Scene Independent Real-Time Indirect Illumination

Jeppe Revall Frisvad* Rasmus Revall Frisvad† Niels Jørgen Christensen‡ Peter Falster§

Informatics and Mathematical Modelling
Technical University of Denmark

Abstract

A novel method for real-time simulation of indirect illumination is presented in this paper. The method, which we call Direct Radiance Mapping (DRM), is based on basal radiance calculations and does not impose any restrictions on scene geometry or dynamics. This makes the method tractable for real-time rendering of arbitrary dynamic environments and for interactive preview of feature animations. Through DRM we simulate two diffuse reflections of light, but can also, in combination with traditional real-time methods for specular reflections, simulate more complex light paths. DRM is a GPU-based method, which can draw further advantages from upcoming GPU functionalities. The method has been tested for moderately sized scenes with close to real-time frame rates and it scales with interactive frame rates for more complex scenes.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: direct radiance mapping, global illumination, hardware acceleration, programmable graphics hardware, real-time rendering

1 Introduction

A difficult step along the path towards real-time global illumination is to model indirect light which is reflected diffusely.

In this paper we introduce a method for real-time simulation of indirect illumination. We call the method Direct Radiance Mapping (DRM) and it models two diffuse reflections of light, that is, a single diffuse bounce of indirect light. In order to simulate more complex light paths DRM can be combined with traditional real-time methods for specular reflections.

Recently PDI/DreamWorks have successfully incorporated a single bounce of indirect light in a micro-polygon based scan line renderer [26]. Since we simulate the same part of the global illumination problem, DRM may be useful for a real-time preview before an expensive micro-polygon based rendering. The idea of a real-time method approximating a single bounce of indirect light is especially motivated by figure 6 in [26], which shows that the first bounce is indeed most visually significant.

Another area of application is computer games. Games often make use of Light Mapping which provides a static global illumination of the background scenery. DRM makes dynamic diffuse indirect illumination available for real-time applications such as games. Even though DRM does not offer full global illumination, it is one step along the way and the implementation is easily accomplished using fragment programs.

*e-mail: jrf@imm.dtu.dk
†e-mail: rasmus@revall.dk
‡e-mail: njc@imm.dtu.dk
§e-mail: pfa@imm.dtu.dk

2 Related Work

An extensive survey of techniques for interactive global illumination is given in [3]. Most of these techniques handle only few scene changes. An example could be that only the camera and a few rigid objects are allowed to move, while light sources, surface attributes, object shapes, etc. are static. Such interactive methods usually resort to interactive refinement of traditional global illumination solutions or rely on parallel computations using multiple processors. To the contrary the method presented in this paper imposes no restrictions on scene changes, therefore we call our method scene independent, and it recomputes the entire solution for each frame using a single CPU in cooperation with a GPU.

Light Mapping, originating in [22], is an approach which imposes a minimum of restrictions on scene changes. The tradeoff is, as previously mentioned, that Light Mapping provides only static global illumination. A disadvantage of Light Mapping and real-time or interactive Radiosity approaches (such as those reviewed in [3]) is that they depend on mapping of textures onto objects. This complicates shape animation and such methods are, therefore, not scene independent.

Other approaches to global illumination on a single CPU cooperating with a GPU includes methods such as Precomputed Radiance Transfer [25, 24] and Cube Map Rendering [14, 18], where radiance transfer functions are computed and represented in spherical harmonics. The former relies on heavy precomputation of transfer functions for individual objects, which results in support of rigid objects only and limitations when several objects start interacting with their surroundings. The latter recomputes the spherical harmonics coefficients each frame by rendering cube maps at regularly spaced points throughout the scene. The placing of the cube maps and the following interpolation of the spherical harmonics coefficients may give problems in oddly shaped scenes. Both methods assume low-frequency lighting environments since they represent their transfer functions in a spherical harmonics basis as proposed previously in [23].

Another approach is Real-Time Simulation of Photon Mapping using the GPU. The method described in [20, 4] cannot afford a final gathering step and uses a filtered direct evaluation of a global photon map instead. This results in significant low-frequency noise and the frame rates are hardly real-time. A method for real-time photon mapping, which includes a final gathering step, was presented in [13]. This method is, however, dependent on scene changes, since appropriate points must be chosen on the surfaces in the scene. Additionally a ray tracer is used for selective photon tracing and therefore each dynamic object must be pointed out for efficient BSP (Binary Space Partitioning) tree optimized ray tracing. The use of precalculated BSP trees means that all objects must be rigid.

Through DRM we seek a method which needs no time consuming precomputations, no restrictions on scene dynamics such as rigid objects only, and no placement of sample points throughout the scene requiring one or several rendering passes each. In other words we seek a method which is independent of scene dynamics and scene geometry while still able to simulate some of the light paths encompassed in global illumination.

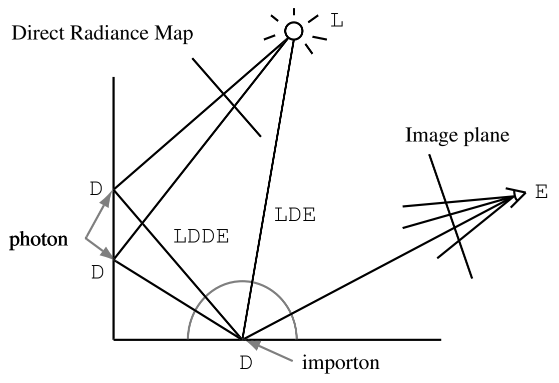

Figure 1: An illustration of the conceptual idea.

3 The Concept

Think of the final gathering step employed in most multi-pass methods for global illumination. Through a mapping of the radiance arriving directly from the light sources to the surfaces in a scene, we can perform a simple final gathering using rasterization only. This simple final gathering captures a smooth approximation to single bounce indirect illumination.

Explained differently the idea is on one hand to have a photon map (see eg. [10, 9]), containing photons emitted directly from the light sources only, and on the other hand an importon[1] map, containing importons emitted directly from the observer only. In light transport notation (introduced by Heckbert in [8]) the photon map we describe will represent the light paths `LD` while the importon map will represent the light paths `DE`.

Suppose we let each direct importon receive a contribution from each direct photon according to a suitable BRDF, then we have performed the final gathering and accounted for the light paths `LDDE`. In order to avoid tracing a ray from each photon to each importon, we must assume visibility between all photons and all importons. This means that our method does not account for indirect shadows.

Instead of ray tracing we employ a rasterization approach to construct the photon map and the importon map. The data structure of a photon essentially contains a position, an RGB power value, and an incident direction (see [9]). Instead of photon (or importon) objects, each containing the aforementioned data fields, we might as well consider a texture holding positions of the direct photons, a texture holding the power of the direct photons, and the position in space from where the photons were emitted (this is sufficient since we consider direct photons only).

When taking a picture from the observer, each fragment of that picture provides sufficient information to represent an importon. If we supply a fragment program with the textures mentioned above or samples from the textures uploaded as uniform parameters, it becomes possible for each direct importon (that is, each fragment) to receive a contribution from each direct photon. Figure 1 illustrates the concept.

Note that no textures are mapped onto objects in the scene. Instead the textures we use comprise a 'direct radiance map' (see fig. 6), which maps a pair of coordinates $(s, t) \in [0, 1]^2$ to the different data fields describing a photon. This map is used merely as an alternative way to provide sufficient information for the simple final gathering. Picking regularly spaced coordinate pairs (s, t) gives direct photons spaced regularly throughout the scene.

[1]The term 'importon' was coined by Peter and Pietrik in [19] to denote photons emitted from the observer. They also describe an importance map, a notion fitting better the dual of a photon map is perhaps an importon map as we call it.

4 The Resulting Method

Combining our method with stenciled planar reflections (see [12]) and cube environment mapping (see [1]) as described in [17], we will in the following describe how to model the light paths $\texttt{LD?S}_{rt}\texttt{*E}$ and $\texttt{LS}_{rt}\texttt{*DDS}_{rt}\texttt{*E}$ in real-time, where $\texttt{S}_{rt}$ denotes reflections under the usual limitations of the real-time reflection methods mentioned above and `?` denotes one or zero events.

To model the mentioned light paths we take a brief look at the rendering equation, which was introduced in [11] and is given as follows:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + L_r(\boldsymbol{x}, \boldsymbol{\omega}) \tag{1}$$

where L_o is *o*utgoing (or exitant) radiance at the surface location $\boldsymbol{x}$ in the direction $\boldsymbol{\omega}$, L_e is *e*mitted radiance, and L_r is *r*eflected radiance. While L_e is often merely modeled as a constant, the reflected radiance (L_r) is an infinitely recursive term, which is given as follows:

$$L_r(\boldsymbol{x}, \boldsymbol{\omega}) = \int_\Omega f_r(\boldsymbol{x}, \boldsymbol{\omega}', \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega}') \cos\theta \, \mathrm{d}\omega' \tag{2}$$

where L_i is *i*ncident radiance at the surface location $\boldsymbol{x}$ from the direction $\boldsymbol{\omega}'$ and Ω is the hemisphere, which we integrate all incoming directions across. θ is the angle between the normal $\boldsymbol{n}$ at the surface location $\boldsymbol{x}$ and the direction towards the incoming light. f_r is the BRDF as described in [15].

In [9] it is shown how the rendering equation can be split up into four different terms: Direct illumination, specular and glossy reflections, caustics, and multiple diffuse reflections. In this paper we partly simulate the direct illumination term, the reflections term, and the multiple diffuse reflections. The caustics term is ignored. In other words DRM compromises the image quality and gain flexibility and speed as compared to the methods mentioned in section 2.

Direct Illumination

To simulate direct illumination (light paths `LD?E`) we take two different cases into account. (a) Direct illumination from isotropic point light sources and (b) direct illumination from isotropic area light sources assuming that the area light is very distant from the objects that it illuminates. In the following we let the subscript 0 denote case (a) described above and we let 1 denote case (b). We also assume invariance of radiance along straight paths (which is in fact the case in a vacuum, see eg. [5]) and we omit function arguments for simplicity of notation.

Since a point light source is a hypothetical invention, which has no areal extension, the emitted radiance term $L_{e,0}$ of the rendering equation in case (a) is non-existent. Therefore $L_{o,0} = L_{r,0}$. Since light intensity ($I = \mathrm{d}\boldsymbol{\Phi}/\mathrm{d}\omega$) is flux dependent solely on a differential solid angle $\mathrm{d}\omega$ describing a directional volume, we can define an isotropic point light source as a point in space emitting a constant light intensity, $I_{e,0}$, in all directions.

Suppose the point light source emits a constant flux $\boldsymbol{\Phi}_s$ in all directions, then by a simple integration over the unit sphere we obtain the result that:

$$I_{e,0} = \frac{\boldsymbol{\Phi}_s}{4\pi} \tag{3}$$

In [6] the following relation between intensity I and irradiance $E = \mathrm{d}\boldsymbol{\Phi}/\mathrm{d}A$ is derived:

$$E = I \frac{\cos\theta}{r^2} \tag{4}$$

where r is the distance between the point of the intensity I and the surface location of the irradiance E and θ is the angle between the direction towards the point of I and the normal at the surface location of E.

Inserting (3) in (4) we obtain the irradiance incident on the surfaces surrounding an isotropic point light source:

$$E_{i,0} = I_{e,0}\frac{\cos\theta}{r^2} = \frac{\mathbf{\Phi}_s}{4\pi}\frac{\cos\theta}{r^2}$$

where it is assumed that no occluding objects lie between the point light and the surface location of the incident irradiance. To include visibility in our calculations a simple visibility term V is introduced. V is one if the point of emission and the surface location of incidence are mutually visible and zero otherwise.

In [15] it is shown that:

$$f_r = \frac{\mathrm{d}L_r}{\mathrm{d}E_i}$$

and, since we consider direct illumination only, an integration gives the following exitant radiance resulting from isotropic point light sources only:

$$L_{o,0} = L_{r,0} = \sum_{j=1}^{N_0} f_r \frac{\mathbf{\Phi}_{s,j}\cos\theta_j}{4\pi r_j^2} V_j \tag{5}$$

where N_0 is the number of point light sources.

As derived in [5] we can find a formula for the incident radiance $L_{i,1}$ resulting from the constant radiance $L_{e,1}$ emitted from an isotropic area light source:

$$L_{i,1} = L_{e,1} = \frac{\mathbf{\Phi}_s}{\pi A} \tag{6}$$

where A is the area of the light source.

In order to calculate the integral of the reflected radiance (2) resulting from light emitted from an area, we can use the following relation between a differential area and a differential solid angle, which is derived eg. in [16]:

$$\mathrm{d}\omega' = \frac{\cos\theta'\mathrm{d}A}{r^2} \tag{7}$$

where θ' is the angle between the normal at the surface location of light emission and the direction towards the surface location of incident light, and r is the distance between the two locations. If we let S denote the union of all surface areas, (7) results in the following area formulation of (2):

$$L_r = \int_S f_r L_i \frac{\cos\theta\cos\theta'}{r^2} V\,\mathrm{d}A \tag{8}$$

Considering the direct illumination resulting from isotropic area light sources assumed to be very distant, we obtain the following:

$$L_{o,1} = L_{e,1} + \sum_{j=1}^{N_1} f_r \frac{\mathbf{\Phi}_{s,j}\cos\theta_j\cos\theta'_j}{\pi r_j^2} V_j \tag{9}$$

which is comparable to (5).

To include shadows in our direct illumination term, that is, to estimate the visibility term V, we employ a version of the shadow mapping method originally proposed in [27].

The direct illumination can, now, easily be typed into a fragment program using a suitable BRDF (it could be the BRDF for perfectly diffuse surfaces, $f_r = \rho_d/\pi$, the modified Blinn-Phong model, or more advanced models) inserted in (5) and (9).

Illumination Reflected Diffusely Twice

In order to include a single diffuse bounce of indirect illumination we assume as mentioned previously that the direct photons and the direct importons are mutually visible and therefore (according to the invariance of radiance along straight paths) that

$$L_{i,\text{importon}}(\boldsymbol{x}, \boldsymbol{\omega}') = L_{o,\text{photon}}(\boldsymbol{y}, -\boldsymbol{\omega}')$$

which implies that the indirect radiance reflected diffusely twice can be calculated by integration over $L_{i,\text{importon}}$ according to (2).

To construct the textures for the direct radiance map, we take pictures from the light source (as done in shadow mapping).

To find the position of each direct photon we make a simple vertex program (or fragment program) that scales the position of each vertex (in world coordinates) to the interval $[0, 1]$. This is accomplished using an AABB (Axis Aligned Bounding Box) of the scene to be rendered. The scaled vertex position is then used as the color of the vertex (or fragment) when a picture is taken from the light source. This results in a texture holding positions of the direct photons (cf. fig. 6a). If floating point buffers are available the scaling is not necessary.

Instead of the power of the photon, which is stored in the traditional photon map, we rather need sufficient information to solve (5) and (9). In other words we need a mapping of the directly reflected radiance, hence the name of the method: *Direct Radiance Mapping*. If the scene contains Lambertian surfaces only, the exitant radiance from each photon will be the same in all directions. In that case a picture is taken from the light source using (5) and/or (9) for calculation of the exitant direct radiance at each photon position. This picture constitutes the second texture needed for the calculation of indirect illumination (cf. fig. 6b).

If the scene contains non-Lambertian surfaces, textures holding sufficient information for calculation of the directly reflected radiance should be constructed. Usually two textures: (a) a texture picturing the normals encountered at each photon position (cf. fig. 6c) and (b) a texture picturing the true color of the surface where the photon is stored, will replace the texture storing exitant radiance (fig. 6b) from the case of Lambertian surfaces only. We consider all the textures needed (positions, normals, true color, or the like) to be a part of the direct radiance map.

Having the direct radiance map available in a fragment program (that is, having the mentioned textures available or samples from the textures uploaded as uniform parameters) enables us to approximate L_i in (2). In the simple case L_i is given for each direct photon as a texture look-up. In order to establish $\cos\theta$ in (2) another texture look-up is needed (using the same texture coordinates), which retrieves the position of the photon from where the incident radiance came. To approximate the integration we sum the contributions from an arbitrary number of photons, M:

$$L_{r,\text{importon}} \approx \sum_{j=1}^{M} f_r L_{o,\text{photon},j}\cos\theta_j\,\Delta\omega'_j \tag{10}$$

the difficult, and hitherto unaddressed, part of (10) is the approximation of the differential solid angle $\Delta\omega'_j$, which is the solid angle subtended by the surface area that photon j represents. In the following the subscript j will denote that the variable is different for each photon and the subscript k will denote that the variable is different for each importon (meaning that the variable must be recalculated for each fragment).

The magnitude of a solid angle is given as the ratio of the spherical-surface area A_s, which the solid angle intercepts, to the square radius of the sphere r^2, see eg. [16]:

$$\omega = \frac{A_s}{r^2}$$

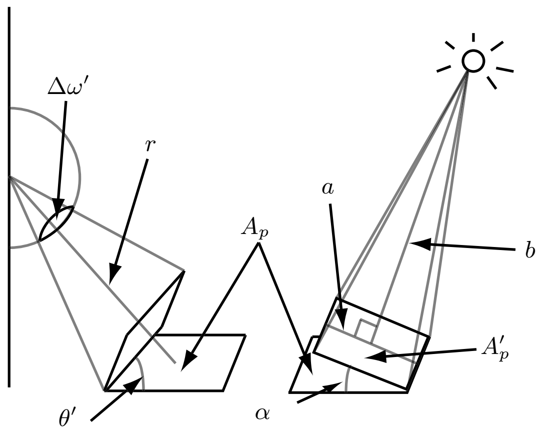

Figure 2: The area A_p pictured by a texel (representing a photon).

First we can assume that the radiance is incident on a locally flat surface area and that the incident radiance, therefore, at most covers the unit hemisphere, ($\Delta\omega'_{\max} = 2\pi$). The simplest approximation of $\Delta\omega'$ is then to split up the unit hemisphere into M equally sized solid angles (one for each photon):

$$\Delta\omega' \approx \frac{2\pi}{M} \tag{11}$$

Since a photon is practically stored as *texels* at the same position in a few different low-resolution textures (found as rasterized pictures taken from the light source), we can more correctly estimate the surface area A_p that a single photon/texel represents. Consider the case sketched in figure 2. First we can calculate the area A'_p which is perpendicular to the direction towards the light source:

$$\begin{aligned} a_j &= b_j \tan\left(\frac{\gamma}{2w}\right) \\ A'_{p,j} &= (2a_j)^2 \end{aligned} \tag{12}$$

where w is the width of the textures measured in number of texels and b is the distance between the photon and the light source. For the calculation of A_p we can assume that the frustum of texel j is constrained by nearly parallel planes (since the light source was assumed to be very distant), this assumption leads to the following:

$$A_{p,j} \approx \frac{A'_{p,j}}{\cos\alpha_j} \tag{13}$$

where α is the angle between the incoming direction of the photon and the normal at the surface location where the photon impinges.

Having a surface area, such as A_p, the usual approximation to the solid angle is a simple projection to the tangent plane of the unit sphere as given in (7):

$$\Delta\omega'_{j,k} \approx \frac{\cos\theta'_{j,k} A_{p,j}}{r^2_{j,k}}$$

Considering a differential solid angle this expression is exact at the limit, but in an approximation we must take Lambert's "five times rule of thumb" into account. In [21] this is done using the following approximation instead:

$$\Delta\omega'_{j,k} \approx \cos\theta'_{j,k} \frac{A_{p,j}}{A_{p,j} + r^2_{j,k}} \tag{14}$$

To evaluate (14) two additional cosine terms ($\cos\alpha_j$ in (13) and $\cos\theta'_{j,k}$ in (14)) must be calculated meaning that the normals encountered by the photons at the surface areas must be pictured from the light source and stored in a texture as a part of the direct radiance map. Note that the tangent term in (12) does not need to be calculated in the fragment program.

To avoid the two extra cosine terms (and the picture of the normals) an inexpensive compromise between (11) and (14) is simply to say that $A'_p \approx A_p$ and the distance between a photon and an importon can be approximated by a constant. In many cases the scene diagonal is an appropriate choice of constant, meaning that if $B_{\max}$ and $B_{\min}$ denotes the maximum and minimum points of the AABB containing the scene, the third possible approximation of $\Delta\omega'$ is as follows:

$$\Delta\omega'_j \approx \frac{A'_{p,j}}{\|B_{\max} - B_{\min}\|^2} \tag{15}$$

In the preceding we have presented three alternative ways of calculating approximate solutions for (10). (11) is the simple, inexpensive, and imprecise solution, (14) is the most correct and also the most expensive approximation presented, and (15) is a compromise.

To include two diffuse reflections of light in a traditional picture from the observer each fragment must evaluate a version of (10). As mentioned in section 3 each fragment conceptually represents an importon, but direct importons always have the importance 1 and therefore:

$$L_o = L_{o,0} + L_{o,1} + L_{r,\text{importon}}$$

which includes both direct light and a single diffuse bounce of indirect light.

Perfectly Specular Reflections

As mentioned previously multiple perfectly specular reflections can be obtained in real-time. To include more than just the light paths LDDE in our indirect illumination term, we can merely include these techniques in our picture capturing direct illumination as seen from the light source (giving $\texttt{LS}_{rt}\texttt{*D}$) and in our final rendering from the observer. The resulting indirect illumination term will include the light paths $\texttt{LS}_{rt}\texttt{*DDS}_{rt}\texttt{*E}$ and the direct illumination term will include the paths $\texttt{LD?S}_{rt}\texttt{*E}$.

Creating the cube environment map for perfectly specular reflections is rather expensive if it also must include the calculation of two diffuse reflections of light. The eye, however, quickly discovers the error if we use only direct illumination for the reflection. One inexpensive way to trick the untrained eye into believing the reflected image with no indirect illumination, is to let the indirect illumination affect the specular object as if it were diffuse. The trick is illustrated in figure 8 and also used for the images in figure 9.

5 Implementation

Summing up the rendering procedure, the following steps are carried out for each frame:

1. For each light source:
   - Generate a depth cube map for shadow mapping.
   - Generate a direct radiance map consisting of
     (a) a picture of positions seen from the light source.
     (b) a picture of direct radiance (or true colors) seen from the light source.
     (c) a picture of normals seen from the light source (if needed).
2. Sample the pictures and bind the sampled values to a fragment program for calculation of indirect illumination. If the number of registers for uniform parameters is insufficient for storage of all the values, bind some of the pictures as textures instead.

3. For each curved specular object; generate a cube environment map for specular reflections. Each face of the cube is rendered as in steps 4, 5 and 6.

4. For each planar specular object; render a mirrored scene where the object should have been using steps 5 and 6.

5. Render the scene to the accumulation buffer using a fragment program calculating (5) to account for point light sources and (9) to account for area light sources. Use the depth cube maps to estimate visibility. Swap the fragment program with one sampling the cube environment map when a curved specular object is rendered.

6. Render the scene again using the fragment program for indirect illumination, where (10) is evaluated using either (11), (14), or (15). Add the result to the accumulation buffer.

7. Return the final result from the accumulation buffer.

The summation needed for evaluation of (10) in step 6 is an expensive process in a fragment program, therefore we can not account for each texel of the direct radiance map in our estimate of $L_{r,\mathrm{importon}}$. The most sensible choice is then to pick out a number of regularly spaced sample photons. Experimenting with the number of samples we found that $M = 2 \times 2$, though better than a constant ambient term, were too few to catch details such as changing shades of color bleeding across a surface. $M = 4 \times 4$ were too many to keep up real-time frame rates and the result, though more detailed, was not markedly better than the final choice of $M = 3 \times 3$ sample photons. It should also be noted that when $M = 3 \times 3$ we reach the limit of uniform parameters that can be uploaded to the fragment program and some calculations must (though it is not necessary in theory) be moved from the CPU to the GPU[2]. When the size of the grid exceeds $M = 5 \times 5$ samples we surpass the limit of the number of instructions allowed in fragment programs available for our current graphics card.

To improve the frame rates only variables having the subscript k in (11), (14), and (15) are calculated in the fragment program. The variables having the subscript j are found using the CPU and uploaded to the fragment program as uniform parameters once for each frame as indicated in step 2. If the CPU is needed for other purposes all texture look-ups and calculations can be performed on the GPU.

The relatively few sample photons have the result that the indirect illumination may flash when some samples suddenly find a different surface. To improve on this issue the texture capturing direct radiance (or true color) can be MIP mapped[3] to a level where the few samples will cover an interpolation of almost the entire texture.

Few sample photons also result in an issue when multiple light sources are needed. At the moment we work with a single light source only. When it is a point light source we give it a spotlight direction in our modeling tool. This provides it with a direction in which the pictures from the light source should be taken. Hence, the spotlight direction decides in which direction from the light source the sample photons should be found.

The bottleneck of the method is the expensive fragment program which accumulates the contributions from the sample photons. This means that the method will improve gradually as fragment programs get faster in general and as better facilities for looping emerges. We also believe that our method could draw advantage from an upcoming GPU functionality (which is currently not available to us) called “multiple render targets”[4].

[2]This is the case if we evaluate (14). It is not necessary in order to evaluate (11) or (15).

[3]MIP mapping originates in [28].

[4]Multiple render targets is, for example, used in deferred shading, see eg. [7]

	a	**b**	**c**
Figure 3	769.231	137.300	48.622
Figure 4	48.622	48.622	20.870
Figure 8	15.178	15.178	9.898
Figure 9	20.761	11.888	7.695

Table 1: Frame rates (frames/second) for the pictures presented in the different figures.

Our rendering method employs no particular vertex processing and the number of triangles that can be rendered in real-time is solely dependent on the number of triangles that the GPU can process. The scalability of the method is substantiated by this fact.

6 Results and Discussion

The Cornell box [2] is a benchmark scene for calculation of diffusely reflected indirect illumination. Figure 3 shows different renderings of a Cornell box. Figure 4 compares the three different methods for calculation of $\Delta\omega'$. Figure 8 shows a scene including perfectly specular surfaces and indirect illumination calculated using (10) and (14). Figure 9 shows the effect of our method in a more complex scene containing 39,956 triangles.

Frame rates of the different pictures using a 1.7 GHz Pentium4 CPU and a GeForce FX 5950 GPU are given in table 1. Resolution is 512×512. The textures used for the direct radiance map are shown in figure 6. We chose a resolution of 16×16 for the textures. The frame rates for the cave scene (fig. 9) are (a) 48.0, (b) 18.4, and (c) 10.3 if the mirrors are not activated. To indicate the scalability of DRM it should be mentioned that a rendering of The Stanford Dragon consisting of 871,414 triangles results in interactive frame rates (0.92 fps using (14) and 1.8 fps using (15)).

The constant approximation of $\Delta\omega'$ given in (11) has the result that the indirect illumination gets overexposed when objects come close to the light source, see figure 7. The reason is that the surface area a photon/texel covers depends on how close the surface is to the light source. One small triangle being sufficiently close to the light source can, for example, cover all the texels in the direct radiance map, therefore it is quite important to account for the area that a texel covers. A constant approximation also has the result that the indirect illumination is more globally affected by the exact spots where the sample photons are chosen.

Both the approximation in (11) and (15) suffers from the fact that they do not account for the normal where the indirect illumination is reflected. The result is that indirect illumination is reflected on the backside of the surface as well as at the front. (15), however, more closely resembles the more correct solution provided by (14) and as shown in table 1 it is not more expensive to evaluate (15) instead of (11).

Figure 5 compares DRM (fig. 5c) with a standard radiosity solution (fig. 5a) and a direct visualization of a photon map (fig. 5b). To emphasize that it is the quality of the indirect illumination we wish to compare, the shadows are hard in the photon mapping reference. The overall indirect illumination is, not surprisingly, more crude in the DRM approximation. The different shades of color bleeding are considerably more mixed across the floor, ceiling, and back wall. Missing indirect shadows result in too bright shadow regions. The bright spot above the tall box, which shows in the photon mapping reference is not visible, since the samples are too few. Because of Gouraud shading, the spot does not show in the radiosity reference either. The color bleeding captured by DRM is best at the tall box. The reason why the tall box is more green at the front using DRM, is that the area light source does not contribute to the surface. This happens since in DRM all light is emitted from the center of the light source, while the reference methods distribute the emission over the entire surface. Nevertheless we find that DRM offers a reasonable approximation considering that this image was rendered thousands of times faster than the references.

7 Conclusion

In this paper we have shown how basal radiance calculations can lead to a GPU based method for simulation of indirect illumination reflected diffusely twice. We also include traditional real-time techniques for perfectly specular reflections in our solution and are thereby able to account for the light paths $\texttt{LS}_{rt}\texttt{*DDS}_{rt}\texttt{*E}$ and $\texttt{LD?S}_{rt}\texttt{*E}$. The method presented relies on pictures taken from the light source(s) and is called Direct Radiance Mapping (DRM).

DRM is independent of scene changes both with respect to geometry and dynamics. This means that the particular shape, movement, and deformation of the scene contents is of no consequence to the method. The number of light sources is the limiting factor and too few samples can result in sudden shifts in the indirect illumination. With MIP mapping and improved efficiency of graphics cards, we expect that these problems will fade away. Conceptual limitations of DRM are that it models no more than two diffuse reflections and no indirect shadows.

Our frame rates are close to real-time rates and we have confidence that graphics cards already available on the market can improve our frame rates significantly.

We think that DRM is quite useful for real-time applications in general since it features, in choice of calculation method and choice of number of samples (M), a quality/quantity trade of between image correctness and rendering frame rate. This means that it can be adapted to the performance level of the particular machine it is running on.

Finally all it takes to implement DRM is for the renderer to take a few different low-resolution pictures from the light source(s) (picturing eg. positions and direct radiance), make a few regularly spaced look-ups in these pictures, and do a summation over the look-ups in a fragment program according to formulas given in this paper.

8 Acknowledgement

Thanks to Bent Dalgaard Larsen for the suggestion of MIP mapping to improve on the sudden shifts in the indirect illumination.

References

[1] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, October 1976.

[2] Cornell University Program of Computer Graphics. *The Cornell Box*. http://www.graphics.cornell.edu/online/box/, January 1998. Accessed 25th of August 2004.

[3] Cyrille Damez, Kirill Dmitriev, and Karol Myszkowski. State of the art in global illumination for interactive applications and high-quality animations. *Computer Graphics Forum*, 22(1):55–77, 2003.

[4] Craig Donner and Henrik Wann Jensen. Faster GPU computations using adaptive refinement. In *Proceedings of SIGGRAPH 2004, Technical Sketches*, August 2004.

[5] Philip Dutré, Philippe Bekaert, and Kavita Bala. *Advanced Global Illumination*. A K Peters, Natick, Massachusetts, 2003.

[6] Pat Hanrahan. Rendering concepts. Chapter 3. In M. F. Cohen and J. R. Wallace: *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[7] Shawn Hargreaves and Mark Harris. Deferred shading. NVIDIA Presentation, 2004. http://developer.nvidia.com.

[8] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):145–154, August 1990.

[9] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Natick, Massachusetts, 2001.

[10] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, March 1995.

[11] James T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986.

[12] Mark J. Kilgard. Improving shadows and reflections via the stencil buffer. NVIDIA White Paper, November 1999. http://developer.nvidia.com.

[13] Bent Dalgaard Larsen and Niels Jørgen Christensen. Simulating photon mapping for real-time applications. In H. W. Jensen and A. Keller, editors, *Eurographics Symposium on Rendering*, 2004.

[14] Rafal Mantiuk, Sumanta Pattanaik, and Karol Myszkowski. Cube-map data structure for interactive global illumination computation in dynamic diffuse environments. In *Proceedings of International Conference on Computer Vision and Graphics*, pages 530–538, September 2002.

[15] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. Technical report, National Bureau of Standards (US), October 1977.

[16] Fred E. Nicodemus, editor. *Self-Study Manual on Optical Radiation Measurements*. National Institute of Standards and Technology, U.S. Department of Commerce, 1976–1985. NBS Technical notes, Series 910-1 through 8. Chapters were published as completed. http://physics.nist.gov/Divisions/Div844/manual/studymanual.html.

[17] Kasper Høy Nielsen and Niels Jørgen Christensen. Real-time recursive specular reflections on planar and curved surfaces using graphics hardware. *Journal of WSCG*, 3:91–98, 2002.

[18] Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Interactive global illumination in dynamic environments using commodity graphics hardware. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 450–454, 2003.

[19] Ingmar Peter and Georg Pietrek. Importance driven construction of photon maps. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proc. of the Ninth Eurographics Workshop on Rendering)*, pages 269–280, Vienna: Springer-Verlag, 1998.

[20] Timothy J. Purcell, Craig Donner, Mike Cammarano, Henrik Wann Jensen, and Pat Hanrahan. Photon mapping on programmable graphics hardware. In M Doggett, W. Heidrich, W. Mark, and A. Schilling, editors, *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 41–50. Eurographics Association, 2003.

[21] Alexander D. Ryer. *Light Measurement Handbook*. International Light Inc., 1998. http://www.intl-light.com/handbook/.

[22] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadows and lighting effects using texture mapping. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26:249–252, 1992.

[23] François X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):187–196, August 1991.

[24] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics*, 22(3):382–391, July 2003.

[25] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002.

[26] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. In *Proceedings of SIGGRAPH 2004*, 2004.

[27] Lance Williams. Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings)*, pages 270–274, August 1978.

[28] Lance Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):1–11, July 1983.

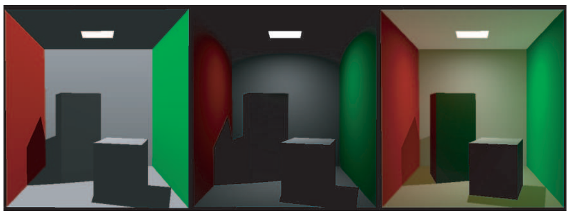

Figure 3: Different renderings of a Cornell box. From left to right: (a) A standard Blinn-Phong shaded rendering, (b) direct illumination as described in (9), and (c) direct illumination and indirect illumination reflected diffusely twice using (9), (10) and (15).

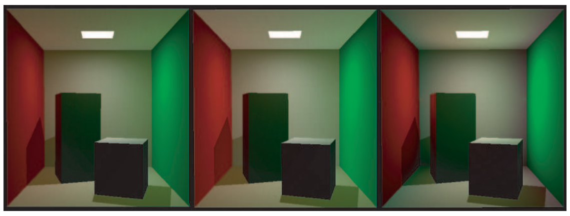

Figure 4: Comparison of the different methods for calculation of $\Delta\omega'$. From left to right: (a) Using (11), (b) using (15), and (c) using (14).

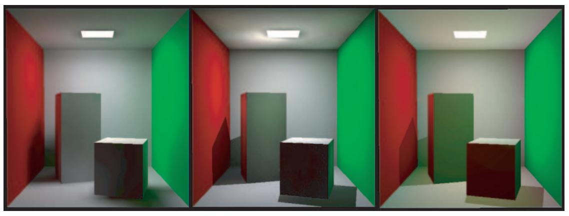

Figure 5: Comparison of DRM with traditional methods for global illumination. Here the images have been gamma corrected. From left to right: (a) Radiosity, (b) Photon Mapping, and (c) DRM using (14).

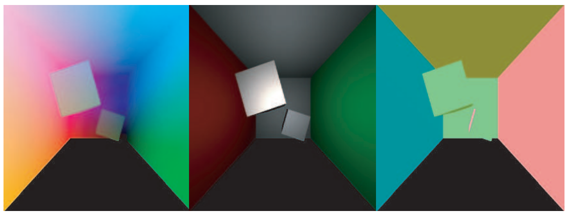

Figure 6: The textures found as pictures taken from the light source in the Cornell box. These textures comprise the direct radiance map. The picture representing the normals is only used when (14) is evaluated. From left to right: (a) Positions, (b) direct radiance, and (c) normals.

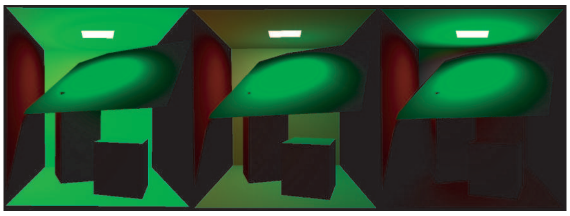

Figure 7: Problems resulting from the less accurate approximations of $\Delta\omega'$. From left to right: (a) Using (11), (b) using (15), and (c) using (14).

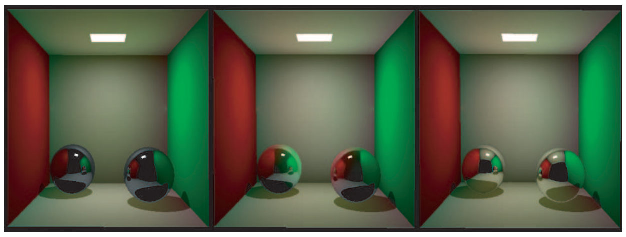

Figure 8: A Cornell box containing two mirror balls obtained by real-time calculation of perfectly specular reflections. From left to right: (a) Excluding indirect illumination in the mirror reflections, (b) using a visual trick, and (c) including indirect illumination in the mirror reflections.

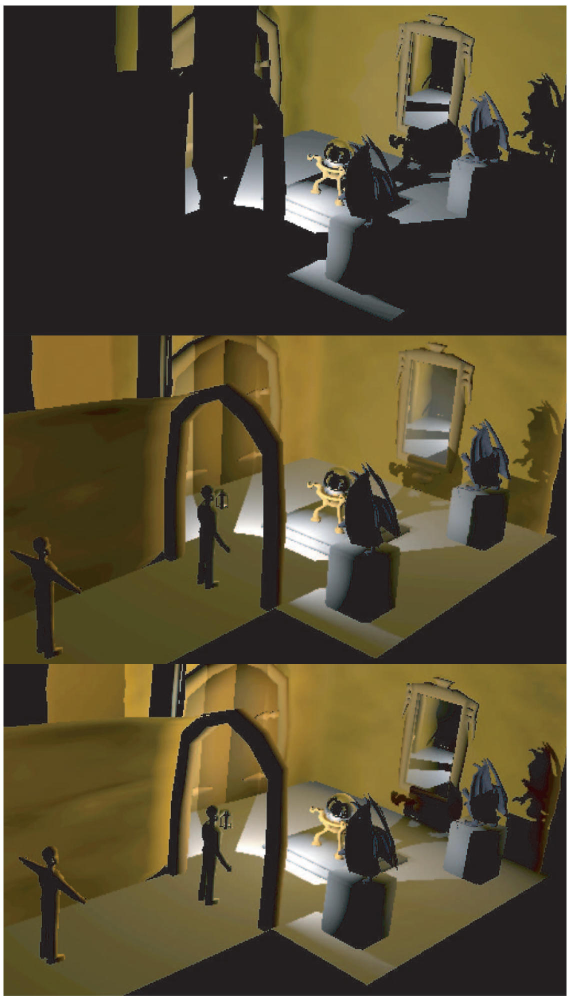

Figure 9: A cave scene consisting of 39,956 triangles. The scene includes a planar mirror and a mirror ball. The light source is in a lantern held by an animated character. From top to bottom: (a) Direct illumination, (b) indirect illumination using (15), and (c) indirect illumination using (14).