Issues

While our results match the computed solutions surprisingly well, there are many limitations to how well we can currently mimic real appearance. Some of these are fundamental to the technique (e.g. mirrors simply require too much directional information), but others could be alleviated with changes in both our implementation and in the graphics display interfaces.

Some improvements, such as using positional instead of directional lights or finding a perceptually better fitting process, are possible now. But many others would require extensions or additions to the current graphics API's. Some potentially useful extensions would be the ability to vary the specular exponent per light and having a separate specular coefficient for each light at each vertex.

Gouraud interpolation is a major source of artifacts and requires that the curved surfaces be finely tessellated. True Phong shading would reduce these problems, but is rarely available because it is more computationally demanding.

While we use standard graphics API's, we use them in a hitherto unusual way. Many systems are not properly optimized for the sequence of operations we use. For instance on many OpenGL systems there is a very large ($> 4\times$) performance penalty for varying more than one property per vertex (in our case emitted and specular coefficients). We have achieved good performance using a two pass technique, one pass for the diffuse component and a second for the specular. Another possibility is to leave the specular coefficient fixed at the cost of some additional loss of quality. But this problem should largely disappear if our method gains acceptance and is considered during system optimization.

## 5 CONCLUSIONS

We have presented a technique for using a simple shading model, such as Phong, to approximate the non-diffuse appearance calculated by some more accurate method. This technique can translate view-independent non-diffuse global illumination solutions into a form that is more compact, portable, and suitable for fast display. This allows for non-diffuse walkthroughs which are perceptually better than traditional diffuse-only walkthroughs.

Moreover, by targeting our results toward standard graphics API's such as OpenGL, we can utilize the existing 3D graphics display infrastructure and allow designers to easily optimize their systems for our type of solutions. Finally, we have suggested a few ways in which future graphics API's could be enhanced to better enable the reproduction of non-diffuse appearance.

## Acknowledgments

Our thanks to Ben Trumbore, the modelers, and our anonymous reviewers. This work was supported by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and by NSF ASC-9523483 and performed with workstations generously donated by the Hewlett-Packard Corporation.

## References

[1] J. Blinn and M. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, October 1976.

[2] P. J. Diefenbach. *Pipeline Rendering: Interaction and Realism through Hardware-base Multi-pass Rendering.* PhD thesis, University of Pennsylvania, 1996.

[3] A. S. Glassner. *Principles of Digital Image Synthesis.* Morgan-Kaufman, San Francisco, 1995.

[4] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. *Computer Graphics*, pages 43–54, August 1996. ACM Siggraph '96 Conference Proceedings.

[5] N. Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, Nov. 1986.

[6] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics*, 20(4):133–142, August 1986. ACM Siggraph '86 Conference Proceedings.

[7] M. Levoy and P. Hanrahan. Light field rendering. *Computer Graphics*, pages 31–42, August 1996. ACM Siggraph '96 Conference Proceedings.

[8] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, pages 39–46, August 1995. ACM Siggraph '95 Conference Proceedings.

[9] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide.* Addison-Wesley, New York, 1993.

[10] B.-T. Phong. Illumination for computer generated images. *Communications of the ACM*, 18(6):311–317, June 1975.

[11] F. X. Sillion, J. Arvo, S. Westin, and D. Greenberg. A global illumination algorithm for general reflection distributions. *Computer Graphics*, 25(4):187–196, July 1991. ACM Siggraph '91 Conference Proceedings.

[12] D. Voorhies and J. Foran. Reflection vector shading hardware. *Computer Graphics*, 28(3):163–166, July 1994. ACM Siggraph '94 Conference Proceedings.

[13] B. Walter, P. M. Hubbard, P. Shirley, and D. P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, October 1997.