On Making Light Maps Dynamic

László Szécsi and László Szirmay-Kalos

Budapest University of Technology and Economics
szirmay@iit.bme.hu

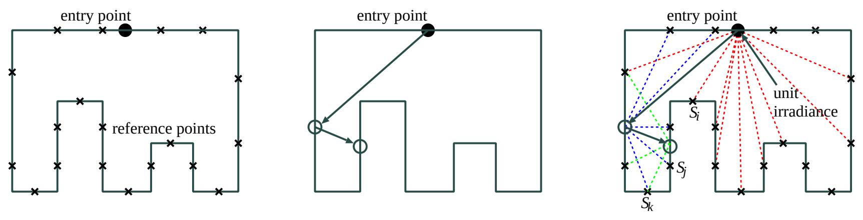
Abstract. This paper presents a real-time global illumination method for static scenes illuminated by arbitrary, dynamic light sources. The method consists of a preprocessing phase and a real-time rendering phase. The real-time rendering algorithm obtains the indirect illumination caused by the multiple scattering of the light from partial light paths that are precomputed and stored in the preprocessing phase. The starting points of partial light paths are called entry points and are sampled randomly on the surfaces. The end points of partial light paths, called exit points, correspond to texel centers of a texture atlas. During preprocessing the radiance of exit points are determined assuming that the entry point has unit irradiance, and the results are stored in texture maps, called *Precomputed Radiance Maps*. Precomputed Radiance Maps represent the self-illumination capabilities of the whole scene. Introducing light sources and the camera in the real-time animation phase, the direct illumination of the real light sources can be computed by evaluating local illumination at the entry points. Modulating the Precomputed Radiance Maps by the obtained entry point irradiance values, the indirect illumination due to multiple reflections can be calculated at the exit points. Adding the direct illumination, the global illumination solution can be obtained instantly even when the camera or the light sources move. The role of the proposed method in games and interactive applications is similar to that of *light maps*. It also renders indirect lighting of static geometry, but unlike light maps, it allows for dynamic lighting and updates indirect shadows and color bleeding effects when light sources move.

1 Introduction

Rendering requires the identification of those light paths that connect light sources to the eye via reflections and then the computation of the sum of the path contributions. Accurate results require a high number of light paths, which are generally impossible to evaluate in real-time. However, in static scenes we can exploit the fact that those parts of light paths which connect surface points do not change when lights or the camera may move. This recognition allows us to precompute these light paths and combine the prepared data with the actual lighting conditions during real-time rendering [3]. This means that having preprocessed the scene we can obtain global illumination results paying the cost of local illumination rendering.

2 The new method

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



1. Exit points are defined, entry points are sampled 2. Path generation from entry points 3. Illumination of exit points PRM $= \{(\bullet, \times, S)\}$

Fig. 1. Overview of the preprocessing phase. Entry points are depicted by $\bullet$, and exit points by $\times$. The PRM is a collection of (entry point $\bullet$, exit point $\times$, radiance S_k) triplets, called items.

2.1 Preprocessing

The preprocessing phase determines the self illumination capabilities of the static scene. This information is computed for finite number of *exit points* on the surface, and we use interpolation for other points. Exit points are depicted by symbol $\times$ in figure 1. Exit points are defined as points corresponding to the texel centers of a texture atlas.

The first step of preprocessing is the generation of certain number of *entry points* on the surface. These entry points are the samples of first hits of the light emitted by moving light sources. Entry points are depicted by symbol $\bullet$ in figure 1. Entry points are sampled randomly and are used as the start of a given number of light paths. A light path is a random or a quasi-random walk [1] on the surfaces, which is terminated randomly according to Russian-roulette.

Let us consider a random light path originating at entry point $\boldsymbol{z}^0$ and visiting points $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^l, \ldots, \boldsymbol{z}^n$. If the scene is diffuse, random directions are sampled with cosine distribution, and the termination probability of Russian-roulette is set to the luminance of the albedo (a), then the probability density of generating subpath $(\boldsymbol{z}^0, \boldsymbol{z}^1, \ldots, \boldsymbol{z}^l)$ is

$$p(\boldsymbol{z}^0, \boldsymbol{z}^1, \ldots, \boldsymbol{z}^l) = P(\boldsymbol{z}^0) \cdot \prod_{i=0}^{l-1} \frac{a(\boldsymbol{z}^i)}{\pi} \cdot G(\boldsymbol{z}^i, \boldsymbol{z}^{i+1}) \qquad (1)$$

where $P(\boldsymbol{z}^0)$ is the probability density of sampling $\boldsymbol{z}^0$ as an entry point and

$$G(\boldsymbol{x},\boldsymbol{y}) = v(\boldsymbol{x},\boldsymbol{y}) \cdot \frac{\cos\theta_{\boldsymbol{x}} \cdot \cos\theta_{\boldsymbol{y}}}{|\boldsymbol{x}-\boldsymbol{y}|^2}$$

is the geometric factor. Here v is the visibility indicator, which is 1 if the two points are visible from each other, and zero otherwise, and $\theta_{\boldsymbol{x}}$ and $\theta_{\boldsymbol{y}}$ are the angles between the direction from $\boldsymbol{x}$ to $\boldsymbol{y}$ and the surface normals at $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

Assuming that the entry point has unit irradiance, the power delivered by this path to hit point $\boldsymbol{z}^l$ is:

$$\Phi(\boldsymbol{z}^0,\boldsymbol{z}^1,\ldots,\boldsymbol{z}^l) = \prod_{i=0}^{l-1} f_r(\boldsymbol{z}^i)\cdot G(\boldsymbol{z}^i,\boldsymbol{z}^{i+1}), \qquad (2)$$

where f_r is the diffuse reflection coefficient, also called the BRDF, which is a function of the wavelength as well. If only a single wavelength is considered, then $f_r = a/\pi$.

Note that in case of sampling directions with cosine distribution and the path continuation probability is equal to the albedo (Russian-roulette), the factors in equation 1 compensate the respective factors in the transported power.

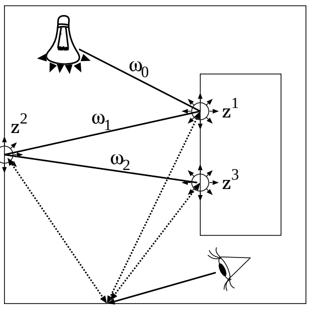


Fig. 2. Virtual light sources algorithm

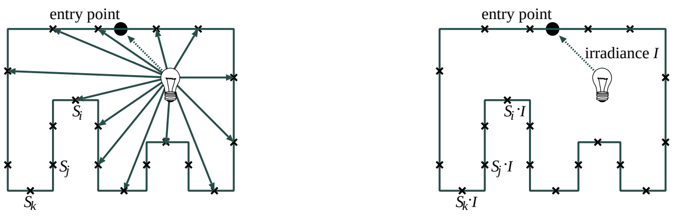
Hit points of the random light path act like virtual, point-like light sources [1, 4]. In order to calculate the radiance at exit points caused by these virtual lights, the points on the surfaces visible through the pixels are connected to all of these virtual light sources by shadow rays in a deterministic manner, as depicted in figure 2. The primary Monte Carlo estimate of the radiance at exit point $\boldsymbol{x}$ is the ratio of the subpath contributions and path probabilities, that is:

$$L(\boldsymbol{x}) = \sum_{l=1}^{n} \frac{\Phi(\boldsymbol{z}^0,\boldsymbol{z}^1,\ldots,\boldsymbol{z}^l)}{p(\boldsymbol{z}^0,\boldsymbol{z}^1,\ldots,\boldsymbol{z}^l)} \cdot f_r(\boldsymbol{z}^l)\cdot G(\boldsymbol{z}^l,\boldsymbol{x})\cdot f_r(\boldsymbol{x}) =$$

$$\frac{1}{P(\boldsymbol{z}^0)} \cdot \sum_{l=1}^{n} \left(\prod_{i=0}^{l-1} \frac{f_r(\boldsymbol{z}^i) \cdot \pi}{a(\boldsymbol{z}^i)} \right) \cdot f_r(\boldsymbol{z}^l) \cdot G(\boldsymbol{z}^l, \boldsymbol{x}) \cdot f_r(\boldsymbol{x}). \tag{3}$$

The average of the primary estimates converges to the real radiance values, which is stored with each entry and exit point pair. We call this data structure the *precomputed radiance map*, or *PRM* for short. Thus a PRM contains *items* corresponding to entry and exit point pairs. Items that belong to the same entry point constitute a PRM *pane*. A PRM pane is an array or a 2D texture of exit point radiances computed with the assumption that the corresponding entry point has unit irradiance while all other entry points have zero irradiance.

2.2 Rendering



1. Direct illumination + entry visibility 2. Weighting irradiance I with items S_i

Fig. 3. Overview of the rendering phase. The irradiance of the entry points are computed, from which the radiance of the exit points is obtained by weighting according to the PRM.

The rendering step is realized completely on the GPU. During real-time rendering, PRM is taken advantage of to speed up the global illumination calculation. 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.

A PRM pane stores the indirect illumination computed for the case when the respective entry point has unit irradiance. During the rendering phase, however, we have to adapt to a different lighting environment. The PRM panes associated with entry points should be weighted in order to make them reflect the actual lighting. Computing the weighting factors involves a visibility check that can effectively be done in a shader using the shadow map already computed for direct illumination. This shader renders into a one-dimensional texture of weights. Although these values 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.

In order to find the indirect illumination at an exit 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. 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.

Having obtained the radiance for each exit point, the scene is rendered in a standard way with linear interpolation between the exit points. Since exit points correspond to texel centers, the required linear interpolation is automatically provided by the bi-linear filtering of the texturing hardware.

3 Storing and compressing PRMs

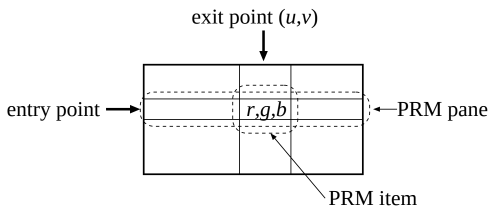


Fig. 4. Representation of a PRM as an array indexed by entry points and exit points. A single element of this map is the PRM item, a single row is the PRM pane.

PRMs are stored in textures for real-time rendering. A single texel stores a PRM item that represents the contribution of all paths connecting the same entry point and exit point. A PRM can thus be imagined as an array indexed by entry points and exit points, and storing the radiance on the wavelengths of red, green, and blue (figure 4). Since an exit 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 5), where each 2D texture represents a single PRM pane (i.e. a row of the table in figure 4, which includes the PRM items belonging to a single entry point).

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.

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 even the latest hardware in terms of texture memory and texture access performance. To cope

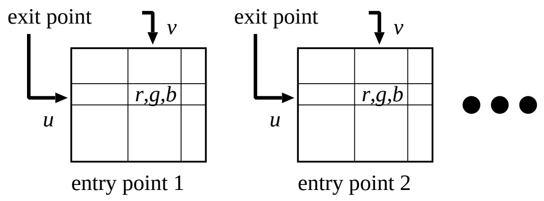


Fig. 5. PRM stored as 2D textures

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 increases.

The compression algorithm clusters those entry points that are close and are on a similarly oriented surface. Contributions of a cluster of 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.

The key recognition behind clustering 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. This property can also be understood if we examine how clustering affects the represented 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. Furthermore, errors in the low frequency domain 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.

4 Results

Figure 6 shows a marble chamber test scene consisting of 3335 triangles, rendered on 1024×768 resolution. We used 4096 entry points (figure 6). Entry points were organized into 256 clusters. We set the PRM pane resolution to 256×256, and used the 32 highest weighted entry clusters. In this case the peak texture memory requirement was 128 Mbytes.

For this scene, the preprocessing took 8.5 sec, which can further be decomposed as building the kd-tree for ray casting (0.12 sec), light tracing with ray casting (0.17 sec), and PRM generation (8.2 sec). Having obtained the PRM, we could run the global illumination rendering interactively changing the camera and light positions. We could measure 40 FPS and 200 FPS in full screen mode on NVidia GeForce 6800 GT and 8800 GTX graphics cards, respectively. The scenes of Figure 7 were also rendered with the same speed.

Figure 6 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.

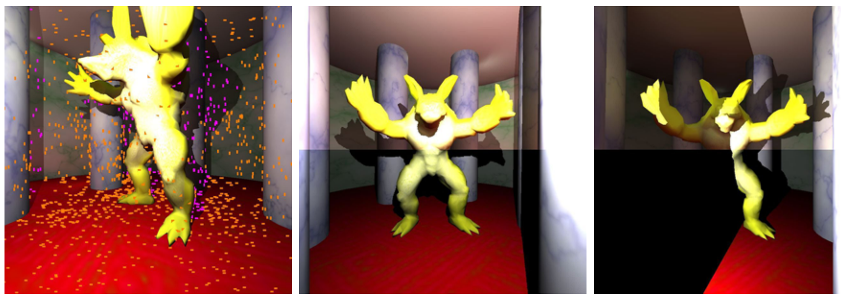


Fig. 6. Entry points and comparison of local illumination and the proposed global illumination rendering methods. The lower half of these images has been rendered with local illumination, while the upper half with the proposed method.

5 Conclusions

The role of the proposed method in games and interactive applications is similar to that of *light maps*. It renders indirect lighting of static geometry, but unlike light maps, it also allows for dynamic lighting. Global illumination computations are performed in a precomputing step, using ray casting and the virtual light sources method. The contributions of virtual light samples are computed with depth mapping on the GPU. However, instead of computing a single light map, multiple texture atlases (constituting the PRM) are generated for the scene objects, all corresponding to a cluster of indirect lighting samples. These atlases are combined according to actual lighting conditions. Weighting factors depend on how much light actually arrives at the sample points used for PRM generation, which is also computed in a GPU pass.

Unlike Precomputed Radiance Transfer [2], the method proposed in this article requires moderate preprocessing time, does not assume low-frequency hemispherical lighting, and can also work well for small light sources that may get close to the surfaces.

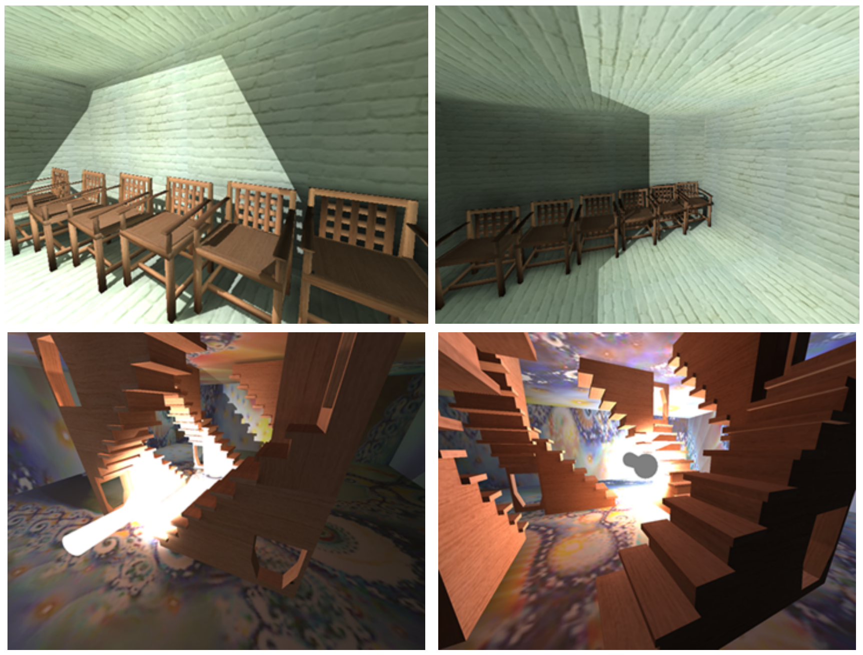


Fig. 7. The chairs and the Escher staircases scenes lit by moving lights

The final result will be a plausible rendering of indirect illumination. Indirect shadows and color bleeding effects will appear, and illumination will change as the light sources move.

Acknowledgements

This work has been supported by OTKA (T042735), GameTools FP6 (IST-2-004363) project, and by the National Office for Research and Technology (Hungary).

References

1. A. Keller. Instant radiosity. In *SIGGRAPH '97 Proceedings*, pages 49–55, 1997.
2. 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.
3. L. Szécsi, L. Szirmay-Kalos, and M. Sbert. Light animation with precomputed light paths on the GPU. In *GI 2006 Proceedings*, 2006.
4. 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.