Incoming First-Shot for Non-Diffuse Global Illumination[*]

László Szirmay-Kalos, Mateu Sbert, Roel Martínez, Robert Tobler
Department of Control Engineering and Information Technology, Technical University of Budapest
Budapest, Műegyetem rkp. 11, H-1111, HUNGARY
szirmay@fsz.bme.hu

Abstract

This paper presents a method that can replace the small and medium size lightsources by their effect in non-diffuse global illumination algorithms. Incoming first-shot is a generalization of a preprocessing technique called the first-shot that was developed for speeding up global diffuse radiosity algorithms. In order to reduce the prohibitive memory requirements of the original first-shot when it is applied to non-diffuse scenes in a direct manner, the proposed new method computes and stores only the incoming radiance generated by the lightsources and the reflected radiance is obtained from the incoming radiance on the fly taking into account the local BRDF. Since the radiance function of the reflection is smoother and flatter then the original lightsource function, this replacement makes the integrand of the rerdering equation have significantly smaller variation, which can speed up global illumination algorithms. The paper also discusses how the first-shot technique can be built into a stochastic iteration algorithm using ray-bundles, and provides run-time statistics.

Keywords: Non-diffuse global illumination, stochastic iteration, Monte-Carlo quadrature, global methods, finite-element techniques, first-shot

1 Introduction

Global illumination algorithms aim at obtaining the power detected by a collection of measuring devices. The measurement process is characterized by the following equation

$$\int_S \int_\Omega L(\vec{y},\omega) \cdot \cos\theta \cdot W^e(\vec{y},\omega)\, d\vec{y}\, d\omega = \mathcal{M}L, \qquad (1)$$

where $L(\vec{y},\omega)$ is the *radiance*, θ is the angle between the surface normal and direction ω and $W^e(\vec{y},\omega)$ is the *sensitivity* of the measuring device. A measuring device can detect, for example, the power reaching the eye through a pixel.

The radiance function can be obtained by solving the *rendering equation* [4] that has the following form:

$$L = L^e + \mathcal{T}L. \qquad (2)$$

In this integral equation, operator $\mathcal{T}$ describes the light transport

$$\mathcal{T}L(\vec{x},\omega) = \int_\Omega L(h(\vec{x},-\omega'),\omega') \cdot f_r(\omega',\vec{x},\omega) \cdot \cos\theta'\, d\omega' \qquad (3)$$

where $L(\vec{x},\omega)$ and $L^e(\vec{x},\omega)$ are the radiance and emission of the surface in point $\vec{x}$ at direction ω, Ω is the directional sphere, $h(\vec{x},\omega')$ is the visibility function defining the point that is visible from point $\vec{x}$ at direction ω', $f_r(\omega',\vec{x},\omega)$ is the bi-directional reflection/refraction function, and θ' is the angle between the surface normal and direction $-\omega'$ (figure 1).

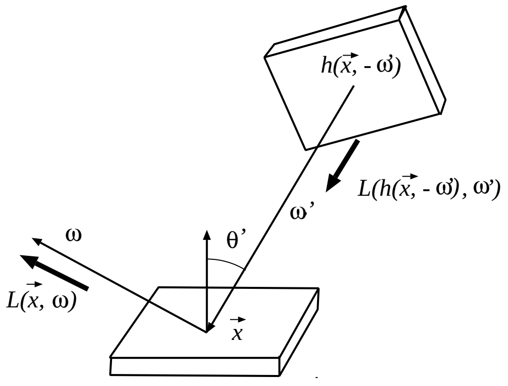


Figure 1: Geometry of the rendering equation

Let us substitute function L in the right side by the complete right side (which equals to L) recursively. If the integral operator is a contraction, this provides the solution in the form of an infinite series:

$$L = L^e + \mathcal{T}L = L^e + \mathcal{T}(L^e + \mathcal{T}L) =$$
$$(L^e + \mathcal{T}(L^e + \mathcal{T}(L^e + \ldots)\ldots). \qquad (4)$$

Thus the measured power is

$$\mathcal{M}L = \mathcal{M}(L^e + \mathcal{T}(L^e + \mathcal{T}(L^e + \ldots)\ldots). \qquad (5)$$

Random-walk [8] and stochastic iteration [9] algorithms evaluate the integrals of this formula by Monte-Carlo

[*]This work has been supported by the National Scientific Research Fund (OTKA ref.No.: T029135), the Spanish-Hungarian Fund (ref.No.: E9) and the Spanish-Austrian Action Fund.

quadrature. Monte-Carlo integration is justified by the facts that its complexity does not grow with the dimension of the domain of the integration and it does not accumulate the error.

The integrals providing the solution of the rendering equation have the following form:

$$\mathcal{T}(L^e + \mathcal{T}(L^e + \ldots)\ldots) = \mathcal{T}(L^e + L^i) = \int_\Omega (L^e + L^i) \cdot f_r \cdot \cos\theta' \, d\omega'$$

where L^i is the indirect illumination computed by the subsequent integration. Monte-Carlo estimates are accurate if the integrand

$$L^r(\omega') = (L^e + L^i) \cdot f_r \cdot \cos\theta'$$

is "flat", i.e. close to constant, otherwise the estimates have high variance. Let us examine this statement formally. Suppose that this integral is evaluated by Monte-Carlo quadrature, thus it is converted to an expected value, which is estimated by an average. Assume that a random direction ω' is sampled from a probability density ω'. The integral to be computed is:

$$\int_\Omega L^r(\omega')\, d\omega' = \int_\Omega \frac{L^r(\omega')}{p(\omega')} \cdot p(\omega')\, d\omega' =$$

$$E\left[\frac{L^r(\omega')}{p(\omega')}\right] \approx \hat{L}^r = \frac{1}{N}\sum_{i=1}^{N} \frac{L^r(\omega'_i)}{p(\omega'_i)}. \qquad (6)$$

Estimator $\hat{L}^r$ is also a random variable whose standard deviation is $\sigma/\sqrt{N}$ where σ^2 is

$$\sigma^2 = \int_\Omega \left(\frac{L^r(\omega')}{p(\omega')} - \int_\Omega L^r(\omega)\, d\omega\right)^2 \cdot p(\omega')\, d\omega'. \qquad (7)$$

This standard deviation is small if $L^r(\omega')/p(\omega')$ is close to the value of the integral $\int_\Omega L^r(\omega)\, d\omega$ everywhere in the domain.

One way of reducing the variance of the Monte-Carlo integration is the application of some form of *importance sampling* [7], which means that $p(\omega')$ mimics the integrand $L^r(\omega')$ to make $L^r(\omega')/p(\omega')$ approximately constant. Unfortunately, the integrand of the rendering equation is not available explicitely, thus the probability density is usually based only on the local BRDFs [3, 5] — i.e. it mimics $f_r \cdot \cos\theta'$ instead of $(L^e + L^i) \cdot f_r \cdot \cos\theta'$, which can be quite inaccurate.

Another possibility is a different formulation of the global illumination problem as an integral, where the integrand is significantly flatter. Since the problematic part is the incoming radiance which stems both directly and indirectly from the emission of the lightsources, we aim at replacing the lightsource term by a different function which is flatter. For example, we can replace the emissions of the lightsources by their first reflection, which leads us to the core idea of the first-shot methods.

2 The basic idea of first-shot

First-shot is a preprocessing method that shoots the power of the small lightsources onto other surfaces, increase the emission of the other surfaces by the reflection, then removes the original lightsources from the scene.

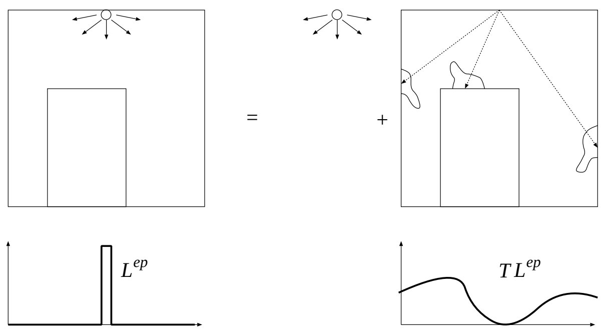


Figure 2: First-shot technique

Formally, the unknown radiance L is decomposed into two terms:

$$L = L^{ep} + L^{np} \qquad (8)$$

where L^{ep} is the emission of the small area and point lightsources, L^{np} is the emission of the larger area lightsources and the reflected radiance. Substituting this into the rendering equation we have:

$$L^{ep} + L^{np} = L^e + \mathcal{T}(L^{ep} + L^{np}). \qquad (9)$$

Expressing L^{np} we obtain:

$$L^{np} = (L^e - L^{ep} + \mathcal{T}L^{ep}) + \mathcal{T}L^{np}. \qquad (10)$$

Introducing the new lightsource term

$$L^{e*} = L^e - L^{ep} + \mathcal{T}L^{ep} \qquad (11)$$

which just replaces the point lightsources (L^{ep}) by their single reflection ($\mathcal{T}L^{ep}$), the equation for L^{np} is similar to the original rendering equation:

$$L^{np} = L^{e*} + \mathcal{T}L^{np}. \qquad (12)$$

Note that when this equation is solved, integrand

$$L^{r*}(\omega') = (L^{e*} + L^{i*}) \cdot f_r \cdot \cos\theta'$$

is flatter than the original integrand.

Summarizing, first-shot computes the direct illumination caused by the small lightsources, then removes these lightsources from the scene during global illumination calculation, and adds them again at the end of the computation.

The reflection of the small lightsources can be computed in a preprocessing phase of the global illumination algorithm, or simultaneously with the global illumination algorithm when it is needed. Furthermore, it is also possible to do some parts of the calculation in the preprocessing phase while completing the computation on-the-fly with the global illumination algorithm.

We can consider the following alternatives:

1. *Classical first-shot*: The reflected radiance is computed completely in the preprocessing phase. This method [2] works well in the radiosity setting, since in this case, the representation of the reflected radiance requires a diffuse "emission" in each patch, thus the memory overhead of the first-shot is just one variable per patch. However, in non-diffuse scenes the classical first-shot has prohibitive memory requirements, since even if the original light-sources are diffuse, their reflection may have general directional function, which requires the representation of the complete reflected, non-diffuse radiance function. If the directional variation of the radiance is represented by n basis functions (i.e. n is the number of small solid angles in which the radiance can be supposed to be constant) in each patch, then the method requires n new variables for each patch.

2. *Diffuse first-shot*: The BRDF, the light-transport operator, and the reflected radiance are decomposed into diffuse and non-diffuse components and the previous first-shot is applied only to the diffuse reflected radiance. This method can be used in those finite-element, non-diffuse global illumination algorithms which can make a distinction between the first and the other bounces of the light.

3. *Incoming first-shot*: The incoming radiance is computed in the preprocessing phase and the reflected radiance is obtained from the incoming radiance on the fly. Since the surfaces can also be non-diffuse, the incoming radiance received by the patches from each point lightsource should be stored (this requires l additional variables per patch, where l is the number of point samples of the lightsources) [10]. The secondary, non-diffuse emission to a direction is computed from these irradiances. The method is feasible if l is small, which is the case if the scene contains a few point lightsources and small area lightsources whose contribution can be accurately evaluated.

4. *On-the-fly direct lightsource computation*: Everything is done simultaneously to the global illumination algorithm. This approach requires neither preprocessing nor storage but is slower than the previous methods. finite-element tesselation. When L^{e*} is needed in a point, shadow rays are traced towards the real lightsources and L^{e*} is computed as their reflection.

This paper discusses the incoming first-shot method and its application in a stochastic iteration algorithm.

3 Incoming first-shot of point light-sources

Suppose that the scene contains l point lightsources at locations $\vec{y}_1, \ldots \vec{y}_l$ with powers $\Phi_1, \ldots, \Phi_l$, respectively, then their reflection at point $\vec{x}$ is:

$$(\mathcal{T}L^{ep})(\vec{x},\omega) = \sum_{i=1}^{l} \frac{\Phi_i \cdot v(\vec{y}_i,\vec{x})}{4\pi|\vec{y}_i - \vec{x}|^2} \cdot f_r(\omega_i', \vec{x}, \omega) \cdot \cos\theta_i', \tag{13}$$

where ω_i' is the direction of lightsource i, θ_i' is the angle between ω_i' and the surface normal, and $v(\vec{y}_i, \vec{x})$ indicates the mutual visibility of $\vec{x}$ and $\vec{y}_i$. Suppose that the patch under consideration is patch j and its area is A_j. The average reflected radiance is:

$$\langle \mathcal{T}L^{ep}\rangle_j(\omega) = \langle \mathcal{T}L^{ep}\rangle_j(\omega) = \frac{1}{A_j}\cdot \int_{A_j} (\mathcal{T}L^{ep})(\vec{x},\omega)\, d\vec{x} =$$

$$\sum_{i=1}^{l} \frac{1}{A_j} \cdot \int_{A_j} \frac{\Phi_i \cdot v(\vec{y}_i,\vec{x})}{4\pi|\vec{y}_i - \vec{x}|^2} \cdot f_r(\omega_i', \vec{x}, \omega) \cdot \cos\theta_i'\, d\vec{x}. \tag{14}$$

To compute the reflection of a lightsource at a point, the visibility of the lightsource from the point must be determined. We can use *shadow ray*s evaluated by ray-shooting, but this is rather slow. Another alternative is to exploit the image synthesis hardware in the following way. The eye is put at the lightsource and the window is defined as one of the faces of a cube placed around the eye. Rendering the images for each faces using constant shading and using the index of the patches as color values, the visible areas of the patches from the lightsource can be determined.

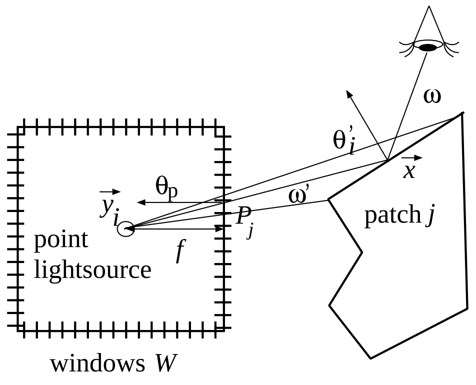


Figure 3: Computation of the lightsource visibility by hardware

The integral in equation (14) can also be evaluated on the six window surfaces (W) that form a cube around the lightsource. To find formal expressions, let us express the solid angle $d\Omega_p$, in which a differential surface area $d\vec{x}$ is seen through pixel area $d\vec{p}$, both from the surface area and from the pixel area:

$$d\Omega_p = \frac{d\vec{x} \cdot \cos\theta'_i}{|\vec{y}_i - \vec{x}|^2} = \frac{d\vec{p} \cdot \cos\theta_p}{|\vec{y}_i - \vec{p}|^2}, \qquad (15)$$

where θ_p is the angle between direction pointing to $\vec{x}$ from $\vec{y}_i$ and the normal of the window (figure 3). The distance $|\vec{y}_i - \vec{p}|$ between pixel point $\vec{p}$ and the lightsource $\vec{y}_i$ equals to $f/\cos\theta_p$ where f is the distance from $\vec{y}_i$ to the window plane, that is also called the *focal distance*. Using this and equation (15), differential area $d\vec{x}$ can be expressed and substituted into equation (14), thus we can obtain:

$$\langle \mathcal{T}L^{ep}\rangle_j(\omega) =$$

$$\sum_{i=1}^{l} \frac{1}{A_j} \cdot \int_W \frac{\Phi_i \cdot v(\vec{y}_i, \vec{x})}{4\pi} \cdot f_r(\omega'_i, \vec{x}, \omega) \cdot \frac{\cos\theta_p^3}{f^2}\, d\vec{p}.$$

Let P_j be the set of pixels in which patch j is visible from the lightsource. P_j is computed by running a z-buffer/constant shading rendering step for each sides of the window surface, assuming that the color of patch j is j, then reading back the “images”. The reflected radiance on patch j is approximated by a discrete sum as follows:

$$\langle \mathcal{T}L^{ep}\rangle_j(\omega) \approx$$

$$\sum_{i=1}^{l} \frac{\Phi_i}{4\pi f^2 A_j} \cdot \sum_{p \in P_j} f_r(\omega'_i, \vec{x}, \omega) \cdot \cos\theta_p^3 \cdot \delta P, \qquad (16)$$

where δP is the area of a single pixel in the image. If R is the resolution of the image — i.e. the top of the hemicube contains $R \times R$ pixels, while the side faces contain $R \times R/2$ pixels – then $\delta P = 4f^2/R^2$. If the BRDF can be assumed to be $\tilde{f}_j(\omega'_i, \omega)$ in patch j, then the reflected radiance can be decomposed into 3 factors: the power spectrum of the lightsource Φ_i, the BRDF $\tilde{f}_j(\omega'_i, \omega)$ which is also a spectrum and is the only factor which depends on viewing direction ω, and a scalar factor:

$$r_{ij} = \frac{1}{\pi R^2 A_j} \cdot \sum_{p \in P_j} \cos\theta_p^3.$$

These scalar factors are computed and stored at each patch, which requires just one float variable per each patch and each point lightsource.

If variables r_{ij} are available, then the incoming first-shot phase is complete. During global illumination when the reflected radiance $\langle \mathcal{T}L^{ep}\rangle_j(\omega)$ is needed at point $\vec{x}$ of patch j, this is computed on the fly from the stored scalar parameter r_{ij}, from the directions pointing from $\vec{x}$ to the lightsources and from the power of the lightsources:

$$\langle \mathcal{T}L^{ep}\rangle_j(\omega) = \sum_{i=1}^{l} \Phi_i \cdot r_{ij} \cdot f_r^j(\omega'_i, \omega). \qquad (17)$$

4 Small area light-sources

Now let us discuss the computation of a single reflection of the light coming from a small area lightsource S of emission $L^e(\vec{y}, \omega)$ to a point $\vec{x}$. The reflection at point $\vec{x}$ is

$$(\mathcal{T}L^{ep})(\vec{x}, \omega) =$$

$$\int_{\Omega_S} L^e(h(\vec{x}, -\omega'), \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'\, d\omega' =$$

$$\int_S \frac{L^e(\vec{y}, \omega') \cdot \cos\theta \cdot v(\vec{y}, \vec{x})}{|\vec{y} - \vec{x}|^2} \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'\, d\vec{y}, \qquad (18)$$

where Ω_S is the solid angle in which lightsource S is visible, $\vec{y}$ is a running point on the lightsource and θ is the angle between ω' and the surface normal of the lightsource at $\vec{y}$.

The average reflected radiance of patch j is

$$\langle \mathcal{T}L^{ep}\rangle_j(\omega) = \frac{1}{A_j} \cdot \int_{A_j} (\mathcal{T}L^{ep})(\vec{x}, \omega)\, d\vec{x} =$$

$$\int_S \frac{1}{A_j} \cdot \int_{A_j} \frac{L^e(\vec{y}, \omega') \cdot \cos\theta \cdot v(\vec{y}, \vec{x})}{|\vec{y} - \vec{x}|^2} \cdot \cos\theta' \cdot f_r(\omega', \vec{x}, \omega)\, d\vec{x}\, d\vec{y}, \qquad (19)$$

The outer integral is estimated by trapezoidal rule. It means that the lightsource area is tesselated to triangles (or quadrilaterals). The integrand is evaluated at the common vertices and is assumed to be linear between the vertices. If the number of vertices is l, then the quadrature rule is:

$$\langle \mathcal{T}L^{ep}\rangle_j(\omega) \approx$$

$$\sum_{i=1}^{l} \frac{1}{A_j} \cdot \int_{A_j} \frac{L^e(\vec{y}_i, \omega'_i) \cdot \cos\theta_i \cdot S_{ti} \cdot v(\vec{y}_i, \vec{x})}{3|\vec{y}_i - \vec{x}|^2} \cdot \cos\theta'_i \cdot f_r(\omega'_i, \vec{x}, \omega)\, d\vec{x},$$

where S_{ti} is the total area of the lightsource triangles that share vertex i and factor $1/3$ comes from the fact that a triangle has 3 vertices.

Note that the inner integral is the same as the integral in equation (14), with the substitution

$$\frac{\Phi_i}{4\pi} \Leftarrow L^e(\vec{y}_i, \omega'_i) \cdot \cos\theta_i \cdot \frac{S_{ti}}{3}.$$

There is another slight difference in the window surface. A one-sided area lightsource can emit light into that half-space which is “above” the plane of lightsource. Thus the window surface becomes a *hemicube* (figure 4). An even better window surface is the *cubic tetrahedron* [1], since it has just 3 faces while the hemicube has 5.

Summarizing the incoming first-shot from a small area lightsource consists of the following steps. First the lightsource is decomposed into a triangle mesh. A hemicube or a cubic tetrahedra is placed at each vertex $\vec{y}_i$ of the

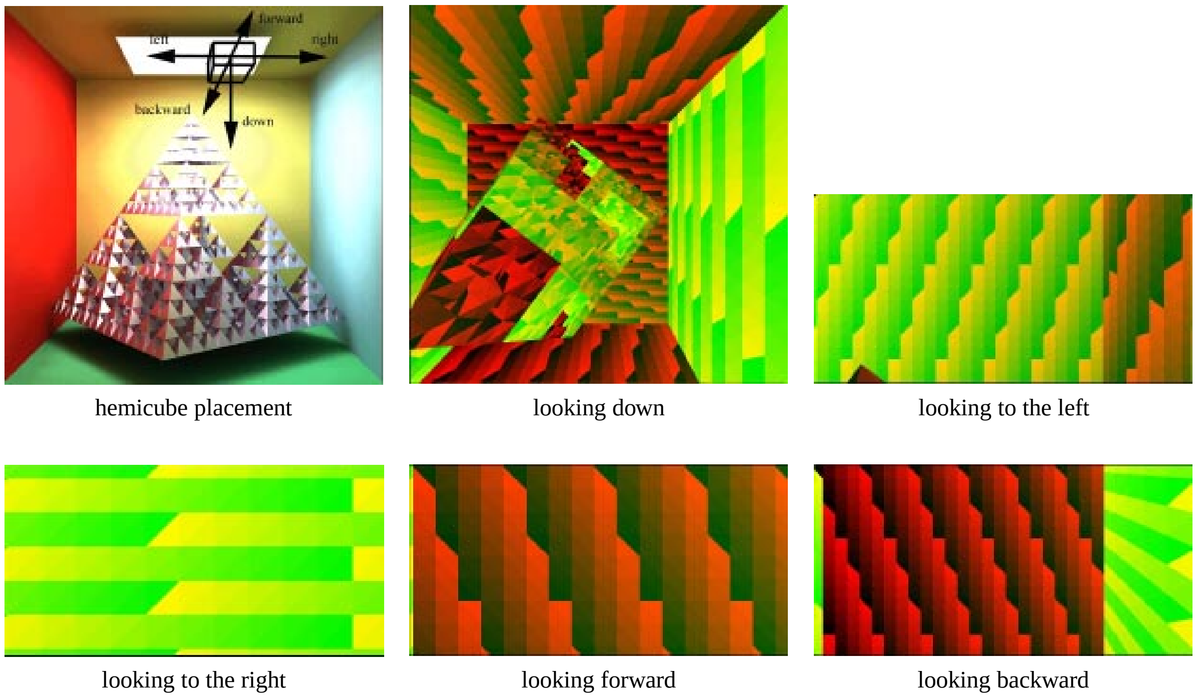


Figure 4: Placement of the hemicube around a lightsource point and the images on the 5 hemicube faces

mesh and the visibility of the other surfaces are determined. Scalar factors

$$r_{ij} = \frac{4S_{ti} \cdot \cos\theta_i}{3R^2 A_j} \cdot \sum_{p \in P_j} \cos\theta_p^3$$

are stored in each patch.

The reflected radiance can be obtained from this scalar factor during the global illumination computation in the following way:

$$\langle \mathcal{T} L^{ep} \rangle_j(\omega) = \sum_i L^e(\vec{y}_i, \omega_i') \cdot r_{ij} \cdot f_r^j(\omega_i', \omega). \tag{20}$$

5 Application of the incoming first- shot to ray-bundle based stochastic iteration

In this section the incoming first-shot technique is applied to stochastic iteration.

The ray-bundle based stochastic iteration [9] works as follows. At each step of the iteration a uniformly distributed random global direction is sampled, and the radiances of all patches in the scene are transferred into this direction. Having computed the transfer, each patch may have some incoming radiance depending what is seen in selected direction. This incoming direction is reflected towards the eye, which results in an image estimate. The average of image estimates of subsequent iteration steps will provide the final result. Note that in the next iteration step, when the radiance is transfered again in the new direction, the radiance is obtained from the incoming radiance of the previous transfer. Thus the method requires just one variable per patch which stores the incoming radiance of the previous iteration step.

The combination of this method with the proposed incoming first-shot techniques is quite straightforward. At a given iteration step not only the incoming radiance of the previous transfer is reflected towards to new direction but also the illumination of the lightsources that are associated with the given patch. Thus the overhead is just l BRDF computations per each patch at each iteration, where l is the number of those point lightsources and vertices of the area lightsources which are visible from the patch.

6 Simulation results

The presented algorithms have been implemented in C++ in OpenGL environment. The running times have been measured on a PC with 400 MHz Pentium II processor.

The scene of figure 7 contains a 3D Sierpiensky set and has 22768 patches. The diffuse albedo of the patches in this set is $(0.18, 0.06, 0.12)$ on the wavelengths 400 nm, 552 nm and on 700 nm, respectively. The specular albedo is wavelength independent and is between 0.8 and 0.4 depending on the viewing angle. The non-diffuse reflection was modelled by the physically plausible stretched Phong model [6]. The “shine” parameter is 3.

Figure 6 compares the speed of the convergence of stochastic iteration with and without the proposed incoming first-shot step. In figure 7 the timing and the image quality of the two methods can also be compared. For the first-shot, the area lightsource has been subdivided into a mesh of 8 triangles and 9 vertices. The incoming first-shot phase took 55 seconds, which were needed by the 9×5 z-buffer/constant-shading rendering steps. A single radiance transfer by a ray-bundle took 1.5 seconds without the first-shot results and 2 seconds when the incoming first-shot was also used. The 0.5 second overhead is due to the reflection of the result stored by the incoming first-shot both towards the eye and towards to next global direction.

However, we can conclude that incoming first-shot is worth for this small extra time, since the resulting algorithm converges very quickly, and the image is almost fully converged after 2.5 minutes. Comparing the error curves, we can see that the stochastic iteration is about $10 - 20$ times faster with the incoming first-shot than without it.

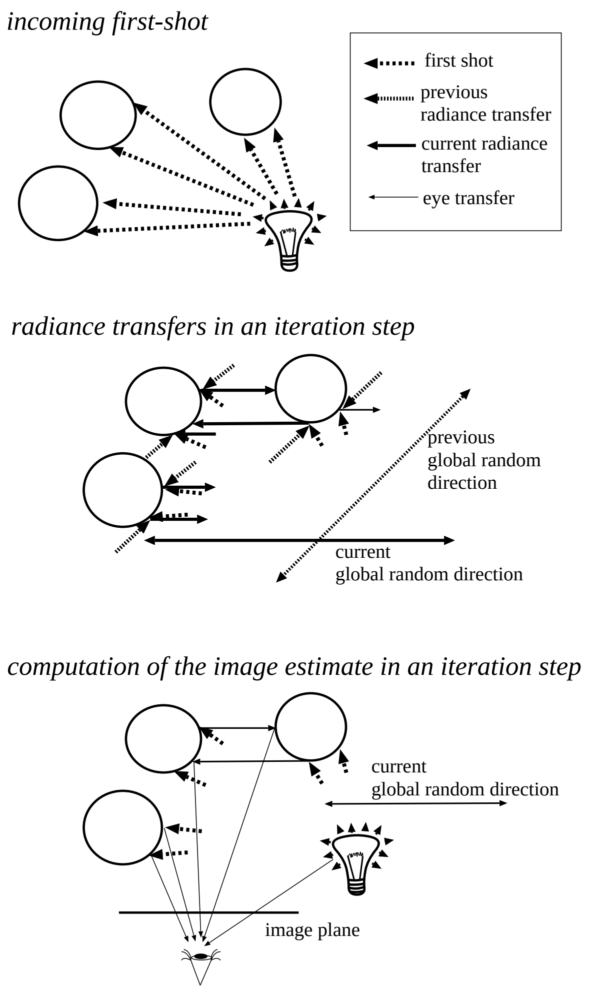

incoming first-shot
first shot
previous radiance transfer
current radiance transfer
eye transfer
radiance transfers in an iteration step
previous global random direction
current global random direction
computation of the image estimate in an iteration step
current global random direction
image plane

Figure 5: Ray-bundle stochastic iteration with incoming first-shot

7 Conclusions

This paper has presented a preprocessing method which replaced the emission of small and point lightsources by their reflection. This replacement makes the integrand significantly flatter, which improves most of the global illumination algorithm. The incoming first-shot method has also been combined with ray-bundle based stochastic iteration. With this combination, the non-diffuse global illumination solution of quite complex scenes becomes possible interactively (within a few minutes).

References

[1] J. C. Beran-Koehn and M. J. Pavicic. A cubic tetrahedral adaptation of the hemicube algorithm. In James Arvo, editor, *Graphics Gems II*, pages 299–302. Academic Press, Boston, 1991.

[2] F. Castro, R. Martinez, and M. Sbert. Quasi Monte-Carlo and extended first-shot improvements to the multi-path method. In *Spring Conference on Computer Graphics '98*, pages 91–102, 1998.

[3] Ph. Dutre, E. Lafortune, and Y. D. Willems. Monte Carlo light tracing with direct computation of pixel intensities. In *Compugraphics '93*, pages 128–137, Alvor, 1993.

[4] J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 143–150, 1986.

[5] E. Lafortune and Y. D. Willems. Bi-directional path-tracing. In *Compugraphics '93*, pages 145–153, Alvor, 1993.

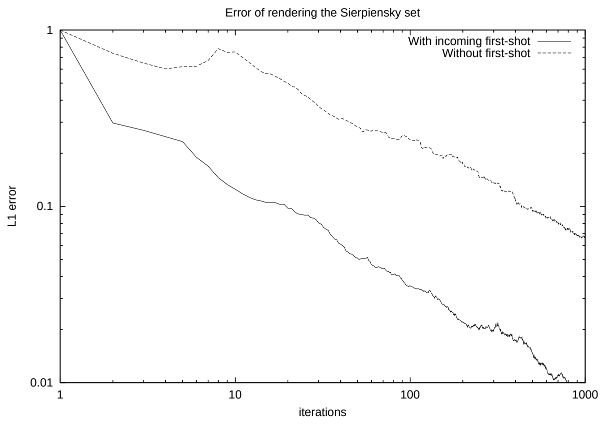
[6] L. Neumann, A. Neumann, and L. Szirmay-Kalos. Compact metallic reflectance models. *Computer Graphics Forum (Eurographics'99)*, 18(3):161–172, 1999.

[7] I. Sobol. *Die Monte-Carlo Methode*. Deutscher Verlag der Wissenschaften, 1991.

[8] L. Szirmay-Kalos. Monte-Carlo methods for global illumination. In *Spring Conference of Computer Graphics '99*, pages 1–28, Budmerice, 1999. invited talk.

[9] L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum (Eurographics'99)*, 18(3):233–244, 1999.

[10] L. Szirmay-Kalos and W. Purgathofer. Global ray-bundle tracing with hardware acceleration. In *Rendering Techniques '98*, pages 247–258, 1998.



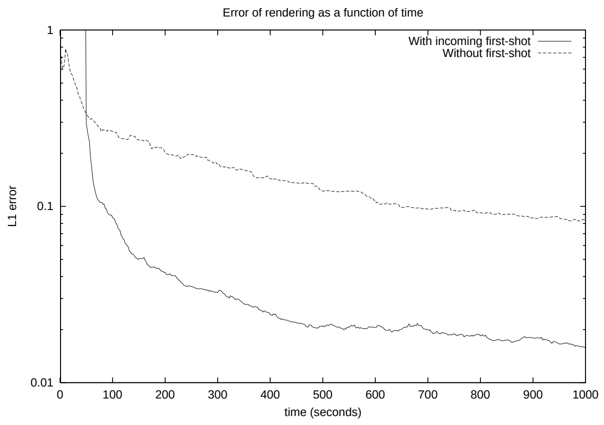
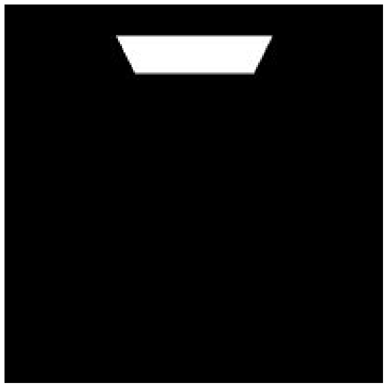
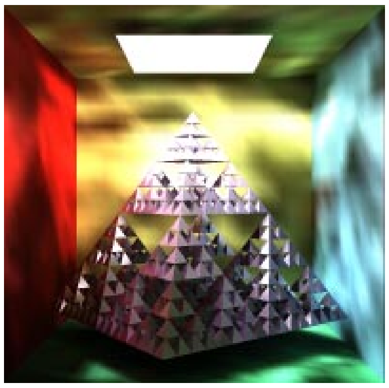
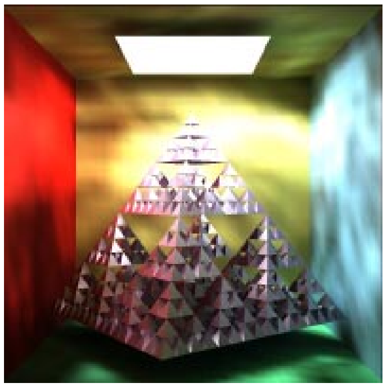
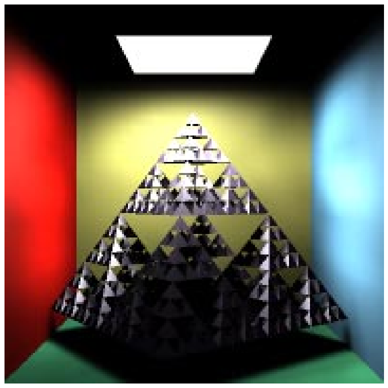


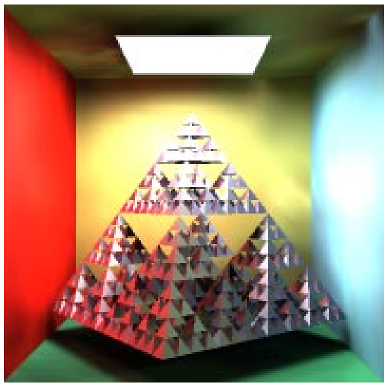
Figure 6: Error of ray-bundle stochastic iteration with and without incoming first-shot


0 iteration, 0 secs


47 iterations, 70 secs


100 iterations, 150 secs


first-shot + 0 iteration, 50 secs


first-shot + 10 iterations, 70 secs

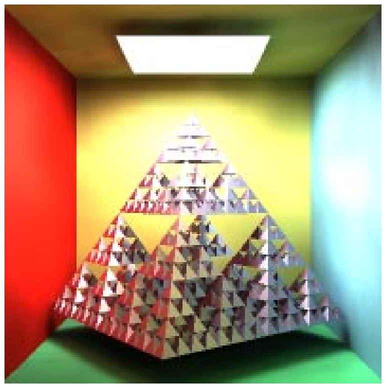

first-shot + 50 iterations, 150 secs

Figure 7: Comparison of stochastic iteration without (upper-row) and with incoming first-shot (lower-row)