# Light Animation with Precomputed Light Paths on the GPU

László Szécsi
TU Budapest
szecsi@iit.bme.hu

László Szirmay-Kalos
TU Budapest
szirmay@iit.bme.hu

Mateu Sbert
University of Girona
mateu@ima.udg.es

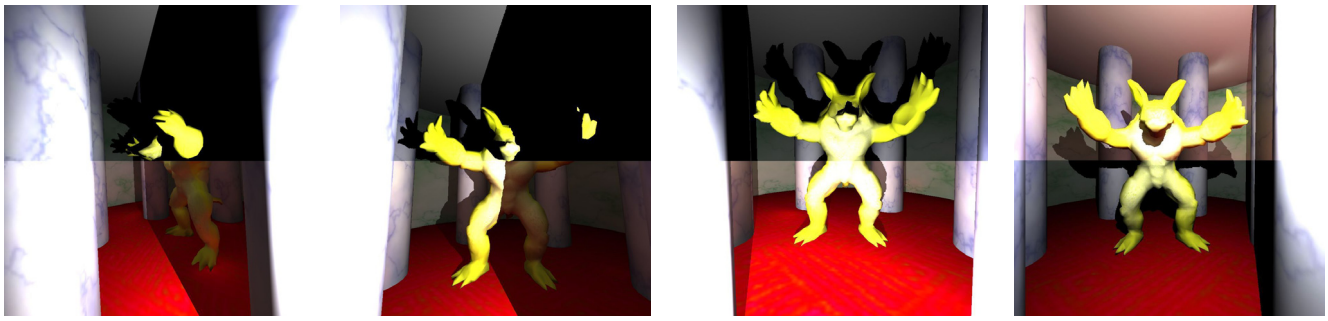


Figure 1: Comparison of local illumination and the proposed global illumination rendering methods. One half of these images has been rendered with local illumination, while the other half with the proposed global illumination method at 40 FPS.

## Abstract

This paper presents a real-time global illumination method for static scenes illuminated by arbitrary, dynamic light sources. The algorithm obtains the indirect illumination caused by the multiple scattering of the light from precomputed light paths. The indirect illumination due to the precomputed light paths is stored in texture maps. Texture based representations allow the GPU to render the scene with global illumination effects at high frame rates even when the camera or the lights move. The proposed method requires moderate preprocessing time and can also work well for small light sources that are close to the surface. The implemented version considers only diffuse reflections. The method scales up very well for complex scenes and storage space can be traded for high frequency details in the indirect illumination.

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

**Keywords:** Real-time global illumination, GPU, light animation.

## 1 Introduction

Rendering requires the identification of those light paths that connect light sources to the eye via reflections and refractions, and then the computation of the sum of their contributions. This summation becomes a high-dimensional integral, which is generally impossible to evaluate in real-time. However, in static scenes we can exploit the fact that the light paths visiting the points on objects do not change when lights or the camera move. This recognition allows us to precompute those integrals that are responsible for the indirect illumination of the scene, and combine the prepared data with the actual lighting conditions during rendering. This means that during rendering just a low dimensional integral needs to be evaluated, which is possible at high frame rates.

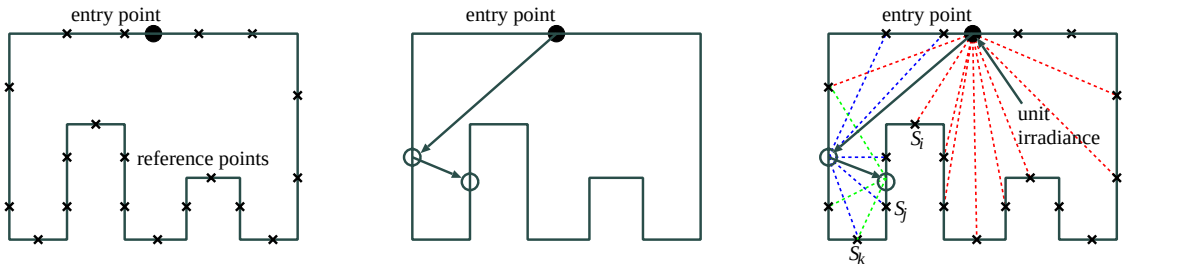
## 2 Previous work

Several approaches have emerged that limit the freedom of object changes in order to make global illumination computations fast. If the scene is static, then radiance transfer coefficients between surface elements do not change, which is exploited in finite-element based radiosity algorithms computing form factors only once. The price is that the required storage is a quadratic function of the number of patches. The idea has been further generalized to include arbitrary number of specular reflections with the introduction of *extended form factors* [14]. Radiance transfer precomputation has also been applied to translucent objects [10]. The storage requirements is a quadratic function of the number of vertices, which prohibits complex scenes and requires well-shaped patches.

*Precomputed radiance transfer* (PRT) [16, 15] can realistically render rigid bodies assuming that a single object is illuminated by low-frequency hemispherical image based lighting. The original idea has been improved in many different ways, including arbitrary BRDFs [6], speeding up preprocessing [9] and introducing principal component analysis to compress data [15]. Ng [12] replaced spherical harmonics by wavelets to allow high frequency environment maps. The fundamental limitation of PRT methods of assuming infinitely distant lighting has been addressed in [1]. A recent paper on local radiance transfer [8] proposed the computation of the response to a set of point lights placed in the scene using PRT, and the weighting of these responses when real lights are introduced during on-line rendering. The preprocessing is complex and large data is dealt with clustered principal component analysis.

*Spherical radiance transfer maps* [11] are data structures to represent both mutual and self-shadows. These maps have to be precomputed for every mesh vertex, and their resolution should be large enough to cover hundreds of sampled directions. This requires a large number of shadow maps, and could result in inaccurate per-pixel results, depending on the tessellation.

*Light path reuse* [13] is a Monte Carlo technique to speed up animated light sequences assuming that the objects and the camera are still. The combination of all paths computed in all frames is governed by multiple importance sampling guaranteeing that the variance of the solution in a frame is close to what could be obtained if we dedicated all paths solely to this particular frame.




1. Reference points are defined and entry points are sampled

2. Path generation from the entry points

3. $S$ is the illumination of reference points
PRM = $\{(\bullet, \times, S)\}$

Figure 2: Overview of the preprocessing phase. Entry points are depicted by $\bullet$, and reference points by $\times$. The PRM is a collection of (entry point $\bullet$, reference point $\times$, illumination $S_k$ ) triplets, called items.

The method presented in this paper is related to the mentioned previous approaches in the sense that it also uses a preprocessing step that computes the radiance of certain reference points on the surface assuming a standard illumination. From a different point of view, the preprocessing computes radiance transfer factors between so called entry and reference points. Then, in the rendering phase the precomputed data is combined with the information of the actual lighting conditions. However, the new approach is not limited to environment lighting, does not require finite-elements, such as spherical harmonics or wavelets, but represents the light transport by a set of pre-generated random light paths stored in a compact way. Representing the lighting information in path space rather than in the space of directional finite elements allows high frequency light sources such as point lights that can be close to the surfaces of the scene, and makes preprocessing times affordable. In this sense our method is strong where PRT is weak. Comparing to local PRT, we also initiate the preprocessing from discrete points but unlike in local PRT that samples the 3D space, our sample points are regularly placed on the 2D surfaces. Reducing the dimension of sampling by one, similar accuracy can be obtained using significantly smaller sample numbers and preprocessing time. Similarly to the PRT method, in our case it is also possible to apply compression that trades accuracy for storage space. However, due to the path space representation, this compression scheme is significantly simpler to implement and results in visually pleasing images even in highly compressed cases.

In our method the light transfer representation is made independent of the vertices of the mesh, which has the advantage that the proposed method does not require carefully prepared meshes consisting of well-shaped triangles of similar size, but performs well on irregular meshes as well. Based on the recognition made by previous Monte Carlo global illumination research [3], the number of light paths needed to render a scene is roughly independent of the complexity of the scene. It means that the number of entry points serving as the origins of the light paths can be set independently of the vertices. Similarly, the reference points that gather the illumination can also be separated from the vertices allowing a more uniform representation. Thus our initial storage complexity, that is the product of the entry and reference points, can be made independent of the vertices, which makes the method scale well for more complex scenes.

## 3 Method Overview

The proposed method consists of a preprocessing step and a fast rendering step.

### 3.1 Preprocessing

The preprocessing step determines the indirect illumination capabilities of the static scene. This information is computed for finite number of *reference points* on the surface, and we use interpolation for other points. The reference points are depicted by symbol $\times$ in figure 2. The reference points can be defined as points corresponding to the texel centers of the texture map of the surface.

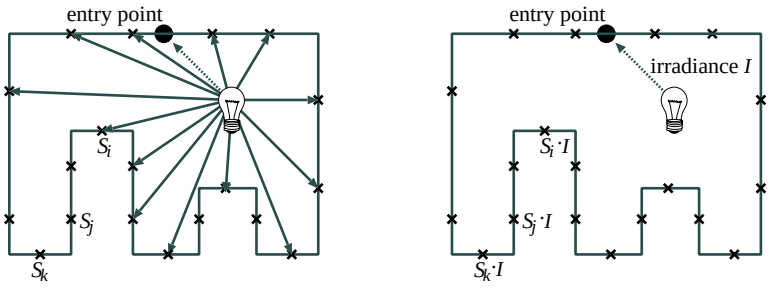
The first step of preprocessing is the generation of certain number of *entry points* on the surface. These entry points are samples of first hits of the light emitted by moving light sources. During preprocessing we usually have no specific information about the position and the intensity of the animated light sources, thus entry points should cover the surfaces densely, and unit incoming radiance is assumed at these sample points. Entry points are depicted by symbol $\bullet$ in figure 2. Entry points are used as the start of a given number of light paths. A light path is a random or a quasi-random walk [7] along the surface. In order to limit the length of paths, we can use Russian-roulette.

The visited points of the generated paths are connected to all those reference points that are visible from them. In this way we obtain a lot of paths originating at an entry point and arriving at one of the reference points. The contribution of a path divided by the probability of the path generation is a Monte Carlo estimate of the indirect illumination caused by the given reference lighting environment. The sum of the Monte Carlo estimates of paths associated with the same entry and reference point pair is stored. We call this data structure the *precomputed radiance map*, or *PRM* for short. Thus a PRM contains *items* corresponding to groups of paths sharing the same entry and reference points. Items that belong to the same entry point constitute a PRM *pane*.

### 3.2 Rendering

During real-time rendering, PRM is taken advantage of to speed up the global illumination calculation. The lights and the camera are placed in the virtual world (figure 3). The direct illumination effects are computed by standard techniques, which usually include some shadow algorithm to identify those points that are visible from the light source. PRM can be used to add the indirect illumination. This step requires visibility calculations, which is for free, since this visibility information was already obtained during direct illumination computation when shadows were generated [2].

A PRM pane stores the indirect illumination computed for a light ray coming from the sampled direction and causing unit irradiance. During the rendering phase, however, we have to adapt to a different lighting environment, that is, to consider other light rays of different




1. Direct illumination + entry point visibility  2. Weighting irradiance $I$ with items $S$

Figure 3: Overview of the rendering phase. The illumination of the entry points are computed, from which the illumination of the reference points is obtained by weighting according to the PRM.

radiance and direction arriving at the same entry point. Taking into account the differences of carried radiance and incoming direction, the PRM pane associated with this entry point should be weighted in order to make it reflect the actual lighting situation. Doing this for every entry point hit by a light ray and adding up the results, we can obtain the visible color for each reference point. Then the object is rendered in a standard way with linear interpolation between the reference points.

In order to compute the weights, we have to take into account the contribution and the probability density of the light paths obtained during preprocessing. The mathematics needed for this computation is discussed in the following section.

## 4 Formal discussion of the method

A global illumination approach should evaluate the infinite Neumann series containing high-dimensional integrals. In the proposed approach these high-dimensional integrals are partly precomputed, i.e. the integral along all but one variable is calculated in the preprocessing phase. Then the remaining one-variate quadrature is evaluated during rendering.

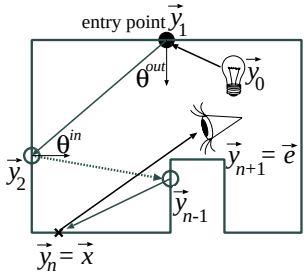



Figure 4: Notations used in the formal discussion

Global illumination computes the radiance (power density) of point $\vec{x}$ in the direction of eye $\vec{e}$ by summing the contribution of all light paths that originate at the light sources and arrive at the eye from point $\vec{x}$. Let us denote the visited points of a path by $\vec{y}_0$ (light source), $\vec{y}_1$ (entry point on the surface), $\vec{y}_2,\ldots,\vec{y}_{n-1}$ (internal path points), $\vec{y}_n = \vec{x}$ (visible point of the surface), $\vec{y}_{n+1} = \vec{e}$ (eye) (figure 4). The contribution of this path is

$$L^e(\vec{y}_0 \to \vec{y}_1) \cdot G_0 \cdot f_1 \cdot G_1 \cdot \ldots \cdot f_{n-1} \cdot G_{n-1} \cdot f_n,$$

where $L^e(\vec{y}_0 \to \vec{y}_1)$ is the emitted radiance from $\vec{y}_0$ toward $\vec{y}_1$,

$$G_k = G(\vec{y}_k,\vec{y}_{k+1}) = \frac{\cos\theta^{out}_{\vec{y}_k} \cdot \cos\theta^{in}_{\vec{y}_{k+1}}}{|\vec{y}_k - \vec{y}_{k+1}|^2} \cdot v(\vec{y}_k,\vec{y}_{k+1})$$

is the *geometry factor* where $\theta^{out}_{\vec{y}_k}$ is the angle between the surface normal at $\vec{y}_k$ and outgoing direction $\vec{y}_k \to \vec{y}_{k+1}$, $\theta^{in}_{\vec{y}_{k+1}}$ is the angle between the surface normal at $\vec{y}_{k+1}$ and incoming direction $\vec{y}_{k+1} \to \vec{y}_k$, *visibility function* $v(\vec{y}_k,\vec{y}_{k+1})$ indicates if the two points are not occluded from each other, and $f_k = f(\vec{y}_{k-1} \to \vec{y}_k \to \vec{y}_{k+1})$ is the BRDF of point $\vec{y}_k$ for directions $\vec{y}_{k-1} \to \vec{y}_k$ and $\vec{y}_k \to \vec{y}_{k+1}$. In case of diffuse materials, the BRDF depends only on reflection point $\vec{y}_k$.

In order to calculate the radiance of $\vec{x}$ at the direction of the eye, all light paths ending at $\vec{x}$ should be considered and their contribution added, which leads to an infinite sum of high-dimensional integrals:

$$L(\vec{x} \to \vec{e}) = \sum_{n=2}^{\infty} \int_{\vec{y}_0} \ldots \int_{\vec{y}_{n-1}} L^e \cdot G_0 \cdot f_1 \cdot \ldots \cdot G_{n-1} \cdot f_n \, dy_{n-1} \ldots dy_0. \quad (1)$$

Note that the length of the light paths starts at 2, because now we are interested in the indirect illumination.

In order to speed up the evaluation of these high-dimensional integrals during rendering, the inner integrals along $\vec{y}_1,\ldots,\vec{y}_{n-1}$ are estimated in a preprocessing step. Suppose that we have a sampling scheme that generates partial light path $\vec{y}_1,\ldots,\vec{y}_{n-1}$ with probability density $p(\vec{y}_1,\ldots,\vec{y}_{n-1})$, where path length $n$ is also a subject of sampling.

Having obtained $N$ samples $(\vec{y}^i_1,\ldots,\vec{y}^i_{n_i-1})$, $(i = 1,\ldots,N)$ with the sampling scheme, we can replace the inner integrals of equation 1 by their Monte Carlo estimate:

$$\int_{\vec{y}_0} \ldots \int_{\vec{y}_{n-1}} L^e(\vec{y}_0 \to \vec{y}_1) \cdot G_0 \cdot f_1 \cdot \ldots \cdot G_{n-1} \cdot f_n \, dy_{n-1} \ldots dy_0 \approx$$

$$\sum_{i=1}^{N} \int_{\vec{y}_0} L^e(\vec{y}_0 \to \vec{y}^i_1) \cdot G^i_0 \cdot \frac{f^i_1 \cdot G^i_1 \cdot f^i_2 \cdot \ldots \cdot G^i_{n_i-1} \cdot f^i_{n_i}}{N \cdot p(\vec{y}^i_1,\ldots,\vec{y}^i_{n_i-1})} \, dy_0.$$

Note that factor

$$R_i = \frac{f^i_1 \cdot G^i_1 \cdot f^i_2 \cdot \ldots \cdot G^i_{n_i-1} \cdot f^i_{n_i}}{N \cdot p(\vec{y}^i_1,\ldots,\vec{y}^i_{n_i-1})} \quad (2)$$

depends only on $(\vec{y}_1,\ldots,\vec{y}_n)$ and is independent of the illumination and viewing directions. This term can be precomputed for each reference point $\vec{x}=\vec{y}_n$, and stored together with pair $(\vec{x},\vec{y}_1^i)$. From stored values $R_i$, the reflected radiance can be computed as a low dimensional integral:

$$L(\vec{x}\to\vec{e}) \approx \sum_{i=1}^{N} \int_{\vec{y}_0} L^e(\vec{y}_0\to\vec{y}_1^i)\cdot G_0^i \cdot R_i \, dy_0,$$

where the integrand is precomputed value $R_i$ multiplied by *weight* $G_0^i$ and emission $L^e(\vec{y}_0\to\vec{y}_1^i)$, both representing the actual lighting and depend only on entry point $\vec{y}_1^i$. For those samples that share this point, the order of weighting and summation can be exchanged, thus different precomputed factors $R_i$ can be summed in the preprocessing phase. Let us denote the sum of those factors $R_i$ which share entry point $k$ by $S_k$. Using these summed factors we can express the radiance in the following form:

$$L(\vec{x}\to\vec{e}) \approx \sum_{k=1}^{K} \int_{\vec{y}_0} L^e(\vec{y}_0\to\vec{y}_1^k)\cdot G_0^k \cdot S_k \, dy_0, \qquad (3)$$

where $K$ is the number of different entry points. Summed factors $S_k$ are the *items* of the *precomputed radiance map* (*PRM*) associated with reference point $\vec{x}$. An item of the PRM is selected by entry point $\vec{y}_1$ and reference point $\vec{x}$, and represents the Monte Carlo estimate of the indirect illumination of point $\vec{x}$ when the light ray causing unit irradiance arrives at the surface at $\vec{y}_1$. The objective of preprocessing is the computation of these items for the reference points and entry points.

Having obtained the items of the PRM, the computation of the indirect illumination caused by a small light source is straightforward. Let us denote the origin and the area of the source by $\vec{y}_0$ and $\Delta y_0$, respectively. Substituting these into equation 3, the indirect reflected illumination of reference point $\vec{x}$ is:

$$L(\vec{x}\to\vec{e}) \approx \sum_{k=1}^{K} L^e(\vec{y}_0\to\vec{y}_1^k)\cdot S_k \cdot \int_{\vec{y}_0} G_0^k \, dy_0 \approx$$

$$\sum_{k=1}^{K} L^e(\vec{y}_0\to\vec{y}_1^k)\cdot v(\vec{y}_0,\vec{y}_1^k)\cdot \frac{\cos\theta_{\vec{y}_0}^{out}\cdot\cos\theta_{\vec{y}_1^k}^{in}}{|\vec{y}_0-\vec{y}_1^k|^2+\Delta y_0/\pi}\cdot S_k. \qquad (4)$$

Note that we applied the point to disc form factor approximation [4]. In the special case when the source is a point light with total emission power $\Phi^e$, the formula is written as

$$L(\vec{x}\to\vec{e}) = \sum_{k=1}^{K} \frac{\Phi^e}{4|\vec{y}_0-\vec{y}_1^k|^2\pi}\cdot v(\vec{y}_0,\vec{y}_1^k)\cdot\cos\theta_{\vec{y}_1^k}^{in}\cdot S_k.$$

In order to use these formulae, we have to check whether or not the entry points are visible from the light source and carry out the summation only for the visible entry points.

### 4.1 Definition of the sampling scheme

The core of the presented method is the sampling scheme which obtains light paths $(\vec{y}_1,\ldots,\vec{y}_{n-1})$ with probability density $p(\vec{y}_1,\ldots,\vec{y}_{n-1})$. All those schemes where $p$ is not zero for paths carrying nonzero radiance are unbiased in Monte Carlo sense, i.e. the expected value of the estimator gives back the correct result. We should prefer those sampling schemes that meet this criterion and have small variance, and consequently result in small error.

The first step of sampling is the generation of $N_e$ entry points on the surfaces with probability density $p^e(\vec{y}_1)$. The simplest approach obtains these points uniformly, first selecting patches proportionally to their area, then a random point on the patch with uniform distribution.

Taking the entry point as the origin $N_s\cdot a(\vec{y}_1)$ number of paths are tried to be generated where $a(\vec{y}_1)$ is the albedo of the surface, and *splitting factor* $N_s$ is a global constant of the method. The reason behind splitting is that in this way the number of random paths can be increased without increasing the number of entry points, i.e. the size of the PRM. The direction of the ray originating in a particular entry point is obtained with cosine distribution.

If the ray hits the surface again, then at the hit point a new direction is sampled with BRDF sampling and this step is continued until the path is terminated by Russian roulette. If the path is terminated, then the hit points are assumed to be *virtual light sources* [7, 18] that illuminate the reference points visible from them. In order to compute this, all reference points are tried to be connected with the points of the paths by shadow rays.

The proposed instructions establish a sampling scheme that obtain point sequences $(\vec{y}_1,\ldots,\vec{y}_{n-1})$ of random length $n$ on the surface. Let us now consider the probability density of these sequences.

At entry point $\vec{y}_1$ we decide whether or not a random path is initiated using probability $a(\vec{y}_1)$. If the path is needed, we sample the first direction from cosine distribution, thus the probability density of selecting $\vec{y}_2$ as the second point of the path, given entry point $\vec{y}_1$, is:

$$\frac{a(\vec{y}_1)}{\pi}\cdot G(\vec{y}_1,\vec{y}_2).$$

In the following steps, we apply Russian roulette and BRDF sampling again to obtain a new direction. Note that in case of diffuse materials BRDF sampling results in the application of cosine distribution similarly to the entry point. The probability density of a complete path $(\vec{y}_1,\ldots,\vec{y}_{n-1})$ given that it originates at $\vec{y}_1$ is then

$$p_{\vec{y}_1}(\vec{y}_1,\ldots,\vec{y}_{n-1}) =$$

$$\frac{a(\vec{y}_1)}{\pi}\cdot G(\vec{y}_1,\vec{y}_2)\cdot\ldots\cdot\frac{a(\vec{y}_{n-2})}{\pi}\cdot G(\vec{y}_{n-2},\vec{y}_{n-1}).$$

The unconditional density of sequences $(\vec{y}_1,\ldots,\vec{y}_{n-1})$ is the product of this conditional probability and the probability of selecting entry point $\vec{y}_1$ ($p^e(\vec{y}_1)$).

## 5 Implementation

The new algorithm consists of a preprocessing step, which builds the PRM, and a rendering step, which takes advantage of the PRM to evaluate indirect illumination. For the preprocessing step we implemented a combined CPU ray-tracing and GPU method. On the other hand, the rendering step was realized completely on the GPU.

The first step of preprocessing is the generation of entry points and of paths, which is executed by the CPU. Having completed a path, we compute the visibility between each path point and each reference point. Assuming that the entry point has unit irradiance, the contributions at the reference points are evaluated. For each reference point, the sum of the contributions is stored together with the index of the entry point, which constitutes the PRM. In fact, we should execute a virtual light source algorithm [7, 18]. Since the reference points are the texels of a texture map, the virtual light source algorithm should be implemented in a way that it renders into a texture map. Items are computed by rendering into the texture with the random walk nodes as light sources. Visibility can be determined using the shadow map technique.

A single texel stores a PRM item that represents the contribution of all paths connecting the same entry point and reference point. A PRM can thus be imagined as an array indexed by entry points and

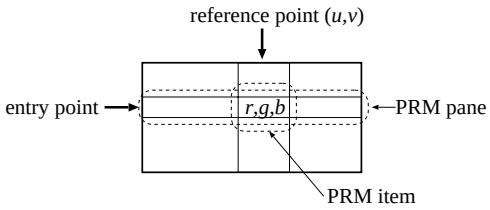



Figure 5: Representation of a PRM as an array indexed by entry points and reference points. A single element of this map is the PRM item, a single row is the PRM pane.

reference points, and storing the radiance on the wavelengths of red, green, and blue (figure 5). Since a reference point itself is identified by two texture coordinates $(u,v)$, a PRM can be stored either in a 3D texture or in a set of 2D textures (figure 6), where each represents a single PRM pane (i.e. a row of the table in figure 5, which includes the PRM items belonging to a single entry point).

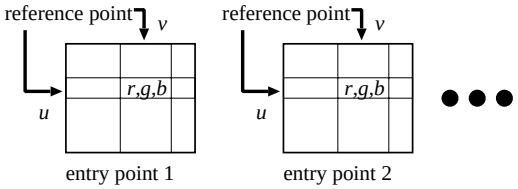



Figure 6: PRM stored as 2D textures

The number of 2D textures is equal to the number of entry points. However, the graphics hardware has just a few texture units. Fortunately, this can be sidestepped by tiling the PRM panes into one or more larger textures.

### 5.1 Entry point clusters

Using the method as described above allows us to render indirect illumination interactively with a typical number of 256 entry points. While this figure is generally considered sufficient for a medium complexity scene, difficult geometries and animation may emphasize virtual light source artifacts as spikes or flickering, thus requiring even more samples. Simply increasing the number of entry points and adding corresponding PRM panes would quickly challenge the hardware in terms of texture memory. To cope with this problem, we can apply an approximation (a kind of lossy compression scheme), which keeps the number of panes under control when the number of entry points increase.

The key recognition is that if two entry points are close and lay on similarly aligned surfaces, then their direct illumination will be probably very similar during the light animation. Of course this is not true when a hard shadow boundary separates the two entry points, but due to the fact that a single entry point is responsible just for a small fraction of the indirect illumination, these approximation errors can be tolerated and do not cause noticeable artifacts. Based on this recognition, near entry points are clustered and only single weight per reference point is stored for the cluster.

Errors caused by clustering are in the low frequency domain, which are not disturbing for the human eye. Clustering also helps to eliminate animation artifacts. When a small light source moves, the illumination of an entry point may change abruptly, possibly causing flickering. If multiple entry points are clustered together, their average illumination will change smoothly. This way clustering also trades high-frequency error in the temporal domain for low-frequency error in the spatial domain. Our clustering approach aims at compressing indirect illumination information similarly to precomputed radiance transfer and to Lightcuts [19]. However, in our case not the incoming radiance field is compressed, but the indirect illumination capabilities, which have low-frequency characteristics.

To reduce the number of panes, contributions of a cluster of nearby entry points are added and stored in a single PRM pane. As these *clustered entry points* cannot be separated during rendering, they will all share the same weight when the entry point contributions are combined. This common weight is obtained as the average of the individual weights of the entry points. Clusters of entry points can be identified by the K-means algorithm [5] or, most effectively, by a simple object median splitting kd-tree. It is notable that increasing the number of samples via increasing cluster size $N_c$ has only a negligible overhead during rendering, namely the computation of more weighting factors. The expensive access and combination of PRM items is not affected. This way the method can be scaled up to problems of arbitrary complexity at the cost of longer preprocessing and smoothing the indirect illumination. Clustering entry points corresponds to a low-pass filtering of the indirect illumination, which is usually already low-frequency by itself, thus the filtering does not cause significant error.

### 5.2 Rendering

While rendering the final image, the values stored in the PRM should be summed according to equation 4. Computing *weighting factor*

$$v(\vec{y}_0,\vec{y}_1^k)\cdot\frac{\cos\theta_{\vec{y}_0}^{out}\cdot\cos\theta_{\vec{y}_1^k}^{in}}{|\vec{y}_0-\vec{y}_1^k|^2+\Delta y_0/\pi}$$

involves a visibility check that could be done using ray casting, but, as rendering direct illumination shadows would require a shadow map anyway, it can effectively be done in a shader, rendering to a one-dimensional texture of weights. Note that for spot sources one shadow map is enough, but for omnidirectional lights we need to maintain several shadow maps per light source. Although textures generated during shadow mapping would later be accessible via texture reads, they can be read back and uploaded into constant registers for efficiency. Furthermore, zero weight textures can be excluded, sparing superfluous texture accesses. Furthermore, excluding not only zero but negligible weight clusters, the performance can be increased at the cost of a minor decrease of the accuracy.

In order to find the indirect illumination at a reference point, the corresponding PRM items should be read from the textures and their values summed having multiplied them by the weighting factors and the light intensity. For the sake of efficiency, we can limit the number of entry points to those having the highest weights. Selection of the currently most significant texture panes can be done on the CPU before uploading the weighting factors as constants.

## 6 Results

The proposed method has been implemented on a P4/2GHz computer with NV6800GT graphics card. Figures 1 and 8 show a marble chamber test scene rendered on $1024\times768$ resolution using splitting factor 2. The parameters of the tests are summarized by table 1. We set the PRM pane resolution to $256\times256$, and used the 256 highest weighted entry clusters. In this case the peak texture memory requirement was 128 Mbytes. The constant parameters of the implementation were chosen to fit easily with hardware capabilities, most notably the maximum texture size and the number of temporary registers for optimized texture queries. Assuming an average albedo of 0.66 and a splitting factor of 2, the 4096 entry points

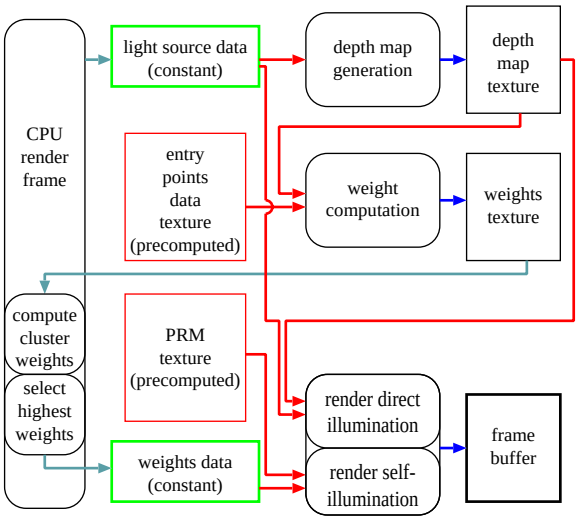



Figure 7: Dataflow in the rendering step

displayed in figure 8 translate to approximately 24000 virtual light sources.

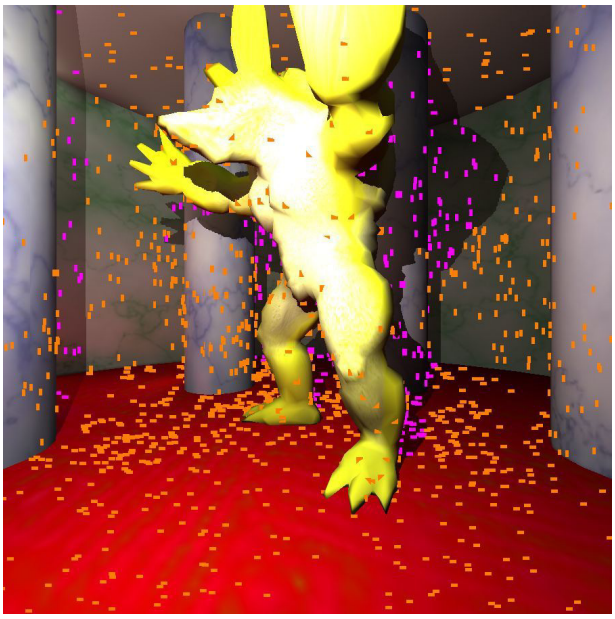


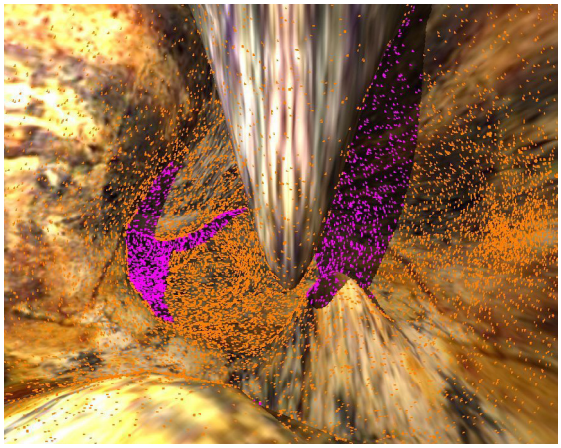
Figure 8: Entry points

For this scene, the preprocessing took 310 sec, which can further be decomposed as building the kd-tree for ray casting, light tracing with ray casting, and PRM generation. Having obtained the PRM, we could run the global illumination rendering at 40 frames per second interactively changing the camera and light positions. Figure 1 shows screen shots where half of the image was rendered with the new algorithm, and the other half with local illumination to allow comparisons. The effects are most obvious in shadows, but also notice color bleeding and finer details in indirect illumination that could not be achieved by fake methods like using an ambient lighting term.

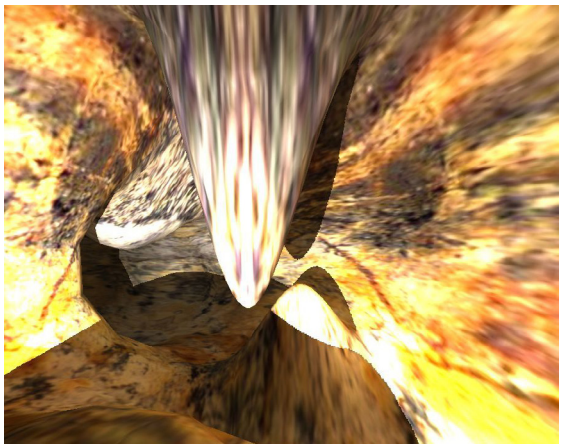
Figure 9 shows a cave rendered with similar parameters, but increasing the number of entry points to 65536.

| Scene | PRM res. | tris | entry pts/ clusters | memory | FPS |
|---|---|---|---|---|---|
| Chamber | $256 \times 256$ | 3335 | 4096/256 | 128Mb | 40 |
| Cave | $256 \times 256$ | 6148 | 65536/256 | 128Mb | 35 |
| Chamber | $128 \times 128$ | 3335 | 4096/32 | 4Mb | 40 |
| Chairs | $128 \times 128$ | 7938 | 4096/32 | 28Mb | 35 |

Table 1: Test scenes and results.



entry points

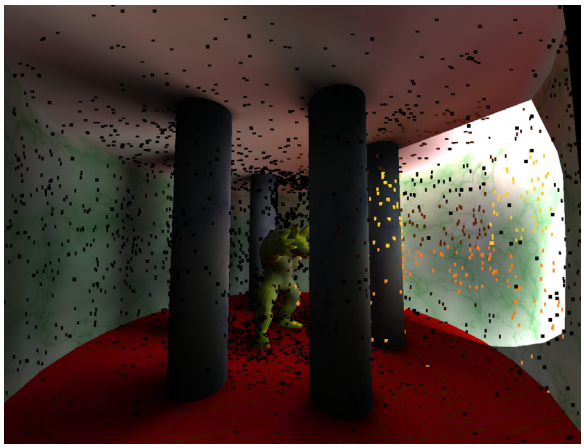


direct + indirect illumination

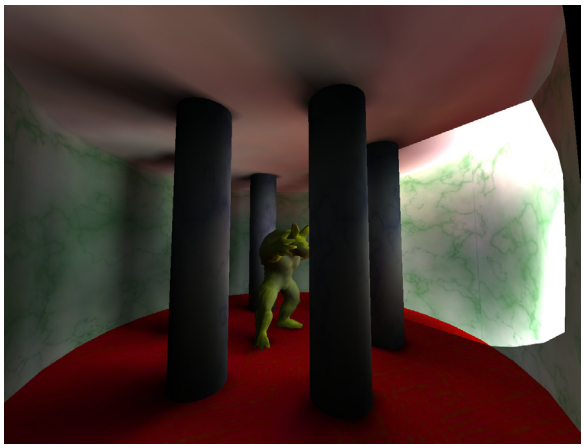
Figure 9: The Cave defined by 6148 faces rendered at 35 FPS

Then we rendered the marble chamber scene using 4096 entry points organized in 32 clusters and reducing the PRM pane resolution to $128 \times 128$ (figure 10). This reduced the preprocessing time and the texture space to 10 seconds and 4Mb, respectively, but caused just minimal image quality degradation.

Finally we took a scene of chairs in a room, and used again just $128 \times 128$ resolution panes and 32 entry point clusters. The results of figure 11 show that these settings provide good results for this scene as well, but need only 28Mb texture space.



entry points



direct + indirect illumination

Figure 10: The chamber scene rendered using 32 entry point clusters. The rectangular spot light source illuminates the wall on the right, at other parts of the scene only indirect illumination is visible.

## 7 Conclusions

This paper proposed a global illumination algorithm for static scenes, but allowing fast moving light sources and camera. Since the geometry is static, we could pre-compute most of the integrals of the infinite Neumann series of a full global illumination solution. This means that in the rendering phase, when light sources are introduced, only a one-variate integral needs to be evaluated, which is possible in real-time. The method can be used in walk-through animations when the avatar takes his own light sources with him, and in lighting design when the light sources are placed interactively and also in games and real-time animations. Of course, the geometry is not constant in games, but in a typical gaming environment, the static environment is usually much larger than the moving dynamic objects. In this case we can suppose that the dynamic objects alter only the direct illumination and the caustic patterns on the large static environment, but its indirect illumination is not affected. Thus we can still use the proposed method for rendering the environment under dynamic lighting, and apply some fast final gathering technique [17] to shade dynamic objects.


## Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T042735), GameTools, and by TIN2004-07451-C03-01 from the Spanish Government. The scenes have been modeled by Maya that was generously donated by AliasWavefront.


## References


[1] T. Annen, J. Kautz, F. Durand, and H-P. Seidel. Spherical harmonic gradients for mid-range illumination. In *Eurographics Symposium on Rendering*, 2004.

[2] C. Dachsbacher and M. Stamminger. Reflective shadow maps. In *SI3D '05: Proc. of the 2005 Symp. on Interactive 3D Graphics and Games*, pages 203–231, 2005.

[3] P. Dutre, P. Bekaert, and K. Bala. *Advanced Global Illumination*. A K Peters, 2003.

[4] A. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, Inc., San Francisco, 1995.

[5] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Analysis and Mach. Int.*, 24(7):881–892, 2002.

[6] J. Kautz, P. Sloan, and J. Snyder. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *12th EG Workshop on Rendering*, pages 301–308, 2002.

[7] A. Keller. Instant radiosity. In *SIGGRAPH '97 Proceedings*, pages 49–55, 1997.

[8] A.W. Kristensen, T. Akenine-Moller, and H.W. Jensen. Precomputed local radiance transfer for real-time lighting design. In *SIGGRAPH 2005*, 2005.

[9] J. Lehtinen and J. Kautz. Matrix radiance transfer. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 59–64, 2003.

[10] H. Lensch, M. Goesele, Ph. Bekaert, J. Kautz, M. Magnor, J. Lang, and H.-P. Seidel. Interactive rendering of translucent objects. *Computer Graphics Forum*, 22(2):195–195, 2003.

[11] Ch. Mei, J. Shi, and F. Wu. Rendering with spherical radiance transport maps. *Computer Graphics Forum (Eurographics 04)*, 23(3):281–290, 2004.

[12] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph.*, 22(3):376–381, 2003.

[13] M. Sbert, L. Szécsi, and L. Szirmay-Kalos. Real-time light animation. *Computer Graphics Forum (Eurographics 04)*, 23(3):291–300, 2004.

[14] F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, Inc., San Francisco, 1994.

[15] P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered principal components for precomputed radiance transfer. In *SIGGRAPH 2003*, 2003.

[16] P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH 2002 Proceedings*, pages 527–536, 2002.

[17] L. Szirmay-Kalos, B. Aszódi, I. Lazányi, and M. Premecz. Approximate ray-tracing on the GPU with distance impostors. *Computer Graphics Forum*, 24(3):695–704, 2005.

[18] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slussalek. Interactive global illumination using fast ray tracing. In *13th Eurographics Workshop on Rendering*, 2002.

[19] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: A scalable approach to illumination. In *SIGGRAPH 2005*, 2005.

Figure 11: The chairs scene lit by a rectangular spot light, whose illumination can be easily identified. The rest is pure indirect illumination obtained with the proposed method at 35 FPS.