**Rafal Mantiuk[1,2], Sumanta Pattanaik[1], Karol Myszkowski[3]**
[1] University of Central Florida, USA, [2] Technical University of Szczecin, Poland, [3] Max-Planck-Institut für Informatik, Germany
*rafal.mantiuk@wi.ps.pl*, *sumant@cs.ucf.edu*, *karol@mpi-sb.mpg.de*

# CUBE-MAP DATA STRUCTURE FOR INTERACTIVE GLOBAL ILLUMINATION COMPUTATION IN DYNAMIC DIFFUSE ENVIRONMENTS

**Abstract**

In this paper we present Cube-map data structure for global illumination computation at interactive rates. Our algorithm computes global illumination by iterative computation of irradiance from multiple bounces. Emission from light source is simulated using the direct lighting capability of the GPU. Light bouncing out of surfaces of the scene are captured in cube-maps distributed over the volume of the scene. Subsequent bounces are simulated by querying the cube-map data structure and ambient lighting capability of the GPU based rendere. We are able to compute plausible approximation of indirect light for static and dynamic scenes involving both moving objects and changing light sources. The accuracy of our solution can progressively improve as computation time is increased.

**keywords** : *Global Illumination, radiosity, hardware-assisted rendering.*

## 1 INTRODUCTION

Accurate lighting computation is one of the key ingredients to realism in rendered images. In the past two decades, great progress has been made towards the accurate lighting computation with the development of physically based global illumination algorithms. These algorithms simulate the propagation of light in a three-dimensional environment and compute the distribution of light to desired accuracy. However these algorithms may need minutes to hours to render a single image. Because of the view independent nature of illumination in static diffuse scenes, it is possible to pre-compute the light distribution and use it for interactive rendering. However, such approaches are not of much use in case of interactive rendering of dynamic scenes. Due to this high computation time, many of the today's rendering systems (*e.g.* game systems) use *ad hoc* approaches to provide plausible realistic appearance. For example, they pre-compute light map textures [1] using radiosity method to

emulate global illumination on the surface of static objects (walls, ceiling). Even though static objects give impression of being lit properly, the same can not be said about dynamic objects. Also storing the extensive amount of light maps in texture memory is very often problematic. Pre-computed light maps may give plausible results in game systems but they are useless in design applications, where design decisions are based on the accuracy of the rendering.

In this paper we present a cube-map data structure for dynamic capture of inter-reflection of light from the scene along with a GPU based solution for global illumination computation and realistic rendering of scenes. The OpenGL implementation of our algorithm takes about 3 seconds[1] to compute light distribution from scratch for our example scene without any pre-computations.

## 2 PREVIOUS WORK

The rapidly improving speed and functionality of graphics hardware makes it possible to render images displaying advanced lighting effects (refer to [2] for a complete survey of such techniques). However, the explicit global illumination computation is usually not performed, and although the resulting images look believable, they poorly predict the appearance of the real world. A successful combination of hardware and software rendering was used by Udeshi and Hansen [3] to obtain reflection and refraction effects (ray tracing), soft shadows (shadow volumes), and approximate one-bounce indirect lighting at interactive rates (a small set of virtual point light sources is used). However, a massively parallel machine (64 processors and 8 graphics pipelines) was needed to achieve this goal.

Graphics hardware has been used to improve the efficiency of global illumination computation. For example, interactive rendering of surfaces with arbitrary BRDFs under distant illumination has been performed using environment mapping [4,5,6,7]. However, we have not come across any interactive implementation of global illumination algorithm. In this paper we present such an algorithm.

We combine the algorithm proposed by Ramamoorthi and Hanrahan [7] for efficient prefiltering of environment maps, with the concept of irradiance volumes introduced by Greger et al. [8] to compute the spatially varying illumination in static and dynamic environments.

## 3 ALGORITHM

Our algorithm simulates inter-reflection by repeated gathering of light distribution from each bounce of light rays. The salient steps of this algorithm are:

1. Compute irradiance due to *first* bounce of light from direct light sources (OpenGL point lights and spotlights).
2. Compute irradiance due to $n^{th}$ bounce using irradiance computed from $(n\text{-}1)^{th}$ bounce.
3. Sum up the irradiance values from all bounces of light to compute the final image

[1] For details on time measurements refer to the section *Results.*

For each bounce of light we capture the incoming radiance at a point in the scene by rendering a cube map at that point. Cube map is a projection of an environment on a cube. It can be rendered for any environment by placing a camera in the center of a cube and rendering six rasters, one for each face of the cube. Using Ramamoorthi and Hanrahan's algorithm [7] we filter the cube map pixel values to compute the spherical harmonic representation for the irradiance function at the sample point. We compute such irradiance functions at a grid of sample points, as done by Greger et al [8]. Next, for each vertex of the scene geometry we find 8 neighboring grid points and we interpolate irradiance value between them. The steps required to compute each bounce of light are summarized in Figure 1.

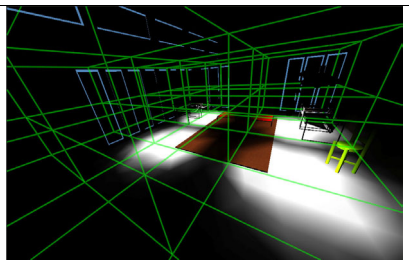
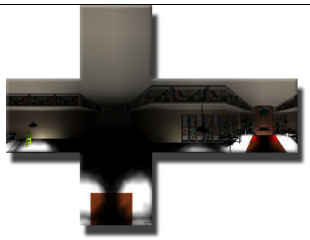
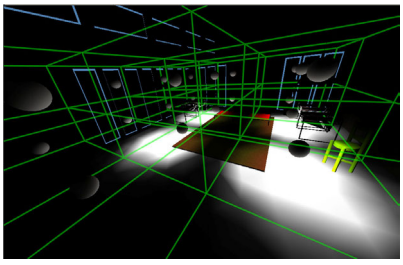
| | | |
|---|---|---|
| Step 1 |  | Divide 3D space into uniform grid of volumes. |
| Step 2 |  | Render a cube map at the center of each volume. The cube map gathers information about incoming radiance. |
| Step 3 |  | Compute spherical harmonic coefficients for every cube map. These coefficients represent the vector irradiance at the center point of the volume. Irradiance for a Lambertian surface point can be computed from coefficients and the normal to the surface at that point. |
| Step 4 |  | Compute irradiance at each surface vertex by linearly interpolating irradiance of 8 neighboring volumes. Irradiance is assumed to vary smoothly both with direction and position. Use the irradiance value to set the ambient color for the vertex. |

*Figure 1: Computation of a single bounce of light reflection*

Because we use hardware to gather information on incoming radiance, we can compute light distribution much faster than any existing global illumination method. Unlike Monte Carlo ray tracing, our solution does not show stochastic noise due to under sampling. If fewer samples are chosen, the resulting accuracy will be lower, however lack of accuracy won't be noticeable as high frequency noise. Our solution exploits smooth nature of irradiance and interpolates its values both between points in space and incoming directions.

## 4 RESULTS

In this section we will focus on three aspects of our implementation: time required to compute global illumination, quality of the resulting images and comparison of our method with global illumination implementation using photon tracing.

Computing a single bounce of light for "Room" scene took only 1.5 seconds, as shown on Figure 2. Usually two bounces of light give plausible results and further computation is not necessary. Therefore good approximation of global illumination could be computed in just about 3 seconds. Even though this is still not a real-time performance, those times will be reduced at least several times in the near future. Over 75% of the time was spent on transferring cube map rasters from GPU to CPU memory and updating color values for vertices (items 2 and 4 in the Figure 2). We expect that both of those activities will be feasible in the future graphics hardware thus those times will be practically reduced to 0. Moreover, each face of the cube was rendered with full level of detail of the scene, whereas simplified geometry would be sufficient for the purpose of irradiance measure. Updating light distribution is necessary only in rare situations: when light sources changes or the scene is significantly reorganized. Computed irradiance can be reused in successive frames for interactive walkthroughs and in situations where small objects are moved. Furthermore, our solutions can show to the user successively refined images, as it computes $n^{th}$ bounce of light.

| | 1 bounce | 2 bounces | 3 bounces | 4 bounces | 5 bounces | 6 bounces | % total |
|---|---|---|---|---|---|---|---|
| 1. Geometry rendering | 393 | 768 | 1068 | 1366 | 1754 | 2131 | 21% |
| 2. Transferring cube maps to CPU memory | 811 | 1655 | 2574 | 3495 | 4295 | 5106 | 51% |
| 3. Computing spherical harmonics | 31 | 47 | 63 | 94 | 140 | 171 | 2% |
| 4. Applying color values | 218 | 702 | 1155 | 1608 | 2077 | 2545 | 26% |
| **5. Total** | **1468** | **3187** | **4890** | **6593** | **8296** | **9999** | |

*Figure 2: Time to compute successive bounces of light (in milliseconds). Times are accumulated for all previous bounces. The time was measured for "Room" scene (12.400 mesh elements) with cube map size: 32x32 and volume grid 3x2x7 = 42 volume elements, on Pentium IV 1.7GHz with ATI Radeon 8500.*

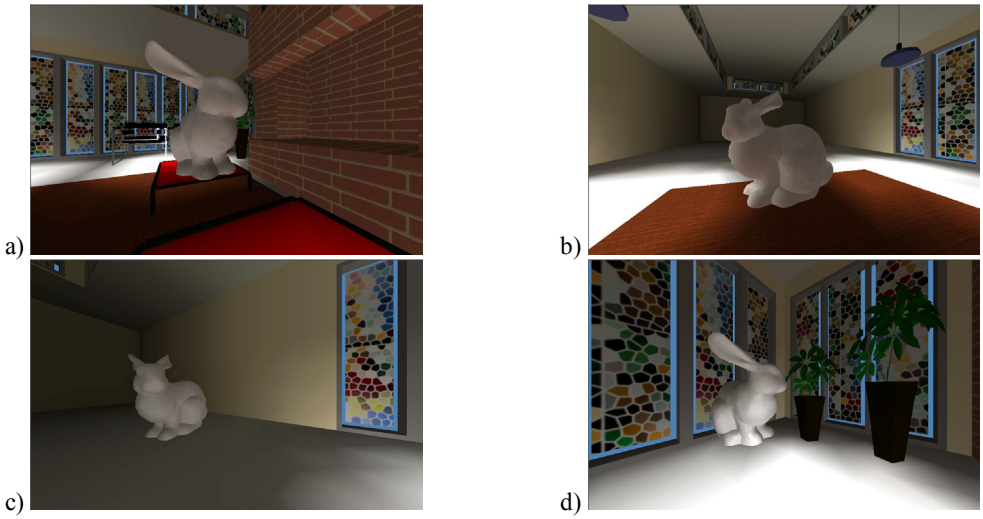



*Figure 3: Object "Bunny"[2] was placed in several places of the "Room" to show differences in illumination. "Bunny" was a bit reddish on a carpet and red table a) b). It became gray when it was moved to the darker part of the "Room" c).*

Figure 3 shows four frames from an interactive demo of moving object "Bunny" in our "Room" scene, where indirect lighting for the moving object is computed on the fly from the grid of irradiance volumes. As we can see from the images, the indirect light on the "Bunny" is appropriate for the local illumination conditions.

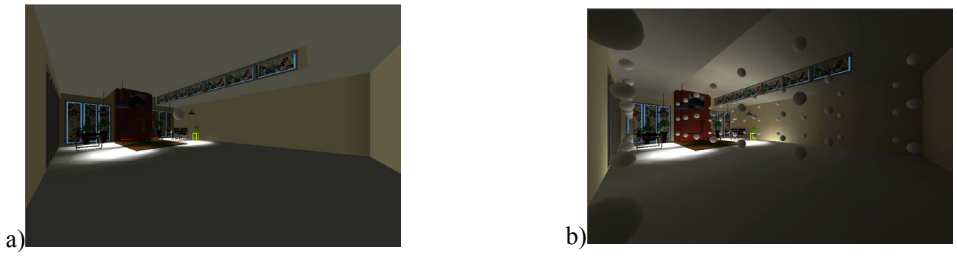



*Figure 4: Image on the left a) was rendered using single volume, whereas right image b) was rendered with a grid of volumes.*

A single environment map failed to capture light distribution in our example scene. The left image on Figure 4 shows a scene rendered with single environment map. Large and flat objects, like walls and a ceiling, are uniformly illuminated while they should be brighter near the light sources. This happens because cube map captures radiance for varying direction but only for a single position whereas radiance in real world varies significantly with both direction and position. Grid of volumes proved to be effective solution for this problem, as it can be seen on right image on Figure 4.

Resulting images from our method were compared with photon tracing. Figure 5 shows a result of such comparison. Both methods show different artifacts thus there are small differences. In our approach ceiling is slightly reddish because volume is located just beneath brown-red carpet. Photon tracing shows shaded areas in the corners, where fewer photons could reach. But in general both images are perceivably similar and give impression of the same light conditions.

[2] Object "Bunny" comes from *The Stanford 3D Scanning Repository* (http://graphics.stanford.edu/data/3Dscanrep/)

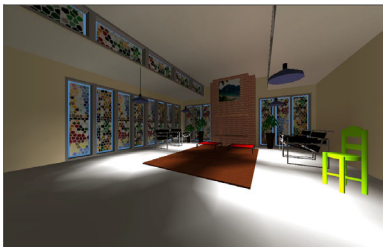
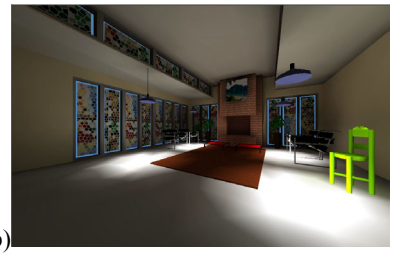
a)  b) 

*Figure 5: Room scene rendered using a) our method (42 volumes) b) photon tracing (1 300 000 rays). Both images show 5 bounces of indirect light. Our method took approximately 6.5 seconds, whereas photon tracing took 27 seconds.*

## 5 CONCLUSIONS

We have provided an algorithm for plausible approximation of global illumination can be computed in near real-time using cube-map data structure and the current graphics hardware. We believe that the computation time will be significantly shorter with the future generations of graphics hardware. Realistic rendering using our algorithm will be much faster, and images generated will be accurate representation of the actual scene that the rendering process is trying to simulate. Thus realistic rendering will find wider acceptance as a tool in diverse fields such as Architecture and Design; Lighting Engineering; Entertainment Industry; Digital Imaging and Retail World.

## REFERENCES


[1] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran and P. Haeberli. “Fast Shadows and Lighting Effects Using Texture Mapping”, In SIGGRAPH 92 Conference Proceedings, pp. 249-252.

[2] W. Heidrich. Interactive Display of Global Illumination Solutions for Non-Diffuse Environments. State of the Art Reports, Eurographics 2000.

[3] T. Udeshi and C.D. Hansen. Towards Interactive Photorealistic Rendering of Indoor Scenes: A Hybrid Approach. In 9th Eurographics Rendering Workshop 1999, pp. 63-76.

[4] N. Greene. Environment Mapping and Other Applications of World Projections. In IEEE Computer Graphics and Applications 1986, 6(11), pp. 21-29.

[5] W. Heidrich and H.-P. Seidel. Realistic, hardware-accelerated shading and lighting. In SIGGRAPH 1999 Conference Proceedings, pp. 171-178.

[6] J. Kautz, P.-P. Vazquez, W. Heidrich, and H.-P. Seidel. A Unified Approach to Prefiltered Environment Maps. In 11th Eurographics Rendering Workshop 2000, pp. 185-196.

[7] R. Ramamoorthi and P. Hanrahan. An Efficient Representation for Irradiance Environment Maps. In SIGGRAPH 2001 Conference Proceedings, pp. 497-500.

[8] G. Greger, P. Shirley, P. Hubbard and D. Greenberg. The Irradiance Volume. IEEE Computer Graphics & Applications 1998, 18(2), pp. 32-43.