Textures and Radiosity: Controlling Emission and Reflection with Texture Maps

Reid Gershbein **Peter Schröder** **Pat Hanrahan**

Department of Computer Science
Princeton University

Abstract

In this paper we discuss the efficient and accurate incorporation of texture maps into a hierarchical Galerkin radiosity algorithm. This extension of the standard algorithm allows the use of textures to describe complex reflectance and emittance patterns over surfaces, increasing the realism and complexity of radiosity images. Previous approaches to the inclusion of textures have either averaged the texture to yield a single color for the radiosity computations, or exhaustively generated detail elements—possibly as many as one per texture pixel. The former does not capture important lighting effects due to textures, while the latter is too expensive computationally to be practical.

To handle texture maps requires a detailed analysis of the underlying operator equation. In particular we decompose the radiosity equation into two steps: (i) the computation of irradiance on a surface from the radiosities on other surfaces, and (ii) the application of the reflectance operator ρ to compute radiosities from irradiances. We then describe an algorithm that maintains hierarchical representations of both radiosities and textures. The numerical error involved in using these approximations is quantifiable and a time/error tradeoff is possible. The resulting algorithm allows texture maps to be used in radiosity computations with very little overhead.

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*; J.2 [Physical Sciences and Engineering]: *Engineering*.

Additional Key Words and Phrases: global illumination, wavelets, hierarchical radiosity, texture mapping.

1 Introduction

Radiosity methods compute the illumination, both direct and indirect, from environments consisting of perfectly diffuse (Lambertian) reflecting and emitting surfaces [10, 15, 6, 5]. The resulting pictures contain subtle but important lighting effects, such as soft shadows and color bleeding, that enhance their realism. However, radiosity algorithms are still complex and far from general. For example, most computer graphics radiosity algorithms are limited to environments consisting of polygonal elements whose radiosities and reflectances are constant. Naturally such assumptions limit both the realism of the resulting pictures, and the utilization of the radiosity method in many applications. This immediately leads to the question of what additional rendering and modeling techniques can be combined with the radiosity method to further enhance the realism of the final imagery.

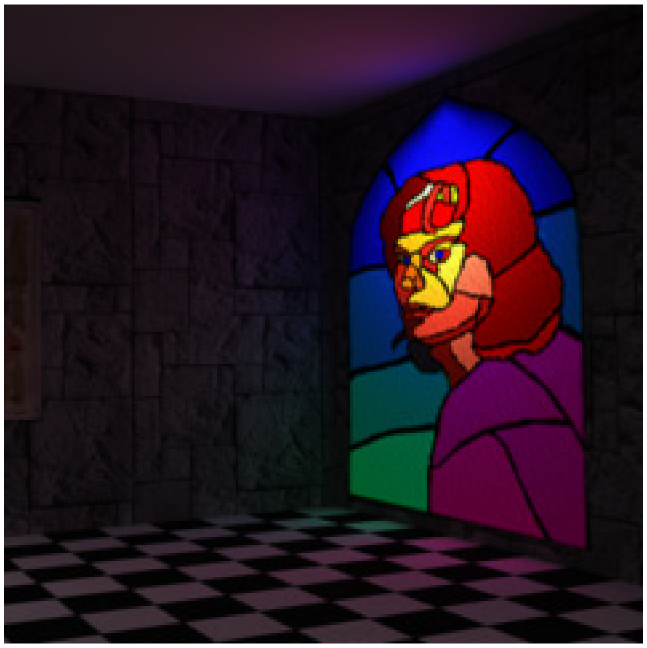

Figure 1: *A complex scene illustrating the lighting effects due to texture maps. The window and surrounding wall are a single polygon with emittance and reflectance textures (see Figure 9).*

One of the oldest rendering techniques is the use of texture maps to modulate surface color [3]. In the case of radiosity, texture maps may be used to spatially modulate the emittance and reflectance of the surface. The advantage of texture maps is that the apparent surface complexity increases without the increase in geometric complexity that arises if the surface were to be subdivided into many small surface polygons. This last distinction is particularly important in the case of radiosity, since the computational cost of a radiosity simulation may grow quadratically with the number of primitives in the scene.

These difficulties notwithstanding earlier researchers have incorporated emissive as well as reflective textures into radiosity systems to increase the realism of the generated images. Cohen *et al*. [5] incorporated textures as a post process. During the radiosity computations a texture mapped polygon would have a constant reflectance equal to the average texture map reflectance. For final renderings the textures were incorporated at the resolution of the original texture. This was done by dividing the computed radiosities by the average reflectance and then multiplying them by the true reflectance. This yields visually complex images, but does not accurately account for lighting effects because the spatially varying emission and reflection across a surface is not considered during the radiosity calculation. Dorsey *et al*. [8] also incorporated texture maps. They were motivated by the need to simulate lighting effects for opera lighting design. Consequently the simple average/post-process technique was not applicable. Instead they created many small polygons to get a reasonable approximation of the original texture. As mentioned above this quickly leads to excessive computation times, but does lead to extraordinary pictures.

To summarize, the averaging technique does not compute the correct illumination effects, and the exhaustive approach is expensive in both time and storage. An approach that captures the indirect illumination effects due to texture maps to within some accuracy while having low computational cost is clearly desirable, and is the motivation behind this paper.

Two developments in recent radiosity research are relevant for our present discussion, hierarchical radiosity, and higher order Galerkin methods. A two level hierarchy was first proposed by Cohen *et al*. [7] as substructuring. They allowed coarse subdivision of sources while requiring a finer subdivision of receivers. Hanrahan *et al*. [12] introduced a multi level hierarchy. In their algorithm objects are allowed to exchange energy at many different levels of detail. This is based on the observation that interactions between relatively far away primitives can be approximated with a coarse subdivision, while only close interactions require a fine subdivision. The level of detail at which two objects (or parts of objects) interact is determined by their ability to capture an interaction correctly within a specified error criterion. An asymptotically faster radiosity algorithm results.

Galerkin approaches, first introduced by Heckbert [13] and consequently elaborated by Zatz [20], and Troutman and Max [18], use classical finite element techniques to solve the underlying radiosity integral equation. The main goal is to compute smoother answers than classical radiosity. The higher order basis functions also reduce the need for subdivision and accelerate convergence overall. In any Galerkin approach all functions (emittance, reflectance, and the geometric kernel) are written out with respect to some set of basis functions. It has been pointed out by Zatz that this trivially allows for emittances which vary in some complex way over a surface so long as this variance can be described with the chosen set of basis functions.

The benefits of hierarchical radiosity and Galerkin methods have recently been unified under the framework of *wavelets* by Gortler *et al*. [11] and Schröder *et al*. [16].

Starting with a wavelet radiosity system we extend it to handle both emission and reflection textures. The theory and the resulting algorithm is the subject of this paper. We first carefully examine the structure of the underlying operator equation. In particular we show that the usual computation of radiosity as the product of reflectance and irradiance (and possibly active emission) is complicated by the hierarchical framework, which aims to preserve the computational advantage afforded by multi level descriptions. In fact, we formally and algorithmically separate the computation of irradiance from that of radiosity. This allows us to exploit economies in each of the computational steps which would be hard to take advantage of in the usual combined framework. In the resulting algorithm, the cost of performing a radiosity computation with texture maps is very similar to the cost without texturing.

2 Theory

2.1 The radiosity equation

The radiosity equation can be written as a simplification of the rendering equation [14] by assuming that all participating surfaces are perfectly diffuse (Lambertian), giving rise to the classical radiosity integral equation

$$\begin{aligned} B(x) &= B^e(x) + B^r(x) = B^e(x) + \rho(x)E(x) \qquad (1) \\ &= B^e(x) + \rho(x)\int dA_{x'}G(x,x')\pi^{-1}B(x') \end{aligned}$$

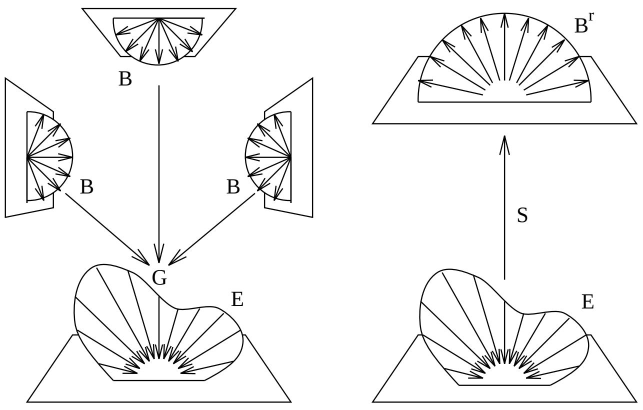

Figure 2: *Reflected radiosity,B^r, is computed in two steps. First the computation of irradiance, denoted by G, second the computation of the scattered irradiance, denoted by S.*

where $B(x)$ is the radiosity at point x, consisting of emitted and reflected radiosity, $\rho(x)$ the reflectance, and $G(x,x') = \frac{\cos\theta_x \cos\theta_{x'}}{\|x-x'\|^2}V(x,x')$ characterizes the radiant coupling between points x and x'. G accounts for relative surface orientations, distance, and visibility, $V=1$ or $V=0$, depending on whether x can or cannot see x'. The integral is taken over the hemisphere about x and represents the amount of energy per unit area received from other surfaces. This is the irradiance $E(x)$, which contributes to radiosity after multiplication by $\rho(x)$.

2.2 The radiosity operator

The radiosity equation may be decomposed into two steps as shown in Figure 2:

1. *Compute irradiance from scene radiosities.*
   Define the operator $\mathcal{G}$ for a general function f
   $$\mathcal{G}(f)(x) = \int dA_{x'}G(x,x')\pi^{-1}f(x')$$
   then $E(x) = \mathcal{G}(B)(x)$. Note that $\mathcal{G}$ is a geometric operator that takes into account the radiant exchange between a receiver surface and all other source surfaces in the scene.
   $\mathcal{G}$ is dense and non-local, but it is smooth (in the absence of shadows). It is dense since $\mathcal{G}$ may potentially couple any part of the environment to any other part. It is smooth because $\mathcal{G}$ varies slowly as a function of position on the source and receiver. In the parlance of integral equations we say that $\mathcal{G}$ has a smooth kernel [2].
2. *Compute radiosities from irradiances and reflectivities.*
   Define the scattering operator $\mathcal{S}$
   $$\mathcal{S}(f)(x) = \rho(x)f(x)$$
   yielding $B^r(x) = \mathcal{S}(E)(x)$. Note that $\mathcal{S}$ is completely determined by surface properties and as such does not depend on the environment.
   In contrast to $\mathcal{G}$, $\mathcal{S}$ is a local operator but not always smooth. This is because the texture map representing the reflectance may contain high frequencies. Since it is a local operator it does however have efficient representations as a diagonal, sparse operator in the proper basis.

Using this operator notation we can rewrite Equation 1 as

$$B(x) = B^e(x) + \mathcal{K}(B)(x) = B^e(x) + \mathcal{S}\circ\mathcal{G}(B)(x)$$

where $\circ$ denotes concatenation. The separation into $\mathcal{G}$ and $\mathcal{S}$ allows us to pursue different strategies to efficiently represent each operator individually. Sparse operators such as $\mathcal{S}$ are easy to represent efficiently. In contrast, dense operators such as $\mathcal{G}$ are normally difficult to represent efficiently. However, methods exist to efficiently approximate dense, smooth integral operators [2]. In particular, hierarchical or wavelet radiosity algorithms provide an efficient method for representing $\mathcal{G}$ [12, 11, 16]. They work because the smoothness of $\mathcal{G}$ allows it to be approximated efficiently using a hierarchy of levels of detail. Unfortunately, combining $\mathcal{S}$ with $\mathcal{G}$ creates an operator which is dense, but no longer smooth, causing difficulties for the hierarchical methods.

2.3 Projection methods

In order to make a radiosity system tractable it must be approximated by using a finite dimensional function space. This is done by projecting the radiosity, emittance, and reflectance functions onto a finite set of basis functions $\{N_i\}_{i=1,\ldots,n}$ (for some arbitrary but fixed n). For example, these basis functions might be piecewise constant, as in classical radiosity, they might be higher order polynomials as proposed by Zatz [20] for Galerkin radiosity, or wavelets [11]. The level of resolution in the reflectance is typically limited by the solution of the input geometry. We treat the case of incorporating reflectance functions which have potentially much finer resolution than the element subdivision induced by the radiosity solver. The term *nodal basis* will be used to describe a basis consisting of functions at some finest level. For example, a piecewise constant or linear basis over small elements. This is in contrast to hierarchical bases which contain functions at many levels of resolution and often overlapping support. The projection results in approximations $B(x) \approx \hat{B}(x) = \sum_{i=1}^{n} B_i N_i(x)$, $\rho(x) \approx \hat{\rho}(x) = \sum_{l=1}^{n} \rho_l N_l(x)$. Limiting everything to these basis functions, the solution to the original radiosity integral equation can be approximated by solving the following linear system[1]

$$\forall i: \; B_i = B_i^e + \sum_{j=1}^{n} K_{ij} B_j$$

$$K_{ij} = \int dx\, \hat{\rho}(x) \int dA_{x'}\; G(x, x') \pi^{-1} N_j(x') N_i(x)$$

Writing it in this conventional form (e.g. for constant radiosity $K_{ij} = \rho_i F_{ij}$) we have approximated the *compound* action of $\mathcal{S} \circ \mathcal{G}$ by a single set of coefficients K_{ij}.

Instead let us consider each step separately. First irradiance is computed with respect to the chosen basis. This step is not fundamentally different from the compound step if ρ is assumed constant across the support of the given basis function. In this case one typically finds a computation step of the form $E_i = \sum_{j=1}^{n} F_{ij} B_j$. However, if $\rho(x)$ is allowed to vary, the second step, multiplication with $\rho(x)$, changes in a significant way. Since $\rho(x)$ is given in some basis (typically the same as that used for E), we get

$$B^r(x) = \hat{\rho}(x) E(x) = \left(\sum_{l=1}^{n} \rho_l N_l(x) \right) \left(\sum_{i=1}^{n} E_i N_i(x) \right) \quad (2)$$

Finally the emitted radiosity is added. This is straightforward and we will therefore focus on the scattering operator in the remaining discussion.

2.4 Representation of $\mathcal{S}$

If our chosen basis is a piecewise constant nodal basis, as is typically used in classical radiosity, the application of $\mathcal{S}$ is straightforward. For such a basis the supports of two basis functions with

[1] To avoid cluttering the following derivations and help expose the fundamental connections between basis functions, we will ignore all scaling constants and assume an orthogonal basis.

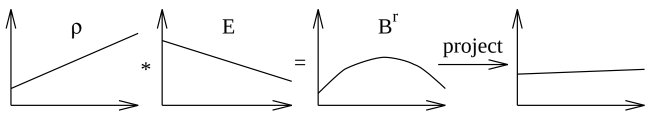

Figure 3*: Multiplying an irradiance and reflectance, both representable by a given basis (e.g. piecewise linear), can yield a reflected radiosity (e.g. piecewise quadratic) not representable in the same basis, requiring a reprojection step.*

different index do not intersect. Therefore the application of the operator reduces to a simple multiplication of E_i's and ρ_i's. In this basis $\mathcal{S}$ is a diagonal, sparse operator. Suppose instead that our basis was a piecewise polynomial basis of some higher order, but still with local support. In this case the matrix corresponding to $\mathcal{S}$ would be block diagonal. Each block couples the basis functions with overlapping support. Once again the resulting operator is sparse since only a small constant number of functions overlap.

The important observation is that the local support property of a given basis results in a (block-) diagonal scattering operator $\mathcal{S}$, which can be executed efficiently due to its sparse nature.

When multiplying basis functions a difficulty arises. Consider the use of a piecewise linear basis set and the case depicted in Figure 3. A linearly varying irradiance is multiplied with a linearly varying reflectance resulting in a quadratically varying radiosity. The latter cannot be represented in the chosen basis. The radiosity is of higher polynomial order than the individual component functions. This difficulty can be addressed by following up the multiplication with a reprojection step (see the rightmost step in Figure 3). The error incurred by the reprojection will be analyzed below in the context of the proper level of representation for irradiance and radiosity.

The above operator decomposition can be turned into a straightforward algorithm. Whichever basis is used, the application of $\mathcal{S}$ leads to the following four step algorithm:

Algorithm 1

1. Transform irradiance from its preferred basis to a nodal basis
2. Apply (block-) diagonal $\mathcal{S}$
3. Reproject onto the nodal basis
4. Transform back to preferred basis for radiosity

This algorithm has the advantage that $\mathcal{S}$ is represented efficiently. However, it has the disadvantage that it requires an intermediate transformation between the preferred basis and the nodal basis. In general, this transformation may require n^2 operations. Fortunately, if the preferred basis is the wavelet basis, the transformation to and from the nodal basis requires only a linear number of operations.

At this point the two step decomposition of $\mathcal{G}$ and $\mathcal{S}$ has already resulted in a very efficient algorithm. The first step proceeds as usual, but by separating out the second step we avoid forcing unnecessary subdivision on all surfaces due to the presence of textures. The second step is linear in the complexity of the texture because of the diagonal, sparse nature of $\mathcal{S}$. However, we can do even better since much of the detail computed in going to the finest level will not be needed for the next iteration step.

An obvious improvement would be to express $\mathcal{S}$ directly in the wavelet basis. Unfortunately, the multiplication of the representations of ρ and E is more expensive. The support of many of the functions may overlap, resulting in many non-zero terms. Hence, the representation of $\mathcal{S}$ is no longer sparse. Fortunately, if we are only computing to within some chosen accuracy it may turn out that many of the products in Equation 2 make a negligible contribution to the total. It would be desirable to *only* compute the components that matter.

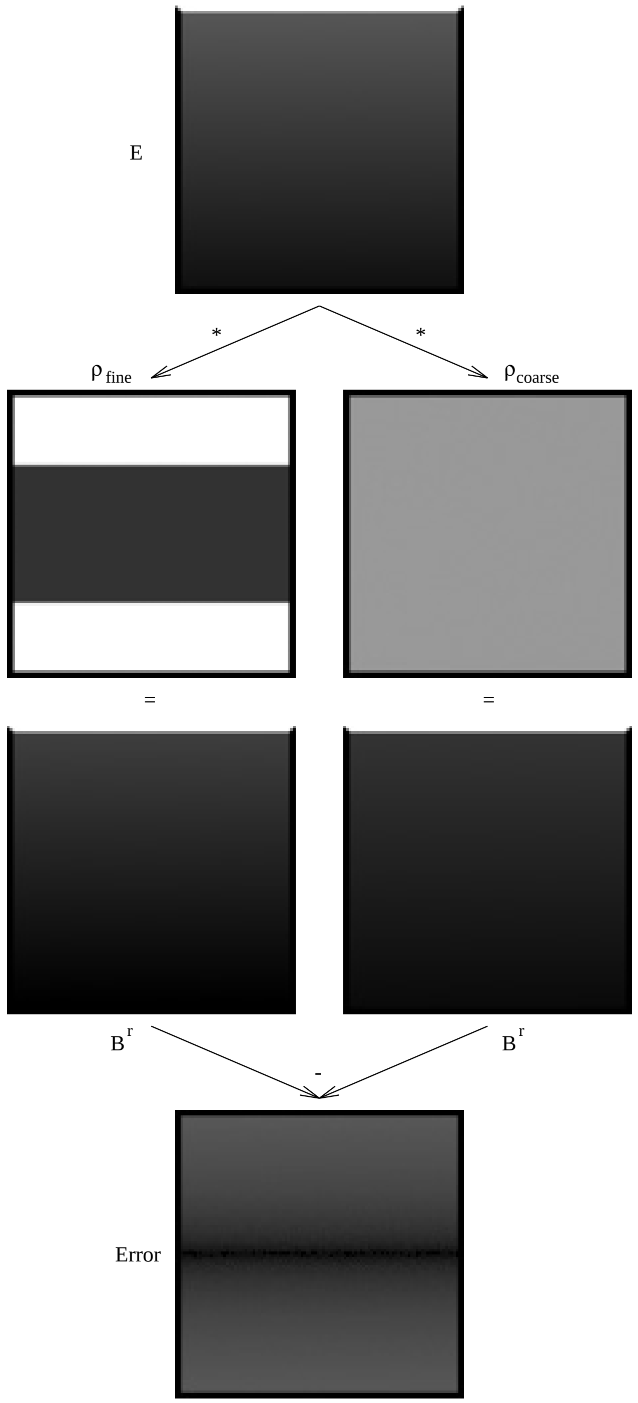

Figure 4: *Two possible algorithms for applying $\mathcal{S}$ to the irradiance E for an example which exhibits a large difference. The bottom shows error magnitude (white, high error).*

Two observations suggest methods for producing such an algorithm:

1. $\mathcal{S}$ may be smooth and if represented in the wavelet basis it may have a sparse representation. If, for example, the texture has regions with fairly little high frequency detail we will be able to adequately represent the texture in those regions with few coefficients. Wavelets afford us one way to optimally take advantage of smoothness if present.
2. The operator $\mathcal{S}$ will be followed by the operator $\mathcal{G}$ in the next iteration. Hierarchical radiosity systems [12, 11] use different levels of detail to compute the irradiance from the radiosity. The decision as to how far to subdivide the surfaces is based on the smoothness of $\mathcal{G}$. As a result there is no need to compute the radiosity at a finer level of detail than needed in the next iteration. This is distinct from the finest level of radiosity used in a final rendering.

These observations can be formulated into two variants of an algorithm which work directly with the given basis:

Algorithm nodal:

1. Push the irradiance to the resolution level of ρ
2. Apply $\mathcal{S}$
3. Reproject
4. Pull to level of outgoing radiosity

We use the term *outgoing radiosity* to denote the level at which radiosity is needed for the next iteration of the solver. This is distinct from the level at which irradiance was computed, since incoming power may need a finer (or coarser!) subdivision than outgoing power.

Algorithm wavelet:

1. Push/pull[2] the irradiance to the radiosity resolution level
2. Apply $\mathcal{S}$
3. Reproject

The term *push* denotes transfer (addition) from a parent to children in a subdivision hierarchy, while *pull* refers to averaging a quantity from children to their parent [12].

These two algorithms are illustrated in Figure 4. At the top is the computed irradiance on the far wall due to a light source at the top. On the left $\mathcal{S}$ is a applied at the finest level of the ρ texture. The resulting radiosity is reprojected and pulled to the level of outgoing radiosity. On the right the application of $\mathcal{S}$ is performed at the level of outgoing radiosity followed by reprojection. The results can be markedly different as shown by the difference between the two at the bottom. Note that it is also possible for radiosity to be computed at a finer level than the reflectance texture. In this case the texture is simply interpolated, or *pushed* in the parlance of wavelet radiosity.

2.5 Error analysis

In order to facilitate the following error analysis we define all the terms in the matrix realizations of the operators $\mathcal{G}$ and $\mathcal{S}$. Replacing $\mathcal{K} = \mathcal{S} \circ \mathcal{G}$ in the traditional form of the linear system we have

$$B_i = B_i^e + \sum_{j=1}^{n} K_{ij} B_j = B_i^e + \sum_{k=1}^{n} S_{ik} \sum_{j=1}^{n} G_{kj} B_j$$

Using $\langle f, g \rangle$ to denote the inner product, we can write $K_{ij} = \langle \mathcal{S} \circ \mathcal{G}(N_j), N_i \rangle$, $G_{kj} = \langle \mathcal{G}(N_j), N_k \rangle$. Finally S_{ik} as given as a function of the coefficients of ρ

$$S_{ik} = \langle \mathcal{S}(N_k), N_i \rangle = \sum_{l=1}^{n} \rho_l \langle N_l N_k, N_i \rangle \tag{3}$$

We now analyze the error introduced by the approximate algorithm. In what follows let $\mathcal{P}$ denote the projection operator onto the level of the outgoing radiosity, and let $\rho_d(x) = (\rho - \mathcal{P}(\rho))(x)$ denote the difference between ρ and its projection at a particular level of detail. This corresponds to the finer levels of detail below the given level. We may then write

$$\mathcal{P} \circ \mathcal{S}(E) = \mathcal{P}(\mathcal{P}(\rho)E) + \mathcal{P}(\rho_d E)$$

[2]We say push/pull since in some regions of the reflectance texture one may need to pull while in other regions one may need to push.

```
Gather( QuadNode p )
  p.E = 0
  ForAllElements( q, p.Interaction )
    p.E += Interaction[i].K * q.B
  Gather( p.sw )
  Gather( p.se )
  Gather( p.nw )
  Gather( p.ne )

PushPull( QuadNode p )
  if( !p.Leaf() )
    p.children.E += WaveletTransformDown( p.E )
    PushPull( p.sw )
    PushPull( p.se )
    PushPull( p.nw )
    PushPull( p.ne )
    p.B = WaveletTransformUp( p.children.B )
  else
    p.B = p.Be + ApplyOpS( p.rho, p.E )
```

Figure 5: *Pseudo code for* `Gather` *and* `PushPull`. *Note that* `B`, `E`, `rho`, *and* `Be` *are two dimensional arrays of coefficients and the operations on them are matrix operations. The function* `ApplyOpS` *is given by Equation 4.*

Examining the right hand side we see two terms. The first, $\mathcal{P}(\mathcal{P}(\rho)E)$, arises from multiplying the approximation of ρ at the chosen subdivision level with the irradiance computed at that level, followed by a reprojection. The second term, $\mathcal{P}(\rho_d E)$, describes the effect of multiplying the finer detail with the irradiance and again reprojecting. Both are potential sources of error. If we only compute $\mathcal{S}$ at the level of the outgoing radiosity we are completely dropping the second term, $\mathcal{P}(\rho_d E)$ (Figure 4 was designed to make this term large). Using the fact that the magnitude of a linearly transformed vector is bounded above by the product of the magnitudes of the vector and the linear operator, $\|Ax\| \leq \|A\| \, \|x\|$, we may write

$$\|\mathcal{P}(\rho_d E)\| \leq \|\mathcal{P}\| \, \|\rho_d\| \, \|E\|$$

For orthogonal projections $\mathcal{P}$ we have $\|\mathcal{P}\| = 1$ and consequently the error in this term can be bounded by the product of the irradiance magnitude and the magnitude of the ignored texture detail. Recall that ρ_d was the detail lost when using ρ as projected at a given level ($\mathcal{P}(\rho)$) compared to the actual ρ.

In the standard wavelet radiosity algorithm an *oracle* function is used to decide whether a given interaction needs subdivision to meet the error criterion [11] with regard to $\mathcal{G}$. The presence of reflectance textures requires an additional oracle to decide at what level of detail $\mathcal{S}$ can be executed. This oracle can be based on a test whether $\|\rho_d\| \, \|E\|$ is less than some error threshold. This is very similar to the BF refinement [12] in ordinary hierarchical radiosity. In BF refinement an interaction between two elements ($\mathcal{G}$) is refined if the product of the magnitudes of radiosity and the form factor error estimate is above the error threshold. This BF refinement also insures the proper handling of emissive textures.

The other source of error arises from the fact that some information is lost in the reprojection step after $\mathcal{P}(\rho)$ and E have been multiplied. This error however, is already implicitly taken into account by the oracle which governs the expansion of $\mathcal{G}$. This oracle decided that the radiosity at the receiving element is only needed to the level of detail at which we computed E.

3 Implementation

We have added reflectance and emittance textures to a wavelet radiosity system based on [11, 16] and [17]. Pseudo code for the two functions which changed from previous hierarchical radiosity systems, `Gather` and `PushPull` is given in Figure 5.

In the paragraphs below we give some more details regarding the following three parts of the overall system

1. functions to build wavelet representations of the texture (image) input files
2. extending the push/pull algorithm to include non-constant emissions and a new function `ApplyOpS`
3. implementation of RenderMan shaders [19] for final rendering output.

3.1 Wavelet encoding of textures

For the encoding of texture maps we used the same basis functions as were used in the rest of the system. These are the multiwavelets introduced by Alpert [1], which are based on Legendre polynomials up to some order (we used order 0 through 3). The mathematics do not require us to use the same wavelet basis for the textures as for the other parts of the system, but it is a natural choice with the least modifications to the system overall.

After reading in a texture, which is currently limited to have a power-of-2 u and v extent, we pyramid transform it using the low pass and high pass filter sequences associated with the multiwavelets [16]. This results in averaged representations of the original texture from the finest (input) level all the way to the overall average for use during the solution process. Of the detail information (high pass bands) only a running total of the energy is maintained for use by the oracle.

To start the transform we treat each pixel value as the coefficient of a constant basis function whose domain is the pixel's area. These constant functions are then projected, i.e. taking the inner product with the multi-wavelets, onto the finest level of the hierarchy. This was chosen for simplicity, but other choices are possible. For example, one might use a cubic spline interpolant of the texture values at the finest level and evaluate the integrals of this function with the multi-wavelet bases at the finest level, before beginning the pyramid transform. The rest of the hierarchy is created by running the `PyramidUp` algorithm described in [11, 16].

At this point we have a full quadtree representation of the texture. For space reasons it may be desirable to prune subtrees in regions which do not have enough spatial detail to warrant finer level descriptions. This is akin to a lossy compression scheme applied to the original texture. During actual production use of our system we have seen a significant savings from this pruning.

3.2 Extending push/pull

The modification to the push/pull algorithm only occurs when it reaches the leaves of the hierarchy. After gathering irradiance, which is the $\mathcal{G}$ step, we need to push the irradiance to the leaves of the subdivision as induced by the $\mathcal{G}$ oracle. Ordinarily we would now multiply the coefficients of the basis functions with the scalar ρ before executing the pull to get ready for the next iteration. Instead we apply $\mathcal{S}$ as given by Equation 3 at this level or an even finer level, if required by the original ρ map and the modified oracle. In practice most applications of $\mathcal{S}$ happen at the irradiance level without any need to go to finer levels (Figure 4 shows an extreme case where application of $\mathcal{S}$ must occur at a lower level).

So far we have used a single index on all coefficients to indicate which basis function they belong to. In so doing we have abstracted the fact that all functions are defined with respect to surfaces, i.e. with 2 parameters. We will be explicit about the 2 parameter nature of our basis functions for a moment to give the exact expression for the application of $\mathcal{S}$. Let $\{N_i(u)\}_{i=1,\ldots,M}$

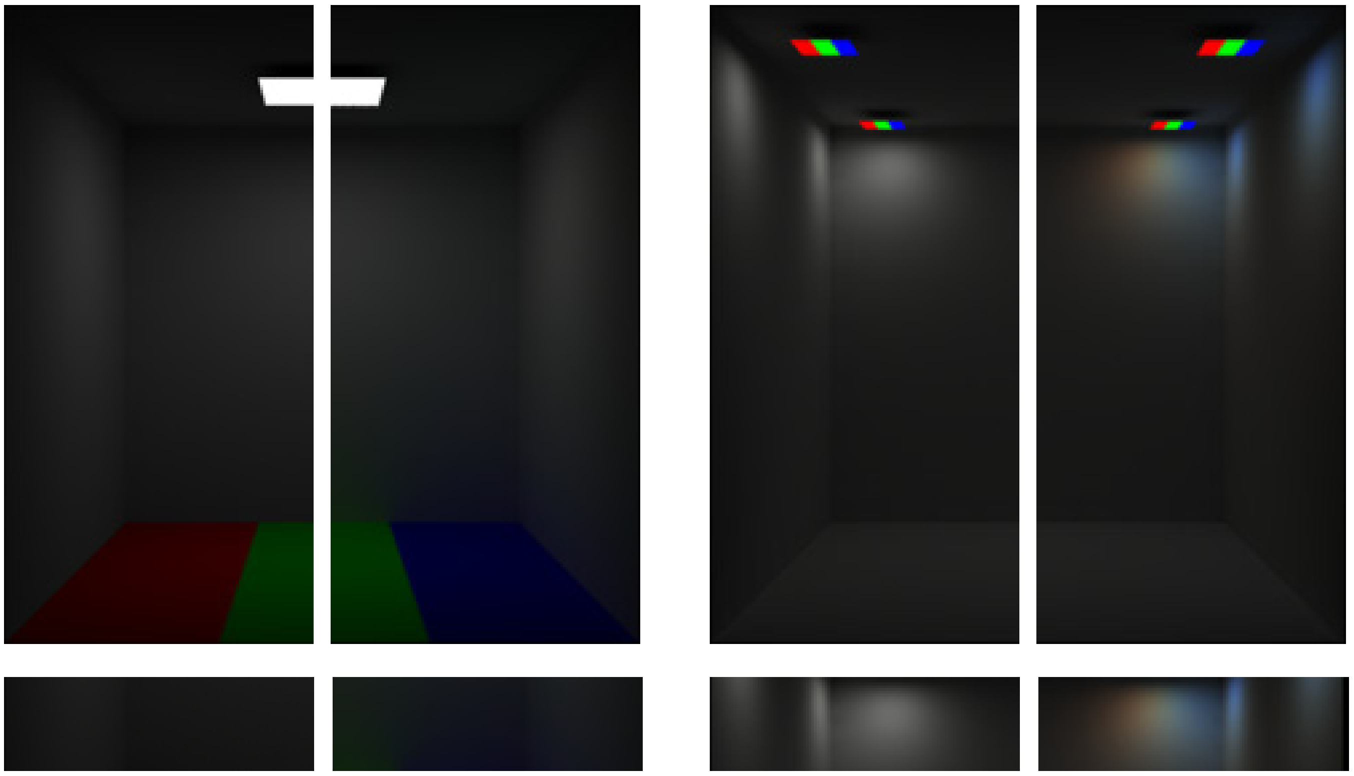

Figure 6*: These pictures illustrate the difference in radiosity solutions produced by using true texture maps and approximating them with average values. The left column shows the effect of a reflectance texture with averaging (left half) and without averaging (right half). Similarly the right column shows the difference between averaging and correct solution for emittances. These images were computed with cubic basis functions (M_4 multi-wavelets).*

be the multi-wavelets of order M. For a moment the index numbering is limited to a *single* element. Surface basis functions are given by $\{N_i(u)N_j(v)\}_{i,j=1,\ldots,M}$ and consequently we use double indices on B, E, and ρ. For example, if we use cubic basis functions with $M = 4$ all the coefficients can be thought of as 4×4 matrices. We still need the reprojection coefficients $R_{ikm} = \int du\ N_k(u)N_m(u)N_i(u)$. With this notation we have

$$B^r_{ij} = \sum_{kl=1}^{M} \sum_{mn=1}^{M} R_{ikm}R_{jln}\rho_{kl}E_{mn} \quad i,j = 1,\ldots,M \tag{4}$$

Note that R_{ikm} can be precomputed once for a given choice of basis. Taking the example of $M = 4$ again we see that the 4×4 coefficients B^r_{ij} are a sum over all combinations of the 4×4 irradiance coefficients with the 4×4 reflectance coefficients. The weights in this sum are given by the $4 \times 4 \times 4$ reprojection coefficients. Only emittance is left to be added in before executing the pull and starting the next iteration

$$B_{ij} = B^e_{ij} + B^r_{ij}.$$

3.3 Final output

As the renderer for final output we use RenderMan [19]. This requires the creation of a RenderMan Interface Bytestream (rib) file containing descriptions of the geometry of each leaf element generated during subdivision, as well as the computed coefficients for the basis functions associated with the element. We have written RenderMan shaders which accept these coefficients as parameters and evaluate the radiosity function at a given point (u, v) in parameter space of the surface element. In the case of multi-wavelets this is achieved by implementing the Legendre basis functions in the shading language. Note that we do no post processing, such as, for example, vertex averaging, as is done in many traditional radiosity systems. The rendered images show the functions exactly as computed by the radiosity system, because the color of each sample is interpolated using the appropriate basis function during the rendering process.

For surfaces which do not have texture maps associated with them, we write out the coefficients of the *radiosity*. For surfaces with associated texture maps (emittance and/or reflectance) we write out the coefficients of the *irradiance* together with texture coordinates and references to the original texture files. The shaders take this description and for a given (u, v) evaluate the irradiance, multiply with the reflectance texture value, and add the emittance texture value.

Although our final renderings were done with RenderMan, the texture mapping and basis function interpolation scheme used could be easily added to any rendering system.

4 Results

Figure 6 shows the difference between using averaged textures and true textures during the radiosity computation:

- The left column demonstrates the case of a reflectance texture mapped onto the floor. The texture map has equal amounts of red, green and blue, and these were chosen so the average texture value is gray and colorless. The left half shows the result of using the average texture color and adding the texture as a post process; the right half shows the result achieved using the true texture map. Below each half images is a cutout of the region just above the floor to show the difference between the two computations more clearly. Note the strong color bleeding near the floor. These colors are absent if the average value is used.

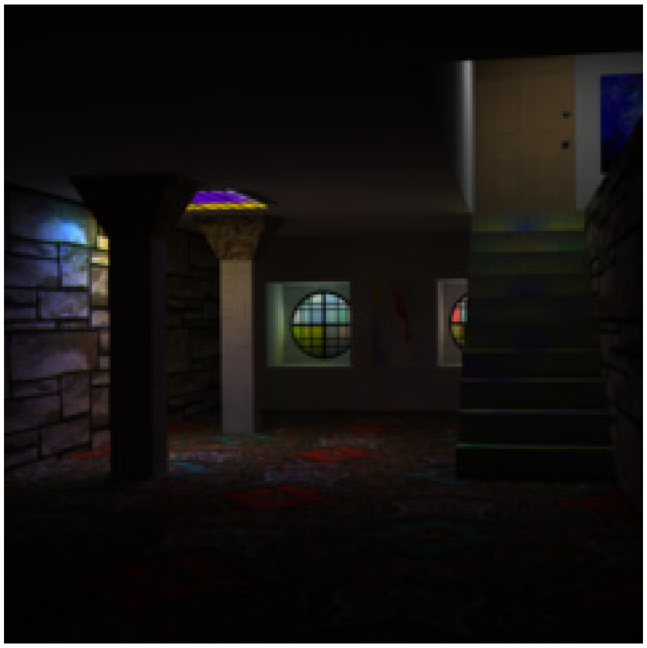

Figure 7: *This scene illustrates the visual complexity added to a radiosity simulation by the inclusion of reflectance and emission textures (62). Total simulation time to convergence 10 minutes.*

- The right column shows the results of a similar experiment for emittances. In this case there are 4 ceiling lights with the same texture map, this time used for emittance instead of reflectance. Once again the left half shows the solution when using only the average color, and the right when using the true colors during the simulation itself. In both cases we also have a cutout of the region just below the ceiling lights where the differences are most pronounced. Again, with averaging no colored light is cast onto the ceiling and walls near the lights.

For these examples we used multi-wavelet M_4 (piecewise cubic) basis functions.

To test the robustness of the system, and to measure the impact of texturing on the running time of the radiosity algorithm, we computed more complicated scenes shown in Figures 1 and 7. Figure 1 was modeled with 23 quadrilaterals and 10 texture maps. The stained-glass window is a single quadrilateral with the emittance and reflectance maps shown in Figure 9. We solved the system using cubic multi-wavelets. A total of 29 seconds was spent on hierarchy creation for all texture maps (each with a resolution of 128^2 pixels). The solution was run to convergence in 10 complete gather sweeps. 1795 elements with 12831 interactions were created. The process size reached 51Mb out of which 26Mb were consumed by texture map data. The execution time was 36 minutes on an SGI Indigo (R4000 50MHz) computer. Over 90% of this time was spent computing quadratures. The increased cost in the application of $\mathcal{S}$ due to the textures is only felt during the iterations (the remaining 10%), which become approximately 10% more expensive. Figure 7 shows another scene. It contained 99 quadrilaterals and 62 texture maps (most had a resolution of 32^2 pixels), which resulted in 9515 elements and 90362 interactions using linear basis functions. Total memory size grew to 54Mb of which 9Mb were used for textures. The solution required 10 minutes and 15 complete gather iterations for convergence on an SGI Indigo (R4000 50MHz) computer.

These performance numbers demonstrate that the resulting system can simulate scenes with many textures efficiently incurring

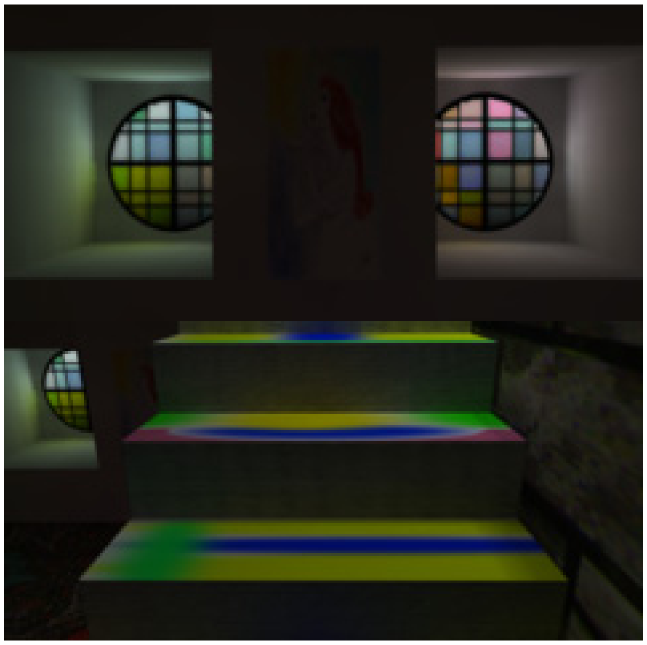

Figure 8: *Two details from Figure 7. On top the recessed colored windows on the far wall of the room illustrate the subtle lighting effects possible. Each window is a single polygon. On the bottom the effect of reflectance maps on the staircase. The top of each step has a reflectance map. Notice the patterns in the induced color bleeding.*

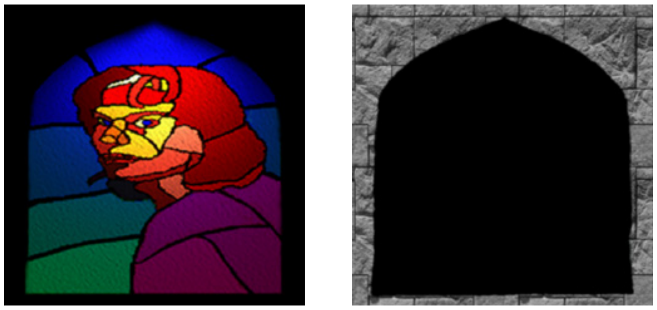

Figure 9: *The stained glass window was created with the emissive texture on the left and the reflective texture (matte) on the right.*

only a small time penalty. Extending the existing wavelet radiosity system was straightforward since only a few functions needed minor modifications.

Figure 8 shows close-up views of some of the areas that were most influenced by the texture maps. The color changes around the colored glass windows and the reflection off the staircase steps are effects that would have been difficult to model without the use of texture maps. Similarly in Figure 1 the use of texture maps allowed the system to cluster many floor tiles or window panes together when determining interactions by using the standard hierarchical error criteria [12, 11]. The fact that high visual complexity was achieved with only a few quadrilaterals attests to the power of the texture mapping paradigm in the radiosity context.

5 Summary and conclusion

We have presented a technique which extends hierarchical radiosity algorithms to include reflectance and emittance textures during the solution process. This extension was facilitated by a careful examination of the radiosity integral equation. We separate the computation of irradiance, a global process, from the computation of reflected radiosity, a local process. In so doing we can take advantage of efficiencies in each step individually. These efficiencies are not available if the two steps are merged, as is normally done. The new method has been implemented in a wavelet radiosity system. The resulting images demonstrate the desired lighting effects with little penalty in performance. As a result increasing the visual complexity of radiosity computations with texture maps is now practical.

Although the computational cost of using texture maps is now quite small, the storage and memory costs may still be large. An interesting research question is how to manage the texture data. Since only limited resolutions are ever needed, and since many radiosity runs may be performed using the same input textures, it is reasonable to precompute and store the wavelet representation of the texture. The radiosity program then need only read in the resolution sets that are required. This also suggests the intriguing possibility of applying a lossy wavelet compression to the texture maps to begin with. The compression would be limited so as to not produce any artifacts to the human observer, but could still reduce the cost of reading the textures, and the amount of disk space needed to store them.

The use of texture mapping techniques in computer graphics has a long tradition and many complex effects can be achieved. One such example has already been demonstrated in Figure 1, where two texture maps, one for emittance and another for reflectance are combined with a matte (see Figure 9). As a result the odd shaped stained glass window is created from texture maps and a single quadrilateral. Other uses for texture maps include environment mapping and bump mapping [4]. The algorithm described in this paper may be easily modified to use environment maps; an interesting application of this would be to the modeling of skylighting, which is currently very expensive. A clever algorithm for approximating bump mapping in the context of radiosity is described in [4]; similar approximations could likely be used in a hierarchical radiosity algorithm, although this would require further research.

Some of these techniques may also be useful for the clustering problem. Imagine a surface described by a displacement map over a regular grid. Now we have actual geometric complexity, yet a multi resolution description still appears promising and could be incorporated in a straightforward manner into the texture mapping system. In fact one might imagine replacing entire complex ensembles of geometry with a "picture" during parts of the radiosity solution process. Another use for this system is in the area of inverse problems. Consider taking a photograph of a diffuse surface, i.e. capturing its radiosity, and using this as a texture in a radiosity simulation. This could be used to match real scenes with computed scenes [9]; and more generally might be useful for problems arising in computer vision, such as the computation of the reflectance map.

The technique we applied in our analysis, separating out the global $\mathcal{G}$ part of the computation from the surface local $\mathcal{S}$ part, may also be a fruitful analysis technique for other rendering problems. For example, in volume rendering it may yield new insights to separate the representation of the attenuation operator from the representation of the scattering operator. Another example is the computation of radiance, where one might also gain new efficiencies from separating the incident radiance operator from the scattering operator.

Acknowledgments

We would like to thank Seth Teller and Celeste Fowler for their contributions to the radiosity testbed which greatly facilitated the research reported in this paper. Many thanks also to Don Mitchell, Michael Cohen, and Steven Gortler for stimulating discussions and David Laur for production help. The first author would like to thank Sondra Au for artistic advice and contributions. Thanks to James Shaw for the stained-glass window texture (Figure 1) and SGI for many of the other texture maps used in Figures 1 and 7. Other support came from Apple, Silicon Graphics and the NSF (contract no. CCR 9207966).

References

[1] 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).

[2] Beylkin, G., Coifman, R., and Rokhlin, V. Fast Wavelet Transforms and Numerical Algorithms I. *Communications on Pure and Applied Mathematics 44* (1991), 141–183.

[3] Blinn, J. F., and Newell, M. E. Texture and Reflection in Computer Generated Images. *Communications of the ACM 19*, 10 (October 1976), 542–547.

[4] Chen, H., and Wu, E.-H. An Efficient Radiosity Solution for Bump Texture Generation. *Computer Graphics 24*, 4 (August 1990), 125–134.

[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] Cohen, M. F., Greenberg, D. P., Immel, D. S., and Brock, P. J. An Efficient Radiosity Approach for Realistic Image Synthesis. *IEEE Computer Graphics and Applications 6*, 3 (March 1986), 26–35.

[8] Dorsey, J. O., Sillion, F. X., and Greenberg, D. P. Design and Simulation of Opera Lighting and Projection Effects. *Computer Graphics 25*, 4 (July 1991), 41–50.

[9] Fournier, A., Gunawan, A. S., and Romanzin, C. Common Illumination between Real and Computer Generated Scenes. In *Proceedings of Graphics Interface 93* (1993), pp. 254–261.

[10] 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.

[11] Gortler, S., Schröder, P., Cohen, M., and Hanrahan, P. Wavelet Radiosity. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 221–230.

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

[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] 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.

[16] Schröder, P., Gortler, S. J., Cohen, M. F., and Hanrahan, P. Wavelet Projections For Radiosity. In *Fourth Eurographics Workshop on Rendering* (June 1993), Eurographics, pp. 105–114.

[17] Teller, S., and Hanrahan, P. Global Visibility Algorithms for Illumination Computations. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 239–246.

[18] Troutman, R., and Max, N. Radiosity Algorithms Using Higher-order Finite Elements. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 209–212.

[19] Upstill, S. *The RenderMan Companion*. Addison Wesley, 1992.

[20] Zatz, H. R. Galerkin Radiosity: A Higher-order Solution Method for Global Illumination. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 213–220.