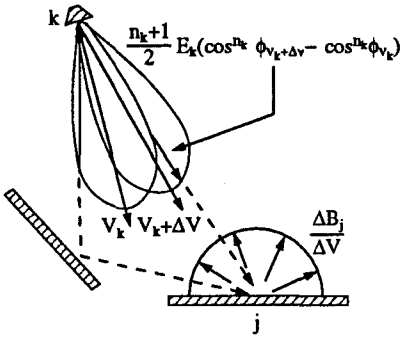



Figure 2: Estimation of $\partial B_j/\partial \mathbf{V}_k$ by shooting a delta emission from source $k$.

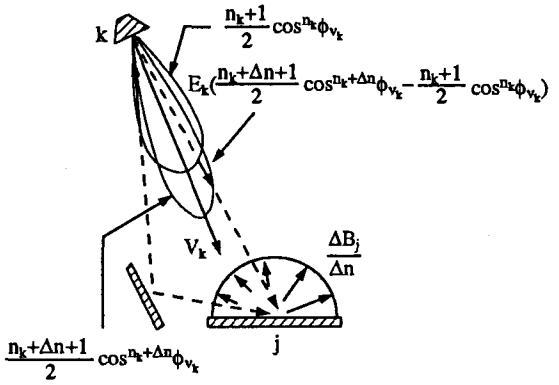



Figure 3: Estimation of $\partial B_j/\partial n_k$ by shooting a delta emission from source $k$.

by finite differences. A small "delta" emission, $\Delta E$, is shot from the variable emission light source as indicated in figure 1 and allowed to interreflect. The iterative shooting operations are very rapid since the links representing the form factors are precomputed during the baseline rendering.

The result of shooting a small amount of energy through the network of links results in an effect on each element radiosity, $\Delta B_j$, thus providing all the derivative estimates $\Delta B_j/\Delta E$. If the only free variables in the optimization are light emissions, these influence factors need only be evaluated once, due to linearity. On the other hand, if any spotlight directionality or element reflectance is allowed to be variable, light emission influence factors must be updated each iteration.

The partial derivative of the objective with respect to a variable element reflectivity is handled in a similar fashion. The element reflectivity $\rho_k$ is adjusted by a small delta $\Delta\rho$. The effect on all other elements can be evaluated by "shooting" the unshot radiosity due to the change in reflectivity: $B_k\Delta\rho$. As with light sources, several shooting iterations may be necessary to account for multiple bounce effects. Once convergence has been achieved, the effect of $\Delta\rho$ on element radiosity $\Delta B_j$ is available and the influence factor estimate $\Delta B_j/\Delta\rho$ can be recorded.

Influence factors for spotlight directionality variables, $\mathbf{V}_k$ and $n_k$, are also approximated through finite differences. For example, a small change, $\Delta\mathbf{V}$, can be made to the direction vector $\mathbf{V}_k$ and the effect on each element radiosity can be determined by a series of shooting steps. The first shooting step, illustrated in figure 2, shoots a delta emission from the modified spotlight to all other elements. The delta emission is determined according to the change in the directionality parameter, in this case, $E_k\frac{(n_k+1)}{2}(cos^{n_k}(\phi_{v_k+\Delta v}) - cos^{n_k}(\phi_{v_k}))$ where $\phi_{v_k}$ is the angle between the original direction vector of the light and the direction of the element and $\phi_{v_k+\Delta v}$ is the angle between the *new* spotlight direction vector and the direction of the element. Subsequent shooting steps proceed in the normal fashion in order to handle multiple bounce effects. The same technique can be used when the distribution pattern parameter $n_k$ is changed as illustrated in figure 3. In this case the radiosity cast is $E_k(\frac{(n_k+\Delta n+1)}{2}cos^{n_k}(\phi_{v_k}) - \frac{(n_k+1)}{2}cos^{n_k}(\phi_{v_k}))$.

The cost functions that measure patterns of light or subjective impressions are defined in terms of perception. The partial derivatives of the functions examining lighting patterns with respect to light emission $E_k$ are:

$$\frac{\partial f_{brightness}}{\partial E_k} = -\frac{\sum_i \frac{\partial P_i}{\partial E_i} A_i}{\sum_i A_i}$$

$$\frac{\partial f_{non-uniform}}{\partial E_k} = -\left[\frac{\sum_i (P_{avg,i} - P_i)^2 A_i}{\sum_i A_i}\right]^{-\frac{1}{2}} * \left[\frac{\sum_i (P_{avg,i} - P_i)\left(\frac{\partial P_{avg,i}}{\partial E_k} - \frac{\partial P_i}{\partial E_k}\right)}{\sum_i A_i}\right]$$

$$\frac{\partial f_{peripheral}}{\partial E_k} = \left[\frac{\sum_i \frac{\partial P_i}{\partial E_k} A_i}{\sum_i A_i} - \frac{\sum_j \frac{\partial P_j}{\partial E_k} A_j}{\sum_j A_j}\right]$$

The partials of the subjective impressions are just a linear combination of the partial derivatives of $f_{brightness}$, $f_{non-uniform}$, and $f_{peripheral}$.

The partials $\partial P_j/\partial E_k$ are derived by differentiating equation 2 giving,

$$\frac{\partial P_j}{\partial E_k} = 10^{\gamma}\left[\frac{aa}{B_j}\frac{\partial B_j}{\partial E_k} + \frac{\partial \alpha}{\partial E_k}\zeta\right] \quad (5)$$

where $\alpha$ is the adaption level which can be approximated by $(\sum_i log_{10}(B_i/10,000)A_i)/\sum_i A_i$, $\gamma$ is $aa * log_{10}(B_i/10,000) + bb$, and $\zeta$ is $0.4log_{10}(B_i/10,000) - ln(10)(0.8\alpha + 2.6)$.

If the the adaptation level is assumed constant with respect to a change in emission $E_k$, $\partial\alpha/\partial E_k = 0$, otherwise

$$\frac{\partial \alpha}{\partial E_k} = \frac{A_j}{B_j ln(10)\sum_i(A_i)}\frac{\partial B_j}{\partial E_k}$$

### 3.4 Optimization

The optimization process uses the BFGS algorithm, which evaluates the objective function and gradient at a current step in the design space in order to compute a search direction. Once a search direction is derived, a line search is performed in this direction. Each step in the line search involves a reevaluation of the objective function, hence a reevaluation of the element radiosities which are displayed, allowing the user to watch the progress of the optimization. This process is repeated until the system has converged to a minimum.

## 4 Experiences and Results

The first implementation of the Radioptimization system allowed an objective function based only on photometric measures and did not take into account the psychophysical properties of lighting. The system could successfully optimize lighting but required quite a bit of unintuitive "tweaking" of the objective function weights in order to achieve lighting that had the right subjective appearance. These early experiences led to the investigation of the psychophysical objective functions.

Figure 6 shows the effects that the subjective impressions have on an optimization. The top image constrains the table to have a small amount of illumination while conserving energy and creating an overall impression of visual clarity. To improve efficiency the optimization was run at a low resolution on a simplified model, without the chairs and television set. The optimization process took 1 minute and 21 seconds on an IBM Model 550 RISC System 6000. The bottom image has the same design goals as the top image except that it tries to elicit an impression of privateness. This optimization took 2 minutes and 11 seconds.

It took two or three hours of performing design iterations before developing an intuitive "feel" for the optimization process and the effects of the weights on the objective function. One of the problems with the design cycle is that there may be local minima of the specified objective that are visually unattractive. For example, in addition to the design goals mentioned above for figure 6, we needed to add an additional constraint limiting the illumination of the ceiling because pointing the lights directly at the ceiling was an optimal way of increasing the overall brightness of the room.

One drawback of the system at this point is that it is not fast enough to allow a highly interactive feedback cycle for complex models. However since the system allows a designer to think in terms of their own design goals, it requires fewer design iterations to achieve the desired result.

## 5 Conclusions

This paper has presented a new method of designing illumination in a computer simulated environment, based on goal directed modeling. A library of functions were developed that approximate a room's success in meeting certain lighting design goals such as minimizing energy or evoking an impression of privacy. The objective functions were developed through an experiment in which subjects ordered a set of images according to a particular impression. Processing this data with INDSCAL, showed a correlation between quantitative lighting patterns and subjective measures of visually clarity, pleasantness, and privacy. Once the lighting design goals have been set, the software system searches the space of lighting configurations for the illumination pattern that "best" meets the design specifications. The system absorbs much of the burden for searching the design space allowing the user to focus on the goals of the illumination design rather than the intricate details of a complete illumination specification.

The radioptimization system explores only one possible path in the application of optimization techniques to image synthesis design problems. Constrained optimization techniques may be more suitable than the unconstrained penalty method technique used here when the design goals must be satisfied precisely. Discrete optimization methods may be appropriate in some instances, for example when emissivities are constrained to a finite set, *e.g.* {60 Watts ,100 Watts ,$\cdots$}. Geometric properties of the model, such as the position of the lights or the size and position of the windows, could be allowed as free variables. More general image synthesis methods could be applied to account for non-diffuse effects such as glare.

## Acknowledgments

Pat Hanrahan, Larry Aupperle and David Salzman provided the radiosity software used as a basis for this work. Greg Ward offered suggestions on useful objective functions for lighting design. Shinichi Kasahara participated in discussions about the work as it developed.

The first and second author's work was supported by NFS (CCR-9210587). All opinions, findings, conclusions, or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

## References

[1] J. J. Chang and J. D. Carroll. How to use INDSCAL: a computer program for canonical decomposition of N-way tables and individual differences in multidimensional scaling. Technical report, Bell Telephone Laboratories, 1972.

[2] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):75–82, July 1988.

[3] M. F. Cohen and D. P. Greenberg. The hemi-cube: A radiosity for complex environments. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):31–40, July 1985.

[4] J. E. Flynn. A study of subjective responses to low energy and nonuniform lighting systems. *Lighting Design and Application*, Feb. 1977.

[5] J. E. Flynn, C. Hendrick, T. J. Spencer, and O. Martyniuk. A guide to methodology procedures for measuring subjective impressions in lighting. *Journal of the IES*, Jan. 1979.

[6] J. E. Flynn, T. J. Spencer, O. Martyniuk, and C. Hendrick. Interim study of procedures for investigating the effect of light on impression and behavior. *Journal of the IES*, Oct. 1973.

[7] P. E. Green, F. J. Carmone, Jr., and S. M. Smith. *Multidimensional Scaling Concepts and Applications.* Smith, Allyn, and Bacon, 1989.

[8] P. Hanrahan, D. Salzman, and L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.

[9] J. T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, Aug. 1986.

[10] H. N. McKay. Energy optimization and quality lighting design. *Lighting Design and Application*, Mar. 1986.

[11] P. Y. Papalambros and D. J. Wilde. *Principles of Optimal Design.* Cambridge University Press, Cambridge, England, 1988.

[12] P. Poulin and A. Fournier. Lights from highlights and shadows. *1992 Symposium on Interactive 3D Graphics*, pages 31–38, Mar. 1992.

[13] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes*. Cambridge University Press, New York, 1986.

[14] J. B. Rosen. The gradient projection method for non-linear programming, part i: Linear constraints. *SIAM*, 8:181–217, 1960.

[15] J. B. Rosen. The gradient projection method for non-linear programming, part ii: Non-linear constraints. *SIAM*, 9:514–532, 1961.

[16] P. C. Sorcar. *Architectural Lighting for Commercial Interiors*. John Wiley and Sons Inc., 1987.

[17] S. S. Stevens and J. C. Stevens. Brightness function: Effects of adaptation. *Journal of the Optical Society of America*, 53(3), Mar. 1963.

[18] J. Tumblin and H. Rushmeier. Tone reproductions for realistic computer generated images. Technical Report GIT-GVU-91-13, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, July 1991.

Figure 5: Sample interface which allows the user to set the weights of the objective and/or specify constraints.

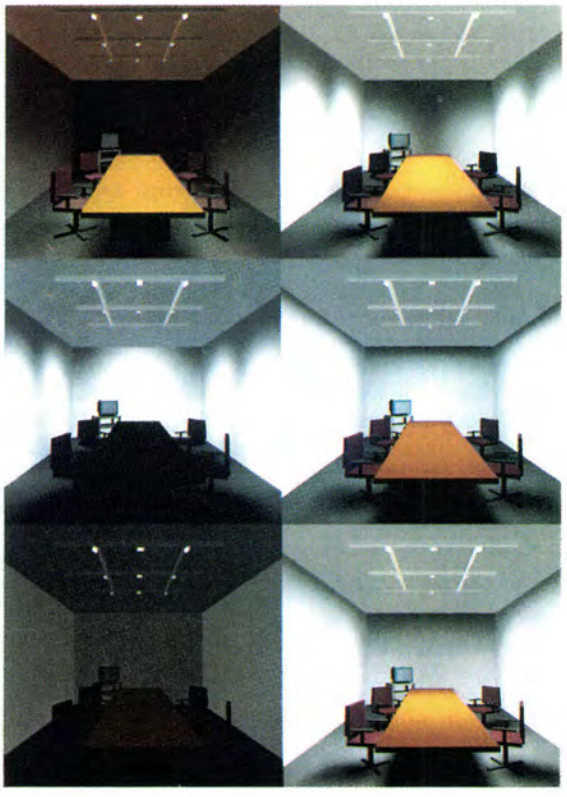


Figure 4: Computer generated rooms used to test subjects on which illumination patterns illicit particular subjective impressions.

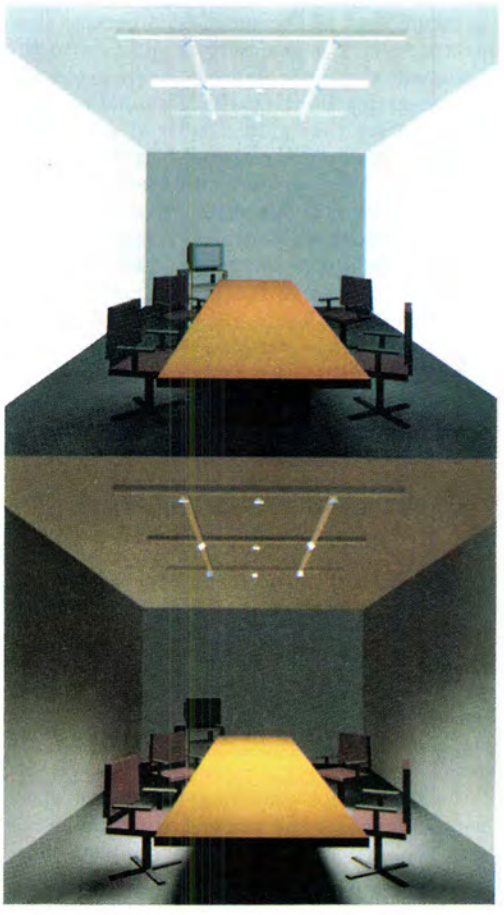


Figure 6: The top image constrains the table to have a small amount of illumination while preserving energy and creating an overall impression of visual clarity. The bottom image also constrains the table to have a small amount of illumination while preserving energy. In addition, it trys to create a feeling of privacy.

# A Hierarchical Illumination Algorithm for Surfaces with Glossy Reflection

Larry Aupperle Pat Hanrahan

Department of Computer Science
Princeton University

## Abstract

We develop a radiance formulation for discrete three point transport, and a new measure and description of reflectance: *area reflectance*. This formulation and associated reflectance allow an estimate of error in the computation of radiance across triples of surface elements, and lead directly to a hierarchical refinement algorithm for global illumination.

We have implemented and analyzed this algorithm over surfaces exhibiting glossy specular and diffuse reflection. Theoretical growth in light transport computation is shown to be $O(n+k^3)$ for sufficient refinement, where $n$ is the number of elements at the finest level of subdivision over an environment consisting of $k$ input polygonal patches — this growth is exhibited in experimental trials. Naive application of three point transport would require computation over $O(n^3)$ element-triple interactions.

**CR Categories and Subject Descriptors:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

**Key Words**: adaptive meshing, global illumination, radiosity, ray tracing.

## 1 Introduction

A major open problem in image synthesis is the efficient solution of the rendering equation. Radiosity methods have been quite successful over environments containing surfaces that exhibit only diffuse reflection. Unfortunately, very few materials are purely Lambertian reflectors, and efficient solution techniques have not yet been developed for more general specular or glossy reflection functions.

The rendering equation is an integral equation, and the solutions to complicated integral equations are generally obtained using either Monte Carlo or finite element techniques. Monte Carlo algorithms sometimes go under the name of distributed or stochastic ray tracing and are the most commonly employed in computer graphics (e.g. see [4, 5, 9, 12, 16]). Monte Carlo techniques have the advantage that they are easy to implement and can be used for complicated geometries and reflection functions. Unfortunately, their disadvantage is that they are notoriously inefficient. The second approach, the finite element method, has been very successfully applied to the rendering equation under the radiosity assumption, but has only begun to be employed in the general case, and with limited success. For example, Immel et al. [8] discretized radiance into a lattice of cubical environment maps, and solved the resulting system. More recently, Sillion et al. [13] used a mesh of spherical harmonic functions to represent radiance, and solved the resulting system using a shooting algorithm.



There are many ways to parameterize the rendering equation, and each leads to a different choice of basis functions. In the transport theory community two techniques are common: directional subdivision (the method of discrete ordinates or $S_N$), and spherical harmonics ($P_N$). These two techniques roughly correspond to the methods of Immel et al. and Sillion et al., although many interesting variations are possible. Our approach is somewhat different, and based on Kajiya's original formulation of the rendering equation [9]. Under this formulation, the rendering equation is expressed in terms of three point transport. That is, the kernel of the integral expresses the transport of light from a point on the source to a point on the receiver, via a point on a reflector. Given this formulation, the three point rendering equation can be discretized over pairs of elements to form a linear system of equations. Solving this system yields the radiance transported between elements. Note that this approach is very similar to the radiosity formulation.

The problem with finite element methods is that the matrix of interactions is very large for interesting environments. For a given environment of $k$ input polygonal patches containing $n$ elements at the finest level of refinement, the three point discretization that we are proposing generates an $n^3$ matrix of interactions. However, in this paper we show that we can accurately approximate the $n^3$ reflectance matrix with $O(n + k^3)$ blocks, in a way very similar to our recent hierarchical radiosity algorithm [7]. In that paper we showed how the $n^2$ form factor matrix could be approximated with $O(n + k^2)$ blocks, resulting in a very efficient algorithm in both space and time. Although the results presented in this paper are preliminary, we believe a hierarchical finite element approach along these lines will ultimately lead to a fast, efficient algorithm.

In the following section we describe our application of the finite element element method to the three point rendering equation, yielding a radiance formulation for discrete transport. In Section 3 we present a simple adaptive refinement algorithm for computation over this formulation, and the iterative solution technique employed for the actual calculation of transport. In Section 4 we discuss our implementation of the algorithm over glossy reflection, and in Section 5 we present some experiments and results. An appendix to this paper contains details of our error analysis for discrete transport under the glossy model.

## 2 Discrete Three Point Transport

The algorithm presented in this paper operates through two functions: refinement of the environment to form a hierarchy of discrete interactions, patches and elements, and the actual computation of illumination over this hierarchy.

In this section we develop the basis for both discretization and transport. We derive a radiance formulation for three point transport, and a new measure and description of reflectance, *area reflectance*. This radiance formulation and associated reflectance provide a natural criterion for discretization under illumination and reflection, and allow both the computation of radiance across triples of individual surface elements, and the expression and computation of all light transport over all surfaces.

 ACM-0-89791-601-8/93/008/0155 $01.50

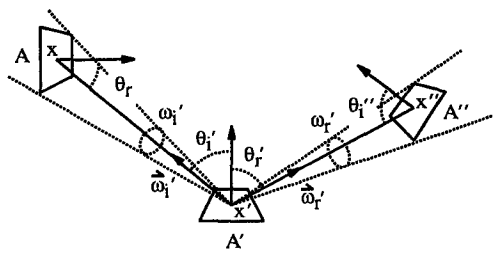



Figure 1: Geometry of Reflection

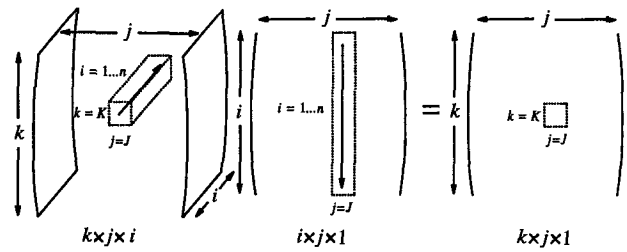



Figure 2: A Reflection Product

## 2.1 A Radiance Formulation for Three Point Transport

When computing and imaging illumination within an environment, we are interested in the transport of light from surface to surface — it is this interaction of surfaces that characterizes illumination, in the absence of participatory media. Reflection within an environment may thus be naturally expressed over triples of surfaces. Consider surfaces $A$, $A'$, and $A''$ (Figure 1) — we will examine the transport of light incident at $A'$ originating at $A$ and reflected toward $A''$.

Let $\omega_i'$ and $\omega_r'$ be the solid angles subtended at point $x'$ by $A$ and $A''$, respectively. Consider differential solid angles at $\vec{\omega}_i'$ and $\vec{\omega}_r'$ — by definition of the bidirectional reflectance-distribution function (BRDF), $f_r$ [11], the radiance $L(\vec{\omega}_r')$ along $\vec{\omega}_r'$ due to illumination through solid angle $\omega_i'$ is:

$$L(\vec{\omega}_r') = \int_{\omega_i'} f_r(\vec{\omega}_i', \vec{\omega}_r') L(\vec{\omega}_i') \cos\theta_i' d\omega_i'$$

Integrating this expression over $\omega_r'$, and introducing $\cos\theta_r'$, we have:

$$\int_{\omega_r'} L(\vec{\omega}_r') \cos\theta_r' d\omega_r' = \int_{\omega_r'} \int_{\omega_i'} f_r(\vec{\omega}_i', \vec{\omega}_r') L(\vec{\omega}_i') \cos\theta_i' \cos\theta_r' d\omega_i' d\omega_r'$$

We may then reparameterize over $A$ and $A''$ to yield:

$$\int_{A''} L(x', x'') G(x', x'') dx'' = \int_A \int_{A''} f_r(x, x', x'') L(x, x') G(x, x') G(x', x'') dx'' dx$$

where

$$G(x, x') = \frac{\cos\theta_r \cos\theta_i'}{|x - x'|^2} v(x, x')$$

where $v(x, x')$ is 1 if points $x$, $x'$, are mutually visible, and 0 otherwise. Note that $G$ is very similar to a differential form factor.

We integrate over $A'$, thus introducing all three areas into the formulation:

$$\int_{A'} \int_{A''} L(x', x'') G(x', x'') dx'' dx' = \int_A \int_{A'} \int_{A''} f_r(x, x', x'') L(x, x') G(x, x') G(x', x'') dx'' dx' dx \quad (1)$$

We may now rewrite the equation in discrete form. Let $A_j$ and $A_k$ be subareas of $A'$ and $A''$ such that $L(x', x'')$ is nearly constant over their surfaces. The left side of equation (1) may then be rewritten, bringing radiance out of the integral as $L_{jk}$:

$$L_{jk} \int_{A_j} \int_{A_k} G(x', x'') dx'' dx' = \pi L_{jk} A_j F_{jk}$$

by definition of the diffuse form factor, $F_{jk}$.

We may similarly discretize $A$ as $A_i$, and rewrite the right side of equation (1) as:

$$L_{ij} \int_{A_i} \int_{A_j} \int_{A_k} f_r(x, x', x'') G(x, x') G(x', x'') dx'' dx' dx = \sum_i \pi L_{ij} A_i F_{ij} R_{ijk}$$

where $R_{ijk}$ is defined such that

$$\pi A_i F_{ij} R_{ijk} = \int_{A_i} \int_{A_j} \int_{A_k} f_r(x, x', x'') G(x, x') G(x', x'') dx'' dx' dx$$

Note that, by the symmetry of $f_r$ and $G$:

$$A_i F_{ij} R_{ijk} = A_k F_{kj} R_{kji}$$

We thus have:

$$\pi L_{jk} A_j F_{jk} = \pi \sum_i L_{ij} A_k F_{kj} R_{kji} = \pi \sum_i L_{ij} A_j F_{jk} R_{kji}$$

by the reciprocity of form factors, and thus:

$$L_{jk} = \sum_i L_{ij} R_{kji}$$

The three dimensional character of $R_{kji}$ over indices $i$, $j$, $k$ leads naturally to a three dimensional matrix formulation for the above system. Consider a product over an $n \times n \times n$ $R_{kji}$ "matrix" and an $n \times n \times 1$ $L_{ij}$ matrix producing an $n \times n \times 1$ matrix of reflected radiances, as shown in Figure 2. Note that the $R_{kji}$ matrix is of size $O(n^3)$ — the hierarchical method discussed in subsequent sections of this paper addresses more tractable representation of this matrix.

Taking into account emission, we have derived a radiance formulation for three point transport:

$$L_{jk} = E_{jk} + \sum_i L_{ij} R_{kji} \quad (2)$$

This formulation states that:

> *The radiance at Area j in the direction of Area k is equal to the radiance emitted by j in the direction of k, plus, for every Area i, the radiance at i in the direction of j multiplied by the area reflectance $R_{kji}$.*

Note that equation (2) is very similar to the radiosity formulation:

$$B_j = E_j + \rho_j \sum_i B_i F_{ji}$$

## 2.2 Area Reflectance

The quantity $R_{kji}$ has a natural and satisfying physical significance — it is an expression of reflectance over areas $A_i$, $A_j$, and $A_k$.

Consider the fraction of the radiant flux transported from $A_i$ incident to $A_j$ that is reflected in the direction of area $A_k$:

$$\frac{\int_{A_i}\int_{A_j}\int_{A_k} f_r(x,x',x'')L(x,x')G(x,x')G(x',x'')dx''dx'dx}{\int_{A_i}\int_{A_j} L(x,x')G(x,x')dx'dx}$$

If we assume that incident radiance is uniform and isotropic over both $\omega_i'$ (as induced by $A_i$) and $A_j$, we may divide through by $L(x,x')$, yielding:

$$\rho(A_i,A_j,A_k) \equiv \frac{\int_{A_i}\int_{A_j}\int_{A_k} f_r(x,x',x'')G(x,x')G(x',x'')dx''dx'dx}{\int_{A_i}\int_{A_j} G(x,x')dx'dx}$$

We define $\rho(A_i,A_j,A_k)$ to be *area reflectance*. Note that area reflectance is similar to biconical reflectance [11], save that it is also integrated over the reflecting surface.

By definition of $R_{ijk}$:

$$R_{ijk} = \rho(A_i,A_j,A_k)$$

Conservation of energy over reflection, and the reciprocity relation derived for $R_{ijk}$ above, constitute fundamental properties of area reflectance:

1. $\sum_k R_{ijk} \leq 1$, for fixed $i$, $j$.
2. $A_iF_{ij}R_{ijk} = A_kF_{kj}R_{kji}$.

where equality is achieved in property 1 over complete enclosures and perfect reflectivity.

## 2.3 Evaluation of $R_{kji}$

In this section we examine the evaluation of $R_{kji}$ over given patches $A_i$, $A_j$, $A_k$.

Recall:

$$R_{kji} = \frac{\int_{A_k}\int_{A_j}\int_{A_i} f_r(x'',x',x)G(x'',x')G(x',x)dxdx'dx''}{\int_{A_k}\int_{A_j} G(x'',x')dx'dx''}$$

We assume that discrete areas $A_i$, $A_j$, $A_k$ are of small enough scale that $f_r$ and $G$ are relatively constant over their surfaces. Then:

$$\begin{aligned} R_{kji} &= \frac{S_{kji}G_{kj}G_{ji}A_kA_jA_i}{G_{kj}A_kA_j} \\ &= S_{kji}G_{ji}A_i \end{aligned}$$

where $S$ is the discretized value of $f_r$, $S_{ijk} = S_{kji} = S_{x_kx_jx_i}$.

Note that the average value of $G(x',x)$ over $A_i$ and $A_j$ is $\pi F_{ji}/A_i$ — we thus estimate $G_{ji}A_i$ by $\pi F_{ji}$, and compute $R_{kji}$ as:

$$R_{kji} = \pi F_{ji}S_{kji}$$

In practice, it will not be possible to compute the exact values of $F_{ji}$ and $S_{kji}$ over $A_i$, $A_j$, $A_k$. We assume that we are able to estimate these values, along with error bounds for each estimation. Let $\Delta F_{ji}$ and $\Delta S_{kji}$ be error estimates for computed $F_{ji}$ and $S_{kji}$, respectively. We then have an estimate for area reflectance in the form:

$$\begin{aligned} R_{kji} &= \pi(F_{ji}+\Delta F_{ji})(S_{kji}+\Delta S_{kji}) \\ &= \pi(F_{ji}S_{kji} + \Delta F_{ji}S_{kji} + \Delta S_{kji}F_{ji} + \Delta F_{ji}\Delta S_{kji}) \\ &\approx \pi(F_{ji}S_{kji} + \Delta F_{ji}S_{kji} + \Delta S_{kji}F_{ji}) \end{aligned}$$

Assuming $\Delta F_{ji} < F_{ji}$, $\Delta S_{kji} < S_{kji}$, we have neglected the last term and estimate the error in $R_{kji}$ as $\pi(\Delta F_{ji}S_{kji} + \Delta S_{kji}F_{ji})$.

In general, and as is shown for glossy reflection in Section 4, the accuracy of estimators for $F_{ji}$ and $S_{kji}$ is dependent on the size of the patches over which reflectance is computed, relative to their distance apart. As relative size decreases, so does error in computation, leading directly to the adaptive refinement strategy for illumination presented in Section 3 below.

# 3 Algorithms for Three Point Transport

## 3.1 Introduction

Recall equation (2):

$$L_{jk} = E_{jk} + \sum_i L_{ij}R_{kji}$$

This equation suggests both a solution strategy for radiance under three point transport, and a natural representation for illumination within the solution system.

We may interpret equation (2) as a gathering iteration similar to that employed for radiosity under diffuse reflection: the radiance $L_{jk}$ at patch $A_j$ in the direction of patch $A_k$ is found by gathering radiances $L_{ij}$ in the direction of $A_j$ at patches $A_i$. We may solve for transport by gathering radiance for each $L_{jk}$, and successively iterating to capture all significant re-reflection.

We are left with the question of what structure we are gathering over and iterating upon. Note that all illumination is expressed as the radiance at a given patch in the direction of another — it is these patch-patch interactions that form the primary structure within the solution system. All operation is over interactions: both the representation and transport of radiance, and the iteration and solution for illumination.

Consider the following structure:

```
typedef struct _interaction {

    Patch *from;
    Patch *to;

    Color  L;
    Color  Lg;

    List  *gather;

    struct _interaction *nw, *sw, *se, *ne;

} Interaction;
```

A given interaction $ij$ is defined by two patches `ij->from` and `ij->to`, and represents the radiance at `from` in the direction of `to`. This radiance is stored within the interaction as attribute `L`. `Lg` is radiance gathered during the current solution iteration from interactions contained in the list `gather`. Subinteractions `nw`, `sw`, `se`, `ne` are the children of $ij$, induced by subdivision over either `from` or `to`. The structure assumes quadtree refinement, leaving northwest, southwest, southeast, and northeast descendants.

In the following sections we will present an algorithm for the refinement and computation of illumination over a hierarchy of interactions. The algorithm will operate by refining pairs of interactions $ij$, $jk$ (such that `ij->to == jk->from`), to ensure that computed reflectance across the interaction pairs, and associated patch triples, satisfies user specified error bounds. If a given interaction pair $ij$, $jk$ is satisfactory, the interactions are linked to record that radiance may be gathered from $ij$ to $jk$, otherwise one or both interactions are subdivided and refinement applied to their descendants.

After refinement, a gathering iteration may be carried out, each interaction gathering radiance from interactions to which it has been linked. The gathered radiances are then distributed within each receiving interaction hierarchy, and subsequent iterations computed until satisfactory convergence has been achieved.

Note that, within this system, the eye may be regarded as simply another object with which patches may interact. The radiance along interactions to the eye provides the resulting view.

## 3.2 Adaptive Refinement

Consider the following procedure:

```
Refine(Interaction *ij, Interaction *jk,
       float Feps, float Seps, float Aeps)
{
    float feps, seps;

    feps = GeometryErrorEstimate(ij);
    seps = ReflectionErrorEstimate(ij, jk);

    if (feps < Feps && seps < Seps)
        Link(ij, jk);

    else if (seps >= Seps) {

        switch(SubdivS(ij, jk, Aeps)) {

        case PATCH_I:

            Refine(ij->nw, jk, Seps, Feps, Aeps);
            Refine(ij->sw, jk, Seps, Feps, Aeps);
            Refine(ij->se, jk, Seps, Feps, Aeps);
            Refine(ij->ne, jk, Seps, Feps, Aeps);

            break;

        case PATCH_J:
            /* refine over children of ij and jk */
        case PATCH_K:
            /* refine over children of jk        */
        case NONE:
            Link(ij, jk);
        }
    }

    else {    /* feps >= Feps */

        switch(SubdivG(ij, jk, Aeps)) {

        /* refine over children, or link, as    */
        /* directed by PATCH_I, J, K, or NONE.  */

        }
    }
}
```

This procedure computes over pairs of interactions, and associated patch triples, subdividing and recursively refining if estimated error exceeds user specified bounds, linking the interactions for gathering if the bounds are satisfied, or if no further subdivision is possible. `Feps` and `Seps` are the bounds for geometric and reflection error, respectively; `Aeps` specifies the minimum area a patch may possess and still be subdivided. `GeometryErrorEstimate` and `ReflectionErrorEstimate` provide estimations for $\pi \Delta F_{ji} S_{kji}$ and $\pi \Delta S_{kji} F_{ji}$.

`SubdivS` and `SubdivG` control refinement for reflection and geometry error, respectively. Both routines select a patch for refinement, subdividing the patch and associated interaction(s) if required. An identifier for the selected patch is returned — if no patch may be subdivided, then `NONE` is passed back. Note that a given interaction/patch may be refined against many different interactions within the system, and thus may have already been subdivided when selected by a `Subdiv` routine — in this case, the routine simply returns the proper identifier.

The `Subdiv` routines should select for refinement patches that are of large size relative to their distance from their partner(s) in the transport triple. Form factor estimation is a convenient criterion for the determination of such patches — a large differential to area form factor $F_{dpq}$ indicates that patch $q$ is of large relative size. Care must be taken in subdivision, however, to ensure that each interaction is always subdivided in the same way for all refinements involving that interaction.

The `Subdiv` routines thus choose for refinement the patch of size at least `Aeps` that is of greatest form factor within $ij$ and/or $jk$ that will not induce multiple sets of children over either interaction. If patch $p_j$ is of greatest form factor over both $ij$ and $jk$, and of area greater than `Aeps`, then it is chosen for refinement (Figure 3 at middle). Otherwise, if $p_j$ is selected over one interaction, but $p_i$ or $p_k$ is selected over the other, then the "outside" patch is chosen for refinement. Given two selected outside patches, `SubdivS` selects the one of greater form factor relative to $p_j$; `SubdivG` selects $p_i$ over $p_k$, as $p_k$ has no direct effect on geometric accuracy. Note, however, that even under `SubdivG`, if only $p_j$ and $p_k$ are allowed subdivision, $p_k$ will be selected, although with further subdivision the triple will eventually balance sufficiently to allow refinement over $p_j$.

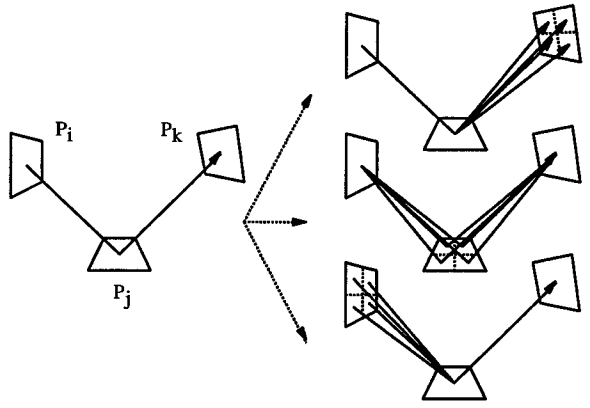



Figure 3: Refinement and Subdivision

## 3.3 Gathering Radiance

Gathering radiance over interactions may be written as a simple procedure:

```
Gather(Interaction *jk)
{
    Interaction *ij;

    if (jk) {

        jk->Lg = 0;

        ForAllElements(ij, jk->gather)
            jk->Lg += ij->L * Reflectance(ij, jk);

        Gather(jk->nw);
        Gather(jk->sw);
        Gather(jk->se);
        Gather(jk->ne);
    }
}
```

We gather radiance into `jk->Lg` rather than directly into `jk->L` to avoid the necessity of a push/pull with every invocation of the procedure (see Section 3.4). The solution method is thus simple Jacobi iteration, as opposed to Gauss-Seidel, as the hierarchical structure imposes simultaneous rather than successive displacement.

## 3.4 Radiance within a Hierarchy

A gathering iteration results in received radiance scattered throughout each interaction hierarchy. This gathered radiance must be distributed and accounted for over all ancestors and descendants of each receiving interaction, in order to maintain the consistency and correctness of the hierarchical representation of radiance between patches.

We employ a distribution algorithm similar to that presented in [7] for radiosity over patch/element hierarchies: gathered radiance is "pushed" to the leaf interactions within each hierarchy to ensure propagation to all descendants, and then "pulled" and distributed back up from the leaves through all higher level interactions to their common ancestor at the root. As is shown in [2], radiance may be pushed unchanged within the interaction hierarchy, and area averaged as it is pulled from child to parent.

# 4 Application over Glossy Reflection

In this section we discuss our implementation of the above algorithms over glossy reflection.

## 4.1 The Reflection Function

We employ a highly simplified Torrance-Sparrow [15] model for our glossy reflection function:

$$f_g(\vec{\omega}_i, \vec{\omega}_r) = \frac{\kappa + 2}{8\pi} \frac{\cos^\kappa \theta_m}{\cos\theta_i \cos\theta_r} sh(\theta_i, \theta_r)$$

This function incorporates the facet distribution function $\cos^\kappa \theta_m$ developed by Blinn [3], normalized for projected facet area under

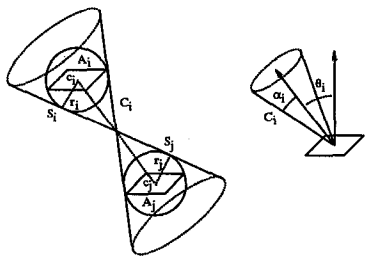


Figure 4: Estimating Cones

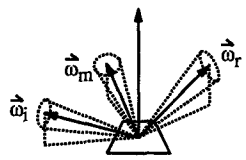


Figure 5: $C_i$, $C_r$, and $C_m$

[10]. Angle $\theta_m$ is that made to the mean surface normal by $\vec{\omega}_m$, the microfacet mirror orientation normal lying halfway between $\vec{\omega}_i$ and $\vec{\omega}_r$.

Function $sh(\theta_i, \theta_r)$ expresses self-shadowing over microfacets — for near specular surfaces, such self-shadowing or masking does not become critical until relatively high $\theta_i$ or $\theta_r$ [6]. The implemented system thus simply clamps $sh$ from 1 to 0 when $\theta_i$ or $\theta_r$ exceeds a preset $\theta_{\text{bound}}$ near the horizon. This scheme serves as a crude approximation to the shadowing function; however, a better strategy would be to employ a much fuller tabulation of the function, incorporated into the error analysis presented below. A more complete discussion of shadowing and conservation of energy over $f_g$ is presented in [2].

## 4.2 Error Estimation

Recall the general expression for error derived in Section 2.3:

$$\pi(\Delta F_{ji} S_{kji} + \Delta S_{kji} F_{ji})$$

In implementation we have estimated the form factor $F_{ji}$ by $F_{dji}$, the form factor from a differential area at $A_j$ to a disk of area $A_i$ centered at $A_i$, as was employed in [7]. As discussed in [7], the relative error in this estimate is proportional to the estimate itself. In our implementation we have thus estimated absolute error $\Delta F_{ji}$ as at most proportional to $F_{dji}^2$.A brief discussion of relative and absolute error over hierarchical methods is presented in [2].

We now consider the error estimate $\Delta S_{kji}$. As discussed in the appendix to this paper, we may compute bounding cones $C_i$, $C_r$, and $C_m$ over all possible incident, reflected, and mirror orientation directions induced at $A_j$ by $A_i$ and $A_k$ (Figures 4 and 5 — these figures are discussed more fully in the appendix). We may then compute maximum and minimum $\cos^\kappa \theta_m, \cos\theta_i, \cos\theta_r$ over these cones, and estimate error by interval width. The full expression for estimated error over transport is given in the appendix.

## 4.3 Clamping and Visibility

Evaluation of glossy reflectance over three surface areas, as required by the gather iteration, may be difficult, particularly if surface subdivision has been limited by Aeps rather than satisfaction of error bounds, and if $\kappa$, the facet distribution exponent, has high value. In this case we must estimate the integral of a spikey function over a relatively broad area.

Our solution is to band limit the BRDF in a fashion similar to that presented by Amanatides [1]. We employ the cone estimation techniques of the previous section to determine if the BRDF varies significantly over the given patches — if this variance exceeds a set bound, we "roughen" the reflecting surface, lowering $\kappa$ to broaden the resulting reflection over the estimated cones. We then renormalize the resulting blurred function, as described in [1], to

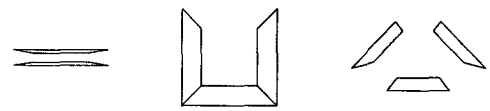


Figure 6: Geometric Configurations

prevent amplification of its low frequency components. We note that the resulting antialiasing is relatively aggressive, significantly dimming or eliminating reflections requiring overmuch blurring.

In implementation, we have computed visibility via jittered ray casting and inheritance similar to that of [7], storing visibility data in interactions as it is computed.

# 5 Results

## 5.1 Growth in Transport

We have measured the growth in transport triples (linked interactions) versus $n$, the maximum number of elements at the finest level of subdivision, over parallel, perpendicular, and "oriented" patches (Figure 7). The corresponding geometries are shown in Figure 6. The graphs show linear or near linear behavior over each range — the graph of triples vs. $n$ for the perpendicular case is slightly concave over the lower data points, but subsides to linear with further refinement.

In previous work [7] on hierarchical refinement for radiosity, it was shown that for error estimate proportional to $F_{dji}$, and sufficient refinement, each subpatch may only interact with other patches in a limited local neighborhood. As discussed in [7], each patch may thus participate in at most $c$ interactions, for some constant $c$ independent of $n$ and $k$. Adaptive refinement thus generates at most $O(n)$ transport interactions. We will show a similar bound for discrete three point transport under glossy reflection.

Recall that the estimate for error in computed transport is proportional to $\Delta F_{ji} S_{kji} + \Delta S_{kji} F_{ji}$. Our argument depends on two assumptions:

1. We may bound both $\Delta S_{kji}$ and $S_{kji}$ by some $S_{\max}$.

As discussed below, the lower this $S_{\max}$, the smaller the magnitude of the leading coefficient underling the resulting bound.

Note that our argument thus does not apply to perfect specular reflection, as the corresponding BRDF incorporates the Dirac delta function [11]. Equivalently, the argument does not hold over $f_g$ for $\kappa = \infty$ (inducing mirror reflection), as we can not provide a finite bound for $S$ in this case.

For finite $\kappa$, however, the desired bound over glossy reflection is achieved by:

$$\frac{\kappa + 2}{8} \max(\cos^\kappa \theta_m) \max(\sec\theta_i) \max(\sec\theta_r)$$

The maxima over the secant terms are bounded by microfacet self-shadowing.

2. $\Delta F_{ji}$ and $F_{ji}$ within our error estimate are at most proportional to $F_{dji}$.

Recall that we estimate $F_{ji}$ as $F_{dji}$, and $\Delta F_{ji}$ as $F_{dji}^2$, thus satisfying this assumption.

Given these assumptions, estimated error is at most proportional to $S_{\max} F_{dji}$.

We may now show $O(n)$ growth, for sufficient refinement. Consider refinement over interaction $ij$ under an error estimate at worst proportional to $S_{\max} F_{dji}$. The error estimate is thus proportional to $F_{dji}$, and therefore, for sufficient refinement, there are at most $O(n)$ such interactions, as discussed in [7].

Consider now an error satisfied link from $ij$ to an interaction $jk$. For sufficient refinement under our subdivision scheme, we may assume that form factors $F_{ij}$, $F_{ji}$, $F_{jk}$, $F_{kj}$ over $p_i, p_j$, and $p_k$ are roughly equal. Furthermore, these satisfying form factors depend only on the error estimate, reflection function, and error bounds, not on $n$ or $k$.

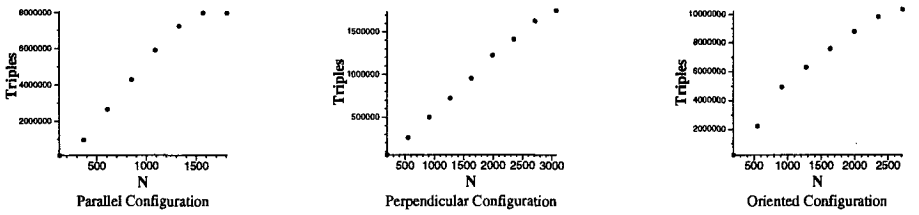



Figure 7: Triples vs. $N$ over Geometry. Error bounds $e = 0.1$. Glossy exponent $\kappa = 25$. For oriented case $e = 0.005$.

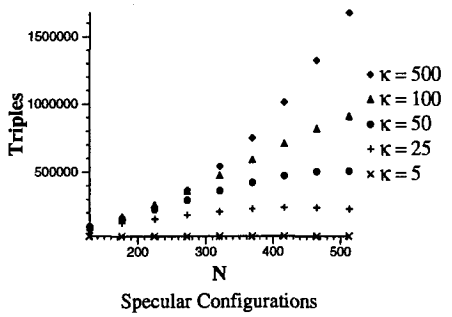



Figure 8: Triples vs. $N$ over $\kappa$. The graph is over parallel polygons for which the error bounds and interpolygon distance have been doubled.

At worst the above form factors are such that $F_{**}S_{\max} < \texttt{Eps}$, where Eps is the most restrictive error bound. Note that, as stated above, $F_{**}$ depends only on the error estimate, reflection function (ie. $S_{\max}$), and error bounds. Only some constant number of such form factors may be fitted over the directional hemisphere above $p_j$, and thus $ij$ may only be linked to some constant number of interactions $jk$. The total number of linked interactions, and corresponding transport triples, is thus $O(n)$.

Note that the above argument, although it establishes the desired bound, may overstate the potential for links at a given interaction. For a given $ij$, much of the directional reflection into the hemisphere over $p_j$ may not achieve $S_{\max}$, and may even be of maximum 0. That is, the analysis ignores the modulation between the paired error and value terms within the error estimate.

As $\kappa$ increases in magnitude, the corresponding bound $S_{\max}$ must increase as well. We may thus expect greater growth in transport computation with higher specular exponent, as shown in Figure 8. Within this graph, growth is superlinear for $\kappa = 500$, though further trials over a higher range of $n = 500 \ldots 2000$ have shown that the rate subsides to linear as $n$ increases, allowing sufficient refinement for the local neighborhood property to obtain.

Finally, we note that under specular reflection each element is reflected across every other element perfectly, and to a first approximation is visible from a constant number of other elements in the environment (at least in the case of a convex enclosed room; the analysis is complicated by occlusion and certain worst case alignments). Thus, the number of interactions is at least $O(n^2)$ — we conjecture that it is no worse than this bound.

## 5.2 Illumination and Refinement

Figure 9 shows illumination and meshing over surfaces of varying glossiness (specular exponent). Within each image, the reflecting surface is perpendicular to the diamond shaped light source, and we see the resulting reflection in the direction of the eye. Note the conformation of meshing to the highlight over each surface. The "stretched" nature of the highlight along the axis to the eye is characteristic of Torrance-Sparrow reflection over fairly oblique angles, and accounts for the increased sensitivity of meshing along this axis. The rightmost three images in the figure show the meshing from above. The illumination shown in these images is somewhat unusual - it shows the reflection to the eye as though it had been painted on the reflecting surface, and then viewed from a different location, directly above. The images in Figure 11 show similar eye/offset views for the reflection of a garish checkerboard.

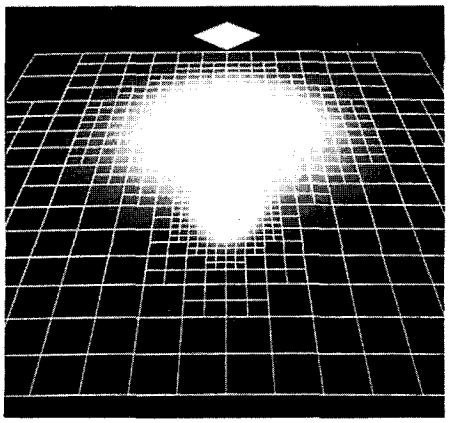


Figure 10: Meshing for glossy and diffuse reflection

| Figure 9 | | | | |
|---|---|---|---|---|
| Max Elements | | 4160 | | |
| Max Triples | | 262144 | | |
| Computed | patches | elements | triples | time |
| $\kappa = 25$ | 790 | 593 | 6706 (2.6%) | 2.2s |
| $\kappa = 100$ | 1290 | 968 | 24214 (9.2%) | 8.0s |
| $\kappa = 500$ | 874 | 656 | 12106 (4.6%) | 4.1s |
| **Figure 10** | | | | |
| Max Elements | | 16448 | | |
| Max Triples | | 1048576 | | |
| Computed | patches | elements | triples | time |
| $\kappa = 500$ | 1578 | 1184 | 7834 (0.75%) | 5.0s |
| **Figure 12** | | | | |
| Max Elements | | 15138 | | |
| Max Triples | | 222385209344 | | |
| Computed | patches | elements | triples | time |
| $\kappa = 500$ | 6479 | 4866 | 70995 (0.00003%) | 3m13s |

Table 1: Image Statistics

The image in Figure 10 shows contrasting illumination and meshing induced by diffuse and glossy reflection. Note the distinct meshing for each highlight. Glossy reflection is at a less oblique angle, and thus both the highlight and meshing exhibit less distortion in the direction of the eye.

Note that these scenes are extremely simple — application to more complex environments is still very expensive, despite the employment of hierarchical methods. Motivated by the work of Smits et al. [14] in hierarchical radiosity, we are currently experimenting with importance and radiance weighting over three point transport — preliminary results of this work are shown in Figure 12. The given environment contains four reflectors: the broad face of each of the three "slabs" and the top of the central cube. In addition to the reflections seen in the slabs, note the play of light originating at the lamp at left, reflected off the cube top, and over the upper part of the green wall at right. Total potential transport triples over this environment at the finest level of subdivision is just over 222 billion — our system, under importance and radiance weighting, employs 70,995, a reduction to 3 hundred-thousandths of 1 percent.

Table 1 provides further statistics for the images. Timings are given for a Silicon Graphics indigo workstation with a single 50 MHz R4000 processor. The image shown in Figure 12 was generated after seven complete iterations (gathers to all interactions), and total time just over three minutes.

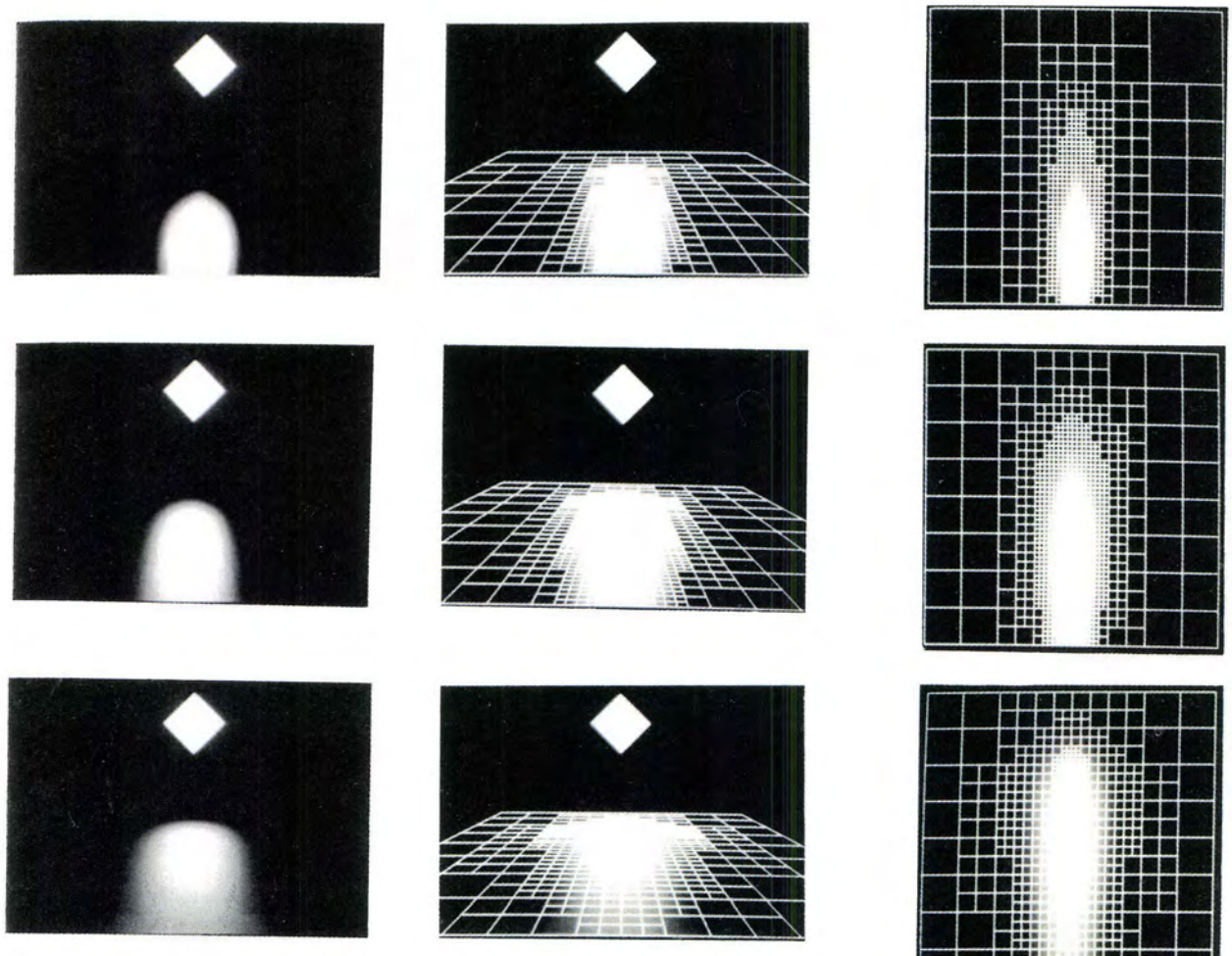


Figure 9: Illumination and Refinement

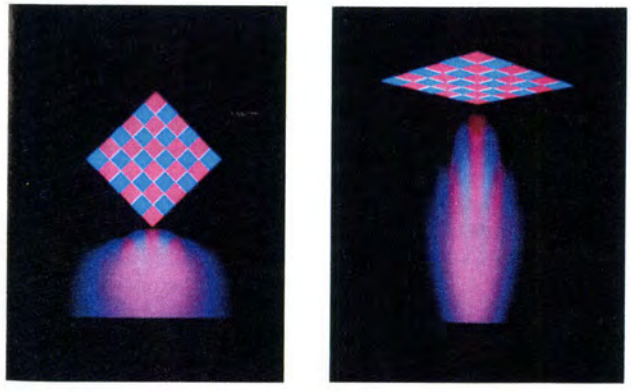


Figure 11: Eye and Offset Views

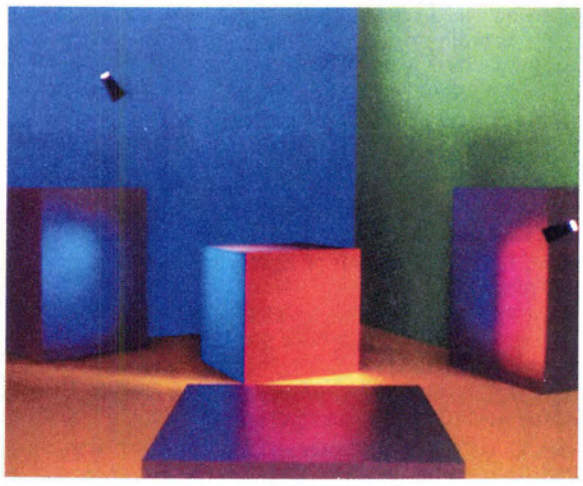


Figure 12: Cube and Slabs

# 6 Discussion

Recall the matrix formulation shown in Figure 2. For any $n$ of reasonable size, the resulting $n^3$ matrix will be unmanageable — we have shown, however, that for sufficient refinement the $n^3$ entries in the matrix may be approximated to within user specified bounds by $O(n)$ subblocks. The gather and push/pull procedures described in preceding sections allow manipulation and solution over this representation. As discussed in [2], the resulting system may be shown to converge.

Growth in transport is more accurately described as $O(n + k^3)$, where $k$ is the number of input polygonal patches within the environment, as opposed to elements. The $k^3$ term is generated by the initial examination of all polygon triples for reflection, and is subsumed by $n$ as the number of elements increases. As the number of polygons in an environment grows, however, the $k^3$ term will become prohibitively large. As discussed in [14] with respect to the related problem under hierarchical radiosity, the capability to cluster as well as refine polygons would reduce the difficulty of unnecessary initial interactions. Clustering is arguably the most important open problem in the computation of global illumination.

The hierarchical approach described in this paper was derived by writing the rendering equation in a three point transport formulation. Another option would be to parameterize radiance by position and direction – we believe that a similar hierarchical approach could be employed with the method of discrete ordinates or spherical harmonics.

Finally, we note that, similarly to other algorithms for hierarchical illumination [7, 14], the algorithm described in this paper bounds estimated error over individual transport computations. As discussed in [14], bounding estimated error over individual transport does not easily or necessarily provide a rigorous bound for overall error in the solution. An analysis and means of computing such a bound over hierarchical illumination remains an interesting open problem.

# 7 Acknowledgements

This research was partially supported by equipment grants from Apple and Silicon Graphics Computer Systems and a research grant from the National Science Foundation (CCR 9207966). The authors would like to thank Dr. P. Prusinkiewicz for access to the graphics research facilities at the University of Calgary during the final stages of this work, and Deborah Fowler for her crucial assistance in shooting test images, paste up and much other support and encouragement. Thanks to Cullen Jennings and David Laur for all of their help recording images. We especially thank the anonymous referees for their many helpful comments and suggestions.

# 8 References

[1] Amanatides, J. (1992) Algorithms for the detection and elimination of specular aliasing. *Proc. Graphics Interface '92*, 86-93.

[2] Aupperle, L. (1993) Hierarchical algorithms for illumination. Doctoral Dissertation, Princeton University.

[3] Blinn, J.F. (1977) Models of light refection for computer synthesized pictures. *Computer Graphics* 11 (2), 192-198.

[4] Chen, S.E., Rushmeier, H.E., Miller, G., Turner, D. (1991) A progressive multi-pass method for global illumination. *Computer Graphics* 25 (4), 165-174.

[5] Cook, R.L. (1986) Stochastic sampling in computer graphics. *ACM Transactions on Graphics* 5 (1), 51-72.

[6] Hall, R. (1989) Illumination and color in computer generated imagery. Springer-Verlag, New York.

[7] Hanrahan, P., Salzman, D., Aupperle, L. (1991) A rapid hierarchical radiosity algorithm. *Computer Graphics* 25 (4), 197-206.

[8] Immel, D.S., Cohen, M.F., Greenberg, D.P. (1986) A radiosity method for non-diffuse environments. *Computer Graphics* 20 (4), 133-142.

[9] Kajiya, J.T. (1986) The rendering equation. *Computer Graphics* 20 (4), 143-150.

[10] Mitchell, D. (1992) Manuscript.

[11] Nicodemus, F.E., Richmond, J.C., Hsia, J.J., Ginsberg, I.W., Limperis, T. (1977) Geometrical considerations and nomenclature for reflectance. National Bureau of Standards monograph, no. 160.

[12] Shirley, P. (1990) A ray tracing method for illumination calculation in diffuse-specular scenes. *Proc. Graphics Interace '90*, 205-212.

[13] Sillion, F.X., Arvo, J.R., Westin, S.H., Greenberg, D.P. (1991) A global illumination solution for general reflectance distributions. *Computer Graphics* 25 (4), 187-196.

[14] Smits, B.E., Arvo, J.R., Salesin, D.H. (1992) An importance-driven radiosity algorithm. *Computer Graphics* 26 (2), 273-282.

[15] Torrance, K.E., Sparrow, E.M. (1967) Theory for off-specular reflection from roughened surfaces. *J. of the Optical Society of America* 57 (9), 1105-1114.

[16] Ward, G.J., Rubinstein, F.M., Clear, R.D. (1988) A ray tracing solution for diffuse environments. *Computer Graphics* 22 (3), 85-92.

## Appendix: Error Analysis

Recall the error expression derived in Section 2.3:

$$\pi(\Delta F_{ji} S_{kji} + \Delta S_{kji} F_{ji})$$

In implementation, we have divided $\Delta S_{kji}$ into separate components for each subfactor of $f_g$. We thus have:

$$\frac{\kappa+2}{8}\left(\Delta F_{ji}\frac{\cos^\kappa\theta_m}{\cos\theta_i\cos\theta_r} + \Delta\cos^\kappa\theta_m F_{ji}\frac{1}{\cos\theta_i\cos\theta_r} + \Delta\sec\theta_i F_{ji}\frac{\cos^\kappa\theta_m}{\cos\theta_r} + \Delta\sec\theta_r F_{ji}\frac{\cos^\kappa\theta_m}{\cos\theta_i}\right)$$

In implementation, the refinement procedure of Section 3.2 takes an additional argument, `ceps`, against which the two estimates of error in reciprocal cosine are tested.

We are left with the computation of $\Delta\sec\theta_i$, $\Delta\sec\theta_r$, and $\Delta\cos^\kappa\theta_m$. The variance (and associated error) in these cosine terms over given patches $A_i$, $A_j$, $A_k$ is determined by the set of possible $\vec{\omega}_i$, $\vec{\omega}_r$ lying between the patches (we dispense with $'$ notation in this section).

Consider patches $A_i$, and $A_j$ (Figure 4): we enclose these patches in spheres $S_i$, $S_j$ with centers $c_i$, $c_j$, and radii $r_i$, $r_j$, respectively. For the moment we will assume that the interiors of $S_i$ and $S_j$ do not intersect, and thus there exists a tangent cone lying between the spheres.

Note that this cone is a right circular cone centered on the line joining $c_i$ and $c_j$. Consider the nappe containing $S_i$: it may be regarded as a cone of direction vectors centered about the vector $c_i - c_j$. We will call this vector cone $C_i$. If $p_i$ and $p_j$ are any two points on or in $S_i$, $S_j$, then the vector $p_i - p_j$ lies within $C_i$. $C_i$ thus bounds the set of possible $\vec{\omega}_i$. We may characterize $C_i$ by the angle $\alpha_i$ defined by its axis, $c_i - c_j$, and boundary — cone $C_r$ and angle $\alpha_r$ may be similarly defined over $A_j$ and $A_k$. If either pair of spheres intersect, we set the corresponding $\alpha = \pi$. We may easily compute maxima and minima for $\sec\theta_i$ and $\sec\theta_r$ given $C_i$ and $C_r$, and may then compute error in estimation as $(\max - \min)/2$.

The cones $C_i$ and $C_r$ centered about $\vec{\omega}_i$ and $\vec{\omega}_r$ induce a similar cone of variation about $\vec{\omega}_m$ (Figure 5). Application of basic spherical trigonometry yields [2]:

$$\alpha_m \leq \arcsin\min\left(\frac{\sin(\alpha_i/2)+\sin(\alpha_r/2)}{\vec{\omega}_i\cdot\vec{\omega}_m}, 1.0\right)$$

Given $\alpha_m$, determination of $\max(\cos^\kappa\theta_m)$, $\min(\cos^\kappa\theta_m)$, and thus $\Delta\cos^\kappa\theta_m$ immediately follows.

Having computed these estimates and maxima, and incorporating the estimates for form factor computation, we may bound and estimate error in transport as:

$$\frac{\kappa+2}{8}\left(F_{dji}^2\max(\cos^\kappa\theta_m)\max(\sec\theta_i)\max(\sec\theta_r) + \Delta\cos^\kappa\theta_m F_{dji}\max(\sec\theta_i)\max(\sec\theta_r) + \Delta\sec\theta_i F_{dji}\max(\cos^\kappa\theta_i)\max(\sec\theta_r) + \Delta\sec\theta_r F_{dji}\max(\cos^\kappa\theta_i)\max(\sec\theta_i)\right)$$

It is this error measure that we employ in our implementation.

# On the Form Factor between Two Polygons

**Peter Schröder** **Pat Hanrahan**

Department of Computer Science
Princeton University

## Abstract

Form factors are used in radiosity to describe the fraction of diffusely reflected light leaving one surface and arriving at another. They are a fundamental geometric property used for computation. Many special configurations admit closed form solutions. However, the important case of the form factor between two polygons in three space has had no known closed form solution. We give such a solution for the case of general (planar, convex or concave, possibly containing holes) polygons.

**CR Categories and Subject Descriptors:** I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism – Radiosity*; J.2 [Physical Sciences and Engineering]: *Engineering*.

**Additional Key Words and Phrases:** Closed form solution; form factor; polygons.

## 1 Introduction

When using the radiosity technique to create images the form factor plays a central role. It describes the fraction of radiation diffusely emitted from one surface reaching another surface. The accurate computation of form factors is the central theme in many recent papers. Goral et al. [4], who introduced radiosity to the computer graphics community, used numerical contour integration to compute form factors between polygons. Cohen and Greenberg [3] took visibility into account with their hemicube algorithm. More recent hierarchical and adaptive algorithms compute still more accurate form factors [10; 5]. Nishita and Nakamae [8] and Baum et al. [2] have used an exact solution for the form factor between a differential surface element and a polygon. Most radiosity algorithms are restricted to polygonal environments, and so a closed form solution for the form factor between polygons is potentially of great utility.

The history of computing form factors is very long. A closed form expression for the form factor between a differential surface element and a polygon was found by Lambert in 1760 [7]. Lambert proceeded to derive the form factor for a number of special configurations among them the form factor between two perpendicular rectangles sharing a common edge. He writes about the latter derivation:

> Although this task appears very simple its solution is considerably more knotted than one would expect. For it would be very easy to write down the differential expression of fourth order, which one would need to integrate four fold; but the highly laborious computation would fill even the most patient with disgust and drive them away from the task.

Other workers have derived closed form solutions for the form factors between many different geometric configurations and these can be found in standard textbooks. However, we are not aware of a closed form solution for the form factor between two general polygons. Thus, this problem has remained open for over 230 years.



In this paper we present a formula for the form factor integral between two general polygons. The derivation of this formula is quite involved, and the interested reader is referred to [9] for a detailed derivation. The purpose of this paper is to bring this result to the attention of the graphics community.

## 2 Closed form solution

The form factor integral can be reduced to a double contour integral by two applications of Stokes' theorem [6]

$$\begin{aligned}\pi A_1 F_{12} &= \int_{A_1}\int_{A_2} \frac{\cos\theta_1 \cos\theta_2}{\|\vec{r}\|^2}\, dA_2\, dA_1 \\ &= \tfrac{1}{4}\int_{\partial A_1}\int_{\partial A_2} \ln(\vec{r}\cdot\vec{r})\, d\vec{x}_2 \cdot d\vec{x}_1\end{aligned}$$

where $\theta_1$, $\theta_2$ are the angles between the normal vector of the respective surface and a radius vector $\vec{r}$, which connects two points on the surfaces. The above equation holds for all surfaces such that every point on either surface sees the same contour of the other surface.

In the case of polygons $P_1$ and $P_2$ the contour integral reduces to a sum of double line integrals over all pairwise combinations of edges

$$4\pi A_{P_1} F_{P_1 P_2} = \sum_{E_i\, E_j} \cos\angle E_i E_j \int_{E_i}\int_{E_j} \ln(\vec{r}\cdot\vec{r})\, ds_j\, dt_i$$

Ignoring the factor $\cos\angle E_i E_j$ we are left with the task of giving a solution to integrals of the general form $\int_0^{c_2}\int_0^{c_0} \ln f(s,t)\, ds\, dt$. $c_0$ and $c_2$ are the lengths of the edges over which a given double contour integral is taken and $f(s,t) = s^2 + c_1 st + t^2 + c_3 s + c_4 t + c_5$ is the bi-quadratic form which arises from the expansion of the dot product (see Table 2 for definitions of all variables). If the two line segments lie in a common plane we can factor $f(s,t)$ into two bi-linear forms and a solution is readily obtained with standard integration tables (see [9]). Lines in general position lead to the following result:

$$\begin{aligned}&\int_0^{c_2}\int_0^{c_0} \ln f(s,t)\, ds\, dt \\ &= \left\{\left[(s+\frac{c_3}{2})\mathrm{G}(f(.,t))(s) + \frac{c_1}{2}\mathrm{H}(f(.,t))(s)\right]\Big|_{s=0}^{s=c_0}\right\}\Big|_{t=0}^{t=c_2} \\ &\quad -2c_0c_2 + c_{14}c_{15}\left\{\left[\pi(2k(s)+1)\mathrm{M}(t)\right.\right. \\ &\quad \left.\left. -i\left(\mathrm{L}(-c_{17}(s))(t) + \mathrm{L}(-c_{18}(s))(t)\right.\right.\right. \\ &\quad \left.\left.\left. -\mathrm{L}(c_{17}(s))(t) - \mathrm{L}(c_{18}(s))(t)\right)\right]\Big|_{s=0}^{s=c_0}\right\}\Big|_{t=\sqrt{\frac{c_{13}}{c_{13}}}}^{t=\sqrt{\frac{c_{13}+c_2}{c_{13}+c_2}}}\end{aligned}$$

where $k(s) \in \{-1, 0, 1\}$ according to the particular branchcut of the complex logarithm choosen in L. The auxiliary functions G, H, L, and M are given in Table 1.

## 3 An example

We have implemented our closed form solution in Mathematica [11] (this code is available from `ps@princeton.edu`). The

$$
\begin{aligned}
\mathrm{L}(b)(y) &:= \int^y t^2(1-t^2)^{-3}\ln(b+t)\,dt &&= \frac{1}{16}\left[\frac{-b\ln(y-1)}{(b+1)^2} - \frac{b\ln(1+y)}{(b-1)^2} + \left(\frac{2(b+y)(1+by)[(b-y)^2+(by-1)^2]}{(b^2-1)^2(y^2-1)^2} + \ln\frac{(1-y)(1-b)}{(1+y)(1+b)}\right)\ln(b+y)\right.\\
&&&\qquad \left. + \frac{2(b-y)}{(b^2-1)(y^2-1)} + \mathrm{Li}_2\left(\frac{1-y}{1+b}\right) - \mathrm{Li}_2\left(\frac{1+y}{1-b}\right)\right]\\
\mathrm{M}(y) &:= \int^y t^2(1-t^2)^{-3}\,dt &&= \frac{1}{16}\left[4y(y^2-1)^{-2} + 2y(y^2-1)^{-1} + \ln\frac{y-1}{y+1}\right]\\
\mathrm{G}(q)(y) &:= \int^y \ln q(t)\,dt &&= \frac{q'(y)}{2a}\ln q(y) - 2y + \frac{d}{a}\tan^{-1}\frac{q'(y)}{d}\\
\mathrm{H}(q)(y) &:= \int^y t\ln q(t)\,dt &&= \left(\frac{y^2}{2} + \frac{c}{2a} - \frac{b^2}{4a^2}\right)\ln q(y) - \frac{y(ay-b)}{2a} - \frac{bd}{2a^2}\tan^{-1}\frac{q'(y)}{d}
\end{aligned}
$$

**Table 1**: Four auxiliary integrals needed in the solution. Notice that $\mathrm{L}(b)(y)$ uses the dilogarithm [1], $\mathrm{Li}_2(z) = \sum_1^\infty \frac{z^k}{k^2}$, $\frac{d}{dz}\mathrm{Li}_2(z) = -\frac{\ln(1-z)}{z}$. In G and H the argument $q$ is an arbitrary quadratic polynomial $q(t) = at^2 + bt + c$ and $d = \sqrt{4ac - b^2}$.

$$
\begin{aligned}
c_0 &= \|E_j\| & c_{13} &= \frac{c_{11} - \sqrt{c_{11}^2 - 4c_{10}c_{12}}}{2c_{10}}\\
c_1 &= -2\vec{d}_i \cdot \vec{d}_j & c_{14} &= \frac{\sqrt{c_{11}^2 - 4c_{10}c_{12}}}{c_{10}}\\
c_2 &= \|E_i\| & c_{15} &= \sqrt{c_{10}}c_{14}\\
c_3 &= -2\vec{d}_j \cdot (\vec{p}_i - \vec{p}_j) & c_{16}(s) &= c_1c_{13} - c_3 - 2s\\
c_4 &= 2\vec{d}_i \cdot (\vec{p}_i - \vec{p}_j) & c_{17}(s) &= \frac{-c_{15} + \sqrt{c_{15}^2 - 4|c_{16}(s)|^2}}{2i\bar{c}_{16}(s)}\\
c_5 &= \|\vec{p}_i - \vec{p}_j\|^2 & c_{18}(s) &= \frac{-c_{15} - \sqrt{c_{15}^2 - 4|c_{16}(s)|^2}}{2i\bar{c}_{16}(s)}\\
c_{10} &= 4 - c_1^2 & & \\
c_{11} &= 4c_4 - 2c_1c_3 & & \\
c_{12} &= 4c_5 - c_3^2 & &
\end{aligned}
$$

**Table 2**: All expressions for two edges $E_{ij}$ with parameterization $\vec{x}_i(t) = \vec{p}_i + t\vec{d}_i$ and $\vec{x}_j(s) = \vec{p}_j + s\vec{d}_j$ ($\|\vec{d}_{i,j}\| = 1$).

implementation requires some care because of the complexities of the functions that are involved.

A simple example, which requires the full power of our formula, concerns the form factor between two equal width rectangles sharing an edge with an enclosing angle $\theta \in [0, \pi]$. The configuration is illustrated in Figure 1 together with the form factor as a function of $\theta$ for different aspect ratios $l = \frac{a}{b}$ (common edge length $b$).

## 4 Conclusion

We have given a closed form solution for the form factor between two general polygons. This solution is non-elementary since it involves the dilogarithm function. The principal value of our solution is in determining exact answers for general polygonal configurations. This can be used in practice for reference solutions to check more efficient approximations. Baum et al. [2] have also shown that the error in the computed solution can be reduced significantly when using a closed form solution near singularities of the integrand.

There has been a long history of computing closed form expressions for form factors starting with Lambert in 1760. The literature lists many special cases for which closed form solutions exist, but hitherto no solution had been given for general polygonal configurations. The present paper closes this gap.

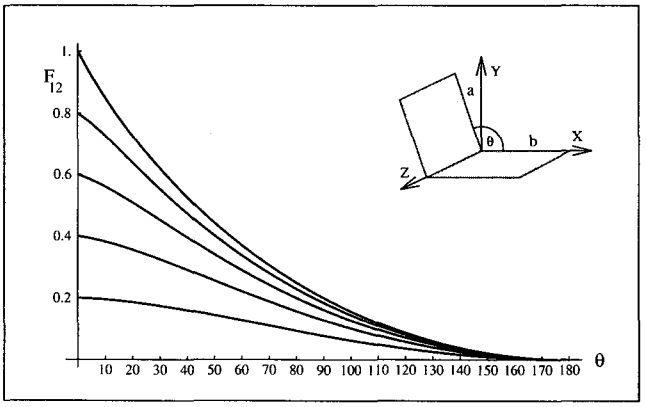



**Figure 1**: Geometry for two rectangles sharing a common edge with an enclosing angle of $\theta$. The graphs show the form factor as a function of $\theta$ for edge ratios $l = \frac{a}{b}$ of .2, .4, .6, .8, and 1.0.

## Acknowledgements

The first author would like to thank the Sci-Vis group at HLRZ for their support. Other support came from Apple, Silicon Graphics and the NSF (contract no. CCR 9207966).

## References

[1] Abramowitz, M., and Stegun, I. A. *Handbook of Mathematical Functions*, 9th ed. Dover Publications, 1970.

[2] Baum, D. R., Rushmeier, H. E., and Winget, J. M. Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors. *Computer Graphics 23*, 3 (July 1989), 325–334.

[3] Cohen, M. F., and Greenberg, D. P. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics 19*, 3 (July 1985), 31–40.

[4] Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. Modelling the Interaction of Light between Diffuse Surfaces. *Computer Graphics 18*, 3 (July 1984), 212–222.

[5] Hanrahan, P., Salzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics 25*, 4 (July 1991), 197–206.

[6] Herman, R. A. *A Treatise on Geometrical Optics*. Cambridge University Press, 1900.

[7] Lambert. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. 1760. German translation by E. Anding in *Ostwald's Klassiker der Exakten Wissenschaften*, Vol. 31-33, Leipzig, 1892.

[8] Nishita, T., and Nakamae, E. Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection. *Computer Graphics 19*, 3 (July 1985), 23–30.

[9] Schröder, P., and Hanrahan, P. A Closed Form Expression for the Form Factor between Two Polygons. Tech. Rep. CS-404-93, Department of Computer Science, Princeton University, January 1993.

[10] Wallace, J. R., Elmquist, K. A., and Haines, E. A. A Ray Tracing Algorithm for Progressive Radiosity. *Computer Graphics 23*, 3 (July 1989), 315–324.

[11] Wolfram, S. *Mathematica*. Addison-Wesley, 1988.

# Reflection from Layered Surfaces due to Subsurface Scattering

Pat Hanrahan

Department of Computer Science
Princeton University

Wolfgang Krueger

Department of Scientific Visualization
German National Research Center
for Computer Science

## Abstract

The reflection of light from most materials consists of two major terms: the specular and the diffuse. Specular reflection may be modeled from first principles by considering a rough surface consisting of perfect reflectors, or micro-facets. Diffuse reflection is generally considered to result from multiple scattering either from a rough surface or from within a layer near the surface. Accounting for diffuse reflection by Lambert's Cosine Law, as is universally done in computer graphics, is not a physical theory based on first principles.

This paper presents a model for subsurface scattering in layered surfaces in terms of one-dimensional linear transport theory. We derive explicit formulas for backscattering and transmission that can be directly incorporated in most rendering systems, and a general Monte Carlo method that is easily added to a ray tracer. This model is particularly appropriate for common layered materials appearing in nature, such as biological tissues (e.g. skin, leaves, etc.) or inorganic materials (e.g. snow, sand, paint, varnished or dusty surfaces). As an application of the model, we simulate the appearance of a face and a cluster of leaves from experimental data describing their layer properties.

**CR Categories and Subject Descriptors:** I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism.*

**Additional Key Words and Phrases:** Reflection models, integral equations, Monte Carlo.

## 1 Motivation

An important goal of image synthesis research is to develop a comprehensive shading model suitable for a wide range of materials. Recent research has concentrated on developing a model of specular reflection from rough surfaces from first principles. In particular, the micro-facet model first proposed by Bouguer in 1759 [4], and developed further by Beckmann[1], Torrance & Sparrow[26], and others, has been applied to computer graphics by Blinn [2] and Cook & Torrance[8]. A still more comprehensive version of the model was recently proposed by He et al[12]. These models have also been extended to handle anisotropic microfacets distributions[24, 5] and multiple scattering from complex microscale geometries[28].

Another important component of surface reflection is, however, diffuse reflection. Diffuse reflection in computer graphics has almost universally been modeled by Lambert's Cosine Law. This law states that the exiting radiance is isotropic, and proportional to the surface irradiance, which for a light ray impinging on the surface from a given direction depends on the cosine of the angle of incidence. Diffuse reflection is qualitatively explained as due to subsurface scattering [18]: Light enters the material, is absorbed and scattered, and eventually exits the material. In the process of this subsurface interaction, light at different wavelengths is differentially absorbed and scattered, and hence is filtered accounting for the color of the material. Moreover, in the limit as the light ray is scattered multiple times, it becomes isotropic, and hence the direction in which it leaves the material is essentially random. This qualitative explanation accounts for both the directional and colormetric properties of diffuse materials. This explanation is also motivated by an early proof that there cannot exist a micro-facet distribution that causes equal reflection in all outgoing directions independent of the incoming direction [10].

The above model of diffuse reflection is qualitative and not very satisfying because it does not refer to any physical parameter of the material. Furthermore, there is no freedom to adjust coefficients to account for subtle variations in reflection from different materials. However, it does contain the essential insight: an important component of reflection can arise from subsurface scattering. In this paper, we present a model of reflection of light due to subsurface scattering in layered materials suitable for computer graphics. The only other work in computer graphics to take this approach is due to Blinn, who in a very early paper presented a model for the reflection and transmission of light through thin clouds of particles in order to model the rings of Saturn[2]. Our model differs from Blinn's in that it is based on one-dimensional linear transport theory—a simplification of the general volume rendering equation [19]— and hence is considerably more general and powerful. Of course, Blinn was certainly aware of the transport theory approach, but chose to present his model in a simpler way based on probabilistic arguments.

In our model the relative contributions of surface and subsurface reflection are very sensitive to the Fresnel effect (which Blinn did not consider). This is particularly important in biological tissues which, because cells contain large quantities of water, are translucent. A further prediction of the theory is that the subsurface reflectance term is not necessarily isotropic, but varies in different directions. This arises because the subsurface scattering by particles is predominantly in the forward direction. In fact, it has long been known experimentally that very few materials are ideal diffuse reflectors (for a nice survey of experiments pertaining to this question, see [18]).

We formulate the model in the currently emerging standard terminology for describing illumination in computer graphics [16, 11]. We also discuss efficient methods for implementation within the context of standard rendering techniques. We also describe how to construct materials with multiple thin layers. Finally, we apply the model to two examples: skin and leaves. For these examples, we build on experimental data collected in the last few years, and provide pointers to the relevant literature.

Another goal of this paper is to point out the large amount of recent work in the applied physics community in the application of linear transport theory to modeling appearance.



 ACM-0-89791-601-8/93/008/0165 $01.50

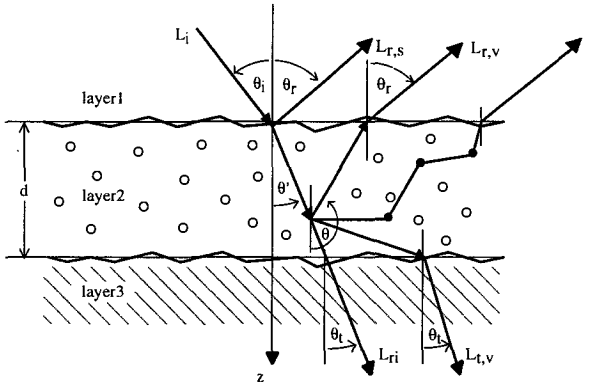



**Figure 1**: The geometry of scattering from a layered surface

| | |
|---|---|
| $(\theta_i, \phi_i)$ | Angles of incidence (incoming) |
| $(\theta_r, \phi_r)$ | Angles of reflection (outgoing) |
| $(\theta_t, \phi_t)$ | Angles of transmission |
| $L(z; \theta, \phi)$ | Radiance [W / (m$^2$ sr)] |
| $L_i$ | Incident (incoming) radiance |
| $L_r$ | Reflected (outgoing) radiance |
| $L_t$ | Transmitted radiance |
| $L_+$ | forward-scattered radiance |
| $L_-$ | backward-scattered radiance |
| $f_r(\theta_i, \phi_i; \theta_r, \phi_r)$ | BRDF |
| $f_t(\theta_i, \phi_i; \theta_t, \phi_t)$ | BTDF |
| $f_{r,s}(\theta_i, \phi_i; \theta_r, \phi_r)$ | Surface or boundary BRDF |
| $f_{t,s}(\theta_i, \phi_i; \theta_t, \phi_t)$ | Surface or boundary BTDF |
| $f_{r,v}(\theta_i, \phi_i; \theta_r, \phi_r)$ | Volume or subsurface BRDF |
| $f_{t,v}(\theta_i, \phi_i; \theta_t, \phi_t)$ | Volume or subsurface BTDF |
| $n$ | Index of refraction |
| $\sigma_s(z; \lambda)$ | Scattering cross section [mm$^{-1}$] |
| $\sigma_a(z; \lambda)$ | Absorption cross section [mm$^{-1}$] |
| $\sigma_t(z; \lambda)$ | Total cross section ($\sigma_t = \sigma_a + \sigma_s$) [mm$^{-1}$] |
| $W$ | Albedo ($W = \frac{\sigma_s}{\sigma_t}$) |
| $d$ | Layer thickness [mm] |
| $p(z; \theta, \phi; \theta', \phi'; \lambda)$ | Scattering phase function (($\theta', \phi'$) to ($\theta, \phi$)) |

**Table 1**: Nomenclature

## 2 Reflection and Transmission due to Layered Surfaces

As a starting point we will assume that the reflected radiance $L_r$ from a surface has two components. One component arises due to surface reflectance, the other component due to subsurface volume scattering. (The notation used in this paper is collected in Table 1 and shown diagramatically in Figure 1.)

$$L_r(\theta_r, \phi_r) = L_{r,s}(\theta_r, \phi_r) + L_{r,v}(\theta_r, \phi_r)$$

where:

$L_{r,s}$ - reflected radiance due to surface scattering

$L_{r,v}$ - reflected radiance due to volume or subsurface scattering

The models developed in this paper also predict the transmission through a layered surface. This is useful both for materials made of multiple layers, as well as the transmission through thin translucent surfaces when they are back illuminated. The transmitted radiance has two components. The first component is called the *reduced intensity*; this is the amount of incident light transmitted through the layer without scattering inside the layer, but accounting for absorption. The second is due to scattering in the volume.

$$L_t(\theta_t, \phi_t) = L_{ri}(\theta_t, \phi_t) + L_{t,v}(\theta_t, \phi_t)$$

where:

$L_{ri}$ - reduced intensity

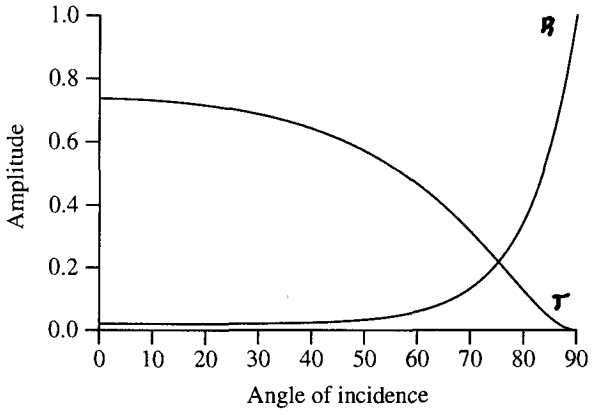



**Figure 2**: Fresnel transmission and reflection coefficients for a ray leaving air ($n$ = 1.0) and entering water ($n$ = 1.33).

$L_{t,v}$ - transmitted radiance due to volume or subsurface scattering

The bidirectional reflection-distribution function (BRDF) is defined to the differential reflected radiance in the outgoing direction per differential incident irradiance in the incoming direction [23].

$$f_r(\theta_i, \phi_i; \theta_r, \theta_r) \equiv \frac{L_r(\theta_r, \phi_r)}{L_i(\theta_i, \phi_i) \cos\theta_i d\omega_i}$$

The bidirectional transmission-distribution function (BTDF) has a similar definition:

$$f_t(\theta_i, \phi_i; \theta_t, \theta_t) \equiv \frac{L_t(\theta_t, \phi_t)}{L_i(\theta_i, \phi_i) \cos\theta_i d\omega_i}$$

Since we have separated the reflected and transmitted light into two components, the BRDF and BTDF also have two components.

$$\begin{aligned} f_r &= f_{r,s} + f_{r,v} \\ f_t &= f_{ri} + f_{t,v} \end{aligned}$$

If we assume a planar surface, then the radiance reflected from and transmitted across the plane is given by the classic Fresnel coefficients.

$$\begin{aligned} L_r(\theta_r, \phi_r) &= R^{12}(n_i, n_t; \theta_i, \phi_i \rightarrow \theta_r, \phi_r) L_i(\theta_i, \phi_i) \\ L_t(\theta_t, \phi_t) &= T^{12}(n_i, n_t; \theta_i, \phi_i \rightarrow \theta_t, \phi_t) L_i(\theta_i, \phi_i) \end{aligned}$$

where

$$\begin{aligned} R^{12}(n_i, n_t; \theta_i, \phi_i \rightarrow \theta_r, \phi_r) &= R(n_i, n_t, \cos\theta_i, \cos\theta_t) \\ T^{12}(n_i, n_t; \theta_i, \phi_i \rightarrow \theta_t, \phi_t) &= \frac{n_t^2}{n_i^2} T = \frac{n_t^2}{n_i^2}(1 - R) \end{aligned}$$

where $R$ and $T$ are the Fresnel reflection formulae and are described in the standard texts (e.g. Ishimura[14]) and $\theta_t$ is the angle of transmission. Besides returning the amount of reflection and transmission across the boundary, the functions $R^{12}$ and $T^{12}$, as a side effect, compute the reflected and refracted angles from the Reflection Law ($\theta_r = \theta_i$) and Snell's Law ($n_i \sin\theta_i = n_t \sin\theta_t$). Note also the factor of $(n_t/n_i)^2$) in the transmitted coefficient of the above formula; this arises due to the change in differential solid angle under refraction and is discussed in Ishimura[pp. 154-155]. Plots of the Fresnel functions for the boundary between air and water are shown in Figure 2.

In our model of reflection, the relative contributions of the surface and subsurface terms are modulated by the Fresnel coefficients.

$$f_r = R f_{r,s} + T f_{r,v} = R f_{r,s} + (1 - R) f_{r,v}$$

Thus, an immediate prediction of the model is that reflection due to subsurface scattering is high when Fresnel reflection is low, since more light enters the surface layer. Notice in Figure 2 that the percentage of transmission is very high for a quite wide range of angles of incidence. Thus, the reflectance properties of materials impregnated with water or oil (dielectrics with low indices of refraction) are dominated by subsurface reflectance components at near perpendicular angles of incidence, and surface components at glancing angles of incidence.

Actually, light returning from the subsurface layers must refract across the boundary again. Thus, it will be attenuated by yet another Fresnel transmission factor. Recall that if light returns from a media with a higher index of refraction, then total internal reflection may occur. All light with an incident angle greater than the critical angle ($\theta_c = sin^{-1}\, n_i/n_t$) will not be transmitted across the boundary. By assuming an isotropic distribution of returning light, we can compute the percentage that will be transmitted and hence considered reflected. This sets an upper bound on the subsurface reflectance of $1 - (n_i/n_t)^2$ (remember, $n_t > n_i$). For example, for an air-water boundary, the maximum subsurface reflectance is approximately .44.

## 3 Description of Materials

The aim of this work is to simulate the appearance of natural materials such as human skin, plant leaves, snow, sand, paint, etc. The surface of these materials is comprised of one or more layers of material composed of a mixture of randomly distributed particles or inhomogeneities embedded in a translucent media. Particle distributions can also exist, in which case the properties are the material are given by the product of each particle's properties times the number of particles per unit volume.

The layers of such materials can be described by a set of macroscopic parameters as shown in the following table. Measurements of these properties have been made for a large variety of natural materials.

| Symbol | Property |
|---|---|
| $n$ | index of refraction |
| $\sigma_a$ [mm$^{-1}$] | absorption cross section |
| $\sigma_s$ [mm$^{-1}$] | scattering cross section |
| $d$ [mm] | depth or thickness |
| $p(\cos j)$ | scattering phase function |
| $g$ | mean cosine of phase function |

- *Index of Refraction*
  The materials considered are dielectrics where $n$ is on the order of the index of refraction of water (1.33).

- *Absorption and scattering cross section*
  The intensity of the backscattered and transmitted light depends on the absorption and scattering properties of the material. The cross section may be interpreted as the probability per unit length of an interaction of a particular type. The total scattering cross section $\sigma_t = \sigma_a + \sigma_s$. The mean free path is equal to the reciprocal of the total cross section. An important quantity is the albedo, which equals $W = \sigma_s/\sigma_t$. If the albedo is close to 1, the scattering cross section is much greater than the absorption cross section, whereas if the albedo is close to 0, absorption is much more likely than scattering.

- *Scattering phase function*
  The phase function, $p(\vec{x}; \theta, \phi; \theta', \phi')$ represents the directional scattering from $(\theta', \phi')$ to $(\theta, \phi)$ of the light incident onto a particle. This function depends on the nature of the scattering medium. The form of $p$ is affected by the size, form and orientation of the suspended particles, the dielectric properties of the particles, and the wavelength of the incident light. The scattering of light from particles small compared to the wavelength of light is given by the Rayleigh scattering formula, and the scattering due to dielectric spheres of different radii by the Mie formula.

  However, most materials contain distributions of particles of many different sizes, so simple single particle phase functions are not applicable. For this reason, we describe the material phase function with the empirical formula, the Henyey-Greenstein formula[13].

$$p_{HG}(\cos j) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos j)^{3/2}}$$

  where $j$ is the angle between the incoming and the outgoing direction (if the phase function depends only on this angle the scattering is symmetric about the incident direction). The Henyey-Greenstein formula depends on a single parameter $g$, the mean cosine of the scattered light. The Henyey-Greenstein phase function for different values of $g$ is shown in Figure 3. Note that if $g = 0$ the scattering is isotropic, whereas positive $g$ indicates predominantly forward scattering and negative $g$ indicates predominantly backward scattering.

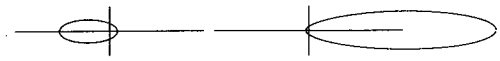


**Figure 3**: Henyey-Greenstein phase function for $g = -.3$ and $g = .6$.

In the model employed in this paper, material properties are described macroscopically as averages over the underlying microscopic material property definitions. If the material is made of several components, the resulting properties of the composite materials can be computed by simple summation.

$$\sigma_a = \sum_{i=1}^{n} w_i\, \sigma_{a,i}$$

$$\sigma_s\, p(\cos j, g) = \sum_{i=1}^{n} w_i\, \sigma_{s,i}\, p(\cos j, g_i)$$

and so on. Here $w_i$ is the volume fraction of the volume occupied by material $i$.

Another very important property of real materials is that the properties randomly vary or fluctuate. Such fluctuations cause variation in the appearance of natural surfaces. This type of fluctuation is easy to model with a random noise function or a texture map.

Optical propagation in random media has been studied in a variety of applications, including blood oximetry, skin photometry, plant physiology, remote sensing for canopies and snow, the paint and paper industry, and oceanic and atmospheric propagation. For many examples the macroscopic parameters have been measured across many frequency bands. A major attempt of our work is the simulation of the appearance of natural surfaces by using measured parameters to be inserted into the subsurface reflection and transmission formulas. This approach is similar to the attempt of Cook & Torrance [8] to simulate the appearance of metallic surfaces by using appropriate values for the refractive index and the roughness parameters.

## 4 Light Transport Equations

Linear transport theory is a heuristic description of the propagation of light in materials. Transport theory is an approximation to elec-

tromagnetic scattering theory, and hence cannot predict diffraction, interference or quantum effects. In particular, the specular reflection of light from rough surfaces whose height variation is comparable in size to the wavelength of incident light requires the full electromagnetic theory as is done in He et al[12]. A nice discussion of the derivation of transport theory from electromagnetism and the conditions under which it is valid is contained in an recent article by Fante[9]. The applicability of transport theory, however, has been verified by its application to a large class of practical problems involving turbid materials, including inorganic materials such as ponds, atmospheres, snow, sand and organic materials such as human skin and plant tissue[14].

Transport theory models the distribution of light in a volume by a linear integro-differential equation.

$$\frac{\partial L(\vec{x},\theta,\phi)}{\partial s} = -\sigma_t L(\vec{x},\theta,\phi) + \sigma_s \int p(\vec{x};\theta,\phi;\theta',\phi') L(\vec{x},\theta',\phi')\, d\theta'\, d\phi'$$

This equation is easily derived by accounting for energy balance within a differential volume element. It simply states that the change in radiance along a particular infinitesimal direction $ds$ consists of two terms. The first term decreases the radiance due to absorption and scattering. The second term accounts for light scattered in the direction of $ds$ from all other directions. Thus, it equals the integral over all incoming directions.

For layered media, the assumption is made that all quantities only depend on $z$ and not on $x$ and $y$. This assumption is valid if the incoming illumination is reasonably constant over the region of interest. It is also roughly equivalent to saying the reflected light emanates from the same point upon which it hits the surface. With this assumption, the above equation simplifies to

$$\cos\theta \frac{\partial L(\theta,\phi)}{\partial z} = -\sigma_t L(\theta,\phi) + \sigma_s \int p(z;\theta,\phi;\theta',\phi')\, L(\theta',\phi')\, d\theta'\, d\phi'$$

The above equation is an integro-differential equation. It can be converted to an equivalent double integral equation, whose solution is the same as the original integro-differential equation.

$$L(z;\theta,\phi) = \int_0^z e^{-\int_0^{z'} \sigma_t \frac{dz''}{\cos\theta}} \int \sigma_s(z')\, p(z';\theta,\phi;\theta',\phi')\, L(z';\theta',\phi')\, d\omega' \frac{dz'}{\cos\theta}$$

This is the basis of most current approaches to volume rendering.

The 1-dimensional linear transport equation must also satisfy certain boundary conditions. This is most easily seen by considering the forward and the backward radiance separately.

$$L(\theta,\phi) = L_+(\theta,\phi) + L_-(\pi-\theta,\phi)$$

Where $L_+$ is energy propagating in the positive $z$ direction, and $L_-$ in the negative direction. Note that $L_-$ is defined to be a function of of $\pi-\theta$, the angle between the backward direction of propagation and the negative $z$ axis. It is important to remember this convention when using formulas involving backward radiances.

At the top boundary the forward radiance is related to the incident radiance.

$$L_+(z=0;\theta',\phi') = \int f_{t,s}(\theta_i,\phi_i;\theta',\phi')\, L_i(\theta_i,\phi_i)\, d\omega_i$$

This simply states that the forward component of radiance entering the volume at the boundary is due to light transmitted across the surface. If we assume a planar surface and parallel incident rays, then $f_{t,s}$ equals the Fresnel transmission term times a $\delta$-function that picks up the appropriate angle of incidence.

$$L_+(z=0;\theta',\phi') = T^{12}(n_i,n_t;\theta_i,\phi_i \rightarrow \theta',\phi') L_i(\theta_i,\phi_i)$$

In the more general case of a rough surface, $f_{t,s}$ is given by a transmission coefficient times the probability that light will refract in the desired direction.

The boundary conditions at the top let us formally state the contribution to reflection due to subsurface scattering in terms of the solution of the integral equation at the boundary $z=0$.

$$L_{r,v}(\theta_r,\phi_r) = \int f_{t,s}(\theta,\phi;\theta_r,\phi_r)\, L_-(z=0;\theta,\phi)\, d\omega$$

Assuming a planar surface, this integral simplifies to

$$L_{r,v}(\theta_r,\phi_r) = T^{21}(n_i,n_t;\theta,\phi \rightarrow \theta_r,\phi_r) L_-(z=0;\theta,\phi)$$

Similar reasoning allows the transmitted radiance to be determined from the boundary conditions at the bottom boundary.

$$L_{t,v}(\theta_t,\phi_t) = \int f_{t,s}(\theta,\phi;\theta_t,\phi_t)\, L_+(z=d;\theta,\phi)\, d\omega$$

Once again, assuming a smooth surface,

$$L_{t,v}(\theta_t,\phi_t) = T^{23}(n_2,n_3;\theta,\phi \rightarrow \theta_t,\phi_t) L_+(z=d;\theta,\phi)$$

Thus, the determination of the reflection functions has been reduced to the computation of $L_-(z=0)$ and $L_+(z=d)$—the solution of the one-dimensional transport equation.

## 5 Solving the Integral Equation

There are very few cases in which integro-differential equations can be directly solved. The most famous solution is for the case of isotropic scattering and was derived by Chandrasekhar[7, p. 124]. Even for this simple phase function the solution is anisotropic.

The classic way to solve such an equation is to write it in terms of the Neumann series. Physically, this can be interpreted as expanding the solution in terms of the radiance due to an integer number of scattering events. That is,

$$L = \sum_{i=0}^{\infty} L^{(i)}$$

where $L^{(0)}$ is the direct radiance assuming no scattering, $L^{(1)}$ is the radiance due to a single scattering event, and $L^{(i)}$ is the radiance due to $i$ scattering events. Similar equations apply to the forward and backward radiances, $L_+^{(i)}$ and $L_-^{(i)}$.

The radiance due to the $i$ scattering events can be written using the following recurrence.

$$L^{(i+1)}(z;\theta,\phi) = \int_0^z e^{-\int_0^{z'} \sigma_t \frac{dz''}{\cos\theta}} \int \sigma_s(z')\, p(z';\theta,\phi;\theta',\phi')\, L^{(i)}(z';\theta',\phi')\, d\omega' \frac{dz'}{\cos\theta}$$

This is the basis for most iterative approaches for numerically calculating transport quantities.

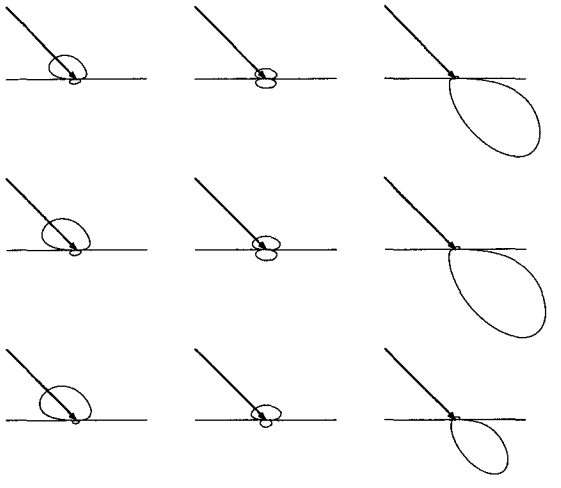


**Figure 4**: Solutions for $f_{r,v}^{(1)}$ and $f_{t,v}^{(1)}$ for different values of $g$ and $\tau_d$. From left to right the phase function shifts from predominately backward scattering ($g = -0.3$) to isotropic scattering ($g = 0.0$) to forward scattering ($g = 0.6$). From top to bottom the optical depth of the layer increases from 0.5 to 1.0 to 2.0.

## 5.1 First-Order Approximation

Another classic result in radiative transport, also derived by Chandrasekhar[7], is the analytic solution to the integral equation assuming only a single scattering event. As mentioned previously, this is equivalent to the method described by Blinn but derived using a completely different technique [2].

The 0th-order solution assumes that light is attenuated by the scattering and absorption, but not scattered. The attenuated incident light is called the *reduced intensity* and equals

$$L_+^{(0)}(z) = L_+(z=0)e^{-\tau/\cos\theta}$$

Here,

$$\tau(z) = \int_0^z \sigma_t \, dz$$

is called the *optical depth*. If $\sigma_t$ is constant, then $\tau_d = \sigma_t d$.

Using the boundary conditions for incident and reflected light, and also rewriting the above equation in terms of the angles of incidence and reflection, we arrive at the following formula for the 0th-order transmitted intensity

$$L_{t,v}^{(0)}(\theta_t, \phi_t) = T^{12}T^{23}e^{-\tau_d}L_i(\theta_i, \phi_i)$$

By substituting the 0th-order solution, or reduced intensity, into the integral equation, the 1st-order solutions for forward and backward scattering can be calculated. The details of this calculation are described in Chandrasekhar and Ishimura and there is no need to repeat them here.

Using the boundary conditions for incident and reflected light, and also rewriting in terms of the angles of incidence and reflection, we arrive at the following formula for the backscattered radiance:

$$L_{r,v}^{(1)}(\theta_r, \phi_r) = WT^{12}T^{21}p(\pi-\theta_r,\phi_r;\theta_i,\phi_i)\frac{\cos\theta_i}{\cos\theta_i+\cos\theta_r}(1-e^{-\tau_d(1/\cos\theta_i+1/\cos\theta_r)})L_i(\theta_i,\phi_i)$$

This general formula shows that the backscattered light intensity depends on the Fresnel transmission coefficients, the albedo, the layer depth, and the backward part of the scattering phase function.

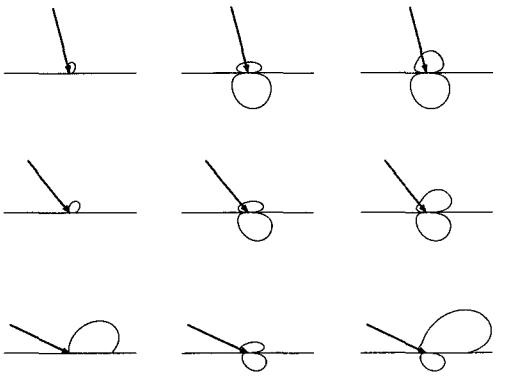


**Figure 5**: Solutions for $f_r$ and $f_t$. In the left column is the surface specular reflection and in the middle is the subsurface reflection and transmission. On the right is the sum of surface and subsurface modulated by the Fresnel coefficients. From top to bottom the angle of incidence increases from 10 to 40 to 65 degrees.

A special case of this equation is Seeliger's Law, the first attempt to model diffuse reflection from first principles[25]. Seeliger's Law can be derived by assuming a semi-infinite layer ($\tau_d = \infty$) and ignoring Fresnel effects.

$$L_{r,v}(\theta_r, \phi_r) = \frac{\cos\theta_i}{\cos\theta_i + \cos\theta_r}L_i(\theta_i, \phi_i)$$

At the boundary $z = d$, the forward scattered radiance is given by

$$L_{t,v}^{(1)}(\theta_t, \phi_t) = WT^{12}T^{23}p(\theta_t,\phi_t;\theta_i,\phi_i)\frac{\cos\theta_i}{\cos\theta_i-\cos\theta_t}(e^{-\tau_d/\cos\theta_i}-e^{-\tau_d/\cos\theta_t})L_i(\theta_i,\phi_i)$$

For $\cos\theta_t = \cos\theta_i$, the singular factors can be avoided by using L'Hospital's rule, yielding

$$L_{t,v}^{(1)}(\theta_t, \phi_t) = WT^{12}T^{23}p(\theta_t, \phi_t; \theta_t, \phi_t)\frac{\tau_d}{\cos\theta_t}e^{-\tau_d/\cos\theta_t}L_i(\theta_t, \phi_t)$$

Figure 4 shows $f_{r,v}$ and $f_{t,v}$ for various values of $g$ and $d$. Figure 5 shows the surface and subsurface components of the reflection model for various angles of incidence. These reflection and transmission distribution functions have several interesting properties:

1. The reflection steadily increases as the layer becomes thicker; in contrast, the transmission due to scattering increases to a point, then begins to decrease because of further scattering events.
2. Subsurface reflection and transmission can be predominately backward or forward depending on the phase function.
3. As the angle of incidence becomes more glancing, the surface scattering tends to dominate, causing both the reflection and the transmission due to subsurface scattering to decrease.
4. Due to the Fresnel effect, the reflection goes to zero at the horizons. Also, the reflection function appears "flattened" relative to a hemicircle. Thus, reflection for near normal angles of incidence varies less than Lambert's Law predicts.
5. The distributions vary as a function of reflection direction. Lambert's Law predicts a constant reflectance in all directions (which would be drawn as a hemicircle in these diagrams).

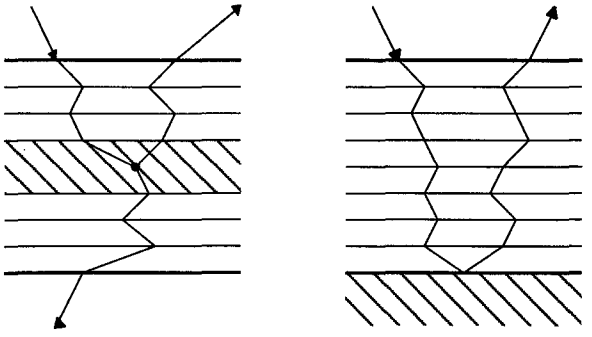


**Figure 6**: Determining first-order solutions for multiple layers. On the left, the contribution to the first order solution for a single layer. One the right, the contribution to the first order solution due to reflectance off a single layer.

The above formulas can be used to generate first-order solutions for multiple layers. (This is shown diagrammatically in Figure 6.) The total first-order scattering will be the sum of the first-order scattering from each layer, weighted by the percentage of light making it to the layer and returning from the layer. The percentage of light making it to the layer is the product of the 0th-order transmission functions (or reduced intensity) for a path through the layers above the reflecting layer. Similarly, the percentage of light leaving the entire layer after reflection is equal to the product of the 0th-order transmission functions for the path taken on the way out. Note that across each boundary the light may refract, and thus change direction and be attenuated by the Fresnel coefficient, but this is easy to handle. The process simplifies, of course, if each layer has the same index of refraction, since no reflection or change of direction occurs between layers. Given the above formulas it is very easy to construct a procedure to perform this calculation and we will make use of it in the results section.

The above formula can also be generalized to include reflection from a boundary between layers. In many situations reflection can only occur from the bottom layer. In this case, we add a single term accounting for the reduced intensity to reach the lower boundary, and also weight the returning light from that boundary. Such a model is commonly employed to model the reflection of light from a pool of water[15], and has been employed by Nishita and Nakamae[22]. Further generalizations of this type are described in Ishimura[14, p. 172].

## 6 Multiple Scattering

The above process of substituting the ith-order solution and then computing the integral to arrive at the (i+1)th-order solution can be repeated, but is very laborious. Note that subsequent integrals now involve angular distributions, because, although the input radiance is non-zero in only a single direction, the scattered radiance essentially comes from the directional properties of the phase function. Thus, this approach to solving the system analytically quickly becomes intractable.

We have implemented a Monte-Carlo algorithm for computing light transport in layered media. This algorithm is described in Figure 7. A thorough discussion of the application of Monte Carlo algorithms for layered media is discussed in the book [21], and the techniques we are using are quite standard.

To investigate the effects of multiple scattering terms, we simulated a semi-infinite turbid media with different albedos. The reflectance was computed and when the particles returning from the media are scored, we keep track of how many scattering events they underwent. Figure 8 shows the results of this experiment. The top curve is the total reflectance, and the lower curves represent scattering up to some order. Note that when the albedo is high, implying that $\sigma_s >> \sigma_a$, the first order term is only a small percentage of the total reflectance. However, as the albedo decreases, corresponding to greater absorption, a few low-order terms accurately approximate the reflectance. This effect can be explained by recalling that each term in the Neumann series representing the reflection is on the order of $W^i$, and since $W$ is always less than one, the magnitude of higher-order terms quickly goes to zero.

We have also computed the BRDF as a function of the angle of reflection using our Monte Carlo algorithm for the same configuration as described in the last experiment. The results are shown in Figure 9. Recall that the 1st-order reflection due to a semi-infinite media is given by Seeliger's Law: $\cos\theta_i/(\cos\theta_i + \cos\theta_r)$. The computed 1st-order BRDF matches the theoretical result quite well. In this figure we also plot the total BRDF due to any number of scattering events, and the difference between the total and the 1st-order BRDF. Note as in the previous experiment when the albedo $W$ is small, the BRDF is closely approximated by the 1st-order term. However, note that the shape of the reflection function is also largely determined by the shape of the 1st-order reflection, which in turn is largely determined by the phase function. Fur-

1 *Initialize*: A particle enters the layer at the origin. Initialize $\vec{p}$ to the origin and the direction $\vec{s}$ to the direction at which the ray enters the layer. Set the weight $w = 1$.

2 *Events*: Repeat the following steps until the ray weight drops below some threshold or the ray exits the layer.

**2A** *Step*: First, estimate the distance to the next interaction:

$$d = -\frac{\log r}{\sigma_t}$$

Where $r$ in this and the following formulas is a uniformly distributed random number between 0 and 1. Then, compute the new position:

$$\vec{p} = \vec{p} + d\,\vec{s}$$

And, finally set the particle weight to

$$w = w\frac{\sigma_s}{\sigma_s + \sigma_a}$$

Note: If $d$ causes the particle to leave the layer, break from the repeat loop and adjust the weight using the distance to the boundary.

**2B** *Scatter*: First, estimate the cosine of the scattering angle for the Henyey-Greenstein phase function using the following formula.

$$\cos j = \frac{1}{|2g|}(1 + g^2 - (\frac{1 - g^2}{1 - g + 2gr})^2)$$

and $\cos\phi$ and $\sin\phi$ with $\phi = 2\pi r$. Then, compute the new direction:

$$\vec{t} = \begin{pmatrix} (\vec{s}.x\cos\phi\cos\theta - \vec{s}.y\sin\phi)/\sin\theta \\ (\vec{s}.y\cos\phi\cos\theta + \vec{s}.x\sin\phi)/\sin\theta \\ \sin\theta \end{pmatrix}$$

$$\vec{s} = \vec{s}\cos j + \vec{t}\sin j$$

Here, $\cos\theta = \vec{s}.z$ and $\sin\theta = \sqrt{1 - \vec{s}.z^2}$. Note: Care must be taken if $\sin\theta = 0$.

3 *Score*: Divide the sphere into regions of equal solid angle and add the weight of the particle to the weight associated with the bin in which it is contained.

**Figure 7**: Basic Monte Carlo algorithm for layered media

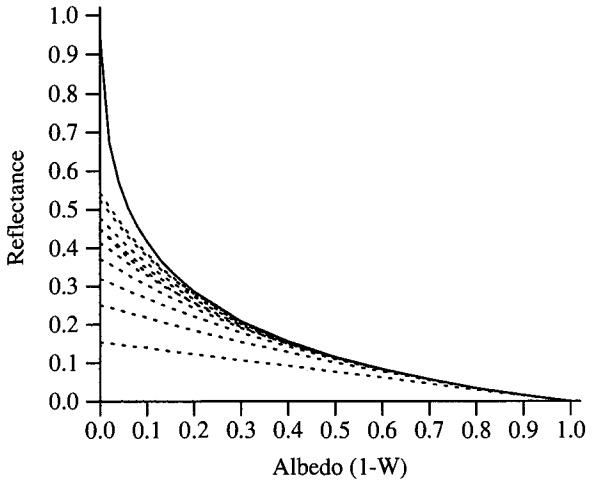



**Figure 8**: A plot of reflectance versus albedo for a semi-infinite media. The top curve is the total reflectance (the total radiant energy per unit area reflected divided by the incident irradiance). The bottom curve is the reflectance assuming only a single scattering event. Moving upward is a sequence of curves consisting of additional terms corresponding to a single additional scattering event. The first 10 terms in the solution are shown; In our simulations, we recorded terms involving thousands of scattering events.

ther, observe that the difference between the 1st-order solution and the full solution is approximately independent of the angle of reflection. Thus, the sum of the higher order terms roughly obeys Lambert's Law. For this reason it is often convenient to divide the subsurface reflection into two terms:

$$L_{r,v}(\theta_r, \phi_r) = L^{(1)}(\theta_r, \phi_r) + L^m$$

where $L^m$ is constant and represents the sum of all the multiple scattering terms.

Finally, we have begun preliminary experiments where we incorporate a Monte Carlo subsurface ray tracer within a standard ray tracer. When the global ray tracer calls the subsurface ray tracer it attempts to estimate the BRDF and BTDF to a particular light source. This is done by *biasing* the Monte Carlo procedure to estimate the energy transported to the light. A simple method to do this is to send a ray to the light at each scattering event, as described in Carter and Cashwell[6]. This ray must be weighted by the phase function and the attenuation caused by the traversal through the media on the way to the light. If the albedo is less than 1, then only a few scattering events are important, and thus the subsurface ray tracer consumes very little time on average (the cost is proportional the the mean number of scattering events). Also, since the subsurface ray tracer does not consider the global environment when tracing its rays, the cost of subsurface Monte Carlo simulation at every shading calculation is relatively low. The advantage of this approach is that the BRDF's do not have to precomputed, and so if material parameters are varying across the surface, the correct answer is still estimated correctly at each point.

# 7 Results

The subsurface scattering models developed in this paper has been tested on two common natural surfaces: human skin and plant leaves. The goal of these experiments are twofold: First, to compare our anisotropic diffuse reflection model with Lambertian shading. Second, to attempt to simulate the optical appearance from measured parameters. Our experiments are meant to be suggestive of the power of this approach; we do not claim to have an experimentally validated model.

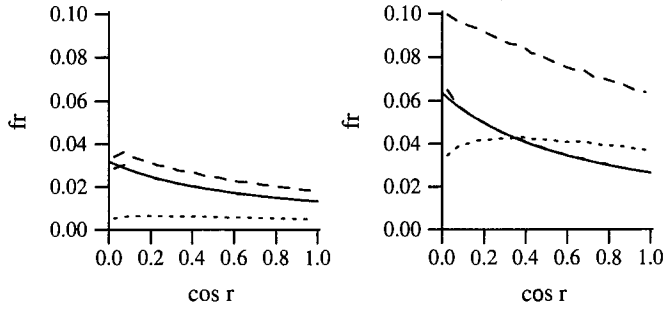



**Figure 9**: Graphs of the BRDF ($f_r$) as a function of the angle of reflection for a semi-infinite slab with different albedos (on the left $W$ = 0.4 and on the right $W$ = 0.8) and an angle of incidence of 45°. The solid line is the theoretical BRDF as given by Seeliger's Law (the superimposed dashed line is the computed 1st-order BRDF showing a good match). The top dashed curve is the total computed BRDF; The bottom dotted curve is the difference between the total BRDF due to multiple scattering events and the 1st-order BRDF.

| Property | Epidermis | Dermis | Pigment | Blood |
|---|---|---|---|---|
| $n$ | 1.37-1.5 | 1.37-1.5 | 1.37-1.5 | 1.37-1.5 |
| $\sigma_a$ [mm$^{-1}$] | 3.8 | 0.3 | | 32.6 |
| $\sigma_s$ [mm$^{-1}$] | 50.0 | 21.7 | | 0.96 |
| $d$ [mm] | 0.001-0.15 | 1-4 | | |
| $g$ | 0.79 | 0.81 | .79 | .0 |

**Table 2**: Two Layer Skin Model Properties. Pigment coefficients are mixed with epidermal coefficients to compute the properties of the outer layer. Blood coefficients are mixed with dermal coefficients to compute the properties of the inner layer.

## 7.1 Skin

Human skin can be modeled as two layers with almost homogeneous properties. Both layers are assumed to have the same refractive index but a different density of randomly distributed absorbers and scatterers. The outer epidermis essentially consists of randomly sized tissue particles and imbedded pigment particles containing melanin. The pigment particles act as strongly wavelength dependent absorbers causing a brown/black coloration as their density increases. The inner dermis is considered to be a composition of weakly absorbing and strongly scattering tissue material and of blood which scatters light isotropically and has strong absorption for the green and blue parts of the spectrum. Experimental evidence also supports the hypothesis that light scattering in the skin is anisotropic with significant forward scattering. A comprehensive study of optical properties of human skin can be found in van Gemert et al.[27]. The values chosen for our test pictures are given in Table 2. We also add a thin outer layer of oil that reflects light using the Torrance-Sparrow model of rough surfaces.

A head data set was acquired using a medical MRI scanner. Unfortunately, the ears and the chin were clipped in the process, but enough of the head is visible to test our shading models. A volume ray tracer was adapted to output the position and normal vector of the skin layer for each pixel into a file, and this input was used to evaluate the shading models described in this paper.

The influence of the various factors appearing in the subsurface reflection formula are shown on Plate 1. These pictures are

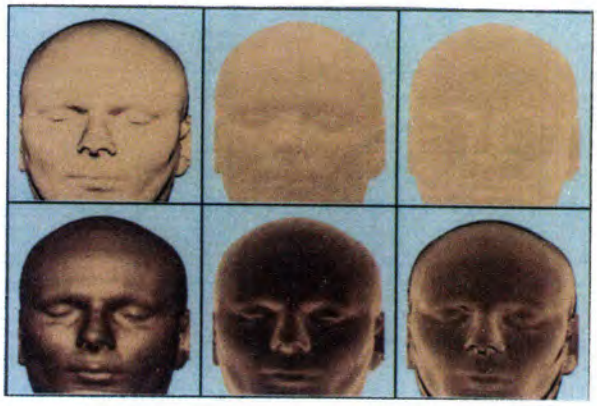


**Plate 1.**

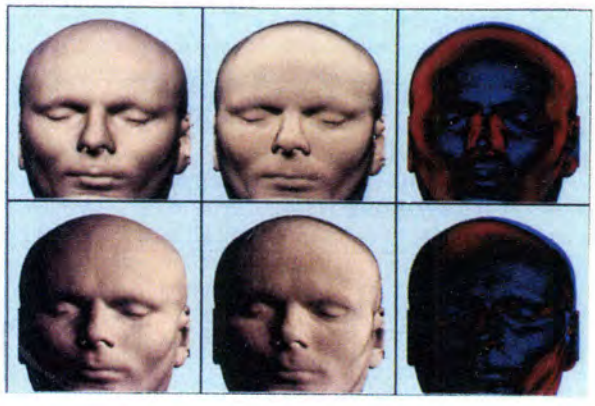


**Plate 2.**

not shaded in the conventional way. In particular, a Lambertian shading model would yield a constant image. The first picture (upper left) shows the influence of the Fresnel factors. Observe that the intensity is almost flat, but strongly attenuated for glancing incident and viewing angles. The second picture (upper middle) shows the action of Seeliger's Law alone. Seeliger's Law leads to very little variation in shading, which makes the surface appear even more chalky or dusty. The third picture (upper right) demonstrates the action of the factor accounting for the finite layer depth giving only weak enhancements for glancing angles. This is a minor effect. The fourth picture (lower left) shows the influence of the Henyey-Greenstein scattering phase function for small backward scattering ($g = -.25$) and the fifth picture (lower middle) shows the effect of large forward scattering ($g = .75$). The result is strong enhancement of glancing reflection for low angles of incidence and viewing, assuming they are properly aligned. The last picture (lower right) shows the superposition of these four factors with $g = .75$ giving a complex behavior. An overall smoothing of the reflection appears; the surface appears to be more "silk-like" (see also Plate 3). Although these effects are all subtle, their combination when controlled properly can create a wide variation in appearance.

The appearance of the face with the new subsurface reflection model is compared to the Lambertian diffuse reflection model for different angles of incidence in Plate 2. The left column shows the results for the Lambert scattering for angles 0 and 45 degrees, and the middle column is rendered for the new model. Again,

**Plate 3:** Dark complexion controlled by setting the concentration of melanin. On the left are images with just subsurface scattering. On the right, an specular surface term is added to simulate an oily coat. In these pictures $g = .65$.

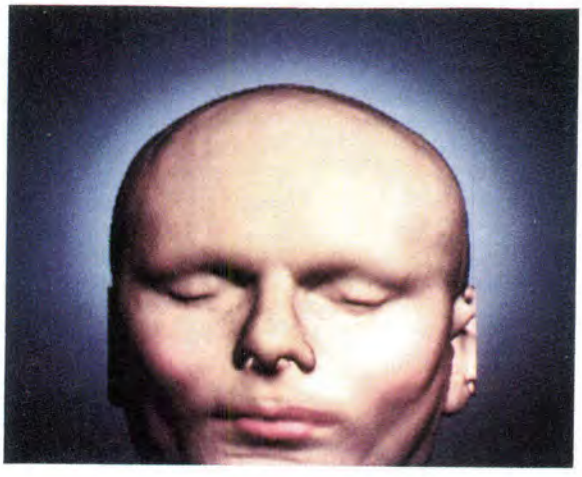


**Plate 4:** Human face with variation in subsurface blood concentration, an oily outer layer and Gaussian variation in parameters to create the "freckles."

notice a much smoother "silk-like" appearance. The right column gives the relative difference of both models, red indicates more reflection from the new model, and blue vice versa.

To illustrate the degrees of freedom of the model, we rendered several faces with their parameters controlled by texture maps. One texture map controls the relative concentration of blood in the dermis; another texture map controls the concentration of melanin in the epidermal layer. These faces are shown in Plates 3 and 4. To create a dark complexion we modulate the percentage of pigment in the otherwise transparent epidermis. This creates a dark brown appearance due to the strong absorption of melanin (in this case we set the absorption to .6). For the lips the epidermis is set to be very thin such that the appearance is dominated by the reflection from the dermis which has for the lips a large blood content (strong absorption for green and blue light component). The epidermis pigment part also has been varied locally with about 20% with a Gaussian process. This allows us to create a wide variety of skin colors, from black to suntanned to Caucasian, and from flushed to burnt to relaxed. The pictures in Plate 3 also show the effect of an additional specular term due to a thin layer of oil on the skin. Finally, Plate 4 shows another picture created by our program.

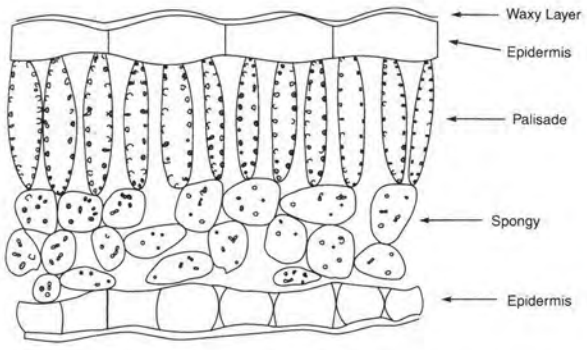



**Figure 10**: Typical leaf cross-section (Redrawn from [20]).

**Plate 5:** Leaf Model. On the left is the albedo image and on the right is a thickness image (white indicates thick)

This picture took approximately 20 seconds to render on a Silicon Graphics Personal Iris.

### 7.2 Leaves

Figure 10 shows an idealized leaf in cross-section. The leaf is composed of several layers of cells. On the top and bottom are epidermal cells with a thin smooth, waxy cuticular outer layer. The waxy cuticular layer is largely responsible for specularly reflected light. Below the upper epidermal cells there are a series of long palisaide cells which are highly absorbing due to the numerous chloroplasts contained within them. Below the palisaide cells are a loosely packed layer of irregularly shaped spongy cells. The spaces between the spongy cells are filled with air, which causes them to scatter light. Both the palisaide and the spongy cells are quite large (approximately 20 $\mu$m) compared to the wavelength, so their scattering phase function is forward directed. Furthermore, the cells are high in water content, so the index of refraction of the leaf is approximately equal to that of water—1.33. A typical leaf is .5 to 1 mm thick, with an optical depth of 5 to 10.

To test our model on a leaf, we constructed a leaf model using the technique described in Bloomenthal[3]. Although spectral transmission and reflectance curves are available for leaves[29], we have set the color of the leaf from an image acquired from a digital scanner. An albedo image is texture mapped onto a series of simply-shaped, bent polygons to create the leaf. Where the texture map is transparent the polygon is considered transparent and the leaf is not visible. We also modulate the thickness of the leaf with a thickness map drawn on top of the original leaf image. The texture maps we used are shown in Plate 5. The waxy cuticle is modeled using a rough specular surface with a specular exponent of 10. The interior of the leaf is modeled as a single homogeneous layer with an optical depth of 5 and a mean scattering cosine of .3[20].

Pictures were generated by modifying a conventional ray tracer

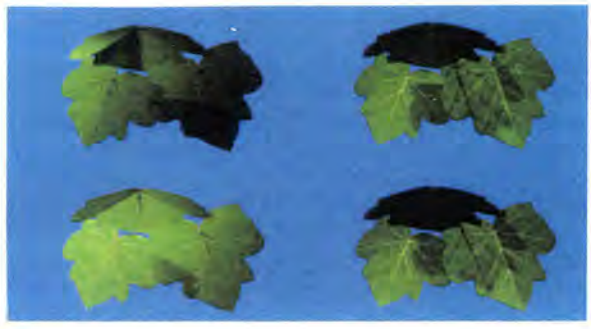


**Plate 6:** A cluster of leaves. A series of leaf images under different simulated lighting conditions. On the left are two backlit images, on the right, front lit.

to account for subsurface reflection and transmission. When a ray encounters a leaf, the BRDF and BTDF are evaluated for direct illumination from light sources. Shadow rays are cast to the light source, and if the ray stabs any other leaves the light intensity is attenuated by the 0th-order transmission function through each leaf. Plate 6 shows a picture of a cluster of leaves with the sun in different positions. Note that the reflection from leaves is largely determined by specular reflection due to the waxy cuticle; there is very little diffuse reflection and hence when the light source is on the same side of the leaf as the viewer, the leaf is quite dark. The transmission term, however, can be quite large, and therefore the leaves may actually be brighter when the are illuminated from behind. Note also that the increased thickness of the veins cause dark shadows to be cast on other leaves. The veins also appear dark when the leaf is back lit because they absorb more light, and bright when the leaf is front lit because their increased thickness causes more light to be reflected.

## 8 Summary and Discussion

We have presented a reflectance model consisting of two terms: the standard surface reflectance and a new subsurface reflectance due to backscattering in a layered turbid media. This model is applicable to biological and inorganic materials with low indices of refraction, because their translucent nature implies that a high percentage of the incident light enters the material, and so the subsurface reflection is quite large. This model incorporates directional scattering within the layer, so the resulting subsurface reflection is not isotropic. This model can be interpreted as a theoretical model of diffuse reflectance. Thus, this model predicts a directionally varying diffuse reflection, in contrast to Lambert's Law. However, if multiple scattering contributes significantly to the reflection, then the higher scattering terms contribute to a reflection function with roughly the same shape.

As in any model, our model makes many assumptions. The two most important are that the physical optics may be approximated with transport theory, and that the material can be abstracted into layered, turbid media with macroscopic scattering and absorption properties. An "exact" model of biological tissues would explicitly model individual cells, organelles and so on, in considerably more detail. The Monte-Carlo algorithm for simulating reflection by Westin et al.[28] is an example of such an approach. Although such an approach may seem more accurate, often the experimental data needed to describe the arrangements of these structures is simply not available, and so in the end the results may be difficult to validate. An advantage of the transport theory approach is that the parameters of the model often may be directly extracted from experimental data.

A legitimate criticism of our work is that we did not directly compare the predictions of our model with experiment. The predictions of our model and the influence of measured material parameters should be checked carefully. However, we believe that this model has many applications in computer graphics even if it does not perfectly predict measured reflection functions. The metaphor of layered surfaces is very easy for users to understand because is a natural way to describe phenomenologically the appearance of many materials. It also fits easily into most rendering systems and can be implemented efficiently.

Finally, transport theory is a heuristic theory based on abstracting microscopic parameters into statistical averages. Transport theory is also the basis of the rendering equation, which is widely viewed as the correct theoretical framework for global illumination calculations. In this paper we propose to model surface reflection from layered surfaces with transport theory. Thus, when our reflectance model for layered surfaces is incorporated into a ray tracer, there is a hierarchy of transport calculations being performed. Within this hierarchy, the lower level transport equation computes the reflectance for the higher level transport equation. When performing this calculation, the lower level transport equation uses as its initial conditions the values from the higher level transport solution. Thus the two levels are coupled in a very simple way. In fact, it is possible to reformulate transport theory entirely in terms of reflection functions, the result is an integral equation for the reflection function itself; in this formulation the radiance does not appear at all. Coupling transport equations at different levels of detail in this manner is a promising approach to tackling the problem of constructing representations with many different levels of detail as proposed by Kajiya[17].

## 9 Acknowledgements

We would like to thank Craig Kolb for his help with RayShade and the leaf pictures. We would also like to thank David Laur for his help with the color plates. This research was partially supported by Apple, Silicon Graphics Computer Systems, David Sarnoff Research Center, and the National Science Foundation (CCR 9207966).

## References

[1] BECKMANN, P., AND SPIZZICHINO, A. *The scattering of electromagnetic waves from rough surfaces.* Pergamon, Oxford, 1963.

[2] BLINN, J. F. Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. *Computer Graphics 16*, 3 (July 1982), 21–29.

[3] BLOOMENTHAL, J. Modeling the Mighty Maple. *Computer Graphics 19*, 3 (July 1985), 305–311.

[4] BOUGUER, P. *The Gradation of Light.* University of Toronto Press, 1960.

[5] CABRAL, B., MAX, N., AND SPRINGMEYER, R. Bidirectional reflection functions from surface bump maps. *Computer Graphics 21*, 4 (July 1990), 273–281.

[6] CARTER, L., AND CASHWELL, E. *Particle Transport Simulation with the Monte Carlo Method.* Energy Research and Development Administration, 1975.

[7] CHANDRASEKHAR, S. *Radiative Transfer.* Dover, New York, 1960.

[8] COOK, R. L., AND TORRANCE, K. E. A Reflection Model for Computer Graphics. *ACM Transactions on Graphics 1*, 1 (1982), 7–24.

[9] FANTE, R. Relationship between Radiative Transport Theory and Maxwell's Equations in Dielectric Media. *J. Opt. Soc. Am. 71*, 4 (April 1981), 460–468.

[10] GRAWBOSKI, L. *Astrophysics J. 39* (1914), 299.

[11] HANRAHAN, P. From Radiometry to the Rendering Equation. *SIGGRAPH Course Notes: An Introduction to Radiosity* (1992).

[12] HE, X. D., TORRANCE, K. E., SILLION, F. X., AND GREENBERG, D. P. A Comprehensive Physical Model for Light Reflection. *Computer Graphics 25*, 4 (July 1991), 175–186.

[13] HENYEY, L. G., AND GREENSTEIN, J. L. Diffuse radiation in the galaxy. *Astrophysics J. 93* (1941), 70–83.

[14] ISHIMURA, A. *Wave Propagation and Scattering in Random Media.* Academic Press, New York, 1978.

[15] JERLOV, N. G. *Optical Oceanography.* Elsevier, Amsterdam, 1968.

[16] KAJIYA, J. Radiometry and Photometry for Computer Graphics. *SIGGRAPH Course Notes: State of the Art in Image Synthesis* (1990).

[17] KAJIYA, J. Anisotropic Reflection Models. *Computer Graphics 19*, 3 (July 1985), 15–22.

[18] KORTUM, G. *Reflectance Spectroscopy.* Springer-Verlag, Berlin, 1969.

[19] KRUEGER, W. The Application of Transport Theory to the Visualization of 3-D Scalar Fields. *Computers in Physics 5* (April 1991), 397–406.

[20] MA, Q., ISHIMURA, A., PHU, P., AND KUGA, Y. Transmission, Reflection and Depolarization of an Optical Wave For a Single Leaf. *IEEE Transactions on Geoscience and Remote Sensing 28*, 5 (September 1990), 865–872.

[21] MARCHUK, G., MIKHAILOV, G., NAZARALIEV, M., DARBINJAN, R., KARGIN, B., AND ELEPOV, B. *The Monte Carlo Methods in Atmospheric Optics.* Springer Verlag, Berlin, 1980.

[22] NAKAMAE, E., KANEDA, K., OKAMOTO, T., AND NISHITA, T. A Lighting Model Aiming at Drive Simulators. *Computer Graphics 24*, 4 (August 1990), 395–404.

[23] NICODEMUS, F. E., RICHMOND, J. C., AND HSIA, J. J. *Geometrical Considerations and Reflectance.* National Bureau of Standards, October 1977.

[24] POULIN, P., AND FOURNIER, A. A Model for Anisotropic Reflection. *Computer Graphics 24*, 4 (August 1990), 273–282.

[25] SEELIGER, R. *Munch. Akad. II. Kl. Sitzungsber 18* (1888), 201.

[26] TORRANCE, K. E., AND SPARROW, E. M. Theory of Off-Specular Reflection From Roughened Surfaces. *Journal of the Optical Society of America 57* (September 1967), 1104–1114.

[27] VAN GEMERT, M. F. C., JACQUES, S. L., STERENBERG, H. J. C. M., AND STAR, W. M. Skin Optics. *IEEE Transactions on Biomedical Engineering 36*, 12 (December, 1989), 1146–1154.

[28] WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. Predicting Reflectance Functions from Complex Surfaces. *Computer Graphics 26*, 2 (July 1992), 255–264.

[29] WOOLLEY, J. T. Reflectance and Transmittance of Light by Leaves. *Plant Physiology 47* (1971), 656–662.

# Display of The Earth Taking into Account Atmospheric Scattering

**Tomoyuki Nishita** **Takao Sirai**
Fukuyama University
Higashimura-cho, Fukuyama, 729-02 Japan

**Katsumi Tadamura** **Eihachiro Nakamae**
Hiroshima Prefectural University
Nanatsuka-cho, Shoubara City, 727 Japan

## Abstract

A method to display the earth as viewed from outer space (or a spaceship) is proposed. The intention of the paper is application to space flight simulators (e.g., reentry to the atmosphere) and the simulation of surveys of the earth (comparisons with observations from weather satellites and weather simulations); it is not for geometric modeling of terrains and/or clouds viewed from the ground, but for displaying the earth including the surface of the sea viewed from outer space taking into account particles (air molecules and aerosols) in the atmosphere and water molecules in the sea.

The major points of the algorithm proposed here are the efficient calculation of optical length and sky light, with lookup tables taking advantage of the facts that the earth is spherical, and that sunlight is parallel.

**CR Categories and Subject Descriptors:**
I.3.3 [**Computer Graphics**]:Picture/Image Generation
I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism
**Key Words:** Earth, Atmospheric Scattering, Optical Length, Sky light, Color of Water, Photo-realism, Radiative Transfer

## 1 INTRODUCTION

Research on image synthesis of realistic 3-D models is one of the most popular fields these days. Displays of natural scenes such as mountains, trees, sea, clouds have been attractively rendered, and an image synthesis of the earth has also been developed. Images of the earth are widely used in movies or TV commercials, e.g., the CG library of earth images[6] was recently released for use in this field. These images, however, are focused on how to create attractive images without any requirement of physical based accuracy. However, physically-based images are required for the study of the simulation of surveys of the earth, such as observation from weather satellites in comparison to weather simulation, and flight simulators in space. The color of the earth when viewed from space varies according to the relationship between the view direction and the position of the sun. In the famous words of the astronaut, "the earth was blue". When we observe the earth from relatively close to the atmosphere, the atmosphere surrounding the earth appears as blue, and the atmosphere near the boundary of the shadow due to the sun appears red (i.e., sunset). The color of clouds also varies according to the sun's position. These phenomena are optical effects caused by particles in the atmosphere, and cannot be ignored. The color of the surface of the sea is not uniform, such as navy blue; it has various colors which depend on incident light to the sea and absorption/scattering effects due to water molecules.

This paper proposes an algorithm of physically-based image synthesis of the earth viewed from space. The method proposed here has the following advantages:

(1) Calculation of the spectrum of the earth viewed through the atmosphere; the earth is illuminated by direct sunlight and sky light affected by atmospheric scattering.

(2) Calculation of the spectrum of the atmosphere taking account of absorption/scattering due to particles in the atmosphere.

(3) Calculation of the spectrum on the surface of the sea taking into account radiative transfer of water molecules.

The major parts in 1) and 2) are concerned with the calculation of optical length and sky light. For these calculations, numerical integrations taking into account atmospheric scattering are required, but they are effectively solved by using several (various) lookup tables making good use of the facts that the shape of the earth is a sphere and that sunlight is a parallel light. For 3), we show that an analytical solution is available instead of numerical integrations.

In the following sections, the basic idea of the lighting model for rendering the color of the earth taking into account atmospheric scattering, rendering the color of clouds, and spectrum calculation of the sea is described. Finally, several examples are demonstrated in order to show the effectiveness of the method proposed here.

## 2 BASIC IDEAS

In order to render the earth, the following elements should be taken into account: a geometric model of the earth, the atmosphere (air molecules, aerosols), sea, clouds, and the spectrum of the sunlight.

This paper discusses rendering an algorithm of the earth, the atmosphere, sea, and clouds viewed from outer space or various positions within the atmosphere; the following optical characteristics should be considered:

(1) The color of the atmosphere: the atmosphere contains air molecules and aerosols, and scattered sunlight from those particles reaches the viewpoint; the intensity of the light reaching the viewpoint is obtained by integrating scattered light from every particle on the ray, and the light scattered from the atmosphere around the earth also reaches this viewpoint.

(2) The color of the earth's surface: the earth is illuminated by both direct sunlight and sky light. Sunlight is absorbed when light passes through the atmosphere, and sky light consists of light scattered by particles in the air. On the way, passing through the atmosphere the light is attenuated, and its spectrum changes.

(3) The color of the sea: sunlight reaching the surface of the sea is divided into reflected light at the surface and light scattered from water molecules. Both of them pass through the atmosphere and reach the viewpoint.

(4) The color of clouds: sunlight is scattered from particles of clouds, the scattered light is attenuated and reaches the viewpoint.

These phenomena should be simulated as precisely as possible in the calculation of the spectrum of the earth and the atmosphere. As we intend to concentrate on close views of the earth, the bumped terrain model of the earth is used instead of a simple sphere; the continents are modeled by 3D fractals, and the sea is expressed by a sphere consisting of some curved surfaces. Geometric models such as a spaceship are also dealt with.

For hidden surface removal, the scanline algorithm for free form surfaces developed by the authors is employed[11]; the surfaces are expressed by Bézier surfaces.

# 3 MODELING OF THE EARTH

Even though we may use a modeling in which the earth is treated as a sphere and the land is modeled by bump mapping, we consider the earth as having two components, land and sea: the sea consist of eight cubic Bézier patches, and the land consists of a set of curved surfaces.

The land data is made by mapping small patches onto the sphere, which are subdivided by using fractals after giving the altitude data for each mesh point overlapped onto a world map: the random midpoint displacement algorithm is employed as a fractal.

A scanned image of the map is used as the texture of the land. Therefore the color is not the real color of the earth.

# 4 SPECTRUM OF THE ATMOSPHERE

Previous work taking account scattering/absorption due to particles include; a) the display of Saturn's rings (reflective ice particles)[1], b) for light scattering from particles in the air, shafts of light caused by spot lights[12], and light beams passing through gaps in the clouds or through trees[8], c) scattered light due to nonuniform density particles such as clouds and smoke[12][4], d) sky color taking account atmospheric scattering[5]. In this paper we focus our discussion on the atmosphere. On this topic, Klassen[5] approximated the atmosphere as multiple layers of plane-parallel atmosphere with uniform density; however, this method results in a large error near the horizon. We discuss here a spherical-shell atmosphere with continuous variation of density in order to improve accuracy. Though his method can only render the color of the sky viewed from a point on the earth, the method discussed here can render the color of the atmosphere viewed from space.

The color of the atmosphere is much influenced by the spectrum of the sunlight, scattering/absorption effects due to particles in the air, reflected light from the earth's surface, and the relationship between the sun's position and the viewpoint (and direction). The sunlight entering the atmosphere is scattered/absorbed by air molecules and aerosol, and ozone layers. The characteristics of scattering depend on the size of particles in the atmosphere. Scattering by small particles such as air molecules is called Rayleigh scattering, and scattering by aerosols such as dust is called Mie scattering. Light is attenuated by both scattering and absorption.

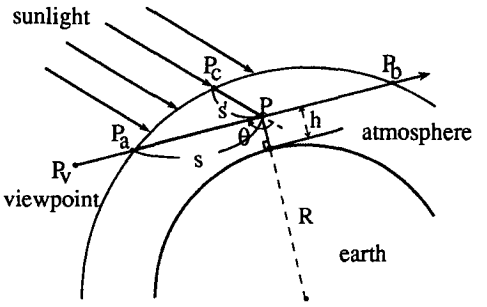



Figure 1: Intensity calculation for the ray intersecting only with the atmosphere.

## 4.1 Assumptions for Spectrum Calculation

For the spectrum calculation, we use the following assumptions:

(1) The multiple scattering of light between air molecules and aerosols in the atmosphere is ignored because of its negligible values and large computational cost, so only single scattering is considered. The interreflection of light between the earth's surface and particles in the air is also neglected because of the same reasons.

(2) For visible wavelengths, absorption in the ozone layer is negligible compared to absorption by air molecules and aerosols.

(3) The density distributions of air molecules and aerosols are taken into account; their densities vary exponentially with altitude[16].

(4) It is assumed that light travels in a straight line even though the actual path is curved due to the variation of index of refraction with altitudes.

## 4.2 Atmospheric Scattering

Let's consider scattering due to air molecules and aerosols.

First, single scattering due to air molecules is described. The light reflected due to Rayleigh scattering, $I$, is generally given by the following equation;

$$I(\lambda,\theta) = I_0(\lambda)K\rho F_r(\theta)/\lambda^4$$
$$K = \frac{2\pi^2(n^2-1)^2}{3N_s} \quad (1)$$

where $I_o$ is the intensity of incident light, $K$ is a constant for the standard atmosphere (molecular density at sea level), $\theta$ the scattering angle (see Fig. 1), $F_r$ the scattering phase function indicating the directional characteristic of scattering ( given by $3/4(1+cos^2(\theta))$), $\lambda$ the wavelength of incident light, $n$ the index of refraction of the air, $N_s$ the molecular number density of the standard atmosphere, and $\rho$ the density ratio. $\rho$ depends on the altitude $h$ ($\rho = 1$ at sea level) and is given by

$$\rho = exp(\frac{-h}{H_0}), \quad (2)$$

where $H_0$ is a scale height ($H_o = 7994$m), which corresponds to the thickness of the atmosphere if the density were uniform.

Eq. (1) indicates that the intensity of scattering is inversely proportional to the 4th power of the wavelength. Short wavelength light is very strongly attenuated by traversing the atmosphere, but long wavelength light is scarcely affected. This is why the sky appears blue in the daytime. Conversely, at sunset or sunrise, the distance traversed by the light increases, and the color of sky changes

to red because of increased scattering of short wavelengths. The attenuation coefficient $\beta$ (i.e., the extinction ratio per unit length) is given by

$$\beta = \frac{8\pi^3(n^2-1)^2}{3N_s\lambda^4} = \frac{4\pi K}{\lambda^4} \tag{3}$$

As shown in Fig.1, the light reaching viewpoint $P_v$ can be obtained as the remainder after scattering and absorption due to air molecules along the path between $P_b$ and $P_v$. The light at $P$ has been attenuated due to travel in the atmosphere ($P_cP$), and the light scattering from $P$ is also attenuated before reaching $P_v$.

To calculate the attenuation caused by particles for light of wavelength $\lambda$ traversing distance $s$, we use the optical depth, which is obtained by integrating $\beta$ of Eq. (3) along the path $s$. Let's denote the integration variable $s$ and the distance $S$, then the optical depth is given by

$$t(S,\lambda) = \int_0^S \beta(s)\rho(s)ds = \frac{4\pi K}{\lambda^4}\int_0^S \rho(s)ds \tag{4}$$

Next, single scattering due to aerosols is described. Scattering optics and the density distribution for aerosols differ from air molecules; Eq. (4) is different, too. Because the size range of particles of aerosols is very great, Mie scattering is applied for the phase function in Eq. (1) which exhibits a strong forward directivity. The Henyey-Greenstein function is well known as a phase function. Recently, Cornette[18] improved it, which gives a more reasonable physical expression:

$$F(\theta,g) = \frac{3(1-g^2)}{2(2+g^2)}\frac{(1+cos^2\theta)}{(1+g^2-2gcos\theta)^{3/2}}, \tag{5}$$

where $g$ is an asymmetry factor and given by

$$g = \frac{5}{9}u - (\frac{4}{3} - \frac{25}{81}u^2)x^{-1/3} + x^{1/3},$$
$$x = \frac{5}{9}u + \frac{125}{729}u^3 + (\frac{64}{27} - \frac{325}{243}u^2 + \frac{1250}{2187}u^4)^{1/2},$$

where if $g = 0$ then this function is equivalent to Rayleigh scattering. $u$ is determined by the atmospheric condition (e.g., haze) and wavelength; $u$ varies from 0.7 to 0.85(see [18]).

Like the density distribution of air molecules, the density of aerosols decreases exponentially with altitude, but the rate of decrease is different from that of air molecules. The density can be obtained by setting the scale height, $H_o$, of Eq. (2) to 1.2km[16].

## 4.3 Intensity Calculation due to Atmospheric Scattering

Let's discuss a ray from viewpoint $P_v$ to the earth, the light reaching the viewpoint has the following three passes: a) the ray passing through only the atmosphere, b) the ray intersecting with the earth, c) the ray passing through only space. For c) intensity calculation is not required. The calculation methods for a) and b) are described in the following.

### 4.3.1 Spectrum calculation for only the atmosphere

Let's discuss light scattering due to air molecules on the ray passing just through the atmosphere. The discussion for aerosols is omitted because the optics is similar except for $1/\lambda^4$ dependence. As shown in Fig.1, the light reaching $P_v$ can be obtained as the remainder after scattering and absorption due to air molecules along the intersection line

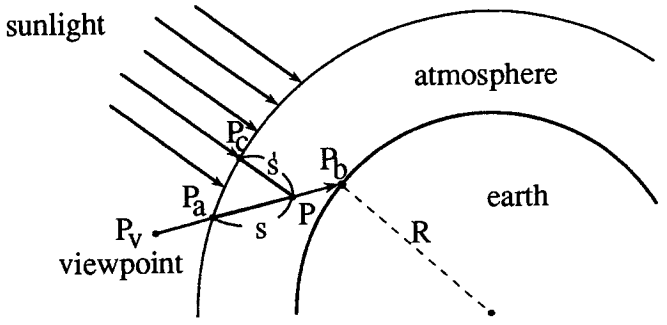



Figure 2: Intensity calculation for the ray intersecting with the earth.

between the ray and the atmosphere, $P_bP_a$. The intensity of the light scattered at point P (at distance s from $P_v$) in the direction of $P_v$, $I_p$, is obtained by Eq.(1). The light scattered at $P$ is attenuated before arriving at $P_v$. The intensity of the light arriving at $P$, $I_p$, can be obtained by setting the integration interval to $P_cP$ in Eq. (4) of optical depth, that is

$$I_p(\lambda) = I_s(\lambda)KF_r(\theta)\rho\frac{1}{\lambda^4}exp(-t(PP_c,\lambda)), \tag{6}$$

where $I_s$ is the solar radiation at the top of the atmosphere, and $t(PP_c,\lambda)$ the optical depth from the top of the atmosphere to point P ($l$ is the integration variable) and given by

$$t(PP_c,\lambda) = \int_P^{P_c}\beta(l)\rho(l)dl.$$

As the light scattering from $P$ is also attenuated before reaching $P_v$, the intensity of the light reaching $P_v$, $I_{pv}$, can be obtained by multiplying the attenuation by the intensity at $P$, that is

$$I_{pv}(\lambda) = I_p(\lambda)exp(-t(PP_a,\lambda)). \tag{7}$$

As the distance to the sun can be considered almost infinite, the sunlight can be assumed to be a parallel beam. Thus the scattering angle at every point along $P_aP_b$ can be considered constant. That is, $I_v$ reaching $P_v$ can be obtained by integrating scattered light due to air molecules on $P_aP_b$:

$$I_v(\lambda) = \int_{P_a}^{P_b} I_{pv}(\lambda)ds$$
$$= I_s(\lambda)\frac{KF_r(\theta)}{\lambda^4}\int_{P_a}^{P_b}\rho\, exp(-t(PP_c,\lambda) - t(PP_a,\lambda))ds \tag{8}$$

### 4.3.2 Spectrum calculation of the earth

Let's consider the ray intersecting with the earth as shown in Fig.2. The intensity scattered due to particles on the path, $P_aP_b$, can be obtained in the same manner as the description in 4.3.1. When point $P$ coincides with point $P_b$ (i.e., on the earth surface), the light reaching the viewpoint is obtained by adding reflected light from the earth to the light scattered due to molecules on $P_aP_b$. The intensity of light reaching viewpoint $P_v$, $I'_v$, is expressed by

$$I'_v(\lambda) = I_v(\lambda) + I_e(\lambda)exp(-t(P_aP_b,\lambda)), \tag{9}$$

where $I_v$ is the scattered light of Eq. (8). $I_e$ is reflected light at the earth;the direct component of sunlight and ambient light. The ambient light is mainly sky light. By considering

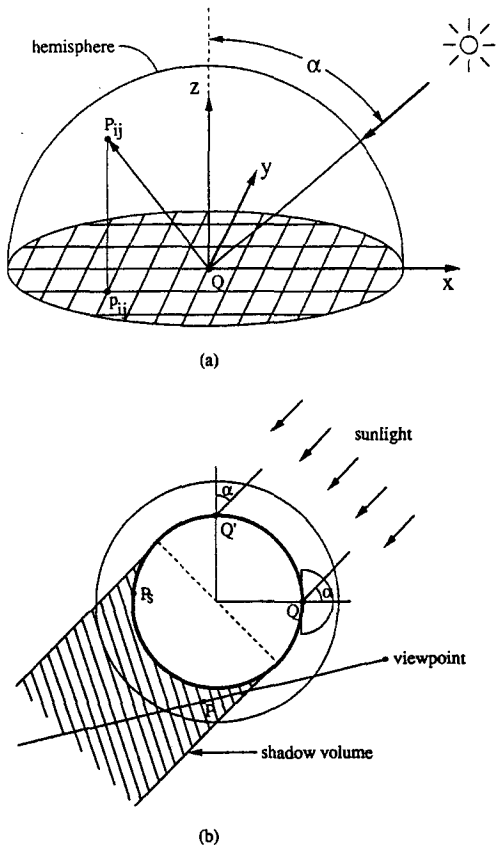



Figure 3: Calculation of sky light and shadow detection due to the earth.

attenuation of sunlight reaching the earth surface, $I_e$ is given by

$$I_e(\lambda) = r(\lambda)(cos\alpha\ I_s(\lambda)\ exp(-t(P_cP_b, \lambda)) + I_{sky}(\lambda, \alpha)), \tag{10}$$

where $r(\lambda)$ is the diffuse reflection of the earth, $\alpha$ the angle between the normal vector of the earth and light vector (sunlight), and $I_{sky}$ sky light. The direct component is small at the region where $\alpha$ is large (i.e., nearby the boundary of shadow) and tends to be reddish because of its long optical length.

Sky light is scattered light due to particles in the atmosphere. The radiance distribution of sky light can be obtained by setting the viewpoint on the earth in Eq.(8). As we are discussing the earth as viewed from space, shadows caused by obstacles on the surface are ignored, even though we take into account shadows due to the earth itself. That is, for shadow calculation, the earth is assumed to be a sphere with a smooth surface. Sky light due to scattered light from clouds is also ignored here. The illuminance at point $Q$ on the earth due to the whole sky is obtained by using the following method: let's consider an element on a hemisphere whose center is $Q$ (see Fig.3), calculate the intensity at each element on the hemisphere, and project each element onto the base of the hemisphere, then the illuminance is obtained by integrating the intensity of each element by weighting its projected area[13].

$I_{sky}$ is calculated as follows: as shown in Fig.3 (a), the base of the hemisphere is divided into a mesh. Let's consider point $P_{ij}$ on the hemisphere, which is mapped onto the hemisphere of the mesh point $p_{ij}$ inversely, and calculate the intensity in the direction of $QP_{ij}$. The illuminance due to the whole sky is obtained by adding intensities at every mesh point within the base circle of the hemisphere. As shown in Fig. 3(a), the $x$-axis is set so that the sun exists on the $x - z$ plane; the region in the half circle (e.g., $y > 0$) is enough to get $I_{sky}$ because of symmetry.

The radiance distribution of the sky is determined by angle $\alpha$ between the normal of the surface of the earth and the direction of the sunlight. Even though the direction of the sunlight is different at each point on the earth, the illuminance due to sky light (integrated values) at any point with the same angle $\alpha$ has the same value (e.g., $Q$ and $Q'$ in Fig.3). This means that the illuminance due to sky light at arbitrary angle $\alpha$ can be obtained by linear interpolation of a precalculated lookup table of $I_{sky}$. Note that $I_{sky}$ is not zero at regions where there is no direct sunlight ($\alpha > 90$ degrees, e.g., $P_s$ in Fig.3), so that $I_{sky}$ for $\alpha = 0$ to $\alpha = 110$ degrees must be prepared in the lookup table.

### 4.3.3 Detection of shadow caused by the earth

As shown in Fig.3 (b), point $P$ on the ray exists in the shadow region caused by the earth (we refer to it as a *shadow volume*), the scattered light in this region is zero because there is no incident light. Therefore it is sufficient to consider only attenuation in this region.

As the shadow volume is expressed by a cylinder, which is obtained by sweeping the circle (i.e., the contour of the earth viewed from the sun), the shadow segment on the ray can be calculated as the intersection segment between the cylinder and the ray.

### 4.3.4 Calculation of optical depth

The optical length of air molecules is calculated by numerical integration of Eq. (4) ( in the case of aerosols, the density distribution and the extinction coefficient are different). The optical length is calculated by trapezoidal integration of sampled density. The optical length at sampling point $P_i$ on the ray is obtained by adding the optical length of interval $P_{i-1}P_i$ to the optical length at $P_{i-1}$. Therefore the integration of the optical depth should start from the viewpoint. The optical length between the light source and point $P_i$ on the ray is also required (e.g., $PP_c$ in Fig.1). This calculation is required at every sampling point on the ray; optimization should be considered because of computational expense. We use a lookup table to save on computation time.

The density distribution of particles in the atmosphere varies exponentially with altitude. This means that the errors in the numerical integration become large when it is performed with a constant interval. Intervals which are inversely proportional to the density are desired; that is small intervals for low altitude and long intervals for high altitude. In order to realize this condition, the atmosphere is assumed as multiple spherical-shells. The radius of each sphere is set so that the difference in density between every adjacent sphere is within a given value. As a result, the difference between the radii of the shell is small for low altitude, and is large for high altitude, as shown in Fig.4. As Rayleigh scattering governs the calculation of optical length, the radius of each sphere is determined by the density distribution of air molecules. Let's consider N layers of spheres. The radius is given by(see Fig. 4)

$$r_i = H_0 log(\rho_i) + R, \quad \rho_i = 1. - i/N, \tag{11}$$

where $R$ is the radius of the earth. For $i = N$, $r_N$ is set to the radius of the atmosphere. For aerosols, the scale height is smaller than that for air molecules; aerosols mainly exist at low altitude. Therefore aerosols exist in the dense radii of shells; this fact assures the correctness of the above mentioned algorithm.

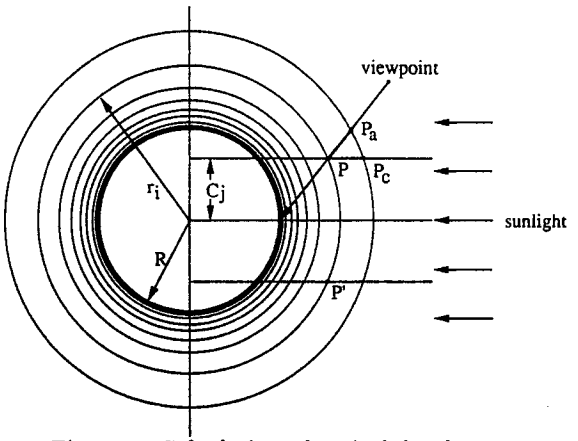



Figure 4: Calculation of optical depth.

The sampling points used in the integration are employed as the intersection points between the ray (view sight or light ray) and the multi-imaginary spheres and these intersection are easily obtained. The density at every sampling point is easily found from the lookup table indexed by the index numbers of the sphere, which is easily get from the altitude of the point.

The optical length between the sun and an arbitrary point on the ray can easily be precalculated because the earth is a sphere and sunlight is parallel light. As shown in Fig.4, let's consider a cylinder defined by sweeping the circle which passes through the center of the earth and is perpendicular to the light direction. Every optical length at the intersection (i.e. circle) between the cylinder and each one of the multi-imaginary spheres is equal (e.g., $P$ and $P'$ in figure). The optical lengths at the intersection points between the cylinders with radius $C_j$ and the spheres with radius $r_i$ is calculated(e.g.,$P_aP$ in fig.) and are stored in the lookup table. The optical depth at arbitrary point $P$ on the ray is easily calculated by linear interpolation, after the radius of the cylinder including $P$ and the radius of the sphere are calculated. The lookup table here is 2D array: $[r_i, C_j]$. After getting indeces $i$ and $j$ from point $P$, the optical depth can be obtained by linear interpolation from $[r_i, C_j]$,$[r_{i+1}, C_j]$, $[r_{i+1}, C_{j+1}]$,$[r_i, C_{j+1}]$.

As described above, the light intensity of one wavelength reaching the viewpoint can be calculated by numerical integration with respect to pass length. Therefore the light intensity in the range of visible wavelengths ( r, g, b in this paper) can be calculated.

# 5 THE COLOR OF CLOUDS

Since the geometric modeling of clouds is not our main subject, we are displaying the earth as viewed from space, clouds are simply modeled by applying 2D fractals. That is, the density distribution of clouds is expressed by mapping the fractal images of the necessary Mandelbrot set (0.39032+ 0.23775i is used in this paper)[15]. To take into account clouds with various altitudes, multiple imaginary spheres are employed to map fractal images on them.

Their color is determined by the following two light paths. One is on the light which passes through the atmosphere of scattered light due to cloud particles, again passing through the atmosphere, and reaches the viewpoint. Another one is on the light which passes through the atmosphere, reflected light at the earth's surface is attenuated by cloud particles, again passing through the atmosphere. Multiple scattering in clouds is ignored here.

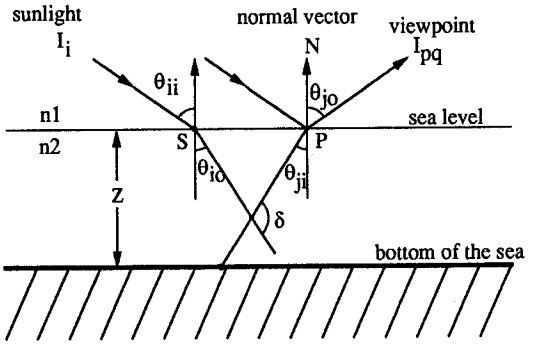



Figure 5: Calculation of color of water surface.

The size of particles in clouds is larger than that of air molecules or of aerosols. Light scattered by such large particles is little influenced by wavelength. ( However, the spectrum of incident sunlight onto clouds depends fairly strongly on the sun position.) The light reflected from clouds depends on the phase function ( the angle between the view vector and light vector); the phase function is expressed by Eq.(5) (see reference[18] on the value $u$). In the case of clouds not being illuminated by the sunlight because of the shadow due to the earth; the shadow detection is executed by using the shadow volume described before. The shadows on the earth due to clouds are ignored in this paper. In the near future, a more precise model for clouds is slated in order to get images of the earth viewed from relatively close to the earth's surface.

# 6 COLOR OF THE SEA

Let's consider the light reaching a viewpoint from the surface of the sea, There are three paths (see Fig. 5): (1) reflected light on the water surface, (2) scattered light due to particles within the water leaving the water surface (3) attenuated light passing through the sea after reaching the bottom of water.

Calculation methods of the color of water have been developed by Max[8], Fournier[2], Ts'o[17], and Mastin[7] . However their methods focused on (1) and shapes of waves, and did not refer to (2)(scattered light due to particles in the water). The method proposed here takes into account (1) and (2). Furthermore the attenuation of the light passing through the atmosphere is taken into account. For (3), the light from the bottom of the sea can be neglected because of the depth of the sea.

When the light is incident to the water surface, the light path is divided into reflection and refraction. The relation between the reflection and refraction on the water surface obeys Fresnel's law of reflection. Incident light is refracted at the water surface; the relation between the incident angle and reflection angle obeys Snell's law. The refracted light is scattered/absorbed by water molecules in the sea, and reaches the viewpoint after refracting at the water surface again. For this phenomena, Gordon and McCluney [3, 9] proposed a quasi-single-scattering (QSS) model based on the radiative transfer equation. However, in the model the sun's position is limited to the zenith. We improved upon this. The light intensity transmitted in water, $I_{PQ}$, is given by

$$I_{PQ}(\theta_{ii}, \theta_{io}, z) = \frac{I_i(\lambda)T_i(\theta_{ii}, \theta_{i0})T_o(\theta_{ji}, \theta_{jo})\beta(\delta, \lambda)}{n^2(\cos\theta_{io} + \cos\theta_{ji})c(\lambda)[1 - \omega_0(\lambda)F(\lambda)]}$$
$$\times(1 - exp(-zc(\lambda)[1 - \omega_0(\lambda)F(\lambda)](\sec\theta_{ji} + \sec\theta_{io})),$$

(12)

where $\lambda$ is wave length, $z$ the depth of the sea, $\theta_{ii}$ the angle between the surface normal at point $P$ and the direction of the viewing direction, $\theta_{io}$ the angle between the direction of the zenith and the direction of incident sunlight, $\theta_{jo}$ the angle between the reverse direction of the zenith and the sunlight after refraction, $I_i(\lambda)$ the irradiance of sunlight just above the water surface, $n$ the refractive index of water, $T_i$ and $T_o$ the transmittance of the incident light at point $S$ and $P$, respectively, $c(\lambda)$ the attenuation coefficient of light which expresses the ratio of lost energy of light when the light travels a unit length, $\beta$ a volume scattering function $\omega_0$ the albedo of water , and $F$ the fraction of the scattering coefficient in a forward direction. Data of $\beta, \omega_0$, and $F$ used in this paper is obtained from [10]. Eq. (12) shows that the color of water depends on the depth, the incident angles and viewing direction. The surface of the sea is not flat, and is a spherical surface (i.e., the normal vector of each point on the surface is different); the color of the sea varies according to the position because the incident and viewing angles to the surface normal at each position are different.

As described above, both the incident light to the sea and the color (intensity) of the sea are attenuated by the atmosphere. By using the same method as described in 4.3.2, this effect can be calculated by taking into account two optical lengths; from the sun to the surface and from the surface to the viewpoint.

# 7 EXAMPLES

Fig. 6 shows an example of the color of the atmosphere. The color of the earth is assumed to be black in order to demonstrate the atmospheric color only. The position of the sun is behind and to the left of the observation point. Even though the earth is assumed to be a black body, it looks blue, and the boundary of the earth is white.

Fig. 7 shows the images of the earth with texture-mapped continents viewed from space; the location of the observation is at altitude 36,000 km, which corresponds to the altitude of the Japanese weather satellite called *Himawari*, at 135°E 0° N and the direction of the sun is 70° E 20° N. In Fig. (a), the color of the sea, direct sunlight, and sky light are taken into account, but the attenuation from the earth to the viewpoint is ignored(i.e., it corresponds to the color when the observer stands on the earth). In Fig. (b), atmospheric scattering/absorption is also taken into account (i.e., the color of the atmosphere is added). In Fig. (c), clouds are added.

Figs. 8,9 show examples of the earth viewed from relatively close-by; the viewpoint is at altitude 500km at 0° E 60° N. The direction of the sun in Fig. 8 is 0° E 20N, and the directions of the sun in Fig. 9 are 200° E 20°N and 240° E 15°N. Fig. 8 corresponds to noon(daytime), and Fig. 9 correspond to evening or dawn sky. In Fig. 9(b), one can observe the shadow (the dark part in the red atmosphere) due to the earth. The color of clouds changes to red due to the change of color of direct sunlight. These examples depict beautiful variations in color of the earth and the atmosphere. The space shuttle in the figure consists of 178 Bézier patches.

Let's show the photographs taken by the first Japanese astronaut aboard space shuttle, Dr. M. Mouri(NASDA), in Fig.10 (altitude 300km, September, 1992). Fig.11 displays the results of our simulation. One may observe differences between the photos and the simulation results. One of the reasons on Fig.11(a) may be due to the poor modeling of clouds and lands. In Fig.(b) some horizontal layers(e.g., orange color) are observed, one of them may be aerosols due to explosion of Volcano in Philippine. These facts suggest the necessity for further researching.

For hidden surface removal, the scanline algorithm for curved surfaces [11] is employed, and for anti-aliasing the multi-scanning algorithm[14] is employed. The calculation was done on an IRIS Indigo Elan. The computation times for Fig.7 (c) and Fig. 9 were 3.8 minutes and 12.0 minutes, respectively(image size=500 x 490).

# 8 CONCLUSION

We have proposed an algorithm for physically-based image synthesis of the earth viewed from space. As shown in the examples, the proposed method gives us photo-realistic images taking into account the color of the earth, clouds, and the sea. The advantages of the proposed method are as follows:

(1) The spectrum of the surface of the earth is calculated by taking into account direct sunlight and sky light as affected by atmospheric scattering.
(2) The spectrum of the atmosphere is calculated by taking into account absorption/scattering due to particles in the atmosphere.
(3) The spectrum on the surface of the sea is calculated by taking into account radiative transfer of water molecules.
(4) The optical depth and illuminance due to sky light are efficiently calculated by using several lookup tables taking advantages of the facts that the earth is spherical and that sunlight is parallel.

**Acknowledgment :** The authors would like to acknowledge A. Wakayama (currently Fujitsu Co.) for his help in coding of the prototype of our program.

## References

[1] J.F. Blinn, "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces," *Computer Graphics*, Vol. 16, No. 3 (1982),pp. 21-29.

[2] A. Fournier, " A Simple Model of Ocean Waves," *Computer Graphics*, Vol. 20, No. 4,(1986),pp. 75-84.

[3] H.R, Gordon, " Simple Calculation of the Diffuse Reflectance of the Ocean," *Applied Optics*, Vol. 12, No. 12,(1973),pp. 2803-2804.

[4] J.T. Kajiya, "Ray tracing Volume Densities," *Computer Graphics*, Vol.18, No.3 (1984) pp.165-174.

[5] RV Klassen, "Modeling the Effect of the Atmosphere on Light," *ACM Transaction on Graphics*, Vol. 6, No. 3,(1987),pp. 215-237.

[6] LINKS Corporation, leaflet of "LINKS CG LIBRARY", (1991)

[7] G. A. Mastin, P. A. Watterberg, and J. F. Mareda,. " Fourier Synthesis of Ocean Scenes," *IEEE Computer Graphics & Applications*, Vol. 7, No. 3,(1987),pp. 16-23.

[8] N. Max, "Light Diffusion through Clouds and Haze," *Graphics and Image Processing*, Vol.33, No.3 (1986) pp.280-292.

[9] McCluney, W.R. Ocean Color Spectrum Calculations. *Applied Optics*, Vol. 13, No. 10,(1974),pp. 2422-2429.

[10] N. G. Jerlov, "Optical Oceanography," *Elsevier, Amsterdom* (1968).

[11] T. Nishita, K. Kaneda, E. Nakamae, " A Scanline Algorithm for Displaying Trimmed Surfaces by Using Bézier Clipping," *The Visual Computer*, Vol.7, No.5 (1991) pp.269-279.

[12] T. Nishita, Y. Miyawaki, E. Nakamae, "A Shading Model for Atmospheric Scattering Considering Distribution of Light Sources," *Computer Graphics*, Vol. 21, No. 4,(1987),pp. 303-310.

[13] T. Nishita, E. Nakamae, "Continuous tone Representation of Three-Dimensional Objects Illuminated by Sky Light," *Computer Graphics*, Vol. 20, No. 4,(1986),pp. 125-132.

[14] T. Nishita, E. Nakamae, "Half-Tone Representation of 3D Objects with Smooth Edge by Using a Multi-Scanning Method," *J.Information Processing(in Japanese)*, Vol.25, No.5,(1984), pp.703-711.

[15] K. Sato, "Fractal Graphics," *Rassel Co.(in Japanese)* (1989) p.74

[16] S. Sekine, "Optical characteristics of turbid atmosphere," *J Illum Eng Int Jpn*, Vol.71, No.6, (1987) pp.333

[17] P. Y. Ts'o, B. A Barsky,. " Modeling and Rendering Waves: Wave-Tracing Using Beta-Splines and Reflective and Refractive Texture Mapping," *ACM Transactions on Graphics*, Vol. 6, No. 3,(1987),pp. 191-214.

[18] W.M. Cornette, J.G. Shanks, "Physical reasonable analytic expression for the single-scattering phase function," *Applied Optics*, Vol.31, No.16 (1992), pp.3152-

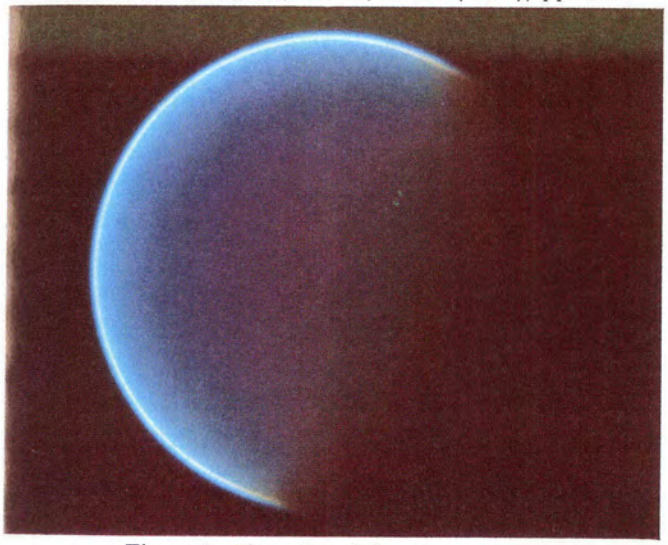


Figure 6: The color of the atmosphere.

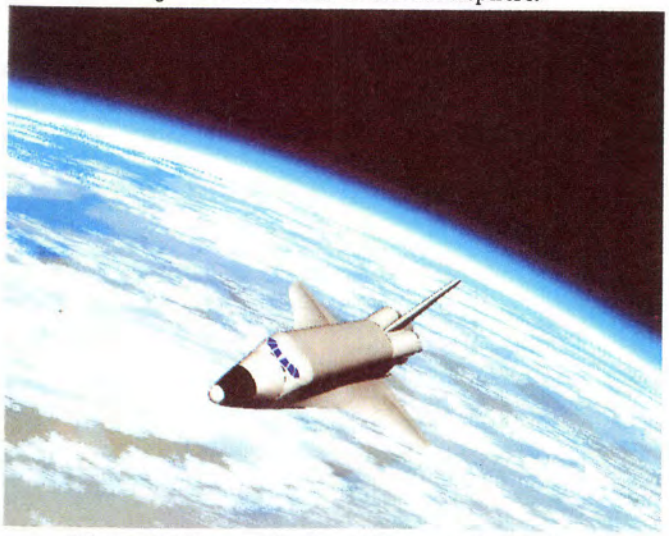
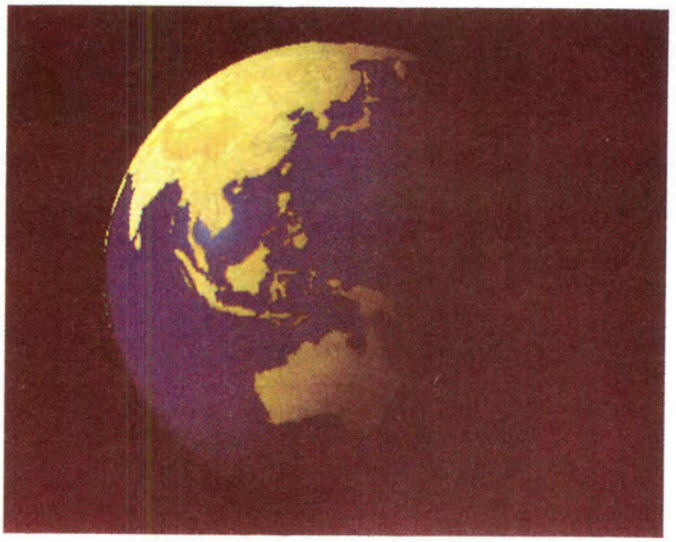
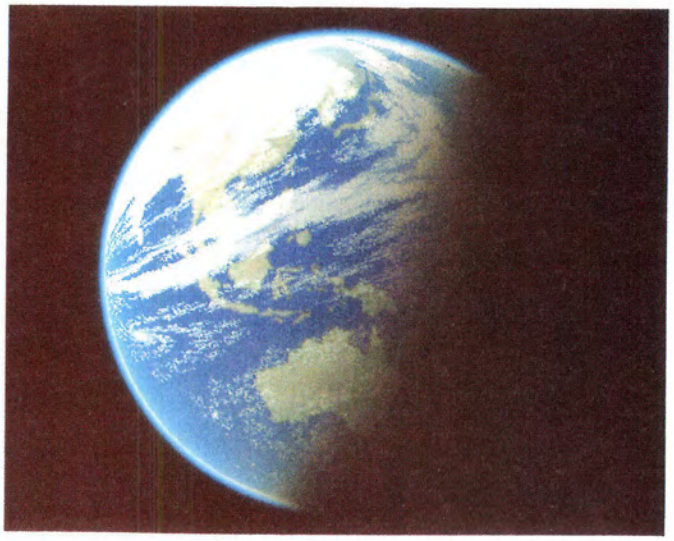


Figure 8: The earth viewed from relatively close-by.



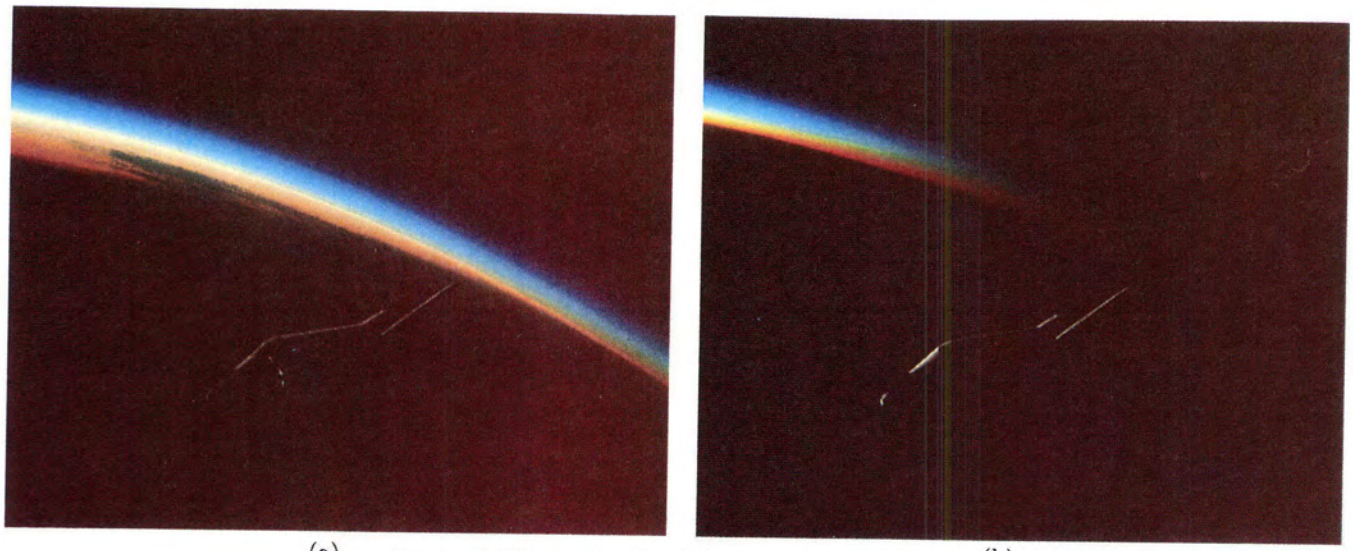
(a)

(b)



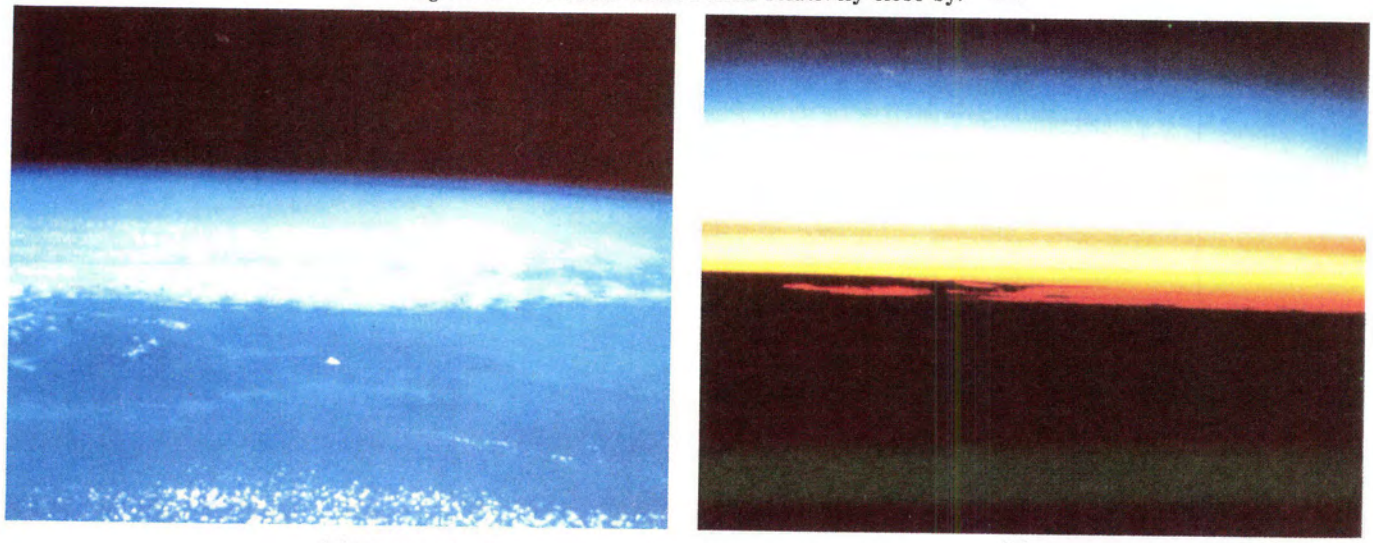
(c)

Figure 7: The earth viewed from space.



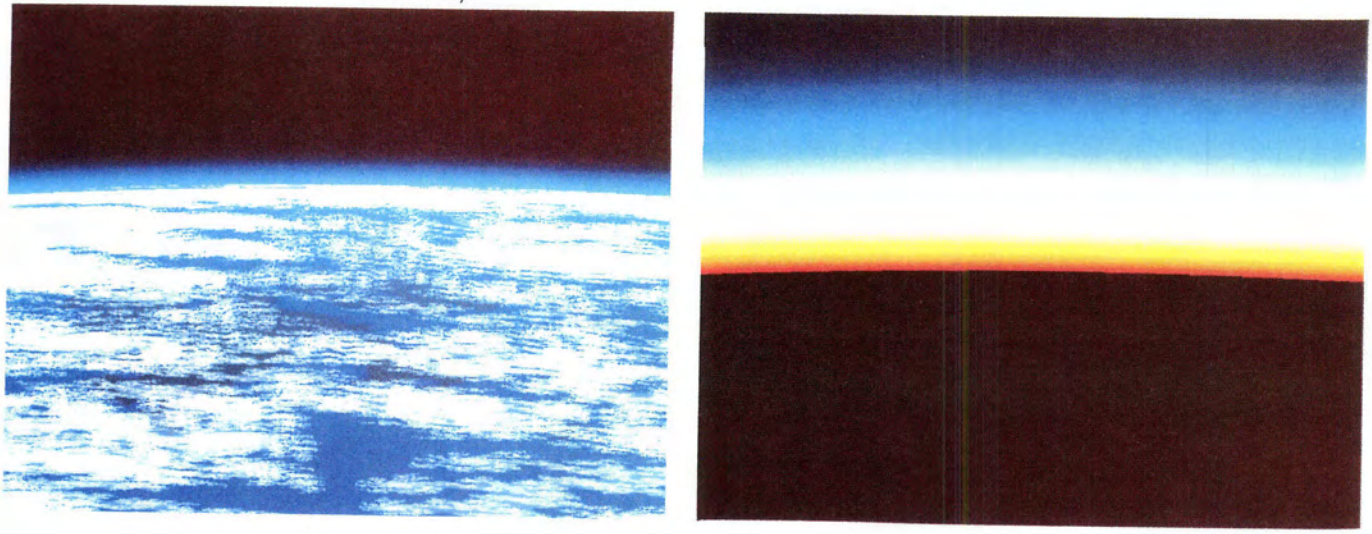
(a) (b)

Figure 9: The earth viewed from relatively close-by.



(a) (b)

Figure 10: Real photographs from space shuttle (courtesy of NASA).



(a) (b)

Figure 11: Comparisons with simulation.



# Smooth Transitions between Bump Rendering Algorithms

Barry G. Becker[1]
Nelson L. Max[2]

University of California, Davis
and
Lawrence Livermore National Laboratory

## ABSTRACT

A method is described for switching smoothly between rendering algorithms as required by the amount of visible surface detail. The result will be more realism with less computation for displaying objects whose surface detail can be described by one or more bump maps. The three rendering algorithms considered are a BRDF, bump-mapping, and displacement-mapping. The bump-mapping has been modified to make it consistent with the other two. For a given viewpoint, one of these algorithms will show a better tradeoff between quality, computation time, and aliasing than the other two. The decision as to which algorithm is appropriate is a function of distance, viewing angle, and the frequency of bumps in the bump map.

**CR Categories:** I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.5 [Computer Graphics]: Three-Dimensional Graphics and Realism.

**Keywords:** animation, BRDF, bump map, displacement map, rendering, surface detail, volume texture.

## 1. INTRODUCTION

Objects in animation are sometimes distant specks; at other times a tiny part of one will fill the whole screen. If these objects have rough surfaces, the same rendering algorithm should not be used in both cases. Almost all real materials have a hierarchy of surface detail. We assume that the macro-structure of all objects is described by parameterized patches or a polygonal mesh. The micro-structure is then described by one or more bump tables for each level of detail below the geometrical, each giving bump height as a function of the 2-D surface parameters. An alternative way to describe the surface detail is through the use of volume textures to specify bump height as a function of 3-D coordinates[10, 12].

LLNL, P.O. Box 808/L-301, Livermore, CA 94550
1. (510)422-3724 becker@mozart.llnl.gov
2. (510)422-4074 max2@llnl.gov

The Bidirectional Reflection Distribution Function or BRDF[13, 14, 6] captures the surface properties which are too small to be visible. Most real surfaces are neither purely specular (mirror-like) nor purely diffuse, but rather somewhere in between. To represent this non-trivial distribution of light reflectance a BRDF is used. It can be represented by a table indexed by a lighting direction and a viewing direction, to give the reflectance as a function of these directions. The BRDF used for this research is constructed from distributions of normals recorded from various views of a single displaced surface patch.

Bump-mapping[2] is an inexpensive way to achieve a good approximation to macroscopic surface roughness. The parameterized surface is treated as smooth for the purpose of visible surface determination, while the surface normals are perturbed to a first order approximation of what the actual bump normals would be.

The third algorithm, displacement-mapping[4, 5], is used when any shortcut in computation will be noticeable to the eye. Displacement-mapping is different in that the surface is actually offset by the appropriate bump height so that the full 3-D geometry can be rendered. For purposes of maintaining consistent shading, the same approximated normal is used to shade the displaced surface as was used in the bump map. However, now it is applied to the displaced surface rather than to the flat parametric one.

Bump-mapping is good for economically rendering bumps which can be described as a height field. Unfortunately it does not account for occlusion. It is necessary to modify flat bump-mapping so that it yields images statistically similar to images produced by the other two methods. This revised procedure will be termed 'redistribution bump-mapping' because it redistributes the normals in a way that is statistically similar to those seen on the displaced surface viewed from a specific direction.

The three methods are blended together so that the parts of the scene which are close to the viewer, or close to the extreme edge (silhouette), would be displacement-mapped, since this is where missing detail would be noticed most. Smooth silhouette edges are an artifact of bump mapping which is easy to detect. Parts farther away, or whose normals are parallel to the viewing direction, will be bump-mapped. When surfaces have microscopic material-specific qualities or are very far from the viewer, they are rendered using a BRDF. More

specifically, for a given scene, those features with a spatial frequency higher than one half cycle per pixel (the Nyquist limit) are considered in the BRDF. At the other end of the spectrum, features that are large enough to cause noticeable occlusion need to be displacement-mapped. The parts in between are rendered with varying degrees of redistributed bump-mapping. Most importantly, there is a smooth transition among the three. The effect is that the whole scene looks as if it were displacement-mapped, when in fact much of it was rendered with cheaper algorithms. Extending this concept we can have high frequency rough surfaces on top of low frequency rough surfaces, each bumpy level of detail having three rendered representations.

In Figure 1 we see a teapot rendered in the four different ways. All renderings are based on the same height function. A major consideration for a smooth transitions among these is the consistency of the shading between methods. The amount of light emitted by a surface rendered with one method does not necessarily equal that amount emitted by the same surface rendered with another. Nor is the distribution of that light necessarily equivalent. A key aspect of this research is the determination of how the varying algorithms need to be modified in order to have their overall area-averaged light intensity contributions consistent.

There are five reasons why the average reflected intensity from a bump-mapped image is inconsistent with the reflected intensity from either the BRDF rendered image or the displacement-mapped image of the same object. Usually the BRDF is constructed under the assumption that the microfeatures of the surface are composed entirely of specular, mirrored facets. Bump- and displacement-mapping contain both specular and diffuse components. The easy solution to this inconsistency is to include a diffuse component for each microfacet when constructing the BRDF for the highest frequency bumps. Usually there is an inconsistency between bump- and displacement-mapping because actual surface displacement creates a geometrically computed facet normal for the shader while the perturbed normals for bump maps are only approximations. As previously mentioned this is overcome by using the approximated bump-mapped normals on the displaced surface. The approximated bump normals also vary more smoothly than the facet normals, especially with our quadratic interpolation, which is smoother than Blinn's approximation[2]. Note that if a procedural displacement function is employed, it is possible to compute the surface normal analytically. Since the BRDF is constructed from a displacement-mapped patch, the same inconsistency may arise for it. Again the solution is remedied by using the bump normal for tabulating the BRDF. The most difficult consistency problem is caused by occlusion. Occlusion, which is the hiding of some bumps by others, can change the distribution of visible surface normals. A solution is presented which redistributes bump normals so they match a distribution of normals similar to one derived from displacement-mapping. Lastly, there is the problem of consistency of shadowing. We have not yet found a general solution for shadowing, so we draw our images and compute our BRDF without it.

The concept of blending between methods is not new. The difficulty in overcoming the intensity distribution inconsistencies is perhaps the main reason why there are few coded examples. Kajiya[8] mentioned a hierarchy of scale which is appropriate for modelling the complexity of nature. He states that each level of detail contains the three subscales discussed above. Westin *et al.*[14] describes these levels as the geometrical, milliscale, and microscale. Perlin[11] proposed a method to shift between the BRDF and perturbed normals. Perlin's method does not include an explicit height table for determining the new normals, making displacement-mapping difficult. Fournier[7] has presented a promising approach for filtering normal maps by recording a discrete number of Phong peaks.

The software for each of the three algorithms described in this paper has been combined according to the previously discussed considerations. The result is an animation which explores a surface from changing distances and directions, showing that there are no significant side effects while transitioning between renderers. For more detail concerning the implementation refer to Becker[1].

## 2. BASIC ALGORITHMS

### 2.1 Bidirectional Reflection Distribution Functions

The BRDF is used to capture the microscopic reflectance properties of a surface. The BRDF itself can be a table of reflectivities or it can be represented by a spherical harmonic series approximation[3, 14]. It is a function of either three or four variables representing the polar and azimuthal angles of the light rays. The polar angle is called $\theta$ and it measures the angle away from the normal. Its domain is $[0, \pi/2]$. The azimuthal angle is denoted by $\phi$ and has domain $[0, 2\pi)$, with 0 and $2\pi$ both in the direction of the viewer. An isotropic surface is one for which the emitted intensity does not vary as the surface is rotated radially about its surface normal. If only isotropic textures are used, then the arguments to the BRDF reduce to the two polar viewing directions and the difference in the azimuthal angle between the viewing and lighting directions. In the most general anisotropic case, the BRDF is a function of viewing direction and lighting directions, requiring all four angles.

There are several different ways to construct a BRDF. Cabral[3] constructed the BRDF directly from a bump map using horizon tables. Westin *et al.*[14] ray traced a generalized 3-D surface sample in order to calculate the intensities for their BRDF. Our method uses normal distributions. They are already required in order to create redistribution functions for the new bump-mapping method. The same normal distributions are used to create the BRDF. Fournier[7] has also discussed normal distributions.

A normal distribution is obtained by tabulating sampled normals from a projected displacement-mapped flat patch. The range of normals is a hemisphere. The hemisphere can be discretized into a finite number of $(\theta_N, \phi_N)$ bins. When the displacement map is projected, each pixel of the projected image represents a sample normal, and the count for the bin containing that normal is incremented. If bump-mapping is used to draw the flat patch, then the approximated normal distribution is independent of $\theta$. However, when looking from some direction with $\theta > 0$, self-occlusion may occur in the displacement-mapped image. This occlusion is accounted for by rendering the displacement-mapped geometry with a hardware z-buffer, coding the normal directions into the pixel colors. For grazing angles many potentially occluding patches may have to be rendered in order to get the occlusion correct on a single patch. The problem is solved by rendering a single patch using parallel projection, and then using a block read from the screen buffer to copy the patch to all the positions where it is needed, in a back to front ordering. In a postprocess the sample normals are scanned in and the distributions are created. These distributions will be used to find

the redistribution functions and to make the BRDF. The normal distributions are stored in a 3-D table. The first index is the viewing polar angle $\theta_V$. The second and third indices are the $\theta_N, \phi_N$ angles specifying the normal direction. For simplicity a table access is described by $distr[\theta_V, N]$, where $N = (\theta_N, \phi_{N-V})$, and $\phi_{N-V}$ denotes $\phi_N - \phi_V$. The difference between viewing and lighting $\phi$'s is denoted by $\phi_{V-L}$. To improve the statistics of the distribution, the patch is viewed in many $\phi_V$ directions for each $\theta_V$. The result is normal distributions for each $\theta_V$ which account for proper occlusion. To use these distributions in constructing the BRDF, the algorithm in Figure 2 is used.

```
for each level n from highest to lowest frequency
 for each θ_V
  for each θ_L
   for each φ_{V-L}
    {H = (V + L)/|V + L|
     for each θ_N
      for each φ_{N-V}
       if highest frequency BRDF
          { increment BRDF^n_diff[θ_V, θ_L, φ_{V-L}] by
              (L · N) distr^n[θ_V, N]
            increment BRDF^n_spec[θ_V, θ_L, φ_{V-L}] by
              (H · N)^Phong distr^n[θ_V, N]
          }
       else
          { compute θ'_V, θ'_L and φ'_{V-L}
            increment BRDF^n_diff[θ_V, θ_L, φ_{V-L}] by
              BRDF^{n-1}_diff[θ'_V, θ'_L, φ'_{V-L}] distr^n[θ_V, N]
            increment BRDF^n_spec[θ_V, θ_L, φ_{V-L}] by
              BRDF^{n-1}_spec[θ'_V, θ'_L, φ'_{V-L}] distr^n[θ_V, N]
          }
    }
```

**Figure 2.** The algorithm to compute the BRDF using a table of normal distributions.

Note that there are two components to the BRDF, one for the diffuse information and one for the specular. This way the amount of diffusivity and specularity chosen can be used as a parameter later. The $\theta'_V$ and $\theta'_L$ represent the angles between the viewing or lighting direction and the bin normal $N$, rather than with the flat surface patch normal. The angle $\phi'_{V-L}$ is the difference between $L$ and $V$ when projected to the plane perpendicular to the bump normal. It is computed by

$$\phi'_L = \arctan((L \cdot (N \times x)), (L \cdot (y \times N)))$$

$$\phi'_V = \arctan((V \cdot (N \times x)), (V \cdot (y \times N)))$$

$$\phi'_{V-L} = mod((\phi'_V - \phi'_L + \pi), 2\pi) - \pi \qquad (1)$$

where $x = (1, 0, 0)$ and $y = (0, 1, 0)$ are the axis directions of the bump table. This technique will give the same BRDF as if the combined displacement maps were used, as long as there is no correlation between the bumps at the different levels.

A smooth surface patch is rendered by interpolating the BRDF trilinearly in the angles $\theta_V, \theta_L$, and $\phi_{V-L}$. The indices for the table are computed from a local coordinate on the patch surface. The smooth surface normal points in the direction of $\theta = 0$. The origin of the azimuthal angle is the projection of the viewing direction onto the surface.

For a given patch parameterization, $P(u, v)$, the partial derivatives, $P_u = \frac{\partial P}{\partial u}$ and $P_v = \frac{\partial P}{\partial v}$, are rarely the same length (causing stretching), and not always perpendicular (causing warping). For these reasons special care must be taken when indexing the BRDF to determine an intensity. The method for computing the difference in azimuthal angle is as follows:

$$V_n = [V \cdot P_u, V \cdot P_v, 0]$$

$$L_n = [L \cdot P_u, L \cdot P_v, 0]$$

$$\Phi_{V-L} = \arccos(\frac{V_n}{|V_n|} \cdot \frac{L_n}{|L_n|}) \qquad (2)$$

The stretching will actually change the normal directions making the BRDF inaccurate. The BRDF would need to be recalculated to yield a theoretically correct result, but equation (2) does get the occlusion correct and gives nice anisotropic highlight effects in places where they would be expected.

### 2.2 Bump-Mapping

In Blinn's bump-mapping[2], the surface is not actually altered from its smooth parametric form, but it is shaded as though it were.

Blinn used a bump height table B to calculate a linear approximation to the bump normal at a point P on an object surface. If $\vec{P}_u$ and $\vec{P}_v$ are the partial derivatives as above, the unnormalized surface normal is $\vec{N} = \vec{P}_u \times \vec{P}_v$. In the bump map $B$, the partial derivatives $B_u$ and $B_v$ at the interpolated point corresponding to $P$ can also be computed using finite differences.

$$B_u = (B[u + \epsilon, v] - B[u - \epsilon, v])/(2 * \epsilon) \qquad (3)$$

and $B_v$ is similar. Each evaluation of $B$ uses bilinear interpolation.

Truncating insignificant terms, Blinn[2] has showed that the new normalized normal is very close to

$$\vec{N}' = \frac{\vec{N} + B_u(\vec{N} \times \vec{P}_v) - B_v(\vec{N} \times \vec{P}_u)}{|\vec{N} + B_u(\vec{N} \times \vec{P}_v) - B_v(\vec{N} \times \vec{P}_u)|} \qquad (4)$$

We have chosen to compute the bump map derivatives by a quadratic rather than linear scheme. Mach bands are eliminated by replacing Blinn's linear formula by a $C^1$ partial derivative formula, defined by taking the derivative of the $C^2$ cubic B spline curve approximation to the bump heights as a function of $u$ or of $v$. Let $du = u - \lfloor u \rfloor$, then

$$B_u = (-du^2/2 + du - .5)B[\lfloor u \rfloor - 1, v] + (3du^2/2 - 2du)B[\lfloor u \rfloor, v]$$

$$+(-3du^2/2 + du + .5)B[\lfloor u \rfloor, v] + (du^2/2)B[\lfloor u \rfloor + 2, v]$$

and $B_v$ is similar. Here each function evaluation requires only a linear interpolation in $v$. This method uses the same eight neighboring values in the height table as does (3), but with quadratic rather than linear weights.

The normals generated by this process do not lie in a distribution consistent with the other two algorithms. As previously discussed, $\vec{N}'$ must be further modified so that on average it will contribute to a normal distribution similar to displacement-map normals. This new algorithm, redistribution bump-mapping, is described in detail in Section 3.

It should also be noted that Perlin's volume textures[12], with the improvement by Max and Becker[10], can be substituted for bump maps when computing height values. The advantage of this is that there is no explicit parameterization to be concerned with, and thus no stretching to cause singularities or anisotropy. If a square patch has an isotropic texture mapped onto it, the texture becomes anisotropic as soon as the patch is stretched unevenly. Many parameterizations have singularities which lead to degenerate patches. If anisotropy is undesirable, then volume textures should be used. Perlin also used volume textures, and redistributed the normals to make them gaussian (personal communication) in his implementation of [11].

### 2.3 Displacement-Mapping

Displacement-mapping is the direct approach to rendering surface detail. For parameterized surfaces, each patch in the object has a $u$ and $v$ parameterization. The $u$ and $v$ coordinates are used as indices to look up height values in the bump height table. The corresponding vertex is then displaced along its normal vector by that height[4]. The normal generated from the bump approximation is also used on the displaced vertices. There is little loss of accuracy in doing this, and continuity during the transition is assured. Occlusion, the main problem with bump-mapping, is accounted for automatically when the vertices are displaced.

Having multiple bump maps for many levels of detail means the displaced bumps will be rendered with the BRDF constructed from the next bump map of higher frequency. To keep combined displacements consistent with BRDFs representing several combined bump maps, surface perturbations for the $i^{th}$ level must be perpendicular to the $(i-1)^{th}$ displaced surface. This means that for each vertex, $P_u$ and $P_v$ vectors must be computed for each level of detail which has been displaced. Since $P_u$ and $P_v$ are not necessarily perpendicular it is recommended that the following formula be used to compute them, given that the surface normal is $N$.

$$P_u[level+1] = P_u[level] + B_u[level]N[level]$$

where $B_u[i]$, $B_v[i]$ are the $i^{th}$ bump map partial derivatives. The equation for $P_v$ is similar.

## 3. REDISTRIBUTION BUMP-MAPPING

### 3.1 Normal Redistribution

The problem of eliminating inconsistencies between the different rendering models lies at the heart of making smooth transitions from one algorithm to another. Primarily we are concerned with keeping the integral of intensities equal over a small area on the surface while the rendering method changes.

Unfortunately, normals from bump-mapping do not yield a distribution similar to that of displacement-mapping or the BRDF. Since the polygon or patch itself is not displaced, it is possible to see normals which ought to be hidden by occluding bumps. In order to overcome this problem a redistribution function $q$ is created. This is a function which accepts as input a normal generated by Blinn's[2] bump approximation, and outputs a normal which is statistically consistent with the distribution used to form the BRDF.

Since the distribution of normals on a displacement-mapped flat patch is different for each viewing angle, it is necessary to have redistribution functions for each one. When the viewing angle is vertical, the identity function is used. When the viewing angle is just above the horizon, the redistribution of bump normals is necessarily quite drastic. The effect is to pull forward normals that might be facing away, and push upward those that might be hidden. This new scheme for doing bump-mapping might appropriately be termed redistribution bump-mapping.

### 3.2 Redistribution Function Construction

Suppose a bumpy surface is viewed from a direction with polar angle $\theta_V$. Let $g$ denote the distribution of normals $distr(\theta, N)$ at this fixed $\theta_V$, computed as above from the displacement map. Let $f$ denote the distribution of normals in a (non-displaced) bump-mapped image. Note that f is the same as $distr(0, N)$. If $q$ is the redistribution function described above, then the requirement that $q$ take the distribution $f$ to the distribution $g$ is that for any region $R$ in the hemisphere $H$ of possible normals,

$$\int_{q(R)} f(\theta,\phi)d\omega = \int_R g(\theta,\phi)d\omega \tag{5}$$

It is easier to explain how to specify $q$ in a 1-D case. So suppose $f(x)$ and $g(x)$ are two distributions on $[0,1]$, such that

$$\int_0^1 f(x)dx = \int_0^1 g(x)dx = 1 \tag{6}$$

The problem is to find $q : [0,1] \rightarrow [0,1]$ such that

$$\int_{q(a)}^{q(b)} f(x)dx = \int_a^b g(x)dx \tag{7}$$

where $a$ and $b \in [0,1]$. It is enough to guarantee that

$$\int_0^{q(b)} f(x)dx = \int_0^b g(x)dx. \tag{8}$$

Let

$$G(b) = \int_0^b g(x)dx$$

and

$$F(b) = \int_0^b f(x)dx.$$

Then

$$G(b) = F(q(b))$$

hence

$$q(b) = F^{-1}(G(b)). \tag{9}$$

The redistribution function $q$ maps a point $b$ so that the area under the curve before $b$ in $g$ is equal to the area under the curve before the point $q(b)$ in $f$.

The problem in 2-D can be handled similarly. One method is to define 1-D redistribution functions separately for $\theta$ and $\phi$. This gives adequate results for most bump maps, whose $\theta$ and $\phi$ distributions are fairly independent. This independence assumption is confirmed by the animation. For a more precise redistribution function, one can first redistribute $\phi$, and then for each fixed $\phi$, establish a separate redistribution function for $\theta$. For details see Becker[1].

## 4. TRANSITIONS

### 4.1 Partial Bump Displacement

For control of appearance and for smooth transitions we want the ability to change the height of the bumps in the bump map. This will alter the normal distribution and occlusion information. By close consideration we can see that the change can be accounted for without having to recalculate the redistribution functions every time the bump heights are altered. If the heights are multiplied by a factor $t$, then the tangent of the angle between the bump normal and the smooth surface normal should also change by a factor $t$; i.e., $\tan(\theta_{N_t}) = t \cdot \tan(\theta_N)$. The normal, $N = (\theta_N, \phi_N)$, needs to be replaced by $N_t = (\arctan(t \cdot \tan(\theta_N)), \phi)$. In order to keep the visibility information the same, the viewing angle, $\theta_V$, must be replaced with $\theta_W = \text{arccot}(\cot(\theta_V)/t)$. See discussion below concerning Figure 3.

The height of the bumps used to calculate the BRDF and redistribution functions must be the same as that of the bumps being rendered. This is because the BRDF is changed in a non-trivial way as the bump heights change. If we were only concerned with bump- and displacement-mapping, we could change the indexing on the redistribution functions to get the occlusion correct for changing bump heights. Unfortunately there is no easy way to re-index the BRDF to account for scale changes. Between the BRDF and redistribution bump-mapping, an intensity is computed for both methods. The resulting intensity is an interpolation of the two.

For the transition between bump- and displacement-mapping, intensity interpolation is not used, since it would cause the bump shading (particularly the highlights) to cross-dissolve rather than correctly adjust in position. As the bumps go from no displacement to full displacement the surface normals do not change, since they are always represented by Blinn's bump normal. The visible subset of bump normals does change, however, due to changing occlusion. Let *disp* be the transition parameter which gives the fraction of the full bump height. With $disp = 0$ all normals are seen, even those on the back of bumps. With $disp = 1$, only the visible subset of these normals are seen. In Figure 3 the segments of the visible surface are shown in bold. The redistribution of normals takes normals from standard bump-mapping into this visible subset. For partially displaced bumps there is a different subset of visible normals, but there is a relationship between the bump height and this subset which can be exploited to give the necessary redistribution.

Different redistribution functions for varying heights are not stored, only different functions for different viewing $\theta$'s. Fortunately the two are equivalent. For the fractional bump height, *disp*, we can determine a new $\theta_W$ for which the same distribution of full height bump normals will be seen. Figure 3 shows that the distribution of normals for this partially displaced surface, viewed from $\theta_V$, is identical to the distribution of visible normals for the fully displaced surface viewed from $\theta_W$. The slope of the line $V$ in Figure 3 is *disp* times the slope of line $W$, so $\cot(\theta_V) = disp \cdot \cot(\theta_W)$ and the formula for finding $\theta_W$ is:

$$\theta_W = \text{arccot}(\cot(\theta_V)/disp).$$

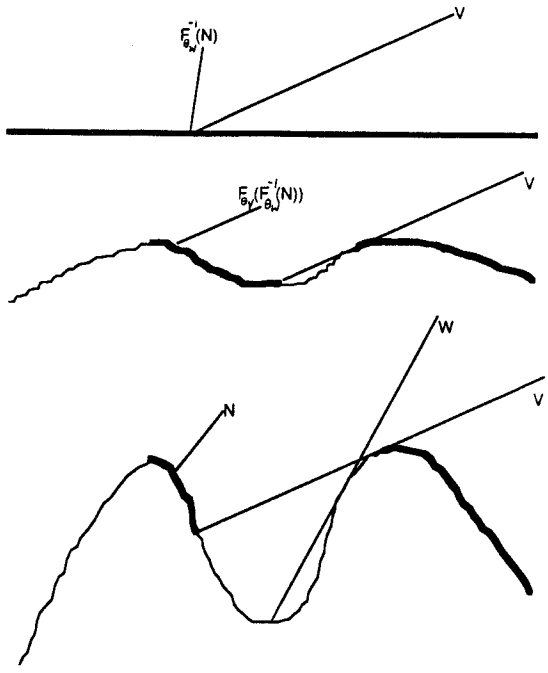



**Figure 3** Top: the non-displaced surface. Middle: surface displaced by bump height fraction *disp*. Bottom: Fully displaced surface.

The inverse redistribution function for $\theta_W$ is applied to take the visible bump normal from the partially displaced surface into a distribution similar to one from a flat bump-mapped surface. Next the redistribution function for $\theta_V$ is applied to that normal to take it all the way forward to match statistically a full displacement-mapped normal. Thus the change from bump-mapping to displacement-mapping is done through two table based function evaluations. Notice that as the bumps decrease in height, the new viewing $\theta_W$ approaches vertical. This means that the inverse function needs to alter the normals less in order to get them back to the bump-map distribution.

### 4.2 Algorithm Selection Criterion

Now that it is known how to modify the algorithms so that they will not deviate from a fundamental reflection model, it must be decided when to apply which algorithm. Clearly displacement-mapping should be applied when the view is close, and the BRDF when the view is far. The relationship is $1/d$, where $d$ is distance, since that is how the projected size of an object relates to distance. Another variable to consider is viewing angle, $\theta_V$. If $f$ is the wavelength of a feature then $f\cos(\theta_V)/d$ is the wavelength of the projected feature (in the direction of maximum foreshortening), and should be no smaller than two pixels. When the object is close, we would like to see a rough silhouette; when it is far, aliasing becomes a problem on the edge so use of the BRDF is desirable. This implies that as the object moves away from the viewer, the transition from displaced bumps to BRDF will be far more rapid on the object silhouette than on that area where the patch normal points toward the viewer. The

threshold at which the switch occurs is determined by a constant $D$. Summarizing these properties, we define a transition parameter

$$T(d, \theta_V) = (1/d - D)/(\cos(\theta_V) + \epsilon). \qquad (10)$$

Here $d$ is the distance from the viewpoint to the surface, $\theta_V$ is the angle between the viewing ray and the surface normal, and $D$ is dependent on individual bump maps. To avoid an instantaneous transition on the silhouette an $\epsilon$ is added to the cosine term in the denominator. The constant $D$ should be large if the highest frequency component of the bump map is large. Note that $D$ controls where the function changes from positive to negative, and thus lies midway between displacement-mapping and the BRDF. The formula for determining $D$ is

$$D = c \cdot freq \cdot S$$

where $freq$ is the highest frequency in the bump map and $S$ is the amount the $u$ and $v$ values are scaled. If $S$ is large, then the bump map will be repeated more times over the same area, and the partial derivatives, $P_u$ and $P_v$, are made shorter by a factor of $S$. The constant $c$ controls computational effort by globally shifting the scene toward more BRDF or alternatively more displacement. If shadows are included, the shadow terminator should be treated just like the silhouette. Areas far from the terminator are likely to be completely illuminated or shadowed, but on the terminator, displacement-mapping will make the shadowing exact. The parameter given by equation (10) determines the algorithm or algorithms used for rendering. Let the threshold values for choice of renderer be $e1 < e2 < 0 < e3 < e4$. If $T < e1$ then use the BRDF, if $T > e4$ then use displacement mapping, and if $e2 < T < e3$ use redistribution bump mapping. Values of $T$ other than these indicate regions where algorithms are blended. Values of -1, -.3, .3, and 1 respectively, were found to give good results.

### 4.3 Multiple Levels of Detail

With multiple levels of detail there are many more than two possible transition points. Many other cases need to be considered. The displacement-mapped image of the $i^{th}$ layer is rendered using the BRDF for the $(i-1)^{th}$ layer. As the camera continues to zoom in, the BRDF will switch to bump-mapping and then again to displacement-mapping.

Since each bump map has its own independent transition regions, some areas may have bump-mapping from two or more different levels. Perlin [11] suggests that each set of bumps be limited to a narrow range of frequencies. The result of implementing two levels of detail is shown in Figure 4. The bump map describing the surface detail is broken up into high and low order band-limited frequencies. The low frequencies compose the first level bump map and the high frequencies compose the second level. The left half of Figure 4 is color coded according to the algorithm used to render the most refined level of detail visible. Hence one can see bumpy sections colored yellow to indicate the BRDF from the next lower level was used to render the displaced bumps.

## 5. RESULTS

### 5.1 Consistency Comparison

In Figure 5 we can see the four rendering methods compared. The difference between the lighting and viewing $\phi$ is zero. Note that since the lighting and viewing directions are in alignment the patch becomes brighter for grazing angles. The rows are rendered with bump-mapping, redistribution bump-mapping, BRDF, and displacement mapping respectively. Note that redistribution bump-mapping is far more consistent with the BRDF and displacement-mapping than is ordinary bump-mapping. Figure 6 is a table which shows quantitative results for viewing angles corresponding to those shown in Figure 5.

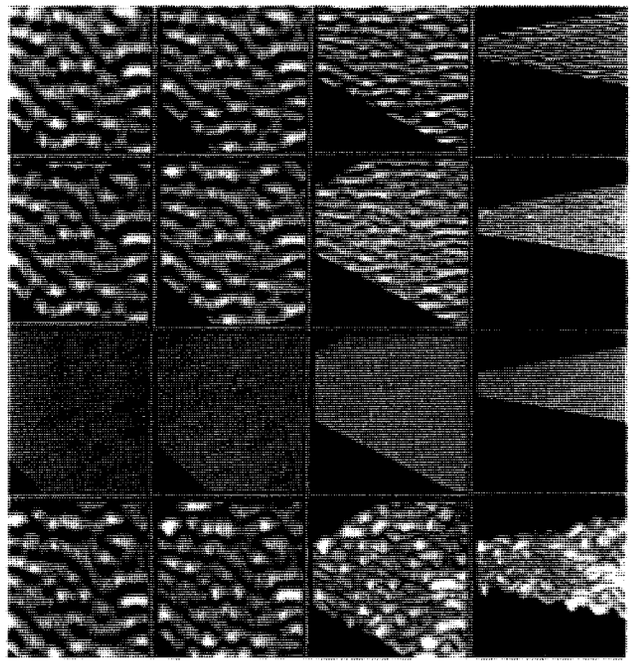


**Figure 5** Intensity comparisons. The lighting direction is consistently $\theta_L = \pi/4$. The rows from top to bottom represent bump-mapping, redistribution bump mapping, BRDF, and displacement mapping.

| | $\theta_v$=0 | $\theta_v$=π/6 | $\theta_v$=π/3 | $\theta_v$=4π/9 |
|---|---|---|---|---|
| Bump | 128 | 129 | 129 | 129 |
| Redistribution | 128 | 143 | 170 | 194 |
| BRDF | 129 | 146 | 172 | 192 |
| Displacement | 128 | 146 | 175 | 194 |

**Figure 6** Area averaged intensities for the diffuse component.

In Figure 7, a single flat patch is drawn in perspective. Regions in the foreground are clearly displacement-mapped. The middle region is redistribution bump-mapped, and the furthest edge is almost completely shaded with the BRDF. It should be apparent that there is no intensity inconsistency between methods and that the transition is smooth.

### 5.2 Conclusions

Combining displacement-mapping, bump-mapping and a BRDF into one algorithm makes it possible to explore great

scale changes, without changing the geometrical data base. Using a series of bump maps we can generate a variety of rough surfaces simulating different material properties. Objects in the scene will have a complex underlying structure but only the minimum amount of effort necessary to give the impression of complete geometrical representation will be expended. Current animations are restricted by the amount of geometrically represented detail. If the view gets too close to a feature, large drab polygons fill the display. With hierarchy of detail, the polygon level need never be reached, no matter how close the viewer gets. Even at intermediate and far distances the light interacts with flat polygonal surfaces as if they were truly composed of millions of smaller micro-polygons. As a result the otherwise drab polygons become alive with texture and interesting highlights. Those smaller micro-polygons may actually get rendered, but only if the viewer zooms in much closer.

### 5.3 Future Research

Shadowing is the main enhancement yet to be considered. One way to do the shadowing of displaced bumps is to use the two-pass z-buffer method developed by Williams[15]. Horizon mapping[9] has been shown to generate shadows for bump-mapped images. It will also work for redistribution bump-mapping since the horizon is determined by the $u$ and $v$ parameterization, not the normal. However, this may cause a problem since the rendering is according to a redistributed normal, and the shadows are according to the parameterization. The shadowing may look inappropriate for the rendered bumps. The shadowing for BRDFs can be done using horizon mapping, as was demonstrated by Cabral[3]. Another possibility is to use only the unshadowed normals from a displaced, rendered, and shadowed flat patch to generate the distributions for the BRDF and the redistribution function. The result should be consistent in terms of average intensity, but may not look qualitatively correct.

### 5.4 Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

## BIBLIOGRAPHY

1. Becker, Barry, "Smooth Transitions Between Rendering Algorithms During Animation", Master's thesis, University of California at Davis, Davis, CA, December, 1992.

2. Blinn, James F., "Models of Light Reflection for Computer Synthesized Pictures", *Proceedings of SIGGRAPH '77, Computer Graphics*, Vol. 11, No. 2, July, 1977, pp192-198.

3. Cabral, Brian, Nelson Max, and Rebecca Springmeyer, "Bidirectional Reflection Functions from Surface Bump Maps", *Proceedings of SIGGRAPH '87, Computer Graphics*, Vol. 21, No. 4, July, 1987, pp273-281.

4. Cook, Robert L., "Shade Trees", *Proceedings of SIGGRAPH '84, Computer Graphics*, Vol. 18, No. 3, July, 1984, pp223-231.

5. Cook, Robert L., Loren Carpenter, and Edwin Catmull, "The Reyes Image Rendering Architecture", *Proceedings of SIGGRAPH '87, Computer Graphics*, Vol. 21, No. 4, July, 1987, pp95-102.

6. Cook, Robert L, and Kenneth Torrance,"A Reflectance Model for Computer Graphics",*Proceedings of SIGGRAPH '81, Computer Graphics*, Vol. 15, No. 3, August, 1981, pp307-316.

7. Fournier, Alain, "Normal Distribution Functions and Multiple Surfaces",*GI '92 Workshop on Local Illumination*, 1992, pp 45-52.

8. Kajiya, James, "Anisotropic Reflection Models", *Proceedings of SIGGRAPH '85, Computer Graphics*, Vol. 19, No. 3, July, 1985, pp15-21.

9. Max, Nelson L., "Horizon Mapping: Shadows for Bump-mapped Surfaces", *The Visual Computer*, Springer-Verlag, Vol. 4, No. 2, 1988, pp109-117.

10. Max, Nelson L., and Barry Becker, "Bump Shading for Volume Textures", to appear in *IEEE Computer Graphics and Applications*, 1993.

11. Perlin, Kenneth, "A Unified Textural Reflectance Model", *Advanced Image Synthesis course notes, Proceedings of SIGGRAPH '84, Computer Graphics*, July, 1984.

12. Perlin, Kenneth, "An Image Synthesizer", *Proceedings of SIGGRAPH '85, Computer Graphics*, Vol. 19, No. 3, July, 1985, pp287-296.

13. Torrance, Kenneth, and Ephraim Sparrow, "Theory for Off-Specular Reflection from Roughened Surfaces", *Journal of the Optical Society of America*, 57(9), 1967, pp1105-1114.

14. Westin, Stephen H., James R. Arvo, and Kenneth E. Torrance, "Predicting Reflectance Functions from Complex Surfaces", *Proceedings of SIGGRAPH 92, Computer Graphics*, Vol. 26, No. 2, July, 1992.

15. Williams, Lance, "Casting Curved Shadows on Curved Surfaces",*Proceedings of SIGGRAPH '78, Computer Graphics*, Vol. 12, No. 3, July, 1978, pp270-274.

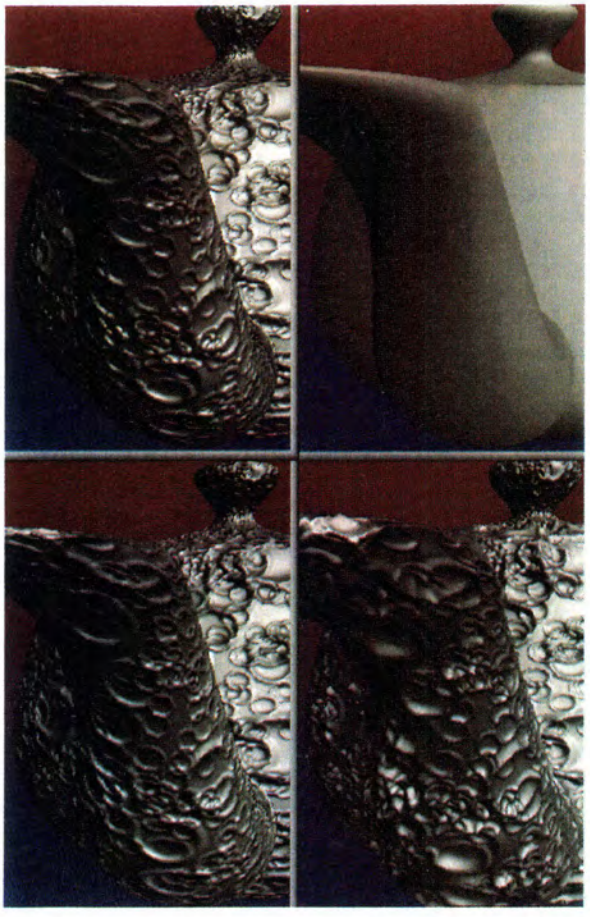


Figure 1. Counter-clockwise from upper left: bump-mapping, redistribution bump-mapping, displacement-mapping, BRDF. The difference between redistribution bump-mapping and plain bump-mapping is apparent near the bottom of the spout.

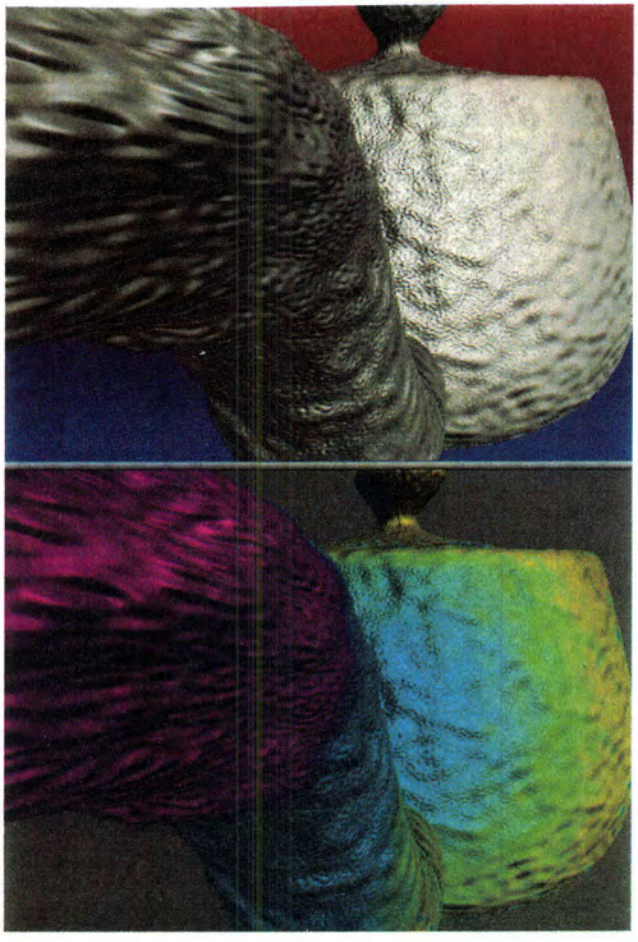


Figure 4. Two levels of bumpy detail. Colors in the bottom half indicate BRDF(yellow), redistribution bump-mapping(blue), and displacement-mapping(red) for the higher frequency bumps.

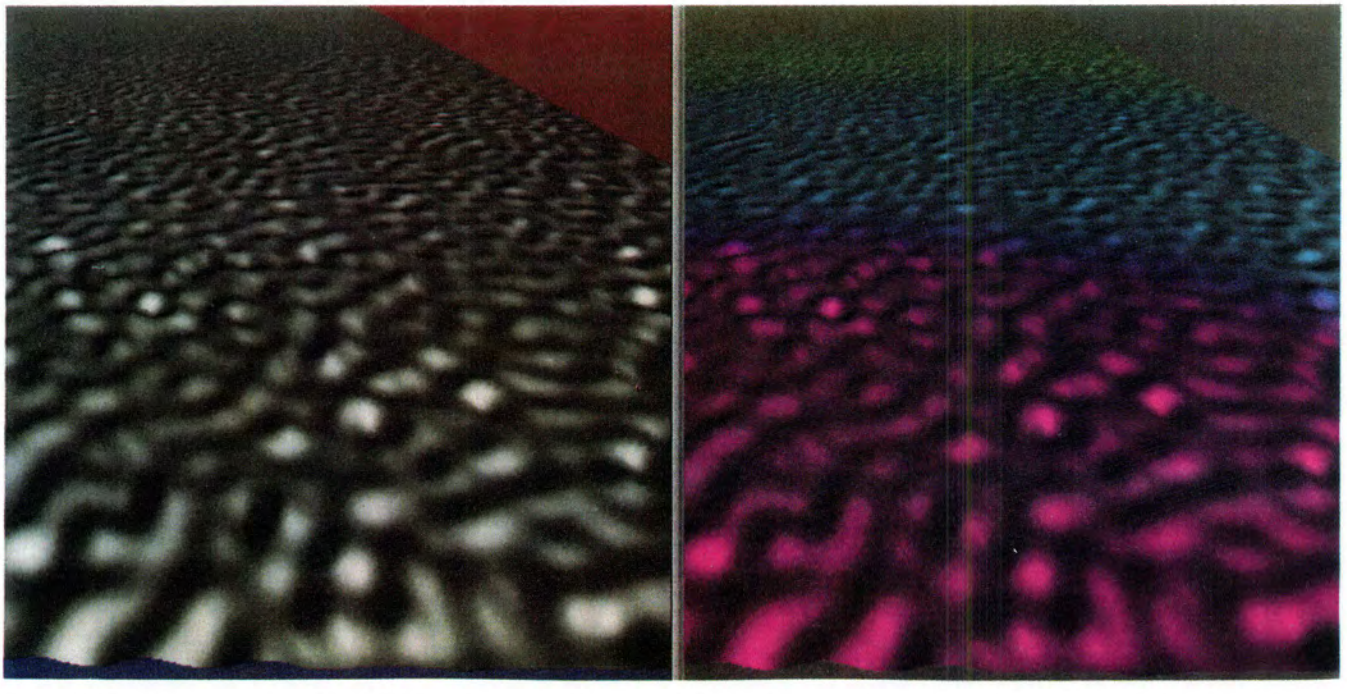


Figure 7. Transitions on a flat surface. BRDF(yellow) in the back, redistribution bump-mapping(blue) in the middle, and displacement-mapping(red) in the foreground.



# Linear Color Representations for Full Spectral Rendering

Mark S. Peercy
Department of Applied Physics
Stanford University

## Abstract

We present a general linear transform method for handling full spectral information in computer graphics rendering. In this framework, any spectral power distribution in a scene is described with respect to a set of fixed orthonormal basis functions. The lighting computations follow simply from this decision, and they can be viewed as a generalization of point sampling. Because any basis functions can be chosen, they can be tailored to the scenes that are to be rendered. We discuss efficient point sampling for scenes with smoothly varying spectra, and we present the use of characteristic vector analysis to select sets of basis functions that deal efficiently with irregular spectral power distributions. As an example of this latter method, we render a scene illuminated with fluorescent light.

**CR Categories and Subject Descriptors:** I.3.3 [**Computer Graphics**]: Picture/Image Generation–Display Algorithms; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism.

**Additional Keywords:** linear color representations, full spectral rendering, linear models, tristimulus values.

## 1 Introduction

Accurate color rendering in computer graphics must account for the full spectral character of the lights and surfaces within a scene. The rendering procedure must preserve enough spectral information to compute final values for output to some display device, such as an $RGB$ monitor. However, one wishes to minimize the computational cost of the rendering to reduce the time required to create an image. Therefore, one desires efficient methods of handling full spectral information during image synthesis.

Author's address: Dept. of Applied Physics, Stanford University
Stanford, CA 94305-4090
peercy@kaos.stanford.edu (415)725-3301



Some suggested techniques in dealing with full spectral information include the use of the tristimulus values for the lights and surfaces [1], the use of polynomial representations of spectra [16], and the use of linear models of surfaces and lights [20] [12]. The typical method employed is point sampling of the surfaces and the lights at a given number of wavelengths. These point samples are used in a numerical integration method to compute approximate tristimulus values before being transformed to values appropriate for display. To minimize the total number of samples, one seeks an efficient integration approximation; one approximation that has been studied in various forms is Gaussian quadrature [14] [19] [2].

In this paper, we consider a more general method for handling full spectral information in synthetic image generation; our technique is closely related to the use of linear models presented in [20]. The principal idea is that we describe the spectral power distribution of the light at every step of the rendering procedure with respect to a single collection of orthonormal basis functions. This formalism encompasses point sampling, which uses delta functions as its basis functions.

The constraint of describing all of the spectral power distributions with respect to the basis functions is advantageous for two reasons. First, it makes the rendering process completely linear. Therefore, this technique can be considered a generalization of point sampling and can be readily incorporated into standard renderers. Second, one has the freedom to select any orthonormal set of basis functions. This freedom can be exploited to increase the efficiency of the rendering process.

The body of this paper is divided into two main sections. In Section 2 we discuss the mathematical formalism of linear color representations of the lights and surfaces, and in Section 3 we address the problem of selecting appropriate basis functions. In this latter section, we discuss Riemann summation for efficient point sampling in scenes with smoothly varying spectra, and we present the use of characteristic vector analysis to provide efficient basis functions for scenes with complex spectra.

 ACM-0-89791-601-8/93/008/0191 $01.50

## 2 Linear Color Representations

During the rendering process, we demand that any spectral power distribution in the scene be described by $m$ orthonormal basis functions $E_i(\lambda); i = 1, ..., m$. By any distribution, we mean not only the light coming directly from a light source but also light that has been once, twice, or an arbitrary number of times reflected from surfaces in the scene. In this section, we use this restriction to derive the color representations of both the spectral power distributions and the surfaces, and we discuss the transformation of this color information to values appropriate for display.

### 2.1 Spectral Power Distributions

To obtain a representation for the spectral power distributions in a scene, we can project the spectral power distribution, $I(\lambda)$, of any light source onto the subspace spanned by the basis functions;

$$I(\lambda) = \sum_{i=1}^{m} \epsilon_i E_i(\lambda), \tag{1}$$

where

$$\epsilon_i = \int_\lambda I(\lambda) E_i(\lambda) d\lambda. \tag{2}$$

follows from the orthonormality condition. Thus, any light within the scene can be described with the $m$ elements $\epsilon_i$. These elements are simply the coefficients of the linear transformation defined by the set of basis functions, so we refer to this method as a general linear transform method.

### 2.2 Surface Reflectances

To obtain a representation for the surfaces, we project the spectral power distribution of the light reflected from those surfaces onto the set of basis functions (for clarity and without loss of generality, we neglect transmission and attenuation in this discussion). Lighting models typically divide the reflected light into three terms: ambient, diffuse, and specular ([6] discusses lighting models in detail); the spectral power distribution of light reflected from a surface, $I_o$, is given by

$$\begin{aligned} I_o(\Omega, \lambda) &= R_a(\lambda) I_a(\lambda) + G_d(\Omega) R_d(\lambda) I_s(\lambda) \\ &\quad + R_s(\Omega, \lambda) I_s(\lambda). \end{aligned} \tag{3}$$

Here, $\Omega$ denotes a general dependence on the geometry of the reflection, and $\lambda$ denotes a general dependence on wavelength. $I_a(\lambda)$ is the distribution of the ambient light, $I_s(\lambda)$ is the distribution of directional incoming light, and $G_d(\Omega)$ is the diffuse geometry term. $R_a(\lambda), R_d(\lambda)$, and $R_s(\Omega, \lambda)$ are the ambient, diffuse, and specular reflectances of the surface, respectively. In general, the specular reflectance is a function both of geometry and wavelength. However, empirical models often replace the specular reflectance with a separable term, resulting in a *piecewise separable* lighting model

$$\begin{aligned} I_o(\Omega, \lambda) &= R_a(\lambda) I_a(\lambda) + G_d(\Omega) R_d(\lambda) I_s(\lambda) \\ &\quad + G_s(\Omega) R_s(\lambda) I_s(\lambda). \end{aligned} \tag{4}$$

As described in Section 2.1, the ambient light and directional light are represented by their transform coefficients,

$$I_a(\lambda) = \sum_{i=1}^{m} \epsilon_i^a E_i(\lambda) \tag{5}$$

$$I_s(\lambda) = \sum_{i=1}^{m} \epsilon_i^s E_i(\lambda). \tag{6}$$

By using Equations 5 and 6 in Equation 3, the spectral power distribution reflected from a surface is

$$\begin{aligned} I_o(\Omega, \lambda) &= \sum_{i=1}^{m} \epsilon_i^a R_a(\lambda) E_i(\lambda) + \sum_{i=1}^{m} \epsilon_i^s G_d(\Omega) R_d(\lambda) E_i(\lambda) \\ &\quad + \sum_{i=1}^{m} \epsilon_i^s R_s(\Omega, \lambda) E_i(\lambda). \end{aligned} \tag{7}$$

To obtain the surface representations, we project this result back onto the the basis functions as in Equation 1;

$$I_o(\Omega, \lambda) = \sum_{j=1}^{m} \epsilon_j^o E_j(\lambda). \tag{8}$$

From Equation 2 and Equation 7,

$$\epsilon_j^o = \int_\lambda I_o(\Omega, \lambda) E_j(\lambda) d\lambda \tag{9}$$

$$= \sum_{i=1}^{m} R_{ij}^a \epsilon_i^a + G_d(\Omega) \sum_{i=1}^{m} R_{ij}^d \epsilon_i^s + \sum_{i=1}^{m} R_{ij}^s(\Omega) \epsilon_i^s \tag{10}$$

where

$$R_{ij}^a = \int_\lambda R_a(\lambda) E_i(\lambda) E_j(\lambda) d\lambda \tag{11}$$

$$R_{ij}^d = \int_\lambda R_d(\lambda) E_i(\lambda) E_j(\lambda) d\lambda \tag{12}$$

$$R_{ij}^s(\Omega) = \int_\lambda R_s(\Omega, \lambda) E_i(\lambda) E_j(\lambda) d\lambda. \tag{13}$$

$R_{ij}^a$ is the projection onto the $j^{th}$ basis function of the spectral power distribution obtained from the reflection of the $i^{th}$ basis function from the ambient reflectance of the surface. $R_{ij}^d$ and $R_{ij}^s(\Omega)$ are analogous terms for the diffuse and specular reflections, respectively.

Writing Equation 10 in matrix form, we obtain

$$\begin{aligned} \begin{pmatrix} \epsilon_1^o \\ \vdots \\ \epsilon_m^o \end{pmatrix} &= \begin{pmatrix} R_{11}^a & \cdots \\ \vdots & \ddots \end{pmatrix} \begin{pmatrix} \epsilon_1^a \\ \vdots \\ \epsilon_m^a \end{pmatrix} + \\ & G_d(\Omega) \begin{pmatrix} R_{11}^d & \cdots \\ \vdots & \ddots \end{pmatrix} \begin{pmatrix} \epsilon_1^s \\ \vdots \\ \epsilon_m^s \end{pmatrix} + \\ & \begin{pmatrix} R_{11}^s(\Omega) & \cdots \\ \vdots & \ddots \end{pmatrix} \begin{pmatrix} \epsilon_1^s \\ \vdots \\ \epsilon_m^s \end{pmatrix} \end{aligned} \tag{14}$$

In vector notation, this equation can be written

$$\vec{\epsilon^o} = R^a \vec{\epsilon^a} + G_d(\Omega) R^d \vec{\epsilon^s} + R^s(\Omega) \vec{\epsilon^s}. \tag{15}$$

This final equation reveals the mathematical formalism behind the linear transform method. The spectral power distributions ($I_a(\lambda), I_s(\lambda)$, and $I_o(\Omega, \lambda)$) are represented by

column vectors of length $m$ containing the transform coefficients ($\vec{\epsilon^a}, \vec{\epsilon^s}$, and $\vec{\epsilon^o}$, respectively). Each component of the surface reflectance ($R_a(\lambda), R_d(\lambda)$, and $R_s(\Omega, \lambda)$) is represented by a single $m \times m$ matrix ($R^a, R^d$, and $R^s(\Omega)$, respectively). The interaction of light with a surface component assumes the form of simple matrix multiplication, converting the coefficients of the incoming light into the coefficients of the outgoing light. This result is a generalization of the point sampling case; with point samples, the surface matrices are diagonal, and the matrix product multiplies respective sample values. Because this technique is linear, it can be included without difficulty in standard renderers.

For the general lighting model case, the specular matrix is a function of the geometry. Because the elements of the surface matrices are obtained through integration over the basis functions, this integration must be performed for each geometry configuration. If, however, one uses a piecewise separable lighting model, the geometry and wavelength dependence separate in the specular term,

$$\vec{\epsilon^o} = R^a \vec{\epsilon^a} + G_d(\Omega) R^d \vec{\epsilon^s} + G_s(\Omega) R^s \vec{\epsilon^s}, \tag{16}$$

and the three surface matrices, $R^a, R^d$, and $R^s$, can be precomputed.

The above discussion addresses only surface reflection, but effects such as transmission and attenuation can be included straightforwardly in this framework. As with the reflectance components, these terms take the form of $m \times m$ matrices that act on the coefficients of the incoming light.

### 2.3 Conversion to *RGB*

The rendering algorithm determines the spectral contributions to a pixel by computing multiple reflection paths from each of the light sources to the viewer. These contributions are transform coefficients, and by linearity they can be combined to provide a final set of coefficients for that pixel, $\epsilon_i^p; i = 1, ..., m$. Equation 1 gives the approximation to the spectral power distribution arriving at the pixel,

$$I_p(\lambda) = \sum_{i=1}^{m} \epsilon_i^p E_i(\lambda). \tag{17}$$

To compute appropriate values for display, one first computes the tristimulus values, $XYZ$, for the pixel by integrating the final spectrum over the three color matching functions [21]

$$\begin{aligned} X = \int \bar{x}(\lambda) I_p(\lambda) d\lambda &= \int \sum_{i=1}^{m} \epsilon_i^p \bar{x}(\lambda) E_i(\lambda) d\lambda \\ &= \sum_{i=1}^{m} T_{xi} \epsilon_i^p \end{aligned} \tag{18}$$

$$\begin{aligned} Y = \int \bar{y}(\lambda) I_p(\lambda) d\lambda &= \int \sum_{i=1}^{m} \epsilon_i^p \bar{y}(\lambda) E_i(\lambda) d\lambda \\ &= \sum_{i=1}^{m} T_{yi} \epsilon_i^p \end{aligned} \tag{19}$$

$$\begin{aligned} Z = \int \bar{z}(\lambda) I_p(\lambda) d\lambda &= \int \sum_{i=1}^{m} \epsilon_i^p \bar{z}(\lambda) E_i(\lambda) d\lambda \\ &= \sum_{i=1}^{m} T_{zi} \epsilon_i^p. \end{aligned} \tag{20}$$

In matrix form, this set of equations can be written

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} T_{x1} & T_{x2} & \cdots & T_{xm} \\ T_{y1} & T_{y2} & \cdots & T_{ym} \\ T_{z1} & T_{z2} & \cdots & T_{zm} \end{pmatrix} \begin{pmatrix} \epsilon_1^p \\ \epsilon_2^p \\ \vdots \\ \epsilon_m^p \end{pmatrix}. \tag{21}$$

With $\vec{x} = (X, Y, Z)^T$, this equation yields

$$\vec{x} = T \vec{\epsilon^p}. \tag{22}$$

The elements $T_{xi}, T_{yi}$, and $T_{zi}$ of the matrix $T$ are coefficients that result from integration of the $i^{th}$ basis function over the three color matching functions. For point sampling, these elements are modified based on the method of numerical integration. For example, common Riemann summation over evenly spaced samples includes the distance between the sample points [17], and Gaussian quadrature has its own unique weights [5].

Assuming that an $RGB$ display monitor is properly gamma-corrected [4], the color values, $\vec{c} = (R, G, B)^T$, of a given pixel are computed from the tristimulus values by applying a 3×3 matrix, $M$, derived from the chromaticities of the phosphors of the monitor [6]

$$\vec{c} = M \vec{x} \tag{23}$$

$$= MT \vec{\epsilon^p} \tag{24}$$

$$= C \vec{\epsilon^p}. \tag{25}$$

Therefore, the $RGB$ values can be obtained directly through a linear transformation of the final coefficient values by a $3 \times m$ matrix $C$. Because this step is linear, it can be applied at any time to the separate contributions to the final pixel values.

## 3 Selection of Basis Functions

It is in the selection of the basis functions that the flexibility of the general transform method is demonstrated. In this section, we describe some factors that determine the effective selection of basis functions, and we present two methods for determining basis functions that are tailored to the spectral power distributions in a scene.

As mentioned in Section 2, the lighting model is a significant influence on the choice of basis functions. If the lighting model is not piecewise separable, the surface matrices must be computed for each geometry configuration, so the most efficient basis functions are most likely point samples. If, however, the lighting model is piecewise separable, we have another consideration. The components of the surface reflectances are represented by $m \times m$ matrices. Therefore, the reflection of light from a surface requires, in general, $m^2$ multiplies. If the basis functions are point samples, though, the surface matrices are diagonal, and the reflection requires only $m$ multiplies. Indeed, only $m$ multiplies are required for any set of non-overlapping basis functions. Consequently, the computational intensiveness of the general transform rises more rapidly than that of point sampling as the number of basis functions increases.

A third consideration when selecting basis functions is the nature of the spectral power distributions in the scene to be rendered. For smoothly varying distributions, point sampling can be quite efficient, but for complicated spectra, a set of general basis functions can be more appropriate. We discuss each of these methods in the following sections.

## 3.1 Point Sampling

Point sampling is typically linked to a numerical integration method used in approximating the tristimulus integrals, Equations 18-20. Gaussian quadrature, which is optimal for integrating polynomials over general weighting functions [5], has been applied to this problem [14] [19] [2]. If the spectral power distributions are well described by lower order polynomials, Gaussian quadrature can provide sufficient accuracy with a small number of sample points; it was shown in [14] that as few as four point samples are adequate for many rendering applications.

Here, we discuss the use of simple Riemann summation for approximating the tristimulus integrals. Rather than being efficient for polynomial functions, Riemann summation is efficient when integrating functions that contain a small number of Fourier coefficients.

### Riemann Summation

Riemann summation is the sum over evenly spaced sample values weighted by the distance between the sample wavelengths [17]. Given $N+2$ evenly spaced sample points $\lambda_0, \lambda_1, ..., \lambda_{N+1}$ separated by a distance $\Delta\lambda = (\lambda_{N+1} - \lambda_0)/(N+1)$ and a spectral power distribution $I(\lambda)$, Riemann summation gives

$$X = \int_\lambda \bar{x}(\lambda)I(\lambda)d\lambda \approx \Delta\lambda \sum_{i=0}^{N+1} \bar{x}(\lambda_i)I(\lambda_i)$$

$$Y = \int_\lambda \bar{y}(\lambda)I(\lambda)d\lambda \approx \Delta\lambda \sum_{i=0}^{N+1} \bar{y}(\lambda_i)I(\lambda_i) \qquad (26)$$

$$Z = \int_\lambda \bar{z}(\lambda)I(\lambda)d\lambda \approx \Delta\lambda \sum_{i=0}^{N+1} \bar{z}(\lambda_i)I(\lambda_i).$$

An appropriate choice of endpoints, $\lambda_0$ and $\lambda_{N+1}$, is the most closely spaced pair of wavelengths that can be chosen such that the color matching functions at these wavelengths can be taken to be zero. We found that $\lambda_0 = 400nm$ and $\lambda_{N+1} = 700nm$ are often reasonable choices; truncation at these limits results in errors significantly smaller than those incurred by undersampling the spectra [17] [18]. Taking $\bar{x}(\lambda_0) = \bar{y}(\lambda_0) = \bar{z}(\lambda_0) = 0$ and $\bar{x}(\lambda_{N+1}) = \bar{y}(\lambda_{N+1}) = \bar{z}(\lambda_{N+1}) = 0$, only the $N$ interior points, $\lambda_1, ..., \lambda_N$, need to be preserved during the rendering process; the basis functions for the spectral power distributions are given by delta functions at these wavelengths.

With the endpoints of the integrands equal to zero, Riemann summation with $N$ points is exact for any linear combination of the first $2N+2$ Fourier functions 1, $sin(2\pi\frac{\lambda-\lambda_0}{\lambda_{N+1}-\lambda_0})$, $cos(2\pi\frac{\lambda-\lambda_0}{\lambda_{N+1}-\lambda_0})$, ..., $sin(2\pi N\frac{\lambda-\lambda_0}{\lambda_{N+1}-\lambda_0})$, $cos(2\pi N\frac{\lambda-\lambda_0}{\lambda_{N+1}-\lambda_0})$, $sin(2\pi(N+1)\frac{\lambda-\lambda_0}{\lambda_{N+1}-\lambda_0})$. Therefore, if the products of the spectral power distributions with each of the color matching functions are well described by a small number of Fourier coefficients, Riemann summation provides an efficient method for integration. For the set of spectral power distributions obtained from the Macbeth Color Checker [13] under CIE Standard Illuminant C [21], Riemann summation with four point samples at 460nm, 520nm, 580nm, and 640nm results in an average error of less than 5% in the tristimulus values. Rendering with these four sampling points is often sufficient; if it is not, selecting five, six, or more evenly spaced samples is straightforward.

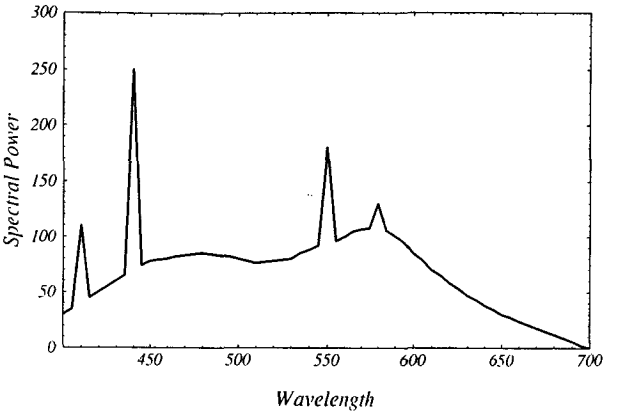



Figure 1: Spectral power distribution of a fluorescent light

## 3.2 General Basis Functions

For scenes with complicated spectral power distributions or surface properties, naive point sampling is insufficient. One notable example is the spectral power distribution of fluorescent light, which is ubiquitous in indoor scenes. Fluorescent light, an example of which is shown in Figure 1 [21], is characterized by narrow emission lines at several wavelengths, a factor leading to aliasing with a small number of point samples. For these complicated cases, one would like to be able to tailor the basis functions to the complex spectra. One attempt in this direction is the use of abutting box functions over the range of wavelengths whose widths are chosen based on the spectra within the scene [7] [6]. Another technique for dealing with these scenes is hand-selecting the basis functions using knowledge of the spectra in the scene. For example, for fluorescent lights, one could ensure that point samples were positioned at the emission lines.

Here, we present an alternative method for the selection of basis functions, gaining insight from studies done on the construction of linear models of surface reflectances and spectral power distributions [3] [8] [9] [15] [12]. Most of these studies have stressed the use of characteristic vector analysis or principal component analysis to characterize lights and surfaces. This technique can be applied to the rendering problem to provide an automated method for selecting an efficient set of basis functions.

### Characteristic Vector Analysis

Given a set of spectral power distributions, characteristic vector analysis computes an ordered set of functions such that the first $m$ functions are the "best" $m$ functions for approximating the distributions. Here, "best" is measured in terms of least squared error between the actual and the approximating spectra. Formally, for the approximation of

the spectral power distribution

$$I(\lambda) \approx \sum_{i=1}^{m} \epsilon_i E_i(\lambda), \tag{27}$$

the basis functions, $E_i(\lambda)$, are computed such that the sum of the approximation error over all of the lights in the set is minimized

$$Err = \sum_{I} \int [I(\lambda) - \sum_{i=1}^{m} \epsilon_i E_i(\lambda)]^2 d\lambda. \tag{28}$$

In practice, this set can be determined by placing the representative spectra in the columns of a matrix and performing a singular value decomposition [10] [11].

The task is then to find a representative set of spectra on which to perform the analysis. For the rendering problem, the basis functions should describe any spectral power distribution within the scene. The distributions contain contributions from the light sources themselves, from once-reflected light, and from multiply-reflected light. Therefore, an appropriate set of spectra is that set derived from possible interreflections within the scene. Given the spectral power distributions of the lights and the components of the surface reflectances in a scene, one can construct a tree of possible interreflection spectra (disregarding any geometry). The lights themselves would be included, and any number of reflections and interreflections could be included. The basis functions computed from a characteristic vector analysis of this set would then approximate these spectral power distributions.

If the number of spectral power distributions to fit is too large, this technique can become inefficient; the cost of computing the basis functions may exceed the savings in rendering time. Also, this method is inapplicable if one does not know *a priori* the spectral character of the surfaces and lights in the scene. However, for many scenes, this technique can readily be applied.

## 3.3 Examples

To demonstrate the use of characteristic vector analysis in selecting basis functions, we present two related examples. Both examples use the fluorescent light in Figure 1 to show the ability of this technique to handle complex spectra. In the first example, we determine the efficiency in computing the tristimulus values of a set of spectral power distributions, and in the second, we render a simple scene.

### Tristimulus Values of Test Spectra

We select as sample spectra the twenty-four squares of the Macbeth Color Checker under the fluorescent light. A set of basis functions can be computed by performing a characteristic vector analysis on the set of twenty-five spectral power distributions given by the light itself and the light reflected from the twenty-four samples. Figure 2 shows the first three basis functions for this set; as can be seen, characteristic vector analysis preserves the narrow peaks that are found in the spectral power distribution of the light source.

From these basis functions, we compute the transform coefficients of the fluorescent light with Equation 2. Assuming only diffuse reflection and ignoring geometry, we use Equation 12 to compute a single matrix for each of the twenty-four

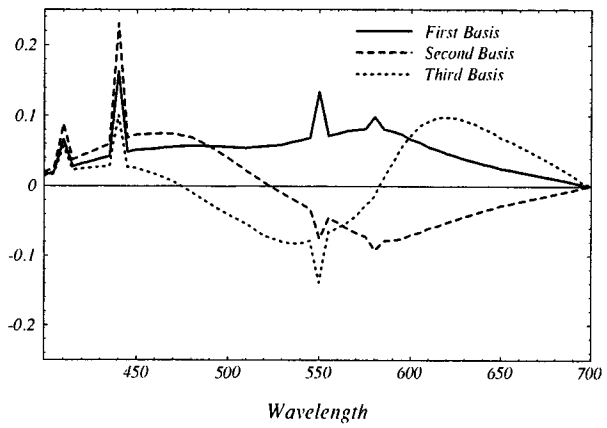



Figure 2: First three basis functions computed with characteristic vector analysis for fluorescent light reflected from the twenty-four squares of the Macbeth Color Checker.

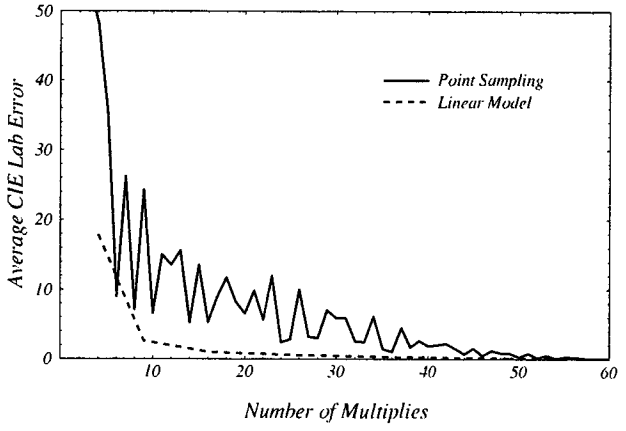



Figure 3: Average CIE Lab Error for set of spectra as a function of the number of multiplies per reflection for evenly spaced point samples and for the general linear transform computed with characteristic vector analysis.

surfaces in the color checker. The product of the vector of coefficients with each of these matrices gives column vectors containing the coefficients of the reflected light. From these vectors, we compute the linear model approximation to the tristimulus values of each of the twenty-four patches with Equations 18-20. The average CIE Lab error in units of $\Delta E$ [21] can then be calculated as a function of the number of basis functions. For reference, we also compute this error as a function of the number of evenly spaced point samples for Riemann summation. To compare the two methods in terms of their computational intensiveness, we plot in Figure 3 the errors as a function of the number of multiplies per reflection.

The general linear model is significantly more efficient than point sampling; the latter shows severe oscillations from the sampling error in computing the narrow peaks in the fluorescent light. Clearly, the point sampling method should (and would) be amended for the fluorescent light case. The most natural method is to ensure point samples lie on the narrow peaks and are weighted appropriately during the integration. This is tantamount to hand-selecting a general

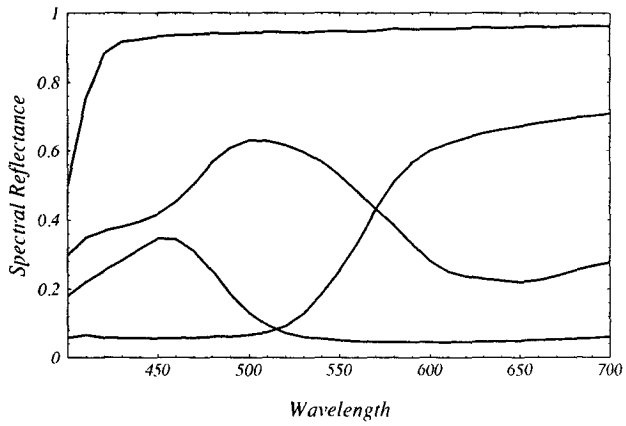



Figure 4: Four surface reflectances from the Macbeth Color Checker used in the example image.

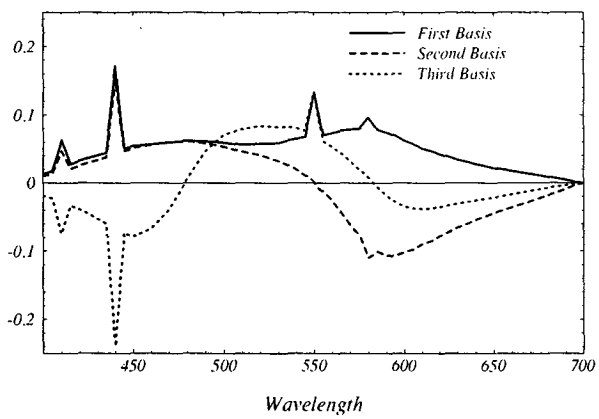



Figure 5: First three basis functions of the general linear model computed with characteristic vector analysis for the example image.

linear model. Characteristic vector analysis is attractive because it matches most anomalies in the spectra without the user being required to address each one distinctly.

### Image Generation

We now apply characteristic vector analysis to select basis functions for ray tracing of a simple scene under fluorescent light. The four distinct surface reflectances in the scene are taken from the Macbeth Color Checker and are shown in Figure 4. To compute the basis functions, we perform a characteristic vector analysis on the set of spectra consisting of the light source itself, all single reflections, and all second interreflections from the four surface samples; the first three basis functions are shown in Figure 5. These functions are used to compute the column vector of the light source and the ambient, diffuse, and specular reflectance matrices for each of the surfaces in the scene.

Figure 6 shows the resultant images for four different numbers of basis functions. The top left image in the figure displays the full resolution rendering of the scene computed at one nanometer intervals. The two columns display the general linear model and evenly spaced point sampling for the same number of multiplies per reflection. The left column shows the general model with 2, 3, 4, and 5 basis functions from top to bottom, and the right column shows 4, 9, 16, and 25 evenly spaced point samples from top to bottom. The linear model based on characteristic vector analysis is superior for all images; with just three basis functions, it is virtually identical to the full resolution image.

## 5 Conclusions

We have presented a general description of the use of linear transform methods in synthetic image generation. This formalism requires that all spectral power distributions be described with respect to a set of orthonormal basis functions. The spectral power distributions are represented by column vectors, and the surfaces are described by matrices. Reflection during the rendering procedure takes the form of matrix multiplication. Because this process is linear, it allows for easy implementation. In addition, this framework guides the choice of basis functions for efficient rendering.

We have discussed two possibilities for the selection of the basis functions, Riemann summation for efficient point sampling and characteristic vector analysis of a representative set of spectra in the scene. Point sampling based on Riemann summation is effective when the spectral power distributions in a scene are well described with low-order Fourier components. The method based on characteristic vector analysis is of comparable efficiency to point sampling techniques when the scenes contain smoothly varying spectra, and it can be significantly more efficient for scenes with complex spectra. We demonstrated this by rendering a scene illuminated by fluorescent light.

A promising direction of future work is the investigation of basis functions that make the rendering procedure more efficient; the techniques in [12] are potentially useful to this end. In addition, we have focussed in this paper on minimizing the cost of full spectral rendering, but the flexibility of the general method might be useful for other issues in computer graphics, such as texturing, that deal with spectral information during rendering.

## Acknowledgements

This material is based upon work supported under a National Science Foundation Graduate Fellowship and partially supported by the National Science Foundation under Grant NSF ECS 88-15815. The author would like to thank Lambertus Hesselink, Marc Levoy, and Paul Ning for helpful discussions. He especially would like to thank Brian Wandell for helpful discussions and for providing the spectral data used in this work.

## References

[1] Borges, Carlos. Trichromatic Approximation for Computer Graphics Illumination Models. Proceedings of SIGGRAPH '91 (Las Vegas, Nevada, July 28-August 2, 1991). In *Computer Graphics 25*,4 (July 1991),101-104.

[2] Borges, Carlos. *Numerical Methods for Illumination Models in Realistic Image Synthesis.* PhD dissertation, University of California, Davis, 1990.

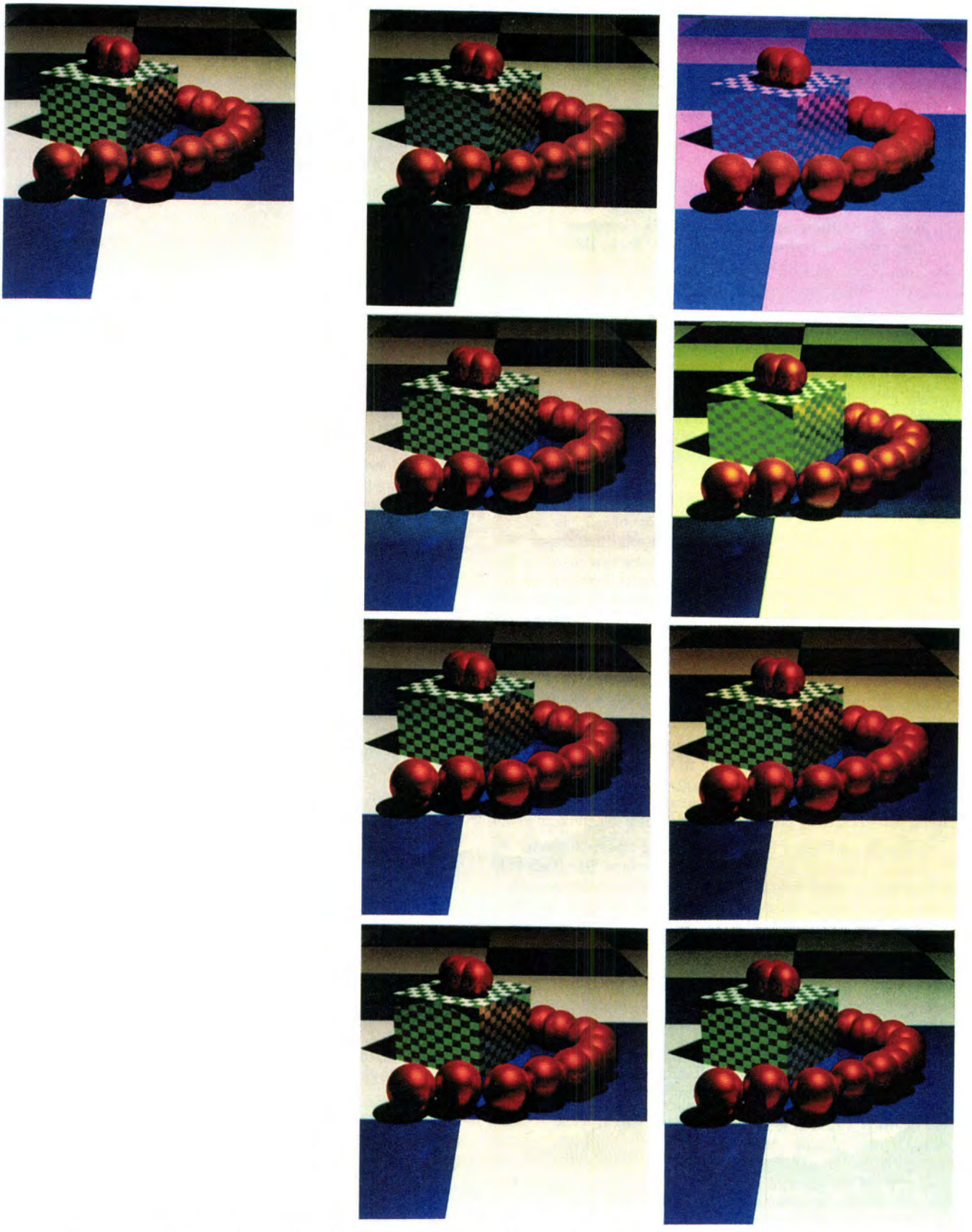


Figure 6: Comparison of general linear model with evenly spaced point sampling. The top left image is a full resolution image computed at one nanometer steps; the left image in each row is the general model with 2, 3, 4, and 5 basis functions from top to bottom; the right image is evenly spaced point sampling with 4, 9, 16, and 25 samples.

[3] Cohen, Jozef. Dependency of the Spectral Reflectance Curves of the Munsell Color Chips. *Psychon. Sci. 1* (1964), 369-370.

[4] Cowan, William. An Inexpensive Scheme for Calibration of a Color Monitor in Terms of CIE Standard Coordinates. Proceedings of SIGGRAPH '83 (Detroit, Michigan, July 25-29, 1983). In *Computer Graphics 17*,3 (July 1983), 315-321.

[5] Davis, P. and Rabinowitz, P. *Methods of Numerical Integration.* Academic Press, New York, 1975.

[6] Hall, Roy. *Illumination and Color in Computer Generated Imagery.* Springer-Verlag, New York, 1989.

[7] Hall, Roy and Greenberg, Donald. A Testbed for Realistic Image Synthesis. *IEEE Computer Graphics and Applications 3* (1983), 10-20.

[8] Judd, Deane, MacAdam, David, and Wyszecki, Gunter. Spectral Distribution of Typical Daylight as a Function of Correlated Color Temperature. *J. Opt. Soc. Am. 54*,8 (1964), 1031-1040.

[9] Maloney, Laurence. Evaluation of linear models of surface spectral reflectance with small numbers of parameters. *J. Opt. Soc. Am. A 3*,10 (1986), 1673-1683.

[10] Maloney, Laurence. *Computational Approaches to Color Constancy.* PhD dissertation, Stanford University, 1985.

[11] Mardia, K., Kent, J., and Bibby, J. *Multivariate Analysis.* Academic, London, 1979.

[12] Marimont, David and Wandell, Brian. Linear models of surface and illuminant spectra. *J. Opt. Soc. Am. A 9*,11 (1992), 1905-1913.

[13] McCamy, C., Marcus, H., and Davidson, J. A Color Rendition Chart. *J. Appl. Photographic Engrg. 11*,3 (1976), 95-99.

[14] Meyer, Gary. Wavelength Selection for Synthetic Image Generation. *Computer Vision, Graphics, and Image Processing 41* (1988), 57-79.

[15] Parkkinen, J., Hallikainen, J., and Jaaskelainen, T. Characteristic Spectra of Munsell Colors. *J. Opt. Soc. Am. A 6*,2 (1989), 318-322.

[16] Raso, Maria, and Fournier, Alain. A Piecewise Polynomial Approach to Shading Using Spectral Distributions. Proceedings of Graphics Interface '91. (Calgary, Alberta, June 3-7, 1991), 40-46.

[17] Smith, Brent, Spiekermann, Charles, and Sember, Robert. Numerical Methods for Colorimetric Calculations: A Comparison of Integration Methods. *COLOR Research and Application 17*,6 (1992), 384-393.

[18] Smith, Brent, Spiekermann, Charles, and Sember, Robert. Numerical Methods for Colorimetric Calculations: Sampling Density Requirements. *COLOR Research and Application 17*,6 (1992), 394-401.

[19] Wallis, Robert. Fast computation of tristimulus values by use of Gaussian quadrature. *J. Opt. Soc. Am. 65*,1 (1975), 91-94.

[20] Wandell, Brian. The Synthesis and Analysis of Color Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-9*,1 (1987), 2-13.

[21] Wyszecki, Gunter and Stiles, W.S. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* John Wiley and Sons, 1982.

# Combining Hierarchical Radiosity and Discontinuity Meshing

*Dani Lischinski* *Filippo Tampieri* *Donald P. Greenberg*

Program of Computer Graphics
Cornell University
Ithaca, NY 14853

## ABSTRACT

We introduce a new approach for the computation of view-independent solutions to the diffuse global illumination problem in polyhedral environments. The approach combines ideas from hierarchical radiosity and discontinuity meshing to yield solutions that are accurate both numerically and visually. First, we describe a modified hierarchical radiosity algorithm that uses a discontinuity-driven subdivision strategy to achieve better numerical accuracy and faster convergence. Second, we present a new algorithm based on discontinuity meshing that uses the hierarchical solution to reconstruct an object-space approximation to the radiance function that is visually accurate. Our results show significant improvements over both hierarchical radiosity and discontinuity meshing algorithms.

**CR Categories and Subject Descriptors:** I.3.3—[**Computer Graphics**]: Picture/Image Generation; I.3.7—[**Computer Graphics**]: Three-Dimensional Graphics and Realism.

**Additional Key Words and Phrases:** diffuse reflector, discontinuity meshing, global illumination, hierarchical radiosity, Mach bands, photorealism, quadratic interpolation, radiance function, radiosity, reconstruction, shadows, view-independence.

## 1 INTRODUCTION

Computing solutions to the global illumination problem is an essential part of photorealistic image synthesis. In this paper, we are interested in computing *view-independent* (or *object-space*) solutions for global illumination. Such solutions provide an approximation to the radiance function across each surface in the environment. Once a solution is computed, images from any viewpoint can be rendered with a relatively small additional effort. These methods are particularly attractive for applications such as architectural design, interior design, lighting design, illumination engineering, and virtual reality, in which the need for multiple views or walk-throughs of static environments arises.

So far, most view-independent methods have been derived from the radiosity method that was originally developed to solve radiative heat transfer problems [23]. Computer graphics researchers adopted this method to compute the global illumination of diffuse polyhedral environments [10, 7, 19]. Radiosity has been extended and improved dramatically since, but there is still much to be done before the method can become a useful tool for its intended users.

The goal of our research is to develop an efficient radiosity system that satisfies the following requirements:

**Objective (numerical) accuracy:** Solutions produced by the system should converge rapidly to the exact solution. This requirement may seem obvious, however, in the computer graphics community results of simulations are too often judged solely by their visual appearance.

**Subjective (visual) accuracy:** While visual appearance should not be used to judge the objective accuracy of the simulation, it is still very important, since the image is the final product. Clearly, accurate visual appearance can be achieved through numerically accurate simulation (if the underlying model is physically accurate.) Unfortunately, experience has shown that the human visual system is extremely sensitive to small perceptual errors that are difficult to quantify. The simulated environments can be very complex and, therefore, the computation of ultra-accurate solutions is generally impractical. Thus, we must have means of producing visually acceptable images even from coarse solutions.

**Ease of control:** (i) The system should be controllable by users who are not necessarily familiar with its inner workings. Therefore, the control parameters should be intuitive and small in number. (ii) In many cases (such as early design stages) the user is interested in a quick solution, even if not exceedingly accurate. At other times, one might be willing to wait overnight for a reliable solution. Therefore, the system should provide the user with the option to trade speed for accuracy.

Most radiosity systems do not satisfy any of these requirements. There are no error bounds on the solutions, because approximations are often used without justifications regarding their impact on the accuracy of the results. The resulting images typically exhibit many visual artifacts such as Mach bands, light and shadow leaks, jagged shadow boundaries, and missing shadows. Radiosity systems are seldom user-friendly and require massive user intervention: typically, a time consuming trial-and-error process is required to produce an image that looks right. Baum *et al.* [1] and Haines [12] provide good discussions of the various pitfalls of radiosity.

In this paper we present a new radiosity method, which comes closer to satisfying our goals. The new method combines two recently developed approaches: hierarchical radiosity [14] and discontinuity meshing [15, 18]. First, we present an improved hierarchical radiosity algorithm that uses a discontinuity-driven subdivision strategy to achieve better numerical accuracy and faster convergence. Second, we describe a new algorithm based on discontinuity meshing that uses the hierarchical solution to reconstruct a visually accurate approximation to the radiance function. Thus, results of



 ACM-0-89791-601-8/93/008/0199 $01.50

high visual quality can be obtained even from coarse global illumination simulations. Previous attempts to improve the visual quality of radiosity solutions were described by Nishita and Nakamae [19], Kok and Jansen [17], Chen *et al.* [4], and Reichert [20]. In all of these cases, however, the improvement takes place in image space, after the view and the resolution have been specified. Our method, instead, operates entirely in object space, and the improved solution is view-independent.

## 2 HIERARCHICAL RADIOSITY

The traditional radiosity approach [10, 7] discretizes the environment into $n$ elements and solves a linear system of $n$ equations, where the radiosities of the elements are the unknowns. The most serious drawback of this approach is the need to compute the $O(n^2)$ coefficients of the linear system, corresponding to the interactions (transfers of light energy) between pairs of elements. In addition to the overwhelming computational complexity, most of these computations are performed to unnecessarily high accuracy, while some are not sufficiently accurate.

Hierarchical radiosity (HR) [14] overcomes these problems by decomposing the matrix of interactions into $O(n)$ blocks, for a given accuracy. These blocks correspond to interactions of roughly equal magnitude, and the same computational effort is required for computing each block. HR operates by constructing a hierarchical subdivision of each input surface. Each node in the hierarchy represents some area on the surface. Two nodes are linked together if the interaction between their corresponding areas can be computed within the required accuracy; otherwise, the algorithm attempts to link their children with each other. Each link corresponds to a block in the interaction matrix.

HR has several important advantages: it is fast, the errors in its approximations are bounded, and it is controlled by only two parameters: the error tolerance and the minimum node area. The smaller the values of these parameters, the more accurate (and expensive) the solution becomes. Thus, HR satisfies our goals of objective accuracy and ease of control.

However, the HR algorithm still suffers from shadow leaks and jagged shadow boundaries. This occurs because surfaces are subdivided regularly, not taking into account the geometry of the shadows. HR uses point sampling to classify the inter-visibility between two surfaces, so it is prone to missing small shadows altogether. Of course, as the user-specified tolerance becomes smaller, the solution becomes more accurate, and the visual artifacts decrease. Nevertheless, images of high visual quality can require solutions of prohibitively high accuracy.

The number of links created by HR is $O(n + m^2)$ where $n$ is the final number of nodes and $m$ is the number of input surfaces. As the complexity of the environment increases, the $m^2$ term eventually becomes dominant, drastically reducing the efficiency of the algorithm. As pointed out by Smits *et al.* [22], this problem could be solved by grouping the input surfaces into higher level clusters. This is an interesting research topic by itself, and it will not be pursued in this paper.

## 3 DISCONTINUITY MESHING

Radiosity methods typically attempt to approximate the radiance function with constant elements and use linear interpolation to display the result. The actual radiance function, however, is neither piecewise constant nor piecewise linear. It is usually smooth, except along certain curves across which discontinuities in value or in derivatives of various order may occur. Discontinuities in radiance functions are discussed in detail elsewhere [16, 15, 18]; what follows is a brief summary of the various types of discontinuity and their causes.

The most significant discontinuities are discontinuities in the radiance function itself (denoted $D^0$). They occur along curves of contact or intersection between surfaces. Discontinuities in the first and the second derivatives ($D^1$ and $D^2$, respectively) occur along curves of intersection between surfaces in the environment and *critical surfaces* corresponding to qualitative changes in visibility, or *visual events*. Visual events in polyhedral environments can be classified into two types [9]: EV events defined by the interaction of an edge and a vertex, where the critical surface is a planar wedge; and EEE events defined by the interaction of three edges, where the critical surface is a part of a quadric. Discontinuities of higher than second order are also possible [16].

Discontinuities are very important both numerically and visually: all the boundaries separating unoccluded, penumbra, and umbra regions correspond to various discontinuities. When a discontinuity curve crosses a mesh element, the approximation to the radiance function over that element becomes less accurate. The resulting errors usually correspond to the most visually distracting artifacts in radiosity images. The traditional radiosity approach uses adaptive subdivision [8] to reduce these errors, however there are several problems with this approach. First, the user must specify an initial mesh that is sufficiently dense, or features will be lost. Second, the shape of the mesh is determined by the geometry of the surface being meshed, and the discontinuities are not resolved exactly. As a result, many small elements are created as the method attempts to converge to shadow boundaries. Furthermore, although the resulting solution may be of adequate visual quality for some views, artifacts may become visible as the view changes (e.g., when we zoom in on a surface.)

Discontinuity meshing (DM) algorithms compute the location of certain discontinuities and represent them explicitly, as boundaries, in the mesh. This leads to solutions which are both numerically and visually more accurate. Another advantage is that higher order elements can be used much more effectively in conjunction with discontinuity meshes [16]. Several algorithms have been described that use the idea of discontinuity meshing to various extents [1, 3, 6, 15].

Recently, a progressive radiosity DM algorithm was described by the authors [18]. The meshing in this algorithm is automatic. Using analytical visibility and form factor computations followed by quadratic interpolation it has produced radiosity solutions of impressive visual accuracy. This algorithm was also shown to be numerically accurate [24].

However, this method is too expensive for computing converged solutions of complex environment and only offers limited user control in trading off speed for accuracy. The main reason for this is that all energy transfers are computed very accurately, regardless of their magnitude.

## 4 A COMBINED APPROACH

Hierarchical radiosity and discontinuity meshing seem to complement each other in their strengths and weaknesses: HR is fast, but the visual appearance of the results can be disappointing; DM, on the other hand, has produced visually accurate results, but so far it has been too expensive for simulation of complex environments. This observation motivated us to look for ways of merging the two methods. Our investigation resulted in the following two-pass approach:

**The global pass** uses a modified HR algorithm to compute a radiosity solution within a prespecified tolerance. Instead of regular quadtree subdivision, the modified algorithm subdivides surfaces along discontinuity segments. This improves the numerical accuracy and results in faster convergence.

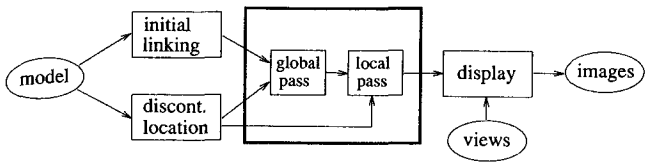



Figure 1: The structure of the new radiosity system

**The local pass** uses DM and quadratic interpolation to refine the approximation to the radiance function locally on each surface in the environment. Thus, the solution computed by the global pass is transformed into a more visually accurate form.

When the computation is arranged in this way the simulation becomes more efficient. The global pass need not be concerned with visual accuracy. This eliminates the need to maintain a topologically connected mesh, to prevent T-vertices, or to use extremely fine subdivision around shadow boundaries, since this has little effect on the global distribution of light in the environment. The local pass, on the other hand, can create as many elements as necessary for a high quality reconstruction of the radiance function, without overburdening the global illumination simulation. As a result, it is possible to produce images of high visual accuracy even from quick simulations.

To test our approach we have implemented a new radiosity system whose overall structure is shown in Figure 1. The global and the local passes are discussed in detail in the next two sections. In the rest of this section we briefly describe the remaining parts.

The initial linking stage creates for each input polygon a list of links to all the polygons that are visible from it. For each link it is determined whether the two polygons are completely or partially visible to each other. This creates a starting point for the global pass, which proceeds to refine these links as needed. We test visibility between two polygons using a combination of shaft-culling [13] and the ray-tracing algorithm that Hanrahan *et al.* [14] used.

The discontinuity location stage computes the location of all the $D^0$ discontinuities, since these are typically responsible for the most severe errors (both numerically and visually.) In most environments the direct illumination by primary light sources is responsible for the most perceptible illumination details. Therefore, all of the $D^1$ and $D^2$ discontinuities caused by EV events involving the primary light sources are computed as well. The computed discontinuities are henceforth collectively referred to as *primary* discontinuities.

EEE events are more difficult to handle because their corresponding critical surfaces are curved, rather than planar. However, the resulting discontinuities always lie within penumbra regions, and never define the outer boundaries of a shadow. For these reasons, we excluded EEE events from our current implementation.

We described the discontinuity location algorithm in a previous paper [18]. Tampieri [24] provides a more detailed description of this algorithm. Heckbert [15] and Teller [25] describe alternative algorithms for locating discontinuities. Teller's algorithm is the only one capable of handling EEE events.

## 5 THE GLOBAL PASS

In order to understand how the accuracy of HR can be improved, we must examine its sources of error. Consider two nodes $s$ and $r$ linked together by the HR algorithm. Let $B_{rs}(x)$ denote the actual radiosity due to node $s$ at point $x$ on node $r$. The algorithm approximates this radiosity by a constant function

$$B_{rs}(x) \sim \hat{B}_{rs} = \rho_r B_s F_{rs} V_{rs}$$

where $\rho_r$ is the reflectivity of node $r$; $B_s$ is the average radiosity of node $s$; $F_{rs}$ is the form factor from $r$ to $s$; and $V_{rs}$ is the inter-visibility factor between $r$ and $s$ (the visible fraction of the area of $s$, averaged over $r$).

We are interested in bounding the error between the computed and the actual radiosities

$$E_{rs} = \sup_{x \in r} \left| B_{rs}(x) - \hat{B}_{rs} \right| \tag{1}$$

To that end, we define the following upper and lower bounds:

$$\begin{aligned} B_s^{\min} &= \inf_{x \in s} B_s(x) & B_s^{\max} &= \sup_{x \in s} B_s(x) \\ F_{rs}^{\min} &= \inf_{x \in r} F_{xs} & F_{rs}^{\max} &= \sup_{x \in r} F_{xs} \\ V_{rs}^{\min} &= \inf_{x \in r} V_{xs} & V_{rs}^{\max} &= \sup_{x \in r} V_{xs} \end{aligned}$$

where $B_s(x)$ is the radiosity at point $x$ on $s$; $F_{xs}$ is the form factor from point $x$ to $s$; and $V_{xs}$ is the fraction of the area of $s$ visible from $x$. Clearly, both $B_{rs}(x)$ and $\hat{B}_{rs}$ lie in the interval

$$\left[ \rho_r B_s^{\min} F_{rs}^{\min} V_{rs}^{\min}, \rho_r B_s^{\max} F_{rs}^{\max} V_{rs}^{\max} \right]$$

Therefore, the error $E_{rs}$ is bounded by the width of the interval

$$E_{rs} \leq \rho_r \left( B_s^{\max} F_{rs}^{\max} V_{rs}^{\max} - B_s^{\min} F_{rs}^{\min} V_{rs}^{\min} \right) \tag{2}$$

Three main factors affect the magnitude of the error:

1. the variation of the radiosity on the source node $s$
2. the variation of the form factor across the receiver node $r$
3. the variation in the visibility of the source from the receiver

Therefore, if we find the potential error in the transfer of light energy from $s$ to $r$ too large, we can try to reduce the error by reducing any of these factors. For instance, subdividing the receiving node will reduce the variation of the form factor. Subdividing the source will reduce the variation of the radiosity on the source. Subdividing either of the two may reduce the variation in the visibility.

Unfortunately, errors due to visibility are more difficult to handle than errors of the other two types. If the two nodes are completely visible to each other, the error usually decreases rapidly as the nodes are subdivided. When the two nodes are completely occluded from each other no light energy transfer occurs, and the error is zero. Partial visibility, on the other hand, often results in very fine subdivisions, primarily because of loose bounds on the variation in visibility between two finite areas. In HR, visibility is estimated by casting a number of rays between the two nodes. Thus, if partial visibility is detected, all we know is that the actual visibility is in the interval $(0, 1)$.

Clearly, it would be to our advantage to use a subdivision strategy that would result in as many totally visible or totally occluded pairs, as quickly as possible. Since discontinuity lines on the receiver correspond to abrupt changes in the visibility of the sources [16, 18], subdividing the receiver along these lines should quickly resolve partial occlusion.

We have modified the HR algorithm to perform discontinuity-driven subdivision instead of regular subdivision. There are two main changes in the data structures used by the new algorithm: first, we store with each node a list of all the discontinuity segments on the corresponding polygon; second, we use a 2D binary space partitioning (BSP) tree [3] instead of a quadtree to represent the hierarchical subdivision of each initial polygon, since BSP trees allow for subdivision of polygons along arbitrarily oriented lines. Pseudocode for subdividing a node is given in Figure 2.

When a node is subdivided we choose one of its discontinuity segments and split the node using the corresponding line equation. The segment is chosen such that the split is as balanced as possible.

```
Boolean Subdivide(node)
if not IsLeaf(node) then
    return TRUE
end if
if node.area < minNodeArea then
    return FALSE
end if
if node.DSegments ≠ NIL then
    DSegment s ← ChooseBestSegment(node)
    (left, right) ← SplitNode(node, s)
    (leftList, rightList) ← SplitSegmentList(node, s)
else
    (left, right) ← SplitEqual(node)
    (leftList, rightList) ← (NIL, NIL)
end if
node.left ← CreateNode(left, leftList)
node.right ← CreateNode(right, rightList)
return TRUE
```

Figure 2: Pseudocode for the *Subdivide* routine

Priority is given to $D^0$ discontinuities over higher order ones, since the former typically bound areas totally occluded from the rest of the environment. The subdivision is completed by splitting the list of segments into two new lists, one for each child. If no segments are stored with the node, we split the node by connecting the midpoint of the longest edge to a vertex or another midpoint chosen so that the resulting children have roughly equal areas.

### 5.1 Results

Figure 4 demonstrates the improved hierarchical algorithm using a simple environment illuminated by two small triangular light sources. A 3D view of the environment is shown in image a1. The radiance function on the floor polygon is shown in image a2. Image a3 shows the discontinuity segments on the floor. $D^0$ discontinuities are drawn in red; $D^1$ and $D^2$ discontinuities in yellow. In rows b and c, we compare the subdivision produced by the discontinuity-driven algorithm to the one produced by regular subdivision. The level of subdivision shown increases from left to right: the leftmost pair shows the subdivision at level 2, then level 4, 6, and 8.

The new algorithm is much quicker to correctly separate regions corresponding to complete occlusion, partial visibility, and complete visibility. Already at subdivision level 4 (image b2), most of the nodes can be classified as either totally visible or totally occluded with respect to each of the light sources. For these areas there are no more visibility errors. At subdivision level 6 (image b3) all of the discontinuities have been used, and the partially visible nodes are now confined exactly to the areas of penumbra.

In order to compare the rates of convergence of the two strategies we computed a set of approximations to the direct illumination on the floor using a successively larger number of elements. Figure 3 shows the RMS and the maximum absolute errors versus the number of elements for the two strategies. These errors were computed with respect to an analytical solution at the vertices of a 400 by 400 grid on the floor. All the values were scaled to set the maximum brightness on the floor to 1.

Our algorithm converges faster in both error metrics. Note that the convergence of the regular subdivision is particularly poor in the maximum absolute error metric. The reason is that there are $D^0$ segments on the floor that are not aligned with the subdivision axes. Thus, there are always elements that are partially covered by the pyramid while the remaining part is brightly illuminated by the

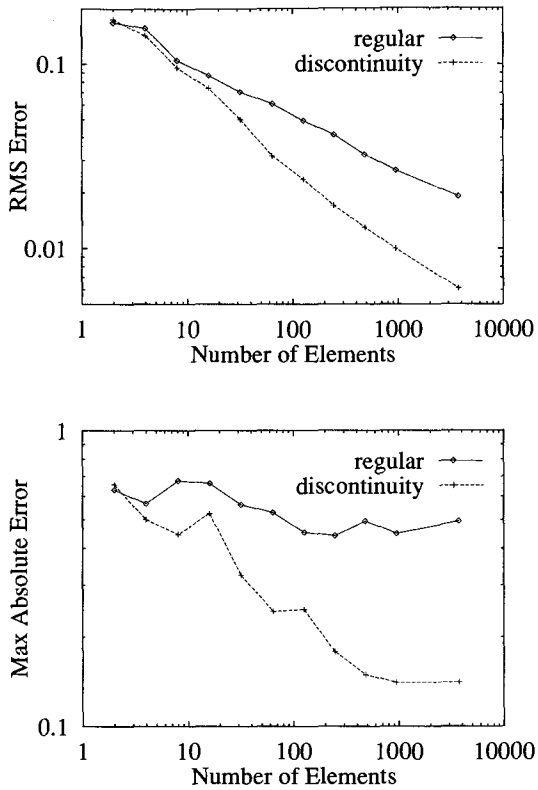



Figure 3: A comparison of errors between the two subdivision strategies using log-log plots

light sources. The algorithm assigns a single constant value to each such element, and this results in a large error there. Our algorithm, on the other hand, resolves $D^0$ discontinuities and therefore does not suffer from this problem.

In the RMS error metric regular subdivision does converge, because the elements that contain the errors become progressively smaller, and this is accounted for by the metric; however, the convergence is slower.

## 6 THE LOCAL PASS

The global pass results in a hierarchical solution that is essentially a piecewise constant approximation to the radiance function on each polygon in the environment. Often, this approximation is quite coarse. Now our goal is to convert this solution into a form more suited for producing visually accurate images. To that end, we need to locally refine the radiance approximation on each polygon.

Our experience with discontinuity meshing [18] has shown that reproducing the discontinuities in the radiance function, while maintaining a smooth approximation elsewhere is key to achieving visual accuracy, especially when multiple views of the same solution are to be rendered. Therefore, we construct a discontinuity mesh containing the precomputed primary discontinuities for each polygon. Mesh nodes are assigned radiance values using the hierarchical solution. This mesh is then used for the shaded display of the environment. Thus, the local pass essentially performs an additional light gathering operation over the environment. However, instead of gathering to the nodes in the hierarchy, we gather to the elements of the discontinuity mesh.

The discontinuity mesh is constructed using constrained Delaunay triangulation (CDT) [5]. The Delaunay triangulation (DT) of a point set maximizes the minimum angle over all possible triangula-

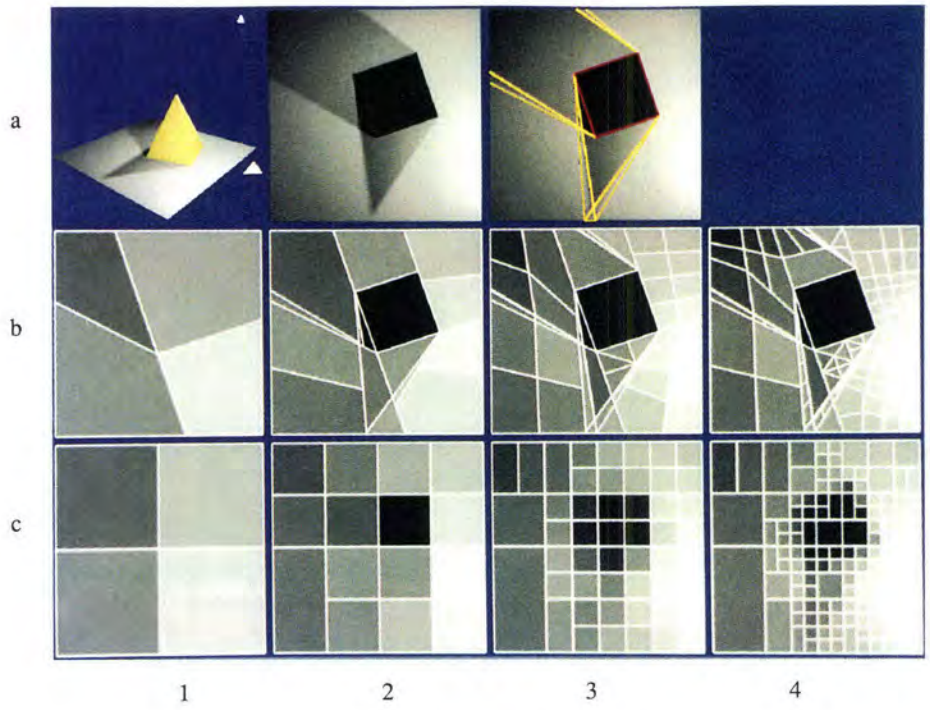



Figure 4: Discontinuity-driven vs. regular subdivision

tions of that set and has a number of other desirable properties [2]. These properties are important because they result in well-shaped elements that yield more accurate approximations and reduce visual artifacts during display [1]. CDT takes as input a point set and a set of edges connecting some of the points, and creates a triangulation of the points that is constrained to include all the input edges. CDT preserves the properties of DT over all the constrained triangulations. We have implemented an incremental CDT algorithm that is a simple extension of the incremental DT algorithm described by Guibas and Stolfi [11]. An alternative easy-to-implement algorithm is described in the excellent survey by Bern and Eppstein [2].

For each input polygon we provide the CDT routine with all of its boundary edges and discontinuity segments. The corners of all the leaf nodes in the corresponding hierarchy are given as well. Thus, the resulting mesh is dense enough to adequately sample the solution computed by the global pass. As a result of the properties of the CDT, most of the triangles are well shaped unless the hierarchy is very coarse.

The radiance across each triangle is approximated using a standard quadratic element commonly used in finite element methods [27]. Six radiance values are computed for each element: three at the vertices, and three at the edge midpoints. Except for $D^0$ edges, these values are shared between adjacent faces (our CDT algorithm constructs a topological data structure suitable for such information sharing [11].) The six values are then interpolated by a quadratic bivariate polynomial. This scheme yields a $C^0$ piecewise quadratic interpolant to the radiance on each polygon. This interpolant was found to provide approximations that look smoother and are less prone to Mach bands than the traditional piecewise linear interpolation [18]. Salesin *et al.* [21] describe a piecewise cubic interpolant that can be used instead, if $C^1$ interpolation is desired.

To obtain a radiance value at a point $x$ we use the information available to us from the hierarchical solution. Below we describe four different methods that we have experimented with. Pseudocode for the last three methods is given in Figure 5.

**Method A.** The simplest approach is to use the radiance value stored in the hierarchy leaf that contains $x$. If $x$ is on the boundary between two or more leaves, their values are averaged to yield the radiance at $x$. This method has no overhead other than locating the containing leaves.

The accuracy of the resulting value depends on the accuracy of the global pass solution. Consider the path from the root of the hierarchy to the leaf containing the point $x$. Every node along this pass has zero or more links to other nodes, representing areas on primary or secondary sources that illuminate $x$. The error at $x$ due to one such link between a containing node $r$ and an illuminating node $s$ is bounded by equation (2). The total error at $x$ is the sum of the errors over all the contributing links.

**Method B.** Each contributing link stores the unoccluded form factor from the center of its node to the corresponding source, as well as the visibility factor. To obtain a more accurate radiance value for $x$ we can recompute the unoccluded form factor to each source at point $x$. Each form factor is multiplied by the visibility stored with the link and by the radiosity of the source. This results in a smaller bound on the error due to a link between $r$ and $s$

$$E_{rs}(x) \leq \rho_r F_{xs} \left( B_s^{\max} V_{rs}^{\max} - B_s^{\min} V_{rs}^{\min} \right) \tag{3}$$

**Method C.** The next logical step is to recompute both the form factor and the visibility of each source as seen from $x$. In order to obtain an accurate visibility value the visible parts of the source are computed analytically [18]. As a result, the error bound shrinks further:

$$E_{rs}(x) \leq \rho_r F_{xs} V_{xs} \left( B_s^{\max} - B_s^{\min} \right) \tag{4}$$

However, the computation becomes more expensive.

**Method D.** To reduce the cost, we can recompute the visibility for links to primary light sources only. This is justified by the fact that primary sources are typically responsible for the most noticeable shadows. Moreover, these are precisely the sources for which discontinuities have been computed and inserted into the mesh. Thus, we obtain the same accuracy as in method C for links to primary sources, while the error due to other links remains the same as in method B.

```
Spectrum Shade(node, x)
rad ← 0
foreach l ∈ node.links do
   ff ← FormFactor(x, l.source)
   v ← Visibility(x, l)
   rad ← rad + ff * v * l.source.radiosity
end for
if IsInterior(node) then
   if Contains(node.left, x) then
      rad ← rad + Shade(node.left, x)
   else if Contains(node.right, x) then
      rad ← rad + Shade(node.right, x)
   else
      rad ← rad + 0.5 * (Shade(node.left, x)
                         +Shade(node.right, x))
   end if
end if
return rad

Real Visibility(x, link)
case ShadingMethod in
   B: v ← link.visibility
   C: v ← RecomputeVisibility(x, link.source)
   D: if IsPrimary(link.source) then
         v ← RecomputeVisibility(x, link.source)
      else
         v ← link.visibility
      end if
end case
return v
```

Figure 5: Pseudocode for the *Shade* routine

### 6.1 Results

We compared methods A, B, C, and D using a simple model of a square exhibit room displaying a modern sculpture illuminated by two small square light sources.

Three global pass solutions of the exhibit room are shown at the top row of Figure 6, in order of increasing accuracy starting from the left. For each solution, the elements (leaf nodes) of the hierarchical subdivision are shown as flat shaded, outlined polygons. The bottom row of the same figure shows the corresponding local pass meshes. Table 1 reports statistics for both passes.

The results of the global pass were fed to the local pass four times, once for each of the methods A, B, C, and D, yielding a total of twelve radiosity solutions shown in Figure 7. Columns 1, 2, and 3 were computed respectively from the low, medium, and high accuracy global pass solutions shown in Figure 6. Each row corresponds to a different shading strategy starting with method A for the top row.

As demonstrated in the top row, method A is prone to visual artifacts: the shading on walls is flat or not sufficiently smooth; some shadows are entirely missing (image A1), while others have incorrect boundaries. These artifacts are the result of interpolating radiance values obtained by sampling the piecewise constant global pass solution.

Method B reduces some of these artifacts. The appearance of unoccluded areas is greatly improved, since accurate form factor are recomputed at every interpolated point in the mesh. However, the penumbra regions of the shadows cast by the sculpture are still

| | Solution Accuracy | | |
|---|---|---|---|
| | low | medium | high |
| input polygons | 47 | 47 | 47 |
| disc. segments | 559 | 559 | 559 |
| initial links | 652 | 652 | 652 |
| total links | 720 | 1316 | 21805 |
| total nodes | 147 | 803 | 6041 |
| total leaf nodes | 97 | 425 | 3044 |
| CDT elements | 1538 | 2384 | 8177 |
| shading calls | 3799 | 5674 | 17984 |
| initial linking | 6 | 6 | 6 |
| discontinuity comp. | 1 | 1 | 1 |
| hierarchical sol. | 1 | 4 | 60 |
| triangulation | 0.53 | 0.81 | 2.78 |
| method A | 1 | 2 | 13 |
| method B | 6 | 10 | 80 |
| method C | 405 | 624 | 2633 |
| method D | 20 | 26 | 110 |

Table 1: Statistics for images in Figures 6 and 7. Timings are in seconds for execution on an HP 9000/720 workstation.

incorrect and shadows are still missing from the coarse solution (image B1.) The reason is that Method B still uses node-to-node visibility factors to approximate node-to-point visibility.

As shown in row C, method C correctly reconstructs all of the shadows. In particular, note the appearance of the shadows in the coarse solution (image C1.) This method results in the best visual accuracy we were able to obtain, given a global solution.

Method D yields results that are almost indistinguishable from those given by method C. However, as can be seen from the timings reported in Table 1, method D takes only a fraction of the time required by method C. In fact, it is not much more expensive than method B.

When using methods C or D, little difference can be seen between the medium and high accuracy solutions (columns 2 and 3). Although the latter solution is objectively more accurate, from a visual standpoint, the former solution is almost as good. If fact, it is apparent that even very low accuracy global pass solutions can yield results of reasonable visual quality when followed by a local pass using method D (image D1.)

When comparing the computation times reported in Table 1, it can be seen that the local pass is in most cases costlier than the global pass. It may be argued that the time used by the local pass could be better spent in further refinement of the subdivision hierarchy in the global pass. One might expect that if the hierarchy were sufficiently refined, even a very simple shading strategy would have sufficed for visually accurate results. Figure 7, however, demonstrates that this is not the case. Image D2, computed from the medium accuracy global pass followed by method D for the local pass, is visually more accurate than images A3 and B3; yet, it took considerably less time to compute (38 versus 83 and 150 seconds, respectively.)

Another set of comparisons was made to illustrate the importance of including discontinuity segments in the mesh for the local pass. Figure 8 shows a view of the floor of the exhibit room. The top row shows the mesh in wireframe with $D^0$ discontinuities in red and $D^1$ and $D^2$ discontinuities in yellow. The bottom row shows the shaded floor as reconstructed by the local pass. All images were computed from the medium accuracy global pass solution shown in image a2 of Figure 6 and all of them used method D in the local pass. As can be seen from the top row of Figure 8, no discontinuity segments were included in the left mesh, only $D^0$ discontinuities were included in the middle mesh, and all the discontinuity segments were included in the right mesh.

When comparing the corresponding images in the bottom row, the higher quality of the right image stands out. Image b1 presents

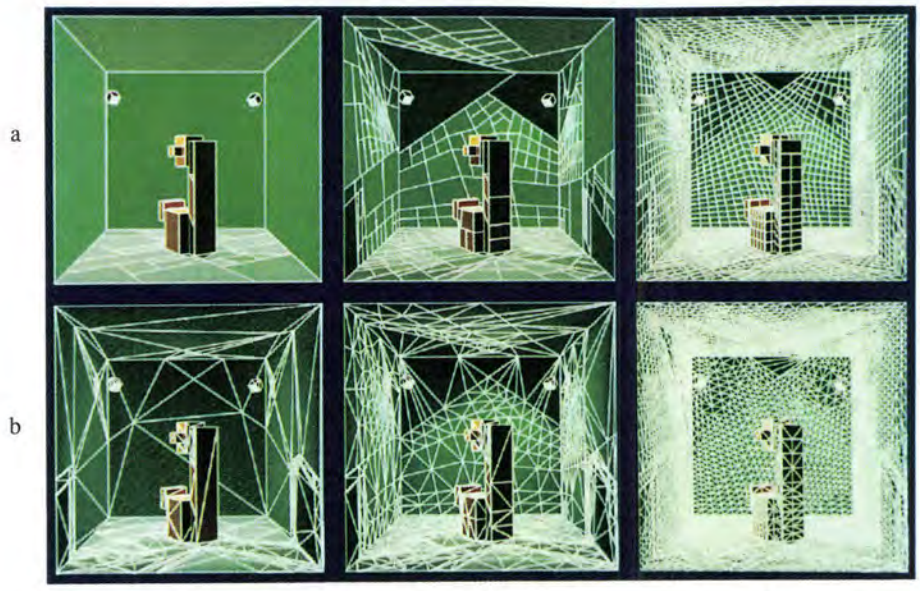



Figure 6: Exhibit Room. Global pass solutions (top row) and the corresponding local pass meshes (bottom row). The accuracy of the solutions increases from left to right.

many of the visual artifacts typical of conventional radiosity methods: shadow and light leaks, fuzzy shadow boundaries, and incorrectly shaped shadows. Image b2 shows how including $D^0$ discontinuities greatly reduces shadow and light leaks, but still has problems reproducing shadow boundaries and penumbra areas. Finally, image b3, correctly captures all shadow boundaries. We conclude, therefore, that it is necessary to represent discontinuities explicitly in the local pass mesh, even though some or all of them may have been resolved by the subdivision in the global pass.

| | Discontinuities in the Mesh | | |
|---|---|---|---|
| | none | $D^0$ | $D^0\ D^1\ D^2$ |
| triangulation | 0.39 | 0.39 | 0.81 |
| shading | 9 | 10 | 26 |
| disc. segments | 0 | 36 | 559 |
| CDT elements | 1170 | 1190 | 2384 |
| shading calls | 2739 | 3027 | 5674 |

Table 2: Statistics for the comparison of meshing strategies shown in Figure 8. Timings are in seconds for execution on an HP 9000/720 workstation.

As the statistics reported in Table 2 show, building a mesh that incorporates discontinuity segments takes longer than building one without discontinuities. Furthermore, including the discontinuities generally results in a larger number of elements and consequently shading the mesh takes longer. We believe, however, that the increased computation time is well justified.

## 7 A FINAL COMPARISON

In this section we demonstrate the performance of our combined approach on an environment of moderate complexity (1,688 input polygons.) Figure 9 shows a rendered view of the scene. There are two primary light sources: a small distant polygonal source outside the room simulates sunlight, and another polygonal source close to the ceiling provides the artificial illumination.

The figure shows two images of the same environment. The left image (HDMR) was generated using primary discontinuity segments in both passes with shading method D in the local pass. To generate the right image (HR) we modified our algorithm to essentially emulate regular HR: discontinuities were not used in either pass, the vertices of the triangles were shaded using method A, and linear interpolation was used for display.

As can be expected in a complex environment, the initial linking stage results in a very large number of initial links, most of which represent interactions of very small magnitude. For efficiency, we use a simple culling strategy: we ignore all the initial links that do not involve a primary light source and whose form factor falls below a user specified threshold. We found that by using a small threshold it is possible to eliminate most of the initial links, without any noticeable change in the resulting images. As was mentioned in Section 2, clustering of input surfaces in the initial linking stage should provide a more comprehensive solution to this problem.

Table 3 reports various statistics for the two solutions from which the images in Figure 9 were rendered. The two solutions have roughly the same number of final triangles, yet the HDMR solution looks dramatically better than the HR solution; while the latter exhibits many of typical problems of radiosity images, HDMR produces sharp shadow boundaries and correct penumbrae, eliminates shadow and light leaks, and captures some small features that are entirely missed by HR. Furthermore, the total computation time was almost twice as long for the HR solution.

We attempted to perform a similar comparison with our progressive DM algorithm [18]. However, we were not able to obtain a converged solution for this environment: after four hours of computation the DM algorithm was still in its fourth iteration.

### 7.1 Complexity of Discontinuity Meshing

A legitimate concern regarding discontinuity meshing is that, in theory, $l$ light source edges and $m$ polygon edges can result in $O(lm)$ distinct EV visual events. In the worst case, each event intersects $O(m)$ polygons, resulting in a total number of $O(lm^2)$ discontinuity segments. In such a case each polygon has $O(lm)$ discontinuity segments, which can result in as many as $O(l^2m^2)$ elements in the discontinuity mesh for that polygon.

We have found that this worst case analysis is too pessimistic in practice. Consider, for example, the environment shown in


Figure 7: Exhibit Room. A comparison of shading strategies. Columns 1, 2, and 3 were computed respectively from the low, medium, and high accuracy global pass solutions shown in Figure 6. Each row corresponds to a different shading strategy; starting from the top: method A, method B, method C, and method D.

Figure 9. In this environment $l$ is 8, and $m$ is 6,744. The worst case upper bound on the number of discontinuity segments on a single polygon is 215,808. In practice, there were 18,664 discontinuity segments in the entire environment, an average of roughly 11 segments per polygon. The highest number of segments on a single polygon (the floor) is 2,175, resulting in only 7,627 triangles in the floor's discontinuity mesh.

## 8 CONCLUSIONS

By combining hierarchical radiosity with discontinuity meshing we have created a new radiosity method that is superior to both of its ancestors: it is more accurate than the HR algorithm, both numerically and visually, and it is faster and more flexible than DM algorithms. The new algorithm is capable of producing high quality images even from quick simulations.

Hierarchical radiosity has been recently extended to deal with very complex environments by introducing the notion of importance into the solution process [22]. This improvement is readily applicable to our algorithm as well: the global pass would simultaneously solve for radiosity and for importance as described by Smits *et al.* [22]; the local pass would only reconstruct the radiance on surfaces which are direct receivers (or emitters) of importance.

There are several aspects of our algorithm that can be substantially improved:

**Visibility computations.** Our implementation uses shaft culling [13] to reliably determine complete visibility between polygons, but point sampling is used to determine whether two polygons are entirely occluded from each other. Our method could be improved by using the accurate and reliable visibility algorithms described by

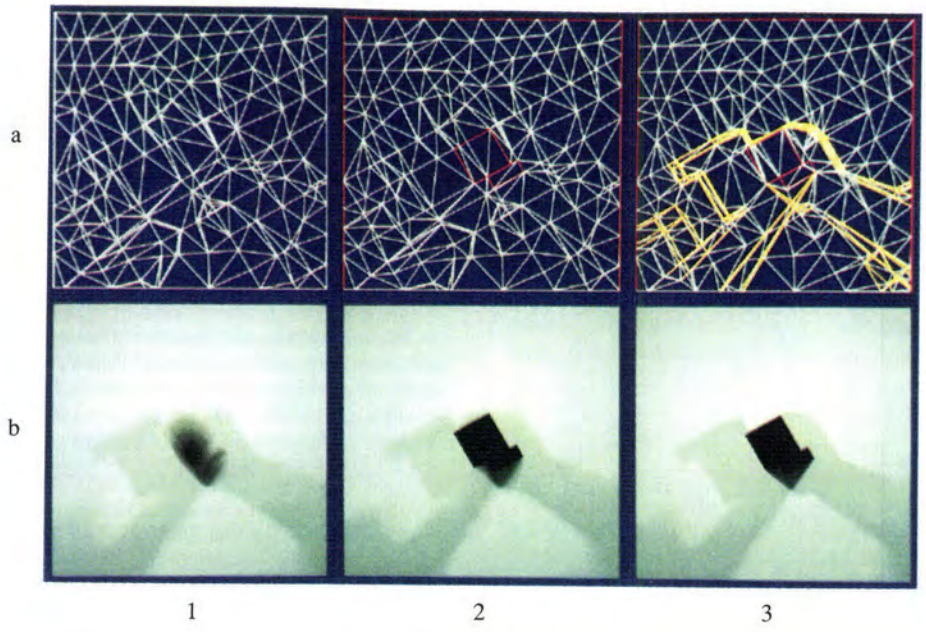



Figure 8: Exhibit Room Floor. A comparison of meshing strategies. Mesh (top row) and computed radiance (bottom row) on the floor using simple CDT (left), CDT with $D^0$ discontinuity segments (middle), and CDT with $D^0$, $D^1$, and $D^2$ discontinuity segments (right).

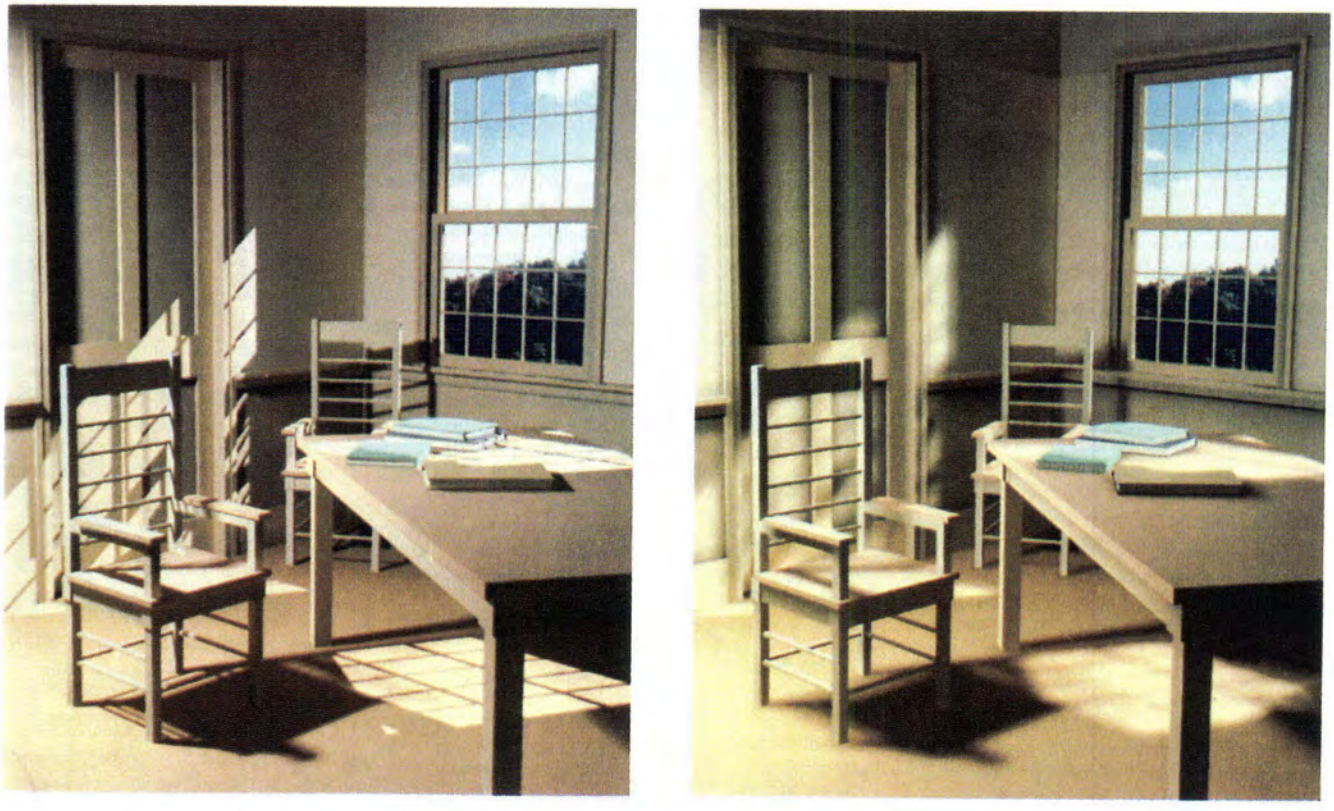


Figure 9: A comparison of Hierarchical Discontinuity Meshing Radiosity (left) vs. Hierarchical Radiosity (right)

| | Radiosity Algorithm | |
|---|---|---|
| | HDMR | HR |
| initial linking | 2 : 16 : 27 | 2 : 16 : 27 |
| discontinuity computations | 0 : 09 : 06 | 0 : 00 : 00 |
| hierarchical solution | 0 : 16 : 42 | 3 : 58 : 01 |
| triangulation | 0 : 00 : 21 | 0 : 00 : 16 |
| shading computations | 0 : 33 : 49 | 0 : 00 : 51 |
| total time (hr:min:sec) | 3 : 16 : 56 | 6 : 15 : 35 |
| input polygons | 1,688 | 1,688 |
| discontinuity segments | 18,664 | 0 |
| initial links | 165,814 | 165,814 |
| links after culling | 27,002 | 27,002 |
| total links | 39,056 | 161,668 |
| total nodes | 5,778 | 35,454 |
| total leaf nodes | 3,733 | 18,571 |
| avg. depth of hierarchy | 1.31 | 2.04 |
| CDT elements | 41,090 | 41,284 |
| shading calls | 109,885 | 101,208 |
| recomputed form factors | 3,609,941 | 0 |
| recomputed visibility terms | 128,705 | 0 |

Table 3: Statistics for the comparison of hierarchical discontinuity meshing radiosity (HDMR) vs. hierarchical radiosity (HR) shown in Figure 9. All timings are for execution on an HP 9000/720 workstation.

Teller and Hanrahan [26].

We need to be able to compute tight bounds on the visibility between two partially occluded polygons. This would improve the efficiency of the global pass by eliminating unnecessary subdivision in penumbral areas.

**Choice of sources.** Our algorithm is particularly effective for environments with a few primary light sources that are responsible for the most noticeable shadows. In general, however, primary light sources do not dominate the illumination on all the surfaces in an environment. Our algorithm should be extended to compute a set of the most dominant sources, primary or secondary, with respect to each receiving surface. This set should be used both for computing the discontinuities on that surface and for determining when visibility should be recomputed in the local pass.

**Choice of discontinuities.** Not all the discontinuities are equally significant. In the global pass, for example, we should choose discontinuities that would resolve partial visibility most effectively, rather than ones that split the node most evenly. In the local pass we need to identify the discontinuities that are visually significant and insert only these discontinuities into the mesh.

## ACKNOWLEDGEMENTS

We would like to thank Brian Smits, Jim Arvo, and Kevin Novins for helpful discussions and for reviewing the manuscript. Ben Trumbore assembled and submitted the review draft when the authors were away on vacation. Suzanne Smits modeled the sculpture used in the examples in Section 6, and Matt Hyatt modeled the room used in the final comparison. This work was supported by the NSF grant, "Interactive Computer Graphics Input and Display Techniques" (CCR-8617880), by the NSF/DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219), and by generous donations of equipment from Hewlett-Packard.

## REFERENCES

[1] Baum, Daniel R., Stephen Mann, Kevin P. Smith, and James M. Winget. "Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions," *Computer Graphics*, 25(4), July 1991, pages 51–60.

[2] Bern, Marshall and David Eppstein. "Mesh Generation and Optimal Triangulation," in Hwang, F.K. and D.-Z. Du, editors, *Computing in Euclidian Geometry*, World Scientific, 1992.

[3] Campbell, III, A. T. *Modeling Global Diffuse Illumination for Image Synthesis*, PhD dissertation, U. of Texas at Austin, Texas, December 1991.

[4] Chen, Shenchang Eric, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. "A Progressive Multi-Pass Method for Global Illumination," *Computer Graphics*, 25(4), July 1991, pages 165–174.

[5] Chew, L. Paul. "Constrained Delaunay Triangulations," *Algorithmica*, 4, 1989, pages 97–108.

[6] Chin, Norman and Steven Feiner. "Fast Object-Precision Shadow Generation for Area Light Sources Using BSP Trees," in Proceedings of 1992 Symposium on Interactive 3D Graphics, March 1992.

[7] Cohen, Michael F. and Donald P. Greenberg. "The Hemi-Cube: A Radiosity Solution for Complex Environments," *Computer Graphics*, 19(3), July 1985, pages 31–40.

[8] Cohen, Michael F., Donald P. Greenberg, and David S. Immel. "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, 6(2), March 1986, pages 26–35.

[9] Gigus, Ziv and Jitendra Malik. "Computing the Aspect Graph for Line Drawings of Polyhedral Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2), February 1990, pages 113–122.

[10] Goral, Cindy M., Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. "Modeling the Interaction of Light Between Diffuse Surfaces," *Computer Graphics*, 18(3), July 1984, pages 213–222.

[11] Guibas, Leonidas and Jorge Stolfi. "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams," *ACM Transactions on Graphics*, 4(2), April 1985, pages 74–123.

[12] Haines, Eric A. "Ronchamp: A Case Study for Radiosity," SIGGRAPH'91 Frontiers in Rendering Course Notes, July 1991.

[13] Haines, Eric A. and John R. Wallace. "Shaft Culling for Efficient Ray-Traced Radiosity," in Proceedings of the Second Eurographics Workshop on Rendering, May 1991.

[14] Hanrahan, Pat, David Salzman, and Larry Aupperle. "A Rapid Hierarchical Radiosity Algorithm," *Computer Graphics*, 25(4), July 1991, pages 197–206.

[15] Heckbert, Paul S. "Discontinuity Meshing for Radiosity," in Proceedings of the Third Eurographics Workshop on Rendering, May 1992, pages 203–216.

[16] Heckbert, Paul S. *Simulating Global Illumination Using Adaptive Meshing*, PhD dissertation, UC Berkeley, California, June 1991.

[17] Kok, Arjan J. F. and Frederik Jansen. "Source Selection for the Direct Lighting Computation in Global Illumination," in Proceedings of the Second Eurographics Workshop on Rendering, May 1991.

[18] Lischinski, Dani, Filippo Tampieri, and Donald P. Greenberg. "Discontinuity Meshing for Accurate Radiosity," *IEEE Computer Graphics and Applications*, 12(6), November 1992, pages 25–39.

[19] Nishita, Tomoyuki and Eihachiro Nakamae. "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflections," *Computer Graphics*, 19(3), July 1985, pages 23–30.

[20] Reichert, Mark C. *A Two-Pass Radiosity Method Driven by Lights and Viewer Position*, Master's thesis, Cornell University, Ithaca, New York, January 1992.

[21] Salesin, David, Dani Lischinski, and Tony DeRose. "Reconstructing Illumination Functions with Selected Discontinuities," in Proceedings of the Third Eurographics Workshop on Rendering, May 1992, pages 99–112.

[22] Smits, Brian E., James R. Arvo, and David H. Salesin. "An Importance-Driven Radiosity Algorithm," *Computer Graphics*, 26(4), July 1992, pages 273–282.

[23] Sparrow, Ephraim M. "On the Calculation of Radiant Interchange between Surfaces," in Ibele, Warren E., editor, *Modern Developements in Heat Transfer*, Academic Press, New York, 1963.

[24] Tampieri, Filippo. *Discontinuity Meshing for Radiosity Image Synthesis*, PhD dissertation, Cornell University, Ithaca, New York, May 1993.

[25] Teller, Seth J. "Computing the Antipenumbra of an Area Light Source," *Computer Graphics*, 26(4), July 1992, pages 139–148.

[26] Teller, Seth and Pat Hanrahan. "Global Visibility Algorithms for Illumination Computations," *Computer Graphics*, 27(4), August 1993.

[27] Zienkiewicz, O. C. and R. L. Taylor. *The Finite Element Method*, pages 128–132, Vol. 1, McGraw-Hill, London, 4th edition, 1989.

# Radiosity Algorithms Using Higher Order Finite Element Methods

Roy Troutman, Nelson L. Max

Lawrence Livermore National Laboratory

## Abstract

Many of the current radiosity algorithms create a piecewise constant approximation to the actual radiosity. Through interpolation and extrapolation, a continuous solution is obtained. An accurate solution is found by increasing the number of patches which describe the scene. This has the effect of increasing the computation time as well as the memory requirements. By using techniques found in the finite element method, we can incorporate an interpolation function directly into our form factor computation. We can then use less elements to achieve a more accurate solution. Two algorithms, derived from the finite element method, are described and analyzed.

**CR Categories and Subject Descriptors**: 1.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms. 1.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

**Additional Key Words and Phrases**: finite elements, form-factor, global illumination, radiosity.

## 1 Introduction

The traditional radiosity algorithm computes the form factors at a collection of points [5]. There have been several techniques used to enhance and speed up the algorithm. Cohen described an algorithm which enabled more complex environments to be rendered by placing a half cube or "hemicube" at each evaluation point and sampling through pixels on the hemicube surface [3]. We can improve the accuracy of the solution by increasing the resolution of the hemicube or by analytically determining the form factors [1], [13]. Further improvements can be made by producing a mesh which follows the discontinuities introduced by shadow boundaries and surface intersections [9], [10].

Rather than assuming the radiosity arrives from piecewise constant patches, Max and Allison introduced an algorithm which assumed a piecewise linear approximation [11]. This algorithm works by placing an interpolation function directly into the form factor computation. Using this technique, a more accurate solution can be obtained with less patches [9]. An extension to this algorithm is to increase the order of the interpolation function to quadratic, cubic or even higher [16]. These interpolation functions are what the finite element method refers to as basis functions [2].

## 2 Basis Functions

The details of the basis functions, elements and nodes can be found in [2]. We will only give a brief overview to establish our terminology.

### 2.1 Approximation Function

The finite element method associates a basis function for each of the local nodes in a representative element. The basis function for a global node becomes a combination of basis functions defined on the local nodes of all elements which contain the global node [2], [9], [14]. In this paper a node is "contained" or "in" an element if it is on the boundary or interior of the element. Using the basis function $f_i$ and radiosity $B_i$ associated with each node $i$, we can approximate the radiosity at a point $x$ in our environment as a linear combination of the radiosities of each node or

$$\tilde{B}(x) = \sum_{i=1}^{n} B_i f_i(x) \qquad \text{(EQ 1)}$$

### 2.2 Element Construction

Due to its compatibility with triangulation and orientation independence using Gouraud shading, we have chosen the triangle as our element. We also need to concern ourselves with the connectivity of this element. Our solution will be much more accurate if we align our mesh to the $D^0$ and $D^1$ discontinuities as described by [9] and [10]. We can obtain elements with $C^0$ continuity by using the same nodes on the boundary of adjacent elements [2], [4]. By definition, the $C^0$ elements can accurately model $D^1$ discontinuities. A $D^0$ discontinuity would result from surface intersection, discrete changes in emissivity or discrete changes in reflectivity. These can be modeled by aligning our edges to the discontinuities and duplicating the nodes along the edge [9]. Higher order discontinuities could be modeled by selectively enforcing higher derivative continuity across the common edges between adjacent elements, but this is quite complex [12] so we approximate them by using smaller elements.

## 3 Finite Element Methods

This section will give a very brief introduction to finite element mathematics to provide us with modifications needed for the radiosity algorithm to incorporate higher order elements and the previously discussed basis functions.

### 3.1 Residual Error

We start by reiterating an equation from [8] which describes the radiosity for all points in the environment

$$B(x) = E(x) + \rho(x) \int_{\Omega} ds\, \kappa(x, s)\, B(s) \qquad \text{(EQ 2)}$$

where

$$\kappa(x, s) = V(x, s) \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \qquad \text{(EQ 3)}$$

Exact solutions to (EQ 2) are known only in the simplest of geometries [8]. The exact solution can be approximated using the linear combination in (EQ 1). Traditional radiosity methods can be thought of as having a constant basis function of $f_i(x) = 1$ for all points $x$ inside patch $i$. These *constant basis* radiosity algorithms will not be reiterated. The method introduced in [11] uses linear basis functions centered on the vertices. We will be presenting algorithms for extending *polynomial basis* radiosity to higher order polynomials.

The traditional radiosity method assigns an emissivity and reflectivity to each patch. We can enhance our radiosity algorithms by claiming that the exact emissivity and reflectivity are also defined by an approximation function similar to (EQ 1). This would allow us to describe variations in emissivity and reflectivity up to the degree of the basis function. For the sake of brevity, we will assume that the

Address: P.O. Box 808, Livermore, Ca. 94550
email: roy@nersc.gov, max2@llnl.gov

emissivity $e_k$ and reflectivity $\rho_k$ are constant across each individual surface, $k$.

We can replace $B(s)$ in (EQ 2) by our approximation function to obtain an approximate solution for $B(x)$. If our approximation is good, then the approximate solution for $B(x)$ and the value obtained by applying (EQ 2) at point $x$ should be close. If our approximate solution is exact, the difference between these two approximations will be zero. This gives us a measure of the accuracy of our approximation and is defined as the *residual error*. More specifically, it is expressed as

$$r(x) = E(x) + \rho(x)\int_{\Omega} ds\, \kappa(x,s)\,\tilde{B}(s) - \tilde{B}(x) \qquad \textbf{(EQ 4)}$$

### 3.2 Method of Weighted Residuals

A general approximation technique is the method of weighted residuals. This technique requires the residual error to be orthogonal to a set $\{w_i(x)\}$ of weighting functions over the domain $\Omega$. It was shown in [8] that the resulting equations could be expressed in matrix form as

$$[\mathbf{M} - \mathbf{K}]\,\mathbf{B} = \mathbf{E} \qquad \textbf{(EQ 5)}$$

where $\mathbf{B}$ is a column vector containing the coefficients to our approximation and

$$M_{ij} = \int_{\Omega} dx\; w_i(x) f_j(x)$$

$$K_{ij} = \rho_k \int_{\Omega} dx\; w_i(x) \int_{\Omega} ds\, \kappa(x,s) f_j(s) \qquad \textbf{(EQ 6)}$$

$$E_i = e_k \int_{\Omega} dx\; w_i(x)$$

where $k$ is the index of the surface supporting weight function $w_i$.

## 4 Higher Order Algorithms

We have presented a set of interpolation functions in section 2 and combined them with our radiosity integral using the finite element method in section 3. This gave us a matrix equation where each component of the matrix contained a weighting function. By replacing the weights with different functions we obtain the point collocation and Galerkin methods [8].

### 4.1 Point Collocation Method

The traditional gathering algorithm as well as the linear vertex radiosity method introduced in [11] are examples of the point collocation method. This method replaces the weighting function in (EQ 6) with the dirac delta [8]. This simplifies the $\mathbf{M}$ in (EQ 5) to be the identity matrix and $\mathbf{E}$ to be a column vector containing the emissivities at each node. The elements of $\mathbf{K}$ have the value

$$K_{ij} = \rho_k \int_{\Omega} ds\, \kappa(x_i, s) f_j(s) \qquad \textbf{(EQ 7)}$$

The contents of the integral describes a differential area to weighted area form factor where the area is defined by the domain of the basis associated with node $j$. We will call this a differential area to basis form factor. This integral can be solved using the approach specified in [11]. The pseudocode is as follows

```
Initialize F to 0
For each pixel h in hemicube
    k = index of patch at h
    Q = point on surface of k
    For each node j in patch k
        F_ij = F_ij + f_j(Q)Δ_h
    EndFor
EndFor
```

### 4.2 Galerkin Method

The Galerkin method replaces the weighting functions with the basis functions giving us the following definitions for the matrices of (EQ 5).

$$M_{ij} = \int_{\Omega} dx\; f_i(x) f_j(x)$$

$$K_{ij} = \rho_k \int_{\Omega} dx\; f_i(x) \int_{\Omega} ds\, \kappa(x,s) f_j(s) \qquad \textbf{(EQ 8)}$$

$$E_i = e_k \int_{\Omega} dx\; f_i(x)$$

We'll start by looking at the equation for $M_{ij}$. We only need to concern ourselves with the area where $f_i$ and $f_j$ are both non-zero. This will only occur if an element can be found which has the nodes $i$ and $j$ on the boundary or interior. Clearly this occurs if $i$ equals $j$. We can easily compute $M_{ij}$ by considering only the elements which contain node $i$ and looking through that small set of elements for the elements which also contain node $j$. We then integrate across these elements individually and sum the results. The formula for 2-D change of variables from the global triangle to the representative element gives us the Jacobian determinant which is the area of the global element. Therefore, the integral across an element is the same as the integral across the representative element multiplied by the area of the element. The final result is a constant multiplied by the area of the global element. The constant is dependent upon the relative positions of the local nodes corresponding to $i$ and $j$. We can store these constants in a matrix $\mathbf{M}_c$. This matrix is symmetrical, which is what we would expect by looking at the equation. The local node numbers for $i$ and $j$ correspond to the row and column of a location this matrix.

Solving for $E_i$ follows a similar path. In this case, we must integrate across the domain of the basis function. We can form a vector $\mathbf{E}_c$ which contains the integral of all of the local nodes across the representative element. To compute $E_i$, we look at each element which has node i, use the local node number as an index into $\mathbf{E}_c$, multiply that array element by the area and then add it to the current value of $E_i$. After we visit each element, we multiply our result by the node emission $e_i$.

Computing $K_{ij}$ is slightly more involved. We know from (EQ 7) that the inner integral is a differential area to weighted area form factor. In the Galerkin case, the weighted area still corresponds to the domain of a basis function, but the differential area corresponds to some point $x$ in the domain of $f_i$. We will express this differential area to basis form factor as $F_{xj}$. Our equation simplifies to

$$K_{ij} = \rho_k \int_{\Omega} dx\; f_i(x) F_{xj} \qquad \textbf{(EQ 9)}$$

The contents of the integral describe a basis to basis form factor. This integral is in a form that is appropriate for Gaussian quadrature [2], [16]. The problem of computing $K_{ij}$ is now reduced to computing a set of form factors, adding the results multiplied by the appropriate weight and multiplying by the reflectivity.

To compute the basis to basis form factor with gaussian quadrature we start by specifying the degree of precision [2]. This provides us with a collection of gauss points on each element. We compute an array of differential area to basis form factors $FF$ (computed by the algorithm in section 4.1) at each of the gauss points. This hemicube will affect the basis to basis form factors associated with each node in the element. After we have completed computing the entire matrix of form factors, we multiply each row by the reflectivity to obtain $\mathbf{K}$. This gives us the following algorithm

```
Initialize F to 0
For each patch p
    a_p = area of patch p
    For each gauss point l
        Q_l = global coordinate of point l
        w_l = weight assigned to point l
        FF = array of form factors computed at Q_l
        For each node i in p
            For j = 1 to total number of nodes
                F_ij = F_ij + a_p × w_l × f_i(Q_l) × FF(j)
            Endfor
        Endfor
    Endfor
Endfor
```

The matrix **M** is very sparse. To avoid using an excessive amount of memory due to random access, the matrix is computed one row at a time. This requires us to visit a node and find the patches which share this node. The winged edge data structure allows us to easily determine adjacent elements. The algorithm for computing **M** and **E** is as follows.

```
Initialize M and E to 0
For each node i
    P = set of patches containing i
    For each patch p ∈ P
        a_p = area of patch p
        J = set of nodes in p
        l_i = local node number of node i
        For each j ∈ J
            l_j = local node number of node j
            M_ij = M_ij + a_p × M_c(l_i, l_j)
        Endfor
        E_i = E_i + a_p × E_c(l_i)
    Endfor
Endfor
```

To solve for **B**, we multiply each row of the form factor matrix by the reflectivities and solve the matrix using the Gauss-Seidel iteration method.

## 5 Analysis

A quantitative measurement of the accuracy of our algorithms are obtained by applying an error metric. We will apply this metric to images generated by our collocation and Galerkin algorithms.

### 5.1 Error Metric

We determine the RMS radiosity reconstruction error by rendering each surface individually at the same distance. Our reference image was obtained by using quadratic basis elements on a discontinuity meshed version of our scene and applying a hemicube with a resolution of 314 x 314 x 157. Since 157 is a prime number, the chance of a correlation with a lower resolution hemicube is reduced. The floating point radiosity of each pixel is compared to the same pixel in each rendered surface of the reference. The error is measured using

$$\sqrt{\frac{\sum_{i=1}^{n}\left(\frac{r_i - e_i}{r_i}\right)^2}{n}} \qquad \text{(EQ 10)}$$

where $r_i$ is the radiosity value at pixel $i$ in the reference scene, $e_i$ is the radiosity at pixel $i$ in a test scene and $n$ is the total number of pixels occupied by a rendered image of every surface. Computing the error in this manner reduces the chance of bias since it is improbable that any particular node in our scene will be on a pixel center.

### 5.2 Collocation Results

The collocation algorithm was implemented on a Cray YMP/C90. Timing information was obtained using the Unix *times* function. The computation time was considered to be the time spent executing the code added to the time spent completing system calls. The total time includes the time spent generating and solving the matrix. We do not include I/O times. We could not include the time spent generating the mesh since some of the following meshes were generated by hand. Because of its flexibility, Heckbert's software z-buffer [7] was used to project the environment onto the hemicube. Although the program computed the radiosity for the red, green and blue components, only the blue component was used to determine the error.

We start by analyzing the source of error in Figure 1. Most of the error in this image is due to the shadow edges on the floor. By rendering the radiosity of the floor as a 3-D shaded surface, we enhance the radiosity discontinuities that would not be visible when the scene is rendered as an image. Figure 1 also shows the radiosity of the floor of the reference scene rendered in this manner. This gives us more information about where errors occur as well as how close our approximation is to a converged reference.

The reference image appeared to be extremely smooth. However, when we looked at the floor rendered as a shaded surface, slight discontinuities due to hemicube aliasing were detected. These artifacts are referred to as plaid patterns in [1] and [15]. When the edge of a light source is parallel or at a 45 degree angle to the edge of the hemicube, the amount of aliasing is greatly enhanced. In some scenes it may be possible to determine an ideal rotation for the hemicube in order to reduce aliasing, but when we introduce occlusion, the apparent edge of a source changes. In general, we can reduce the chance of a poor alignment by introducing a random rotation to the hemicube. This gives a slight improvement in terms of numerical error and a big improvement in terms of visually perceptible error. To improve our reference even further, we solved for the radiosity several times and averaged the results.

We applied a uniform and discontinuity mesh to Figure 1. By increasing the resolution of a mesh, the amount of computation time increases as the error decreases. We did not have access to triangulation software that would easily allow us to create a variable sized discontinuity mesh. An interactive mesh generator called Maze [6] was used to produce a set of quadrilaterals which were then split into triangles. One of the goals used in producing this mesh was to limit the number of slices or poorly formed elements. Once the mesh was created, we were easily able to further subdivide the resulting triangles to improve the accuracy of our solution.

Figure 2 shows the results of the algorithm when applied to the scene shown in Figure 1 with a 100 x 100 x 50 hemicube. A log error of less than -1.3 generated an image which was very difficult to distinguish from the reference. A log error of less than -1.6 generated an image which could not be distinguished from the reference even with high quality display devices. A log error of approximately -2 was mostly due to hemicube aliasing. Note that the linear and constant uniform elements did not obtain these error levels in the time frame shown. The effect on the form factors due to visibility changes is basically quadratic, so we did not expect or see a great deal of improvement in the cubic element over the quadratic element. Discontinuity meshing showed the most impressive results. The linear discontinuity elements produced the same error as the best uniform elements in half the time. The quadratic and cubic discontinuity elements produced an error level so small that further reduction could only be obtained by increasing the resolution of the hemicube.

The collocation algorithm was applied to other simple scenes. In some cases, even the higher order $C^0$ elements did not conform well to the radiosity solution along edges of high variance. These edges can be found near dimly lit corners of a closed room.

### 5.3 Galerkin Results

The Galerkin method was also implemented on the Cray YMP/C90. This method required only minor modifications to the existing collocation algorithm. The program was implemented so that the user could specify the number of degrees of precision. It was shown in [14] that in the case of the uniform mesh, the optimal degrees of

precision for the constant, linear, quadratic and cubic basis were one, one, four and four, respectively. At these levels the algorithm produced less error in less time. We also found that the discontinuity mesh produced these same optimal degrees of precision. The constant basis Galerkin algorithm is identical to the constant basis collocation algorithm [14], so we don't present these results.

Figure 2 shows the results of applying the Galerkin method using a uniform and discontinuity mesh for the scene in Figure 1. We used the same 100 x 100 x 50 resolution hemicube for these tests. The quadratic and cubic basis achieved a lower error than the linear for both meshes. The quadratic and cubic discontinuity mesh again produced very small error levels immediately. In comparison to the collocation method, the Galerkin method as we implemented it took considerably more time to achieve the same error level.

In general, the number of patches for a scene with triangular elements is larger than the number of vertices. Since the number of hemicubes used in Gaussian quadrature depends on the number of patches, our Galerkin implementation required many more hemicubes to compute the matrices for the same scene. Figure 1 was computed using the collocation method with a quadratic basis. This required 2282 hemicubes. Our Galerkin implementation would require 3042 hemicubes if we used just 2 degrees of precision.

## 6 Conclusions and Future Work

We have presented two radiosity algorithms which use the finite element method and higher order basis functions to produce a more accurate solution in less time. The collocation method proved to be easier to implement and converged faster than the Galerkin method for the scenes presented. By applying discontinuity meshing, our algorithms computed an accurate solution in only a fraction of the time used by traditional methods.

Applying adaptive meshing to these algorithms could present a challenging problem. In a more traditional approach, we would subdivide our patches if the gradient became too large. The higher order elements can model these large gradients. The criteria for subdivision would require some investigation [10].

We restricted our elements to planar $C^0$ triangles. We could use square or even curved elements. The curved element may require ray tracing for computing the form factors, but it would allow us to use exact geometries. An additional improvement to the accuracy might be possible by using $C^1$ reconstruction technique [12].

Modifications to the algorithms for progressive radiosity is possible. The collocation method could be used with a ray tracing algorithm similar to [15]. In this case we would have to sample the entire basis domain rather than a single patch. Creating an efficient algorithm would be another puzzle.

**Figure 1** Reference scene used for convergence test. On the left is the rendered scene. On the right is a height field showing the radiosity of the floor.

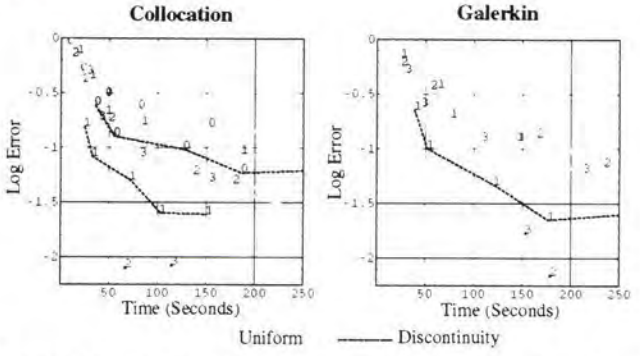



**Figure 2** Error plots. The solid lines are the results from the uniform mesh. The dashed lines and single points are from the discontinuity mesh. The numbers indicate the degree of the basis functions. The error is reduced by increasing the number of elements.

## Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

## References

[1] Baum, Daniel R., Holly E. Rushmeier, James M. Winget, *Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors*, Computer Graphics 23(3), July 1989

[2] Burnett, David S., *Finite Element Analysis*, Addison Wesley Publishing Co., Reading, Massachusetts, May 1988

[3] Cohen, Michael F., Donald P. Greenberg, *The Hemicube: A Radiosity Solution for Complex Environments*, Computer Graphics 19(3), July 1985

[4] Farin, Gerald, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, 1990.

[5] Goral, Cindy M., Kenneth E. Torrance, Donald P. Greenberg, Bennett Battaile, *Modeling the Interaction of Light Between Diffuse Surfaces*, Computer Graphics 18(3), July 1984

[6] Hallquist, John O., *MAZE - An Input Generator for DYNA2D and NIKE2D*, LLNL Tech. Report, UCID-19029, Rev. 2.

[7] Heckbert, Paul S., *Generic Convex Polygon Scan Conversion and Clipping*, Graphics Gems, Academic Press, 1990

[8] Heckbert, Paul S., James M. Winget, *Finite Element Methods for Global Illumination*,U.C. Berkeley, Jan. 1991

[9] Heckbert, Paul S., *Discontinuity Meshing for Radiosity*, Third Eurographics Workshop on Photorealism, Consolidation Express, Bristol, England, May 1992

[10] Lischinski, Dani, Filippo Tampieri, Donald P. Greenberg, *Combining Hierarchical Radiosity and Discontinuity Meshing*, Computer Graphics, Annual Conf. Series, Aug. 1993

[11] Max, Nelson L., Michael J. Allison, *Linear Radiosity Approximations using Vertex-to-Vertex Form Factors*, Graphics Gems III, Academic Press, 1992

[12] Salesin, David, Dani Lischinski, Tony DeRose, *Reconstructing Illumination Functions with Selected Discontinuities*, Third Eurographics Workshop on Photorealism, Consolidation Express, Bristol, England, May 1992

[13] Siegel, Robert, John R. Howell, *Thermal Radiation Heat Transfer*, McGraw-Hill Book Co., N.Y., 1972

[14] Troutman, Roy, *Parallel Radiosity Algorithms using Higher Order Finite Elements*, Master's thesis, U.C. Davis, Dec. 1992

[15] Wallace, John R., Kells A. Elmquist, Eric A. Haines, *A Ray Tracing Algorithm For Progressive Radiosity*, Computer Graphics 23(3), July 1989

[16] Zats, Harold R., *Galerkin Radiosity: Higher Order Global Illumination*, Computer Graphics, Annual Conf. Series, Aug. 1993



# Galerkin Radiosity:

## A Higher Order Solution Method for Global Illumination

Harold R. Zatz[1]
Cornell Program of Computer Graphics

## Abstract

This paper presents an alternative radiosity formulation using piecewise smooth radiance functions that incorporates curved surfaces directly. Using the Galerkin integral equation technique as a mathematical foundation, surface radiance functions are approximated by polynomials. This model eliminates the need for *a posteriori* rendering interpolation, and allows the direct use of non-planar parametric surfaces. Convergence problems due to singularities in the radiosity kernel are analyzed and rectified, and sources of approximation error are examined. The incorporation of a shadow masking technique vastly reduces the need for meshing and associated storage space—accurate radiosity calculations can often be made with no meshing. The technique is demonstrated on traditional radiosity scenes, as well as environments with untessellated curved surfaces.

**CR Categories and Subject Descriptors:** I.3.7 **[Computer Graphics]**: Three-Dimensional Graphics and Realism; I.3.3 **[Computer Graphics]**: Picture/Image Generation.

**Additional Keywords and Phrases:** global illumination, radiosity, integral equations, Galerkin methods, curved surfaces, progressive refinement.

## 1 Introduction

The behavior of light interacting with a macroscopic environment is extremely complex. Despite considerable effort spent searching for a closed-form solution to global illumination problems [10, 22], it seems unlikely that such an approach will be found. To produce computer-generated pictures in a reasonable amount of time, approximations must be used. Typical approximation techniques include the use of direct lighting only, tessellation of the simulated environment into polygonal surfaces, constant or linear shading of surfaces, and sampling the intensity distribution at a limited number of points.

Goral *et al.* [7] introduced the conventional radiosity approximations to computer graphics, assuming surfaces have purely diffuse reflectance distributions, and that finite regions on these surfaces have locally constant radiosity values. Intensity variations across a surface are accounted for by meshing it into a large number of smaller pieces.

Although these assumptions are effective, recent research has demonstrated their limitations. Conventional radiosity techniques generally require that objects be flat or polygonal [1, 7, 3], even though Wallace has demonstrated [21] that radiosity transfers can be computed between non-planar surfaces. Generating images with accurately placed shadows involves a lengthy meshing process, whether surfaces are divided along arbitrary lines [15, 2, 9] or along actual lines of shadow discontinuity [13, 12].

In finite element analysis, it is often possible to trade off a large number of lower-order elements for a smaller number of higher-order elements. Sparrow [18] and Heckbert [10, 11] have successfully applied higher-order radiosity techniques to special-case geometries. Max and Allison [14] explored some of the difficulties of using a linear elements in more general radiosity meshes. In this paper we reformulate the radiosity equations with the goal of applying higher-order *Galerkin* techniques to more general environments, paying particular attention to the difficulties caused by singularities and shadow discontinuities. Benefits of this approach include the direct incorporation of curved surfaces into the solution technique, as well as a significant memory savings due to a drastic reduction of mesh size.

The Galerkin method does have its disadvantages; dealing with shadows and extremely bright light sources can be tricky, and computationally expensive singularities can appear in many places in a complex environment. However, the use of higher-order functions to replace meshing provides a different perspective on the difficulties of the global illumination problem, avoiding some of the difficulties of conventional methods.

[1]now at Rhythm and Hues Studios, Inc., 910 North Sycamore Ave., Hollywood, CA 90038. E-mail: hzatz@rhythm.com or hzatz@alumni.caltech.edu



## 2 Background

The radiosity model of global illumination is based on the principle of energy conservation. All light energy emitted within an enclosure is tracked as it reflects off surfaces within that environment, until it dissipates into heat. Conventional radiosity methods [1, 2, 3, 4, 7, 9, 15, 21] generally simplify the solution procedure by using the Constant Radiosity Assumption [20]—the primary assumption that radiosity values are constant over finite regions, and subsidiary assumptions that emittance, reflectivity, and surface normals are also constant over finite regions. Unfortunately, this constant, polygonal approach to the radiosity problem limits the solution accuracy. Conventional radiosity methods attempt to compensate by increasing the mesh density, assuming that the environment can be accurately approximated if enough polygons. However, the number of polygons needed often exceeds the memory and computational resources available.

Tampieri and Lischinski [20] further explain that the Constant Radiosity Assumption leads to fundamental errors in radiosity computations. A solution computed on a tessellated surface can only be as accurate as the tessellation. The Constant Radiosity Assumption also presents inconsistency between its illumination and rendering phases. During the energy transfer phase, radiosity is assumed constant across each polygon. However, radiosity renderings are made by sampling each polygon at a few points and then interpolating brightness values between these points. Basic signal processing shows that while interpolating a solution may make an image look more accurate, all such interpolation can do is mask error by blurring the image. A consistent radiosity solution must incorporate the interpolation into the energy transfer calculations.

### 2.1 The Radiosity Integral Equation

In order to apply the appropriate mathematical tools to the solution of radiosity problems, it is convenient to express the radiosity equation in parametric form. Parametrically, the key radiosity variables (radiosity, emittance, reflectivity, *etc.*) are represented as functions of two variables, $(s, t)$ or $(u, v)$, over each surface $i$ or $j$. By abstracting all the complexity of surface interaction into a single kernel function $K_{ij}(s, t, u, v)$, the radiosity equation can be written as an integral equation,

$$B_i(s,t) = E_i(s,t) + \sum_j \int\int K_{ij}(s,t,u,v) B_j(u,v)\,du\,dv, \qquad (1)$$

where the kernel function $K_{ij}(s, t, u, v)$ is the product of the double-differential form factor $F_{i-j}(s, t, u, v)$, reflectivity $\rho_i(s, t)$, area $A_i(s, t)$, and visibility $\mathrm{VIS}_{ij}(s, t, u, v)$

$$K_{ij}(s,t,u,v) = \rho_i(s,t) F_{i-j}(s,t,u,v) \mathrm{VIS}_{ij}(s,t,u,v) A_j(u,v). \qquad (2)$$

 ACM-0-89791-601-8/93/008/0213 $01.50

The form factor and area functions can be further expanded in terms of the functions describing surface geometry $\vec{x}_i(s,t)$ and normals $\hat{n}_i(s,t)$:

$$F_{i-j}(s,t,u,v) = \frac{(\hat{n}_i(s,t) - \hat{n}_j(u,v)) \cdot (\vec{x}_j(u,v) - \vec{x}_i(s,t))}{\pi \|\vec{x}_i(s,t) - \vec{x}_j(u,v)\|^4}$$

$$A_j(u,v) = \left\| \frac{\partial \vec{x}_j(u,v)}{\partial u} \times \frac{\partial \vec{x}_j(u,v)}{\partial v} \right\| \tag{3}$$

# 3 Mathematical Background

The Galerkin method provides a method for solving integral equations in terms of a basis set of non-constant functions across each surface. This section provides the mathematical background necessary to apply the Galerkin method to the radiosity equation.

## 3.1 Basis Set Projection

To approximate the radiosity distribution by a combination of functions, we first need formal tools to manipulate an appropriate two-dimensional basis set. We denote this basis set $\{\mathcal{T}_k(s,t) | k = 0, 1, \ldots\}$, where $s$ and $t$ are the parametric variables across a surface, and $k$ specifies a particular function in the set.

Just as geometric vectors have a dot product that projects one onto the other, the inner product of two functions $f(s,t)$ and $g(s,t)$ can be defined,

$$\langle f | g \rangle_{\mathcal{W}} = \int_{-1}^{1} \int_{-1}^{1} f(s,t) g(s,t) \mathcal{W}(s,t) ds\, dt. \tag{4}$$

$\mathcal{W}(s,t)$ is some weighting function that describes the importance of different positions to the inner product. To apply the Galerkin method to radiosity, we use an orthonormal set of basis functions, $\{\mathcal{T}_k(s,t)\}$—a set designed so that for a particular inner product weight function $\mathcal{W}(s,t)$,

$$\forall_{k,l} \quad \langle \mathcal{T}_k | \mathcal{T}_l \rangle_{\mathcal{W}} = \delta_{kl}. \tag{5}$$

Finding the combination of orthonormal basis functions closest to some particular function is relatively simple. Given that the radiosity function over surface $i$ is $B_i(s,t)$, we define the coefficients $B_i^k$

$$B_i^k = \langle B_i | \mathcal{T}_k \rangle_{\mathcal{W}}. \tag{6}$$

The original function can be approximated by the weighted sum,

$$B_i(s,t) \approx \sum_k B_i^k \mathcal{T}_k(s,t). \tag{7}$$

## 3.2 Legendre and Jacobi Polynomials

The Galerkin method is usually solved using an orthonormal polynomial basis set, defined on the interval $[-1, 1]$. Legendre and Jacobi polynomials are one-dimensional, orthonormal polynomials which can be combined into a two-dimensional basis set by multiplying two polynomials in different variables. We limit our analysis in the next two sections to polynomials of one variable.

When the inner product has a weight function equal to one, the polynomials formed are the Legendre polynomials. The unnormalized Legendre polynomials are generated by a recursion rule [8],

$$P_0(x) = 1 \qquad P_1(x) = x$$
$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x). \tag{8}$$

The normalized Legendre polynomials are

$$\bar{P}_n(x) = \sqrt{n + \frac{1}{2}} P_n(x) \tag{9}$$

Polynomial sets can also be created with non-constant inner product weight functions $\mathcal{W}(x)$. Later in this paper (section 4.2), a set of polynomials will be needed with a weight function that has a multiple zero at its endpoints. The Jacobi polynomials $P_i^{(\alpha,\beta)}$ have such behavior, with the weight function,

$$\mathcal{W}(x) = (1-x)^\alpha (1+x)^\beta, \tag{10}$$

where $\alpha$ and $\beta$ are the degree of multiplicity.

The unnormalized Jacobi polynomials have a more complex recursion rule than the Legendre polynomials [5]:

$$P_0^{(\alpha,\beta)}(x) = 1 \qquad P_1^{(\alpha,\beta)}(x) = \frac{\alpha-\beta}{2} + \frac{2+\alpha+\beta}{2}x$$
$$P_{n+1}^{(\alpha,\beta)}(x) = \frac{A_n^{(\alpha,\beta)}x - B_n^{(\alpha,\beta)}}{C_n^{(\alpha,\beta)}} \tag{11}$$

where

$$A_n^{(\alpha,\beta)} = (2n+\alpha+\beta+1)(\alpha^2-\beta^2+2n+\alpha+\beta+2) \times (2n+\alpha+\beta)P_n^{(\alpha,\beta)}(x)$$
$$B_n^{(\alpha,\beta)} = 2(n+\alpha)(n+\beta)(2n+\alpha+\beta+2)P_{n-1}^{(\alpha,\beta)}(x)$$
$$C_n^{(\alpha,\beta)} = 2(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta) \tag{12}$$

These polynomials can be normalized by the factor [8]:

$$\sqrt{\frac{\Gamma(n+1)\Gamma(\alpha+\beta+1+n)(\alpha+\beta+1+2n)}{\Gamma(\alpha+1+n)\Gamma(\beta+1+n)2^{\alpha+\beta+1}}} \tag{13}$$

## 3.3 Quadrature Rules

An informative explanation of one-dimensional quadrature rules has been compiled by Delves and Mohamed [6]. A condensed version is presented here.

A quadrature rule is a method for approximating the integral of a function by a weighted sum of function samples at particular points. Quadrature rules can be used to approximate inner product integrals, like that in (4). Given a fixed function $\mathcal{W}(x)$ and another function $f(x)$, we can choose points $\xi_i$ and weights $w_i$ such that:

$$\int_a^b f(x)\mathcal{W}(x)dx \approx \sum_i w_i f(\xi_i) \tag{14}$$

Quadrature rules can be designed to be exact for a certain class of functions. The Gaussian quadrature rules, by computing optimal positions for the $N$ sample points $\xi_i$, are exact for polynomials up to order $2N-1$. The Gauss quadrature rule with weight function $\mathcal{W}(x)$ is closely tied to the set of orthogonal polynomials with the same weight function.

To develop an $N$-point Gauss quadrature rule for the integral

$$\int_{-1}^{1} \mathcal{W}(x) f(x)\, dx \approx \sum_{i=1}^{N} w_i f(\xi_i), \tag{15}$$

start by choosing a set of orthogonal polynomials $\mathcal{T}_i(x)$ with the same weight function $\mathcal{W}(x)$, and expressed in terms of recursion rules [17] so that:

$$\mathcal{T}_{-1}(x) \equiv 0, \qquad \mathcal{T}_0(x) \equiv 1,$$
$$\mathcal{T}_{i+1}(x) \equiv (x - \delta_{i+1})\mathcal{T}_i(x) - \gamma_{i+1}^2 \mathcal{T}_{i-1}(x). \tag{16}$$

Take these $\delta_i$ and $\gamma_i$ coefficients, and construct a tridiagonal symmetric matrix:

$$\begin{bmatrix} \delta_1 & \gamma_2 & & 0 \\ \gamma_2 & \delta_2 & \ddots & \\ & \ddots & \ddots & \gamma_N \\ 0 & & \gamma_N & \delta_N \end{bmatrix} \tag{17}$$

The eigenvalues of this matrix, which are also the roots of the polynomial $\mathcal{T}_N(x)$, are the quadrature rule's positions $\xi_i$. The square of the first coefficient of the $i^{\text{th}}$ eigenvector is the quadrature weight $w_i$. The eigenvectors and eigenvalues for tridiagonal symmetric matrices can be found using QR factorization [17].

To create the Gauss-Legendre rule of order $N$, exact for polynomials up to degree $2N-1$, the $\gamma_i$ and $\delta_i$ coefficients are [22]:

$$\delta_{i+1} = 0, \qquad \gamma_{i+1} = \sqrt{\frac{i^2}{(2i+1)(2i-1)}}, \tag{18}$$

and for general Jacobi polynomials $P_i^{(\alpha,\beta)}$:

$$\delta_{i+1} = \frac{(\alpha+\beta)(\beta-\alpha)}{(2i+\alpha+\beta+2)(2i+\alpha+\beta)},$$
$$\gamma_{i+1} = \sqrt{\frac{4(i+\alpha)(i+\beta)i(\alpha+\beta+i)}{(\alpha+\beta+2i)^2(\alpha+\beta+2i+1)(\alpha+\beta+2i-1)}}. \tag{19}$$

When using these quadrature rules to project a function into a basis set using (6), it is important to use a sufficiently accurate quadrature rule. If a one-dimensional polynomial basis set includes terms up to order $n$, the projection integral (6) must be accurate up to order $2n$—since the function is represented as a polynomial of order $n$, the projection integrand will be a polynomial of order $2n$. Therefore, a one-dimensional Gaussian quadrature rule must have at least $N+1$ sample points to integrate accurately [6].

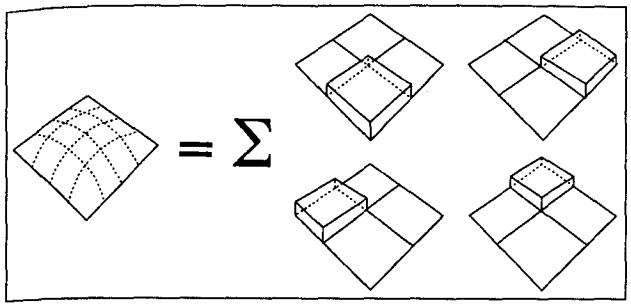


Figure 1: Conventional radiosity methods approximate a surface's radiosity by meshing it into a large number of constant intensity patches. Radiosity is represented by height above the surface.

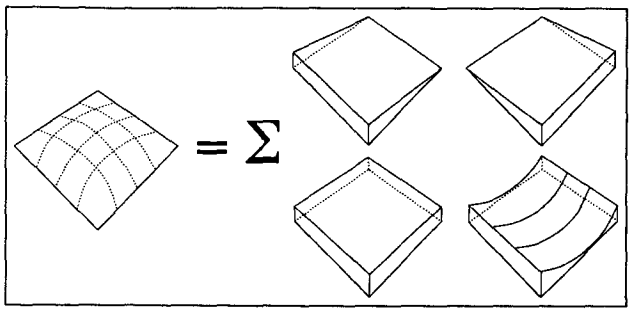


Figure 2: Higher-order radiosity approximates a surface's radiosity by dividing it into several different smooth functions. These smooth functions are scaled and combined to approximate the original radiosity distribution.

# 4 Non-Constant Radiosity

Consider the effect of meshing a single surface into constant radiosity patches (Figure 1). Although the radiosity is smooth on individual patches, combinations describe a discontinuous, stair-step radiosity function. To produce a smooth, consistent solution, we need to formulate radiosity in terms of smooth functions across an entire surface, instead of disjoint patches on parts of a surface.

Figure 2 shows a hypothetical decomposition of a radiosity function. Constant, linear, and higher-order functions are combined to produce a smooth approximation to the radiosity function. If the radiosity of every surface were represented by a combination of these functions, the radiosity problem would reduce to finding their relative weights.

To properly compute these proportions, we use a radiosity formulation based on a linear combination of orthonormal basis functions $\{\mathcal{T}_l(s,t)\}$. Instead of radiosity values, we use radiosity coefficients $\{B^l\}$—the relative contribution of each function $\mathcal{T}_l(s,t)$. The full radiosity distribution on a surface becomes the function

$$B_{\text{total}}(s,t) = \sum_l B^l \mathcal{T}_l(s,t). \tag{20}$$

Functions on different surfaces must interact in a manner analogous to the way conventional patches interact through form factors. Just as conventional radiosity uses form factors to describe the interaction between patches, here the kernel function $K_{ij}(s,t,u,v)$ from (1) details how energy is transferred between functions on different surfaces. When two constant functions on different surfaces interact, the kernel function interaction is equivalent to a classical form factor. Other kernel functions describe higher-order interactions.

## 4.1 The Galerkin Method

Given an orthonormal basis set, the Galerkin technique finds a good [6] fit to the integral equation's solution within that set. Heckbert [10, 11] suggested that the Galerkin method and meshing could be used to solve the radiosity integral equation in a plane. This and subsequent sections demonstrate how it can be applied to three-dimensional radiosity.

Starting with the parametric radiosity equation (1),

$$B_i(s,t) = E_i(s,t) + \sum_j \int\int K_{ij}(s,t,u,v) B_j(u,v)\,du\,dv, \tag{21}$$

expand the $B_j(u,v)$ term inside the integral in terms of the basis set $\{\mathcal{T}_l(u,v)\}$ using (7). The $B_j^l$ coefficient can be moved outside of the integral, and the summations over $j$ and $l$ can be combined to produce the equation

$$B_i(s,t) = E_i(s,t) + \sum_{j,l} B_j^l \int\int K_{ij}(s,t,u,v)\mathcal{T}_l(u,v)\,du\,dv. \tag{22}$$

Now, take the inner product of both sides with the $k$th basis set function $\mathcal{T}_k(s,t)$. Using bilinearity and the relation described in (6),

$$B_i^k = E_i^k + \sum_{j,l} B_j^l \left\langle \int\int K_{ij}(s,t,u,v)\mathcal{T}_l(u,v)\,du\,dv \,\middle|\, \mathcal{T}_k(s,t) \right\rangle_{\mathcal{W}}. \tag{23}$$

The inner product now depends only on known information; the kernel function $K_{ij}$ is a function of the environment, and $\{\mathcal{T}_l(u,v)\}$ is a precomputed basis set. The result of that inner product is denoted $K_{ij}^{kl}$, the kernel matrix. Evaluating this inner product is the most difficult part of a radiosity solution, requiring four integrations—two explicit, and two in the inner product. However, once the kernel matrix has been computed for each value of $i,j,k$, and $l$, the radiosity equation can be written as a matrix equation,

$$B_i^k - E_i^k = \sum_{j,l} B_j^l K_{ij}^{kl}. \tag{24}$$

Just as a conventional form factor matrix relates constant radiosities on different elements, the kernel matrix relates radiosity functions across different surfaces. The $K_{ij}^{kl}$, $B_i^k$ and $E_i^k$ values are analogous to classical form factors, patch radiosities, and emittances, respectively. However, each of these coefficients refers to some function representing part of the distribution of radiosity across a surface, as opposed to a constant value across a surface. Note also that even though (24) is written in terms of four indices, since the surface indices $i,j$ and function indices $k,l$ are independent of each other, (24) is still a two-dimensional matrix equation.

This equation can be solved using any standard matrix technique, such as Gaussian elimination, or progressive refinement techniques [4]. Cohen *et al*'s progressive refinement technique requires slight modification with Galerkin radiosity, because the radiosity coefficients $B_j^l$ may have negative values. These negative values do not indicate negative energies; they are a weight applied to the basis function. The shooting order should be based on unshot magnitude:

$$M_j^l = \|B_j^l\| \int\int |\mathcal{T}_l(u,v)|\,dA_j(u,v)\,du\,dv. \tag{25}$$

## 4.2 Edge Singularities

Near the common edge of two non-coplanar surfaces, the double-differential form factor approaches infinity as a pole of order two[22]. Although the function still has a finite integral, the singularity can cause serious convergence problems. If the singularity is ignored, Galerkin solution methods converge extremely slowly for a mediocre basis set, and may fail entirely for a bad basis set.

To insure reasonable convergence, the basis set must compensate for the singularity. In (23), the singularity appears inside the quadruple integral that generates $K_{ij}^{kl}$. This integral also includes the inner product weight function $\mathcal{W}(s,t)$. If the weight function $\mathcal{W}$ is chosen with zeroes of sufficiently high multiplicity where the kernel function $K_{ij}$ goes to infinity, the two features can cancel and the integral will converge. Since the kernel singularity grows as a pole of order two, the weight function should have zeroes of multiplicity two at its edges. The Jacobi polynomial sets $\mathcal{P}^{(0,2)}$ and $\mathcal{P}^{(2,0)}$ (see section 3.2), have appropriate weight functions.

By using a hybrid Galerkin method, the edge singularities are cancelled. For non-singular light transfers between surfaces that do not touch, a Legendre basis set is used. For the few transfers that are singular, a basis set of Jacobi polynomials is used, either $\mathcal{P}^{(0,2)}$ or $\mathcal{P}^{(2,0)}$ depending on the singularity's location. After computing the $K_{ij}^{kl}$ coefficients and the associated radiosity transferred in a singular shot, project this polynomial function in $s$ and $t$ is back into a Legendre basis set for storage. An empty box computed with this hybrid method is shown in Figure 6.

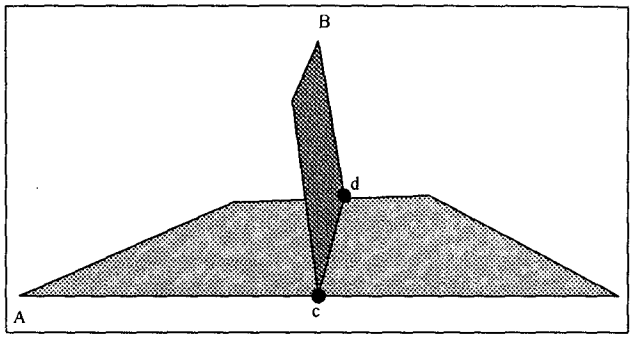



Figure 3: Surfaces A and B meet in a T-intersection; surface B divides surface A into two regions along the line $\overline{cd}$.

Because singularities can be produced at any non-parallel intersection, geometries with T-intersections (the three-dimensional analog to Heckbert's T-corners [11]) like those in Figure 3 make singularities difficult to handle. Although such geometries could be handled by using a basis set with a two-dimensional weight function containing a double zero in the middle of the surface along the curve of intersection, constructing such basis sets would be relatively difficult even for polygonal surfaces. More effective approaches include subdividing significant T-intersections into distinct singular intersections, or ignoring the singularity altogether when possible.

### 4.3 Computing the Energy Transfers

In order to generate radiosity solutions, entries in the kernel matrix (24) must be computed. Each entry is computed by applying a quadrature rule (15) to approximate the inner product of (23) for particular values of $i, j, k$ and $l$. For non-singular energy transfers—those between surfaces that do not share a common edge—the inner product weight function is unity, and the quadrature rule is a Gauss-Legendre quadrature rule constructed with (17) and (18). If the Gauss-Legendre quadrature points and weights are denoted $p^l_\alpha$ and $w^l_\alpha$ respectively, then each kernel matrix element is approximated by the summation,

$$K^{kl}_{ij} \approx \sum_{\alpha,\beta,\gamma,\delta} K_{ij}(p^l_\alpha, p^l_\beta, p^l_\gamma, p^l_\delta)\mathcal{T}^l_k(p^l_\alpha, p^l_\beta)\mathcal{T}^l_l(p^l_\gamma, p^l_\delta)w^l_\alpha w^l_\beta w^l_\gamma w^l_\delta. \tag{26}$$

Since each kernel sample requires a full intersection test with the environment, caching samples $K_{ij}(p^l_\alpha, p^l_\beta, p^l_\gamma, p^l_\delta)$ or results of the associated intersection tests can save significant CPU time, at the expense of additional storage.

Singular energy transfers—those between two surfaces that meet in a singular edge—require additional processing. Kernel matrix elements for a singular transfer are computed using a Gauss-Jacobi quadrature rule matching the Jacobi basis set. Once the quadrature rule's points $p^l_\alpha$ and weights $w^l_\alpha$ have been computed using (17) and (19), the kernel matrix elements can be computed by the summation,

$$K'^{kl}_{ij} \approx \sum_{\alpha,\beta,\gamma,\delta} K_{ij}(p^l_\alpha, p^l_\beta, p^l_\gamma, p^l_\delta)\mathcal{T}^l_k(p^l_\alpha, p^l_\beta)\mathcal{T}^l_l(p^l_\gamma, p^l_\delta)w^l_\alpha w^l_\beta w^l_\gamma w^l_\delta. \tag{27}$$

When this weighted sum is evaluated, the resulting matrix entries $K'^{kl}_{ij}$ are in terms of a Jacobi basis set, while the $E^k_i$ and $B^k_i$ values in storage are in terms of a Legendre basis set. The Jacobi matrix entries must be projected into the Legendre basis set before they can be combined with the other coefficients. Since the $K'^{kl}_{ij}$ coefficients are simply leading multipliers for polynomials, they can be converted from Jacobi coefficients to Legendre coefficients by expanding the Jacobi coefficients into an ordinary polynomial in $s$ and $t$, and then converting that polynomial back into a sum of Legendre polynomials.

## 5 Shadow Discontinuities

As with any illumination algorithm, dealing with occlusions presents a special challenge. The easiest way to deal with shadows is to let the basis functions find a best fit. Unfortunately, shadows produce sharp edges which cannot be expressed in terms of a few polynomials. Attempting to model such edges with a small polynomial basis set produces a fuzzy shadow with ripples around it—the Gibbs behavior visible in Figure 7.

Shadow edges come from discontinuities in the radiosity function [10]. One way to remove these discontinuities is to mesh the environment along curves of discontinuity [13, 12], a process which eliminates the occlusion difficulties of Galerkin radiosity. Unfortunately, discontinuity meshing methods magnify the number of surfaces in the scene, vastly increasing computation time. Even though shadows are primarily an interaction between a light source and a receiving surface, subdividing the receiving surface to produce accurate shadows complicates interactions with the rest of the environment.

### 5.1 Shadow Masking

To smooth the shadow discontinuities out of the radiosity distribution seen by the Galerkin method, we propose using a *shadow mask* approximation. For the majority of emitter-receiver pairs, where shadows do not have a high-frequency effect on the solution, traditional visibility calculations can be used. However, for a select group of emitter-receiver pairs, we move the visibility term $\mathrm{VIS}_{ij}(s, t, u, v)$ out of the kernel function and integral in equations (2) and (1), and replace it with a normalized shadow mask function $M_{i\leftarrow j}(s, t)$,

$$M_{i\leftarrow j}(s, t) = \frac{\iint \mathrm{VIS}_{ij}(s, t, u, v)\,du\,dv}{\iint du\,dv}. \tag{28}$$

This function approximates the fraction of the light originating from emitter $j$ that arrives at a particular location on receiving surface $i$. The shadow mask is one where the emitter is fully visible, zero where the emitter is fully occluded, and takes on intermediate values when the light is partially occluded. It is essentially a texture map for painting the shadow onto the receiving surface.

During the radiosity pass, if the energy transfer from emitter $j$ to receiver $i$ involves a shadow mask, the radiosity is accumulated without visibility calculations in the special coefficients $B^k_{i\leftarrow j}$ instead of $B^k_i$. When light is re-emitted from surface $i$'s basis functions, the kernel samples are multiplied by the shadow mask across surface $i$, restoring some of the occlusion information. The radiosity across a surface, $B_i(s, t)$, becomes the combination of ordinary Galerkin basis functions and shadow mask-weighted basis functions. If $h$ represents all light sources casting a shadow on surface $i$,

$$B_i(s, t) = \sum_k B^k_i \mathcal{T}_k(s, t) + \sum_{h,k} M_{i\leftarrow h}(s, t)B^k_{i\leftarrow h}\mathcal{T}_k(s, t). \tag{29}$$

By using coefficients $B^k_{i\leftarrow h}$, radiosity in the shadow mask is maintained separately from radiosity coming from other parts of the environment. When a receiving surface has shadow masks associated with it, every surface interacts either with a shadow mask, or with the standard surface description—not both.

In this implementation, shadow masks were computed from equation (28) using multiple point-to-point visibility samples regularly spaced in the parametric dimensions. Values of $M_{i\leftarrow j}(s, t)$ were computed by linear interpolation between these sample points. Shadow mask samples could conceivably be taken along lines of discontinuity, or in some more complicated non-regular structure to improve efficiency or accuracy.

In all environments tested, even accounting for the time spent constructing shadow masks, the time required to compute a radiosity solution using shadow masks was significantly smaller than that for a full discontinuity mesh. For the simple environment in Figure 8, the shadow mask was a regular 40 by 40 grid of sample points on the floor. Without Gibbs phenomena to transfer energy into higher order basis functions as in Figure 7, the radiosity pass actually required fewer shots and less time to converge than the non-shadow masked version.

Since a shadow mask only adds one surface to the rows (but not the columns) of the radiosity matrix for each associated emitter-receiver pair in the environment, shadow masks add relatively little to radiosity solution time compared to discontinuity meshing methods. Shadow masks can be precomputed for portions of the environment where shadow details are expected to be significant. Furthermore, since shadow masks are defined in parametric space, a single implementation can cast shadows to and from any type of surface.

Unfortunately, shadow masks also have significant disadvantages. By moving the visibility term out of the radiosity equation's integral, any correlation between the emitter's light distribution and the shape of the occluding

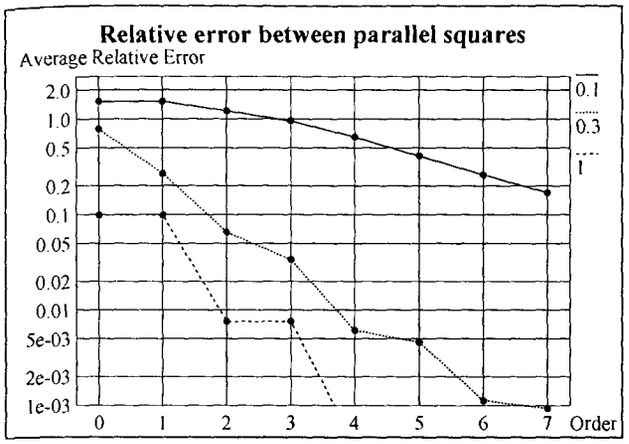



Figure 4: Average relative error for Galerkin radiosity transfers between two parallel squares of width $l$, distance $l$, $0.3l$, and $0.1l$ apart.

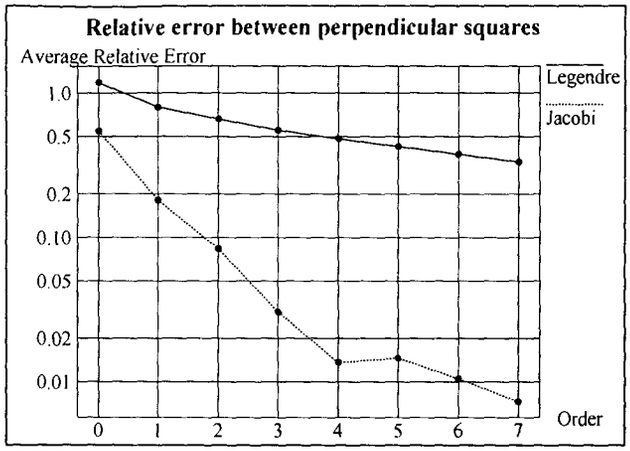



Figure 5: Average relative error for Galerkin radiosity transfers between two adjacent perpendicular squares at a corner. The Jacobi basis set computation produces significantly less error for this singular transfer than the Legendre basis set.

surface is destroyed. Because of this, a shadow mask solution will not converge to the "true" solution. Placement of the shadow masks is currently left to the user; some criteria is needed for determining whether or not to use shadow masks. Although shadow masks can be stored in a simple grid fashion, such a grid may not produce the best results when used with a particular quadrature rule. Finally, any attempt at increasing the spatial accuracy of shadow masks can duplicate many of the difficulties of storing a mesh on a surface.

However, there is a significant difference between increasing the density of a mesh and increasing the density of a shadow mask—every element of a mesh becomes another surface interacting with the environment, while even the most complex shadow mask is still only part of one surface. Shadow masks do facilitate the generation of approximate radiosity solutions with the Galerkin method, by smoothing out shadow discontinuities. Further research may suggest ways to avoid their associated disadvantages.

# 6 Sources of Error

The principal cause of error is not using a large enough basis set; as more basis functions are used, the Galerkin method produces a more accurate solution. Improper treatment of shadows can also cause significant inaccuracies in a Galerkin solution; if shadow discontinuities are ignored, they produce Gibbs-behavior ripples, and if shadow masks are used, they introduce approximation error. Additional errors come from inaccuracies in the quadrature rule used to evaluate kernel matrix integrals, or from approximate matrix solution techniques like progressive radiosity.

In this section, error analysis is provided at two different scales. At the level of surface-to-surface energy transfer, Galerkin radiosity results are examined for a few simple cases where comparison with an exact analytical solution is possible. At the level of picture generation, conventional and Galerkin radiosity solutions are compared for a standard radiosity test environment.

## 6.1 Energy Transfer Error

For the simple environment used by Sparrow's variational radiosity solution [18], a fourth-order solution produced a relative error of less than one percent. Using the method of this paper, error computations for a single energy transfer between parallel and perpendicular squares produce similar levels of accuracy. All comparisons in this section are made against an analytic solution using the formulation of Sparrow and Cess [19]. The relative error metric used is

$$E = \left\langle \frac{|B_{\text{Galerkin}}(s,t) - B_{\text{exact}}(s,t)|}{B_{\text{exact}}(s,t)} \right\rangle_{s,t}, \qquad (30)$$

where the error is evaluated on a 500 by 500 grid of sample points on the receiving surface. Transmitting and receiving squares are the same size, and are computed at the same solution order (although for numeric reasons, this often produces the worst results [22]).

The simplest case is for a radiosity transfer between parallel squares with sides of length $l$, as shown in Figure 4. With the distance between the squares equal to their size, the fourth-order transfer gives a relative error of only 0.04%. Since the accuracy increases as the squares are placed farther away with respect to their size, a fourth or fifth order transfer should produce reasonable accuracy for computer graphics applications.

As the squares move closer, the relative error becomes much higher. When the distance between the squares is reduced to one-tenth their width, even a seventh order solution produces an average relative error of 16.9%. Unfortunately, when surfaces are extremely close relative to their size, to achieve reasonable accuracy, the surfaces must still be subdivided.

Experiments with perpendicular rectangles (Figure 5) illustrate the importance of proper treatment of singularities. Using a non-singular Legendre basis set to compute the energy transfer produced large error even at high order; a seventh order transfer produced a relative error of 33.2%. Using a singular Jacobi basis set, results are much more accurate. Fourth and fifth order transfers both produce about 1.4% relative error.

## 6.2 Comparison with Conventional Radiosity

Lischinski and Tampieri provided a reference solution to a two-box radiosity environment. This solution was computed using the discontinuity meshing techniques of [13], with adaptive integration using Wallace [21] point-to-point form factors. Individual triangles in the mesh were treated consistently as quadratic elements, limiting error in their reference solution to a few meshing artifacts, visible near the corners of the top wall. This solution is used as a comparison baseline for images generated with Galerkin radiosity.

The *order* of a solution is the highest total polynomial order used as a basis function for the solution. A zeroth-order solution would be equivalent to a conventional radiosity solution, with radiosity constant across a surface. A first-order solution would have linear radiosity variation, a second-order solution would have quadratic variation, and so on. Note that a basis function's order depends on the sum of the highest orders used in each dimension. Different surfaces in a solution can be different orders; a high order basis could be used for large, visible areas, while a low order basis may be sufficient for shadowed regions.

Figure 9 shows pictures of a simple test environment solved with different solution orders. Shadows were created using a 20 by 20 grid shadow mask. Notice how the floor appears smoother at higher order, even though no post-processing interpolation was used to smooth the meshing. Meshing (Figure 10) was only performed to eliminate T-intersections; the three boxes and light were meshed to 26 polygons.

Figure 11 shows difference images between the different order Galerkin solutions of the test environment and the reference solution. These images were created by converting the Galerkin and conventional radiosity solution images to black and white, and then computing the absolute value of the intensity difference at each pixel. Dark regions of these images are where the two solutions agree; bright regions are where the two solutions differ. The Galerkin image was translated slightly before comparison, so that outlines of the boxes and floors would be visible in the difference images. As would be expected, the difference images get progressively darker as the solution

| Figure | Description | CPU time | Shots |
|---|---|---|---|
| 6 | Empty box | 5.4s | 7 |
| 7 | Box with single occluder | 59.7 s | 33 |
| 8 | Shadow masked box | 42.8 s | 22 |
| 12 | Clay teapot | 6.71 h | 53 |

Table 1: Timings for the shadow generation and radiosity pass combined for various pictures computed with this algorithm. All timings are for an HP 9000/720 workstation.

order increases; the regions where the solution is least accurate tend to be near singular edges.

In this particular test case, the method of [13] took about the same amount of time as the highest-order Galerkin solution. However, the Galerkin method only required 6.5 Megabytes of memory, compared to 75 Megabytes for a more conventional, meshing approach. For all environments tested in this paper, Galerkin and conventional radiosity methods tend to take about the same amount of time to produce equivalent pictures. However, the Galerkin radiosity technique's lower memory usage is maintained in more complex environments.

# 7 Results

The radiosity solution computed by this method is a list of basis set expansion coefficients $B_i^k$ for each surface $i$ and basis function $k$. The actual radiance at a given point $(s, t)$ on surface $i$ is recovered from these coefficients using (7). If shadow masks were used, the additional coefficients $B_{ih}^k$ are incorporated with (29).

In this implementation, environments are rendered by a simple ray-tracing/scanline technique. When a ray intersects a surface, that intersection point is projected back into the surface's parametric space, and the result is used to compute a radiosity value for the appropriate pixel.

## 7.1 Curved Surfaces

Curved surfaces can be easily incorporated into Galerkin radiosity; the kernel term's form factor as expressed in (3), includes surface normals explicitly. To implement curved surfaces, replace the traditional constant surface normal value with a function, computable at any parametric location. Sample pictures are shown with bicubic patches (Figure 12) and other curved surfaces (Figure 13). The Galerkin radiosity method was applied directly to these environments; the curved surfaces were *not* tiled.

For comparison purposes, the teapot environment was also computed using a commercially-available radiosity package [16]. This package uses the point-sampling algorithm of Wallace *et al.* [21] to compute form factors, but does not perform adaptive meshing. Since this radiosity package cannot use bicubic patches directly, each of the teapot's patches were tessellated with a 20 by 20 grid. The radiosity solution took 6.2 hours, and over 54 megabytes of memory to compute; this simple forty-patch scene became a relatively complex, eight thousand polygon environment. In contrast, the Galerkin computation took 6.7 hours, but only required 3.9 megabytes of memory during the radiosity pass. Over 90% of this computation time was spent computing visibility samples.

The significant point of this comparison is that given approximately equivalent amounts of time to produce a solution, conventional and Galerkin methods produced similar results. But since Galerkin methods needn't maintain the detailed geometric structure of a mesh, they use significantly less memory.

## 7.2 Parallelization

Galerkin radiosity environments are not meshed into large, complicated data structures, so it is relatively easy to maintain copies of the environment in memory on multiple hosts. Since each individual light transfer between two surfaces depends only on the geometry and shadow masks, they can be computed on independent machines. Such a parallelization scheme was implemented, running concurrently on DECstations, HP 700's and 800's, and on multiple processors of an Apollo DN10000. The image of Figure 13 was computed in parallel on five DECstations and five HP 700's as a background process over two days.

# 8 Conclusions

Using the Galerkin method, this paper has presented an alternative method for producing radiosity simulations. Through special treatment of the radiosity equation's singularities and discontinuities, the Galerkin technique's dependency on smooth kernels can be overcome. Although the resulting pictures are similar to those produced by conventional radiosity methods, the method used to generate them is fundamentally different:

- The radiosity across a surface is represented as a smoothly varying function. Pictures are rendered directly from the radiosity solution, without an additional blurring step.
- Adequately sampled curved surfaces can be used directly. Since curved surfaces don't need to be tessellated, they can be incorporated into a scene cheaply. Issues of approximating a surface's geometry and approximating a surface's radiosity are separated.
- Energy transfer error analysis shows that meshing is only essential when two surfaces are extremely close to each other relative to their size. Meshing is *not* needed to model variations in intensity across a surface.
- By using shadow masks, the local details of shadow edge generation are separated from the global issues of energy balance.

# 9 Deficiencies of the Method

As with any rendering algorithm, Galerkin radiosity has its own particular disadvantages. Problems with the treatment of shadows are the most significant; if important shadows are missed, a solution will contain significant Gibbs ringing behavior. It may not always be easy to determine ahead of time where detailed shadow masking or meshing will be necessary, possibly requiring multiple solution attempts before all shadows are properly accounted for.

Shadow masking is only a rough approximation to the true occlusion behavior; it eliminates any correlation between variations in light source intensity and the intensity of the shadow, virtually returning to the Constant Radiosity Assumption for a shadow's light source. Furthermore, the distribution of the shadow mask sample points can have a significant impact on the accuracy of the shadow they generate.

Higher order methods also have the potential to be computationally expensive. Because of the $(N + 1)^4$ samples required to transfer radiosity between surfaces of order $N$, radiosity calculations can become extremely expensive if too high a solution order is used. In general, an order of 4 or 5 is sufficient, but self-intersecting or highly curved surfaces may require a higher-order solution.

The method does not mathematically guarantee radiosity continuity between adjacent coplanar surfaces. However, such surfaces appear much less frequently in a shadow masked environment than in a meshed environment. If such continuity is needed, it can be generated by using a high enough order on the adjacent surfaces that the error on each surface is reduced until their radiosity values along their common boundaries match visibly—usually 8 or 9 in our tests.

Finding all the singularities in a system can also be difficult. Environments usually have a large number of T-intersections (see Figure 3), each of which could require a separate meshing step. Although T-intersections can often be ignored, there's always a risk that the ignored singularity will cause the solution to fail to converge, requiring recomputation.

# 10 Future Work

Shadow masks are currently implemented using bilinear interpolation on a simple grid of sample points. Many more efficient sampling schemes are possible, such as adaptive quadtrees, or some method that directly computes the location of shadow discontinuities. Additionally, some method should be developed for automatically determining where shadow masks are needed. Some generalization of shadow masks is needed to account for variations in light source intensity.

A means for enforcing continuity between adjacent surfaces, possibly by using some sort of modified patch/element method could lower the required solution order, and significantly accelerate the algorithm when such surfaces are present. A method combining adaptive meshing and a low order Galerkin solution might produce reasonable images rapidly. Extending

Figure 6: An empty box computed with up to fourth order polynomials, or 15 basis functions across each surface. On an HP 9000/720, the radiosity pass took 5.4 CPU seconds.

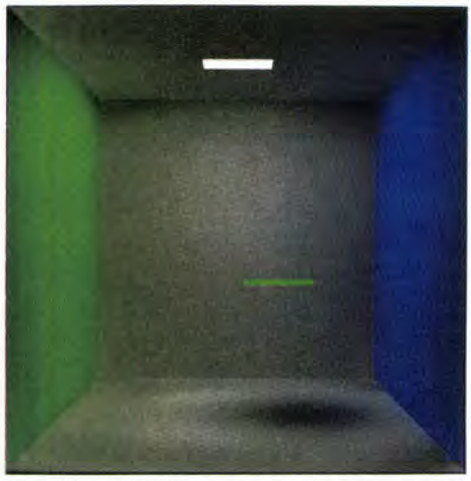


Figure 7: A box with an occluding rectangle computed with a fourth order basis on all surfaces except the floor, which has an eighth order basis. The ripples on the floor of the box appear because shadow discontinuities cannot be accurately described by a low frequency Galerkin basis set.

Hanrahan's hierarchical multigridding technique [9] to higher order functions could produce a means to do this. Some method must also be found to automatically determine an appropriate solution order for each surface, instead of the current area-based heuristic.

The method of this paper uses a Legendre basis set for non-singular energy transfers. Galerkin methods frequently use a Chebyshev basis; by examining the relative accuracy of different basis sets, it may be possible to find a better basis set for the radiosity problem.

This paper is only a first attempt at applying higher order solution methods to the radiosity problem. Much work remains to fully integrate this approach into the general framework of global illumination and radiosity.

## Acknowledgements

The Program of Computer Graphics at Cornell is one of five sites of the NSF/DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (Grant # ASC-8920219). Thanks to HP/Apollo and DEC for equipment donations. This research was partially conducted under a National Science Foundation grant entitled "Interactive Input and Display Techniques." The author was funded by a National Science Foundation Graduate Student Fellowship, a Dabnicorp Computer Graphics award, and by a Cornell University Sage Fellowship. I would like to thank Don Greenberg for providing insightful comments as to what was needed to write up this research, Dani Lischinski and Filippo Tampieri for providing the reference solution of Section 6, and especially Jim Arvo for his advice in developing the research and this text. I'd also like to thank everyone at the Cornell Program of Computer Graphics, Rhythm and Hues, Inc., and the Caltech Graphics Lab for their support and encouragement.

## References

[1] Daniel Baum, Holly Rushmeier, and James Winget, "Improved Radiosity Solutions Through the Use of Analytically Determined Form-Factors", *Computer Graphics*, 23(3), pp. 325-334, 1989.

[2] A. T. Campbell, III and Donald Fussell, "Adaptive Mesh Generation for Global Diffuse Illumination", *Computer Graphics*, 24(4), pp. 155-164, 1990.

[3] Michael Cohen and Donald Greenberg, "The Hemi-Cube: A Radiosity Solution For Complex Environments", *Computer Graphics*, 19(3), 1985, pp. 31-40.

[4] Michael Cohen, Shenchang Chen, John Wallace, Donald Greenberg, "A Progressive Refinement Approach to Fast Radiosity Image Generation", *Computer Graphics*, 22(4), 1988, pp. 75-84.

[5] Philip Davis, *Interpolation and Approximation*, Blaisdell, New York, 1963.

[6] L. M. Delves and J. L. Mohamed, *Computational Methods for Integral Equations*, Cambridge University Press, New York, 1985.

[7] Cindy Goral, Kenneth Torrance, Donald Greenberg, and Bennett Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces", *Computer Graphics*, 18(3), July 1984, pp. 213-222.

[8] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, 4th edition, Academic Press, Inc., New York, 1965.

[9] Pat Hanrahan, David Salzman, and Larry Aupperle, "A Rapid Hierarchical Radiosity Algorithm", *Computer Graphics*, 25(4), pp. 197-206, 1991.

[10] Paul Heckbert, *Simulating Global Illumination Using Adaptive Meshing*, Report No. UCB/CSD 91/636, University of California, Berkeley, 1991.

[11] Paul Heckbert and James Winget, *Finite Element Methods for Global Illumination*, Report No. UCB/CSD 91/643, University of California, Berkeley, 1991.

[12] Paul Heckbert, "Discontinuity Meshing for Radiosity", *Third Eurographics Worshop on Rendering*, Bristol, UK, May 1992.

[13] Dani Lischinski, Filippo Tampieri, and Donald Greenberg, "Discontinuity Meshing for Accurate Radiosity", IEEE CG&A, 12(6), Nov. 1992.

[14] Nelson Max and Michael Allison, "Linear Radiosity Approximations using Vertex-to-Vertex Form Factors", *Graphics Gems III*, Academic Press, 1992, p. 319

[15] Tomoyuki Nishita and Eihachiro Nakamae, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection", *Computer Graphics*, 19(3), 1985, pp. 23-30.

[16] *Starbase Radiosity and Ray Tracing Programmer's Manual*, Hewlett Packard Co., USA, 1990.

[17] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[18] E. M. Sparrow, "Application of Variational Methods to Radiation Heat-Transfer Calculations", *Journal of Heat Transfer*, November 1960, pp. 375-380.

[19] E. M. Sparrow and R. D. Cess, *Radiation Heat Transfer— Augmented Edition*, Hemisphere Publishing Corp., Washington, 1978.

[20] Filippo Tampieri and Dani Lischinski, "The Constant Radiosity Assumption Syndrome", in the Proceedings of the Second Eurographics Workshop on Rendering, Barcelona, 1991.

[21] John Wallace, Kells Elmquist, Eric Haines, "A Ray Tracing Algorithm for Progressive Radiosity", *Computer Graphics*, 23(3), 1989, pp. 315-324.

[22] Harold Zatz, *Galerkin Radiosity: A Higher Order Solution Method for Global Illumination*, Master's Thesis, Cornell University, Ithaca, New York, 1992.

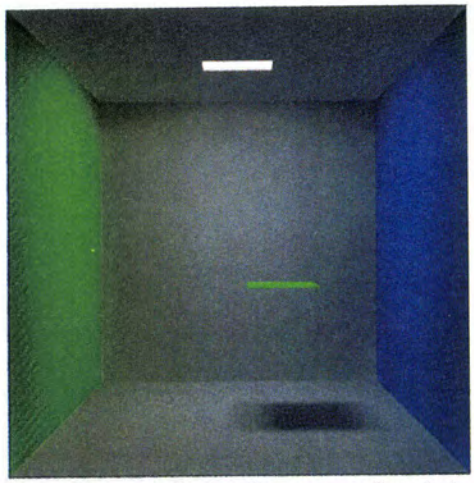


Figure 8: A box with the transfer from light source to floor shadow masked, computed to fourth order on all surfaces except the floor and light source, which are computed to eighth order.

Figure 9: Solving the two box test environment, with solution orders zero, one, three, and seven.

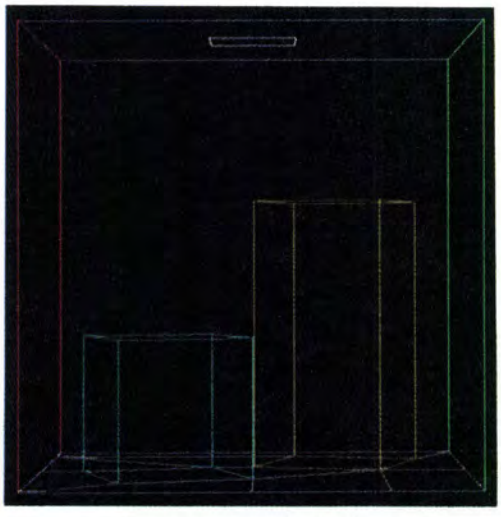


Figure 10: Mesh used for Figure 9. Only the floor has been meshed, to eliminate T-intersections. The boxes, walls, and ceilings were each solved using functions over the entire surface.

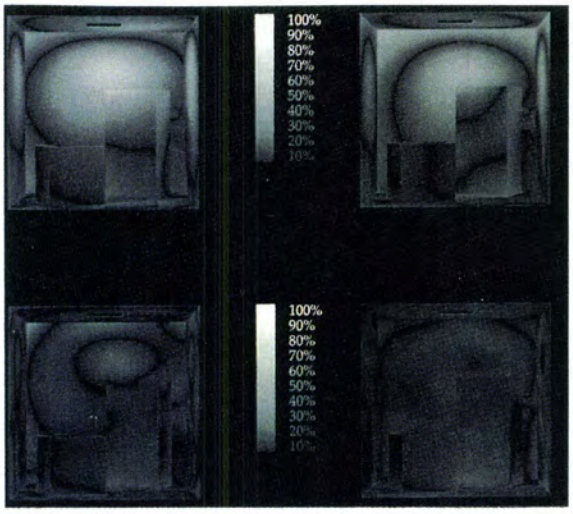



Figure 11: Difference images between the two box test environment and the reference solution, with solution orders zero, one, three, and seven.

Figure 12: The radiosity function across the clay teapot was solved directly, with a sixth-order basis set for each bicubic patch. The floor, walls, and portions of the teapot received shadow masks from the four lights.

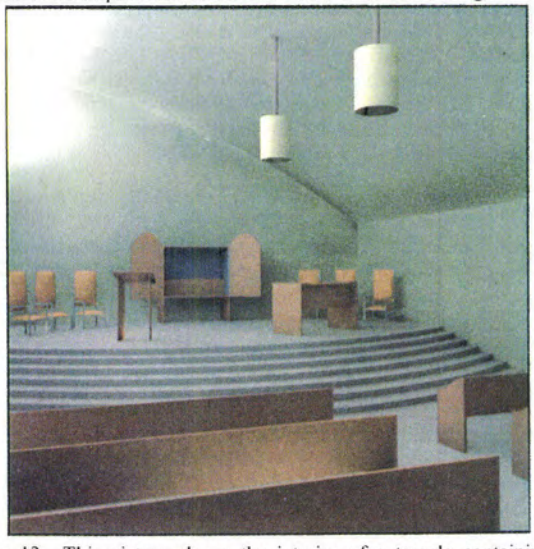


Figure 13: This picture shows the interior of a temple containing 607 parametrically defined, non-meshed surfaces, including polygons, bicubic patches, cylinders, and cubic extrusions. Most surfaces were computed with a fourth or fifth order solution, except for the walls and roof at seventh order, and the cylindrical light fixtures at thirteenth order.

# Wavelet Radiosity

Steven J. Gortler Peter Schröder Michael F. Cohen Pat Hanrahan

Department of Computer Science
Princeton University

## Abstract

Radiosity methods have been shown to be an effective means to solve the global illumination problem in Lambertian diffuse environments. These methods approximate the radiosity integral equation by projecting the unknown radiosity function into a set of basis functions with limited support resulting in a set of $n$ linear equations where $n$ is the number of discrete elements in the scene. Classical radiosity methods required the evaluation of $n^2$ interaction coefficients. Efforts to reduce the number of required coefficients without compromising error bounds have focused on raising the order of the basis functions, meshing, accounting for discontinuities, and on developing hierarchical approaches, which have been shown to reduce the required interactions to $O(n)$.

In this paper we show that the hierarchical radiosity formulation is an instance of a more general set of methods based on *wavelet* theory. This general framework offers a unified view of both higher order element approaches to radiosity and the hierarchical radiosity methods. After a discussion of the relevant theory, we discuss a new set of linear time hierarchical algorithms based on wavelets such as the multiwavelet family and a flatlet basis which we introduce. Initial results of experimentation with these basis sets are demonstrated and discussed.

**CR Categories and Subject Descriptors:** I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism – Radiosity*; G.1.9 [Numerical Analysis]: *Integral Equations – Fredholm equations*.

**Additional Key Words and Phrases:** global illumination, wavelets, hierarchical radiosity.

## 1 Introduction

In computer graphics, radiosity methods have been used to solve the global illumination problem in environments consisting entirely of Lambertian (diffuse) reflectors and emitters. The solution is a radiosity function over the domain of the surfaces in the scene. Classical radiosity [9, 6] (CR), derived from the radiative heat transfer literature, approximates the radiosity function as piecewise constant. An energy balance argument gives rise to a linear system. This system has $n^2$ coefficients called *form factors*. Here $n$ is the number of discrete areas, or *elements*, over which the radiosity function has been assumed to be constant. The form factor describes the fraction of the energy leaving one element and arriving at another. Typically, an iterative algorithm such as Gauss-Seidel iteration [22] or progressive radiosity [5, 10] is used to solve the system of linear equations for the radiosities.

An integral equation called the rendering equation was proposed by Kajiya to model the global illumination problem [14]. He showed that CR is a particular approximation to this equation. By casting the problem in this form, techniques developed for the solution of integral equations [8] can be exploited to solve the radiosity equation.

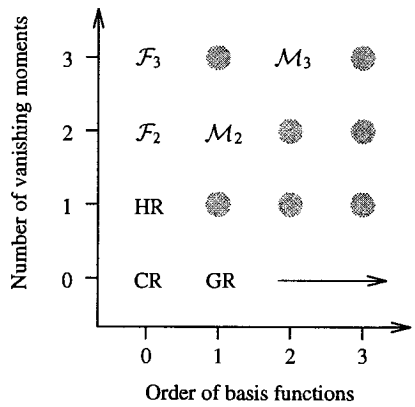



**Figure 1**: The space of projection methods for radiosity.

In particular, Heckbert [12, 13] has demonstrated that the linear system in radiosity can be derived by *projecting* the radiosity integral into a finite dimensional function space. The CR algorithm results from using the *space of piecewise constant functions*, (i.e., projecting the function into a set of constant (or "box") *basis* functions). In general, a function can be projected into any finite dimensional function space. A desirable finite dimensional space is one that can represent the function accurately with as few terms as possible. In his studies, Heckbert considered radiosity functions that are piecewise linear. Zatz [25] has used Legendre polynomials to arrive at solutions that are piecewise polynomial of higher order. Other researchers have explored the use of higher order bases in the mesh construction and reconstruction phases of the algorithm [18] as well as discontinuity meshing [15, 13]. The use of higher order bases, which we will refer to as galerkin radiosity (GR), has been shown to lower the number of basis functions needed to obtain a particular level of accuracy, albeit at a higher cost per basis.

A second avenue of research has attempted to lower the computational complexity of solving the linear system which arises in CR. Hanrahan et al. [11] presented a hierarchical radiosity method (HR) modeled after recent advances in n-body algorithms. HR exploits the fact that neighboring patches in the environment often have similar form factors to distant patches. This reasoning is extended to form a hierarchy of patches, (i.e., a hierarchy of basis functions) in a straightforward manner.

While the methods using higher order bases try to exploit *coherence* in the illumination function, HR tries to exploit the coherence in the form factor itself, more precisely, in the kernel of the radiosity integral. In particular, HR is based on approximating the kernel as a constant function over intervals of varying sizes. In places that the kernel varies slowly, large intervals are used. Where the kernel varies quickly, smaller intervals are needed.

Recently Beylkin et al. [3] made the observation that integral operators satisfying very general smoothness conditions can be approximated to any finite precision with only $O(n)$ coefficients when projected into a wavelet basis instead of the usual $O(n^2)$. This remarkable result means that, in practice, integral equations governed by smooth kernels lead to sparse matrices that can be solved in linear time. Since the radiosity kernel is, in general, a smooth function of the type required by this theorem, wavelet methods can be used to obtain $O(n)$ complexity radiosity algorithms. We call this *wavelet radiosity*.

Hierarchical basis functions have been used before with finite-element methods [24] and applied to problems such as surface interpolation [23]. In those instances, hierarchical basis functions were used to improve the condition number of the matrix. In our context, the hierarchical basis functions (wavelets) are used because many of the resulting matrix coefficients are small enough to be ignored while still allowing for an accurate answer. In some sense we are regarding the matrix as an image on which we are able to perform lossy compression. Coefficients are negligible because over many regions the kernel can be well approximated by a low order polynomial.

The mathematical tools of wavelet analysis provide a general framework offering a unified view of both higher order element approaches to radiosity, and the hierarchical radiosity methods. Figure 1 places earlier algorithms plus the new methods we investigate here into a matrix relating hierarchy versus the order of the underlying basis. CR uses zero order polynomials, while GR uses higher order polynomials (indicated by the arrow). The vertical axis represents the sparseness obtained by exploiting smoothness of some order in the kernel. HR exploits "constant" smoothness in the kernel. Within this context, we recognize HR as a first order wavelet. Higher order wavelets can be used that result in an even sparser matrix. One such family of higher order wavelets is the multiwavelet family of [1] ($\mathcal{M}_{2,3}$ in Figure 1). We will also introduce a new family of wavelets, which we have dubbed *flatlets* ($\mathcal{F}_{2,3}$ in Figure 1) that require only low order quadrature methods while maintaining most of the benefits of other wavelet sets.

This paper proceeds with a review of projection methods for solving integral equations followed by a discussion of recent advances concerning the solution of integral equations using wavelets. Finally we discuss our implementation and report experimental findings. Some of the more technical details of wavelet projections, as well as a detailed analysis of the underlying mathematical framework, are described in [20].

## 2 The Radiosity Integral Equation

If all surfaces and emitters are Lambertian diffuse, the rendering equation can be written as,

$$B(s_1, s_2) = E(s_1, s_2) + \rho(s_1, s_2) \int\int dt_1 dt_2 \frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2} V_{st} B(t_1, t_2) \tag{1}$$

where $B(s_1, s_2)$ gives the radiosity at a point specified by the surface parameters $s_1, s_2$, $E$ the emission, and $\rho$ the reflectivity[1]. The *kernel* of the integral,

$$k(s_1, s_2, t_1, t_2) = \rho(s_1, s_2) \frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2} V_{st}$$

is a function describing the geometric and visibility relationship between two points in the domain; $\theta_s$ and $\theta_t$ are the angles between the surface normals and the line between **s** and **t**; $r_{st}$ is the distance between the two points; $V_{st}$ is 1 if point **s** is visible to point **t** and 0 otherwise.

Over many large intervals, where $r$ is large relative to the size of the patches, the kernel is well represented by a low order polynomial. Notable exceptions include the corners of the environment where $r^2$ goes to 0 and the kernel is singular, and shadow discontinuities where the visibility switches abruptly from 0 to 1.

## 3 Projections

After a short review of function projections we will show how projections can be used to find approximate solutions to integral equations such as the radiosity equation. The ideas presented here can be found in greater detail in [12, 25].

We begin by writing the approximation of a function $B(s)$ in a finite dimensional function space where all functions $\hat{B}(s)$ can be expressed as a linear combination of $n$ *basis functions* $N_i(s)$

$$B(s) \approx \hat{B}(s) = \sum_{i=1}^{n} B_i N_i(s)$$

where the $B_i$ are scalar coefficients with respect to the chosen bases. For example, the space of piecewise constant functions is spanned by a basis of translated "box" functions, and the space of piecewise linear functions is spanned by a basis of translated "hat" functions.

To complete the approximation, we must find a way to derive the coefficients. For this, we define an inner product of two functions $f(s)$ and $g(s)$ as $\langle f, g \rangle = \int ds\, f(s) g(s)$. Two functions are orthogonal iff $\langle f, g \rangle = 0$. We then say that a function $\hat{B}(s)$ is the orthogonal projection of $B(s)$ into the finite dimensional function space if $\langle B - \hat{B}, N_i \rangle = 0$ for all basis functions $N_i(s)$.

If the original basis functions are orthonormal we can find the coefficients of a function $B(s)$ with respect to the basis $\{N_i\}$ by performing inner products

$$\hat{B}(s) = \sum_i B_i N_i(s) = \sum_i \langle B, N_i \rangle N_i(s)$$

In the case of bases which are not orthonormal we must use inner products with the *dual* basis functions (see [20]) to find the coefficients.

Using projection methods, instead of solving the integral equation (1), we solve the related integral equation[2]

$$\hat{B}(s) = \hat{E}(s) + \sum_i \left\langle \int dt\, k(s,t) \hat{B}(t),\ N_i(s) \right\rangle N_i(s) \tag{2}$$

In words, we *operate* on (integrate against the kernel) the projected function $\hat{B}(t)$. After having been operated on, the resulting function generally no longer lies in the finite dimensional function space, so the function is reprojected against the $N_i(s)$. $\hat{B}$ can be obtained by solving the linear system

$$\begin{aligned} B_i &= E_i + \sum_j B_j K_{ij} \\ K_{ij} &= \int ds \int dt\, k(s,t) N_j(t) N_i(s) \end{aligned} \tag{3}$$

To compute the integrals $K_{ij}$ some form of numerical quadrature or closed form solution [21] must be employed. If the basis functions are piecewise constant, these integrals are related to the well known form factors.

[1] The reflectivity, $\rho$, is actually a function of wavelength. Without loss of generality, we will consider only a monochromatic world for the remainder of this paper.

[2] In order to simplify the presentation we will write the radiosity function as having one variable, and the kernel function as having two variables. In the text we will explain what needs to be done for a 3D radiosity implementation.

It is important to remember that the projected equation is only an approximation to the original integral equation. Projections into different finite dimensional spaces will result in different approximations with differing amounts of error and different types of error. In general the projection error is $O(h^{p+1})$ where $h$ is the resolution of the grid, and $p$ the degree of the polynomial used which favors higher order basis functions. Higher order basis functions also result in smoother reconstructed radiosity solutions leading to fewer visual artifacts. However, higher order basis functions require more work to evaluate the associated inner products, possibly offsetting potential savings.

One set of choices for basis functions is given by the family of functions called wavelets.

## 4 Wavelets

Wavelet theory is a rapidly developing field that has its roots in pure mathematics [7] and signal processing [16]. Good introductions to the topic can be found in [17, 4]. In this section we review some wavelet theory focusing on the relevant issues for radiosity.

Wavelets form hierarchical bases which can offer alternative bases for familiar finite dimensional function spaces. The simplest wavelet construction is the Haar construction shown in Figure 2. In the upper left is a set of basis functions which span all piecewise constant functions at resolution 8 on the interval. Using the operators $g$ (pairwise differencing) and $h$ (pairwise averaging) we can construct another basis for the same space (upper right). Four of these functions are just like the original basis, only wider, thus we can repeat the construction (middle right). Repeating once more we finally have a basis for the original space of functions consisting of the overall average $\phi_0$ and the difference functions $\psi_{i,j}$ from all the lower levels. The last set of functions is known as the Haar *wavelet basis*. This construction is very similar to an image pyramid that one might use for texture mapping. In such a pyramid the image (function in our case) is represented at different levels of resolution by successive averaging steps. In the Haar pyramid we only remember the overall average and all the *differences* between successive levels of the pyramid.

The Haar basis is only the simplest example of an infinite family of such constructions, however the basic principles are the same for all wavelet bases. More formally we start with two functions $\psi(s)$ (sometimes called the *detail* function) and $\phi(s)$ (the *smooth* function) defined on the unit interval $s \in [0,1]$. Scales (or levels) $i$ and translates $j$ of $\phi(s)$ and $\psi(s)$ are expressed as

$$\phi_{i,j}(s) = 2^{i/2}\phi(2^i s - j)$$
$$\psi_{i,j}(s) = 2^{i/2}\psi(2^i s - j)$$

with $j = 0, \ldots, 2^i - 1$. According to this indexing, the function $\phi_{i,j}$ is just like the function $\phi_{i-1,j}$ except that $\phi_{i-1,j}$ is twice as wide, and $1/\sqrt{2}$ times as tall (the wider functions are shorter so that $\langle \phi_{i,j}, \phi_{i,k} \rangle$ remains constant independent of $i$). Similarly, $\phi_{i,j}$ is just like the function $\phi_{i,j+1}$ except it is translated. To create an $n = 2^L$ dimensional function space we construct an $L$ level hierarchy of functions that are scales and translates of $\phi$ and $\psi$ (Figure 2 illustrates $L = 3$). We obtain the wavelet basis for the hierarchy by choosing only the detail shapes on all levels plus the smooth shape on the top level, $\psi_{i,j}$, $i = 0, \ldots, L-1$ and $\phi_0$. Between levels there is the so called *two-scale* relationship

$$\phi_{i-1,j} = \sum_k h_{k-2j}\phi_{i,k}$$
$$\psi_{i-1,j} = \sum_k g_{k-2j}\phi_{i,k}$$

In words the $\phi$ functions at a given level can be linearly combined to yield $\phi$ and $\psi$ functions at the next coarser level. This combination can be expressed as a convolution with some sequences $h$ and $g$ with the result subsampled by 2 (expressed by the factor 2 in the index "$k - 2j$" of $h$ and $g$). The sequences $h$ and $g$ can be thought of as a low pass filter and high pass filter respectively.

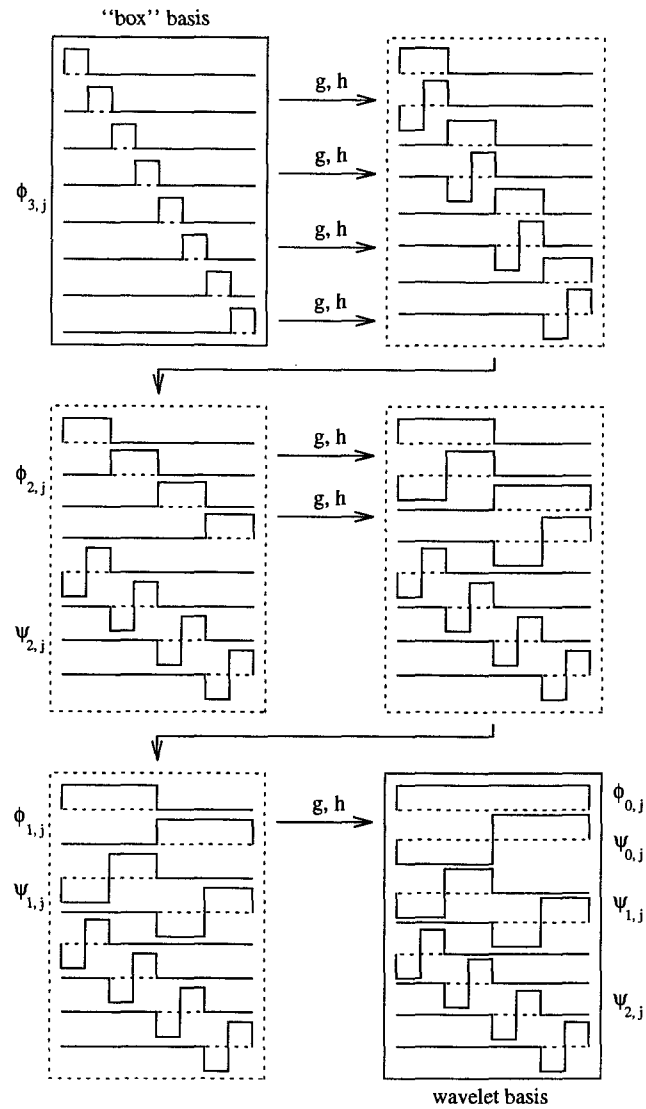



**Figure 2**: Transformation of a piecewise constant basis into the Haar wavelet basis.

The projection of an arbitrary function $B(s)$ into a wavelet[3] basis can be formally written as

$$\hat{B}(s) = \langle B, \phi_0 \rangle \phi_0(s) + \sum_{ij} \langle B, \psi_{i,j} \rangle \psi_{i,j}(s) \qquad (4)$$

Instead of computing all the above inner products, we can find the coefficients efficiently by exploiting the two-scale relationship. Given the projection of some arbitrary function $B(s)$ with respect to the lowest level basis $\phi_{L,j}$ the wavelet coefficients can be found using a pyramid algorithm [16]. Each stage of this algorithm takes a vector of coefficients and convolves it with the filters $h$ and $g$, returning the smooth and detail coefficients one level up

[3] To simplify the discussion we are assuming that we have an orthonormal wavelet basis. We discuss the non orthonormal case in [20].

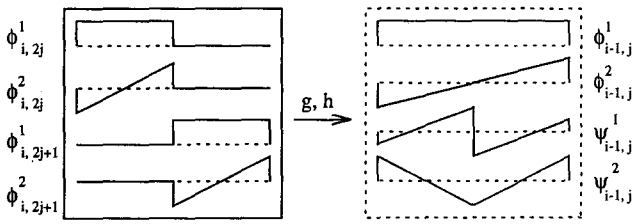



**Figure 3**: The $\mathcal{M}_2$ wavelet construction whose smooth shapes are the first two Legendre polynomials. Both of the detail shapes (lower right) have two vanishing moments.

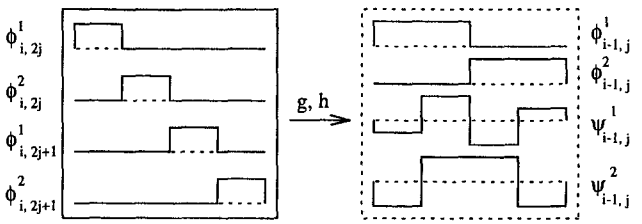



**Figure 4**: The $\mathcal{F}_2$ wavelet construction. $\mathcal{F}_2$ bases have two different detail shapes. Both of the detail shapes have two vanishing moments.

```
XformUp( vector B_φ, int i )
  for( j = 0; j < 2^i/2; j++)
    B_φ^up[j] = Σ_k h_{k-2j} B_φ[k];
    B_ψ^up[j] = Σ_k g_{k-2j} B_φ[k];
  return ( B_φ^up, B_ψ^up );
```

The entire one dimensional pyramid transform is then stated as

```
PyramidUp( vector B_{φ_{L,k}} )
  for( i = L; i > 0; i-- )
    (B_{φ_{i-1,k}}, B_{ψ_{i-1,k}}) = XformUp( B_{φ_{i,k}}, i );
  return ( B_{φ_0}, B_{ψ_{i,k}}, i = 0, ..., L-1 );
```

If the $h$ and $g$ convolutions have constant width (with respect to $i$) then each call to **XformUp** has cost linear in the length of the array passed in. Since each successive call in **PyramidUp** works on only the smooth half left by the previous call the overall runtime to build the pyramid is $O(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1) = O(n)$.

A similar algorithm **PyramidDown** reverses this process using **XformDown** for successive calls

```
XformDown( vector B_φ, vector B_ψ, int i )
  for( j = 0; j < 2 * 2^i; j++ )
    B_φ^down[j] = Σ_k h_{j-2k} B_φ[k] + Σ_k g_{j-2k} B_ψ[k];
  return B_φ^down ;

PyramidDown( B_{φ_0}, B_{ψ_{i,k}}, i = 0, ..., L-1 )
  for( i = 0; i < L; i++ )
    B_{φ_{i+1,k}} = XformDown( B_{φ_{i,k}}, B_{ψ_{i,k}}, i );
  return B_{φ_{L,k}} ;
```

A key property of wavelets essential to this work is that a sufficiently smooth function $B(s)$, when expressed in a wavelet basis (Equation 4) will have many small coefficients. By ignoring these negligible coefficients we are left with a sparse, approximate representation. The negligible coefficients occur because wavelet functions have *vanishing moments*. We say that a function $\psi(s)$ has $M$ vanishing moments if

$$\int ds\, \psi(s) s^i = 0, \quad i = 0, \ldots, M-1$$

The Haar wavelet (Figure 2) has one vanishing moment, thus the projection of a nearly constant function into the Haar basis will have wavelet coefficients near 0. Similarly, if a wavelet basis function has two vanishing moments, the projection of a linear function will vanish. Figures 3 and 4 show examples of wavelets, $\psi$, with two vanishing moments.

## 5 Wavelets In Higher Dimensions

Wavelet bases for functions of two or more variables are required for radiosity. Our goal is to project the kernel, which is a four dimensional function, into a basis set in which it has a sparse representation.

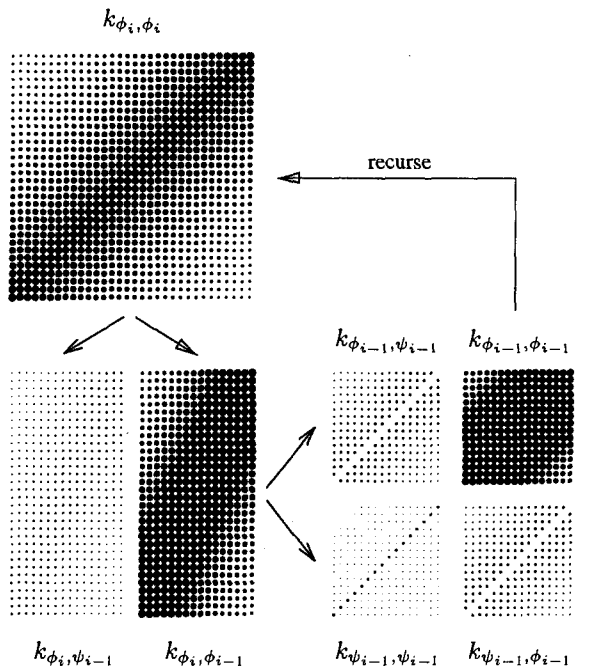



**Figure 5**: The 2D Pyramid Algorithm is applied to form factors taken from the flatland radiosity environment consisting of two parallel line segments. (Flatland [13] is radiosity in a plane). The dot size indicates the magnitude of a given entry in the matrix.

An arbitrary function $k(s,t)$ of two variables on a finite two dimensional interval can be approximated by some function $\hat{k}(s,t)$ that lies in a two variable finite dimensional function space. Given a particular one dimensional wavelet, a 2D wavelet basis[4] is made up of the functions

$$\begin{aligned} &\phi_0(s)\phi_0(t) \\ &\psi_{i,j}(s)\psi_{i,k}(t) \\ &\psi_{i,j}(s)\phi_{i,k}(t) \\ &\phi_{i,j}(s)\psi_{i,k}(t) \end{aligned}$$

where we only couple functions on the same scale $i$, where $i = 0, \ldots, L-1$ and $j, k = 0, \ldots, 2^i - 1$.

The 2D wavelet coefficients may be obtained from the finest resolution coefficients $B_{\phi_{L,j},\phi_{L,k}}$ using a 2D **PyramidUp** algorithm. This algorithm begins with the $B_{\phi_{L,j},\phi_{L,k}}$ written in a 2D matrix tableau. It then applies **XformUp** once to each row, followed by an application of **XformUp** to each resulting column. This procedure is applied recursively to the $B_{\phi_{L-1,j},\phi_{L-1,k}}$ quar-

[4] Another 2D wavelet basis could be constructed from the tensor product of a 1D wavelet basis. The different forms of multidimensional wavelet bases are discussed in [3, 20].

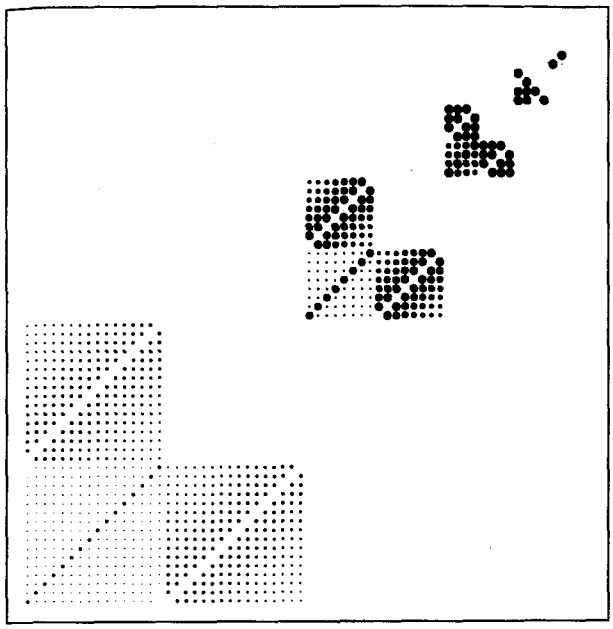


**Figure 6**: To illustrate the sparseness of the kernel matrix we transform the flatland radiosity matrix from Figure 5 into the 2D Haar basis. Many of the coefficients are small in magnitude (small dots).

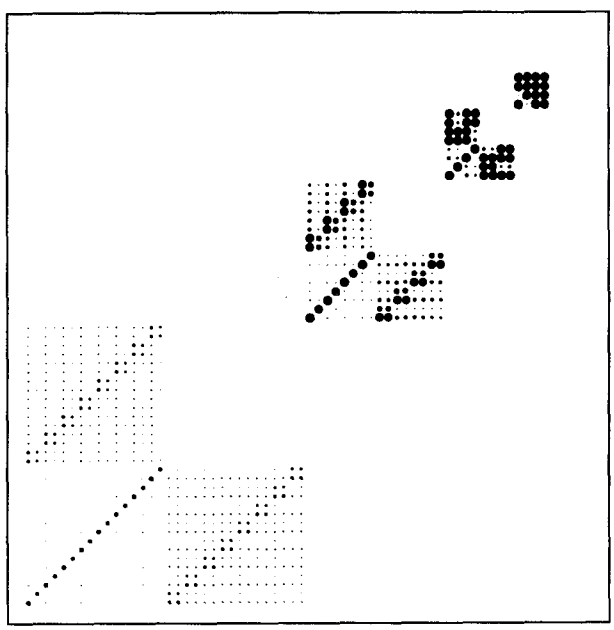


**Figure 7**: We transform the same matrix into the $\mathcal{F}_2$ basis. Notice that even more of the coefficients are negligible now.

ter (Figure 5). The construction of a 2D **PyramidDown** follows analogously from the one dimensional **PyramidDown**.

This construction can be extended to functions of four variables such as the kernel in 3D radiosity $k(s_1, t_1, s_2, t_2)$. For this case, there are sixteen combinations of $\phi$ and $\psi$ functions in four variables. The basis is made up of all fifteen combinations on the same scale $i$ which involve $\psi$ functions. The corresponding pyramid transformation functions are constructed as in the two dimensional case by applying **XformUp** and **XformDown** respectively to each dimension in turn.

For this type of multidimensional wavelet basis Beylkin et al. [3] show that for a given error tolerance, only $O(n)$ coefficients need to be used to attain the prescribed error tolerance in the results of our computations. Figures 6 and 7 visualize the sparseness of a flatland radiosity kernel when written in two wavelet bases with one and two vanishing moments respectively.

## 6 Radiosity with Wavelets

To obtain an efficient radiosity algorithm, we project the kernel by taking inner products with the wavelet basis functions. The coefficients of the kernel with respect to the basis are given by

$$\begin{aligned}
k^{\phi} = k_{\phi_0,\phi_0} &= \int dt \int ds\, k(s,t)\phi_0(s)\phi_0(t) \\
k^{\alpha}_{ijk} = k_{\psi_{i,j},\psi_{i,k}} &= \int dt \int ds\, k(s,t)\psi_{i,j}(s)\psi_{i,k}(t) \\
k^{\beta}_{ijk} = k_{\phi_{i,j},\psi_{i,k}} &= \int dt \int ds\, k(s,t)\phi_{i,j}(s)\psi_{i,k}(t) \\
k^{\gamma}_{ijk} = k_{\psi_{i,j},\phi_{i,k}} &= \int dt \int ds\, k(s,t)\psi_{i,j}(s)\phi_{i,k}(t)
\end{aligned}$$

Because of the vanishing moment properties of the wavelets and the smoothness properties of the kernel, many of these terms are nearly zero.

A projected version of the integral operator can now be derived by projecting the kernel itself. This derivation which we only sketch here is described in greater detail in Beylkin et al. [3]. The $k^{\alpha}$, $k^{\beta}$ and $k^{\gamma}$ coefficients are used to represent the kernel which has been approximated with respect to the wavelet basis. Given this projection, after performing the necessary algebra, the approximate operator can be written as

$$\begin{aligned}
&\int dt\, \hat{k}(s,t)B(t) = \\
&\quad B^{\phi}k^{\phi}\phi_0(s) + \sum_{ij}\Big(\sum_{k} B^{\alpha}_{ik}k^{\alpha}_{ijk}\Big)\psi_{i,j}(s) \\
&\quad + \sum_{ij}\Big(\sum_{k} B^{\beta}_{ik}k^{\beta}_{ijk}\Big)\phi_{i,j}(s) + \sum_{ij}\Big(\sum_{k} B^{\gamma}_{ik}k^{\gamma}_{ijk}\Big)\psi_{i,j}(s)
\end{aligned} \tag{5}$$

where

$$\begin{aligned}
B^{\alpha}_{ik} = B^{\gamma}_{ik} = B_{\psi_{i,k}} &= \int dt\, \psi_{i,k}(t)B(t) \\
B^{\beta}_{ik} = B_{\phi_{i,k}} &= \int dt\, \phi_{i,k}(t)B(t) \\
B^{\phi} = B_{\phi_0} &= \int dt\, \phi_0(t)B(t)
\end{aligned}$$

### 6.1 The Basic Algorithm

Equation 5 suggests the following three phase algorithm to approximate the kernel operating on a radiosity function.

**Step 1** *Pull*: Obtain the $n$ ($n$ = number of bases of the radiosity function) coefficients $B^{\alpha}$ and the $n$ coefficients $B^{\beta}$ of the radiosity function. If we are initially given the coefficients $B_{\phi_{L,j}}$, the $2n$ needed coefficients can be obtained by calling a procedure **Pull** which is just like **PyramidUp** except it returns *both* the $\phi$ and $\psi$ coefficients. This step transforms $n$ coefficients into $2n$

coefficients. A 1D **Pull** would then be

```
Pull( vector B_{φ_{L,k}} )
  for( i = L; i > 0; i-- )
    (B_{φ_{i-1,k}}, B_{ψ_{i-1,k}}) = XformUp( B_{φ_{i,k}}, i );
  return ( B_{φ_{i,k}}, B_{ψ_{i,k}}, i = 0, ..., L-1 );
```

**Step 2** *Gather*: Let the projected kernel operate on the projected radiosity function. This means that we sum over the index $k$, and is equivalent to a matrix multiply. Because of the vanishing moments of the wavelet functions most of the $n^2$ kernel coefficients will be near zero and may be ignored if the action of the kernel is desired to finite precision. The procedure **Gather** results in $2n$ coefficients $G_{\phi_{i,j}}$ and $G_{\psi_{i,j}}$ that represent the resultant radiosity function as a combination of $\phi_{i,j}(s)$ and $\psi_{i,j}(s)$.

**Step 3** *Push*: Reconstruction of the radiosity function using the $2n$ functions $\phi_{i,j}(s)$ and $\psi_{i,j}(s)$ is done with the procedure **Push** which is similar to **PyramidDown** but takes as arguments *both* the $\phi$ and $\psi$ coefficients. A 1D **Push** would then be

```
Push( B_{φ_{i,k}}, B_{ψ_{i,k}}, i = 0, ..., L-1 )
  for( i = 0; i < L; i++ )
    B_{φ_{i+1,k}} += XformDown( B_{φ_{i,k}}, B_{ψ_{i,k}}, i );
  return B_{φ_{L,k}};
```

Wrapping this projected operator within a Jacobi iteration loop results in the following algorithm

```
(k^α, k^β, k^γ) = ProjectKernel();
B_{φ_{L,k}} = E_{φ_{L,k}};
while( !converged )
  G = 0;
  (B_{φ_{i,k}}, B_{ψ_{i,k}}) = Pull( B_{φ_{L,k}} );
  (G_{φ_{i,j}}, G_{ψ_{i,j}}) = Gather( B_{φ_{i,k}}, B_{ψ_{i,k}}, k^α, k^β, k^γ );
  G_{φ_{L,k}} = Push( G_{φ_{i,j}}, G_{ψ_{i,j}} );
  B_{φ_{L,k}} = G_{φ_{L,k}} + E_{φ_{L,k}};
Display();
```

The push and pull can be done in $O(n)$ (linear in the number of elements) steps. The gather step (this is a complete gather sweep which updates all of the entries) can be done in $O(m)$ time where $m$ is the number of terms in the kernel expansion (matrix) that are significant. We want $m$ to be as small as possible. Wavelet bases will lead to $m = O(n)$ where the constant factor in $O(n)$ decreases with the number of vanishing moments.

What remains is to project the kernel into the wavelet basis, which may be done as follows

```
ProjectKernel()
  k_{φ_{L,j},φ_{L,k}} = Quadrature( k, φ_{L,j}, φ_{L,k} );
  (k^α, k^β, k^γ) = PyramidUp( k_{φ_{L,j},φ_{L,k}} );
  where( (k^α, k^β, k^γ) < ε )
    (k^α, k^β, k^γ) = 0;
```

## 6.2 The Top Down Approach

Unfortunately, this bottom up **ProjectKernel** is an expensive implementation requiring quadratic time and space. The costs can be dramatically cut by using an *oracle* which predicts which $m$ of the $n^2$ coefficients of the projected kernel are significant. Then, these $m$ values are computed directly by quadrature or symbolic integration.

Assuming that the oracle can estimate the smoothness of the kernel for a given region (vis-a-vis a given number of vanishing moments), an efficient top down recursive version of **ProjectKernel** can be written as follows

```
ProjectKernel( i, patch p, patch q )
  smooth = AskOracle( p, q );
  if( smooth ) return;
  else
    (k^α_{i,j(p),k(q)}, k^β_{i,j(p),k(q)}, k^γ_{i,j(p),k(q)})
      = Quadrature( k, p, q );
  if( i == L-1 ) return;
  else
    ProjectKernel( i+1, left(p), left(q) );
    ProjectKernel( i+1, left(p), right(q) );
    ProjectKernel( i+1, right(p), left(q) );
    ProjectKernel( i+1, right(p), right(q) );
```

If the oracle finds the region under consideration sufficiently smooth no more recursive calls need be executed, since the coefficients at lower levels will be insignificant by assumption. The function **Quadrature()** computes the projection of the kernel function onto the basis functions at the given level.

## 6.3 3D Radiosity

In 3D radiosity $B$ is a function of two variables so in the main program we use a 2D **Pull** and a 2D **Push** respectively. $k$ is a function of four variables so in the bottom up **ProjectKernel** we use a 4D **PyramidUp** function. In the top down approach to **ProjectKernel** there are fifteen not three quadratures and sixteen recursive calls for all combinations of four children of **p** and **q**.

# 7 Implementation

The top down algorithm described above has been implemented by extending the implementation of hierarchical radiosity described in Hanrahan et al. [11].

## 7.1 Choice of Basis

Two families of wavelets have been explored, multiwavelets [1] and a family of wavelets that we call flatlets. Each of these families have members with any number of vanishing moments.

The construction of $\mathcal{M}_M$ (multiwavelet with $M$ vanishing moments) begins with $M$ smooth functions which are the first $M$ Legendre Polynomials, $\phi^m(s) = L_m(s)$, and $M$ detail functions $\psi^m(s)$ that are piecewise polynomials of degree $M-1$, and have $M$ vanishing moments. A hierarchy is then constructed from these shapes. $\mathcal{M}_1$ is the Haar basis, however, for $M$ greater than 1, $\mathcal{M}_M$ is technically speaking not a true wavelet since it begins with a collection of $\phi$ and $\psi$ functions instead of a single pair.

Multiwavelets form an orthonormal basis. Figure 3 shows the basis functions for the $\mathcal{M}_2$ hierarchy. The two-scale relationship for $\mathcal{M}_2$ is expressed concisely as

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 2 & 0 & 2 & 0 \\ -\sqrt{3} & 1 & \sqrt{3} & 1 \\ 0 & -2 & 0 & 2 \\ 1 & \sqrt{3} & -1 & \sqrt{3} \end{bmatrix}\begin{bmatrix} \phi^1_{i,2j} \\ \phi^2_{i,2j} \\ \phi^1_{i,2j+1} \\ \phi^2_{i,2j+1} \end{bmatrix} = \begin{bmatrix} \phi^1_{i-1,j} \\ \phi^2_{i-1,j} \\ \psi^1_{i-1,j} \\ \psi^2_{i-1,j} \end{bmatrix}$$

Using this relationship the push and pull operations can be computed using a binary tree, instead of as a subsampled vector convolution. A node stores the four coefficients of the functions $\phi^1_{i-1,j}$, $\phi^2_{i-1,j}$, $\psi^1_{i-1,j}$, $\psi^2_{i-1,j}$. During a pull, a node computes the values of its coefficients as a linear combination of the $\phi^1_{i,2j}$, $\phi^2_{i,2j}$ coefficients obtained from its left child, and the $\phi^1_{i,2j+1}$, $\phi^2_{i,2j+1}$ coefficients obtained from its right child. To represent the radiosity function over a patch we need a 2D $\mathcal{M}_2$ basis for which we use

a quad-tree where each node stores sixteen coefficients. During a pull, a node computes its coefficients as a linear combination of the sixteen $\phi\phi$ coefficients from its children (four from each child).

The flatlet basis $\mathcal{F}_M$ is made up entirely of piecewise constant functions. The $\phi^m$ are $M$ adjacent box functions, and the $\psi^m$ are $M$ piecewise constant functions that have $M$ vanishing moments. Figure 4 shows the $\mathcal{F}_2$ hierarchy.

For $\mathcal{F}_2$, the two-scale relationship is given by

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -1 & 3 & -3 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}\begin{bmatrix} \phi^1_{i,2j} \\ \phi^2_{i,2j} \\ \phi^1_{i,2j+1} \\ \phi^2_{i,2j+1} \end{bmatrix} = \begin{bmatrix} \phi^1_{i-1,j} \\ \phi^2_{i-1,j} \\ \psi^1_{i-1,j} \\ \psi^2_{i-1,j} \end{bmatrix} \quad (6)$$

The top two rows of the matrix in the above equation are chosen to give us box functions twice as wide. The bottom two rows are chosen to be orthogonal to constant and linear variation, (the vectors $[1,1,1,1]$, $[0,1,2,3]$)[5]. For a discussion of a similar construction see [2].

Both flatlets and multiwavelets can be constructed to have any number of vanishing moments to increase the sparseness of the integral operator representation. For both bases, the case $M = 1$ reduces to the Haar basis. For $M > 1$ multiwavelets offer the benefits of projecting into a higher order space, resulting in increased convergence rates and smoother basis functions to represent the answer. These benefits come at the expense of higher order quadratures necessary for the inner products. Flatlets for $M > 1$ also offer accelerated convergence while the quadratures remain equivalent to form factor computations for which there exits a large body of literature and code, and for which some closed form solutions are known. The final answer is still represented as a piecewise constant function, albeit at the finest resolution $\phi_{L,j}$. Since the degree of the basis functions does not go up in the flatlet case the width of support needs to be increased as $M$ increases.

With multiwavelets and flatlets there is also a cost incurred by increasing the number of vanishing moments. Larger $M$ will result in $h$ and $g$ filters with wider support. Thus any non-smoothness in $k(s,t)$, such as a shadow discontinuity, will fall under the support of more basis functions. This increases the number of significant terms in the integral operator.

[5] A technical detail concerns the fact that flatlets for $M > 1$ are not orthonormal and thus require the dual basis functions to compute `PyramidUp` (see [20]).

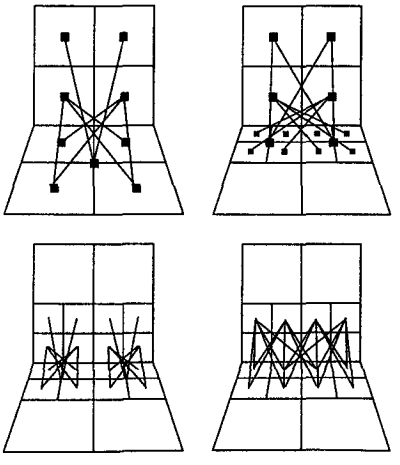


**Figure 8**: Two different oracles and the interaction patterns they generate.

## 7.2 Pull, Push and Gather

Both multiwavelets and flatlets are instances of *tree* wavelets. A tree wavelet has the property that the convolution sequences $h$ and $g$ for two neighboring elements do not overlap. This property allows us to organize all computations along a tree which does not need to have uniform depth. Tree wavelets also allow for another simplification. Since all necessary coefficients reside in the immediate children of a node we can use the two-scale relationship to store only the $\phi\phi$ coefficients and need not represent the $\phi\psi$, $\psi\phi$, and $\psi\psi$ coefficients explicitly. With this simplification `ProjectKernel` is implemented as follows

```
ProjectKernel( i, patch p, patch q )
  ParentLevelsmooth = AskOracle( p, q );
  if( ParentLevelsmooth || i == L )
    k_{φ,φ} = Quadrature( k, p, q );
    CreateLink( k_{φ,φ}, p, q );
  else
    ProjectKernel( i+1, left(p), left(q) );
    ProjectKernel( i+1, left(p), right(q) );
    ProjectKernel( i+1, right(p), left(q) );
    ProjectKernel( i+1, right(p), right(q) );
```

In our implementation of radiosity using the $\mathcal{M}_M$ and $\mathcal{F}_M$ bases, the radiosity function over each polygon is represented by $B_{\phi\phi}$ coefficients that are stored in a quad-tree. Each node holds $M^2$ $B_{\phi\phi}$ coefficients. Pulling and pushing are done in the quad-tree as in [11] except that for different bases, we use different two-scale relationships. The kernel is represented by its $k_{\phi\phi\phi\phi}$ coefficients that are stored on links created between nodes of different polygons' quad-trees. Each such link caries $M^4$ interaction terms. For the $\mathcal{F}_M$ bases the interaction terms are still form factors, but for $\mathcal{M}_M$ the coefficients on the links represent higher order interactions which require quadrature computations of the appropriate order. Gathering is done by moving $B$ values across the links, weighted by the $k$ values on the link. In this context, HR can be viewed as wavelet radiosity using the Haar basis.

## 7.3 Oracle

The oracle must decide whether the kernel is sufficiently smooth over two patches in the environment i.e., resembles a polynomial of degree $M-1$ or less. If the kernel is smooth, all $\psi$ terms will (sufficiently) vanish and thus any work to evaluate the lower interaction terms can be avoided.

The most accurate approach to measure the kernel smoothness is to directly evaluate the integrals of the kernel against the $\psi$ on this and all lower levels and verify that they are below the required threshold. This is computationally too expensive and we approximate this computation in the following way. The kernel is sampled at the points required by a Gauss-Legendre quadrature rule of the appropriate order and an interpolating polynomial of degree $M-1$ is constructed using Neville's algorithm [22]. Given this interpolating polynomial $k_P$ we compute the $L_1$ error $\int |k_P - k|$ with a quadrature rule which places sample points in-between the previously chosen points. If the value of this integral is small we conclude that our current level of (smooth) approximation matches the kernel function well and the `AskOracle` function returns `True`. Note that the sample points for the interpolating polynomial are chosen so that they can be used directly in the computation of the interaction link values. If the `AskOracle` function returns `False` these samples are discarded. A less costly approach could use geometric information, such as the size, orientation, and distance between two patches. In effect this was done in the original HR implementation. However for the $\mathcal{F}_M$ and $\mathcal{M}_M$, $M > 1$ bases it is not immediately clear what the corresponding geometric reasoning would be.

It is important to realize that any such implementation of an oracle will introduce errors due to its approximate nature. If the oracle is not stringent enough, and necessary terms are neglected, artifacts will appear in the image. Figure 8 shows two different oracles and the interactions they force. Two successive levels of interactions are shown (top to bottom). On the left is an oracle allowing patches close to the singularity (where the kernel varies rapidly) to be linked (meaning no further subdivision will be done). For this oracle the interaction patterns separate on the lower level. On the right is a more stringent oracle which does not allow singular interactions until patches have become very small. As a result we do not see the separation.

As in [11] we use *brightness refinement* which means that the stringency of the oracle is weighted by the brightness of the involved patches. Also as in [11] a fast partial visibility test is performed by using a constant number of jittered rays. If two patches are partially occluded and there is sufficient energy being transferred between the two patches the oracle returns `False`.

## 7.4 Quadrature

If the oracle returns `True`, numerical integrations must be performed to compute the $k_{\phi\phi\phi\phi}$ terms associated with the link to be created. Our implementation uses Gauss-Legendre quadrature [22] for this purpose. A Gauss-Legendre quadrature rule provides an accurate integration for polynomials up to order $2p-1$, where $p$ is the number of sample points. The order of the quadrature and the related number of sample points required depends on the sum of the order of the wavelet bases, and the assumed order of the kernel itself.

For the projection of the kernel against a flatlet basis, a two point rule is used for each constant section of the basis function. In the case of multiwavelets $\mathcal{M}_M$, $M > 1$, $M$ points are chosen along each coordinate axis since we need to have a high enough order of integration to account for the polynomial variance in the kernel and the polynomial basis functions themselves. For example, for $M = 3$ we compute coefficients when the kernel varies approximately up to $2^{nd}$ by projecting onto basis functions up to $2^{nd}$ order. Thus the integrand is approximately $4^{th}$ order, and we can use a three point Gauss rule.

The number of integrals which need to be computed for a link is $M^4$, however for all these integrals only a total of $M^4$ samples of the kernel function are required. Using precomputed weights, these samples are combined to give all the desired integrals.

We treat visibility following [11] by casting a constant number of jittered rays between two patches to estimate the fraction of visibility. This is then used to attenuate the quantity returned by the Gauss-Legendre quadrature. This technique relies on the fact that we always subdivide in the vicinity of a shadow discontinuity limiting errors due to the non-smooth nature of the kernel to a small region.

When the two patches that are linked up are close to the singularity in $k$, quadratures will encounter numerical difficulties if they are not properly adapted to the singularity. In particular a Gauss-Legendre rule will produce large errors and an adapted quadrature rule is required. This phenomenon is not unique to wavelet radiosity but applies to all GR methods. Special Gauss rules can be designed for the particular singularity found in the radiosity kernel. Zatz [25] uses such custom rules and notes the need for an automatic decision procedure as to when to switch the type of integration. In our implementation of flatlets, we use a closed form solution for the form factor [21] whenever the patches border on the singularity. While this computation is expensive, it only needs to be invoked in a small fraction of interaction computations and contributes little to overall runtime. For multiwavelets we have no such closed form available. In this case the oracle forces subdivision to small enough patches at the singularity that the resulting errors contribute very little to the overall error. Alternative constructions for singular transports are discussed in [19].

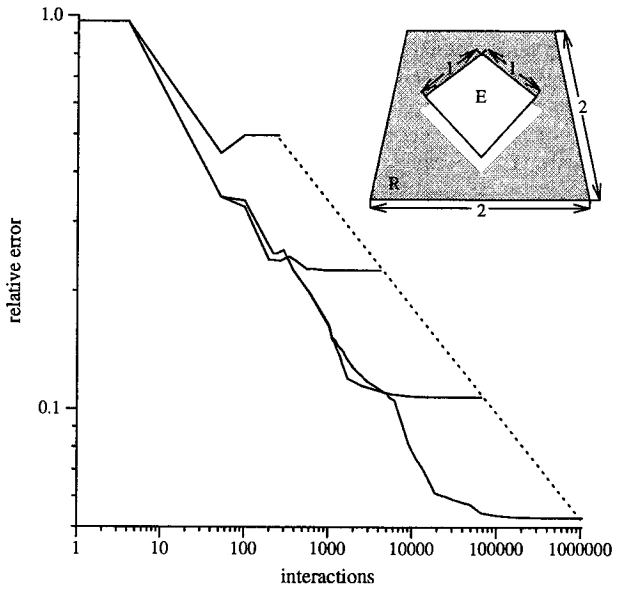



**Figure 9**: Relative $L_1$ error as a function of the number of interaction links for the haar basis with $h = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ (top to bottom). The test configuration is depicted in the upper right corner.

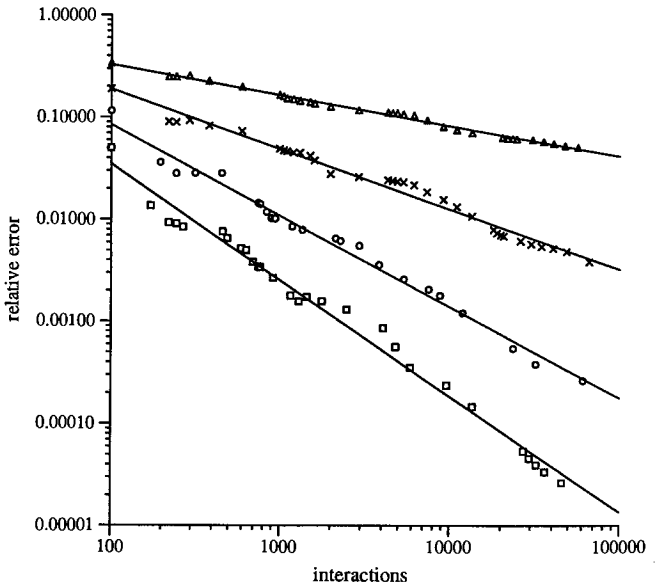



**Figure 10**: Relative $L_1$ error as a function of the number of interactions for the wavelet bases $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ (top to bottom) using the same test configuration as in Figure 9. Here $h = \frac{1}{32}$.

# 8 Experimental Results

In this section we present findings that compare how radiosity behaves using different wavelet bases. We give results from the analysis of a simple 3D configuration, for which we have an analytic solution against which to check our results. We finish with an image of a full environment.

One test case used the configuration depicted in the inset in

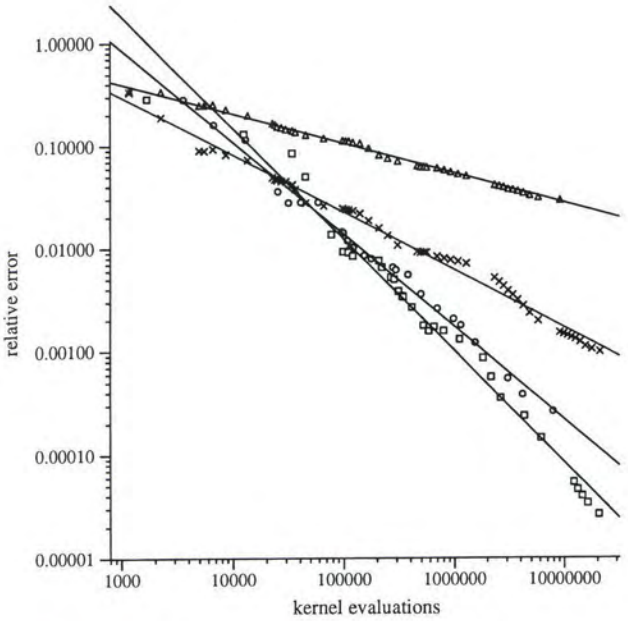



**Figure 11**: Relative $L_1$ error as a function of work.

**Figure 12**: Computed image of perpendicular emitter and receiver. for the Haar basis (left), and $\mathcal{F}_2$ basis (right) using same amount of work. Note that we have not performed any post processing such as Gouraud shading.

Figure 9. A pure emitter of side length 1 is placed 0.1 units above a pure receiver of side length two. For this particular configuration the radiosity on the receiver is given by the differential area to finite area form factor at every point. Figure 9 shows the behavior of the relative $L_1$ error for the Haar basis as a function of the number of interactions for various grid sizes $h$. The far point on each of the lines corresponds to a full matrix solution. Note in particular that the final accuracy is reached well before all matrix elements are computed. Plots for higher order basis functions exhibit the same overall shape but with steeper slopes and overall lesser error. Figure 10 shows the behavior of the $\mathcal{M}_M$ bases for $M = 1, \ldots, 4$ and $h = \frac{1}{32}$. The ratio of successive slopes (as fitted to the points) is almost precisely $1:2:3:4$, as one would expect from the order of basis functions employed. For both plots we have depicted error as a function of number of interactions. However a user experiences error as a function of work which is more accurately measured by the number of kernel evaluations. Since the amount of work increases for higher order methods it is not clear a priori whether a higher order method will always yield better results in a shorter time. Figure 11 shows error as a function of kernel evaluations for the same data as that used in Figure 10. The plot for $M = 1$ is translated with respect to all others since we always use at least a two point quadrature rule even if the basis functions are constant. The plot shows that if

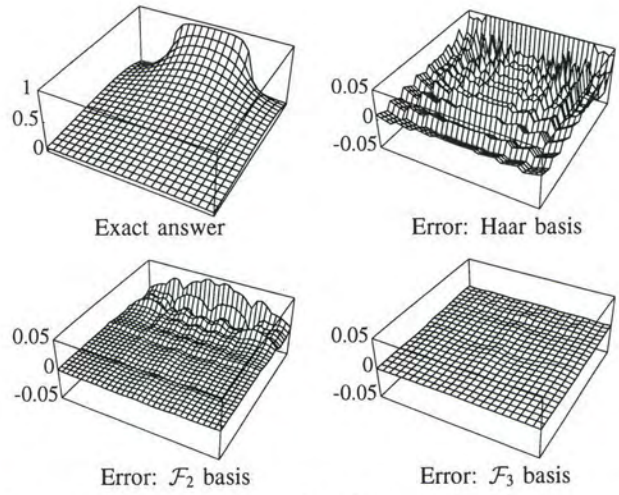



**Figure 13**: Heightfield error plots for perpendicular emitter and receiver.

**Figure 14**: Architectural scene computed with the $\mathcal{M}_2$ basis and rendered directly from the basis functions.

sufficient accuracy is required higher order basis functions achieve lower error for the same amount of work.

We have also examined the behavior of our methods near the singularity of an environment consisting of perpendicular polygons (Figure 12). The emitter was chosen to be half as wide as the receiver to create more variation in the radiosity function. The grid size was set to $\frac{1}{32}$. The upper left plot in Figure 13 shows the exact solution plotted as a height field over the receiver. On the top right is a plot of the difference between exact solution and the computed solution for the Haar basis. On the bottom are similar error surfaces for the $\mathcal{F}_2$ and $\mathcal{F}_3$ bases (left and right respectively). The amount of work was approximately constant (8000 interactions) for all three solutions. The graphs show clearly the lesser and smoother error for the $\mathcal{F}_2$ and $\mathcal{F}_3$ bases demonstrating the effectiveness of bases with more vanishing moments. This is also illustrated by the rendered images in Figure 12.

The algorithm has also been run on a more complex environ-

ment (Figure 14). This picture, as well as Figure 12, does not use any postprocessing such as Gouraud shading. Instead the surface brightness is computed directly from the basis functions and associated coefficients.

## 9 Conclusion and Future Work

In this paper we have presented the basic theory of projections of integral operators into hierarchical bases, and laid out the theoretical foundation of a new set of techniques involving wavelets. With this in hand, we introduced a new set of linear time algorithms we have called *wavelet radiosity*, and shown that the hierarchical radiosity described by Hanrahan et al. was an instance of a first order wavelet approach.

We have introduced a new family of wavelets, dubbed flatlets and also experimented with a second family of wavelets, multiwavelets. Both lead to efficient algorithms. Future work includes examining various wavelet bases which may have better properties than the multiwavelets and flatlets. For example the Coiflet functions of [7, 3] allow for fast one point quadrature methods. The tree wavelets that we implemented do not enforce any kind of continuity at element boundaries, possibly leading to blocky artifacts. Spline wavelets [4] might provide a basis which would alleviate this.

While our initial implementation was limited to quadrilateral polygons there is nothing in the underlying algorithms that prevents the use of any surface whose parameter domain is rectilinear, such as for example bicubic patches. The only change involves the reparameterization (change of variable) in the coupling integrals. It would be very desirable to design bases which work with triangular domains since triangles are a common primitive in meshing algorithms.

There are still fundamental questions that have yet to be addressed. We would like to gain a better understanding of how wavelet expansions interact with the visibility term in the kernel. It is also important to find methods that remain efficient when the environment consists of a large number of small polygons.

## Acknowledgements

The research reported here was partially supported by Apple, Silicon Graphics Computer Systems, and the National Science Foundation (CCR 9207966). We would like to thank S. V. Krishnan for his useful comments. We would also like to thank Ju-sung Lee, Jonathan McAllister, and Michael Neufeld for creating the model of the room.

## References

[1] ALPERT, B. A Class of Bases in $L^2$ for the Sparse Representation of Integral Operators. *SIAM Journal on Mathematical Analysis 24*, 1 (Jan 1993).

[2] ALPERT, B., BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. Wavelet-like Bases for the Fast Solution of Second-kind Integral Equations. *SIAM Journal on Scientific Computing 14*, 1 (Jan 1993).

[3] BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. Fast Wavelet Transforms and Numerical Algorithms I. *Communications on Pure and Applied Mathematics 44* (1991), 141–183.

[4] CHUI, C. K. *An Introduction to Wavelets*, vol. 1 of *Wavelet Analysis and its Applications*. Academic Press Inc., 1992.

[5] COHEN, M., CHEN, S. E., WALLACE, J. R., AND GREENBERG, D. P. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics 22*, 4 (August 1988), 75–84.

[6] COHEN, M. F., AND GREENBERG, D. P. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics 19*, 3 (July 1985), 31–40.

[7] DAUBECHIES, I. *Ten Lectures on Wavelets*, vol. 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1992.

[8] DELVES, L. M., AND MOHAMED, J. L. *Computational Methods for Integral Equations*. Cambridge University Press, 1985.

[9] GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. Modelling the Interaction of Light between Diffuse Surfaces. *Computer Graphics 18*, 3 (July 1984), 212–222.

[10] GORTLER, S. J., COHEN, M. F., AND SLUSALLEK, P. Radiosity and Relaxation Methods; Progressive Refinement is Southwell Relaxation. Tech. Rep. CS-TR-408-93, Department of Computer Science, Princeton University, February 1993.

[11] HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics 25*, 4 (July 1991), 197–206.

[12] HECKBERT, P. S. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, University of California at Berkeley, January 1991.

[13] HECKBERT, P. S. Radiosity in Flatland. *Computer Graphics Forum 2*, 3 (1992), 181–192.

[14] KAJIYA, J. T. The Rendering Equation. *Computer Graphics 20*, 4 (1986), 143–150.

[15] LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. A Discontinuity Meshing Algorithm for Accurate Radiosity. *IEEE CG&A 12*, 4 (July 1992).

[16] MALLAT, S. G. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11* (July 1989), 674–693.

[17] PRESS, W., TEUKOLSKI, S., VETTERLING, W., AND FLANNERY, B. *Numerical Recipies in C, The Art of Scientific Computing*, 2 ed. Cambridge University Press, 1992.

[18] SALESIN, D., LISCHINSKI, D., AND DEROSE, T. Reconstructing Illumination Functions with Selected Discontinuities. *Third Eurographics Workshop on Rendering* (1992), 99–112.

[19] SCHRÖDER, P. Numerical Integration for Radiosity in the Presence of Singularities. In *Fourth Eurographics Workshop on Rendering* (1993).

[20] SCHRÖDER, P., GORTLER, S. J., COHEN, M. F., AND HANRAHAN, P. Wavelet Projections For Radiosity. In *Fourth Eurographics Workshop on Rendering* (June 1993).

[21] SCHRÖDER, P., AND HANRAHAN, P. On The Form Factor Between Two Polygons. In *Computer Graphics, Annual Conference Series, 1003* (August 1993), Siggraph.

[22] STOER, J., AND BULIRSCH, R. *Introduction to Numerical Analysis*. Springer Verlag, New York, 1980.

[23] SZELISKI, R. Fast Surface Interpolation Using Hierarchical Basis Functions. *IEEE Trans. PAMI 12*, 6 (June 1990), 513–439.

[24] YSERENTANT, H. On the Multi-level Splitting of Finite Element Spaces. *Numerische Mathematik 49* (1986), 379–412.

[25] ZATZ, H. R. Galerkin Radiosity: A Higher-order Solution Method for Global Illumination. In *Computer Graphics, Annual Conference Series, 1003* (August 1993), Siggraph.



# Hierarchical Z-Buffer Visibility

Ned Greene* Michael Kass† Gavin Miller†

## Abstract

An ideal visibility algorithm should a) quickly reject most of the hidden geometry in a model and b) exploit the spatial and perhaps temporal coherence of the images being generated. Ray casting with spatial subdivision does well on criterion (a), but poorly on criterion (b). Traditional Z-buffer scan conversion does well on criterion (b), but poorly on criterion (a). Here we present a hierarchical Z-buffer scan-conversion algorithm that does well on both criteria. The method uses two hierarchical data structures, an object-space octree and an image-space Z pyramid, to accelerate scan conversion. The two hierarchical data structures make it possible to reject hidden geometry very rapidly while rendering visible geometry with the speed of scan conversion. For animation, the algorithm is also able to exploit temporal coherence. The method is well suited to models with high depth complexity, achieving orders of magnitude acceleration in some cases compared to ordinary Z-buffer scan conversion.

**CR Categories and Subject Descriptors:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Hidden line/surface removal; J.6 [Computer-Aided Engineering]: Computer-Aided I.3.1 [Computer Graphics]: Hardware Architecture - Graphics Processors

**Additional Key Words and Phrases:** Octree, Pyramid, Temporal Coherence, Spatial Coherence, Z Buffer.

## 1 Introduction

Extremely complex geometric databases offer interesting challenges for visibility algorithms. Consider, for example, an interactive walk-through of a detailed geometric database describing an entire city, complete with vegetation, buildings, furniture inside the buildings and the contents of the furniture. Traditional visibility algorithms running on currently available hardware cannot come close to rendering scenes of this complexity at interactive rates and it will be a long time before faster hardware alone will suffice. In order to get the most out of available hardware, we need faster algorithms that exploit properties of the visibility computation itself.

There are at least three types of coherence inherent in the visibility computation which can be exploited to accelerate a visibility algorithm. The first is object-space coherence: in many cases a single computation can resolve the visibility of a collection of objects which are near each other in space. The second is image-space coherence: in many cases a single computation can resolve the visibility of an object covering a collection of pixels. The third is temporal coherence: visibility information from one frame can often be used to accelerate visibility computation for the next frame. Here we present a visibility algorithm which exploits all three of these types of coherence and sometimes achieves orders of magnitude acceleration compared with traditional techniques.

The dominant algorithms in use today for visibility computations are Z-buffer scan conversion and ray-tracing. Since Z buffers do not handle partially transparent surfaces well, we will restrict the discussion to models consisting entirely of opaque surfaces. For these models, only rays from the eye to the first surface are relevant for visibility, so the choice is between Z buffering and ray-casting (ray-tracing with no secondary rays).

Traditional Z buffering makes reasonably good use of image-space coherence in the course of scan conversion. Implementations usually do a set-up computation for each polygon and then an incremental update for each pixel in the polygon. Since the incremental update is typically much less computation than the set-up, the savings from image-space coherence can be substantial. The problem with the traditional Z-buffer approach is that it makes no use at all of object-space or temporal coherence. Each polygon is rendered independently, and no information is saved from prior frames. For extremely complex environments like a model of a city, this is very inefficient. A traditional Z-buffer algorithm, for example, will have to take the time to render every polygon of every object in every drawer of every desk in a building even if the whole building cannot be seen, because the traditional algorithm can resolve visibility only at the pixel level.

Traditional ray-tracing or ray-casting methods, on the other hand, make use of object-space coherence by organizing the objects in some type of spatial subdivision. Rays from the eye are propagated through the spatial subdivision until they hit the first visible surface. Once a ray hits a visible surface, there is no need to consider any of the surfaces in the spatial subdivisions further down along the ray, so large portions of the geometry may never have to be considered during rendering. This is an important improvement on Z buffering, but it makes no use of temporal or image-space coherence. While ray-casting algorithms that exploit temporal coherence have been explored, it seems extremely difficult to exploit image-space coherence in traditional ray casting algorithms.

Here we present a visibility algorithm which combines the strengths of both ray-casting and Z buffering. To exploit object-

*Apple Computer, U.C. Santa Cruz
†Apple Computer



 ACM-0-89791-601-8/93/008/0231 $01.50

space coherence, we use an octree spatial subdivision of the type commonly used to accelerate ray tracing. To exploit image-space coherence, we augment traditional Z-buffer scan conversion with an image-space Z pyramid that allows us to reject hidden geometry very quickly. Finally, to exploit temporal coherence, we use the geometry that was visible in the previous frame to construct a starting point for the algorithm. The result is an algorithm which is orders of magnitude faster than traditional ray-casting or Z buffering for some models we have tried. The algorithm is not difficult to implement and works for arbitrary polygonal databases.

In section II, we survey the most relevant prior work on accelerating ray casting and scan conversion. In section III, we develop the data structures used to exploit object-space, image-space and temporal coherence. In section IV, we describe the implementation and show results for some complex models containing hundreds of millions of polygons.

# 2 Prior Work

There have been many attempts to accelerate traditional ray-tracing and Z buffering techniques. Each of these attempts exploits some aspect of the coherence inherent in the visibility computation itself. None of them, however, simultaneously exploits object-space, image-space and temporal coherence.

The ray-tracing literature abounds with references to object-space coherence. A variety of spatial subdivisions have been used to exploit this coherence and they seem to work quite well (e.g. [1, 2, 3, 4, 5]). Temporal coherence is much less commonly exploited in practice, but various techniques exist for special cases. If all the objects are convex and remain stationary while the camera moves, then there are constraints on the way visibility can change[6] which a ray tracer might exploit. On the other hand, if the camera is stationary, then rays which are unaffected by the motion of objects can be detected and used from the previous frame[7]. When interactivity is not an issue and sufficient memory is available, it can be feasible to render an entire animation sequence at once using spacetime bounding boxes[8, 9]. While these techniques make good use of object-space coherence and sometimes exploit temporal coherence effectively, they unfortunately make little or no use of image-space coherence since each pixel is traced independently from its neighbors. There are heuristic methods which construct estimates of the results of ray-tracing a pixel from the results at nearby pixels (e.g. [10]), but there seems to be no guaranteed algorithm which makes good use of image-space coherence in ray tracing.

With Z-buffer methods (and scan conversion methods in general) the problems are very different. Ordinary Z-buffer rendering is usually implemented with an initial set-up computation for each primitive followed by a scan-conversion phase in which the affected pixels are incrementally updated. This already makes very good use of image-space coherence, so the remaining challenge with Z-buffer methods is to exploit object-space and temporal coherence effectively.

A simple method of using object-space coherence in Z-buffer rendering is to use a spatial subdivision to cull the model to the viewing frustum [11]. While this can provide substantial acceleration, it exploits only a small portion of the object-space coherence in models with high depth complexity. In architectural models, for example, a great deal of geometry hidden behind walls may lie within the viewing frustum.

In order to make use of more of the object-space coherence in architectural models, Airey et. al. [12, 13] and subsequently Teller and Sequin[15] proposed dividing models up into a set of disjoint cells and precomputing the potentially visible set (PVS) of polygons from each cell. In order to render an image from any viewpoint within a cell, only the polygons in the PVS need be considered. These PVS schemes are the closest in spirit to the visibility algorithm presented here since they attempt to make good use of both object-space and image-space coherence. Nonetheless, they suffer from some important limitations. Before they can be used at all, they require an expensive precomputation step to determine the PVS and a great deal of memory to store it. Teller and Sequin, for example, report over 6 hours of precomputation time on a 50 MIP machine to calculate 58Mb of PVS data needed for a model of 250,000 polygons[15]. Perhaps more importantly, the way these methods make use of cells may limit their appropriateness to architectural models. In order to achieve maximum acceleration, the cells must be 3D regions of space which are almost entirely enclosed by occluding surfaces, so that most cells are hidden from most other cells. For architectural models, this often works well since the cells can be rooms, but for outdoor scenes and more general settings, it is unclear whether or not PVS methods are effective. In addition, the currently implemented algorithms make very special use of axially-aligned polygons such as flat walls in rectilinear architectural models. While the methods can in principle be extended to use general 3D polygons for occlusion, the necessary algorithms have much worse computational complexity[15]. Finally, although the implementations prefetch PVS data for nearby cells to avoid long latencies due to paging, they cannot be said to exploit temporal coherence in the visibility computation very effectively.

The algorithm presented here shares a great deal with the work of Meagher[16] who used object-space octrees with image-space quadtrees for rendering purposes. Meagher tried to display the octree itself rather than using it to cull a polygonal database, so his method is directly applicable to volume, rather than surface models. Nonetheless his algorithm is one of the few to make use of both object-space and image-space coherence. The algorithm does not exploit temporal coherence.

# 3 Hierarchical Visibility

The hierarchical Z-buffer visibility algorithm uses an octree spatial subdivision to exploit object-space coherence, a Z pyramid to exploit image-space coherence, and a list of previously visible octree nodes to exploit temporal coherence. While the full value of the algorithm is achieved by using all three of these together, the object-space octree and the image-space Z pyramid can also be used separately. Whether used separately or together, these data structures make it possible to compute the same result as ordinary Z buffering at less computational expense.

## 3.1 Object-space octree

Octrees have been used previously to accelerate ray tracing[5] and rendering of volume data sets[16] with great effectiveness. With some important modification, many of the principles of these previous efforts can be applied to Z-buffer scan conversion. The result is an algorithm which can accelerate Z buffering by orders of magnitude for models with sufficient depth complexity.

In order to be precise about the octree algorithm, let us begin with some simple definitions. We will say that a polygon is hidden with respect to a Z buffer if no pixel of the polygon is closer to the observer than the Z value already in the Z buffer. Similarly, we will say that a cube is hidden with respect to a Z buffer if all of its faces are hidden polygons. Finally, we will call a node of the octree hidden if its associated cube is hidden. Note that these definitions depend on the sampling of the Z buffer. A polygon which is hidden at one Z-buffer resolution may not be hidden at another.

With these definitions, we can state the basic observation that makes it possible to combine Z buffering with an octree spatial subdivision: If a cube is hidden with respect to a Z buffer, then all polygons fully contained in the cube are also hidden. What this means is the following: if we scan convert the faces of an octree cube and find that each pixel of the cube is behind the current surface in the Z buffer, we can safely ignore all the geometry contained in that cube.

From this observation, the basic algorithm is easy to construct. We begin by placing the geometry into an octree, associating each primitive with the smallest enclosing octree cube. Then we start at the root node of the octree and render it using the following recursive steps: First, we check to see if the octree cube intersects the viewing frustum. If not, we are done. If the cube does intersect the viewing frustum, we scan convert the faces of the cube to determine whether or not the whole cube is hidden. If the cube is hidden, we are done. Otherwise, we scan convert any geometry associated with the cube and then recursively render its children in front-to-back order.

We can construct the octree with a simple recursive procedure. Beginning with a root cube large enough to enclose the entire model and the complete list of geometric primitives, we recursively perform the following steps: If the number of primitives is sufficiently small, we associate all of the primitives with the cube and exit. Otherwise, we associate with the cube any primitive which intersects at least one of three axis-aligned planes that bisect the cube. We then subdivide the octree cube and call the procedure recursively with each of the eight child cubes and the portion of the geometry that fits entirely in that cube.

The basic rendering algorithm has some very interesting properties. First of all, it only renders geometry contained in octree nodes which are not hidden. Some of the rendered polygons may be hidden, but all of them are "nearly visible" in the following sense: there is some place we could move the polygon where it would be visible which is no further away than the length of the diagonal of its containing octree cube. This is a big improvement over merely culling to the viewing frustum. In addition, the algorithm does not waste time on irrelevant portions of the octree since it only visits octree nodes whose parents are not hidden. Finally, the algorithm never visits an octree node more than once during rendering. This stands in marked contrast to ray-tracing through an octree where the root node is visited by every pixel and other nodes may be visited tens of thousands of times. As a result of these properties, the basic algorithm culls hidden geometry very efficiently.

A weakness of the basic algorithm is that it associates some small geometric primitives with very large cubes if the primitives happen to intersect the planes which separate the cube's children. A small triangle which crosses the center of the root cube, for example, will have to be rendered anytime the entire model is not hidden. To avoid this behavior, there are two basic choices. One alternative is to clip the problematic small polygons so they fit in much smaller octree cells. This has the disadvantage of increasing the number of primitives in the database. The other alternative is to place some primitives in multiple octree cells. This is the one we have chosen to implement. To do this, we modify the recursive construction of the octree as follows. If we find that a primitive intersects a cube's dividing planes, but is small compared to the cube, then we no longer associate the primitive with the whole cube. Instead we associate it with all of the cube's children that the primitive intersects. Since some primitives are associated with more than one octree node, we can encounter them more than once during rendering. The first time we render them, we mark them as rendered, so we can avoid rendering them more than once in a given frame.

## 3.2 Image-space Z pyramid

The object-space octree allows us to cull large portions of the model at the cost of scan-converting the faces of the octree cubes. Since the cubes may occupy a large number of pixels in the image, this scan conversion can be very expensive. To reduce the cost of determining cube visibility, we use an image-space Z pyramid. In many cases, the Z pyramid makes it possible to conclude very quickly a large polygon is hidden, making it unnecessary to examine the polygon pixel by pixel.

The basic idea of the Z pyramid is to use the original Z buffer as the finest level in the pyramid and then combine four Z values at each level into one Z value at the next coarser level by choosing the farthest Z from the observer. Every entry in the pyramid therefore represents the farthest Z for a square area of the Z buffer. At the coarsest level of the pyramid there is a single Z value which is the farthest Z from the observer in the whole image.

Maintaining the Z pyramid is an easy matter. Every time we modify the Z buffer, we propagate the new Z value through to coarser levels of the pyramid. As soon as we reach a level where the entry in the pyramid is already as far away as the new Z value, we can stop.

In order to use the Z pyramid to test the visibility of a polygon, we find the finest-level sample of the pyramid whose corresponding image region covers the screen-space bounding box of the polygon. If the nearest Z value of the polygon is farther away than this sample in the Z pyramid, we know immediately that the polygon is hidden. We use this basic test to determine the visibility of octree cubes by testing their polygonal faces, and also to test the visibility of model polygons.

While the basic Z-pyramid test can reject a substantial number of polygons, it suffers from a similar difficulty to the basic octree method. Because of the structure of the pyramid regions, a small polygon covering the center of the image will be compared to the Z value at the coarsest level of the pyramid. While the test is still accurate in this case, it is not particularly powerful.

A definitive visibility test can be constructed by applying the basic test recursively through the pyramid. When the basic test fails to show that a polygon is hidden, we go to the next finer level in the pyramid where the previous pyramid region is divided into four quadrants. Here we attempt to prove that the polygon is hidden in each of the quadrants it intersects. For each of these quadrants, we compare the closest Z value of the polygon in the quadrant to the value in the Z pyramid. If the Z-pyramid value is closer, we know the polygon is hidden in the quadrant. If we fail to prove that the primitive is hidden in one of the quadrants, we go to the next finer level of the pyramid for that quadrant and try again. Ultimately, we either prove that the entire polygon is hidden, or we recurse down to the finest level of the pyramid and find a visible pixel. If we find all visible pixels this way, we are performing scan conversion hierarchically.

A potential difficulty with the definitive visibility test is that it can be expensive to compute the closest Z value of the polygon in a quadrant. An alternative is to compare the value in the pyramid to the closest Z value of the entire polygon at each step of the recursion. With this modification, the test is faster and easier to implement, but no longer completely definitive. Ultimately, it will either prove that the entire polygon is hidden, or recurse down to the finest level of the pyramid and find a pixel it cannot prove is hidden. Our current implementation uses this technique. When the test fails to prove that a polygon is hidden, our implementation reverts to ordinary scan conversion to establish the visibility definitively.

### 3.3 Temporal coherence list

Frequently, when we render an image of a complex model using the object-space octree, only a small fraction of the octree cubes are visible. If we render the next frame in an animation, most of the cubes visible in the previous frame will probably still be visible. Some of the cubes visible in the last frame will become hidden and some cubes hidden in the last frame will become visible, but frame-to-frame coherence in most animations ensures that there will be relatively few changes in cube visibility for most frames (except scene changes and camera cuts). We exploit this fact in a very simple way with the hierarchical visibility algorithm. We maintain a list of the visible cubes from the previous frame, the *temporal coherence list*, and simply render all of the geometry on the list, marking the listed cubes as rendered, before commencing the usual algorithm. We then take the resulting Z buffer and use it to form the initial Z pyramid. If there is sufficient frame-to-frame coherence, most of the visible geometry will already be rendered, so the Z-pyramid test will be much more effective than when we start from scratch. The Z-pyramid test will be able to prove with less recursion that octree cubes and model polygons are hidden. As we will see in section IV, this can accelerate the rendering process substantially. After rendering the new frame, we update the temporal coherence list by checking each of the cubes on the list for visibility using the Z-pyramid test. This prevents the temporal coherence list from growing too large over time.

One way of thinking about the temporal coherence strategy is that we begin by guessing the final solution. If our guess is very close to the actual solution, the hierarchical visibility algorithm can use the Z pyramid to verify the portions of the guess which are correct much faster than it can construct them from scratch. Only the portions of the image that it cannot verify as being correct require further processing.

## 4 Implementation and Results

Our initial implementation of the hierarchical visibility algorithm is based on general purpose, portable C code and software scan conversion. This implementation uses the object-space octree, the image-space Z pyramid and the temporal coherence list. Even for relatively simple models the pure software algorithm is faster than traditional software Z buffering, and for complex models the acceleration can be very large.

In order to test the algorithm, we constructed an office module consisting of 15K polygons and then replicated the module in a three dimensional grid. Each module includes a stairway with a large open stairwell making it possible to see parts of the neighboring floors. None of the office walls extends to the ceiling, so from a high enough point in any of the cubicles, it is possible to see parts of most of the other cubicles on the same floor.

For simple models with low depth complexity, the hierarchical visibility method can be expected to take somewhat longer than traditional scan conversion due to the overhead of performing visibility tests on octree cubes and the cost of maintaining a Z pyramid. To measure the algorithm's overhead on simple models, we rendered a single office module consisting of 15K polygons at a viewpoint from which a high proportion of the model was visible. Rendering time for a 512 by 512 image was 1.52 seconds with the hierarchical visibility method and 1.30 seconds with traditional scan conversion, indicating a performance penalty of 17%. When we rendered three instances of the model (45K polygons), the running time was 3.05 seconds for both methods indicating that this level of complexity was the breakeven point for this particular model. Hierarchical visibility rendered nine instances of the same model (105K polygons) in 5.17 seconds, while traditional scan conversion took 7.16 seconds.

The chief value of the hierarchical visibility algorithm is, of course, for scenes of much higher complexity. To illustrate the point, we constructed a 33 by 33 by 33 replication of the office module which consists of 538 million polygons. The model is shown rendered in figure 1. 59.7 million polygons lie in the viewing frustum from this viewpoint, about one tenth of the entire model. Using the hierarchical visibility method, the Z-pyramid test was invoked on 1746 octree cubes and culled about 27% of the polygons in the viewing frustum. The bounding boxes of 687 cubes were scan converted which culled nearly 73% of the model polygons in the viewing frustum, leaving only 83.0K polygons of which 41.2K were front facing (.000076 of the total model) to be scan converted in software. On an SGI Crimson Elan, the entire process took 6.45 seconds. Rendering this model using traditional Z buffering on the Crimson Elan hardware took approximately one hour and fifteen minutes. Rendering it in software on the Crimson would probably take days.

The center left panel of figure 1 shows the depth complexity processed by the algorithm for the image in the upper left. The depth complexity displayed in this image is the number of times each pixel was accessed in a box visibility test or in Z-buffer polygon scan conversion. Note the bright regions corresponding to portions of the image where it is possible to see far into the model; these are regions where the algorithm has to do the most work. In this image, the average depth complexity due to box scans is 7.23, and due to polygon scan-conversion is 2.48 for a total of 9.71. The maximum depth complexity is 124. Dividing the number of times the Z pyramid is accessed by the number of pixels on the screen lets us assign a value of .43 for the "depth complexity" of the Z-pyramid tests. Thus, the total average depth complexity of Z-pyramid tests, box scans and polygon scans is 10.14. Note that this is not the depth complexity of the model itself, but only the depth complexity of the hierarchical visibility computation. Computing the true depth complexity of the scene would require scan converting the entire model of 538 million polygons in software, which we have not done. In the lower left of figure 1, we show the viewing frustum and the octree subdivision. The two long strings of finely divided boxes correspond to the two brightest regions in the depth complexity image. Note that the algorithm is able to prove that large octree nodes in the distance are hidden. In the lower right, we show the Z pyramid for the scene. Even at fairly coarse resolutions, the Z pyramid contains a recognizeable representation of the major occluders in the scene.

The office environment of figure 1 was chosen in part because it is a particularly difficult model for PVS methods. From every office cubicle in this environment, there are points from which almost every other cubicle on the same floor is visible. As a result, if the cubicles were used as cells in a PVS method, the potentially visible set for each cell would have to include nearly all the cells on its floor and many on other floors. Since each floor contains about 4 million polygons, the PVS methods would probably have to render many more polygons than the hierarchical method. In addition, the precomputation time for published PVS methods would be prohibitive for a model of this complexity. This model has 2000 times as many polygons as the model described by Teller and Sequin[15] which required 6 hours of pre-processing.

Admittedly, the replication of a single cell in the model means that it may not be a representative example, but it will be some time before people use models of this complexity without a great deal of instancing. The hierarchical visibility program we used for this example makes use of the replication in only two ways. First, the algorithm does not need to store half a billion polygons in main memory. Second, the algorithm only needs to consider a single cell in constructing the octree. These same simplifications would

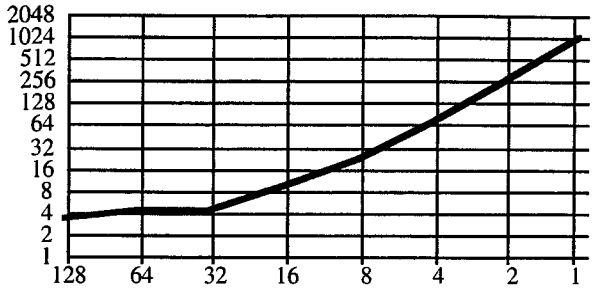


Fig. 3: Total time in seconds to render all windows as a function of the number of pixels on the side of each window.

apply to any complex model using a great deal of instancing.

Figure 2 shows the hierarchical visibility method applied to an outdoor scene consisting of a terrain mesh with vegetation replicated on a two-dimensional grid. The model used for the lower left image consists of 53 million polygons, but only about 25K polygons are visible from this point of view. Most of the model is hidden by the hill or is outside the viewing frustum. The corresponding depth complexity image for hierarchical visibility computations is shown at the top left. The algorithm works hardest near the horizon where cube visibility is most difficult to establish. This frame took 7 seconds to render with software scan conversion on an SGI Crimson. In the lower right, we show a model consisting of 5 million polygons. Even though the model is simpler than the model in the lower left, the image is more complicated and took longer to render because a much larger fraction of the model is visible from this point of view. This image took 40 seconds to render with software scan conversion on an SGI Crimson. The average depth complexity for the scene is 7.27, but it reaches a peak of 85 in the bright areas of the depth complexity image in the upper right. These outdoor scenes have very different characteristics from the building interiors shown in figure 1 and are poorly suited to PVS methods because (a) very few of the polygons are axis-aligned and (b) the cell-to-cell visibility is not nearly as limited as in an architectural interior. Nonetheless, the hierarchical visibility algorithm continues to work effectively.

## 4.1 Parallelizability and Image-space coherence

We have made our hierarchical visibility implementation capable of dividing the image into a grid of smaller windows, rendering them individually and compositing them into a final image. The performance of the algorithm as the window size is varied tells us about the parallel performance of the algorithm and the extent to which it makes use of image-space coherence. If, like most ray tracers, the algorithm made no use of image-space coherence, we could render each pixel separately at no extra cost. Then it would be fully parallelizable. At the other extreme, if the algorithm made the best possible use of image-space coherence, it would render a sizeable region of pixels with only a small amount more computation than required to render a single pixel. Then it would be difficult to parallelize. Note that if we shrink the window size down to a single pixel, the hierarchical visibility algorithm becomes a ray caster using an octree subdivision.

Figure 3 graphs the rendering time for a frame from a walk-through of the model shown in figure 1 as a function of the window size. For window sizes from 32 by 32 on up, the curve is relatively flat, indicating that the algorithm should parallelize fairly well. For window sizes below 32 by 32, however, the slope of the curve indicates that the time to render a window is almost independent of the window size. The algorithm can, for example, render a 32 by 32 region for only slightly more than four times the computational expense of ray-casting a single pixel with this algorithm. Comparing the single pixel window time to the time for the whole image, we find that image-space coherence is responsible for a factor of almost 300 in running time for this example.

## 4.2 Use of graphics hardware

In addition to the pure software implementation, we have attempted to modify the algorithm to make the best possible use of available commercial hardware graphics accelerators. This raises some difficult challenges because the hierarchical visibility algorithm makes slightly different demands of scan-conversion hardware than traditional Z buffering. In particular, the use of octree object-space coherence depends on being able to determine quickly whether any pixel of a polygon would be visible if it were scan converted. Unfortunately, the commercial hardware graphics pipelines we have examined are either unable to answer this query at all, or take milliseconds to answer it. One would certainly expect some delay in getting information back from a graphics pipeline, but hardware designed with this type of query in mind should be able to return a result in microseconds rather than milliseconds.

We have implemented the object-space octree on a Kubota Pacific Titan 3000 workstation with Denali GB graphics hardware. The Denali supports an unusual graphics library call which determines whether or not any pixels in a set of polygons are visible given the current Z buffer. We use this "Z query" feature to determine the visibility of octree cubes. The cost of a Z query depends on the screen size of the cube, and it can take up to several milliseconds to determine whether or not a cube is visible. Our implementation makes no use of the Z pyramid because the cost of getting the required data to and from the Z buffer would exceed any possible savings. On a walk-through of a version of the office model with 1.9 million polygons, the Titan took an average of .54 seconds per frame to render 512 by 512 images. Because of the cost of doing the Z query, we only tested visibility of octree cubes containing at least eight hundred polygons. Even so, 36.5% of the running time was taken up by Z queries. If Z query were faster, we could use it effectively on octree cubes containing many fewer polygons and achieve substantial further acceleration. The Titan implementation has not been fully optimized for the Denali hardware and makes no use of temporal coherence, so these performance figures should be considered only suggestive of the machine's capabilities.

The other implementation we have that makes use of graphics hardware runs on SGI workstations. On these workstations, there is no way to inquire whether or not a polygon is visible without rendering it, so we use a hybrid hardware/software strategy. We do the first frame of a sequence entirely with software. On the second frame, we render everything on the temporal coherence list with the hardware pipeline. Then we read the image and the Z buffer from the hardware, form a Z pyramid and continue on in software. With this implementation, on the models we have tried, temporal coherence typically reduces the running time by a factor of between 1.5 and 2.

In the course of a walk-through of our office model, we rendered the frame in the upper left of figure 1 without temporal coherence, and then the next frame shown in the upper right of figure 1 using temporal coherence. The new polygons rendered in software are shown in magenta for illustration. For the most part, these are polygons that came into view as a result of panning the camera. The center right shows the depth complexity of the hierarchical computation for this frame. The image is much darker in most regions because the algorithm has much less work to do given the previous frame as a starting point. This temporal coherence frame took 3.96 seconds to render on a Crimson Elan, as compared with 6.45 seconds to render the same frame without temporal coherence.

Current graphics accelerators are not designed to support the rapid feedback from the pipeline needed to realize the full potential of octree culling in the hierarchical visibility algorithm. Hardware designed to take full advantage of the algorithm, however, could make it possible to interact very effectively with extremely complex environments as long as only a manageable number of the polygons are visible from any point of view. The octree subdivision, the Z pyramid and the temporal coherence strategy are all suitable for hardware implementation.

# 5 Conclusion

As more and more complex models become commonplace in computer graphics, it becomes increasingly important to exploit the available coherence in the visibility computation. Here we present an algorithm which combines the ability to profit from image-space coherence of Z-buffer scan conversion with the ability of ray tracing to avoid considering hidden geometry. It appears to be the first practical algorithm which materially profits from object-space, image-space and temporal coherence simultaneously. The algorithm has been tested and shown to work effectively on indoor and outdoor scenes with up to half a billion polygons.

The hierarchical visibility algorithm can make use of existing graphics accelerators without modification. Small changes in the design of graphics accelerators, however, would make a large difference in the performance of the algorithm. We hope that the appeal of this algorithm will induce hardware designers to alter future graphics hardware to facilitate hierarchical visibility computations.

# Acknowledgements

We thank Frank Crow and the Advanced Technology Group at Apple Computer for supporting this research. We also thank Mike Toelle, Avi Bleiweiss, Helga Thorvaldsdottir and Mike Keller of Kubota Pacific Corporation for helping us test our algorithm on a Titan workstation.

# References

[1] S. M. Rubin and T. Whitted. A 3-dimensional representation for fast rendering of complex scenes. *Computer Graphics*, 14(3):110–116, July 1980.

[2] A. Glassner. Space subdivision for fast ray tracing. *IEEE CG&A*, 4(10):15–22, Oct. 1984.

[3] D. Jevans and B. Wyvill. Adaptive voxel subdivision for ray tracing. *Proc. Graphics Interface '89*, 164–172, June 1989.

[4] T. Kay and J. Kajiya. Ray tracing complex surfaces. *Computer Graphics*, 20(4):269–278, Aug. 1986.

[5] M. Kaplan. The use of spatial coherence in ray tracing. In *Techniques for Computer Graphics, etc.*, D. Rogers and R. A. Earnshaw, Springer-Verlag, New York, 1987.

[6] H. Hubschman and S. W. Zucker. Frame to frame coherence and the hidden surface computation: constraints for a convex world. *ACM TOG*, 1(2):129–162, April 1982.

[7] D. Jevans. Object space temporal coherence for ray tracing. *Proc. Graphics Interface '92*, Vancouver, B.C., 176–183, May 11-15, 1992.

[8] A. Glassner. Spacetime ray tracing for animation. *IEEE CG&A*, 8(3):60–70, March 1988.

[9] J. Chapman, T. W. Calvert, and J. Dill. Spatio-temporal coherence in ray tracing. *Proceedings of Graphics Interface '90*, 196–204, 1990.

[10] S. Badt, Jr. Two algorithms for taking advantage of temporal coherence in ray tracing *The Visual Computer*, 4:123–132, 1988.

[11] B. Garlick, D. Baum, and J. Winget. Interactive viewing of large geometric databases using multiprocessor graphics workstations. *SIGGRAPH '90 Course Notes: Parallel Algorithms and Architectures for 3D Image Generation*, 1990.

[12] J. Airey. Increasing update rates in the building walkthrough system with automatic model-space subdivision. Technical Report TR90-027, The University of North Carolina at Chapel Hill, Department of Computer Science, 1990.

[13] J. Airey, J. Rohlf, and F. Brooks. Towards image realism with interactive update rates in complex virtual building environments. *ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics*, 24(2):41–50, 1990.

[14] S. Teller and C. Sequin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 25(4):61–69, 1991.

[15] S. Teller and C. Sequin. Visibility computations in polyhedral three-dimensional environments. U.C. Berkeley Report No. UCB/CSD 92/680, April 1992.

[16] D. Meagher. Efficient synthetic image generation of arbitrary 3-D objects. *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, 473–478, June 1982.

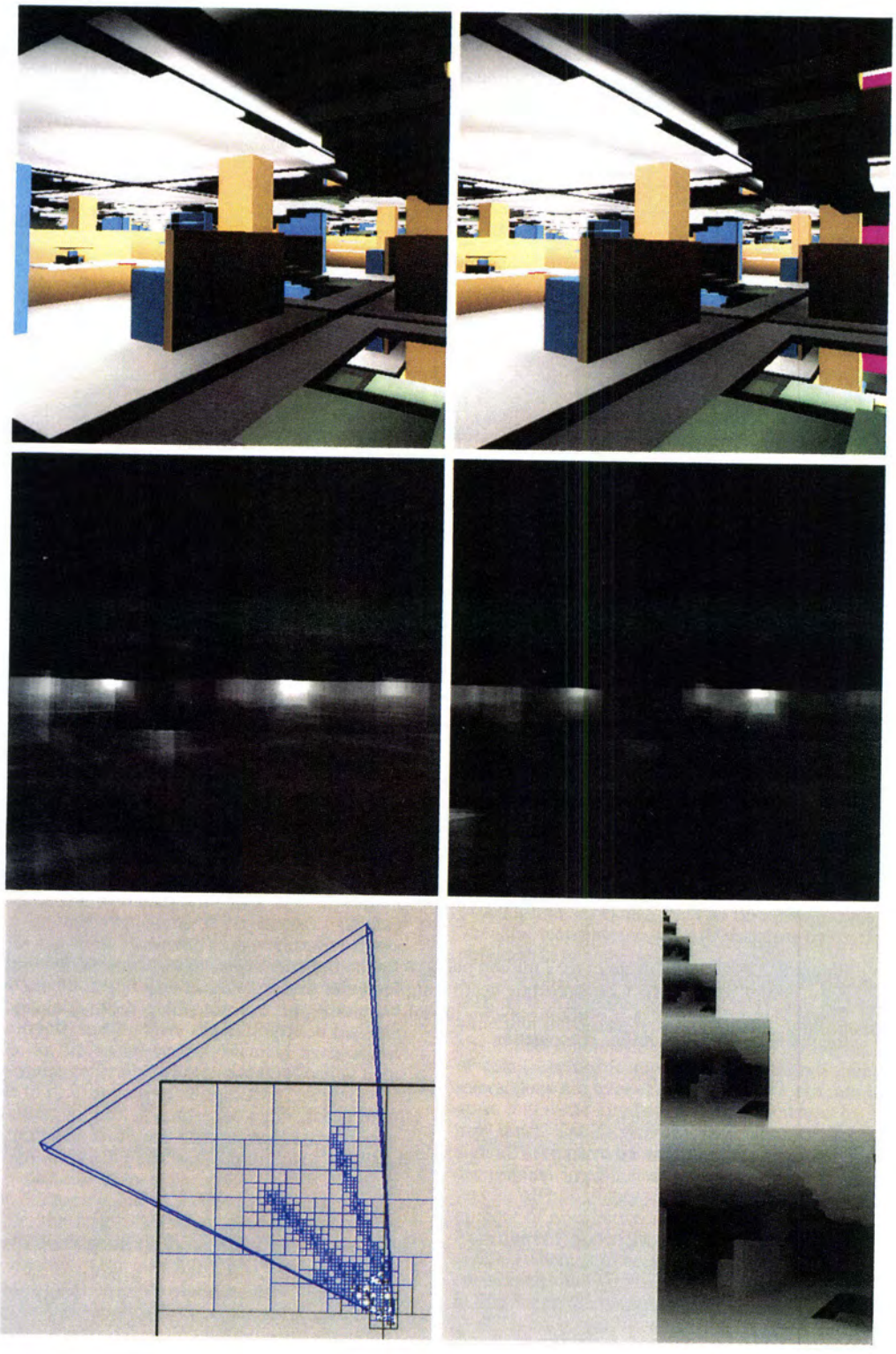


Figure 1: A 538 million polygon office environment rendered with hierarchical visibility. Upper left: Rendered image. Center left: Depth complexity of the hierarchical visibility computation. Lower Left: Viewing frustum and octree cubes examined while rendering the image in the upper left. Lower right: Z pyramid used to cull hidden geometry. Upper right: Image rendered with temporal coherence. Polygons not rendered in the previous frame are shown in magenta. Center right: Depth complexity of the hierarchical visibility computation for the frame rendered using temporal coherence.

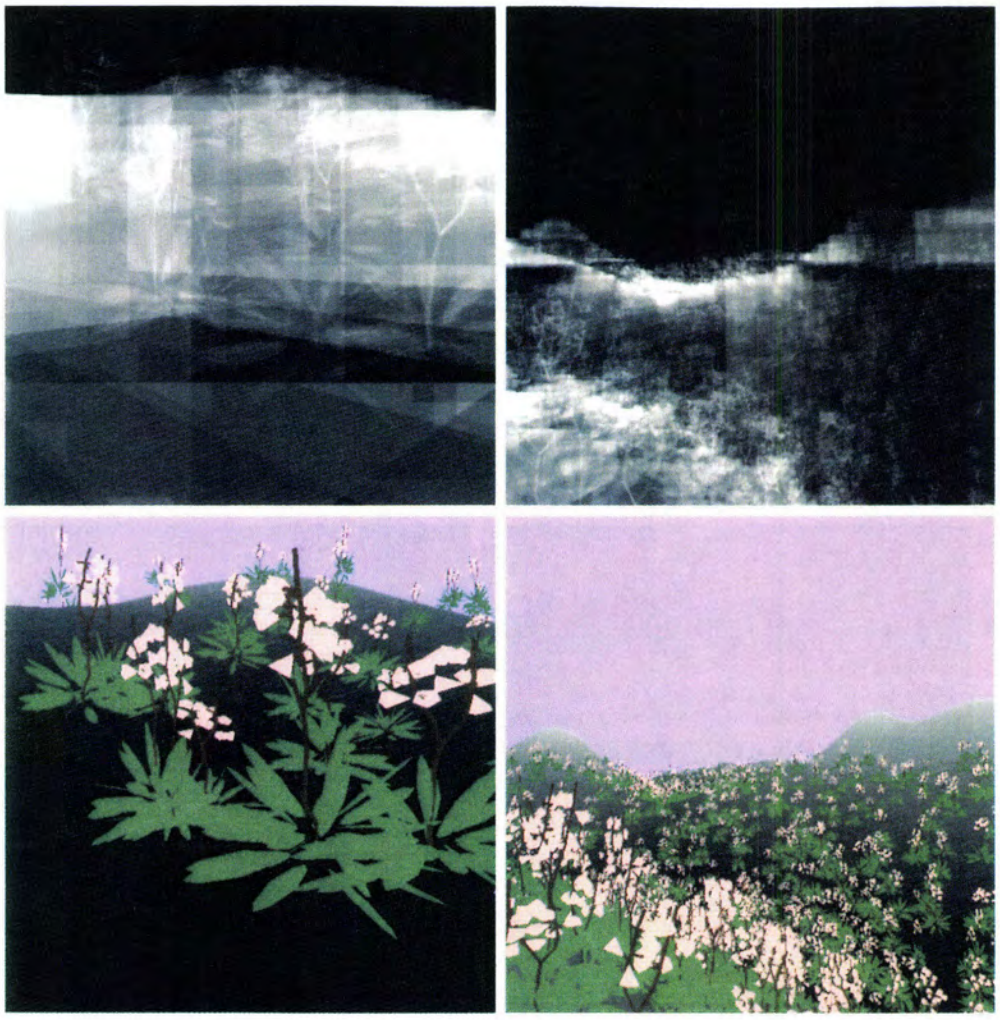


Figure 2: Lower left: Image of a 53 million polygon model (mostly hidden) rendered using hierarchical visibility. Upper left: Corresponding depth complexity for the hierarchical visibility computation. Lower right: Image of a 5 million polygon model. Upper right: Corresponding depth complexity for the hierarchical visibility computation.

# Global Visibility Algorithms for Illumination Computations

Seth Teller
Institute of Computer Science
Hebrew University of Jerusalem

Pat Hanrahan
Department of Computer Science
Princeton University

## Abstract

The most expensive geometric operation in image synthesis is visibility determination. Classically this is solved with hidden surface removal algorithms that render only the parts of the scene visible from a point. Global illumination calculations, however, may require information between any two points in the scene. This paper describes global visibility algorithms that preprocess polygon databases in order to accelerate visibility determination during illumination calculations. These algorithms are sensitive to the output complexity in visibility space; that is, how many pairs of objects are mutually visible. Furthermore, the algorithms are incremental so that they work well with progressive refinement and hierarchical methods of image synthesis. The algorithms are conservative, but exact; that is, when they return visibility predicates they can be proved true. However sometimes they do not return either totally visible or totally invisible, but partially visible, even though in the same situation a better algorithm might return the exact answer. In this paper we describe the algorithms and their implementation, and show that, in a scene with low average visual complexity, they can dramatically accelerate conventional radiosity programs.

**CR Categories and Subject Descriptors:** I.3.5 [Computational Geometry and Object Modeling]: *Geometric Algorithms, Languages, and Systems*; I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism – Radiosity*; J.2 [Physical Sciences and Engineering]: *Engineering*.

**Additional Key Words:** Hidden surface removal, visibility space, radiosity, global illumination, algorithmic triage.

## 1 Introduction

In the early days of image synthesis a central geometric problem was hidden surface removal. With the advent of $z$-buffering, modern workstations can display pictures of 3D scenes containing millions of polygons in real-time. However, such workstations have limited shading capabilities because they make the assumption that all light sources illuminate every object. One major thrust of current research in image synthesis is to remove this restriction so that the shading correctly accounts for the illumination incident on every object. To do this every surface element must assess what light sources, or more generally, what surfaces reflecting light towards it, are visible to it. This type of illumination calculation is termed global, in contrast to local, because the entire scene must be analyzed to determine if there are any occluders interfering with the transfer of light between objects. Collating such visibility information is more difficult than determining merely what is visible from a single vantage point, as is done in hidden surface removal. For example, the fastest algorithm currently known for computing a complete description of the interocclusion due to a polyhedral object of $n$ vertices can take $O(n^6 \lg n)$ time [9].

This paper describes global visibility algorithms that analyze the entire visibility space, and are applicable to a range of illumination problems. Here, we apply them to a hierarchical radiosity algorithm. We have implemented several practical algorithms, and show that they allow efficient global visibility calculations for scenes of low visual complexity. The algorithms are based on three simple ideas:



**Visibility preprocessing.** To compute what is visible from all points on the surfaces of the objects being shaded, we preprocess the scene to speed future visibility tests. For the purposes of global illumination we need only consider all pairwise interactions between objects. Preprocessing removes totally invisible pairs from consideration, and accelerates later queries regarding visibility between points on partially visible pairs.

**Incremental visibility maintenance.** The most efficient global illumination algorithms operate iteratively based on error criteria. Examples are hierarchical radiosity, where surfaces are subdivided with respect to each other according to potential light transfers between them [11], and progressive refinement methods where light is transferred among surfaces in order of brightness [5]. Thus, the visibility algorithms should be lazy and sensitive to required precision. They should also allow refinement so that more precise determinations can be made as needed.

**Conservative triage.** Both the preprocessing and maintenance methods use conservative triage to avoid the combinatorial complexity of exact visibility determination. We classify visibility into three categories: totally INVISIBLE, totally VISIBLE, and PARTIAL (partially visible). The classification is conservative in that all interactions classified as INVISIBLE or VISIBLE are correct; however, it is acceptable for the classification to return PARTIAL when the correct result is either VISIBLE or INVISIBLE. This allows us to forego complex analysis or "punt" if such analysis will take too long to determine the exact answer. Of course, for this to work we need either another visibility algorithm to complete the analysis, or we must expect the situation to simplify eventually (e.g., through subdivision).

The visibility algorithms presented here generalize previous work on preprocessing environments for interactive walkthroughs. In [24], an algorithm was given to preprocess a 2D environment of axial line segments, such as floorplans. This was extended to 3D axial rectangles in [22]. This paper treats the case of convex polygons in general position.

The global visibility algorithms described here have been implemented with a global illumination system that computes radiosity values for polygonal scenes [11]. The algorithm maintains a hierarchy of interactions between subdivided polygons at different levels of detail. A key feature of the algorithm is that only $O(k^2 + n)$ interactions are ever examined (with $k$ the number of input polygons, and $n$ the number of elements created by subdividing those polygons). The hierarchical radiosity algorithm, as originally designed, used pairwise visibility information between polygons. In the original implementation, however, this visibility information was *inexact*. Visibility status was determined by shooting a constant number of rays between two polygons. If all of the rays reached from one polygon to the other, the polygons were considered totally visible, whereas if none of the rays reached, the polygons were considered totally invisible. The conservative algorithms described in this paper, in contrast, are provably more precise.

## 2 Overview

We present novel algorithms that subdivide space, construct a *conservative visibility graph* over the polygons in a geometric model, then maintain the correctness of the graph under recursive subdivision of the polygons. In the context of the hierarchical radiosity computation, this conservative visibility graph guarantees that throughout the computation, all polygons that potentially interact (e.g., exchange energy) will be known. The construction and maintenance of the graph occurs in four stages.

**1. Spatial Subdivision.** The geometric model is first *spatially subdivided* into convex polyhedral cells, linked across shared boundaries only when some *portal*, or transparent region, exists on the boundary. For a large class of models, and particularly for architectural models, the subdivision proves a natural way of hierarchically capturing the geometric and occlusive characteristics of the model.

**2. Visibility Propagation.** Each cell of the spatial subdivision encloses some portion of the geometric model. Clearly, only when two *cells* are mutually visible can their contents (i.e., model polygons) interact. Consequently, we hierarchically enumerate all visibility between portions of the model by first establishing *inter-cell* visibility, then establishing *inter-polygon* visibility only where cells are mutually visible. This is accomplished by *propagating* incremental visibility information through the cells of the spatial subdivision; as each cell "sees" into increasingly distant cells, the visibility graph is augmented to record any previously unknown interactions. (Portal enumeration is simply the first and crudest record of visibility propagation.)

Visibility propagation provably discovers all partially or totally visible cell (polygon) pairs, at the cost of occasionally misclassifying an invisible cell (polygon) pair as visible. The alternative, misclassification of some mutually visible interaction as invisible, is plainly unacceptable, since it may omit from consideration an interaction later to prove important.

**3. Blocker Detection.** In an exacting illumination computation such as global illumination, it is not sufficient to determine simply that two polygons are partially visible; some estimation must be made of the extent to which they are visible, as well as how much error might be incurred by the estimation. Therefore, once potential visibility between a pair of polygons is established, a set of *interfering polygons* or *blockers* is determined that may occlude part of one polygon as seen from some point on the other. This interference computation is again *conservative*; a non-interfering polygon may occasionally be classified as a blocker, but a blocker will *never* be classified as non-interfering. In the visibility graph, *blocker lists* augment existing links between mutually visible polygons; total visibility is established whenever the blocker list is empty. Perhaps surprisingly, we show that these conservative overestimated blocker lists are generally *smaller* than those maintained by existing algorithms.

**4. Blocker Maintenance.** In a hierarchical radiosity algorithm, polygons (*patches*, in radiosity parlance) are allowed to exchange radiant energy only when the interaction satisfies some specified global error bound [11]. Otherwise, the patches are subdivided, and interaction is recommenced among the child patches. It is natural to consider how the conservative visibility graph among the patches can be *incrementally maintained* under subdivision. Each child patch may be partially or totally visible, or completely invisible, to its child counterparts on the other polygon. We show how, given the parent interaction, conservative blocker lists for the children can be determined incrementally. We present a novel blocker maintenance technique involving *linespace*, a five-dimensional representation of 3D lines (i.e., light rays).

The algorithms we present are of interest in several ways. First, they comprise a practical treatment of visibility issues for unrestricted (i.e., non-axial) three-dimensional environments, in contrast to previous work [1, 8, 24]. Second, the conservative visibility description we compute – identification of all mutually visible pairs, and the blocker set for each pair – is a natural, output-sensitive way of characterizing visibility among polygons or more general objects, for any algorithms that require information about occlusion and/or illumination. Finally, we show that the use of these algorithms dramatically improves the time and space efficiency of an existing radiosity computation [11].

## 3 Spatial Subdivision

The geometric model is specified as a set $P$ of convex polygons (Figure 1). The space embedding the geometric model is subdivided into convex polyhedral *cells*, typically separated by polygons (Figure 2). The construction is based on BSP trees [7], but the visibility algorithms we subsequently present are provably correct for any spatial subdivision satisfying a few geometric criteria [22].

**Figure 1**: A geometric model comprised of polygons.

First, a polyhedral *root volume* is constructed as the convex hull of $P$. While polygons of sufficient size are present, a polygon is chosen whose support plane is to partition the remainder of the set. The choice is made using a simple heuristic that determines the polygon whose cross-section in the current cell is largest, when expressed as a fraction of the cell's areal intersection with the polygon's support plane; if any polygons separate the cell into mutually invisible parts, one such polygon is chosen. This heuristic tends to yield effective splitting trees in practice.

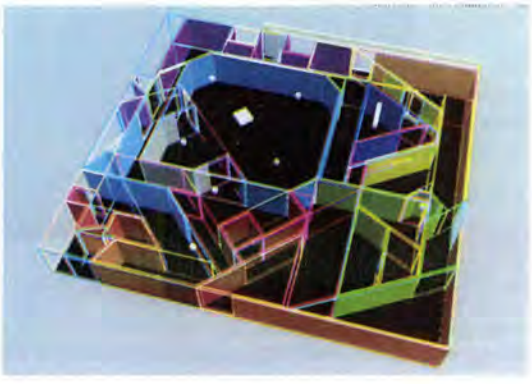


**Figure 2**: Subdivision of the model into cells and portals.

Next, the *portals*, or transparent portions, of each cell boundary are explicitly constructed. Since cell boundaries are induced on the support planes of polygons, these boundaries are typically partially or completely obscured. Each boundary stores a list of coaffine and incident polygons. The portals on each boundary comprise a convex decomposition of the set difference of the cell boundary with the union of these polygons. Each portal stores identifiers for the cells which it connects. The spatial subdivision therefore comprises an *adjacency graph* over the cells, since two cells are adjacent in this graph iff they share a boundary that is not completely opaque.

## 4 Visibility Propagation

Once a conforming spatial subdivision is built, visibility propagation commences. The propagation algorithm operates in object space, and performs a *constrained traversal* of the adjacency graph outward from each *source* cell. Whenever a cell is reached by this traversal, its associated polygons are examined pairwise with those in the source for mutual visibility; unreached entities are definitely invisible from the source.

A given cell can see into its neighbors only through portals, and into more distant cells only through *portal sequences*; i.e., ordered lists of portals such that each consecutive pair of portals lead into and out of the same cell. The cell adjacency graph is searched by

determining cells between which an unobstructed *sightline* exists. A sightline must be disjoint from any occluders and thus must intersect, or *stab*, a portal in order to pass from one cell to the next. To establish inter-cell visibility, it is sufficient to find a stabbing line through a particular portal sequence; since if some point in the *interior* of one cell can see a point in the interior of another, a sightline must exist between the boundaries of the source cell and the reached cell.

Thus, the problem of finding sightlines between cell interiors reduces to finding sightlines through portal sequences of increasing length. Consequently, the primitive visibility operation in a conforming spatial subdivision is the determination of a stabbing line, given a portal sequence, or the determination that no such stabbing line exists. The portal sequences are generated incrementally by a depth-first search (DFS) emanating from a particular cell boundary; when a sequence no longer admits a sightline, the active branch of the DFS terminates.

### 4.1 Inter-Cell Visibility

Sightline determination is an *existence* predicate, in that it merely establishes visibility between two points on different cells. Suppose two cells are mutually visible through a portal sequence. In general, only a portion of each cell is visible to the other, due to occlusion by opaque material abutting the edges of intervening portals (Figures 3, 4). Whenever inter-cell visibility is established, mutually visible *volumes* are constructed for the cell pair; these volumes and the reaching portal sequence are then used to determine inter-polygon visibility among the cells' associated polygons.

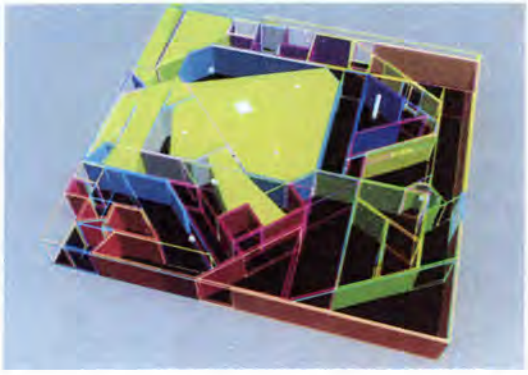


**Figure 3**: Visibility propagation from a source cell $S_1$.

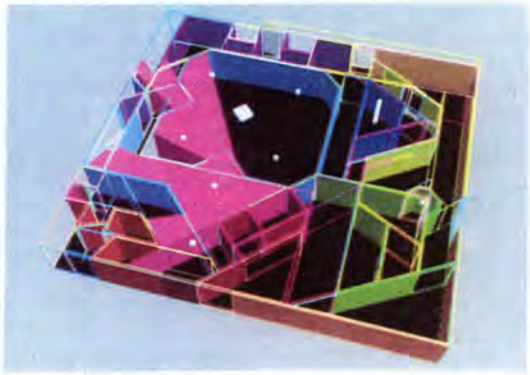


**Figure 4**: Visibility propagation from a source cell $S_2$.

The volume visible to a polygon in the presence of polygonal occluders is, in general, bounded by quadratic surfaces [18]. An algorithm for computing this volume was implemented and described in [21], but is not yet sufficiently robust for use on complex models. Consequently, we have developed a simpler algorithm that computes a polyhedral volume guaranteed to enclose the exact visible region. The algorithm is a straightforward construction that, using separating tangent planes, performs a kind of internal pivoting over the edges and vertices occurring along the portal sequence. We treat the algorithm briefly here; details can be found in [22].

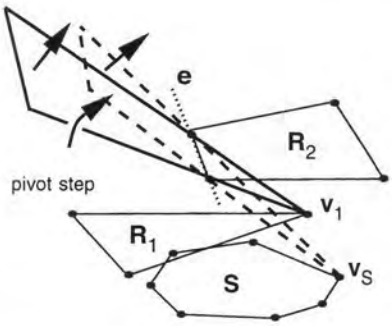



**Figure 5**: A pivoting step on edge $\mathbf{e}$, from $\mathbf{v}_1$ to $\mathbf{v}_S$.

The algorithm exploits the fact that for each portal edge, at most two separating planes can contribute a face to polyhedral bounds on the illuminated volume (Figure 5), since at most one vertex from each halfspace of the associated portal can span a relevant plane with the edge. Consider some edge $\mathbf{e}$ on a portal $\mathbf{R}_2$, and the portals occurring before $\mathbf{R}_2$. Each of these portals has at most one *extremal* vertex that spans a separating plane with $\mathbf{e}$ (in the figure, $\mathbf{R}_1$ has extremal vertex $\mathbf{v}_1$ and $\mathbf{S}$ has extremal vertex $\mathbf{v}_S$). Together with $\mathbf{e}$, only one of these (at most $\frac{n}{3}$) extremal vertices can span a plane that contains all the other extremal vertices in the same halfspace as the portal $\mathbf{R}_2$. This single plane is the only one of the $n$ candidate planes that can contribute faces to the boundary of the illuminated volume. Therefore, for any $n$ portal edges there are at most $2n$ boundary planes, each of which can be identified in $O(n)$ time by pivoting over the vertices of the other portals. The total time to identify the $2n$ relevant planes is therefore $O(n^2)$. Moreover, the set of $O(n)$ planes can be updated incrementally whenever a new portal is encountered, simply by updating the existing halfspaces with respect to the new portal vertices, and introducing planes tight on the new portal edges. The $O(n)$ positive halfspaces of the planes are inspected for an intersection with the $c$ BSP halfspaces bounding the reached cell in time $O(n+c)$ with a linear programming algorithm [15, 19]. If no such intersection exists then the reached cell can not be visible to the source through the active portal sequence.

### 4.2 Inter-Polygon Visibility

Whenever a cell is reached by the graph propagation, an *active set* of halfspaces bounds the volume in the reached cell visible to the source. The orientations of each of these halfspaces are reversed to bound the volume illuminated by the *reached* cell in the *source*. Only polygons in these respective volumes can be mutually visible. Each incident source polygon is prefixed to the front of the active portal sequence (Figure 6). The visible volume in the reached cell due to the augmented sequence is then tested for incidence with the appropriate subset of polygons stored in the reached cell. (The notion of conservative inter-polygon visibility can be simply extended to treat visibility between general objects [22].)

Figure 6 depicts this mechanism for an analogous 2D situation, in which "polygons" and "portals" are line segments. A source cell $S$ (Figure 6-i) establishes inter-cell visibility to a cell $R$ via some portal sequence. The polygon $B$ in $S$ can have no interaction with $R$'s interior, and it is not considered further. Polygon $C$ is incident on the inter-cell visibility volume, and therefore potentially visible from some point in $S$. However, when the portal sequence is augmented with the constraints due to $A$ (Figure 6-ii), polygon $C$ is found to be invisible from $A$. Finally, $D$ is found to intersect $A$'s visible region in $R$, and $A$ and $D$ are established to be mutually

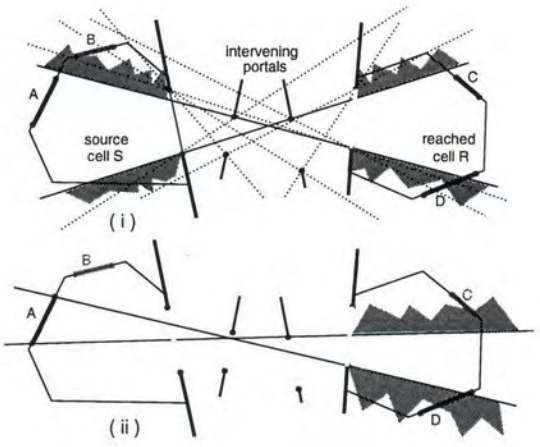



**Figure 6**: Establishing visibility of 2D "polygons" $A$ and $D$.

visible.

The convexity of spatial subdivision cells allows an important optimization. Any two polygons entirely incident on the boundaries of the same cell can have blockers only in the relative interior of that cell. When the cell interior is empty (as it typically will be), the polygons can be immediately classified as entirely mutually visible. Thus, the spatial subdivision quickly identifies many instances of complete mutual visibility between nearby polygons.

Figures 7 and 8 depict the output of the inter-polygon visibility computation in three dimensions, for two polygons incident on different source cells. Display of the spatial subdivision has been suppressed for clarity.

**Figure 7**: Visibility propagation from a polygon in cell $S_1$.

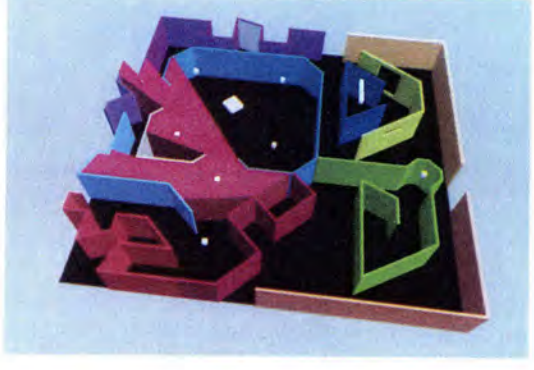


**Figure 8**: Visibility propagation from a polygon in cell $S_2$.

## 5 Blocker Detection

When a pair $(S, R)$ of polygons is found to be mutually visible, we record a visibility *interaction* $I(S, R)$, and proceed to identify the *blocker list* $B(S, R)$ of the pair. One could simply compute the set of blockers as those polygons incident on a convex volume containing $S$ and $R$ (as in [10]). However, the visibility graph and reaching portal sequence generally yield a better (i.e., smaller) blocker list. Denote the convex hull of all vertices of $S$ and $R$ as $conv(S, R)$. Clearly any blocker $B$ must be incident on $conv(S, R)$ to contribute to $B(S, R)$. Moreover, observe that *only polygons visible to $S$ along a sequence reaching $R$, or to $R$ along a sequence reaching $S$* need be considered as blockers of $S$ and $R$. For, if some polygon $C$ is *not* visible to $S$, then *every* ray leaving $S$ (including those rays to any point on $R$) must stab some polygon other than $C$ before stabbing $C$.

The polygons $S$ and $R$ do not generally *see* the same set of blockers (Figures 7 and 8). Therefore, $B(S, R)$ is augmented whenever a search from $S$ ($R$) to $R$ ($S$) discovers a previously unknown blocker. Figure 9 depicts the result of the blocker computation, where all polygons except $S$, $R$, and $B(S, R)$ have been removed. Note that, of the polygons from the large central room, neither the large blue interior wall panels nor the thin blue doorjambs (cf. Figure 1) are classified as blockers. Thus the purely spatial (shaft) cull produces a blocker list of size 12 or more, whereas the blocker detection algorithm presented here computes a list of 6 blockers.

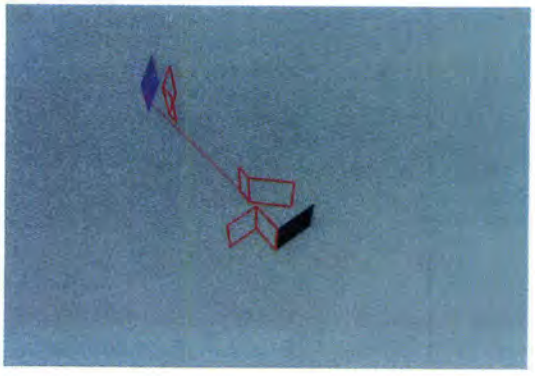


**Figure 9**: The final blocker list of $S$ and $R$.

Finally, the blocker criterion presented above is conservative, since $B(S, R)$ may include polygons that are visible to $S$ or to $R$ but do not affect occlusion between them. The *exact* determination of the blocker list is computationally involved; a polygon $B$ is a blocker of $S$ and $R$ only if some ray from $S$ to $R$ exists whose only front-facing polygon intersection, aside from that with $R$, is with $B$. (The asymmetry of the definition arises from the fact that, in a manifold polyhedral environment of oriented polygons, only front faces can be visible to front faces.)

## 6 Blocker Maintenance

Given a set of polygons $P$, the visibility preprocessing scheme produces, for every polygon $S \in P$, a set of visible polygons $V(S)$. For each polygon $R \in V(S)$, the *blocker list* $B(S, R)$ enumerates all polygons $B \in P \setminus \{S, R\}$ that potentially impede visibility between $S$ and $R$. $B(S, R)$ points only to top-level patches; this makes sense, since blockers should be as large as possible to cause maximal occlusion. For each interaction $I(S, R)$, we store a *tube* data structure, which associates an interacting patch pair, a blocker list, the *visibility status* $V(S, R)$ of the interaction (i.e., VISIBLE or PARTIAL), and some additional geometric information used for incremental visibility tests.

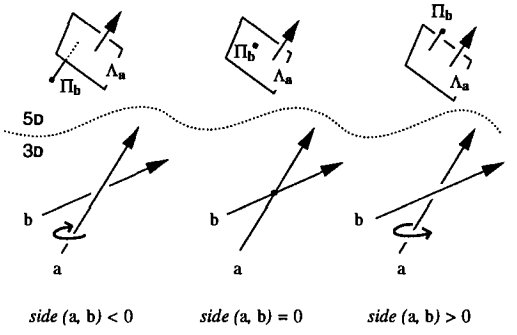
In the hierarchical radiosity algorithm, when the energetic interaction between two patches can not be characterized to within the global error bound, one of the patches of the interaction is symmetrically *subdivided*, and its children are allowed to interact with the other patch [11]. Clearly, interactions between either

patch and the children of its counterpart are highly coherent. The tube data structure exploits this coherence to perform efficient and accurate visibility reclassification after subdivision.

Each child interaction's blocker list is necessarily a subset of the parent's blocker list; we wish to efficiently, and incrementally, determine the child tube's blockers. We say that a blocker $B$ *impinges* on $I(S, R)$ if it occludes $S$ from $R$, and that $B$ is *disjoint* from $I(S, R)$ if $B$ can not cause occlusion. Whenever a blocker list is discovered to be *empty* (i.e., to contain no impinging blockers), complete visibility between the interacting patches will be established, and no further visibility computations need be done for any children of this interaction. Conversely, whenever the blocker list is discovered to be *completely occluding*, there can be no energy transport between $S$ and $R$, and the interaction is discarded (alternatively, the culprit blocker(s) can be retained as "proof" that the patches cannot interact). Finally, when neither complete visibility nor complete occlusion can be quickly determined, the status of the child interaction remains partially visible.

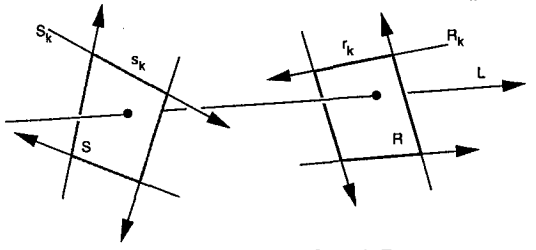
## 6.1 Linespace

The tube structure efficiently encodes the set of all lines between $S$ and $R$, using a five-dimensional line representation known as Plucker coordinates [20], or simply *linespace*. Lines in three dimensions correspond to hyperplanes and points in linespace. Any two 3D rays $a$ and $b$ can be *oriented* by considering their linespace counterparts $\Lambda_a$, a 5D hyperplane, and $\Pi_b$, a 5D point (details of the mapping can be found in [21]).




The signed distance of $\Pi_b$ from $\Lambda_a$ determines the sense in which the lines "go around" each other in 3D; if $\Pi_b$ lies on $\Lambda_a$ the lines $a$ and $b$ are coplanar. This "sidedness" property can be used to represent the set of lines through a collection of convex polygons. In practice, there is one caveat to using the linespace representation [23]. The only portion of linespace corresponding to 3D lines with real coefficients are those linespace points lying on a 4D manifold known as the Plucker quadric [20]; all other linespace points correspond to 3D lines with complex coefficients. Fortunately, the algorithms used in this paper need never consider the Plucker quadric, since they manipulate only lines known *a priori* to have real coefficients.

Consider two convex polygons $S$ and $R$, comprised of sets of oriented edges $S_k$ and $R_k$, respectively. For there to exist some line $\mathbf{L}$ that stabs the interiors of $S$ and $R$, $\Pi_{\mathbf{L}}$ must lie in the appropriate signed halfspaces $h_k$ of the hyperplanes $\Lambda_{\mathbf{S}_k}$ and $\Lambda_{\mathbf{R}_k}$.




Thus, the set of all lines through $S$ and $R$ corresponds to the interior of a five-dimensional convex polytope $\cap_k h_k$ [21]. Rather than attempt to compute this polytope directly, we can manipulate the vertices of its intersection with the Plucker quadric, which are comparatively easy to generate. Each such vertex corresponds to a collection of four support lines from $S$ and $R$, since four 5D hyperplanes must intersect with the Plucker quadric to generate each such vertex. These vertices must correspond to stabbing lines tight on four edges of $S$ and $R$ in 3D; i.e., lines through a vertex of $S$ and a vertex of $R$ (note that these lines necessarily have real 3D coefficients). In our implementation, there are at most sixteen such lines, since all patches are quadrilaterals.

There are several advantages to performing blocker analysis in linespace. The data structure for a single blocker is constant size, and for a single patch interaction is linear in the number of blockers. The linespace analysis obviates complicated 3D topological and numerical computations. The only operations required by the linespace representation are mapping from 3D lines to 5D points and hyperplanes, and computing inner products between points and hyperplanes.

## 6.2 Incremental Blocker Maintenance

The tube data structure, and incremental visibility maintenance, can now be fully described. Suppose patch $R$ is subdivided against patch $S$ into child elements $C(R) \subset R$. The tube for $S$ and each $C_R \in C(R)$ stores $S$, $C_R$, and a constant number of linespace points $\Pi(S, R)$ whose convex hull $conv(\Pi(S, R))$ includes the set of all lines through $S$ and $C_R$. Finally, each blocker in $B(S, R)$ is reclassified with respect to the child tube to produce $B(S, C_R)$, and the visibility status $V(S, C_R)$ of each interaction $I(S, C_R)$ is determined. As before, many instances of total invisibility, partial visibility, and total visibility are discovered quickly. Other situations are considered too complex to analyze completely, and we "punt" and classify the interaction as partially visible (perhaps causing further subdivision [11]).

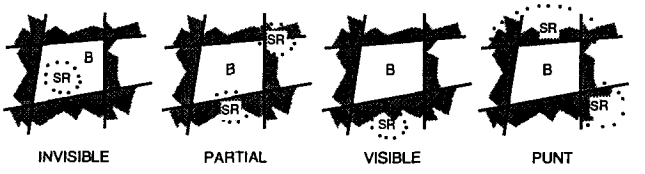



**Figure 10**: Performing 3D triage in 5D linespace.

Consider an interaction $(S, R)$ and a single potential blocker $B$ (Figure 10). We wish to determine, without extensive analysis, whether all, none, or some of the lines through $S$ and $R$ stab the blocker $B$. Respectively, this is equivalent to determining whether $conv(\Pi(S, R))$ lies entirely inside, is disjoint from, or has some intersection with $\cap_k \Lambda_k(B)$, the set of lines through the blocker (Figure 11). We exploit the fact that, in linespace, both sets of lines are convex.

The points in $\Pi(S, R)$ are first classified with respect to the blocker hyperplanes $\Lambda_k(B)$. If all of the points lie inside the $\Lambda_k(B)$, then $conv(\Pi(S, R)) \subset \cap_k \Lambda_k(B)$, by convexity. $B$ is therefore completely occluding and $V(S, R)$ is INVISIBLE. If some of the points lie inside the $\Lambda_k(B)$, and some lie outside, some lines through $S$ and $R$ stab $B$, and $V(S, R)$ is PARTIAL. If *all* of the points lie outside some *single* $\Lambda_k(B)$, $V(S, R)$ is VISIBLE. Finally, the complex case occurs when *all* of the points lie outside *all* of the $\Lambda_k(B)$. This does not guarantee total visibility, since $conv(\Pi(S, R))$ may still have some intersection with $\cap_k \Lambda_k(B)$ (this case is labeled PUNT in Figures 10 and 11); accordingly, $V(S, R)$ is classified as PARTIAL.

The logic for multiple blockers is straightforward; any single blocker can cause $V(S, R)$ to be INVISIBLE, but all blockers must be disjoint in order for $V(S, R)$ to be VISIBLE. Otherwise, any impinging blocker causes $V(S, R)$ to become PARTIAL.

## 6.3 Evolution

Figure 12 depicts an example of blocker list evolution and incremental reclassification of child interactions. White lines connecting quadrilateral centroids represent VISIBLE interactions; green lines represent PARTIAL interactions, and red lines represent the tube

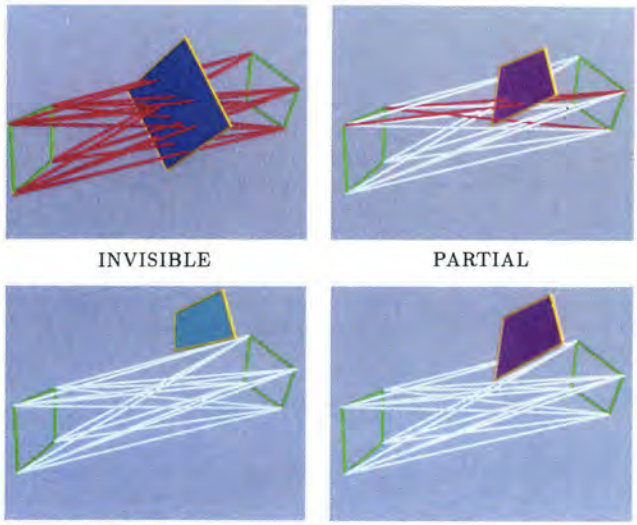



**Figure 11**: The four possible outcomes of a blocker classification.

between two interacting polygons. (INVISIBLE interactions are not shown.) In Figure 12-i, two red polygons interact via a blocker list containing the orange polygon. In Figure 12-ii, the large red polygon is subdivided into quadrants, and its child interactions with the small red polygon are shown. One of these interactions is INVISIBLE. The other three are PARTIAL; the tube for one of them is shown. Finally, in Figure 12-iii, the child polygon is subdivided; three of *its* children become VISIBLE to the polygon at right, but one (shown) remains PARTIAL.

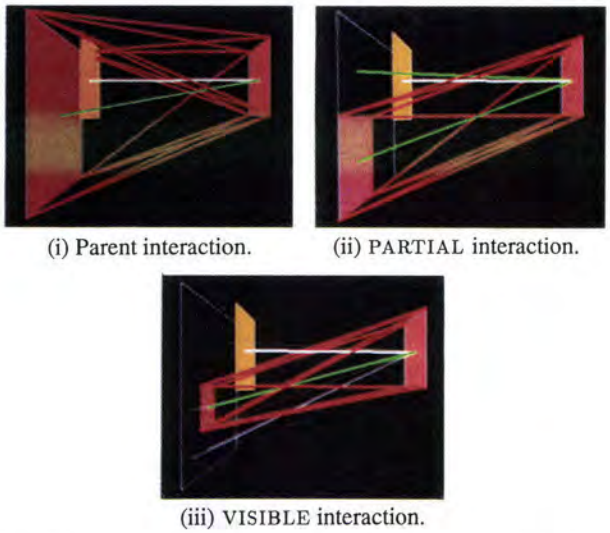



**Figure 12**: Reclassification of child interactions after subdivision.

The linespace algorithms guarantee conservative visibility, in that blockers are only discarded from interactions if they are definitely known to be disjoint. Existing algorithms use point-sampling [2, 6, 11, 16] or point-to-area visibility [3, 4] techniques and therefore do not guarantee correct inter-area visibility determination. In contrast, we establish exact visibility information where possible, and adaptively subdivide until the uncertainty of visibility estimation in the remaining cases is so small as to be unimportant.

The linespace blocker maintenance algorithms are simple and fast, although they sometimes overestimate occlusion by classifying disjoint blockers as impinging, and may not identify INVISIBLE interactions as early as might a more sophisticated algorithm. Establishment of improved algorithms for the determination of inter-polygon visibility in the presence of multiple blockers is an active area of research [17, 21, 25]. An exact algorithm was presented in [21], but is not yet sufficiently robust for application here.

The work and storage expended for the incremental visibility maintenance also serves to accelerate the sampling done to establish inter-patch energy transfer (i.e., to estimate form factors). The ray/blocker machinery is simply applied to random sample rays (as used in [11]). The cost of each ray/blocker test is four 5D inner products.

# 7 Results

## 7.1 Spatial Subdivision

We implemented the spatial subdivision, propagation, interference, and maintenance algorithms described, and instrumented their execution for the data set shown in Figure 1. All execution was on a lightly loaded 50-MIP Silicon Graphics VGX. The input model comprised 403 patches. Constructing the BSP tree required about thirty CPU seconds; the resulting tree contained 220 leaf cells. Each cell had 5.99 boundary faces, and 5.87 patches coaffine with some boundary face, on average (thus the spatial subdivision heuristics produced fairly local partitioning behavior). Another seven CPU seconds were absorbed by cell neighbor-finding and enumeration of the 525 portals between leaf cells of the subdivision.

## 7.2 Visibility Propagation

Computing inter-cell and inter-patch visibility for the model absorbed five CPU minutes. Of the $81003 = (403 \times 402)/2$ patch pairs in the model, 4,391, or 5.4%, were classified as mutually visible. (Thus preprocessing obviated nearly 95% of the potential patch-patch interactions). Of these 4,391 patch pairs, 2,814 (64.1%) were partially visible, and 1,577 (35.9%) were totally visible. Each patch saw, on average, 22 other patches.

The inter-cell traversals performed 10,128 stabbing tests of portal sequences, or about 46 tests per cell. The inter-polygon traversals performed 16,055 further incremental stabbing tests, one for each successfully reached cell and potentially visible patch pair. Thus, about 40 tests per patch were required to establish inter-patch visibility for all patches. The average length of a tested inter-cell portal sequence was just over 5 portals. This is consistent with our experience using a ten-thousand polygon, five-thousand cell axial model, in which the average portal sequence length was less than ten [8, 22]. A histogram of observed portal sequence lengths and stabbing percentages is shown in Figure 13.

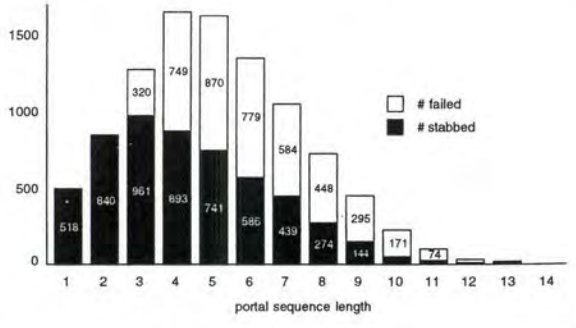



**Figure 13**: Stabbing successes and failures, by sequence length.

The inter-cell visibility determination uses a depth-first-search through the cell adjacency graph, applying an incremental stabbing predicate and visible volume computation at each step. The incremental operation expends linear time in the number of portals currently in the sequence, assuming a constant number of edges per portal, and so requires $O(n^2)$ time to stab a sequence of $n$ portals. In practice, this seems not to prohibit use of the algorithms on real data sets, since most portal sequences are short (less than ten portals), and the algorithmic constants are therefore more important than the asymptotic complexity measure.

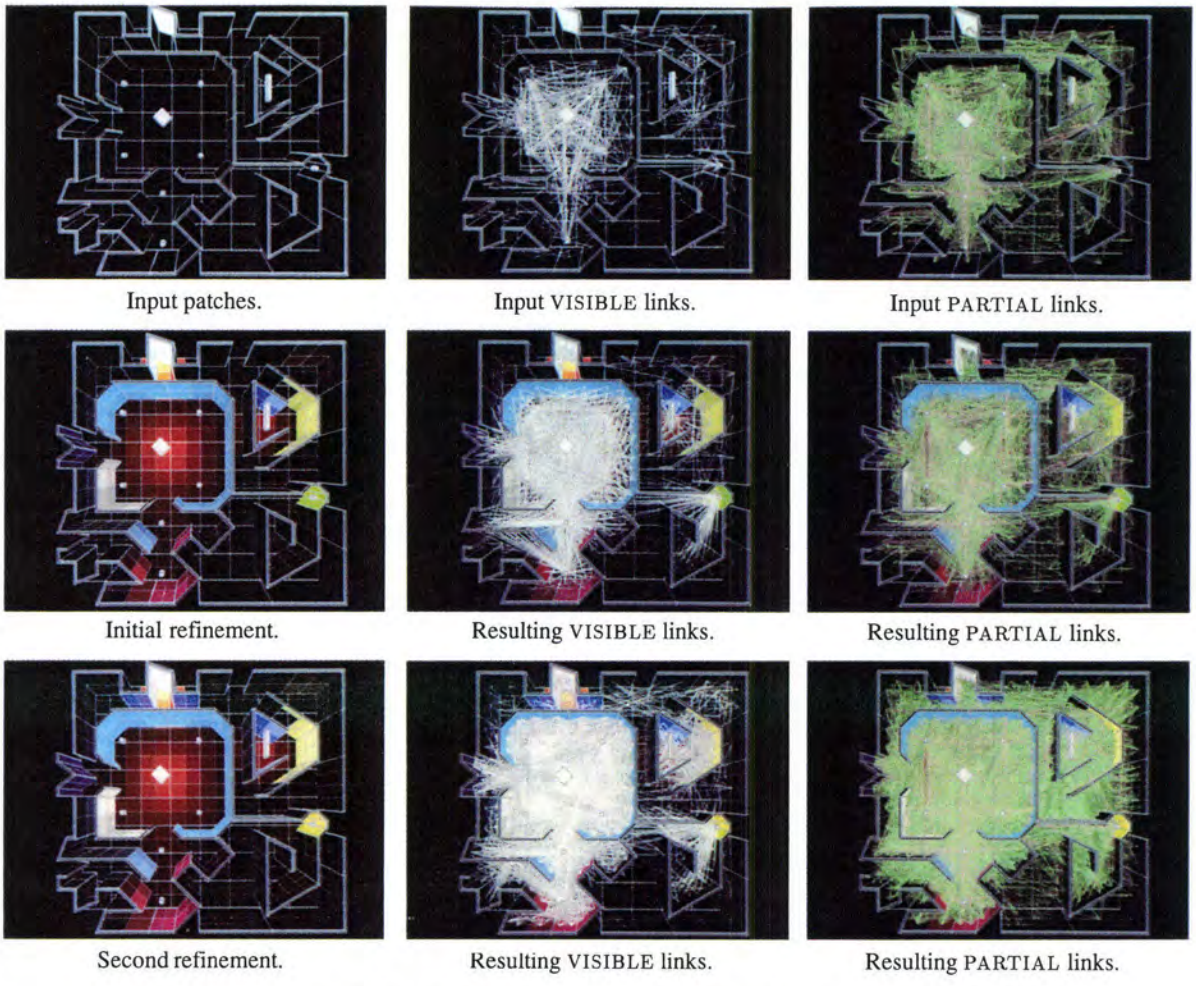



**Figure 14**: Refinement of the input patches (left), VISIBLE links (middle), and PARTIAL links (right).

### 7.3 Blocker Detection

There were 5 blockers between partially visible patches, on average, reached through portal sequences of average length five. The subdivision heuristic was effective; the BSP tree did not suffer from excessive "free-space" splitting, or regions in which subdivision planes were induced due to far away polygons.

The visibility analysis communicated its results to the radiosity computation via an ASCII file. Each file line recorded an interaction $I(S, R)$ between two polygons, the length of the associated blocker list $B(S, R)$, and the blockers themselves. A zero-length blocker list implied total visibility between $S$ and $R$, i.e., $V(S, R)$ =VISIBLE; otherwise the visibility status was PARTIAL.

### 7.4 Blocker Maintenance

The model input to the radiosity computation is shown at the upper left of Figure 14. The input patches form the radiosity program's initial mesh. The 4,391 initial VISIBLE and PARTIAL links are shown, respectively, in white (middle column) and green (right column). Two iterations of patch-patch refinement were performed. The resulting model mesh, PARTIAL and VISIBLE interactions are displayed in the second and third rows of Figure 14. The number of VISIBLE links drastically increases after the first iteration. Their increased density naturally indicates unoccluded regions of the model. Similarly, the green PARTIAL links indicate occlusion. INVISIBLE links are not shown, as they were discarded by the radiosity program upon detection.

Using the results of the visibility preprocessing, the initial refine took only 11 seconds, performing 145,846 interactions. The second refinement stage required 50 seconds, and performed 186,703 interactions. About 90% of the refined interactions were VISIBLE, thus requiring no sampling for form-factor estimation. Table 1 charts the evolution of each link type, the number of elements, and the number of interactions at each refine.

| Input Links | | Refine | | Links | | Refine | | Links |
|---|---|---|---|---|---|---|---|---|
| V 1,577 | (35.9%) | 170,440 | (87.6%) | V 176,474 | | 218,420 | (87.6%) | V 225,763 |
| | | V 6,034 | ( 3.1%) | | | V 7,343 | ( 2.9%) | |
| P 2,814 | (64.1%) | P 16,889 | ( 8.7%) | P 16,889 | | P 22,157 | ( 8.9%) | P 22,157 |
| | | I 1,096 | ( 0.6%) | | | I 1,339 | ( 0.5%) | |
| 4,391 | | 194,459 | | 193,363 | | 249,259 | | 247,920 |
| 403 patches | | | | 15,039 patches | | | | 17,347 patches |
| 4,391 interactions | | | | 145,846 interactions | | | | 186,703 interactions |

**Table 1**: Link evolution by type, with patch and interaction counts.

Since the time complexity of the radiosity algorithm is proportional to the number of interactions, the visibility preprocessing significantly decreased the computation done by the radiosity algorithm. Moreover, the modified radiosity algorithm was more accurate, since no partially visible interactions were missed due to sampling errors (as in [11]).

### 7.5 Blocker Visualization

All of the algorithms described in this paper were implemented using visualization tools that allowed interactive inspection of complex data structures. Figures 15 through 17 depict the use of this tool to investigate some interesting PARTIAL interactions. Again, the white and green line segments represent VISIBLE and PARTIAL interactions, respectively; for a particular partial interaction in each figure, the tube is shown (in red), and the blockers for the interaction

**Figure 15**: The tube data structure (red, orange) for an ordinary PARTIAL interaction.

**Figure 16**: Spatially incident polygons that have not been classified as blockers.

**Figure 17**: A PARTIAL interaction that could be classified as INVISIBLE.

are highlighted in orange. Figure 15 depicts an ordinary PARTIAL interaction. Figure 16 depicts spatially incident polygons that (correctly) have not been classified as blockers. Figure 17 depicts a PARTIAL interaction for which no single blocker occludes the source and receiver; a more sophisticated algorithm could classify this interaction as INVISIBLE.

## 8 Summary and Conclusion

We have presented several novel algorithms that represent an effective application of global visibility analysis to radiosity computations, an important problem in image synthesis. Given the complexity of both the visibility and radiosity approaches used, it was surprisingly easy to couple the two processes. We did so using an abstraction in which interactions between polygons were maintained along with all potentially blocking polygons. We argue that, for an interesting class of large models, inter-polygon visibility has roughly constant complexity throughout the interior of the model. After construction of a spatial subdivision for the model, the visibility algorithms we present are output sensitive; they expend work proportional to the amount of inter-polygon visibility present.

None of the visibility algorithms attempt to compute exact visibility information. However, they achieve precision in a different sense, by reporting all visibilities conservatively; potentially visible interactions are always reported.

Only blockers can occlude a specified source from a specified emitter. Thus, the blocker list formulation is applicable to the problem of discontinuity meshing in the presence of area light sources [12, 13, 14, 21], as well as to the construction of an "oracle" to decide which, if any, among a collection of discontinuities should be meshed upon earliest.

We showed that the visibility analysis significantly accelerated a radiosity computation in a polygonal environment. Finally, we demonstrated the successful application of some elegant concepts such as linespace and algorithmic triage to the concrete problem of construction and incremental maintenance of blocker lists.

## Acknowledgments

The authors are grateful to David Laur for his assistance with the geometric model and the color plates, and to Dani Lischinski for his valuable comments. This work was begun during a visit to the NSF Science and Technology Center for the Visualization of Geometric Structures, in Minneapolis, and partially supported by Apple, Silicon Graphics Computer Systems, and the National Science Foundation (CCR 9207966).

## References

[1] AIREY, J. M. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*. PhD thesis, UNC Chapel Hill, 1990.

[2] BAUM, D. R., MANN, S., SMITH, K. P., AND WINGET, J. M. Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 4 (1991), 51–60.

[3] CAMPBELL III, A., AND FUSSELL, D. S. An Analytic Approach to Illumination with Area Light Sources. Tech. Rep. TR-91-25, Department of Computer Sciences, UT Austin, 1991.

[4] CHIN, N., AND FEINER, S. Fast Object-Precision Shadow Generation for Area Light Sources Using BSP Trees. In *Proc. 1992 Symposium on Interactive 3D Graphics* (1992), pp. 21–30.

[5] COHEN, M. F., CHEN, S. E., WALLACE, J. R., AND GREENBERG, D. P. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics (Proc. SIGGRAPH '88) 22*, 4 (1988), 75–84.

[6] COHEN, M. F., AND GREENBERG, D. P. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics (Proc. SIGGRAPH '85) 19*, 3 (1985), 31–40.

[7] FUCHS, H., KEDEM, Z., AND NAYLOR, B. On visible surface generation by a priori tree structures. *Computer Graphics (Proc. SIGGRAPH '80) 14*, 3 (1980), 124–133.

[8] FUNKHOUSER, T. A., SÉQUIN, C. H., AND TELLER, S. Management of Large Amounts of Data in Interactive Building Walkthroughs. In *Proc. 1992 Workshop on Interactive 3D Graphics* (1992), pp. 11–20.

[9] GIGUS, Z., CANNY, J., AND SEIDEL, R. Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 6 (1991), 542–551.

[10] HAINES, E. A., AND WALLACE, J. R. Shaft Culling for Efficient Ray-Traced Radiosity. In *Proc. 2nd Eurographics Workshop on Rendering* (May 1991).

[11] HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 4 (1991), 197–206.

[12] HECKBERT, P. S. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, Computer Sciences Department, UC Berkeley, June 1991.

[13] LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications 12*, 6 (1992), 25–39.

[14] LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. Combining Hierarchical Radiosity and Discontinuity Meshing. *Computer Graphics (Proc. SIGGRAPH '93) 27* (1993).

[15] MEGIDDO, N. Linear programming in linear time when the dimension is fixed. *Journal of the ACM 31* (1984), 114–127.

[16] NISHITA, T., AND NAKAMAE, E. Half-Tone Representation of 3-D Objects Illuminated by Area Sources or Polyhedron Sources. In *Proc. IEEE COMPSAC, 1983* (1983), pp. 237–242.

[17] PLANTINGA, H. An algorithm for finding the weakly visible faces from a polygon in 3D. Tech. Rep. 92–11, U of Pittsburgh, 1992.

[18] PLANTINGA, W., AND DYER, C. An algorithm for constructing the aspect graph. In *Proc. 27th Annual IEEE Symposium on Foundations of Computer Science* (1986), pp. 123–131.

[19] SEIDEL, R. Small-dimensional linear programming and convex hulls made easy. *Discrete and Computational Geometry* (1991), 423–434.

[20] SOMMERVILLE, D. *Analytical Geometry of Three Dimensions*. Cambridge University Press, 1959.

[21] TELLER, S. Computing the Antipenumbra Cast by an Area Light Source. *Computer Graphics (Proc. SIGGRAPH '92) 26*, 2 (1992), 139–148.

[22] TELLER, S. *Visibility Computations in Densely Occluded Polyhedral Environments*. PhD thesis, CS Dept., UC Berkeley, 1992.

[23] TELLER, S., AND HOHMEYER, M. E. Computing the Lines Piercing Four Lines. Tech. Rep. UCB/CSD 91/665, Computer Science Department, UC Berkeley, 1991.

[24] TELLER, S., AND SÉQUIN, C. H. Visibility Preprocessing for Interactive Walkthroughs. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 4 (1991), 61–69.

[25] ZHAO, J., AND DOBKIN, D. Personal communication, 1992.



# Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments

Thomas A. Funkhouser and Carlo H. Séquin
University of California at Berkeley[‡]

## Abstract

We describe an adaptive display algorithm for interactive frame rates during visualization of very complex virtual environments. The algorithm relies upon a hierarchical model representation in which objects are described at multiple levels of detail and can be drawn with various rendering algorithms. The idea behind the algorithm is to adjust image quality adaptively to maintain a uniform, user-specified target frame rate. We perform a constrained optimization to choose a level of detail and rendering algorithm for each potentially visible object in order to generate the "best" image possible within the target frame time. Tests show that the algorithm generates more uniform frame rates than other previously described detail elision algorithms with little noticeable difference in image quality during visualization of complex models.

**CR Categories and Subject Descriptors:**
**[Computer Graphics]:** I.3.3 Picture/Image Generation – *viewing algorithms*; I.3.5 Computational Geometry and Object Modeling – *geometric algorithms, object hierarchies*; I.3.7 Three-Dimensional Graphics and Realism – *virtual reality*.

## 1 Introduction

Interactive computer graphics systems for visualization of realistic-looking, three-dimensional models are useful for evaluation, design and training in virtual environments, such as those found in architectural and mechanical CAD, flight simulation, and virtual reality. These visualization systems display images of a three-dimensional model on the screen of a computer workstation as seen from a simulated observer's viewpoint under interactive control by a user. If images are rendered smoothly and quickly enough, an illusion of real-time exploration of a virtual environment can be achieved as the simulated observer moves through the model.

[‡]Computer Science Division, Berkeley, CA 94720



It is important for a visualization system to maintain an interactive frame rate (e.g., a constant ten frames per second). If frame rates are too slow, or too jerky, the interactive feel of the system is greatly diminished [3]. However, realistic-looking models may contain millions of polygons – far more than currently available workstations can render at interactive frame rates. Furthermore, the complexity of the portion of the model visible to the observer can be highly variable. Tens of thousands of polygons might be simultaneously visible from some observer viewpoints, whereas just a few can be seen from others. Programs that simply render all potentially visible polygons with some predetermined quality may generate frames at highly variable rates, with no guaranteed upper bound on any single frame time.

Using the UC Berkeley Building Walkthrough System [5] and a model of Soda Hall, the future Computer Science Building at UC Berkeley, as a test case, we have developed an adaptive algorithm for interactive visualization that guarantees a user-specified target frame rate. The idea behind the algorithm is to trade image quality for interactivity in situations where the environment is too complex to be rendered in full detail at the target frame rate. We perform a constrained optimization that selects a level of detail and a rendering algorithm with which to render each potentially visible object to produce the "best" image possible within a user-specified target frame time. In contrast to previous culling techniques, this algorithm guarantees a uniform, bounded frame rate, even during visualization of very large, complex models.

## 2 Previous Work

### 2.1 Visibility Determination

In previous work, visibility algorithms have been described that compute the portion of a model potentially visible from a given observer viewpoint [1, 11]. These algorithms cull away large portions of a model that are occluded from the observer's viewpoint, and thereby improve frame rates significantly. However, in very detailed models, often more polygons are visible from certain observer viewpoints than can be rendered in an interactive frame time. Certainly, there is no upper bound on the complexity of the scene visible from an observer's viewpoint. For instance, consider walking through a very detailed model of a fully stocked department store, or viewing an assembly of a complete airplane engine. In our model of Soda Hall, there are some viewpoints from which an observer can see more than eighty thousand polygons. Clearly, visibility processing alone is not sufficient to guarantee an interactive frame rate.

 ACM-0-89791-601-8/93/008/0247 $01.50

## 2.2 Detail Elision

To reduce the number of polygons rendered in each frame, an interactive visualization system can use *detail elision*. If a model can be described by a hierarchical structure of *objects*, each of which is represented at multiple *levels of detail* (LODs), as shown in Figure 1, simpler representations of an object can be used to improve frame rates and memory utilization during interactive visualization. This technique was first described by Clark [4], and has been used by numerous commercial visualization systems [9]. If different representations for the same object have similar appearances and are blended smoothly, using transparency blending or three-dimensional interpolation, transitions between levels of detail are barely noticeable during visualization.

241 Polygons 44 Polygons

Figure 1: Two levels of detail for a chair.

Previously described techniques for choosing a level of detail at which to render each visible object use static heuristics, most often based on a threshold regarding the size or distance of an object to the observer [2, 8, 9, 13], or the number of pixels covered by an average polygon [5]. These simple heuristics can be very effective at improving frame rates in cases where most visible objects are far away from the observer and map to very few pixels on the workstation screen. In these cases, simpler representations of some objects can be displayed, reducing the number of polygons rendered without noticeably reducing image quality.

Although static heuristics for visibility determination and LOD selection improve frame rates in many cases, they do not generally produce a *uniform* frame rate. Since LODs are computed independently for each object, the number of polygons rendered during each frame time depends on the size and complexity of the objects visible to the observer. The frame rate may vary dramatically from frame to frame as many complex objects become visible or invisible, and larger or smaller.

Furthermore, static heuristics for visibility determination and LOD selection do not even guarantee a *bounded* frame rate. The frame rate can become arbitrarily slow, as the scene visible to the observer can be arbitrarily complex. In many cases, the frame rate may become so slow that the system is no longer interactive. Instead, a LOD selection algorithm should adapt to overall scene complexity in order to produce uniform, bounded frame rates.

## 2.3 Adaptive Detail Elision

In an effort to maintain a specified *target frame rate*, some commercial flight simulators use an adaptive algorithm that adjusts the size threshold for LOD selection based on feedback regarding the time required to render previous frames [9]. If the previous frame took longer than the target frame time, the size threshold for LOD selection is increased so that future frames can be rendered more quickly.

This adaptive technique works reasonably well for flight simulators, in which there is a large amount of coherence in scene complexity from frame to frame. However, during visualization of more discontinuous virtual environments, scene complexity can vary radically between successive frames. For instance, in a building walkthrough, the observer may turn around a corner into a large atrium, or step from an open corridor into a small, enclosed office. In these situations, the number and complexity of the objects visible to the observer changes suddenly. Thus the size threshold chosen based on the time required to render previous frames is inappropriate, and can result in very poor performance until the system reacts. Overshoot and oscillation can occur as the feedback control system attempts to adjust the size threshold more quickly to achieve the target frame rate.

In order to *guarantee* a bounded frame rate during visualization of discontinuous virtual environments, an adaptive algorithm for LOD selection should be *predictive*, based on the complexity of the scene to be rendered in the current frame, rather than *reactive*, based only on the time required to render previous frames. A predictive algorithm might estimate the time required to render every object at every level of detail, and then compute the largest size threshold that allows the current frame to be rendered within the target frame time. Unfortunately, implementing a predictive algorithm is non-trivial, since no closed-form solution exists for the appropriate size threshold.

# 3 Overview of Approach

Our approach is a generalization of the predictive approach. Conceptually, every potentially visible object can be rendered at any level of detail, and with any rendering algorithm (e.g., flat-shaded, Gouraud-shaded, texture mapped, etc.). Every combination of objects rendered with certain levels of detail and rendering algorithms takes a certain amount of time, and produces a certain image. We aim to find the combination of levels of detail and rendering algorithms for all potentially visible objects that produces the "best" image possible within the target frame time.

More formally, we define an *object tuple*, $(O, L, R)$, to be an instance of object $O$, rendered at level of detail $L$, with rendering algorithm $R$. We define two heuristics for object tuples: $Cost(O, L, R)$ and $Benefit(O, L, R)$. The *Cost* heuristic estimates the time required to render an object tuple; and the *Benefit* heuristic estimates the "contribution to model perception" of a rendered object tuple. We define $S$ to be the set of object tuples rendered in each frame. Using these formalisms, our approach for choosing a level of detail and rendering algorithm for each potentially visible object can be stated:

$$\begin{aligned} &\text{Maximize :} \\ &\qquad \textstyle\sum_S Benefit(O, L, R) \\ &\text{Subject to :} \\ &\qquad \textstyle\sum_S Cost(O, L, R) \leq TargetFrameTime \end{aligned} \tag{1}$$

This formulation captures the essence of image generation with real-time constraints: "do as well as possible in a given amount of time." As such, it can be applied to a wide variety of problems that require images to be displayed in a fixed amount of time, including adaptive ray tracing (i.e., given a fixed number of rays, cast those that contribute most to the image), and adaptive radiosity (i.e., given

a fixed number of form-factor computations, compute those that contribute most to the solution). If levels of detail representing "no polygons at all" are allowed, this approach handles cases where the target frame time is not long enough to render all potentially visible objects even at the lowest level of detail. In such cases, only the most "valuable" objects are rendered so that the frame time constraint is not violated. Using this approach, it is possible to generate images in a short, fixed amount of time, rather than waiting much longer for images of the highest quality attainable.

For this approach to be successful, we need to find *Cost* and *Benefit* heuristics that can be computed quickly and accurately. Unfortunately, *Cost* and *Benefit* heuristics for a specific object tuple cannot be predicted with perfect accuracy, and may depend on other object tuples rendered in the same image. A perfect *Cost* heuristic may depend on the model and features of the graphics workstation, the state of the graphics system, the state of the operating system, and the state of other programs running on the machine. A perfect *Benefit* heuristic would consider occlusion and color of other object tuples, human perception, and human understanding. We cannot hope to quantify all of these complex factors in heuristics that can be computed efficiently. However, using several simplifying assumptions, we have developed approximate *Cost* and *Benefit* heuristics that are both efficient to compute and accurate enough to be useful.

# 4 Cost Heuristic

The $Cost(O, L, R)$ heuristic is an estimate of the time required to render object $O$ with level of detail $L$ and rendering algorithm $R$. Of course, the actual rendering time for a set of polygons depends on a number of complex factors, including the type and features of the graphics workstation. However, using a model of a generalized rendering system and several simplifying assumptions, it is possible to develop an efficient, approximate *Cost* heuristic that can be applied to a wide variety of workstations. Our model, which is derived from the *Graphics Library Programming Tools and Techniques* document from Silicon Graphics, Inc. [10], represents the rendering system as a pipeline with the two functional stages shown in Figure 2:

- *Per Primitive:* coordinate transformations, lighting calculations, clipping, etc.
- *Per Pixel:* rasterization, z-buffering, alpha blending, texture mapping, etc.

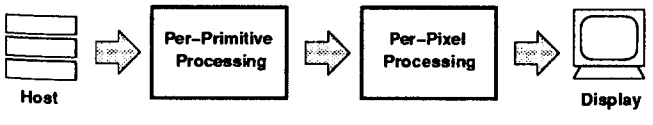



Figure 2: Two-stage model of the rendering pipeline.

Since separate stages of the pipeline run in parallel, and must wait only if a subsequent stage is "backed up," the throughput of the pipeline is determined by the speed of the slowest stage – i.e., the bottleneck. If we assume that the host is able to send primitives to the graphics subsystem faster than they can be rendered, and no other operations are executing that affect the speed of any stage of the graphics subsystem, we can model the time required to render an object tuple as the maximum of the times taken by any of the stages.

We model the time taken by the *Per Primitive* stage as a linear combination of the number of polygons and vertices in an object tuple, with coefficients that depend on the rendering algorithm and machine used. Likewise, we assume that the time taken by the *Per Pixel* stage is proportional to the number of pixels an object covers. Our model for the time required to render an object tuple is:

$$Cost(O, L, R) = max \left\{ \begin{array}{c} C_1 Poly(O, L) + C_2 Vert(O, L) \\ C_3 Pix(O) \end{array} \right\}$$

where $O$ is the object, $L$ is the level of detail, $R$ is the rendering algorithm, and $C_1$, $C_2$ and $C_3$ are constant coefficients specific to a rendering algorithm and machine.

For a particular rendering algorithm and machine, useful values for these coefficients can be determined experimentally by rendering sample objects with a wide variety of sizes and LODs, and graphing measured rendering times versus the number of polygons, vertices and pixels drawn. Figure 3a shows measured times for rendering four different LODs of the chair shown in Figure 1 rendered with flat-shading. The slope of the best fitting line through the data points represents the time required *per polygon* during this test. Using this technique, we have derived cost model coefficients for our Silicon Graphics VGX 320 that are accurate within 10% at the 95% confidence level. A comparison of actual and predicted rendering times for a sample set of frames during an interactive building walkthrough is shown in Figure 3b.

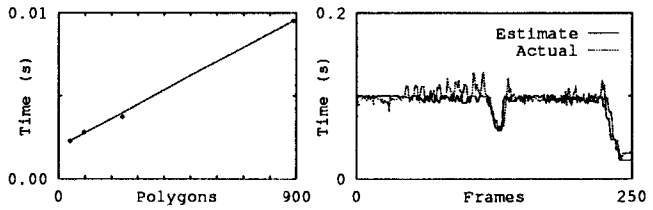



Figure 3: Cost model coefficients can be determined empirically. The plot in (a) shows actual flat-shaded rendering times for four LODs of a chair, and (b) shows a comparison of actual and estimated rendering times of frames during an interactive building walkthrough.

# 5 Benefit Heuristic

The $Benefit(O, L, R)$ heuristic is an estimate of the "contribution to model perception" of rendering object $O$ with level of detail $L$ and rendering algorithm $R$. Ideally, it predicts the amount and accuracy of information conveyed to a user due to rendering an object tuple. Of course, it is extremely difficult to accurately model human perception and understanding, so we have developed a simple, easy-to-compute heuristic based on intuitive principles.

Our *Benefit* heuristic depends primarily on the size of an object tuple in the final image. Intuitively, objects that appear larger to the observer "contribute" more to the image (see Figure 4). Therefore, the base value for our *Benefit* heuristic is simply an estimate of the number of pixels covered by the object.

Our *Benefit* heuristic also depends on the "accuracy" of an object tuple rendering. Intuitively, using a more detailed representation or a more realistic rendering algorithm for an object generates a higher quality image, and therefore conveys more accurate information to the user. Conceptually, we evaluate the "accuracy" of an object tuple rendering by comparison to an *ideal image* generated with an

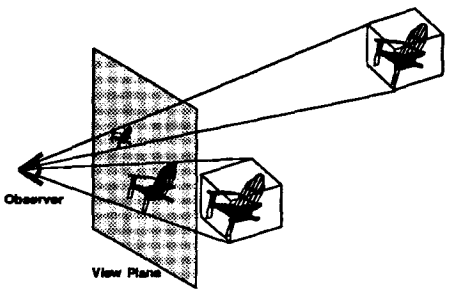
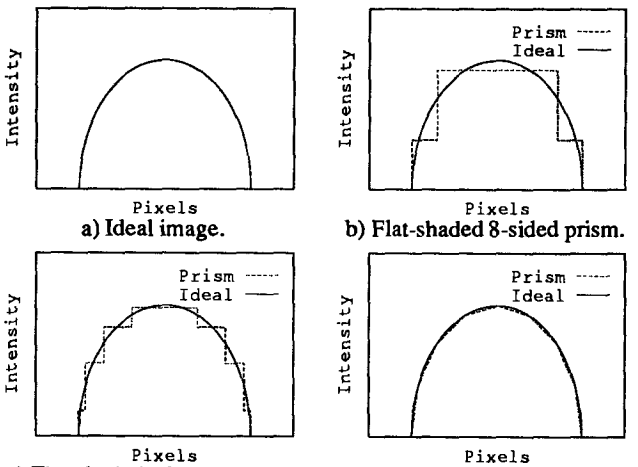



Figure 4: Objects that appear larger "contribute" more to the image.

ideal camera. For instance, consider generating a gray-level image of a scene containing only a cylinder with a diffusely reflecting Lambert surface illuminated by a single directional light source in orthonormal projection. Figure 5a shows an intensity plot of a sample scan-line of an ideal image generated for the cylinder.

First, consider approximating this ideal image with an image generated using a flat-shaded, polygonal representation for the cylinder. Since a single color is assigned to all pixels covered by the same polygon, a plot of pixel intensities across a scan-line of such an image is a stair-function. If an 8-sided prism is used to represent the cylinder, at most 4 distinct colors can appear in the image (one for each front-facing polygon), so the resulting image does not approximate the ideal image very well at all, as shown in Figure 5b. By comparison, if a 16-sided prism is used to represent the cylinder, as many as 8 distinct colors can appear in the image, generating a closer approximation to the ideal image, as shown in Figure 5c.




a) Ideal image. b) Flat-shaded 8-sided prism.

c) Flat-shaded 16-sided prism. d) Gouraud-shaded 16-sided prism.

Figure 5: Plots of pixel intensity across a sample scan-line of images generated using different representations and rendering algorithms for a simple cylinder.

Next, consider using Gouraud shading for a polygonal representation. In Gouraud shading, intensities are interpolated between vertices of polygons, so a plot of pixel intensities is a continuous, piecewise-linear function. Figure 5d shows a plot of pixel intensities across a scan line for a Gouraud shaded 16-sided prism. Compared to the plot for the flat-shaded image (Figure 5b), the Gouraud shaded image approximates the ideal image much more closely.

More complex representations (e.g., parametric or implicit surfaces) and rendering techniques (e.g., Phong shading, antialiasing or ray tracing) could be used to approximate the ideal image even more closely. Based on this intuition, we assume that the "error," i.e., the difference from the ideal image, decreases with the number of samples (e.g., rays/vertices/polygons) used to render an object tuple, and is dependent on the type of interpolation method used (e.g., Gouraud/flat). We capture these effects in the *Benefit* heuristic by multiplying by an "accuracy" factor:

$$Accuracy(O, L, R) = 1 - Error = 1 - \frac{BaseError}{Samples(L, R)^m}$$

where Samples(L, R) is #pixels for ray tracing, or #vertices for Gouraud shading, or #polygons for flat-shading (but never more than #pixels); and $m$ is an exponent dependent on the interpolation method used (flat = 1, Gouraud = 2). The *BaseError* is arbitrarily set to 0.5 to give a strong error for a curved surface represented by a single flat polygon, but still account for a significantly higher benefit than not rendering the surface at all.

In addition to the size and accuracy of an object tuple rendering, our *Benefit* heuristic depends on on several other, more qualitative, factors, some of which apply to a static image, while others apply to sequences of images:

- **Semantics:** Some types of object may have inherent "importance." For instance, walls might be more important than pencils to the user of a building walkthrough; and enemy robots might be most important to the user of a video game. We adjust the *Benefit* of each object tuple by an amount proportional to the inherent importance of its object type.
- **Focus:** Objects that appear in the portion of the screen at which the user is looking might contribute more to the image than ones in the periphery of the user's view. Since we currently do not track the user's eye position, we simply assume that objects appearing near the middle of the screen are more important than ones near the side. We reduce the *Benefit* of each object tuple by an amount proportional to its distance from the middle of the screen.
- **Motion Blur:** Since objects that are moving quickly across the screen appear blurred or can be seen for only a short amount of time, the user may not be able to see them clearly. So we reduce the *Benefit* of each object tuple by an amount proportional to the ratio of the object's apparent speed to the size of an average polygon.
- **Hysteresis:** Rendering an object with different levels of detail in successive frames may be bothersome to the user and may reduce the quality of an image sequence. Therefore, we reduce the *Benefit* of each object tuple by an amount proportional to the difference in level of detail or rendering algorithm from the ones used for the same object in the previous frame.

Each of these qualitative factors is represented by a multiplier between 0.0 and 1.0 reflecting a possible reduction in object tuple benefit. The overall *Benefit* heuristic is a product of all the aforementioned factors:

$$Benefit(O, L, R) = Size(O) * Accuracy(O, L, R) * Importance(O) * Focus(O) * Motion(O) * Hysteresis(O, L, R)$$

This *Benefit* heuristic is a simple experimental estimate of an object tuple's "contribution to model perception." Greater *Benefit* is assigned to object tuples that are larger (i.e., cover more pixels in the image), more realistic-looking (i.e., rendered with higher levels of detail, or better rendering algorithms), more important (i.e.,

semantically, or closer to the middle of the screen), and more apt to blend with other images in a sequence (i.e., hysteresis). In our implementation, the user can manipulate the relative weighting of these factors interactively using sliders on a control panel, and observe their effects in a real-time walkthrough. Therefore, although our current *Benefit* heuristic is rather ad hoc, it is useful for experimentation until we are able to encode more accurate models for human visual perception and understanding.

# 6 Optimization Algorithm

We use the *Cost* and *Benefit* heuristics described in the previous sections to choose a set of object tuples to render each frame by solving equation 1 in Section 3.

Unfortunately, this constrained optimization problem is NP-complete. It is the Continuous Multiple Choice Knapsack Problem [6, 7], a version of the well-known Knapsack Problem in which elements are partitioned into candidate sets, and at most one element from each candidate set may be placed in the knapsack at once. In this case, the set $S$ of object tuples rendered is the knapsack, the object tuples are the elements to be placed into the knapsack, the target frame time is the size of the knapsack, the sets of object tuples representing the same object are the candidate sets, and the *Cost* and *Benefit* functions specify the "size" and "profit" of each element, respectively. The problem is to select the object tuples that have maximum cumulative benefit, but whose cumulative cost fits in the target frame time, subject to the constraint that only one object tuple representing each object may be selected.

We have implemented a simple, greedy approximation algorithm for this problem that selects object tuples with the highest *Value* ($Benefit(O, L, R)/Cost(O, L, R)$). Logically, we add object tuples to $S$ in descending order of *Value* until the maximum cost is competely claimed. However, if an object tuple is added to $S$ which represents the same object as another object tuple already in $S$, only the object tuple with the maximum benefit of the two is retained. The merit of this approach can be explained intuitively by noting that each subsequent portion of the frame time is used to render the object tuple with the best available "bang for the buck." It is easy to show that a simple implementation of this greedy approach runs in $O(n \log n)$ time for $n$ potentially visible objects, and produces a solution that is at least half as good as the optimal solution [6].

Rather than computing and sorting the *Benefit*, *Cost*, and *Value* for all possible object tuples during every frame, as would be required by a naive implementation, we have implemented an incremental optimization algorithm that takes advantage of the fact that there is typically a large amount of coherence between successive frames. The algorithm works as follows: At the start of the algorithm, an object tuple is added to $S$ for each potentially visible object. Initially, each object is assigned the LOD and rendering algorithm chosen in the previous frame, or the lowest LOD and rendering algorithm if the object is newly visible. In each iteration of the optimization, the algorithm first increments the accuracy attribute (LOD or rendering algorithm) of the object that has the highest subsequent *Value*. It then decrements the accuracy attributes of the object tuples with the lowest current *Value* until the cumulative cost of all object tuples in $S$ is less than the target frame time. The algorithm terminates when the same accuracy attribute of the same object tuple is both incremented and decremented in the same iteration.

This incremental implementation finds an approximate solution that is the same as found by the naive implementation if *Values* of object tuples decrease monotonically as tuples are rendered with greater accuracy (i.e., there are diminishing returns with more complex renderings). In any case, the worst-case running time for the algorithm is $O(n \log n)$. However, since the initial guess for the LOD and rendering algorithm for each object is generated from the previous frame, and there is often a large amount of coherence from frame to frame, the algorithm completes in just a few iterations on average. Moreover, computations are done in parallel with the display of the previous frame on a separate processor in a pipelined architecture; they do not increase the effective frame rate as long as the time required for computation is not greater than the time required for display.

# 7 Test Methods

To test whether this new cost/benefit optimization algorithm produces more uniform frame rates than previous LOD selection algorithms, we ran a set of tests with our building walkthrough application using four different LOD selection algorithms:

a) **No Detail Ellsion:** Each object is rendered at the highest LOD.

b) **Static:** Each object is rendered at the highest LOD for which an average polygon covers at least 1024 pixels on the screen.

c) **Feedback:** Similar to *Static* test, except the size threshold for LOD selection is updated in each frame by a feedback loop, based on the difference between the time required to render the previous frame and the target frame time of one-tenth of a second.

d) **Optimization:** Each object is rendered at the LOD chosen by the cost/benefit optimization algorithm described in Sections 3 and 6 in order to meet the target frame time of one-tenth of a second. For comparison sake, the *Benefit* heuristic is limited to consideration of *object size* in this test, i.e., all other *Benefit* factors are set to 1.0.

All tests were performed on a Silicon Graphics VGX 320 workstation with two 33MHz MIPS R3000 processors and 64MB of memory. We used an *eye-to-object* visibility algorithm described in [12] to determine a set of potentially visible objects to be rendered in each frame. The application was configured as a two-stage pipeline with one processor for visibility and LOD selection computations and another separate processor for rendering. Timing statistics were gathered using a $16\mu s$ timer.

In each test, we used the sample observer path shown in Figure 6 through a model of an auditorium on the third floor of Soda Hall. The model was chosen because it is complex enough to differentiate the characteristics of various LOD selection algorithms (87,565 polygons), yet small enough to reside entirely in main memory so as to eliminate the effects of memory management in our tests. The test path was chosen because it represents typical behavior of real users of a building walkthrough system, and highlights the differences between various LOD selection algorithms. For instance, at the observer viewpoint marked 'A', many complex objects are simultaneously visible, some of which are close and appear large to the observer; at the viewpoint marked 'B', there are very few objects visible to the observer, most of which appear small; and at the viewpoint marked 'C', numerous complex objects become visible suddenly as the observer spins around quickly. We refer to these marked observer viewpoints in the analysis, as they are the viewpoints at which the differences between the various LOD selection algorithms are most pronounced.

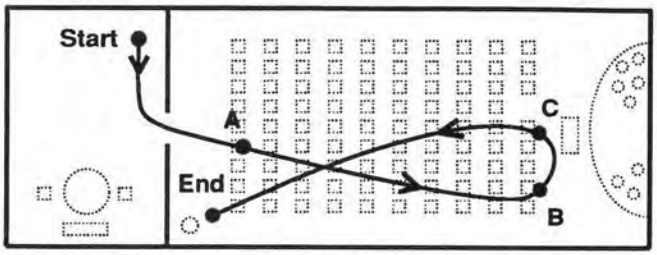


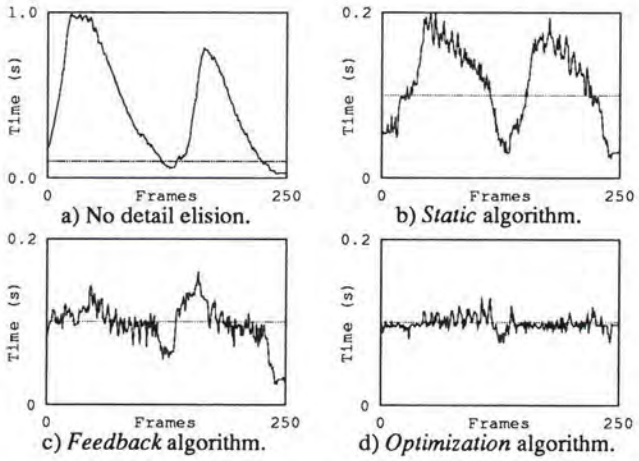

Figure 6: Test observer path through a model of an auditorium.

| LOD Selection Algorithm | Compute Time | | Frame Time | | |
|---|---|---|---|---|---|
| | Mean | Max | Mean | Max | StdDev |
| None | 0.00 | 0.00 | 0.43 | 0.99 | 0.305 |
| Static | 0.00 | 0.01 | 0.11 | 0.20 | 0.048 |
| Feedback | 0.00 | 0.01 | 0.10 | 0.16 | 0.026 |
| Optimization | 0.01 | 0.03 | 0.10 | 0.13 | 0.008 |

Table 1: Cumulative statistics for test observer path (in seconds).

## 8 Results and Discussion

Figure 7 shows plots of the frame time (seconds per frame) for each observer viewpoint along the test path for the four LOD selection algorithms tested. Table 1 shows cumulative compute time (i.e., time required for execution of the LOD selection algorithm) and frame time statistics for all observer viewpoints along the test path.




a) No detail elision. b) *Static* algorithm.

c) *Feedback* algorithm. d) *Optimization* algorithm.

Figure 7: Plots of frame time for every observer viewpoint along test observer path using a) no detail elision, b) static algorithm, c) feedback algorithm, and d) optimization algorithm. Note: the "Frame Time" axis in plot (a) is five-times larger than the others.

If no detail elision is used, and all potentially visible objects are rendered at the highest LOD, the time required for each frame is generally long and non-uniform, since it depends directly on the number and complexity of the objects visible to the observer (see Figure 7a). In our test model, far too many polygons are visible from most observer viewpoints to generate frames at interactive rates without detail elision. For instance, at the observer viewpoint marked 'A' in Figure 6, 72K polygons are simultaneously visible, and the frame time is 0.98 seconds. Overall, the mean frame time for all observer viewpoints on the test path is 0.43 seconds per frame.

If the *Static* LOD selection algorithm is used, objects whose average polygon is smaller than a size threshold fixed at 1024 pixels per polygon are rendered with lower LODs. Even though the frame rate is much faster than without detail elision, there is still a large amount of variability in the frame time, since it depends on the size and complexity of the objects visible from the observer's viewpoint (see Figure 7b). For instance, at the observer viewpoint marked 'A', the frame time is quite long (0.19 seconds) because many visible objects are complex and appear large to the observer. A high LOD is chosen for each of these objects independently, resulting in a long overall frame time. This result can seen clearly in Figure 8a which depicts the LOD selected for each object in the frame for observer viewpoint 'A' – higher LODs are represented by darker shades of gray. On the other hand, the frame time is very short in the frame at the observer viewpoint marked 'B' (0.03 seconds). Since all visible objects appear relatively small to the observer, they are rendered at a lower LOD even though more detail could have been rendered within the target frame time. In general, it is impossible to choose a single size threshold for LOD selection that generates uniform frame times for all observer viewpoints.

a) *Static* algorithm b) *Optimization* algorithm

Figure 8: Images depicting the LODs selected for each object at the observer viewpoints marked 'A' using the *Static* and *Optimization* algorithms. Darker shades of gray represent higher LODs.

The *Feedback* algorithm adjusts the size threshold for LOD selection adaptively based on the time taken to render previous frames in an effort to maintain a uniform frame rate. This algorithm generates a fairly uniform frame rate in situations of smoothly varying scene complexity, as evidenced by the relatively flat portions of the frame time curve shown in Figure 7c (frames 1–125). However, in situations where the complexity of the scene visible to the observer changes suddenly, peaks and valleys appear in the curve. Sometimes the frame time generated using the *Feedback* algorithm can be even longer than the one generated using the *Static* algorithm, as the *Feedback* algorithm is lured into a inappropriately low size threshold during times of low scene complexity. For instance, just before the viewpoint marked 'C', the observer is looking at a relatively simple scene containing just a few objects on the stage, so frame times are very short, and the size threshold for LOD selection is reduced to zero. However, at the viewpoint marked 'C', many chairs become visible suddenly as the observer spins around quickly. Since the adaptive size threshold is set very low, inappropriately high LODs are chosen for most objects (see Figure 9a), resulting in a frame time of 0.16 seconds. Although the size threshold can often adapt quickly after such discontinuities in scene complexity, some effects related to this feedback control (i.e., oscillation, overshoot, and a few very slow frames) can be quite disturbing to the user.

In contrast, the *Optimization* algorithm predicts the complexity of the model visible from the current observer viewpoint, and chooses an appropriate LOD and rendering algorithm for each object to meet the target frame time. As a result, the frame time generated using the *Optimization* algorithm is much more uniform than using any of

a) *Feedback* algorithm b) *Optimization* algorithm

Figure 9: Images depicting the LODs selected for each object at the observer viewpoints marked 'C' using the *Feedback* and *Optimization* algorithms. Darker shades of gray represent higher LODs.

the other LOD selection algorithms (see Figure 7d). For all observer viewpoints along the test path, the standard deviation in the frame time is 0.008 seconds, less than one third of any of the other three algorithms tested. The longest frame time is 0.13 seconds, and the shortest is 0.075 seconds.

As the *Optimization* algorithm adjusts image quality to maintain a uniform, interactive frame rate, it attempts to render the "best" image possible within the target frame time for each observer viewpoint. As a result, there is usually little noticeable difference between images generated using the *Optimization* algorithm and ones generated with no detail elision at all. A comparison of images for observer viewpoint 'A' generated using a) no detail elision, and b) using the *Optimization* algorithm to meet a target frame time of one tenth of a second are shown in Figure 10. Figure 10a has 72,570 polygons and took 0.98 seconds to render, whereas Figure 10b has 5,300 polygons and took 0.10 seconds. Even though there are less than a tenth as many polygons in Figure 10b, the difference in image quality is barely noticeable. For reference, the LOD chosen for each object in Figure 10b is shown in Figure 8b. Note that reduction in rendering time does not map to a linear reduction in polygon count since polygons representing lower levels of detail tend to be bigger on average.

The *Optimization* algorithm is more general than other detail elision algorithms in that it also adjusts the rendering algorithm (and possibly other attributes in the future) for each object independently. Examine Figure 11, which shows three images of a small library on the sixth floor of Soda Hall containing several textured surfaces. Figure $11a_1$, shows an image generated using no detail elision – it contains 19,821 polygons and took 0.60 seconds to render. Figures $11b_1$ and $11c_1$ show images generated for the same observer viewpoint using the *Optimization* algorithm with target frame times of b) 0.15 seconds (4,217 polygons), and c) 0.10 seconds (1,389 polygons). Although the *Optimization* algorithm uses lower levels of detail for many objects (see Figures $11b_1$ and $11c_1$), and generates images that are quite different than the one generated with no detail elision (see Figures $11b_2$ and $11c_2$), all three images look very similar. Notice the reduced tessellation of chairs further from the observer, and the omission of texture on the bookshelves in Figure $11b_1$. Similarly, notice the flat-shaded chairs, and the omission of books on bookshelves and texture on doors in Figure $11c_1$.

Having experimented with several LOD selection algorithms in an interactive visualization application, we are optimistic that variation in image quality is less disturbing to a user than variation in the frame times, as long as different representations for each object appear similar, and transitions between representations are not very noticeable. Further experimentation is required to determine which types of rendering attributes can be blended smoothly during interactive visualization.

# 9 Conclusion

We have described an adaptive display algorithm for fast, uniform frame rates during interactive visualization of large, complex virtual environments. The algorithm adjusts image quality dynamically in order to maintain a user-specified frame rate, selecting a level of detail and an algorithm with which to render each potentially visible object to produce the "best" image possible within the target frame time.

Our tests show that the *Optimization* algorithm generates more uniform frame rates than other previously described detail elision algorithms with little noticeable difference in image quality during visualization of complex models. Interesting topics for further study include algorithms for automatic generation of multiresolution models, and experiments to develop measures of *image quality* and *image differences*.

# 10 Acknowledgements

We are grateful to Thurman Brown, Delnaz Khorramabadi, Priscilla Shih and Maryann Simmons for their efforts constructing the building model. Silicon Graphics, Inc. has been very generous, allowing us to use equipment, and donating a VGX 320 workstation to this project as part of a grant from the Microelectronics Innovation and Computer Research Opportunities (MICRO 1991) program of the State of California. We appreciate the assistance of Greg Ward, Sharon Fischler, and Henry Moreton who helped generate the color prints for this paper. Finally, we thank Seth Teller for his spatial subdivisions, visibility algorithms, and other important contributions to this project.

# References

[1] Airey, John M., Rohlf, John H., and Brooks, Jr., Frederick P. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. *ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics*, 24, 2 (1990), 41-50.

[2] Blake, Edwin H. A Metric for Computing Adaptive Detail in Animated Scenes using Object-Oriented Programming. *Eurographics '87*. G. Marechal (Ed.), Elsivier Science Publishers, B.V. (North-Holland), 1987.

[3] Brooks, Jr., Frederick P. Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings. *Proceedings of the 1986 Workshop on Interactive 3D Graphics*.

[4] Clark, James H. Hierarchical Geometric Models for Visible Surface Algorithms. *Communications of the ACM*, 19, 10 (October 1976), 547-554.

[5] Funkhouser, Thomas A., Séquin, Carlo H., and Teller, Seth J. Management of Large Amounts of Data in Interactive Building Walkthroughs. *ACM SIGGRAPH Special Issue on 1992 Symposium on Interactive 3D Graphics*, 11-20.

[6] Garey, Michael R., and Johnson, David S. Computers and Intractibility: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York, 1979.

[7] Ibaraki, T., Hasegawa, T., Teranaka, K., and Iwase J. The Multiple Choice Knapsack Problem. *J. Oper. Res. Soc. Japan* 21, 1978, 59-94.

[8] Rossignac, Jarek, and Borrel, Paul. Multi-resolution 3D approximations for rendering complex scenes. *IFIP TC 5.WG 5.10 II Conference on Geometric Modeling in Computer Graphics*, Genova, Italy, 1993. Also available as IBM Research Report RC 17697, Yorktown Heights, NY 10598.

[9] Schachter, Bruce J. (Ed.). *Computer Image Generation*. John Wiley and Sons, New York, NY, 1983.

[10] *Graphics Library Programming Tools and Techniques*, Document #007-1489-01, Silicon Graphics, Inc., 1992.

[11] Teller, Seth J., and Séquin, Carlo H. Visibility Preprocessing for Interactive Walkthroughs. Proceedings of SIGGRAPH '91. In *Computer Graphics* 25, 4 (August 1991), 61-69.

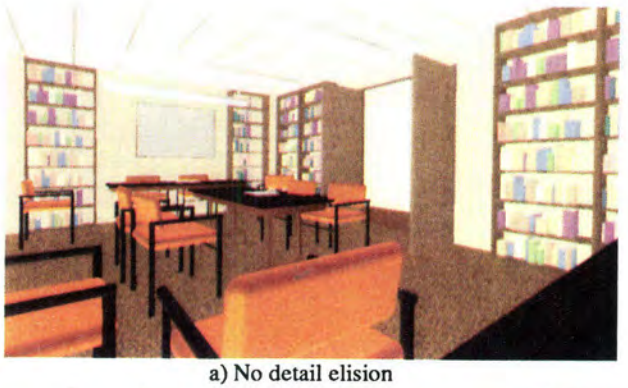
[12] Teller, Seth J. *Visibility Computations in Densely Occluded Polyhedral Environments*. Ph.D. thesis, Computer Science Division (EECS), University of California, Berkeley, 1992. Also available as UC Berkeley technical report UCB/CSD-92-708.

[13] Zyda, Michael J. Course Notes, Book Number 10, Graphics Video Laboratory, Department of Computer Science, Naval Postgraduate School, Monterey, California, November 1991.

a) No detail elision

b) *Optimization* algorithm (0.10 seconds)

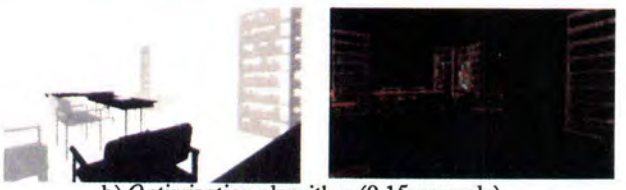
Figure 10: Images for observer viewpoint 'A' generated using a) no detail elision (72,570 polygons), and b) the *Optimization* algorithm with a 0.10 second target frame time (5,300 polygons).



$a_1$

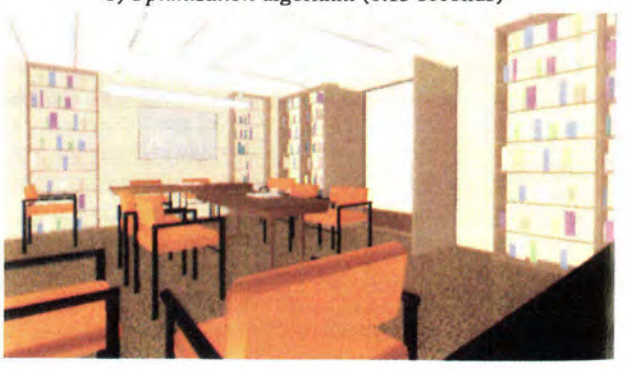
a) No detail elision

$b_1$



$b_2$

b) *Optimization* algorithm (0.15 seconds)



$c_1$

$c_2$ $c_3$

c) *Optimization* algorithm (0.10 seconds)

Figure 11: Images of library generated using a) no detail elision (19,821 polygons), and the *Optimization* detail elision algorithm with target frame times of b) 0.15 seconds (4,217 polygons), and c) 0.10 seconds (1,389 polygons). LODs chosen for objects in $b_1$ and $c_1$ are shown in $b_2$ and $c_c$ – darker shades of gray represent higher LODs. Pixel-by-pixel differences $abs(a_1 - b_1)$ and $abs(a_1 - c_1)$ are shown in $b_3$ and $c_3$ – brighter colors represent greater difference.

# Discrete Groups and Visualization of Three-Dimensional Manifolds

Charlie Gunn
The Geometry Center, The University of Minnesota *

## Abstract

We describe a software implementation for interactive visualization of a wide class of discrete groups. In addition to familiar Euclidean space, these groups act on the curved geometries of hyperbolic and spherical space. We construct easily computable models of our geometric spaces based on projective geometry; and establish algorithms for visualization of three-dimensional manifolds based upon the close connection between discrete groups and manifolds. We describe an object-oriented implementation of these concepts, and several novel visualization applications. As a visualization tool, this software breaks new ground in two directions: interactive exploration of curved spaces, and of topological manifolds modeled on these spaces. It establishes a generalization of the application of projective geometry to computer graphics, and lays the groundwork for visualization of spaces of non-constant curvature.

**CR Categories and Subject Descriptors:** I.3.3 [Picture/Image Generation] display algorithms I.3.5 [Computational Geometry and Object Modeling Graphics]: geometric algorithms, hierarchy and geometric transformations, I.3.7 [Three dimensional Graphics and Realism] color, shading, shadowing, and texture

**Additional Key Words and Phrases:** discrete group, tessellation, quotient space, projective geometry, hyperbolic geometry, spherical geometry, curvature, geodesic.

## 1 Discrete Groups

Symmetry, broadly speaking, implies a redundant supply of information. A mirror image contains the same information as the scene that it mirrors. The theory of discrete groups has been developed over the past 100 years as a formalization of the process of extracting a single copy of the information present in symmetric configurations. The discrete groups which we study here are groups of motions which act on a geometric space, such as Euclidean space, to produce tessellations by congruent non-overlapping cells. Familiar examples include wallpaper patterns, and the interlocking designs of M. C. Escher. We consider two simple examples before introducing mathematical definitions.



*Current address: SFB 288, MA 8-5, Technische Universität, Strasse des 17 Juni 136, 1 Berlin 12, Germany, gunn@sfb288.math.tu-berlin.de

### 1.1 The circle and the line

When we evaluate the expression $\sin(2\pi x)$ we are only interested in $x$ mod 1, since sin is a periodic function: $\sin(2\pi x) = \sin(2\pi(x+k))$, where $k$ is an integer. The set of all motions of the real line $R$ by integer amounts forms a group $\Gamma$, which leaves invariant the function $\sin(2\pi x)$. We can form the *quotient* $R/\Gamma$, which is the set of equivalence classes with respect to this group. This quotient can be represented by the closed interval $[0, 1]$, with the understanding that we identify the two endpoints. But identifying the two endpoints yields a circle. Once we know the values of $\sin(2\pi x)$ on the circle, we can compute it for any other value $y$, simply by subtracting or adding integers to $y$ until the result lies in the range $[0, 1)$.

In this example the *discrete group* $\Gamma$ is the set of transformations of $R$ given by all translations $x \to x + k$ , where $k$ is an integer. $\Gamma$ is *discrete* since no non-trivial sequence in $\Gamma$ converges to the the identity element. The *quotient* of $R$ under this action is $S^1$, the unit circle. We write $R/\Gamma = S^1$.

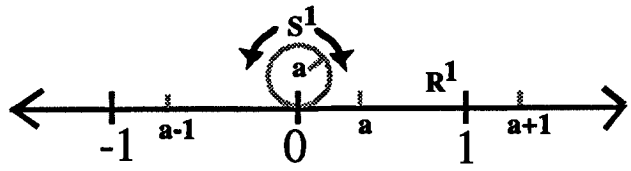



Figure 1: The circle is the quotient of $R$ by the integers.

$I = [0, 1)$ is a *fundamental domain* for this group action. We can recover $R$ from the fundamental domain and $\Gamma$: the union

$$\bigcup_{g \in \Gamma} gI$$

covers $R$ without overlap.

We move into two dimensions to bring out other features of the concepts introduced in this example.

### 1.2 The torus and the plane

Instead of $R$ we now work with $R^2$. Let $\Gamma$ be the group of translations of $R^2$ generated by $(x, y) \to (x + 1, y)$ and $(x, y) \to (x, y+1)$, that is, unit translations in the coordinate directions. What is the quotient $R^2/\Gamma$? Instead of the unit interval with its endpoints identified, we are led to a unit square that has its edges identified in pairs. If we imagine the square is made of rubber and that we can perform the identifications by bending the square and gluing, we find that the resulting surface is the torus $T^2$. See Figure 2.

 ACM-0-89791-601-8/93/008/0255 $01.50

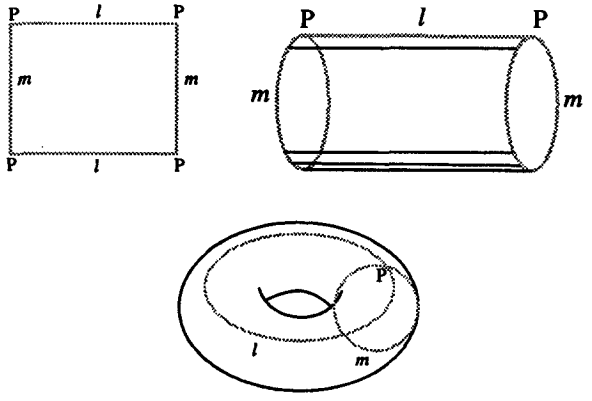



Figure 2: Making a torus from a square

## 1.3 Algebra and geometry: the fundamental group

A key element of this approach is the interplay of algebraic and geometric viewpoints. To clarify this, we introduce the *fundamental group* of a space, formed by taking all the closed paths based at some point $P$ in the space. We get a group structure on this set: we can add paths by following one and then the other, and subtract by going around the second path in the reverse order. The zero-length path is the identity element. If one path can be moved or deformed to another path, the two paths correspond to the same group element. It is easy to check that different $P$'s yield isomorphic groups. We say a space is *simply connected* if every closed path can be smoothly shrunk to a point, like a lasso, without leaving the space. [Mun75] The fundamental group of a simply connected space consists of just the identity element.

In the above example $R^2$ is simply connected; while $T^2$, the quotient, isn't. When $X$ is the quotient of a simply connected space $Y$, we say that $Y$ is the *universal covering space* of $X$. The importance of simply connected spaces in the study of discrete groups is due to a basic result of topology that (subject to technical constraints which we will consider satisfied) every space has a unique universal covering space [Mun75]. So in considering group actions, we need only consider actions on simply connected spaces.

The interplay of algebra and geometry reveals itself in the fact that the fundamental group of the quotient, a purely *topological* object, is isomorphic to the group of symmetries $\Gamma$, which arises in a purely *geometric* context.

## 1.4 Inside versus Outside Views

In the cases we will consider, the universal covering space $X$ is a *geometric* space, that is, it comes equipped with a metric that determines distance between points and angles between tangent vectors. In this case we sometimes refer to $X$ as a *model geometry*. This metric allows us to compute geodesics, or shortest paths, between points in the space [Car76]. The quotient space inherits this metric. $R^2$ is the universal covering space of $T^2$: if we unroll $T^2$ onto $R^2$, the copies of the torus will cover the plane completely, without overlap. We say these copies *tessellate* the plane. For some purposes the rolled-up torus sitting in $R^3$ is useful, but to gain the experience of what it is like to live *inside* the surface, we are better served by examining the *tessellation* of the universal covering space produced by the group.

For example, if we want to make pictures of what an inhabitant of $T^2$ sees, we will make them in $R^2$: Light follows geodesics, which appear to be very complicated on the rolled-up torus, but in $R^2$ are just ordinary straight lines. A complicated closed path based at $P$ which wraps around the torus several times unrolls in the universal cover to be an ordinary straight line connecting $P$ and $hP$ for some $h \in \Gamma$. See Figure 3. An immediate consequence of this is that an observer on the torus based at $P$ sees many copies of himself, one for every closed geodesic on the surface passing through $P$. See [Wee85] for a complete and elementary description of this phenomenon. We say the rolled-up torus represents the *outsider's* view; while the unrolled view we term the *insider's* view, since it shows what someone living inside the space would see. The importance of the insider's view becomes more telling in three dimensional spaces, since to "roll up" our fundamental domains requires four or more dimensions. In this case the insider's view becomes a practical necessity.

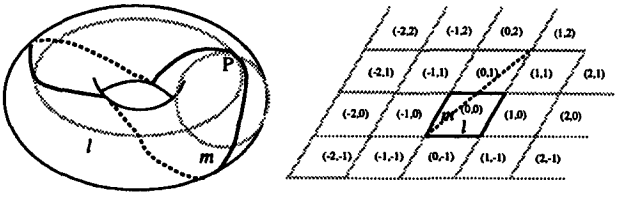



Figure 3: Outside and inside views of a complicated torus path

When we try to perform the analogous construction for the two-holed torus, instead of a square in the Euclidean plane $R^2$, we are led to a regular octagon in the hyperbolic plane $H^2$[FRC92]. We describe hyperbolic geometry in more detail below.

## 1.5 Definition of discrete group

A discrete group is a subgroup $\Gamma$ of a continuous group $G$ such that there is a neighborhood $U$ of the identity in $G$ with $U \cap \Gamma = I$, the identity element.

In the example of the torus above, the group $\Gamma$ *acts* on $R^2$. Such an action on a topological space $X$ is called *properly discontinuous* if for every closed and bounded subset $K$ of $X$, the set of $\gamma \in \Gamma$ such that $\gamma K \cap K \neq \phi$ is finite. In the cases to be discussed here, $\Gamma$ is discrete if and only if the action of $\Gamma$ is properly discontinuous.

If in addition the quotient space $X/\Gamma$ is compact, we say that $\Gamma$ is a *crystallographic*, or *crystal*, group.

The group of the torus discussed in 1.2 above is a crystallographic group, the simplest so-called *wallpaper* group. There are exactly 17 wallpaper groups of the Euclidean plane. See [Gun83] for a full discussion of this case and the details of a computer implementation.

## 1.6 Dirichlet domains

Given a discrete group, there is a technique for constructing a fundamental domain, known as a Dirichlet domain. We define it now for future reference. Given a discrete group $\Gamma$ acting on a space $X$ and a point $P \in X$, the orbit $O(P)$ of $P$ under $\Gamma$ is $\bigcup_{g \in \Gamma} gP$. Then the Dirichlet domain with respect

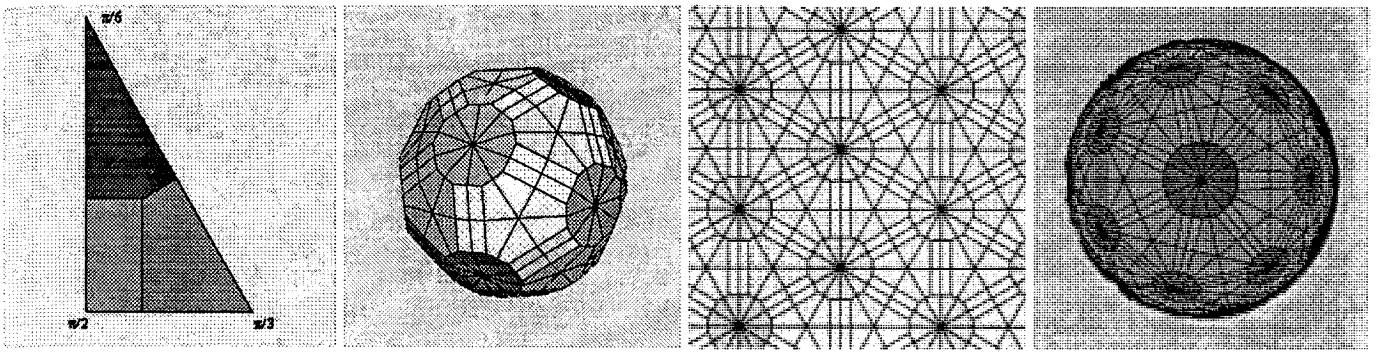



Figure 4: (235), (236) and (237) triangle groups tessellate $S^2, R^2$, and $H^2$.

to $P$ is the set of points in $X$ which are closer to $P$ than to any other point of $O(P)$. We can be more precise. For each $Q \in O(P)$, construct the perpendicular bisector $M$ of the segment $PQ$. Denote by $H_Q$ the half-space containing P bounded by $M$. Then the Dirichlet domain $D_P$ determined by $\Gamma$ and $P$ is

$$\bigcap_{Q \in O(P)} H_Q$$

In practice, for many of the groups the intersection can be assumed to involve only finitely many $H_Q$'s. The resulting polyhedron is convex. If a face $F$ is determined by $g \in \Gamma$, then $g^{-1}F$ will be a congruent face $F'$ determined by $g^{-1}$. This face pairing is used in the sequel. Note that, since $D_P$ depends upon $P$, there are potentially many different shapes for the Dirichlet domain for a given group. [Bea83] Computational geometers may recognize that a Dirichlet domain with respect to $P$ is a *Voronoi cell* with respect to the orbit of $P$.

## 2 Non-Euclidean Geometries

In the examples above, the model geometry was Euclidean. There are two other simply connected two-dimensional spaces in addition to $R^2$ which can serve as our model geometries: the sphere $S^2$ and the hyperbolic plane $H^2$. They have geometries (to be described in more detail below) which satisfy all the postulates of Euclidean geometry except for the Parallel Postulate: Given a line $L$ and a point $P$ not on $L$, there is a unique line $M$ passing through $P$ which is parallel to $L$. The sphere has no parallel lines; while $H^2$ has infinitely many for a given $L$ and $P$. See [Cox65] for an account of the discovery and development of these non-Euclidean geometries.

An equivalent characterization of Euclidean, spherical, and hyperbolic geometry is that the sum of the angles of a triangle is, respectively, equal to, greater than, or less than, $\pi$. Figure 4 shows tessellations of these three spaces by triangles with angles $(\pi/2, \pi/3, \pi/n)$, where $n = (5, 6, 7)$ yields spherical, Euclidean, and hyperbolic space.

We now turn to demonstrating models for these three geometries which share a common root in projective geometry. This will lead directly to techniques for visualizing discrete groups which act on these spaces.

### 2.1 Projective geometry

Projective geometry is the geometry of lines without regard to distance or measure. It was discovered at roughly the same time as the the non-Euclidean geometries discussed above; we show in the sequel how it can be considered to be the fundamental geometry out of which the other geometries arise.

The projective plane $P^2$ is gotten from the ordinary plane by adjoining a *line at infinity*. Projective space $P^n$ can be constructed in every dimension $n$ by adjoining an $n-1$ dimensional hyperplane at infinity. We assume the reader is familiar with homogeneous coordinates for projective space [Cox65]. The group of self-mappings of projective space $P^n$ can then be represented via homogeneous coordinates as elements of the matrix group $PGL(R, n+1)$, the projective general linear group. This group consists of all invertible matrices of dimension $(n+1) \times (n+1)$, where two matrices are equivalent if one is a scalar multiple of the other [Cox87]. Much of the success of the approach described in this paper is due to the circumstance that many computer graphics rendering transformation pipelines support $PGL(R, 4)$.

### 2.2 From projective to metric geometry

Projective geometry does not include a notion of distance or angle measure. However, every projective transformation preserves a quantity known as the **cross ratio** . The cross ratio is a function of four collinear points:

$$\lambda(AB, CD) = \frac{(A-C)(B-D)}{(B-C)(A-D)}$$

Here the points are represented by a homogeneous coordinate system on their common line; for convienience we can assume this is ordinary Euclidean measure on the line. This invariant has been used by Cayley to construct metric geometries on the foundation of projective geometry [Cay59].

First choose a homogeneous conic $Q$ which is to be invariant. The conic is known as the Absolute for the associated geometry. The projective transformations preserving $Q$ form a subgroup $H$ of the full projective group. Two given points $P_0$ and $P_1$ determine a line, which intersects the conic $Q$ in a pair of points $T_0$ and $T_1$, whose coordinates may be complex numbers. Then define a distance function

$$d(P_0, P_1) = K \log \lambda(T_0 T_1, P_0 P_1) \qquad (1)$$

where the constant $K$ is determined according to the nature of $Q$ in order to make the distance function real. Since the cross ratio is a multiplicative function, use of the log function yields an additive function. Measurement of angles between lines $L_0$ and $L_1$ proceeds in like manner, by determining

the two tangent lines to $Q$ which lie in the pencil of lines determined by $L_0$ and $L_1$.

This yields models for spherical, hyperbolic, and Euclidean geometry which share the same straight lines; what is different is how distance along them and between them is measured. The subgroup $H$ becomes the isometry group for the metric geometry.

We will for simplicity's sake work in two dimensions, that is, with homogeneous coordinates $(x, y, w)$, and consider only distance measurement, not angle measurement. All our results generalize directly to arbitrary higher dimension. Since the cases of spherical and hyperbolic geometry are more straightforward, we begin with them.

#### 2.2.1 Spherical geometry

For the spherical case, we choose $Q$ to be the totally imaginary conic $x^2 + y^2 + w^2 = 0$. The proper choice for $K$ is $i/2$. We can derive from $Q$ an *inner product* between pairs of points: if $P_0 = (x_0, y_0, w_0)$ and $P_1 = (x_1, y_1, w_1)$ then $P_0.P_1 = x_0x_1 + y_0y_1 + w_0w_1$. Then (1) reduces to:

$$d(P_0, P_1) = \arccos\left(\frac{P_0.P_1}{\sqrt{(P_0.P_0)(P_1.P_1)}}\right)$$

This is the familiar measurement between points on the unit sphere. Projective transformations which preserve Q constitute the special orthogonal group SO(3), the group of rotations of three-dimensional Euclidean space. Although it is tempting to consider the familiar picture of $S^2$ sitting isometrically in $R^3$, it is more appropriate to think of the model presented purely in terms of $P^2$. In this model, to each point of $P^2$ we assign two antipodal points of $S^2$.

#### 2.2.2 Hyperbolic geometry

For the hyperbolic case, we choose $Q$ to be the totally real conic $x^2 + y^2 - w^2 = 0$, a cone aligned with the $w$-axis. The correct choice for $K$ is $\frac{1}{2}$. The derived inner product of two points $P_0 = (x_0, y_0, w_0)$ and $P_1 = (x_1, y_1, w_1)$ is then $P_0.P_1 = x_0x_1 + y_0y_1 - w_0w_1$, sometimes called the Minkowski inner product. Our model for hyperbolic geometry will consist of the interior of this cone, where $P.P < 0$. Then (1) reduces to:

$$d(P_0, P_1) = \operatorname{arccosh}\left(\frac{P_0.P_1}{\sqrt{(P_0.P_0)(P_1.P_1)}}\right)$$

where $P_0$ and $P_1$ lie in the interior of the cone. The isometry group is SO(2,1), the so-called Minkowski group.

Consider the hyperboloid of two sheets $H$, defined by the condition $P.P = -1$. Just as the unit sphere is a model for spherical geometry, the upper sheet of $H$ is a model for hyperbolic geometry. The most convenient model for $H^2$ is hidden within $H$. Consider the plane $w = 1$. It intersects $Q$ in a circle that bounds a disk $D$. We can project our hyperboloid $H$ onto $D$ from the origin. This projection respects the distance function defined above (it is, after all, a projective invariant). Then $D$ is a model of hyperbolic geometry, the so-called Klein or projective model. It is shown in the right-most figure in Figure 4. In three dimensions, this yields a model of $H^3$ as the interior of the unit ball in $R^3$. There are several other commonly used models of hyperbolic geometry, most notably the Poincaré or conformal model [Bea83]. Our choice of the projective model here was determined by the fact that it yields the correct results for visualizing the insider's view.

#### 2.2.3 Euclidean geometry

Euclidean, or parabolic, geometry arises when apply a limiting process to the conic $\epsilon(x^2 + y^2) + w^2 = 0$. As $\epsilon \to 0$, the expression for distance reduces to

$$d(P_0, P_1) = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

where $P_0$ and $P_1$ have been dehomogenized. The isometry group of this geometry $E(2)$ is the semi-direct product of $SO(2)$, the circle, and $R(2)$, the two-dimensional Euclidean translation group.

### 2.3 Comments

This development in terms of projective geometry is given fully in [Woo22] and is due to Cayley and Klein. For a treatment derived from the modern differential geometric viewpoint see [Car76]; for an implementation description following this viewpoint see [Gun92].

To justify the use of the names spherical and hyperbolic it is worthwhile to verify that the geometries induced by the indicated metrics on the indicated subspaces in fact yield geometries which behave correctly with respect to parallel lines and sums of angles of triangles.

For a detailed discussion of how to construct isometries of hyperbolic 3-space in the projective model discussed here see [PG92].

The above results, stated for the two-dimensional case, can be extended to arbitrary dimension.

## 3 Manifolds and Discrete Groups

An $n$-dimensional *manifold*, or $n$-manifold, is a topological space $X$ such that $X$ is locally homeomorphic to $R^n$, that is, every point of $X$ has a neighborhood that can be mapped 1-1 and continuously onto a small ball in $R^n$. If in addition we can realize $X$ as the quotient of a geometric space $M$ by a discrete group, we say that $X$ has a *geometric structure* modeled on $M$. A related concept to that of manifold is *orbifold*. An orbifold is like a manifold, but it may have singular points where it is locally homeomorphic not to $R^n$ but rather to the quotient of $R^n$ by a finite group. Orbifolds arise, generally speaking, when the elements of the discrete group have fixed points, such as rotations or reflections.

Initial work on the connection of discrete groups and theory of manifolds was done by Henri Poincaré in the 1880's. To this day much research in this field is driven by the Poincaré Conjecture, which asserts that a closed, connected, simply connected 3-dimensional manifold is homeomorphic to the 3-dimensional sphere $S^3$. This conjecture is closely related to the classification problem: making a list of all 3-manifolds. For example, in dimension 2, there is a uniformization theorem which says that any closed 2-dimensional manifold has a geometric structure modeled on one of $S^2$, $R^2$, or $H^2$. Recent work by Thurston and others has shown that many (possibly all) 3-manifolds have essentially unique geometric structures. That is, there are good reasons to believe that to every 3-manifold there corresponds an essentially unique discrete group [Thu82].

The geometric structures for 3-manifolds come from eight model geometries: $R^3$, $S^3$, and $H^3$ plus five additional simply connected spaces. The additional five are not as nice as the first three, since they are not *isotropic*: not all directions in space are the same. In any case, the most prevalent

geometric structure is hyperbolic. For a description of these eight geometries, see [Thu82], [Thuar]. The current software implementation does not support these five additional geometries.

In the discussion that follows, we will concentrate on the insider's, rather than the outsider's, view of three dimensional orbifolds. That is, we will look at the tessellations of the simply connected space (Euclidean, hyperbolic, or spherical) induced by discrete groups.

# 4 Software Implementation

## 4.1 OOGL

In order to visualize the spaces under consideration, we have developed an implementation within an object-oriented graphics library, OOGL. The generic OOGL class is **Geom**. Subclasses include include geometric primitives such as **PolyList**, **Vect**, **Bezier**, and **Mesh**; and organizational objects such as **List** and **Inst** (for instancing geometry). Methods with which **Geoms** come equipped include: **Bound, Create, Copy, Delete, Save, Load, Pick, and Draw** .

An interactive viewer, **Geomview** [MLP+], has been constructed based upon OOGL. It supports viewing in the three geometries discussed above: Euclidean, hyperbolic, and spherical. This is possible since as noted above isometries in the three geometries can be expressed as elements of $PGL(R,4)$. The underlying low-level graphics libraries (in the case of OOGL, GL or Renderman [1] ) support the use of elements of $PGL(R,4)$ for modeling and viewing transformations. This is a result of the fact that $PGL(R,4)$ is the smallest group which contains both the Euclidean isometries and the perspective transformation. The visualization task is also made easier by the fact that OOGL supports 4-dimensional vertices within all primitives. This provides a base for creating geometric models in hyperbolic and spherical space using homogeneous coordinates.

## 4.2 Shading

We have established how it is possible to implement non-Euclidean isometries using standard projective transformations. We have not addressed the question of correct lighting and shading of surfaces in these spaces. Indeed, the standard shading algorithms (in contrast to the standard transformations) are implicitly Euclidean. In order to model the behavior of light correctly in these non-Euclidean spaces, it is necessary to provide customized shaders which replace the default ones. This has been successfully achieved within the Renderman shading language [Ups89],[Gun92]. Figure 5 shows a view inside hyperbolic space from the movie "Not Knot". Interactive software shaders for OOGL for hyperbolic and spherical space have also been written.

These custom shaders use the expressions for distance and angle described in 2.2 to replace the Euclidean ones. Additionally, the decay of light intensity as a function of distance depends on the formula for the surface area of a sphere in each space. That is, the amount of light falling on an area element at distance $d$ from a light source will be inversely proportional to the total area of the sphere with radius $d$. For example, in hyperbolic space light decays exponentially: the area of a sphere of radius r is given by $k\sinh(r)$ and $\sinh(r) \approx \exp(r)$ for large $r$. The shaders used to create figures 7 and 9 also involve a term to model fog.

[1] GL is a trademark of Silicon Graphics, Inc.; and Renderman, of Pixar.

# 5 The DiscreteGroup class

The **DiscreteGroup** class is a subclass of **Geom**. The minimal data includes a set of generating isometries represented by elements of $PGL(R,4)$ and some geometric data, represented by other OOGL objects. The **DiscreteGroup** class supports the standard methods listed above, and other methods of its own.

Because of the close connection to manifolds outlined in Section 3, it can also be thought of as a **Manifold** class. Many design decisions were made to support visualization of the insider's view of a manifold. From this point of view, every element of the scene description belongs to the manifold and hence should be tessellated by the group in the process of creating the insider's view. We have departed from this philosophy in one important respect: we do not tessellate the lights contained in the scene description. To do so would have sacrificed interactivity for a questionable increase in authenticity.

Points of interest among **DiscreteGroup** methods include:

## 5.1 File format

There is an ascii file format for loading and saving discrete groups. This format supports the three geometries described above, and includes lists of generators and group elements and also geometric objects for display within the tessellation.

## 5.2 DiscreteGroupDraw

Each **DiscreteGroup** instance includes a list of group elements and a collection of other **Geoms**. The general algorithm transforms each **Geom** by each group element and then draws it. There are some subtleties. Most of these groups are infinite, but we only compute and store a finite list of elements at any time. One of the difficulties of navigating in the tessellations produced by discrete groups is that normal flight tends to wander to the edge of the computed tessellation. To solve this problem, the **DiscreteGroup** object is provided with an automatic centering mechanism. It detects when the camera leaves the Dirichlet domain defined by the group, and moves the camera by an isometry (determined by the face-pairings), to stay within this central region. Note that since lighting is not tessellated, lights must be defined within the camera coordinate system in order that lighting is invariant under this movement.

Another added feature is that there is a separate associated **Geom** which represents the camera, or observer. Before being tessellated it is moved to the location of the camera, which as described above is constrained to stay within the Dirichlet domain. The observer then becomes aware of his own movement in the space. This is an important feature especially for detecting the singular locus of orbifolds. For example, when the camera approaches a axis of symmetry of order $n$ in an orbifold, this fact is made clear by the approach of $n-1$ other copies of the camera to the same axis, a symmetry which the geometry of the Dirichlet domain alone may not reveal.

### 5.3 DiscreteGroupEnum(int constraint() )

is a method for enumerating lists of group elements given the generators. One such list is used by the draw routine: it defines which copies of the fundamental domain to draw. The constraint function accepts a single group element and returns 0 or 1 according to whether it satisfies its criteria. For example, a matrix may be rejected if it moves the origin far, its determinant is small, or its expression as a word in the generating elements is long. This enumeration software uses software acceleration provided by the theory of automatic groups [ECH+91], [Lev92] if an automatic structure has been provided for the discrete group.

### 5.4 DiscreteGroupDirDom

creates a fundamental domain using the Dirichlet domain algorithm described above. This is useful for exploring groups for which no other geometry has been provided. For display purposes, both a wire-frame of the full polyhedron and a possibly scaled version with faces colored to reflect the face-pairing identities are drawn. See Figure 9. The user can deduce features of the group by examining the face-pairing patterns, or by moving the distinguished point $P$.

## 6 Example applications

A variety of applications have been developed based on the DiscreteGroup software class.

Maniview is short for *Manifold Viewer*. In the paradigm of object-oriented software tools, it is essentially an Inspector for the class DiscreteGroup . Maniview communicates with Geomview via a two-way pipe. Geomview reads the description of the discrete group output by Maniview and displays it. The user typically loads a discrete group into Maniview, and then manipulates the discrete group via a set of control panels. These panels are grouped into: display settings, enumeration of group elements, choice of fundamental tile, and saving and loading various elements. A typical snapshot of a Maniview session is shown in Figure 8.

One of the milestones in the theory of discrete groups was the enumeration of the 230 crystal groups in three dimensional Euclidean space at the end of the nineteenth century. For a survey see [LM78],[Sch80]. eucsyms, an interactive application which allows the exploration of these groups has been developed by Olaf Holt at the Geometry Center, and adapted to use the DiscreteGroup software. eucsyms is connected by a two-way pipe with Maniview. Figure 6 shows a view inside the symmetry group $r\mathcal{I}$.

We have also hooked up Maniview to a powerful program for computing hyperbolic structures on three dimensional manifolds, snappea by Jeff Weeks [Wee93]. This is a popular tool used by research topologists to construct and examine three dimensional manifolds.

Geomview, Maniview, eucsyms, and snappea are all available via anonymous ftp from geom.umn.edu [128.101.25.35]. Some of the computation of the groups and geometrical models shown in the figures have been computed using a Mathematica [2] package developed at the Geometry Center, also available via anonymous ftp from the same site.

[2] Mathematica is a trademark of Wolfram Research, Inc

## 7 Example spaces

### 7.1 "Not Knot"

The mathematical animation "Not Knot" [GM91] pioneered the visualization of the insider's view of hyperbolic space. It features one Euclidean orbifold (see Figure 7) and a series of hyperbolic orbifolds converging to a hyperbolic manifold that is the complement of the three linked circles known as the Borromean rings. Figure 5 shows one of these orbifolds, which tessellates $H^3$ with right-angled dodecahedra. One of the six generators is a rotation of $\frac{\pi}{2}$ around the large red axis. As a matrix this generator is:

$$\begin{pmatrix} -1.618033 & 1.618033 & 0 & -2.058171 \\ -1.618033 & 0 & 0 & -1.272019 \\ 0 & 0 & 1. & 0 \\ 2.058171 & -1.272019 & 0 & 2.618033 \end{pmatrix}$$

Note that all the non-zero entries are powers of the golden ratio. This is an example of an *arithmetic* group and is of particular mathematical interest.

The discrete groups underlying "Not Knot" have been converted into the DiscreteGroup format. Now, viewers interested in exploring the spaces depicted in "Not Knot" can do so.

### 7.2 The Poincaré homology sphere

Possibly the most famous three dimensional spherical manifold is the so-called Poincaré homology sphere. It arises abstractly by identifying the opposite faces of a regular dodecahedron with a twist of $\pi/5$. The tessellation of $S^3$ corresponding to this manifold consists of 120 regular dodecahedra, which meet 3 around each edge, and is known as the 120-cell or dodecahedral honeycomb [Cox73]. In contrast to the right-angled dodecahedron of hyperbolic space, these dodecahedra have dihedral angles of $\frac{2\pi}{3}$.

An inside view of this manifold appears in Figure 9. Note that the largest dodecahedron, which completely fills the view as if it surrounds the viewer, is also the farthest away. This is a typical feature of life in spherical space; as objects move away they decrease in size until they reach a maximum distance of $\pi/2$, then they begin to increase in size until they reach the antipodal point of the viewer at a distance of $\pi$, where they expand to fill completely the field of view, since every geodesic leaving the observer also passes through the antipodal point. Stereo viewing in spherical space would place great strain on Euclidean trained eyes: when an object is exactly at the equator, the lines of sight from an observer's eyes are parallel; as an object moves beyond the equator, the observer must look "anti-crosseyed" at it.

## 8 Directions for further work

Common ancestry in projective geometry means that some important procedures can be shared with traditional Euclidean systems. However, there remain a host of computer graphics issues related to modeling and animation in non-Euclidean spaces to be addressed. Many geometric constructions are very different. For example, consider a equidistant curve, that is, the set of points equidistant from a line. In the Euclidean plane an equidistant curve is a parallel line. But equidistant curves in spherical and hyperbolic space are not straight lines. What, then, is the proper generalization