HELSINKI UNIVERSITY OF TECHNOLOGY
Telecommunications Software and Multimedia Laboratory
5.5.2003
Tik-111.500 Seminar on computer graphics

Precomputed Radiance Transfer

Teemu Mäki-Patola

48473E

Precomputed Radiance Transfer

Teemu Mäki-Patola
HUT, Telecommunications Software and Multimedia Laboratory

tmakipat@tml.hut.fi

Abstract

This paper describes a method for rendering interreflections, self-induced shadows and caustics in real-time in a general low-frequency lighting environment. The method handles environment and area lighting and glossy as well as diffuse objects. The lighting environment is projected onto a low-frequency spherical harmonic basis. Before the rendering a transfer vector (diffuse case) or a transfer matrix table is created for the object through preprocessing. This table takes into account the object's interreflections and self occlusions and is used with the spherical harmonic lighting components to evaluate the total exiting radiance of each visible vertex of the object.

1 INTRODUCTION

This work reviews the Precomputed Radiance Transfer technique first proposed by Peter-Pike Sloan, Jan Kautz and John Snyder [*10*] in SIGGRAPH 2002 conference. Most of the presented information comes from their paper and presentation slides [*11*].

1.1 Background

Until the recent years real-time 3D rendering has utilized mostly point light sources and/or directional lights. The rendering has been using mostly the Phong lighting model. In this model each vertex of each object sums up light from the background ambient light and all light sources in the scene. The light from each source affects the lighting of the vertex in relationship to the material properties of the object that the vertex belongs to. Depending on the material, diffuse light usually falls on the object as a first order cosine function of the surface angle relative to the light direction. Specular components use higher order cosine functions to give the object a specular shine such as reflecting marble or wet surfaces embody [*6*].

Real-time 3D graphics, such as computer games, already implement shadows. Algorithms for these interactive shadows have existed for some time, but they implement mainly sharp shadows with point light or directional light sources. Shadow volume techniques (started by Frank Crow [*3*]) and shadow map techniques (first presented by Williams [*12*]) are the most commonly used in making real-time shadows.

An example of a shadow volume technique is to extract shadow polygons from an object's each edge that connects polygon(s) that are towards a light source and polygon(s) that are facing away from the same light source. Stencil buffer is then used to mask the shadow polygons only to object surfaces so that the shadow is not visible in the air in-between objects. Shadow maps are based on the idea that by rendering the scene from the perspective of each light source shows only the surfaces that are lit by that light. The surfaces that are not visible then are in shadow relative to that light source. The light source visibility information (shadow areas) is coded in stencil buffer by depth test and used to render the final camera view.

These real-time shadow techniques are limited to point and directional light-sources. They can be combined with multipass rendering to generate soft shadows. However the required number of rendering passes gets impractically large as the light sources get bigger. This is because the large lights are modeled as consisting of many point lights.

Limited soft shadows and inter-reflections have recently been implemented in real-time or close to real-time performance [*1*, *7*, *8*]. However, these methods limit at least the dimensions or the amount of light sources to function in real-time. The interreflection methods are limited to point lights. Earlier work on the field has extended lighting from point lights to general environment maps [*2*] but these methods cannot model soft shadows or interreflections either.

Shadow volumes have been extended to process single spherical light sources at interactive rates [*1*]. Some earlier works on the field have dealt with unchanging lighting environments by, for instance, moving a diffuse object through a precomputed lighting environment lighting the object dynamically [*5*]. These methods cannot handle object self-shadowing (the object occluding parts of itself). A method called surface light fields can model also the self-shadows and deal with complex lighting environments, but the lighting environments cannot be dynamically changed.

Sophisticated and commonly used nonreal-time rendering technique, ray tracing, is able to generate very realistic images. Little by little, ray tracing has been expanded to model most aspects of light behavior. Ray tracing doesn't create the image by rasterizing polygons between corner vertices. It shoots rays from the camera to the scene through each image pixel and traces it to object surfaces, continues from there to light sources and possibly branching to a new direction if the object surface is reflecting [*6*]. All these rays contribute to the color of that one image pixel according to where they hit in the scene.

An important step in making ray tracing appear realistic has been the addition of radiosity calculations. Radiosity basically implements the conservation of light energy in a form of considering also the light that gets reflected from surfaces upon other surfaces. Radiosity models only diffuse surface reflections, not glossy materials.

1.2 Motivation

Real-time rendering has not been able to handle general object self-shadowing and interreflections with other than point light sources, directional light sources or specific individual light sources such as a sphere with some physical dimension. Point light

source lighting does not acquire a realistic appearance in most situations (see figure 1). The object's shape is more clearly presented with an area light lighting.

Radiosity has often been precalculated into light maps in current 3D computer games. The calculation takes a long time for an environment and means that the lighting will be static during the game play. Some games have implemented very moody real-time lighting effects. However, they all use point lights, directional lights or omni lights which all have no light source dimension. The shadows are then hard, not soft with a penumbra.

Lighting from area lights or environmental lights always produces soft shadows because while other parts of the light source are occluded by an object some other parts may not be. To achieve realistic image synthesis soft shadows and surface interreflections are required.

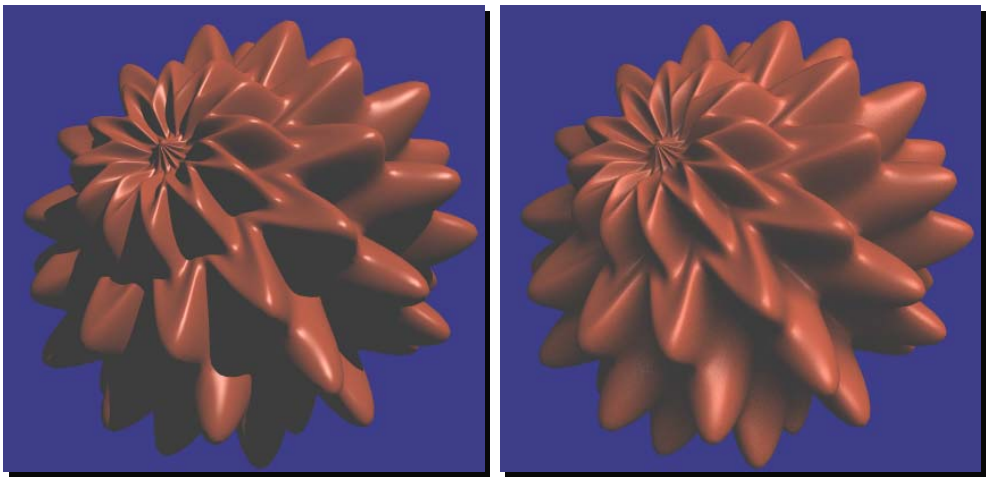


*Figure 1. An object lit by a point light and by an area light both casting shadows. As can be seen the dimensional light source brings about the object's shape more clearly. Source [*11*].*

Evaluating real-time global lighting requires integration over a hemisphere of lighting directions at each rendered point [*6*]. It needs to account for light bouncing and occlusion effects. For instance if one side of a cube viewed from inside gets much light projected onto it and other sides do not directly get any light it does not look realistic if the other sides do not receive some of this light through diffuse reflectance.

Shadowing from occlusions need to be checked as well. This basically requires checking if there is something on the way from the surface point to the light source. The lighting should also model complex and spatially varying bi-directional reflectance distribution function (BRDF) of realistic materials utilizing glossy and anisotropic reflections.

1.3 Basic idea of the presented solution

This work describes a method for rendering interreflections, self-shadowing and caustics for diffuse and glossy surfaces in real-time in a general low-frequency lighting

environment, first presented by Peter-Pike Sloan, Jan Kautz and John Snyder [*10*, *11*] in SIGGRAPH 2002 conference. The method expresses the lighting environment in a spherical harmonic function basis. This is basically a 2D sphere surface analogy for approximating functions with Fourier series. The lighting environment is projected onto a low-frequency spherical harmonic basis containing a rather small amount of basis functions (similar to Fourier basis functions but defined on the surface of a sphere). The term low-frequency lighting environment follows from approximating the lighting with only a few spherical harmonic functions having a long wavelength.

The method handles area lighting and environmental lighting and is able to produce soft shadows as well as object interreflections. The difference between area lighting with and without object self-shadowing can be seen from figure 2. The objects that are to be illuminated in the lighting environment consist of surface patches. The surface should be homogenously divided into almost equal size polygons. This means that even large flat parts that could geometrically be expressed with just one polygon need to be subdivided into many polygons. The reason is that the lighting is calculated only at the surface patch vertices and having too big polygons creates errors to the lighting as each polygon linearly interpolates the light falling on its corner vertices over the whole polygon.

The objects are preprocessed to create a set of radiance transfer functions over the object's surface. The functions represent transfer of arbitrary, low frequency incident lighting into exiting radiance, which includes effects from shadows and interreflections from the object onto itself. The transfer information is stored into each vertex of an object. At runtime the transfer functions are applied to the incident lighting of the scene. An object maybe rigidly rotated in respect to the light but not deformed in shape.

The main idea is to represent how an object scatters the light that falls on it (incident lighting) onto itself or on the neighboring space. Both the lighting and the transfer functions are expressed using a spherical harmonics basis. The whole shading integral becomes reduced into a dot product of vectors for diffuse surface objects and to a matrix multiplication and dot product for glossy surface objects. This real-time evaluation can be done efficiently on graphics hardware.

Dobashi, Kaneda, Nakatani and Yamashita [*4*] have applied spherical harmonics to room lighting models already in 1995. However, their version had several constraints to keep it real-time. The user was able to view the environment from different angles interactively but the lighting could not be much altered in real-time. The scenes were also diffuse. It was not even considered how glossy could be modeled.

This work will now proceed by introducing spherical harmonic functions and their basic properties. Then the whole rendering process of the precomputed radiance transfer is explained step by step. Finally ways to expand the used approach to achieve additional real-time rendering effects are presented.

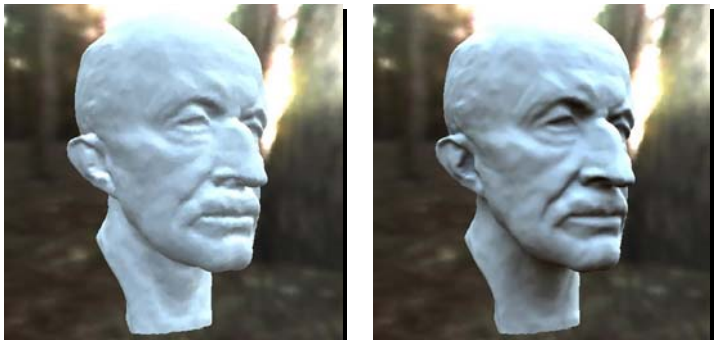


Figure 2. An object under area lighting, without self-shadowing and with self-shadowing. Source [11].

2 SPHERICAL HARMONICS

Spherical harmonics define an orthonormal function basis over a sphere. The basis is analogous to Fourier functions in 1D and 2D.

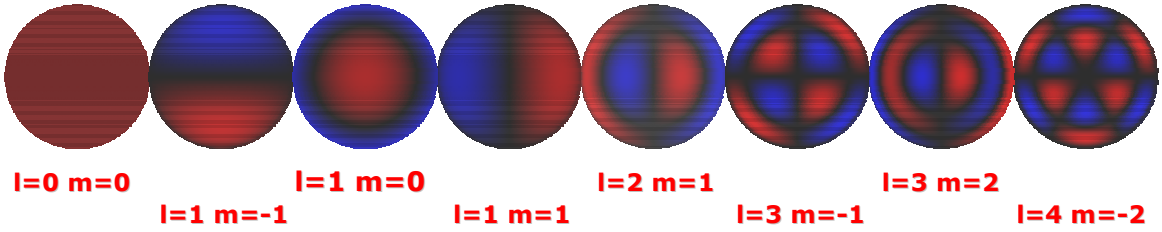


Figure 3. Visualization of low-frequency spherical harmonic basis functions. The blue areas represent negative values and the red areas positive values. Source [11].

2.1 Introduction

Similarly than Fourier functions are used for presenting one- or two-dimensional functions as a weighted sum of Fourier basis functions, spherical harmonics are used for presenting functions over a spherical surface as a weighted sum of spherical harmonic basis functions. The motivation for representing a function as spherical harmonic sum is that spherical harmonic basis functions have several useful mathematical properties, which simplify the processing of the original function that the original function does not have.

Figure 3 shows a visualization of a few spherical harmonic basis functions. A basis function is defined with two index numbers L and M. L is called the band index. For each L, M moves between –L and L and is an integer. Low values of L represent low-frequency basis functions over a sphere.

From the orthonormality condition of the basis functions follows that a scalar function f, defined on a sphere, can be presented as spherical harmonic function coefficients f_l^m, each presenting the influence/weight of one basis function. From this point on we will

mark the coefficients with just a one-index variable l_i. The reason for this is simplicity. The described method also uses all M values (-L to L) for each band. Thus, the number of coefficients is second power of the band index (1, 3, 5, 9, 16, 25, etc.)

The coefficients can be evaluated by projecting f on each of the basis functions with an integral:

$$l_i = \int f(s) B_i(s) ds \qquad \text{(i)}$$

The above equation defines the weight coefficients for each basis function. The original function f (the lighting environment) can then be reconstructed from the coefficients by summing the basis functions multiplying each of them by the associated weight coefficient:

$$\tilde{f}(s) = \sum l_i B_i(s) \qquad \text{(ii)}$$

As with Fourier sum the reconstruction is always an approximation when there are less than infinite amount of coefficients. However, already a small amount of basis functions are able to approximate low-frequency functions quite well.

2.2 General properties

As already mentioned, the basis functions form an orthonormal basis. They can be processed and evaluated individually of each other. A lighting function can be simply projected to a spherical harmonic basis and reconstructed using the projection coefficients. From the orthonormality it also follows that for any a and b defined over a surface of a sphere s their projections satisfy the following equation:

$$\int \tilde{a}(s) \tilde{b}(s) ds = \sum a_i b_i \qquad \text{(iii)}$$

A very important property of the spherical harmonics from the viewpoint of the presented real-time lighting approach is that spherical harmonics are rotationally invariant. The reconstruction function $\tilde{f}(s)$ that is rotated can be projected into the spherical harmonics basis using a linear transformation of only the projection coefficients of f(s). This property means also that spherical harmonic projection causes no aliasing artifacts when samples from the lighting function f are collected at a rotated set of sample points. Evaluating a rotation is thus simple and doesn't require much work. Convolution of projected functions can also be easily calculated using just the projection coefficients. In the approach, when either the object or the lighting environment is rotated, in both cases it is the lighting environment that is rotated to the object's coordinate system.

In their paper Sloan, Kautz and Snyder [*10*] use only a small number of basis functions. The low-frequency lighting restriction comes from this as only functions consisting of long wavelengths can be expressed. The reason for using only a small number of basis functions is calculation complexity, as real-time rates need to be achieved.

The spherical harmonic basis is linear as the modeled lighting is constructed as a weighted sum of the basis functions. The individual basis functions can be treated independently.

3 THE RENDERING PROCESS

This chapter will present step by step the whole process of Precomputed Radiance Transfer. The preprocessing is covered first and then the construction of the frame image itself. Both cases, diffuse and glossy, are presented.

3.1 Object preprocessing

A transfer function maps the incoming radiance to outgoing radiance at a surface vertex. A transfer function of a convex object lit from far away environment lighting simply integrates the light at each point weighting the integral with a cosine function of the light direction relative to the surface normal at the point. The light that falls on the surface from the normal direction affects more than light coming from "side". A more complex integral for a transfer function captures how a concave object shadows/occludes itself. In this case the light being integrated is multiplied by an additional transport factor representing visibility in each direction from a point on the object. Yet more complex evaluation can consider the interreflections from the object surface onto itself. This is generally done by radiosity calculations but in the approach in this paper it is done by caching light into vertices through evaluation passes that consider rays that bounce from the object surface before ending at the evaluated vertex.

The occlusions, light integrating and interreflections cannot be calculated at interactive rates in real-time. However, as the usage of memory is not usually as severe restriction as the calculation power, the approach is to calculate transfer functions for each vertex as a preprocess that simulates the light transport on the object and store them at the object vertices.

The incident radiance and the transfer functions are both expressed in the spherical harmonics basis. As the spherical harmonic basis is linear the original light integral can be expressed by a dot product of the coefficient vectors of the incident radiance and the transfer function vector (diffuse case) or as a transfer function matrix operation (glossy case).

Low frequency lighting consists of only 9 to 25 coefficients. Current graphics cards can be used to calculate the light transforming dot product or matrix multiplication in a single rendering pass.

A convenient result of modeling the lighting as spherical harmonics is that the computational complexity of the lighting problem remains unchanged no matter how many or how big light sources the scene contains.

3.1.1 Diffuse objects

To compute the reflected radiance from a surface vertex of the object in a direction **v** the following integral needs to be evaluated:

$$R(\vec{v}) = \int L(\vec{s})V(\vec{s})f(\vec{s},\vec{v})H_N(s)d\vec{s} \qquad (1)$$

The term L(s) represent the lighting in the scene, the inside s represent the hemisphere over which we are integrating (a half space away from the surface). V(s) is a binary visibility function. Its value is 1 if the part of the lighting environment is visible and 0 if not. V(s) contains all the self-occlusion (shadowing) information of the object.

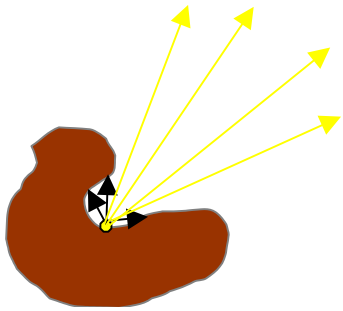


Figure 4. The visibility function is 0 for the directions that are occluded by the object and 1 for directions that reach the light source from the surface point.

F(s,v) represents the bi-directional reflectance distribution function (BRDF) of the surface material. As mentioned earlier the surface reflectance might vary as a function of the reflection direction as well as of the location on the surface. Different BRDFs model this behavior. $H_N(s)$ is the dot product of the surface normal at the vertex with the direction vector s. If $H_N(s)$ is smaller than 0 it is truncated to zero.

When the lighting environment is expressed as coefficients in some basis this can be inserted in the equation in the place of the L(s) term (lighting) resulting in:

$$R(\vec{v}) \cong \int (\sum l_i B_i(\vec{s}))V(\vec{s})f(\vec{s},\vec{v})H_N(s)d\vec{s} \qquad (2)$$

Now the lighting is a reconstruction sum of weighted basis functions $B(s)_i$. By rearranging factors and using the linearity property of light transport we get a sum of the lighting coefficients multiplied with an integral that depends only of the basis functions and of the BRDF but not of the lights anymore.

$$R(\vec{v}) \cong \sum l_i \int B_i(\vec{s})V(\vec{s})f(\vec{s},\vec{v})H_N(s)d\vec{s} \qquad (3)$$

If the BRDF is now diffuse, we know also the f(s,v) already at the preprocessing time. Diffuse BRDF does not depend of the view direction. Thus the whole integral can be evaluated and replaced by a scalar t_i. Now the exiting radiance can be expressed as:

$$R(\vec{v}) \cong \sum l_i t_i \qquad (4)$$

This final form depends only of the incident lighting coefficients and the object properties at the vertex. The object can be relit with only a dot product of lighting coefficients and transfer vector coefficients per vertex.

As a preprocess, the integral in equation (3) is computed for each object vertex and each basis function per vertex. This results in equally many coefficients as there are basis functions to present the lighting environment. The coefficients are then stored as the transfer vector of a vertex.

To evaluate the integral in equation (3) a set of 10 to 30 thousand quasi-random direction vectors $\{s_d\}$ is first created. The visibility mask for the vertex is determined for each of these direction vectors. The integral then becomes a large sum over all direction vectors.

To include also the object inter-reflections, the light that the objects own surfaces distribute on the other surfaces of the object, something more needs to be done. The final reflected radiance in shadowed and interreflected diffuse transfer is

$$R_{ir}(\vec{v}) = R(\vec{v}) + \int \overline{L}_p(\vec{s}) H_N(\vec{s})(1 - V_P(\vec{s}))d\vec{s} \qquad (5)$$

Where $R(\vec{v})$ is the whole integral of the equation (1). The added integral represents the light from the object interreflections. $\overline{L}_P(\vec{s})$ represents the radiance of the object itself in the direction of vector s. However, if the incident lighting does not come from infinitely far the local lighting varies over the object and $\overline{L}_P(\vec{s})$ cannot be known from the incident radiance at a vertex because it depends on the exit radiance of other surface points that depend of local lighting changes. However, if the lighting doesn't vary much over the object (the lighting is in infinity) then $\overline{L}_P(\vec{s})$ is approximated well as if the object was illuminated by $L(\vec{s})$ at every point. Thus, to be able to make preprocessed interreflections a spatially invariant incident lighting over the whole object needs to be assumed. This means that interreflections cannot be calculated for local lighting. The shadowed lighting without interreflections can, as it does not require evaluation of the exiting radiance at other object points.

The self-occlusion component $(1 - V_P(\vec{s}))$ now filters out all other directions than the ones that are occluded in the integral. There can be interreflections only from the directions where the ray along the direction hits the object surface before exiting the object.

The whole reflected radiance of the equation (5) depends linearly on the incident lighting and can be factored into the coefficient vector similarly than the shadowing previously. With diffuse objects where the BRDF is constant and does not depend of the view direction the sum of the interreflection integral and the integral from equation (1) can both be included in the transfer vector coefficients t_i of equation (4). The coefficients are only different than those of equation (4) that consider only shadowing.

Calculating the object interreflections with itself in the preprocessing step is done followingly:

First the shadows are processed as before by calculating the visibility mask $V(\vec{s})$ (by ray tracing) for all 10 to 30 thousand quasi-random directions in the direction set. Then

on subsequent evaluation passes interreflections are added by evaluating only the paths from the light source that bounce at least once from the object before ending up in the vertex being processed. This means that the integral only considers the shadowed directions. At each evaluation pass radiance is gathered at each such vertex. This is again expressed just in the coefficients of the transfer vector separately for each basis function.

3.1.2 Glossy objects

As the radiance of a vertex on glossy surfaces depends also of the view direction (instead of being only a function of s) the transfer function at each vertex is presented as a matrix, instead of a vector. For viewing a glossy surface from different directions the diffuse formulation cannot express the required information anymore. Now the lighting calculation has to depend of the viewing direction at real-time and the $f(\vec{s},\vec{v})$ term of equation (3) cannot be evaluated during the preprocessing.

The function of the transfer matrix is to map the incident radiance into transferred incident radiance at each vertex on a surface. The transferred incident radiance accounts for occlusions (self-shadows) and inter-reflections as in the diffuse case but does not include the bi-directional reflectance distribution function (BRDF). The BRDF is needed because the final lightness of a glossy, reflecting surface vertex depends of the view direction.

If one looks at a mirror from different angles, what he sees changes depending on the view direction although the lighting where the mirror is does not chance. As the incident radiance represents the scene lighting environment without the presence of the object, the transferred incident radiance accounts for the shadowing and scattering effects due to the presence of the object.

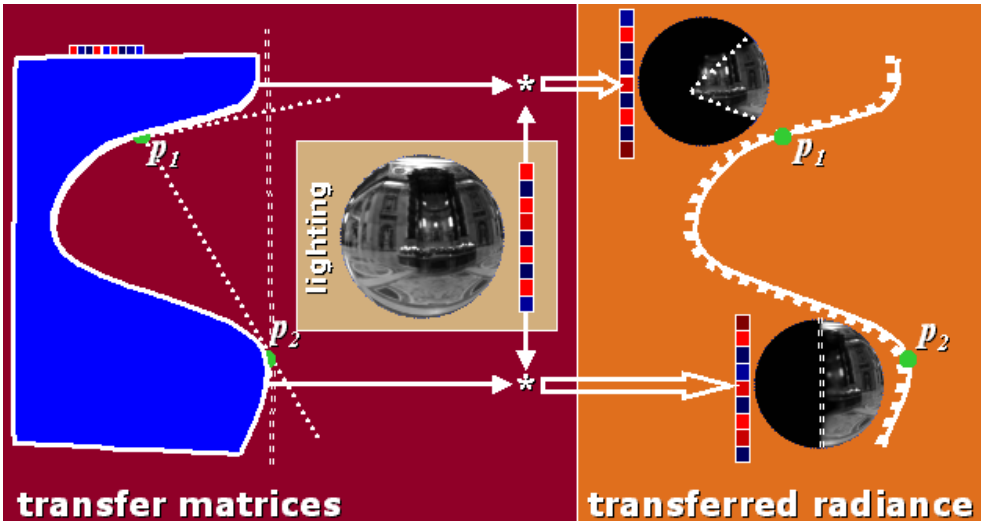


*Figure 5. In glossy preprocessing matrixes that transfer the incident radiance into transferred incident radiance are stored at each vertex. Source [*11*].*

In the figure, vertex p1 is in a concavity. This means that when the lighting incident environment is operated by the vertex's transfer matrix the matrix shadows the lighting

environment more at p1 than at p2. Each point sees at most a full hemisphere of the lighting environment (see p2 at the figure) and filters the rest away. The filtering is done in a way that the transferred incident radiance (after the matrix operation) at each vertex expresses only the remaining lighting as a coefficient vector. Similarly than the whole scene lighting is expressed as the weight of the basis functions, the transferred incident radiance expresses only the part of the scene lighting that the point sees in the spherical harmonic basis.

The diffuse transfer function depended only on the surface normal and not the view direction. The glossy transfer depends on a view depended reflection direction R rather than a fixed surface normal N. The $H_N(s)$ in the equation (1) becomes a more complicated function G(s,R,r), where r is a scalar that defines the "glossiness" of the specular response. Because the transfer matrix does not contain the BRDF it means that the BRDF of the surface can still be changed during run-time without doing the preprocessing again. However it also means that much more calculations are needed on run-time. The exiting radiance for a certain view direction v is now computed by integrating the transferred incident radiance against the used BRDF. In the paper Sloan, Kautz and Snyder used Phong style BRDF as it is circularly symmetric about R and simplifies the evaluation.

In the case of glossy surfaces it is reasonable to express the lighting with more basis functions compared to the diffuse case. With few basis functions the lighting environment is very blurred. To have more interesting reflections of it, it should be represented with more detail. Sloan, Kautz and Snyder [*10*] used 25 coefficients for acquiring satisfactory glossy effects.

The actual preprocessing requires calculating the coefficients for the transfer matrix at each vertex. The transfer matrix will operate the vector consisting of spherical harmonic coefficients. Thus, if we have a basis of 25 spherical harmonic functions the matrix is rather large 25*25 elements. The shadowed radiance equation for glossy surfaces is

$$R_S(L_P, R, r) = \int L_P(s)G(s, R, v)V_P(s)ds \qquad (6)$$

And the interreflected version is

$$R_I(L_P, R, r) = R_S(L_P, R, r) + \int \overline{L}_P(s)G(s, R, v)(1 - V_P(s))ds \qquad (7)$$

Scalar r defines the glossiness/broadness of the specular response. The above function outputs a scalar radiance in view direction R as a function of the incident lighting L_P and R. These quantities are both unknown at the preprocessing time. With some mathematical formulation taking advantage of the circular symmetric Phong kernel G(s,R,r) and the convolution property of spherical harmonic equation (iii) the integrals in equation (7) can be factored as the transfer matrix coefficients. The components of the matrix can be computed by integrating a triple product of basis functions using recurrence formulas derived from Clebsch-Gordan series (that are used a lot in quantum mechanics of atom orbitals). They can be computed with numerical integration without projecting one of the functions beforehand. This makes possible to do lot of preprocessing but still put in the lighting environment only at run-time.

3.2 Determining the exiting radiance

The general process of determining the exiting radiance consists of the following steps:

1. The incident lighting L_P is computed. If the object uses interreflections the incident radiance is approximated to be constant at each vertex of the object. This means that the light source (environment map, for instance) is located in infinity. The incident radiance can then be sampled at only one point, for instance in the center of the object. The sampled radiance is expressed as a coefficient vector of spherical harmonic basis function weights. If only a shadowed lighting without interreflections is used also local lighting may be used. Then the incident lighting is sampled at multiple points around the object surface. These sampled coefficient vectors that now differ from each other are used for determining the incident radiance at each vertex of the object (through interpolation of the sampled points).

2. The incident lighting is rotated to the objects coordinate frame using the rotational invariance property discussed earlier.

3. Only a linear transformation is needed at each vertex to obtain exiting radiance. This is accomplished by a dot product for diffuse objects and with matrix-vector multiplication for glossy objects.

4. Diffuse object have already obtained the final exiting radiance on the previous step. However, glossy objects have only transferred incident radiance vector, which needs to be convolved with the objects BRDF at the vertex and then evaluated in the view-dependent reflection direction R.

Sampling the incident radiance at step one can be efficiently implemented on graphics hardware by approximating the lighting environment with low-resolution cube maps. The coefficient integrals of equation (ii) are done by taking samples of the constructed cube maps. A radiance sampling with projecting into spherical harmonic basis takes about 1.2ms on a PIII-933 PC with ATI Radeon 8500 [*10*]. The transfer vectors can also be stored in texture maps and evaluated using the graphics hardware pixel shaders.

If colored lights are used then the rendering process requires three rendering passes. Each RGB color component is rendered individually in the depicted way. We will now make look at the diffuse and glossy cases a bit more deeply so that the process would be clearly understood.

3.2.1 Diffuse case

After the transfer vector basis has been created as a preprocess the real-time rendering may start. The lighting environment is first projected into the spherical harmonic function basis. This gives the l_i:s of the incident lighting in equation (4).

The rendering of the diffuse self-shadowing and possibly interreflecting object is now done by simply computing the sum of equation (4) at each vertex. This is now a simple

dot product between the lighting environment representation as a coefficient vector and the transfer vector of each vertex.

Note that the influence of each basis function to a vertex is expressed as only one weight. However, this is enough to contain information about the object's self-occlusions and interreflections too. Any low frequency lighting environment can be expressed as coefficients evaluated by projecting that environment onto the basis functions. It can be reconstructed by a weighted sum of all basis functions. If the same lighting environment is rotated the environment can be expressed by projecting it onto the basis after the rotation. However, from the rotational invariance property of the basis functions the projection coefficients after the rotation can be obtained as a function from the coefficients before rotation without really reprojecting the lighting environment again.

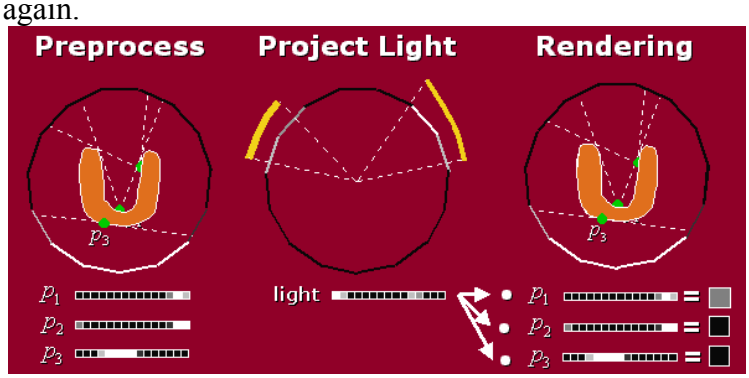


Figure 6. Diffuse object's rendering process. Source [11]. Transfer vectors are computed for each vertex as a preprocess. Then the lighting environment is projected into the spherical harmonic basis functions giving the lighting as another coefficient vector. Rendering is then done by dot product of the lighting vector with each vertex's transfer coefficient vector.

3.2.2 Glossy case

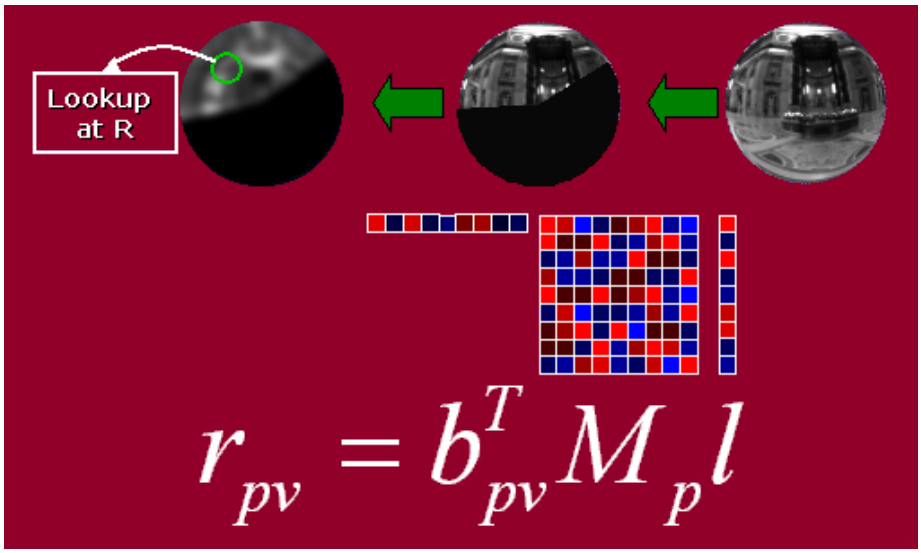


Figure 7. The process of evaluating the lighting of a vertex at real-time for glossy objects. Source [11].

Computing the exiting radiance at a vertex p when a viewing direction v is given requires the evaluation of the above formula for glossy surfaces. The last vector, l, represents the projection of the lighting environment into the spherical harmonic basis. This projection is now done once per every frame. The lighting coefficient vector is then operated by the vertex's transfer matrix M. This maps the global incident radiance into transferred incident radiance. The resulting local radiance sphere needs to be integrated against the BRDF given a fixed view direction v. The BRDF convolution is done as a sum of the transferred incident radiance and the BRDF kernel vector (in the spherical harmonic basis). The convolution step is not shown in the figure. Then the exiting radiance is evaluated along the look direction vector b as a dot product.

The environment map balls in the figure represent the lighting after each step. The rightmost ball is the original lighting. The middle one expresses that the vertex is slightly shadowed on the right as it does not see a full hemisphere. The leftmost ball marks the spot in the seen lighting that the vertex reflects when looked at from the direction R.

The glossy rendering requires a lot more operations in real-time than the diffuse case. However, the rendering can still be real-time if constraints are made. If the view direction is frozen the vector b can be multiplied by the transfer matrix at each vertex. After this the evaluation of reflected radiance requires only the remaining dot product. If the light is frozen it can be operated by the matrix at each vertex. Now b needs to be evaluated at run-time and again remains only a dot product to generate the reflected radiance.

4 EXTENSIONS

In the Sloan et al [*10*] considered expanding the method for additional effects as well. These will now be briefly discussed.

4.1 Arbitrary BRDFs

As has been seen in order to compute the exit radiance the transferred incident radiance needs to be integrated against the BRDF and the cosine factor H_N. The integrals in equation (7) can be rewritten and partially precomputed. However, for arbitrary BRDF's that does not have as convenient properties as the Phong BRDF does we cannot even factor the preprocessing to be calculated in real time. An extension to account for this was suggested where the BRDF is precomputed for all viewing directions. The result is tabulated into texture maps for referral at runtime. The rendering process becomes then different and quite complicated but it can be done in real-time. Kautz, Sloan and Snyder consider the application of the method for arbitrary BRDFs more thoroughly in another article [*9*].

4.2 Volumes

The same framework with small changes can be used also for self shadowing and interreflecting volume models, such as clouds. The volumes are calculated in small

volume units, voxels. Similarly than previously the preprocessing calculates transfer vectors or transfer matrixes but now they are evaluated and stored for each voxel.

4.3 Radiance neighborhood transfer

In the discussed method the objects have influence only on themselves. If there are multiple objects in the scene, the objects do not cast shadows or distribute radiance on each other. This is obviously quite a big limitation. Sloan, Kautz and Snyder suggested a method they called radiance neighborhood transfer to solve this.

Instead of storing the transfer functions on the surface vertices of an object, they are stored also on locations of a point grid in the neighborhood of the object. When another object's surface is places inside the neighborhood volume grid, the stored transfer functions can be used to deduce the other object's influence on the lighting of the other one.

Caustics can be modeled with neighborhood transfer as well. Caustics mean the radiance that is reflected from glossy surfaces onto a glossy or diffuse receiver. Caustics are modeled by simulating them also during preprocessing. Sloan et al [*10*] demonstrated the use of caustics by simulating a mirror surfaced specular ring on a diffuse ground plate. The ground plane received reflected light from the ring as if it was made of shiny metal.

5 DISCUSSION AND RESULTS

The presented method seems quite brilliant indeed. One would expect that, for instance, games could benefit from it greatly with much more realistic real-time graphics. Also lighting designers would be very happy to be able to interactive change lighting and still get almost radiosity quality lighting.

There remain some problems and limitations, though. To the author it seems that one big shortcoming is that objects do not very generally shadow and affect each other's lighting. The neighborhood transfer seems to be a rather limited way to of doing this, as the volume grid size gets large when the radius of influence grows. Another problem for, for instance, real-time computer game graphics is that the objects need to be modeled with rather large amount of vertices. The models that Sloan, Kautz and Snyder used had 50 to 150 thousand vertices. Game objects still usually consist of only few thousand vertices. The amounts are rising quickly, though. And with diffuse surfaces even few thousand vertices would likely look reasonably good.

One more problem is that the objects may not deform or move components relative to each other. So making a human figure where the limbs move relative to each other would already be quite a difficult task. The neighborhood transfer of each limb would need to be considered relative to all others. This was quite slow already with two objects. The preprocessing of an object takes a few hours. However, that does not really matter as in games all object preprocessing results would be saved along with the object files.

Yet one more problem that comes to mind is that games usually model lighting that is not in infinity as the general radiance self transfer with interreflections requires but rather a local surrounding such as a room with lights. Local lighting can be modeled by sampling the lighting at various points near the object surface. However, then only shadows can be used, not interreflections. A general, changing lighting environment can be modeled by creating a low resolution cube map of the lighting environment and then projecting it into spherical harmonic basis. An accurate cube map would require rendering the object's surroundings from 6 directions, basically for each object. Although that the cube map can be very low resolution this is yet much more work to be done each frame.

Low-frequency lighting environments are ofcourse a limitation themselves. Considering games, the method might be mixed with point light sources to bring some moody variations as the shadows in the discussed method are very blurry and cannot reach far. The sharp shadow methods could be mixed with radiance self transfer by cumulating the radiance at each vertex simultaneously from both lighting methods.

REFERENCES

[1] Akenine-Möller, T, Assarsson, U. Approximate Soft Shadows on Arbitrary Surfaces using Penumbra Wedges. *13th Eurographics Workshop on Rendering 2002, pp. 309-318, June 2002.*

[2] Blinn, J.F. Texture and Reflection In Computer Generated Images, *CACM, 19(10), October 1976, pp 542-547.*

[3] Crow, F. Shadows Algorithms for Computers Graphics. *Computer Graphics, Vol. 11, No.3, Proceedings of SIGGRAPH 1977.*

[4] Dobashi, Y, Kaneda, K, Nakatani, H, Yamashita, H. 1995. A Quick Rendering Method Using Basis Functions for Interactive Lighting Design, *Eurographics '95, 229-240.*

[5] Greger, G, Shirley, P, Hubbard, P, Greenberg, D. The Irradiance Volume, *IEEE Computer Graphics and Applications, 6(11):21-29, 1986.*

[6] Hearn, D, Baker, M. P. Computer Graphics C version, second edition. *Prentice Hall 1997.*

[7] Heidrich, W, Seidel, H. Realistic, Hardware-Accelerated Shading and Lighting, *SIGGRAPH 1999, 171-178.*

[8] Heidrich, W, Daubert, K, Kautz, J, Seidel, H. Illuminating Micro Geometry based on Precomputed Visibility, *SIGGRAPH 2000, 455-464.*

[9] Kautz, J.; Sloan, P.; Snyder, J. 2002. Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics. *Eurographics Workshop on Rendering, 2002, p 291-296.*

[10] Sloan, P.; Kautz, J.; Snyder, J. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics, v 21, n 3, 2002, p 527-536.*

[11] Sloan, P.; Kautz, J.; Snyder, J. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *SIGGRAPH presentation slides,* *http://online.cs.nps.navy.mil/DistanceEducation/online.siggraph.org/2002/Papers/08_LightingAndAppearance/sloan.ppt* (Ref 10.3.2003)

[12] Williams, L. Casting Curved Shadows on Curved Surfaces, *SIGGRAPH 78, 270-274.*