Global Illumination with Glossy Surfaces

Wolfgang Stürzlinger

GUP, Johannes Kepler Universität, Altenbergerstr.69, A-4040 Linz, Austria/Europe
wrzl@gup.uni-linz.ac.at

Abstract

Photorealistic rendering is used to generate views of computer stored scenes. Global illumination algorithms take all transfers of light in the scene into account thereby creating a realistic looking image. Previously several approaches have been presented which are able to deal with global illumination for diffuse surfaces. More general surfaces are handled only by few methods.

This work presents a new algorithm for the generation of photorealistic images for scenes with arbitrary surfaces. Initially particle tracing and a reconstruction phase are used to obtain a good approximation to the directionally dependent illumination in the scene. The illumination information is stored and can be used subsequently to generate images from different viewpoints directly from the stored solution. The whole system is structured into several independent phases and is designed to allow parallel processing and incremental refinement.

1 Introduction and Background

This paper presents a new method for the generation of view-independent global illumination solutions of complex environments with arbitrary surfaces. One motivation for a new approach is to avoid the memory overhead associated with storing illumination data for all polygons simultaneously, which makes large scenes impractical to simulate. Also some approaches have quadratic complexity making large scenes prohibitively expensive to compute. Other methods cannot be parallelized for faster computation times. Mirrors and more general surfaces are not taken into account in many approaches.

Lischinski [5] observed that the triangulation for the calculation and the triangulation used in scene display have different purposes and can be handled separately. Shirley *et al.* [7], [11] built upon this concept and proposed a global illumination algorithm which is based on particle tracing and density estimation. This algorithm shoots photons from the light sources, follows these particles until they are absorbed by a surface, and stores the hits. The illumination for a surface is then approximated by examining the density of the absorbed photons for each part of a surface. The approximation is optimized for memory usage and the reduced version is used in the final rendering step to generate images. A more detailed description of each phase follows:

- *Particle-tracing phase:* Each light source emits a number of photons proportional to it's power into the scene. Whenever a photon hits a surface it is randomly reflected, refracted or absorbed depending on the surface characteristics. Each hit point is stored and uses only a few bytes allowing for many photons to be traced.
- *Density-estimation phase:* For each surface the method computes an estimation of the illumination by examining the density of the photon hits. A surface parts which has been hit by many photons is brighter than a surface part which has absorbed only a few photons. An approximation to the irradiance for each surface is computed for a dense grid on each surface.
- *Meshing phase:* The information calculated by the second phase is enough to allow the generation of images. Due to the large amounts of data generated this is impractical. Therefore the dense grid of irradiance samples is simplified with an adaptive triangulation algorithm (as used for height field simplification) yielding a compact representation of the irradiance for each surface.

- *Rendering phase:* The image is generated by tracing rays from the viewpoint into the scene and computing the first surface hit. For each hit point the light reflected to the viewer is computed by interpolating the irradiance at the surface point using the triangulation output by the meshing phase.

The advantages of this method are:
- Diffuse surfaces, perfect mirrors and perfect refractors can be handled within this algorithm.
- The complexity of each phase is less than quadratic which allows the application to large scene datasets.
- The method parallelizes easily. For the particle tracing and rendering phase each photon or viewing ray is independent and can be traced separately. The density estimation and triangulation phase can be applied to multiple surfaces in parallel.

Shirley's method handles diffuse and perfect mirror surfaces well. General surfaces which reflect light depending not only on the outgoing direction but also on the incoming direction cannot be handled consistently. But many surfaces in the real world are neither diffuse nor perfect mirrors. To allow for general surfaces the above algorithm needs to be generalized.

2 Irradiance Reconstruction for Directional Illumination

Similarly to Shirley's approach the algorithm is structured into four phases: particle-tracing, reconstruction, meshing, and rendering. But each of the four phases is enhanced or modified to take directional dependent illumination into account. Also the capability for incremental updates to the global illumination solution is added.

2.1 Particle-tracing phase

Initially each light source λ_i emits a number of photons proportional to it's power Φ_i into the scene. If the light source emits directionally dependent light the emission distribution function is used to generate the directions for the photon, otherwise a uniform distribution is used. Each photon is traced through the scene until it is absorbed or it leaves the scene. At each surface hit a random number is used to decide between absorption, diffuse or glossy reflection. If the photon is reflected the bidirectional distribution function of the material is used to generate a random outgoing direction. The photon hits are stored on disk. The data includes the incoming direction, needed for the reconstruction of the directional irradiance function, a reference to the light source the photon was emitted from, and the attenuation the photon received until it hit the current surface. Storing the attenuation factor renders each hit record independent of the total number of photons emitted by a light source. Whenever the contribution of a photon ϕ is needed it can be recomputed by dividing the power of the light source Φ_i by the current number of emitted photons m_i and weighting the result with the attenuation factor ρ_ϕ:

$$\phi = \frac{\Phi_i}{m_i}\rho_\phi \qquad (1)$$

This scheme is used to implement incremental refinement of the solution. Furthermore it allows to increase of the number of photons emitted by each light source selectively. Note that the selective increase per light source will introduce bias to the solution.

2.2 Reconstruction phase

For each surface the irradiance is reconstructed from the samples stored in the first phase. The total irradiance $E(u,v)$ for each point of the surface and the directional dependent irradiance relative to the total irradiance $D(u,v,\theta,\psi)$ are reconstructed as explained below.

The irradiance $E(u,v)$ for the surface is approximated by constructing a dense regular grid for the surface and inserting the contribution of each hit point ϕ_j by sampling the corresponding kernel function k at the grid points x_j:

$$E(u, v) = \frac{1}{h^2} \sum_{j=1}^{n} \phi_j k\left(\frac{x - x_j}{h}\right) \qquad (2)$$

where h is the scaling factor of the kernel and x is a hit point determined by the surface parameters u,v.

The function $D(u,v,\theta,\psi)$ is the irradiance $L_i(u,v,\theta,\psi)$ coming from a certain direction (θ,ψ) relative to the total irradiance $E(u,v)$:

$$D(u, v, \theta, \psi) = \frac{L_i(u, v, \theta, \psi)}{E(u, v)} \qquad (3)$$

The relative directional irradiance is used because it varies less than the absolute directional irradiance for each surface. The farther a surface is from a light source the smaller the irradiance will be. The relative irradiance cancels this effect and allows for greater compression ratios later on.

To obtain the relative directional irradiance the incoming directional irradiance is computed first. The same reconstruction principle as for the total irradiance is used, only four dimensions are taken into account:

$$L_i(u, v, \theta, \psi) = \frac{1}{h^2 \hat{h}^2} \sum_{j=1}^{n} \phi_j k\left(\frac{x - x_j}{h}\right) k\left(\frac{y - y_j}{\hat{h}}\right) \qquad (4)$$

where y represents the incoming direction (given by θ,ψ) and y_j is a grid point on the hemisphere. The relative directional irradiance is then computed by applying equation (3) at each grid point.

The function $D(u,v,\theta,\psi)$ is stored as set of two dimensional functions for each combination of incoming angles θ and ψ. In other words for each grid point in the grid of all possible incoming angles a two dimensional function on the surface parameter domain is stored. The reasons for this choice of representation are: The irradiance in a certain direction on the hemisphere for neighbouring surface points varies less than the irradiance on neighbouring directions on the hemisphere. This situation is depicted in figure 1:
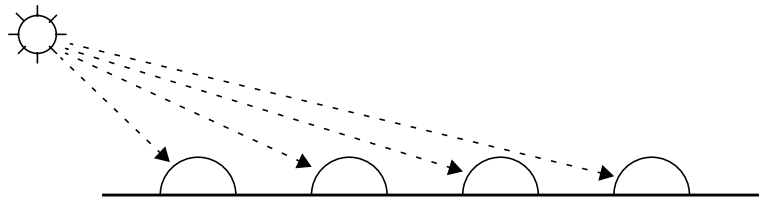

Figure 1. Variation of Directional Irradiance for Surface Points.

This fact allows to benefit from better simplification in the following meshing phase. Another benefit of the chosen representation is that the solid angles needed in the rendering phase are easier to compute.

To obtain a good approximation the irradiance meshes have to be sufficiently dense resulting in large memory requirements in the rendering phase. To ameliorate this the meshes are simplified in the following phase.

2.3 Meshing phase

To reduce the storage requirements of the irradiance representation the meshes are simplified. As the reconstruction phase generates only two dimensional meshes the problem is solved by applying an data dependent adaptive triangulation algorithm for the simplification of height fields to each mesh. This yields a compact representation of the irradiance for each surface and is stored as a triangular subdivision of the surface domain with associated values.

2.4 Rendering phase

The image is generated by tracing rays from the viewpoint into the scene and computing the first surface hit. To compute the diffuse illumination at the hit point the total incoming irradiance is interpolated linearly in the mesh for the function $E(u,v)$. Weighting the irradiance by the diffuse coefficient and colour gives the diffuse component of the outgoing radiance. If the surface is a perfect mirror or refractor a recursive ray is shot into the scene. For general surfaces the outgoing radiance $L_o(u,v,\theta_o,\psi_o)$ is computed by weighting the incoming directional irradiance by the bidirectional reflection distribution function $\rho(\theta_i,\psi_i,\theta_o,\psi_o)$ of the surface:

$$L_o(u, v, \theta_o, \psi_o) = \sum_{\theta_i, \psi_i} L_i(u, v, \theta_i, \psi_i)\rho(\theta_i, \psi_i, \theta_o, \psi_o)d\omega \tag{5}$$

where $d\omega$ is the solid angle associated with each incoming direction θ_i,ψ_i. The directional irradiance is stored for a grid of directions on the hemisphere and the solid angle for a direction is calculated as described in section 3.4.

3 Implementation

The presented global illumination algorithm was realized on a 90 Mhz Pentium PC with 16 megabytes of memory. As local illumination model the model introduced by Ward [8] is used. Scene input uses the MGF file format [10]. In the following a few noteworthy details of the implementation are described for each phase.

3.1 Particle-tracing phase

The information stored on disk for each hit includes the surface number, the surface parameter $(u.v)$, the incoming direction, a reference to the light source the photon was emitted from, and the attenuation factor for the photon due to previous reflections. For storage efficiency the following representations are stored: surface number (4 bytes), surface parameters (2 x 2 bytes), incoming direction angles (2 x 2 bytes), number of the light source (2 bytes), and RGB attenuation factor (3 x 2 bytes) - a total of 20 bytes for each hit point.

3.2 Reconstruction phase

For each surface the contribution of a hit point is inserted in dense grid with an appropriately scaled kernel function. If the support of the kernel function is not totally inside the surface (e.g. for hit points near the border), the contribution is scaled so the volume under the kernel stays unity. This is approximated by calculating the number of grid points inside the kernel support and counting the grid points inside and outside the surface domain. The ratio between inside and outside points is then used to scale the contribution appropriately.

First the total irradiance $E(u,v)$ is computed for the surface. The width of the kernel and the density of the irradiance mesh for a surface k are controlled by the following expressions:

$$h_k = C_1\sqrt{\frac{A_k}{n_k}} \qquad d_k = C_2\sqrt{\frac{A_k}{n_k}} \tag{6}$$

where A_k is the surface area, n_k is the total number of hit points, and C_1 and C_2 are constants. For the relative directional irradiance the width of the kernel and the density are given by:

$$\tilde{h}_k \;=\; C_3\sqrt{\frac{A_0}{n_k}} \qquad \tilde{d}_k \;=\; C_4\sqrt{\frac{A_0}{n_k}} \tag{7}$$

where A_0 is the surface area of the unit hemisphere (2π).

The grid for the hemisphere is the usual azimuth-elevation parametrization. To avoid difficulties with elevation angles near 0 or $\pi/2$ the grid points are shifted by half the grid width in both parameter directions (see figure 2).
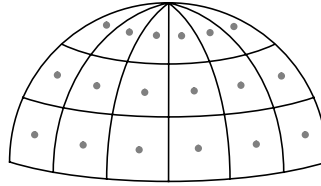

Figure 2. Grid Points on the Hemisphere.

The solid angle corresponding to a grid point is computed easily (see e.g. [1]).
For simplicity the reconstruction uses the three dimensional distance of points on the hemisphere (see equation (4)) when adding the contributions of hit points to the mesh. Although this is not correct the error is small in practise. Useful values for C_1 are in the range of *30-50*. For C_2 the range is $C_1/2$ to $C_1/5$. For the hemisphere the range for C_3 is *75-200*. Useful values for C_4 are found similarly in the range $C_3/2$ to $C_3/5$.
For each surface both functions are stored on disk. To decrease disk usage each function file stores the maximum RGB value across the function and stores all grid points as values relative to this maximum in 3 x 2 bytes.

3.3 Meshing phase

For the simplification of the irradiance function E(u,v) and each θ,ψ slice of the directional irradiance $D(u,v,\theta,\psi)$ an data-dependent adaptive triangulation algorithm as presented by Garland and Heckbert [2] was used. Beginning with a basis mesh the triangle with the highest error is refined until the error is below a predefined threshold.

3.4 Rendering phase

The renderer shoots rays from the eye point through each pixel into the scene. For a surface hit the diffuse and specular component are computed as described in section 2.4.
One problem which arises in practise is that the resolution of the hemisphere is not sufficiently dense for surfaces with a low roughness coefficient. The area on the hemisphere which contributes to the illumination for a given outgoing direction decreases with surface roughness. In the extreme no sample points can be found resulting in no contribution. The image of a red and blue emitter illuminating a plane illustrates the problem and is shown in figure 3:
The constant C_4 for the surface mesh was set to *50* resulting in a mesh with *7* degrees resolution (i.e. a hemisphere mesh with 52x13 entries).
To avoid this situation the algorithm automatically ensures that enough samples are found on the irradiance hemisphere area which contributes for an outgoing direction. New samples are generated adaptively by linear interpolation between the grid points on the hemisphere until

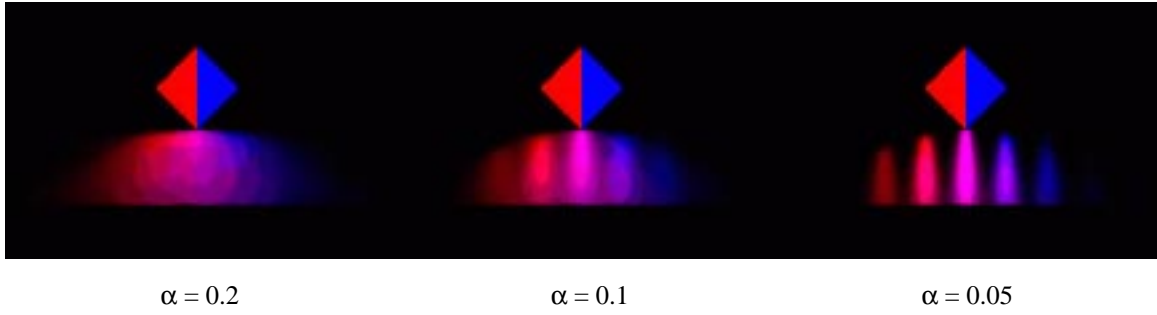$\alpha = 0.2$ $\qquad\qquad$ $\alpha = 0.1$ $\qquad\qquad$ $\alpha = 0.05$

Figure 3. Problem Case: Surfaces with Decreasing Roughness.

the density of the samples is high enough. In the implementation the needed density is determined by the following heuristic dependent on the surface roughness:

$$\partial_{needed} = 100\alpha^2 \tag{8}$$

In figure 4 the results of this method are demonstrated.

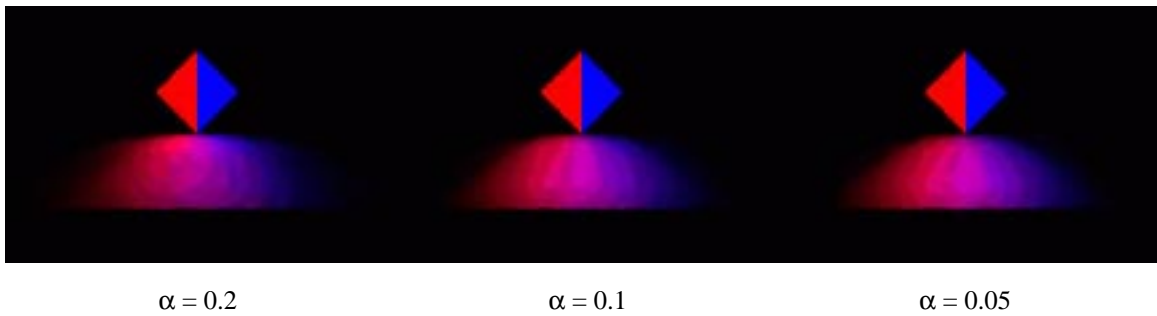$\alpha = 0.2$ $\qquad\qquad$ $\alpha = 0.1$ $\qquad\qquad$ $\alpha = 0.05$

Figure 4. Surfaces with Decreasing Roughness.

3.5 Incremental Refinement

Each hit record stores the attenuation the photon received until it hit the surface. This value and the total number of photons emitted by a light source allow the computation of the contribution of one photon according to equation (1). This technique allows to increase the number of photons emitted by light sources. Even the number of photons emitted by a single light source can be increased if necessary. For an example of incremental refinement for the cornell box see figure 5.
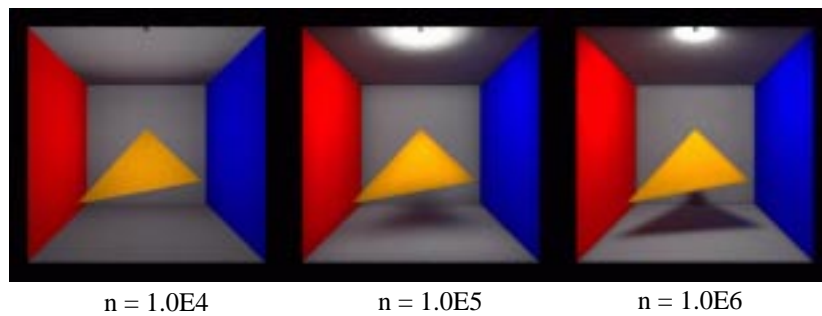

$n = 1.0E4$ $\qquad\qquad$ $n = 1.0E5$ $\qquad\qquad$ $n = 1.0E6$

Figure 5. Increasing the Number of Photons.

3.6 Parallelization

Each photon is independent of others therefore multiple photons can be traced in parallel. The parallel system starts a particle tracing process for each processor available. Each processor outputs the resulting hits to a separate file. After particle tracing has been completed a parallel sort/merge phase separates the photon hit files into separate files for each surface. Each surface is independent of all others and the reconstruction and meshing phases can be run in parallel for multiple surfaces. Parallel rendering proceeds by partitioning the image in horizontal stripes and allocating these stripes to available processors until the image is finished.

4 Results

A simulation of the cornell box with varying surface parameters was performed. The top left view in figure 6 shows an overview of the scene - the background is a perfect mirror surface. In the top right view the larger box is shown with a diffuse material, the lower left shows the box as perfect mirror, and the lower right view is a glossy surface with roughness parameter 0.5.
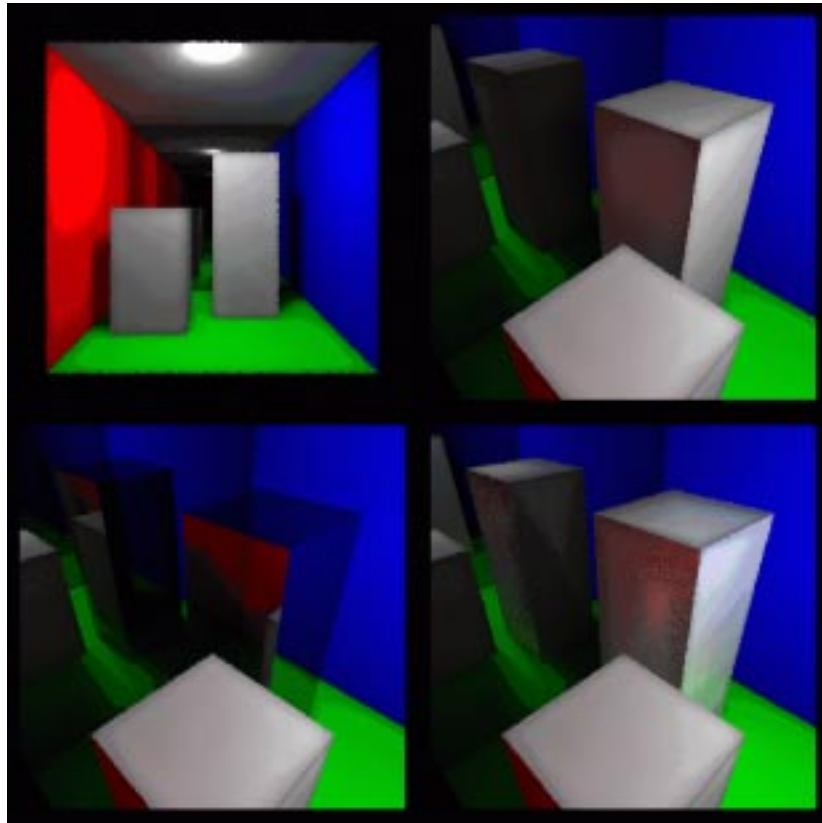

Figure 6. Cornell box with different surfaces (see text).

The second simulation used the office scene available in MGF-format and is shown in figure 7. Timings for these images are shown in the table 1. The large times in the last two examples (marked by an asterisk *) indicate that 16 megabytes main memory were not enough for the rendering phase and the system had to swap to disk.
The code was ported subsequently to a workstation cluster with 8 cpu's. First results show that on average a speed up of about 6.9 is realistic.
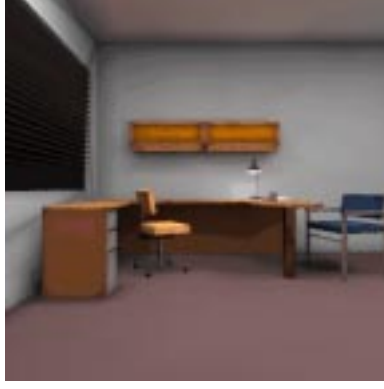
Figure 7. MGF-Office scene.

Parameter/Image	Top Left	Top Right	Bottom Left	BottomRight	Office
# of Photons	1.0E6	1.0E6	1.0E6	1.0E6	2.5E6
Constant C_1	40	40	40	40	30
Constant C_2	20	20	20	20	10
Constant C_3	-	-	-	200	100
Constant C_4	-	-	-	100	50
# of Surface	17	17	17	17	4072
Mesh elements	14029	14029	14155	45781	306093
Resolution	300^2	300^2	300^2	300^2	300^2
Timing (h:m:s)	1:49:20	1:47:28	2:17:00	13:01:58*	11:21:33*

Table 1: Timings for Figures 6 and 7 (see text).

5 Conclusions and Future Work

This work presented a new algorithm for the generation of photorealistic images for scenes with arbitrary surfaces. By storing the incoming direction for each photon in the particle tracing phase the directional irradiance for each surface can be approximated. For memory efficiency the used meshes are simplified subsequently. The rendering phase uses this information to compute the radiance in the direction of the viewer. The system is structured into several independent phases to allow efficient parallel processing. The chosen storage format facilitates also the incremental refinement of the solution.

Other comparable approaches (e.g. [3], [9]) avoid meshing altogether which reduces the memory overhead further. But these method have to evaluate many rays per pixel to reduce the variance significantly whereas the presented algorithm computes the outgoing radiance from the approximated irradiance hemispheres.

Future work includes the investigation of other methods to store the directional irradiance as the memory requirements for meshing each surface are too high for complex scenes (e.g. plants). One solution are spatial density estimation schemes [3], [9]. The integration of Quasi-Monte Carlo Methods [4] with proven subquadratic complexity is planned.

Acknowledgements

References

[1] M. F. Cohen, J. R. Wallace, *"Radiosity and Realistic Image Synthesis"*, Academic Press, 1993.

[2] M. Garland, P. S. Heckbert, *"Fast Polygonal Approximation of Terrains and Height Fields"*,Technical Report Carnegie Mellon University, Pittsburgh, Sept. 1995.

[3] H. W. Jensen, *"Global Illumination using Photon Maps"*, Proceedings of 7th Workshop on Rendering (Porto, 96), pp 22-31, June 1996.

[4] A. Keller, *"Quasi-Monte Carlo Radiosity"*, Proceedings of 7th Workshop on Rendering (Porto, 96), pp 102-111, June 1996.

[5] D. Lischinski, F. Tampieri, D. P. Greenberg, *"Combining Hierarcical Radiosity and Discontinuity Meshing"*, Computer Graphics (SIGGRAPH `93 Proceedings), pp 199-208, Aug 1993.

[6] T. Nöhammer, *"Globale Beleuchtungssimulation mit Particle Tracing"*, Master Thesis, Johannes Kepler University Linz, 1996.

[7] P. Shirley, B. Wade, P. M. Hubbard, D. Zareski, B. Walter, D. P. Greenberg, *"Global Illumination via Density-Estimation"*, Proceedings of 6th Workshop on Rendering (Dublin, 95), pp 187-199, June 1995.

[8] G. J. Ward, *"Measuring and Modeling Anisotropic Reflection"*, Computer Graphics (SIGGRAPH `92), pp 265-272, Aug. 1992.

[9] G. J. Ward, *"The RADIANCE Lighting Simulation and Rendering System"*, Computer Graphics (SIGGRAPH `94), pp 459-472, July 1994.

[10] G. J. Ward, R. Shakespeare, I. Ashdown, H. Rushmeier, *"MGF Parser and Examples"*, http://radsite.lbl.gov/mgf/HOME.html

[11] D. Zareski, B. Wade, P. Hubbard, P. Shirley, *"Efficient Parallel Global Illumination using Density-Estimation"*, Proceedings of ACM Parallel Rendering Symposium, (Atlanta, 95), pp 47-54, Oct. 1995.