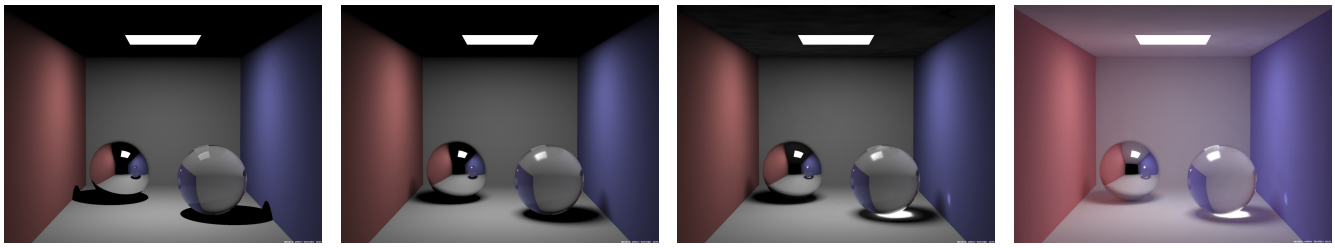
Instant Radiosity for Real-Time Global Illumination

Ingo Radax
Vienna University of Technology



(a) Direct Illumination, Hard Shadows, Reflections (b) + Soft Shadows (c) + Caustics (d) + Indirect Illumination

Figure 1: One scene rendered four times, each time with different lighting effects. We see, by adding more effects to the scene, like soft shadows caused by the area light source, a caustic caused by the glass ball and indirect illumination caused by diffuse reflection at the walls, we gain more realistic and better looking images. Though the here presented effects are not the only that can appear. Polarisation, dispersion and fluorescence are examples for other effects that we might have to consider.

Abstract

Global illumination is necessary to achieve realistic images. Although there are plenty methods that focus on solving this problem, most of them are not fast enough for interactive environments. Instant radiosity is a method that approximates the indirect lighting, as part of global illumination, by creating additional light sources. Thereby it is very fast and does not need lot of preprocessing, so it is perfectly fit to be used within real-time requirements. Further techniques based on instant radiosity have extended the method to provide better image quality or faster rendering. So instant radiosity and its derivations can bring us global illumination in real-time.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—AntialiasingBitmap and framebuffer operationsDisplay algorithmsViewing algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—RadiosityColour, shading, shadowing, and texture

Keywords: virtual point light, VPL, instant radiosity, global illumination, indirect illumination, real-time, interactive

1 Introduction

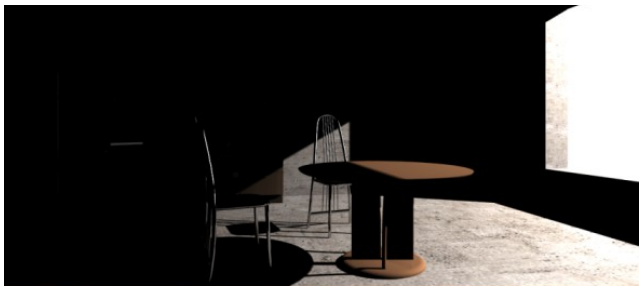
Creating realistic looking images is an important goal in a lot of computer graphic domains. To achieve this we have to consider a lot of optic effects (see figure 1). Since the human vision is not absolutely accurate, people often would not notice if some lighting effects are missing or not physically correct. So it is helpful to support just the important effects. Caustics just appear when we have some special kinds of objects in our scene (as glasses or larger areas of water) and in a lot of scenes the soft shadows are almost sharp so it would be no loss by replacing them through sharp shadows. Indirect illumination is less important for outdoor scenes, where the most light would disappear in the sky, but very important for indoor scenes (as you can see in figure 2).

Although realism is important, there is another issue for computer graphics. It is the time we need to calculate the illumination and render our images. If we want to create an image of a car for advertising, we can spend hours to render the image. If we want to make a computer animated movie we just have some minutes so create a single frame of the movie, so we can create all frames within some time. But if we want a game even minutes are to slow, here we need rendering times less than 1/25th of a second, otherwise the player would mention it.

This two, the realism of an image and its time to render, are the major two issues we have to concern with. But unfortunately enhancing the one, would worsen the other. So when we want both we have to make some restrictions, as a fixed number of polygons in the scene or just static objects.

There are a lot of methods to do the rendering. For example methods based on ray tracing (as [Whitted 1980], [Arvo 1986] or [Veach and Guibas 1997]) that can create very realistic images, but unfortunately need a lot of time. Another method would be Radiosity [Cohen et al. 1993], that can perform the rendering in real-time and also creates realistic images, but unfortunately the fast rendering is gained by a expensive preprocessing step, an enormous amount of storage and a limitation to static scenes, and the realism just includes indirect illumination.

Another method to do the rendering is the so-called Instant Radiosity [Keller 1997] algorithm. As Radiosity this method is limited to indirect illumination, but is has the advantage that it does not need the expensive preprocessing, the enormous amount of storage or the limitation to static scenes any longer. In the following sections we will concentrate in this algorithm. We will present its basic functionality (section 2), show some methods derived from Instant Radiosity (section 3) and present some methods that can be combined with the algorithm to enhance it (section 4). Another issue is



(a) Direct Illumination

(b) Direct and Indirect Illumination

Figure 2: One scene illuminated by a light source outside the window. The first image was rendered just with direct illumination without an ambient term. The second image was rendered with direct and indirect illumination. We see that without indirect illumination some parts of the scene might not be illuminated and appear complete black, so indirect illumination is an important part of rendering.

the question, if the algorithm and its derivations can be used within real-time requirements. A short conclusion (section 5) will finish this paper and recapitulate the algorithm and its derivations use for real-time applications.

1.1 Global Illumination

Global Illumination refers to rendering methods, that include other objects in a scene when calculating the light arriving at a point in a scene. Opposed to this we have the so-called local illumination, that just uses the lighted point and the light sources to calculate the light. Though this classification between global and local illumination is more academic than practice. In practice just some special kinds of effects, as indirect illumination or caustics, are called global illumination. Other effects that are simpler, like hard shadows, are not seen as global illumination, although the previous classification would call them so.

In theory, calculating the amount of light arriving at a point sounds simple. We just take the point, look into all directions and sum up the light coming from there. But to get the light coming from a single direction, we have to find the point that lies in this direction and calculate the light that arrives at him. This will continue endless, and there are also infinite directions, so that a true calculation of the light would not be possible.

A mathematic description of this is given by the rendering equation [Kajiya 1986]

$$I(x,x') = g(x,x') \left[\varepsilon(x,x') + \int_S \rho(x,x',x'')\, I(x',x'')\, dx'' \right] \quad (1)$$

$\mathbf{I}(\mathbf{x},\mathbf{x}')$ describes the amount of light arriving at point x from x'.

$\mathbf{g}(\mathbf{x},\mathbf{x}')$ is related to the scene geometry and describes how much of the emitted and reflected light from x' is sent to x. So it will always result into a value between 0 and 1.

$\varepsilon(\mathbf{x},\mathbf{x}')$ is the light emitted from x' to x. Generally this will just be used for light sources.

$\int_{\mathbf{S}}$ (calculated over S the scene) sums all light reflected at x' towards x.

$\rho(\mathbf{x},\mathbf{x}',\mathbf{x}'')$ describes how much light is sent from x'' towards x by reflection at x'.

We gain a simpler version of the rendering equation by using the operator norm

$$L = L_\varepsilon + T_\rho L \quad (2)$$

where T_ρ stands for the integral in equation 1

The solution to this equation is accessible via the Neumann series. And cause all realistic scenes have $||T_\rho|| < 1$ (no surface reflects more light than it gains) we can approximate the Neumann series by a finite sum

$$L = \sum_{i=0}^{\infty} T_\rho^i L_\varepsilon \approx \sum_{i=0}^{M} T_\rho^i L_\varepsilon \quad (3)$$

2 Instant Radiosity

Figure 3: A conference room rendered with Instant Radiosity.

Instant Radiosity (IR) [Keller 1997] calculates the indirect illumination in a scene (see figure 3) by approximating it through a particle simulation (see figure 4). We create a number of particles, starting at the light source and shoot this particles into the scene. At each

of this hit points we create a virtual point light (VPL). For each VPL we render the scene once, using the VPL as a light source, and accumulate the result in an accumulation buffer. That way we approximate the indirect illumination.

As Radiosity it is limited to diffuse reflections. But IR has the great benefit that we can directly work on the scene geometry, and so do not have to do a lot of preprocessing and do not need a lot of storage. Another great benefit is, that we just need a ray casting method, to gain the VPLs, the rendering itself can be done with common graphics hardware.

2.1 Algorithm

First we have to fix the number of particles N we want to create at the light source of the scene. The more particles we create, the better the resulting image will be. But of course, the more particles we have the longer the rendering will take.

Then we want to shoot this particles into the scene. But therefore we need a method to determine the directions we will shoot this particles. We will use the Halton sequence, which has some advantages compared against other methods like N-rooks or random sampling. One of the advantages is that we can create the Halton points easily, while with the other methods we would have to store them.

Now we can shoot the particles into the scene. When a particles hits a surface it might be absorbed, then it would vanish at the hit point, or it might be reflected, then it would continue its trip into the scene. We simply can use the mean reflectivity $\bar{\rho}$ of the scene to determine which particles will vanish and which will continue their trip. From the N particles started at the light source $\bar{\rho}N$ will be reflected and continue their trip, the others will vanish. From the $\bar{\rho}N$ continuing after the first reflection, $\bar{\rho}^n N$ will be reflected a second time, the others will not. This scheme continues until no more particles remain. We call the total number of hit points during this this process as M, and this number is limited through following formula

$$M < \sum_{j=0}^{\infty} \bar{\rho}^j N = \frac{1}{1-\bar{\rho}} N \tag{4}$$

The starting points at the light source and the following hit points during the particle shooting process will be used as virtual point lights. Thereby we have to consider two things. First we need the light colour of the VPLs. For the VPLs created at the light source it is clear which colour to use, and for the VPLs created at hit points it is almost as simple to gain. We just take the colour of the starting point and subsequently attenuate it with the surface colour at the hit point. Then when it comes to rendering, we have to consider the second thing. We want all images rendered with a VPL to be equally important. But due to the attenuation they would not be if we used the light colour directly, so we have to compensate it for rendering. This compensation can easily be done by multiplying the light colour with the factor $\frac{1}{[\bar{\rho}^j N]}$, where j stands for the j-th reflection of the particle the VPLs belongs to.

Now we have everything we need to to the rendering. We render the scene with shadows once for each VPL and using it as the light source. The resulting images during these separate rendering steps are accumulated with the weight $\frac{1}{N}$. So we gain a good approximation of the indirect illumination in our scene.

2.2 Performance

Now we want to take a look at the performance of Instant Radiosity. Thereby we can focus on two different issues. First is the quality of the indirect illumination, cause what would the method be useful for if it could not approximate it sufficient. And second we want to assess the methods use for real-time purposes.

First we come to the quality of indirect illumination. As we saw earlier (see figure 3) we can use Instant Radiosity to approximate the indirect illumination in a very realistic looking way. When we take a look at figure 5, we see that the quality of indirect illumination directly results from the number of particles starting at the light source. The more particles we start with, the better the result will be, so we can indirectly control the amount of quality.

And now lets take a look at the rendering speed. The scene in figure 5 consists of 402 quadrangles and all three images where rendered within 24 seconds. Since the rendering time was for all three images the same, despite all three used different numbers of VPLs, we can say that the bottleneck of this method is the rendering itself and not the creation of VPLs. And when we now take a direct look at the rendering time, 24 seconds, we can annotate that Instant Radiosity was first published in 1997 [Keller 1997]. Since then hard- and software have become much better. So we can proceed on the assumption that nowadays Instant Radiosity would be capable of calculating the indirect illumination, of at least simple scenes, in real-time.

3 Derivations of Instant Radiosity

Instant Radiosity was first published more than ten years ago, so it is not surprising that there are newer publications that build upon Instant Radiosity. In this section we present several methods that extend or use Instant Radiosity.

3.1 Bidirectional Instant Radiosity

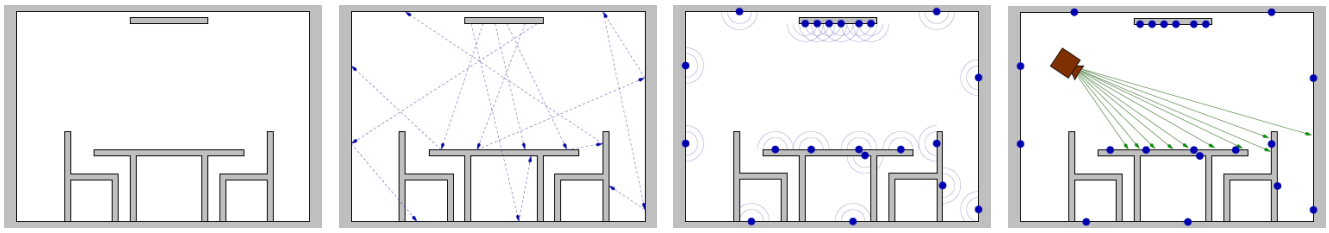
Using Instant Radiosity we start the creation of virtual light sources at the light source. But for some scenes, the light source might be bad placed, so that VPLs created with this light source might not illuminate the part of the scene which can be seen by the camera. Here it would be better not to start at the light source, but at the camera to create the VPLs. Bidirectional Instant Radiosity [Segovia et al. 2006] is a method based on this idea.

Bidirectional Instant Radiosity is based on Instant Radiosity, but besides the VPLs created at the light source, it also includes VPLs created at the camera. The following two sections of this paper will describe the basics techniques and show if it can be used within real-time requirements.

3.1.1 Bidirectional Sampling of the VPL

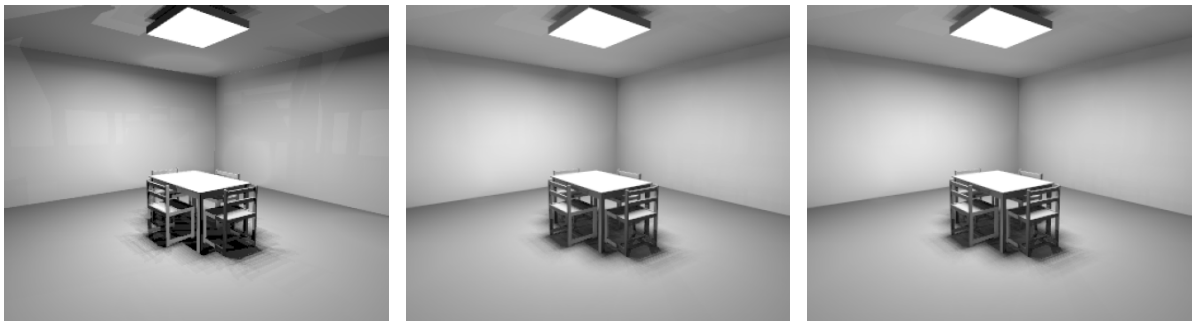
Basically Bidirectional Instant Radiosity consists of two extensions to classic Instant Radiosity. The first one is a technique to create VPLs starting at the camera instead of starting at the light source. The second one, is a method to combine first technique with Instant Radiosity, so that we can use both, VPLs started at the light source and those started the camera.

As we remember, the problem with classic Instant Radiosity was, that the created VPLs might not illuminate the part of the scene,



(a) Scene (b) Shooting rays into the scene (c) Using hit points as VPLs (d) Rendering the scene with VPLs as light sources

Figure 4: Here you see the basic functionality of Instant Radiosity. When we have our scene we approximate the indirect illumination in the scene by VPLs. We start at the light source and shoot some rays into the scene, these rays might be reflected and end after some time. At each hit point of the ray we create a virtual point light. Then, when it comes to rendering, we render the scene once for each VPL and using the VPL as the actual light source. The resulting images are accumulated and form the final result.



(a) N=10, M=20 (b) N=32, M=72 (c) N=64, M=147

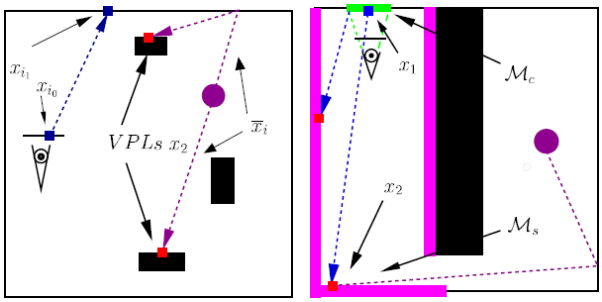
Figure 5: A single scene rendered three times with Instant Radiosity. Each time a different number of particles N was used. As we see in the first image, a relatively small number of leads to discretisation errors in form of hard shadows. We do not have enough enough VPLs so that the hard shadows would merge and they stay visible. As we see in the second and third image, with using more starting particles, the hard shadow artefacts disappear.

seen by the camera. So we want to create VPLs starting at the camera (called Reverse Instant Radiosity), but that is not enough. Another requirement is of course that these Reverse-VPLs can illuminate the appropriate part of the scene, otherwise they would have no influence at the result. We know that light propagates in a linear way, so it is easy to see that we can use those parts of the scene as Reverse-VPL who can see a fraction of the part of the scene, seen by the camera.

Finding Reverse-VPL is quiet simple. We create random paths of length two, starting at the camera (see figure 6(b)). At the end of such a path will be the position of a Reverse-VPL.

Just finding Reverse-VPL is not enough. Cause we started at the camera instead of the light source, some information is missing, so we have to estimate it. The first necessary information, it the Reverse-VPLs density of probability, which tells us how probable it was to reach the Reverse-VPLs position with a length two path. The second necessary information, is the outgoing light of the VPL. For simplicity we expect the Reverse-VPLs light to be constant for all possible angles. Then we have to connect the Reverse-VPL with a light source (as in figure 6(b)) and calculate the amount of light arriving from the light source at the Reverse-VPL. To to this we start multiple paths at the Reverse-VPL and then also create some standard VPLs. Then we connect the paths from the Reverse-VPL with the standard VPLs with shadow rays, and calculate the amount of light arriving at the Reverse-VPL.

As you can see in figure 7, we can use the Reverse Instant Radiosity approach to create better results than by using the classic Instant

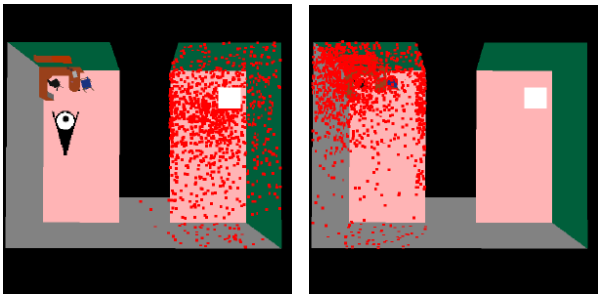


(a) Instant Radiosity (b) Reverse Instant Radiosity

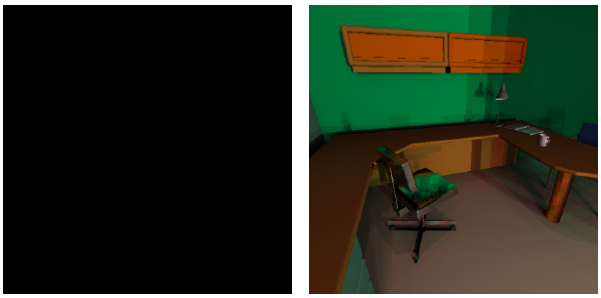
Figure 6: Figure (a) shows classic Instant Radiosity, VPLs are created from the light source. In figure (b) we see Reverse Instant Radiosity. VPLs are created at those points (magenta area) that can illuminate something seen by the camera (green area). Then the created Reverse-VPLs are connected to the light source so calculate the amount of light arriving at the VPL.

Radiosity. Although Reverse Instant Radiosity has it benefits, it is not always the best method, and sometimes it would be better to use the classic approach (see figure 8). So we need a method to combine classic Instant Radiosity with Reverse Instant Radiosity, to yield better results for most of all scenes.

To combine classic and Reverse Instant Radiosity it is actually just necessary to create standard VPLs and Reverse-VPLs and to decide



(a) VPLs of classic Instant Radiosity (b) VPLs of Reverse Instant Radiosity



(c) Rendered with classic Instant Radiosity. (d) Rendered with Reverse Instant Radiosity.

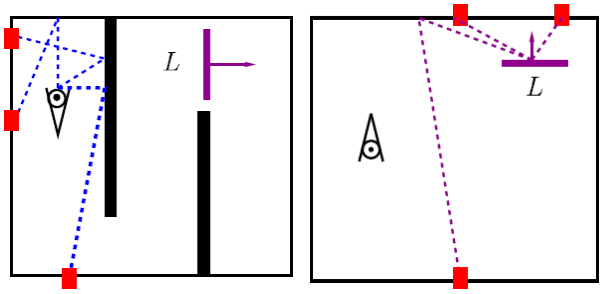
Figure 7: Two offices connected through a small corridor. The camera is in one offices, the light source in the other. In figures (a) and (b) we see the positions of VPLs, once created with standard Instant Radiosity and once created with Reverse Instant Radiosity. In figures (c) and (d) we see the corresponding images to the previous figures. The first image stays complete black because all VPLs where created in the wrong room, so there is nothing that illuminates the scene. Otherwise in the second image, there the office is good illuminated because Reverse Instant Radiosity was able to find VPLs that illuminate the scene.

which of them would have the most influence to the result. So we use a sampling resampling approach. First we create N/2 standard VPLs and N/2 Reverse-VPLs. Then we build a cumulative distribution function (based on the amount of energy arriving from die VPLs at the camera) and then use this distribution function for resampling. Therefore we select the N' VPLs with the most influence to the camera, and use them for rendering.

3.1.2 Performance

Now that we know the basic techniques to use Bidirectional Instant Radiosity, we also want to know if it can be used under real-time requirements. In figure 9 we see three different scenes, all with a different number of triangles and all rendered with the special implementation of Bidirectional Instant Radiosity, presented in the original paper [Segovia et al. 2006].

When we look at the three scenes in figure 9 we can compare both, their convergence time or their frames per second (see table 3.1.2). The convergence time tells us how long the rendering took until the RMS error was smaller than 1%. The frames per second were gained with 30 VPL per second. We see that the convergence time is for all scenes longer that two seconds. Of course this would be to slow for interactivity, but cause of the strict RMS error rate and the fact that the original implementation was not fully optimized, it is not that much expressive. More expressive are the frames per second values. The office is with its 15 fps almost at real-time and also the other two scenes might reach it with some more optimization of the implementation.



(a) Better use VPLs startet at the camera (b) Better use VPLs startet at the light source

Figure 8: In the first scene it would be better to create the VPLs by starting at the camera. Otherwise most of the VPLs would be created at positions where they do not illuminate anything. Otherwise the second scene, here would almost all VPLs started at the light source illuminate something, and it might be hard to find VPLs by starting at the camera.

Scene	Convergence time	Frames per second
Office	2.4 s	15 fps
Conference Room	3.8 s	4.5 fps
Cruiser	5.9 s	2.2 fps

Table 1: Rendering Performance of the scenes in figure 9

As conclusion we can say that Bidirectional Instant Radiosity extends the standard algorithm so we can gain good results also in parts of the scene that are far away from a light source. And with a good implementation of this method it would be also possible reach real-time speed.

3.2 Metropolis Instant Radiosity

Metropolis Instant Radiosity [Segovia et al. 2007] is another variation of the classic Instant Radiosity method. When it was invented, it was the target to create a technique that is numeric robust but also quiet fast. The new method was based on two existing methods. The first was of course Instant Radiosity, the second one is a Metropolis sampler [Veach and Guibas 1997].

The following two sections of this paper will give you an overview of how the method works and answer the question if it can be used under real-time requirements.

3.2.1 Algorithm

In this section we will explain how Metropolis Instant Radiosity works. Cause the algorithm is basically a combination of Instant Radiosity and a Metropolis sampler, we will give just a brief outline. If a more detailed description of the algorithm is wanted, we recommend to read the original paper [Segovia et al. 2007].

The algorithm works this way (description directly taken from the original paper [Segovia et al. 2007]):

1. Set all pixel intensities to 0.

(a) Simple office with about 35.000 triangles. (b) Conference room with about 200.000 triangles (c) Cruiser with about one million triangles.

Figure 9: Three different scenes rendered with Bidirectional Instant Radiosity

2. Compute the power P_c received by the camera.

3. With a Metropolis-Hastings sampler compute a set of n VPLs with a density proportional to the power they bring to the camera. We do not know the the outgoing radiance functions of the VPLs but we know the scene transmits the same amount of the VPL power to the camera.

4. **for** i=1 to n **do**

5.
   - Suppose that VPL i is on a diffuse surface and that it has a constant outgoing radiance function equal to 1. Compute the intensity of each pixel in the screen and the total power P' transmitted to the camera through the scene from VPL i.
   - As we know that VPL i transmits a power equal to $\frac{P_c}{n}$ to the camera and that there is a linear relation between the outgoing radiance function of the VPL and the transmitted power, rescale the intensities of the pixels by a $\frac{P}{nP'}$ factor.
   - Accumulate VPL i contribution

6. **end for**

The interesting part of this algorithm is point 3, where we use a Metropolis sampler to create the VPLs. Actually we have two possible Metropolis sampler we can use here. The first one is a Metropolis-Hastings (HM) sampler described in [Veach and Guibas 1997]. But there is another sampler, called the "Multiple-try Metropolis-Hastings algorithm" (MTHM) [W. 2000] that works better (see figure 10). So we recommend to use the MTHM.

(a) Scene overview (b) HM (c) MTHM

Figure 10: A comparison of a HM and a MTHM sampler. Both times with 256 VPLs and the same computation time. We see that the MTHM scene is lighter than the HM scene, and also the colour bleeding effect is more apparent.

3.2.2 Performance

Now we will take a look at the performance of Metropolis Instant Radiosity and compare it with classic Instant Radiosity and Bidirectional Radiosity. When want to see if the Metropolis variant can keep up with the other two. When we take a look at figure 11 we see two scenes, both rendered three times for each scenes with one of the three methods.

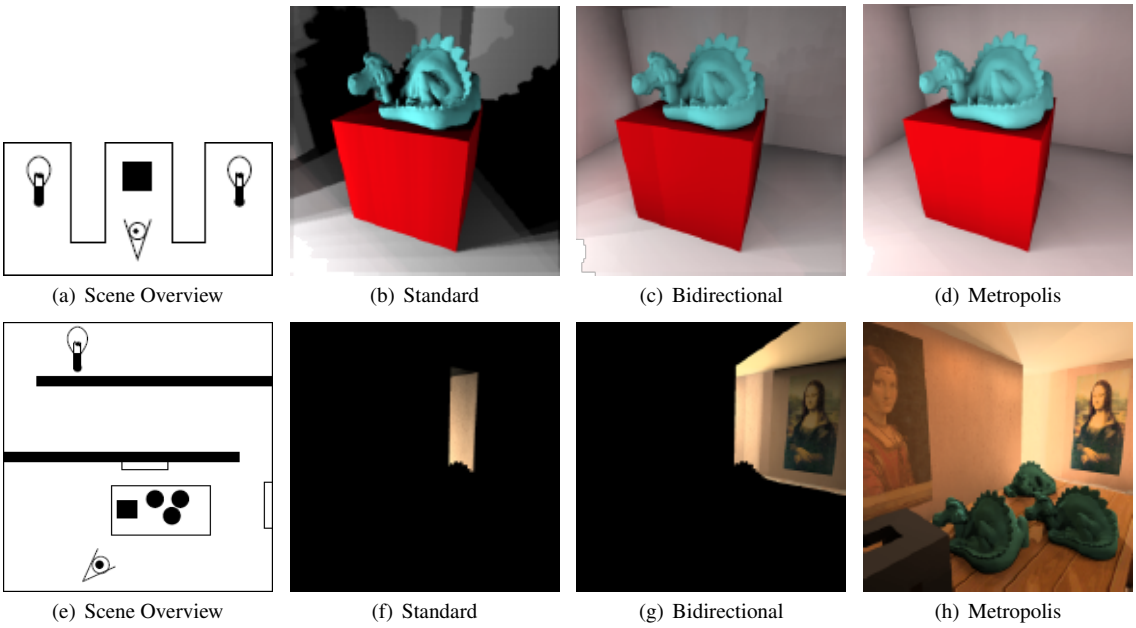
The first scene in figure 11 is a simple dragon on a box. We see that the classic Instant Radiosity provides us a poor result, but both other provide a good result. And although we can see some differences within the Metropolis and Bidirectional rendered images, we can not really say one would be much better than the other. But for a better comparison we can take a look at the second scene, where we see three dragons. Or better said, in two of the three images we actually do not see the three dragons. Standard Instant Radiosity but also die Bidirectional variant fail to illuminate the scene completely. Here it is easy to see that the Metropolis variant of Instant Radiosity provides a better result.

Now we know that Metropolis Instant Radiosity can keep up with Instant Radiosity and Bidirectional Instant Radiosity. That knowledge is good but tells us nothing about its usefulness within real-time requirements. So we take a look at table 3.2.2 where we compare Metropolis Instant Radiosity with Bidirectional Instant Radiosity. In the table we compare the generation time of VPLs for both methods and for multiple scenes. We see that the Metropolis variant is just slower than the Bidirectional variant in one of all scenes.

Scene	Metropolis	Bidirectional
Scene 6	0.32 s	0.82 s
Scene 10	0.31 s	0.82 s
Office	0.31 s	0.62 s
Conf	0.49 s	0.92 s
Theater	1.0 s	1.0 s
Cruiser	0.92 s	1.3 s
Three Dragon Room	2.4 s	1.0 s

Table 2: VPL generation times for different scenes comparing Metropolis and Bidirectional Instant Radiosity

As a short conclusion we can say that Metropolis Instant Radiosity can handle even more complex scenes than Bidirectional Instant Radiosity could, and in most of cases it would be also faster. So as die Bidirectional variant can be a good way to include indirect illumination within real-time requirements.



(a) Scene Overview (b) Standard (c) Bidirectional (d) Metropolis

(e) Scene Overview (f) Standard (g) Bidirectional (h) Metropolis

Figure 11: A comparison of standard Instant Radiosity, Bidirectional Instant Radiosity and Metropolis Instant Radiosity. We see that Instant Radiosity fails to find good VPLs when the light sources are bad places. Bidirectional Instant Radiosity helps to avoid this problem for some scenes, and is almost as good there as Metropolis Instant Radiosity. But as we see in figure (g) and (h), also Bidirectional Instant Radiosity has its limits and fails for a more complex scene to find enough good VPLs.

3.3 Incremental Instant Radiosity

Another variant of Instant Radiosity is is the so-called Incremental Instant Radiosity [Laine et al. 2007]. The central idea behind this method is to reuse VPLs and maintain their good distribution (see figure 12). So we avoid a lot of computations but still have a method for real-time indirect illumination.

Although reusing VPLs sounds great, we have to make some constraints to achieve this. The first is that the contribution to the indirect illumination is limited to the static part of the scene. Each of the VPLs needs a shadow map so we can tell what they can illuminate, and if we just use the scenes static part, the shadow map of a VPL just has to be computed once. Although the contribution is limited to the static part, the created VPLs can illuminate the dynamic part of the scene, and by using a single shadow map for the primary light source we can also have direct lighting and shadowing for the complete scene.

Another constraint is that we limit the generation of VPLs to 'first hit'-VPL, what means when we cast a ray from the light source to find a VPL, we will take the first hit of the ray as the position of the VPL. This might seem to be inaccurate, as that it could be used to calculate the indirect illumination, but as been demonstrated by Tabellion and Lamorlette [Tabellion and Lamorlette 2004] such kind of indirect illumination is good in many cases.

The last constraint is, that we will just consider 180 spotlights with cosine falloff and omnidirectional point lights. Although it should be not to difficult to extend the algorithm for other types of light sources.

3.3.1 Algorithm Outline

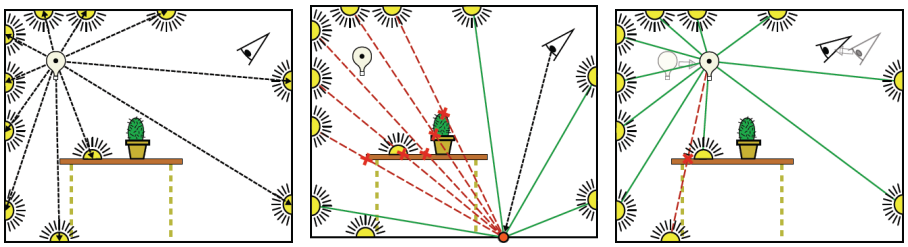
We give now a short outline of how to calculate the indirect illumination with Incremental Instant Radiosity (the outline is taken from the original paper [Laine et al. 2007]):

1. Determine the validity of each VPL.
2. Remove all invalid VPLs and possibly a number of valid ones to improvement the distribution.
3. Create new VPLs according to allotted budget. Render paraboloid shadow maps for them.
4. Compute intensities for VPLs.
5. Render the positions, normals and colours as seen from the camera into a G-buffer (see section 4 'Deferred Shading').
6. Split the G-buffer into a number of tiles.
7. Loop over tiles and accumulate illumination from a subset of VPLs in each of them.
8. Combine the tiles back into a single image.
9. Smooth the accumulated illumination using a spatially-varying filter kernel.

As you can see, the algorithm persists of two blocks. The first block contains the points 1 to 4 and occupied with the management of the VPLs. The second block, the points 5 to 9, does the rendering for the indirect illumination. Although this rendering is important, we will focus on the first block and refer to the original paper [Laine et al. 2007] for further information.

3.3.2 Distribution of VPLs

The distribution of the VPLs can be seen in two ways. The first one is a directional distribution as seen by the primary light source. The second one is a intensity distribution. In the best case the directional distribution should follow the intensity distribution, then each VPL would represent a similar fraction of the primary light



(a) Create VPLs (b) Illumination with VPLs. Red lines show occlusion. (c) Reuse of valid VPLs, deletion of invalid VPLs (occluded ones)

Figure 12: The basic functionality of Incremental Instant Radiosity. We start as common and create VPLs from the primary light source. Then, we can use this VPLs to illuminate the scene. But then, when it is time for the next frame, we dont have to create all VPLs anew. Instead we check the validity of all VPLs and just remove the invalid ones and reuse the valid ones. The movement of the camera does not affect this procedure.

sources power. The job of managing the VPLs is to delete old and add new VPLs, so that the distributions stay close together.

Cause we limited the primary light source to spotlights and omnidirectional point lights, we can map the directional distribution into a 2D domain. For spotlights we can use the unit disc (see figure 13(a)), and for omnidirectional lights we can use the unit sphere (see figure 13(c)).

The great benefit by doing this is that we can work in the 2D domain when managing the VPLs. Most of the managing work can by done by a Delaunay triangulation and the associated Voronoi diagram (seen in figure 13). A second advantage by using the 2D domain is, that when we uniform distribute the VPLs in the 2D domain, they intensity distribution automatically follows the directional distribution.

3.3.3 Validity Testing

The first point of the algorithm tells us that we shall determine the validity of each VPL. In other words, we shall find out which VPLs will be reused and which not. Since we limited our primary light sources to spotlights and omnidirectional point lights the test is quiet simple, a VPL is invalid if:

- it is occluded from the new position of the primary light source, or
- it fall outside the illuminated region of the primary light source

Of course, the second case, might just happen when the primary light source is a spotlight.

3.3.4 Deleting VPLs

Now that we have categorized all VPLs in valid or invalid, we can delete the invalid ones. But to maintain the quality of the VPL distribution it is sometimes necessary to also delete some of the valid ones. We have a simple heuristic so select one VPL to be removed. We use the Delaunay triangulation to select one VPL so that the smallest distance between VPLs in die Voronoi diagram increases when we delete the VPL. This deletion (with a followed Delaunay triangulation to keep it current) is then repeated until there have been enough VPLs removed.

3.3.5 Creating New VPLs

After deleting VPLs we will not have the maximum number of VPLs any more, so we have to create new ones. But it might happen that the deletion removed so much VPLs that creating a new VPL for each deleted might lead us to performance problems. Therefore we have a maximum limit of VPLs to be created in one step. This ensures a good performance but might lead into a temporal degeneration of shadow quality.

Creating new VPLs is quiet as simple as the deleting, and can also be done with the Delaunay triangulation. To keep the uniform distribution of VPLs we want a new VPL to be at that place in the Voronoi diagram where it has the most space, or in other words, where the largest empty circle lies in the diagram. This place can be simply found with the Delaunay triangulation. After we have the position of our new VPL in the Voronoi diagram, we map the position back into the 3D domain, recalculate the Delaunay triangulation and create the shadow map for the VPL.

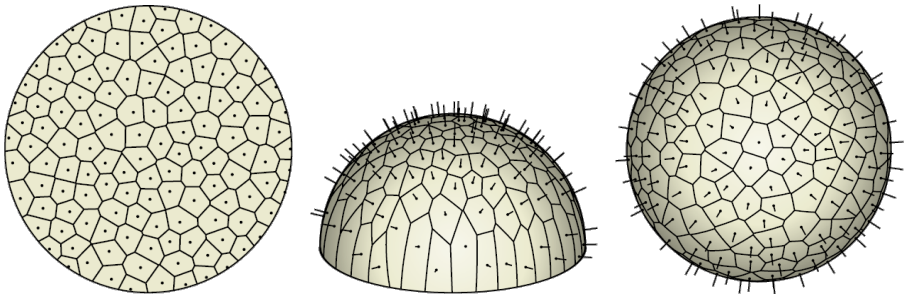
3.3.6 Computing Intensities

The distribution of VPL should ideally be evenly spaced across the domain. But cause of the movement of the primary light source and the limited number of VPLs, it would lead us to incorrect results if we would give all VPLs the same intensity. So we have to recalculate the intensities based on primary light sources intensity and the solid angle it represents. And here again we can use the 2D domain to get this data. We take the area of the Voronoi region of a VPL as the VPLs intensity and scale it so that the sum of all intensities will be the primary light sources intensity.

3.3.7 Performance

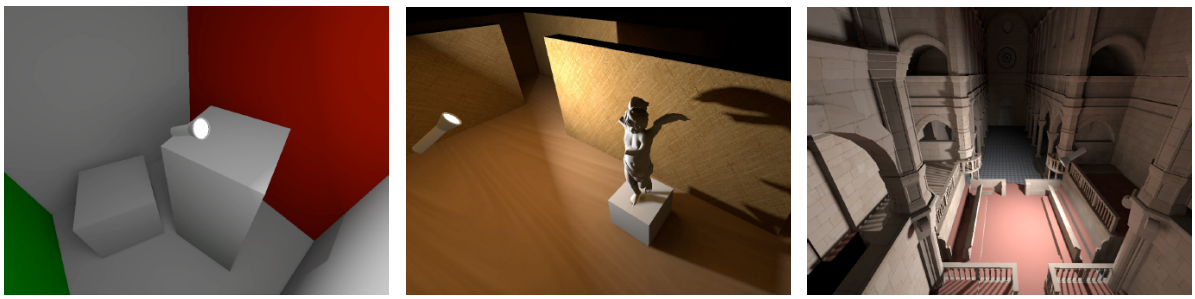
Now we will take a look at Incremental Instant Radiositys applicability for real-time requirements. As we remember, the central idea was to reuse VPLs and so avoid a lot of computations. That is why we will test the method against another, or better said we will test just the part for reusing the VPLs against a method that creates the VPLs anew each frame. Both methods will use the same rendering method. The comparison method corresponds to Bidirectional Instant Radiosity [Segovia et al. 2006].

When we look at figure 14 we see the three scenes we tested. The test have been done with different resolutions and also with different primary light sources. For all this different cases we see about



(a) Unit disc (b) Unit disc mapped onto a hemisphere (c) Unit sphere

Figure 13: 2D domains for VPL distributions with Delaunay triangulation. These domains allow us to perform VPL operations (insertion, deletion, etc.) in this domains. That is much easier than doing this in the 3d scene geometry. Figure (a) shows uniform distributed samples in a unit disc, this can be used for spotlights with a 180 cosine falloff. As we see in figure (b), the areas in the unit disc, correspond to the cosine weighted integrals. Figure (c) shows uniform distributed samples on a unit sphere, this can be used for omnidirectional point lights.



(a) Cornell (b) Maze (c) Sibenik

Figure 14: Scenes used for comparing Incremental Instant Radiosity with a comparison method

the same effects, hence we just present a snapshot of all tests (see table 3.3.7). As we can see, the comparison method is beaten by Incremental Instant Radiosity for all tested scenes, and it also provides good results where the comparison method drops below real-time performance.

Scene	Incremental Instant Radiosity	Comparison method
Cornell	65.1 fps	35.1 fps
Maze	49.2 fps	12.5 fps
Sibenik	48.6 fps	7.1 fps

Table 3: Comparison of Incremental Instant Radiosity with the comparison method

With Incremental Instant Radiosity we have another variant of Instant Radiosity. More than the methods before it can be used within real-time requirements. Although it has to be said that this method still has it disadvantages, like it just uses 'first hit'-VPLs and that the indirect illumination just bounces off from static geometry.

3.4 Interactive Global Illumination

For a long time, ray tracing techniques needed at least minutes to render single images. But newer techniques have speeded up the performance of ray tracing, so that interactive ray tracing can be achieved on common hardware (see [Wald et al. 2002b]. [Wald et al. 2001a] and [Wald et al. 2001b]). This method of fast ray tracing scales well in a distributed system, using commodity computers and networks. The distribution is implemented as a client/server model. Through this distribution, fast ray tracing is capable of rendering scenes with interactive framerates. And with an extension for dynamic scenes [Wald et al. 2002a], we gain a interactive application.

Though this fast ray tracing ought to be combinable with existing global illumination algorithms resting upon ray tracing. But the problem is, that most of this algorithms are not compatible with the constraints imposed by such a distributed ray tracing system.

Interactive global illumination (IGI) [Wald et al. 2002c] is a method, designed to be rest upon the fast ray tracing system and to be compatible with the constraints imposed by fast ray tracing. It uses Instant Radiosity to approximate the indirect illumination, a simpler version of photon maps to calculate caustics and ray tracing for other effects like reflections or refractions.

In the next section we will present the constraints imposed by fast ray tracing, so that we understand what IGI should be capable of. Then follows a section, describing the method behind IGI and then we will take a look at the performance of IGI in the last section.

3.4.1 Constraints

Fast ray tracing allows us to achieve interactive frame rates at dynamic settings. So we could expect it would be useful to combine fast ray tracing with existing global illumination methods, so that we would have a interactive global illumination system. Unfortunately the most of the existing global illumination methods do not work with all the constraints imposed by fast ray tracing. We will now take a closer look at this constraints.

First we have a performance constraint. With fast ray tracing we should be at least capable of calculating 500.000 rays per second. Even when we use a distributed system with multiple computers and processors, we will not have more than some millions of rays per second. A small image with a resolution of 640x480 has about 300.000 pixels. So for each of this pixels we would just have some dozen of rays per second. And this are just rays, and not complete paths of rays we might need. And this is also just for one second, for interactive framerates we would have less rays. So, a global illumination method would have to use this rays effectively as possible to get good results.

Next we have a constraint concerning with parallel and distributed computing. We need a several computers, combined to a distributed system via a common network infrastructure, to have enough rays. But to keep interactive framerates, we have a lot of requirements to our global illumination method. It will have to keep its bandwidth requirements low, it will have to hide latencies, all its calculations will have to be parallelisable, we need a lot of this calculations for a good job scheduling and it will have to minimize the synchronization across the network.

At last we have an interactivity constraint. A lot of common global illumination method need lot of lengthy preprocessing. But we want a interactive system, so we can not need to long for preprocessing. Thus our global illumination method will have to do minimal preprocessing, so we keep interactive framerates.

3.4.2 Algorithm

Interactive global illumination was developed to run upon a fast ray tracing system and to comply to the constraints imposed by this system. The first implementation of interactive global illumination [Wald et al. 2002c], had some scalability problems that led to a maximum framerate of about 5. So it was reimplemented [Benthin et al. 2003] so the former problems where abolished and also performance was increased. Additionally some new features have been implemented to.

Primary interactive global illumination builds upon fast ray tracing and instant radiosity. In a short preprocessing step, we create the VPLs, and then create a primary ray for each pixel to find the corresponding point in the scene for this pixel. Then we shoot shadow rays from this point to the VPLs, check for occlusion, illuminate the point and store it into the framebuffer.

Due to the limited number of VPLs we gain discretisation artefacts, in form of hard shadows. We use interleaved sampling (see section 4) to destroy the correlation between neighbouring pixels and so remove the hard shadow artefacts. But by interleaved sampling we would have structured noise in the scene, so we use a discontinuity buffer (see section 4) to eliminate this structured noise.

With ray tracing we also have the capabilities of simple specular effects, like reflections or refractions. Additionally we have a simple version of photon maps (see figure 15) so we also can have caustics in the scene. Unfortunately these caustics led to scalability problems, there where removed later [Benthin et al. 2003] so the new implementation does not support caustics any longer.

Within the new implementation of the system [Benthin et al. 2003], also some new features to improve image quality have been introduced: support for textures, surface shaders, programmable light sources, efficient anti-aliasing and tone mapping.

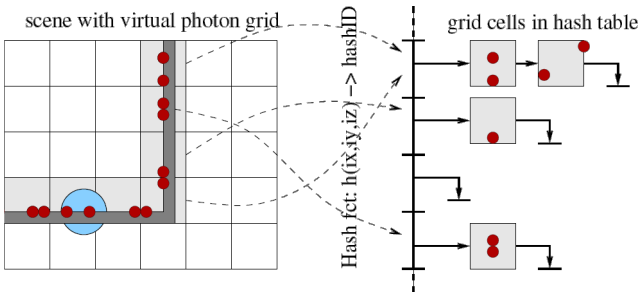


Figure 15: Photon mapping is just there used where the photon density is high. So we can fix the filter radius r (blue circle) to find photons. Instead of storing the photons in a kd-tree we use a grid structure, and fix the cell size of the grid to be $2r$. So we just need to check 8 grid cells to find all photons. Just few of the cells would have photons, so we can avoid storing all and use a hashing scheme to find and access them. Photons stored in a grid structure, and accessible via a hash table.

3.4.3 Performance

Now we will take a look at the performance of interactive global illumination and its use for real time applications. The first implementation of IGI [Wald et al. 2002c] had scalability problems that limited the maximum number of frames per second to about 5. Within the second implementation [Benthin et al. 2003] these problems have been solved. Thus we will focus on the performance of the second implementation.

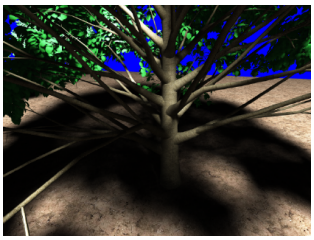
When you take a look at figure 16 you see some images rendered with interactive global illumination. We see the indirect illumination looks good, but we are more interested in the rendering time. With IGI there is a special case, compared to the other instant radiosity derivations. We have a scalable system, so we do not just want to have the rendering time for different number of computers in the system.

In figure 17 we see the resulting frames per second for different number of computers in the system. We see that for to little computers, the performance is to slow as to be called interactive. But we also see that the performances almost perfectly scales linear with increasing number of computers and in some cases also reaches more than 20 frames per second. To, when we would have enough computers, we would also be able to render in real time.

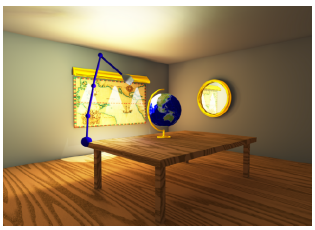
Interactive global illumination builds upon a fast ray tracer and uses instant radiosity to approximate the indirect illumination. Designed as a client/server system, it almost perfectly scales linear with the number of computers. Although is can not be used for single computers now, future developments of hard- and software might make it applicable.

4 Extensions for Instant Radiosity

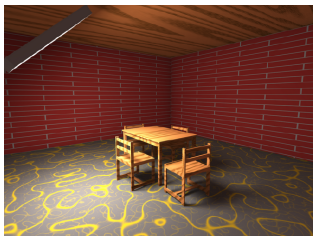
This section covers the topic of possible extensions for Instant Radiosity. This extensions might be combined with Instant Radiosity or its derivations, to accelerate the rendering, to enhance rendering quality or to add new features.



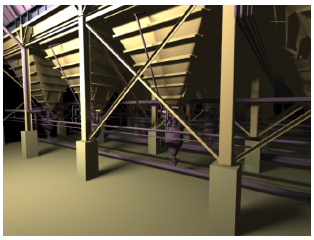
(a) Maple Trees



(b) Room with globe



(c) Shirley 6



(d) Power Plant

Figure 16: Several scenes rendered with interactive global illumination. (a) Complex shadows of an entire maple three with 1.5 million triangles. Rendered with 4-6 fps on a cluster of 24 dual PCs. (b) A room with a globe, with about 20.000 triangles. The lamp illuminates a part of the globe and so reflects a yellowish colour into the scene. (c) The Shirley-6 test scene with 600 triangles using a procedural shader. Rendered with about 12 fps. (d) The power plant scene, containing more than 50 million triangles. Renders with about 2 fps.

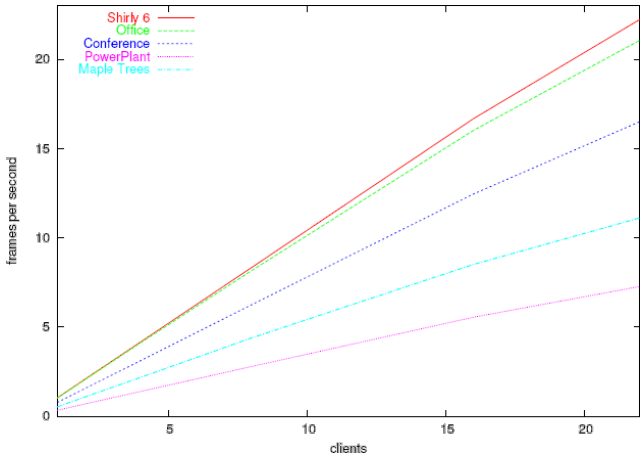


Figure 17: A performance comparison of different scenes rendered with interactive global illumination. Each scene has been rendered multiple times, each time with more computers in the IGI system. We can see that the performance almost perfectly scales linear with the increasing number of clients in the system. So with enough computers we might reach real time.

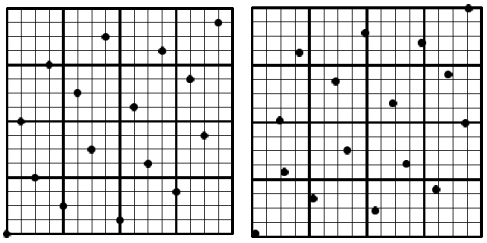
4.1 Jittered Low Discrepancy Sampling

The first extension was published within the first publication of Instant Radiosity [Keller 1997]. When you take a look at figure 18 you see sample points of the Hammersley are aligned to a grid. This ensures a minimum distance between the points, but also might lead us to aliasing. A simple way to get rid of this is to jitter the sample points (see figure 18), not much so that they remain in the unit interval. By jittering the points we reduce aliasing artefacts and it also results in a faster convergence rate.

4.2 Specular Effects

Another extension was published within the first publication of Instant Radiosity [Keller 1997]. As Radiosity IR approximates the indirect illumination, but just for diffuse surfaces. Although it would be difficult to extant the method for all possible specular surfaces, it is quiet simple to extent it to plain specular surfaces.

When a particle hits a surface we check if it is a diffuse or a specular surface. If it is a diffuse surface we continue as so far and use the hit point as a VPL. But if it is a specular surface, we mirror the origin



(a) Hammersley (b) jittered Hammersley

Figure 18: Grid structure of Hammersley and jittered Hammersley for 16 sample points

of the particle (see figure 19) by the surface and use the mirrored origin as a VPL. It is just necessary to mind that the lighting just happens for elements inside the pyramid spanned by the mirrored origin and the specular surface.

4.3 Realtime Walkthroughs

Another extension was published within the first publication of Instant Radiosity [Keller 1997]. It is simple to alter the base algorithm of Instant Radiosity to better fit into an animated situation. Yet we always created all VPLs new per frame, but instead we just create one particle new and follow its path through the scene. We generate the image by the VPLs on this path and store it with its generation time. If we store the last N images, we can simply replace the oldest image when a new path is completed. Then we accumulate all images and display the result. So we gain a simple method for animated scenes and implicitly perform temporal antialiasing.

4.4 Deferred Shading

Using the common rasterisation method for rendering, each pixels colour will be overwritten when a polygon is rendered that is closer at the camera. This method works good, but unfortunately we also waste time for shading of parts of the scene, that will not be seen. What we actually would need, is the result of the last shading computation of each pixel.

This thought is the central idea behind Deferred Shading [Saito and Takahashi 1990]. Here the rendering is split into to steps, first a rasterisation step and then the shading step. In the rasterisation step, we create a so-called G-buffer (geometry buffer) that will contain positions, normals and colours for all pixels. Then we rasterise the

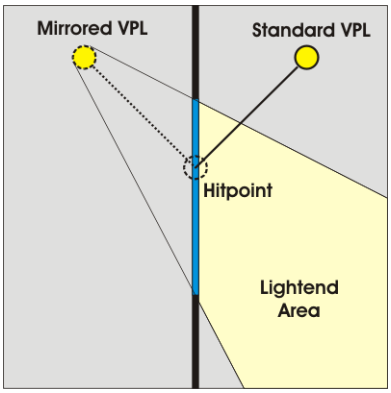


Figure 19: When a particle hits a specular surface (the blue one) we do not use the hit point as a VPL. Instead we mirror the origin VPL at the surface and use this 'Mirror'-VPL as common. We just have to mind that this VPL just illuminates a limited area.

scene and store the corresponding positions, normals and colours into the G-buffer. Now that we have this data, we can use it to do the shading for each pixel. This way we save time because we just need one shading calculation per pixel.

Deferred Shading is perfectly suited to be combined with Instant Radiosity. As we remember, within Instant Radiosity we have to render the scene multiple times per frame, each time with a different VPL as actual light source. But cause it is the same frame, the scene will be static for all this renderings. So it is useful first to rasterise the scene and store it into a G-buffer. Then we can use this buffer for all renderings of the frame, so we save a lot of time by avoiding the scene to be rasterised multiple times.

4.5 Interleaved Sampling and Discontinuity Buffering

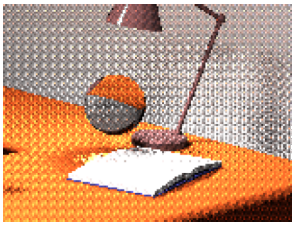
Instant Radiosity is a good method to approximate the indirect illumination in a scene. But since it uses a discretisation of the light distribution, in form of VPLs, we can see discretisation artefacts. Each VPLs brings a hard shadow into our scene (see figure 20(a)), and although the shadows of the VPLs overlap in the final image, they are still visible.

One method to get rid of this hard shadows would be to create more and more VPLs, so the hard shadows would be closer together so they would not be that visible any more. But of course, creating more VPLs is costly so we need an other approach. Another method would be to break the correlation between neighbouring pixels by giving all of them different VPLs. So the hard shadows would disappear, but of course giving all pixels different VPLs would not be possible.

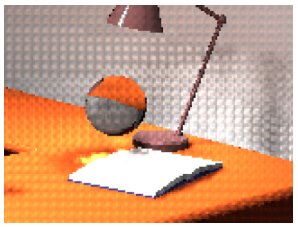
A solution to this would be Interleaved Sampling [Keller and Heidrich 2001]. Instead of giving all pixels different VPLs, we just create a small sample pattern (e.g. a 3x3 pattern) and lay it interleaved over the pixels (see figure 21), so that neighbouring pixels would not have the same VPLs any more. And we would not to have to create that more VPLs as so far. So we can remove the hard shadows in a scene in a very simple way. But this method has a price, instead of discretisation artefacts we now have structured noise in our scene (see figure 20(b)).

This structured noise is still a problem, so we want to remove it. As simple way would be to blur the resulting image. This method

(a) Just Instant Radiosity (IR)



(b) IR + Interleaved Sampling (ILS) 5x5



(c) IR + ILS 5x5 + Discontinuity Buffer 3x3

(d) IR + ILS 5x5 + Discontinuity Buffer 5x5

Figure 20: Section of a scene. (a) We see the hard shadows created by different VPLs. (b) We can get rid of this hard shadows by using interleaved sampling, so the correlation between neighbouring pixels is broken and the hard shadows disappear. Unfortunately now we have structured noise in our scene. (c)-(d) To remove the structured noise we can use a Discontinuity Buffer to find continuous regions and mean the irradiances. So the structured noise is decreased. (d) When we use the same pattern size for calculating the mean irradiance as we use for interleaved sampling, the structured noise completely disappears.

would work, but unfortunately we can not use it cause we want a sharp image. A better solution would be to use a Discontinuity Buffer [Keller and Heidrich 2001] to find continuous regions in the image. These continuous regions might belong to the same polygon, so we can expect the irradiance there as a piecewise smooth function. So we can use the neighbours of a pixel that are continuous and mean their irradiance. This mean irradiance is then used to lighten the pixel (see figure 22). The great advantage of this method is that we just calculate the mean of pixels that are somehow correlated and we just mean the irradiance and not the colour value itself, so the image will stay sharp.

4.6 Lightcuts

We would gain the best results with instant radiosity, by using as much VPLs as possible. But if we would do that, rendering would take longer, so its not practicable. One method that deals with large numbers of light sources is called Lightcuts [Walter et al. 2005].

The central idea behind lightcuts is to group light sources into clusters and to use the cluster as light sources instead of the individual light sources. We form the clusters by creating a so-called binary tree where all leaves are the light sources, that tree is called a light tree. A node of the light tree is a cluster, containing all the light sources of his children. The root node of course, would be a cluster that contains all light sources.

The light tree is now used to partition the light sources into clusters. It is simple to see that each cut through the tree would form a valid partition of of the light sources. We start with a simple cut, and then progressive refine it, until an error criterion is met.

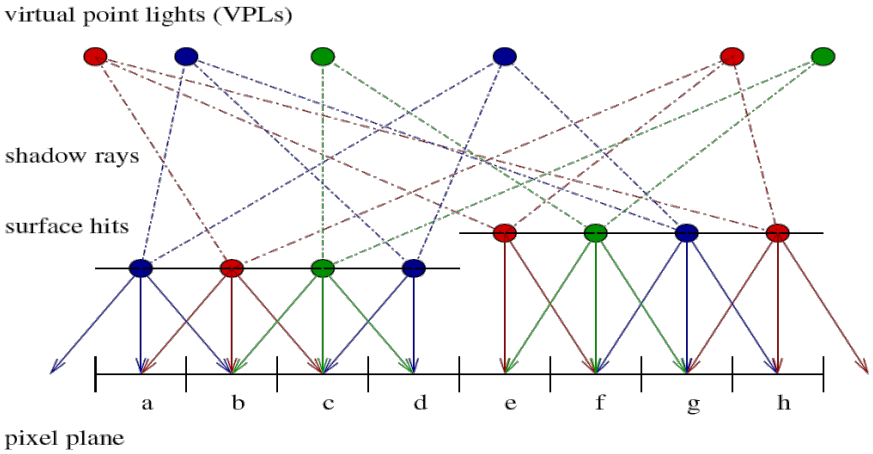


Figure 22: To remove the structured noise, caused by Interleaved Sampling, we use a Discontinuity Buffer to find continuous regions in the scene (here a-d and e-h form two continuous regions). Then we use the irradiances of the actual pixel and of the neighbouring pixels that are continuous and calculate a mean irradiance. This mean irradiance is then used to calculate the lighting for the pixel. That way the structured noise can be removed while keeping the image sharp.

Using lightcuts allows us to use more light sources than before, without great errors (see figure 23). So this method is very useful for instant radiosity and his derivations, since they need many light sources to calculate the indirect illumination in a scene.

5 Conclusion

Indirect lighting as part of global illumination is very important for realistic rendering. In this paper we presented the method of instant radiosity, that allows us to approximate the indirect illumination in a scene with additional light sources. Furthermore this method is faster than common global illumination methods so it can achieve realistic results in real-time, and it does not need a lot of preprocessing or large memory for its calculations.

Instant radiosity has been extended from multiple methods, that improve image quality or rendering speed. Also an method, called interactive global illumination, has been presented, which builds upon instant radiosity and a fast ray tracer. With this interactive global illumination and a cluster of common PCs it is possible to gain interactive framerates while having global illumination. Though we still need a cluster of computers, future developments might lead to a time where we can use such ray tracing based methods on single PCs.

Future developments will have to be spent to make instant radiosity or its derivations faster. Though they can achieve interactive framerates by now, they are not fast enough to be used within an interactive application like computer games.

References

ARVO, J. R. 1986. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, vol. 12.

BENTHIN, C., WALD, I., AND SLUSALLEK, P. 2003. A Scalable Approach to Interactive Global Illumination. *Computer Graphics Forum 22*, 3, 621–630. (Proceedings of Eurographics).

COHEN, M. F., WALLACE, J., AND HANRAHAN, P. 1993. *Radiosity and realistic image synthesis*. Academic Press Professional, Inc., San Diego, CA, USA.

KAJIYA, J. T. 1986. The rendering equation. *SIGGRAPH Comput. Graph. 20*, 4, 143–150.

KELLER, A., AND HEIDRICH, W. 2001. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, Springer-Verlag, London, UK, 269–276.

KELLER, A. 1996. Quasi-monte carlo radiosity. In *Proceedings of the eurographics workshop on Rendering techniques '96*, Springer-Verlag, London, UK, 101–110.

KELLER, A. 1997. Instant radiosity. *Computer Graphics 31*, Annual Conference Series, 49–56.

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J., AND AILA, T. 2007. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007*, Eurographics Association, xx–yy.

NIEDERREITER, H. 1992. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

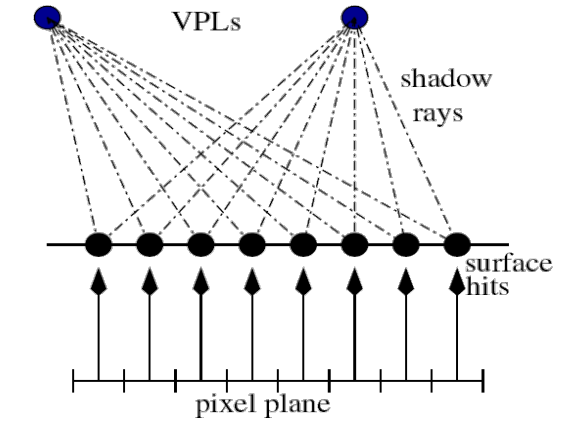
SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph. 24*, 4, 197–206.

SEGOVIA, B., IEHL, J.-C., MITANCHEY, R., AND PÉROCHE, B. 2006. Bidirectional instant radiosity. In *Proceedings of the 17th Eurographics Workshop on Rendering, to appear*.

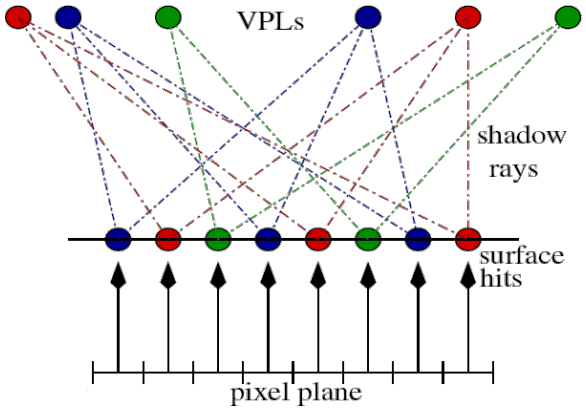
SEGOVIA, B., IEHL, J.-C., AND PÉROCHE, B. 2007. Metropolis instant radiosity. In *Proceedings of Eurographics 2007, to appear*.

TABELLION, E., AND LAMORLETTE, A. 2004. An approximate global illumination system for computer generated films. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 469–476.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. *Computer Graphics 31*, Annual Conference Series, 65–76.



(a) Just Instant Radiosity



(b) Instant Radiosity + Interleaved Sampling

Figure 21: (a) Instant Radiosity uses for all parts of the scene the same VPLs. This causes discretisation artefacts in form of hard shadows. (b) To remove the hard shadows, we use Interleaved Sampling. We use different sets of VPLs. When we render a part of the scene, we decide upon the corresponding pixel which set to be used. The sets are interleaved over the window so neighbouring pixels do not share the same VPLs any more.

(a) Lightcut Image, 128 seconds needed for rendering

(b) Reference Image, 1096 seconds needed for rendering

(c) Error Image

(d) 16 x Error Image

Figure 23: Kitchen scene lightened by 72 area light sources. Each area light is approximated by 64 point lights, in sum 4608 point lights. The scene is rendered twice, once with the light cut approach and once without. There is no practical difference perceivable, but the rendering time using lightcuts is much shorter.

W., J. S. L. F. L. W. H. 2000. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 121–134.

WALD, I., BENTHIN, C., WAGNER, M., AND SLUSALLEK, P. 2001. Interactive rendering with coherent ray tracing. In *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001*, Blackwell Publishers, Oxford, A. Chalmers and T.-M. Rhyne, Eds., vol. 20, 153–164. available at http://graphics.cs.uni-sb.de/ wald/Publications.

WALD, I., SLUSALLEK, P., AND BENTHIN, C. 2001. Interactive distributed ray tracing of highly complex models. In *Rendering Techniques 2001 (Proceedings of the 12th EUROGRAPHICS Workshop on Rendering*, 277–288.

WALD, I., BENTHIN, C., AND SLUSALLEK, P. 2002. OpenRT – A Flexible and Scalable Rendering Engine for Interactive 3D Graphics. Tech. rep., Saarland University.

WALD, I., BENTHIN, C., AND SLUSALLEK, P. 2002. A simple and practical method for interactive ray tracing of dynamic scenes. Tech. rep., Saarland University.

WALD, I., KOLLIG, T., BENTHIN, C., KELLER, A., AND SLUSALLEK, P. 2002. Interactive Global Illumination using Fast Ray Tracing. In *Proceedings of the 13th EUROGRAPHICS Workshop on Rendering*, Saarland University, Kaiserslautern University. avail.at http://www.openrt.de.

WALD, I., BENTHIN, C., AND SLUSALLEK, P. 2003. Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proceedings of the 14th Eurographics Workshop on Rendering*.

WALD, I. 2004. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph. 24*, 3, 1098–1107.

WHITTED, T. 1980. An improved illumination model for shaded display. *Commun. ACM 23*, 6, 343–349.