A Practical Guide to Global Illumination using Photon Maps

Siggraph 2000 Course 8

Sunday, July 23, 2000

Organizer

Henrik Wann Jensen
Stanford University

Lecturers

Niels Jørgen Christensen
Technical University of Denmark

Henrik Wann Jensen
Stanford University

Abstract

This course serves as a practical guide to photon maps. Any reader who can implement a ray tracer should be able to add an efficient implementation of photon maps to his or her ray tracer after attending this course and reading the course notes.

There are many reasons to augment a ray tracer with photon maps. Photon maps makes it possible to efficiently compute global illumination including caustics, diffuse color bleeding, and participating media. Photon maps can be used in scenes containing many complex objects of general type (e.g. the method is not restricted to tessellated models). The method is capable of handling advanced material descriptions based on a mixture of specular, diffuse, and non-diffuse components. Furthermore, the method is easy to implement and experiment with.

This course is structured as a two hour tutorial. We will therefore assume that the participants have knowledge of global illumination algorithms (in particular ray tracing), material models, and radiometric terms such as radiance and flux. We will discuss in detail photon tracing, the photon map data structure, the photon map radiance estimate, and rendering techniques based on photon maps. We will emphasize the techniques for efficient computation throughout the presentation. Finally, we will present some examples of scenes rendered with photon maps and explain the important aspects that we considered when rendering each scene.

Lecturers

Niels Jørgen Christensen

Associate professor
Department of Graphical Communication
Technical University of Denmark
Building 116
2800 Lyngby
Denmark
njc@gk.dtu.dk

Professor Niels Jørgen Christensen has been conducting research in computer graphics including global illumination, virtual reality, parallel rendering techniques and scientific visualization at the Technical University of Denmark for the last 20 years. He was the Ph.D. advisor for Henrik Wann Jensen and co-author on the first papers on photon maps.

Henrik Wann Jensen

Research Associate
Computer Graphics Laboratory
Computer Science 362B
Stanford University
CA 94305
henrik@graphics.stanford.edu
http://graphics.stanford.edu/~henrik

Henrik Wann Jensen is a research associate in the computer graphics laboratory at Stanford where he is working on light scattering and global illumination techniques for complex environments and parallel rendering algorithms. Prior to that he was a postdoc in the computer graphics group at MIT. He received his M.Sc. in 1993 and his Ph.D. in 1996 at the Technical University of Denmark for developing the photon map algorithm.

Course Syllabus

5 minutes: Introduction and Welcome

Henrik Wann Jensen

Why you should attend this course. Overview of the topics.

30 minutes: Overview of Existing Global Illumination Algorithms

Niels Jørgen Christensen

A brief overview of the photon map algorithm including a description of the important differences compared to other global illumination algorithms such as finite element methods (radiosity) and brute force Monte Carlo ray tracing.

30 minutes: Photon Tracing: Building the Photon Maps

Niels Jørgen Christensen & Henrik Wann Jensen

This part of the tutorial will cover efficient techniques for:

- Emitting photons from the light sources in the scene using projection maps
- Simulating scattering and absorption of photons using Russian Roulette
- Storing photons in the photon map
- Preparing the photon map for rendering

Also the use of several photon maps for the simulation of caustics, indirect illumination, and participating media will be described.

55 minutes: Rendering using Photon Maps

Henrik Wann Jensen

The third part of the tutorial will describe how the photon maps are used to simulate global illumination. This part will give details on how to compute a radiance estimate based on the photon map, and how to filter this estimate to obtain better quality with fewer photons. Also described will be techniques for efficiently locating the nearest photons. The rendering pass will be detailed with a description of how the rendering equation is split into several components that each can be rendered using specialized techniques based on the photon maps. This includes methods for rendering caustics, indirect illumination, and participating media.

Finally, we will give some examples of different scenes rendered using photon maps, describe how the photon maps were used, and discuss the issues that are important to ensure good quality and fast results.

Contents

Foreword

Welcome to this Siggraph course on photon maps!

If you find this course material exciting, we encourage you to visit the following web-site where we will add new information based on the course:

`http://www.gk.dtu.dk/photonmap/`

The inspiration behind this course is Alan Chalmers. Without his suggestion (over a beer) this course might never have materialized. Thanks, Alan! Thanks to Per Christensen for moral support and for writing parts of the chapter on photon tracing. In addition, thanks to Martin Blais, Byong Mok Oh, Gernot Schaufler and Maryann Simmons for helpful comments on the notes. Finally, we would like to thank intellectual property counsel Karen Hersey at Massachusetts Institute of Technology.

Introduction

This course material describes in detail the practical aspects of the photon map algorithm. The text is based on previously published papers, technical reports and dissertations (in particular [Jensen96c]). It also reflects the experience obtained with the implementation of the photon map as it was developed at the Technical University of Denmark. After reading this course material, it should be relatively straightforward to add an efficient implementation of the photon map algorithm to any ray tracer.

0.1 Motivation

The photon map method is an extension of ray tracing. In 1989, Andrew Glassner wrote about ray tracing [Glassner89]:

> "Today ray tracing is one of the most popular and powerful techniques in the image synthesis repertoire: it is simple, elegant, and easily implemented. [However] there are some aspects of the real world that ray tracing doesn't handle very well (or at all!) as of this writing. Perhaps the most important omissions are diffuse inter-reflections (e.g. the 'bleeding' of colored light from a dull red file cabinet onto a white carpet, giving the carpet a pink tint), and caustics (focused light, like the shimmering waves at the bottom of a swimming pool)."

At the time of the development of the photon map algorithm in 1993, these problems were still not addressed efficiently by any ray tracing algorithm. The photon map method offers a solution to both problems. Diffuse interreflections and caustics are both indirect illumination of diffuse surfaces; with the photon map method, such illumination is estimated using precomputed photon maps. Extending ray tracing with photon maps yields a method capable of efficiently simulating all types of direct and indirect illumination. Furthermore, the photon map method can handle participating media and it is fairly simple to parallelize [Jensen00].

0.2 What is the photon map algorithm?

The photon map algorithm was developed in 1993–1994 and the first papers on the method were published in 1995. It is a versatile algorithm capable of simulating

global illumination including caustics, diffuse interreflections, and participating media in complex scenes. It provides the same flexibility as general Monte Carlo ray tracing methods using only a fraction of the computation time.

The global illumination algorithm based on photon maps is a two-pass method. The first pass builds the photon map by emitting photons from the light sources into the scene and storing them in a *photon map* when they hit non-specular objects. The second pass, the rendering pass, uses statistical techniques on the photon map to extract information about incoming flux and reflected radiance at any point in the scene. The photon map is decoupled from the geometric representation of the scene. This is a key feature of the algorithm, making it capable of simulating global illumination in complex scenes containing millions of triangles, instanced geometry, and complex procedurally defined objects.

Compared with finite element radiosity, photon maps have the advantage that no meshing is required. The radiosity algorithm is faster for simple diffuse scenes but as the complexity of the scene increases, photon maps tend to scale better. Also the photon map method handles non-diffuse surfaces and caustics.

Monte Carlo ray tracing methods such as path tracing, bidirectional path tracing, and Metropolis can simulate all global illumination effects in complex scenes with very little memory overhead. The main benefit of the photon map compared with these methods is efficiency, and the price paid is the extra memory used to store the photons. For most scenes the photon map algorithm is significantly faster, and the result looks better since the error in the photon map method is of low frequency which is less noticeable than the high frequency noise of general Monte Carlo methods.

Another big advantage of photon maps (from a commercial point of view) is that there is no patent on the method; anyone can add photon maps to their renderer. As a result several commercial systems use photon maps for rendering caustics and global illumination.

0.3 Overview of the course material

Section 1 describes emission, tracing, and storing of photons. Section 2 describes how to organize the photons in a balanced kd-tree for improved performance in the rendering step. The radiance estimate based on photons is outlined in section 3. This section also contains information on how to filter the estimate

to obtain better quality and it contains a description of how to locate photons efficiently given the balanced kd-tree. The rendering pass is presented in section 4 with information on how to split the rendering equation and use the photon map to efficiently compute different parts of the equation. Finally in section 5 we give a number of examples of scenes rendered with the photon map algorithm.

A Practical Guide to Global Illumination using Photon Maps

1 Photon tracing

The purpose of the photon tracing pass is to compute indirect illumination on diffuse surfaces. This is done by emitting photons from the light sources, tracing them through the scene, and storing them at diffuse surfaces.

1.1 Photon emission

This section describes how photons are emitted from a single light source and from multiple light sources, and describes the use of projection maps which can increase the emission efficiency considerably.

1.1.1 Emission from a single light source

The photons emitted from a light source should have a distribution corresponding to the distribution of emissive power of the light source. This ensures that the emitted photons carry the same flux — ie. we do not waste computational resources on photons with low power.

Photons from a diffuse point light source are emitted in uniformly distributed random directions from the point. Photons from a directional light are all emitted in the same direction, but from origins outside the scene. Photons from a diffuse square light source are emitted from random positions on the square, with directions limited to a hemisphere. The emission directions are chosen from a cosine distribution: there is zero probability of a photon being emitted in the direction parallel to the plane of the square, and highest probability of emission is in the direction perpendicular to the square.

In general, the light source can have any shape and emission characteristics — the intensity of the emitted light varies with both origin and direction. For example, a (matte) light bulb has a nontrivial shape and the intensity of the light emitted from it varies with both position and direction. The photon emission should follow this variation, so in general, the probability of emission varies depending on the position on the surface of the light source and the direction.

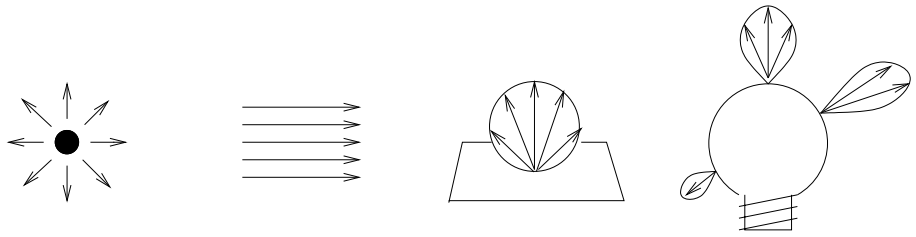


Figure 1: Emission from light sources: (a) point light, (b) directional light, (c) square light, (d) general light.

Figure 1 shows the emission from these different types of light sources.

The power ("wattage") of the light source has to be distributed among the photons emitted from it. If the power of the light is P_{light} and the number of emitted photons is n_e, the power of each emitted photon is

$$P_{photon} = \frac{P_{light}}{n_e} \,. \tag{1}$$

Pseudocode for a simple example of photon emission from a diffuse point light source is given in Figure 2.

```
emit_photons_from_diffuse_point_light() {
  n_e = 0                 number of emitted photons
  while (not enough photons) {
    do {                  use simple rejection sampling to find diffuse photon direction
      x = random number between -1 and 1
      y = random number between -1 and 1
      z = random number between -1 and 1
    } while ( x^2 + y^2 + z^2 > 1 )

    d⃗ = < x, y, z >
    p⃗ = light source position
    trace photon from p⃗ in direction d⃗
    n_e = n_e + 1
  }
  scale power of stored photons with 1/n_e
}
```

Figure 2: Pseudocode for emission of photons from a diffuse point light

To further reduce variation in the computed indirect illumination (during rendering), it is desirable that the photons are emitted as evenly as possible.

This can for example be done with stratification [Rubinstein81] or by using low-discrepancy quasi-random sampling [Keller96].

1.1.2 Multiple lights

If the scene contains multiple light sources, photons should be emitted from each light source. More photons should be emitted from brighter lights than from dim lights, to make the power of all emitted photons approximately even. (The information in the photon map is best utilized if the power of the stored photons is approximately even). One might worry that scenes with many light sources would require many more photons to be emitted than scenes with a single light source. Fortunately, it is not so. In a scene with many light sources, each light contributes less to the overall illumination, and typically fewer photons can be emitted from each light. If, however, only a few light sources are important one might use an importance sampling map [Peter98] to concentrate the photons in the areas that are of interest to the observer. The tricky part about using an importance map is that we do not want to generate photons with energy levels that are too different since this will require a larger number of photons in the radiance estimate (see section 3) to ensure good quality.

1.1.3 Projection maps

In scenes with sparse geometry, many emitted photons will not hit any objects. Emitting these photons is a waste of time. To optimize the emission, *projection map*s can be used [Jensen93, Jensen95a]. A projection map is simply a map of the geometry as seen from the light source. This map consists of many little cells. A cell is "on" if there is geometry in that direction, and "off" if not. For example, a projection map is a spherical projection of the scene for a point light, and it is a planar projection of the scene for a directional light. To simplify the projection it is convenient to project the bounding sphere around each object or around a cluster of objects [Jensen95a]. This also significantly speeds up the computation of the projection map since we do not have to examine every geometric element in the scene. The most important aspect about the projection map is that it gives a conservative estimate of the directions in which it is necessary to emit photons from the light source. Had the estimate not been conservative (e.g. we could have sampled the scene with a few photons first), we could risk missing

important effects, such as caustics.

The emission of photons using a projection map is very simple. One can either loop over the cells that contain objects and emit a random photon into the directions represented by the cell. This method can, however, lead to slightly biased results since the photon map can be "full" before all the cells have been visited. An alternative approach is to generate random directions and check if the cell corresponding to that direction has any objects (if not a new random direction should be tried). This approach usually works well, but it can be costly in sparse scenes. For sparse scenes it is better to generate photons randomly for the cells which have objects. A simple approach is to pick a random cell with objects and then pick a random direction for the emitted photon for that cell [Jensen93]. In all circumstances it is necessary to scale the energy of the stored photons based on the number of active cells in the projection map and the number of photons emitted [Jensen93]. This leads to a slight modification of formula 1:

$$P_{photon} = \frac{P_{light}}{n_e} \frac{\text{cells with objects}}{\text{total number of cells}} . \quad (2)$$

Another important optimization for the projection map is to identify objects with specular properties (i.e. objects that can generate caustics) [Jensen93]. As it will be described later, caustics are generated separately, and since specular objects often are distributed sparsely it is very beneficial to use the projection map for caustics.

1.2 Photon tracing

Once a photon has been emitted, it is traced through the scene using photon tracing (also known as "light ray tracing", "backward ray tracing", "forward ray tracing", and "backward path tracing"). Photon tracing works in exactly the same way as ray tracing except for the fact that photons propagate flux whereas rays gather radiance. This is an important distinction since the interaction of a photon with a material can be different than the interaction of a ray. A notable example is refraction where radiance is changed based on the relative index of refraction[Hall88] — this does not happen to photons.

When a photon hits an object, it can either be reflected, transmitted, or absorbed. Whether it is reflected, transmitted, or absorbed is decided probabilistically based on the material parameters of the surface. The technique used to

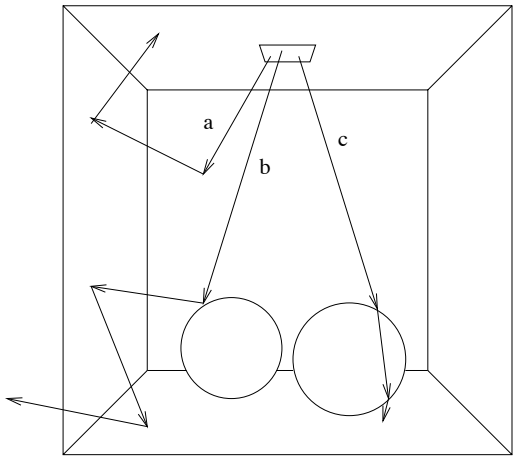


Figure 3: Photon paths in a scene (a "Cornell box" with a chrome sphere on left and a glass sphere on right): (a) two diffuse reflections followed by absorption, (b) a specular reflection followed by two diffuse reflections, (c) two specular transmissions followed by absorption.

decide the type of interaction is known as Russian roulette [Arvo90] — basically we roll a dice and decide whether the photon should survive and be allowed to perform another photon tracing step.

Examples of photon paths are shown in Figure 3.

1.2.1 Reflection, transmission, or absorption?

For a simple example, we first consider a monochromatic simulation. For a reflective surface with a diffuse reflection coefficient d and specular reflection coefficient s (with $d + s \leq 1$) we use a uniformly distributed random variable $\xi \in [0, 1]$ (computed with for example `drand48()`) and make the following decision:

$$\begin{array}{lcl} \xi \in [0, d] & \longrightarrow & \text{diffuse reflection} \\ \xi \in]d, s + d] & \longrightarrow & \text{specular reflection} \\ \xi \in]s + d, 1] & \longrightarrow & \text{absorption} \end{array}$$

In this simple example, the use of Russian roulette means that we do not have to modify the power of the reflected photon — the correctness is ensured by averaging several photon interactions over time. Consider for example a surface that reflects 50incoming light. With Russian roulette only half of the incoming photons will be reflected, but with full energy. For example, if you shoot 1000 photons at the surface, you can either reflect 1000 photons with half the energy

or 500 photons with full energy. It can be seen that Russian roulette is a powerful technique for reducing the computational requirements for photon tracing.

With more color bands (for example RGB colors), the decision gets slightly more complicated. Consider again a surface with some diffuse reflection and some specular reflection, but this time with different reflection coefficients in the three color bands. The probabilities for specular and diffuse reflection can be based on the total energy reflected by each type of reflection or on the maximum energy reflected in any color band. If we base the decision on maximum energy, we can for example compute the probability for reflection as

$$P_r = max(d_r + s_r, d_g + s_g, d_b + s_b)\,,$$

where d_r, d_g, and d_b are the diffuse reflection coefficients in the red, green, and blue color bands, and s_r, s_g, and s_b are the specular reflection coefficients in the red, green, and blue color bands. (The probability of absorption is $P_a = 1 - P_r$.) With this, we can compute the probability P_d of diffuse reflection as:

$$P_d = \frac{d_r + d_g + d_b}{d_r + d_g + d_b + s_r + s_g + s_b}\, P_r\,.$$

Similarly, for the probability P_s of specular reflection, we get:

$$P_s = \frac{s_r + s_g + s_b}{d_r + d_g + d_b + s_r + s_g + s_b}\, P_r = P_r - P_d\,.$$

With these probabilities, the decision of which type of reflection or absorption should be chosen takes the following form:

$$\begin{array}{lcl} \xi \in [0, P_d] & \longrightarrow & \text{diffuse reflection} \\ \xi \in]P_d, P_s + P_d] & \longrightarrow & \text{specular reflection} \\ \xi \in]P_s + P_d, 1] & \longrightarrow & \text{absorption} \end{array}$$

The power of the reflected photon needs to be adjusted to account for the probability of survival. If, for example, specular reflection was chosen in the example above, the power P_{refl} of the reflected photon is:

$$\begin{aligned} P_{refl,r} &= P_{inc,r}\, s_r/P_s \\ P_{refl,g} &= P_{inc,g}\, s_g/P_s \\ P_{refl,b} &= P_{inc,b}\, s_b/P_s \end{aligned}$$

where P_{inc} is the power of the incident photon.

The computed probabilities again ensure us that we do not waste time emitting photons with very low power.

It is simple to extend the selection scheme to also handle transmission, to handle more than three color bands, and to handle other reflection types (for example glossy and directional diffuse).

1.2.2 Why Russian roulette?

Why do we go through this effort to decide what to do with a photon? Why not just trace new photons in the diffuse and specular directions and scale the photon energy accordingly? There are two main reasons why the use of Russian roulette is a very good idea. Firstly, we prefer photons with similar power in the photon map. This makes the radiance estimate much better using only a few photons. Secondly, if we generate, say, two photons per surface interaction then we will have 2^8 photons after 8 interactions. This means 256 photons after 8 interactions compared to 1 photon coming directly from the light source! Clearly this is not good. We want at least as many photons that have only 1–2 bounces as photons that have made 5–8 bounces. The use of Russian roulette is therefore very important in photon tracing.

There is one caveat with Russian roulette. It increases variance on the solution. Instead of using the exact values for reflection and transmission to scale the photon energy we now rely on a sampling of these values that will converge to the correct result as enough photons are used.

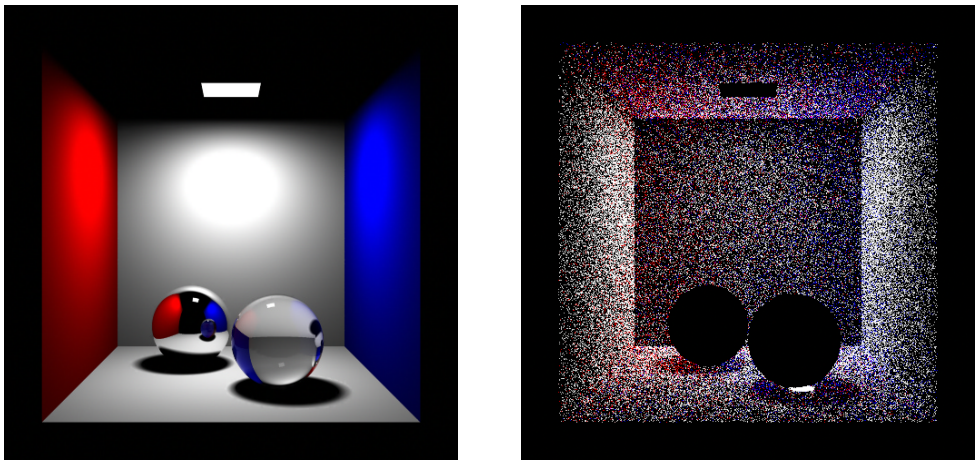
Details on photon tracing and Russian roulette can be found in [Shirley90, Pattanaik93, Glassner95].

1.3 Photon storing

This section describes which photon-surface interactions are stored in the photon map. It also describes in more detail the photon map data structure.

1.3.1 Which photon-surface interactions are stored?

Photons are only stored where they hit diffuse surfaces (or, more precisely, non-specular surfaces). The reason is that storing photons on specular surfaces does not give any useful information: the probability of having a matching incoming photon from the specular direction is zero, so if we want to render accurate



(a) (b)

Figure 4: "Cornell box" with glass and chrome spheres: (a) ray traced image (direct illumination and specular reflection and transmission), (b) the photons in the corresponding photon map.

specular reflections the best way is to trace a ray in the mirror direction using standard ray tracing. For all other photon-surface interactions, data is stored in a global data structure, the *photon map*. Note that each emitted photon can be stored several times along its path. Also, information about a photon is stored at the surface where it is absorbed if that surface is diffuse.

For each photon-surface interaction, the position, incoming photon power, and incident direction are stored. (For practical reasons, there is also space reserved for a flag with each set of photon data. The flag is used during sorting and look-up in the photon map. More on this in the following.)

As an example, consider again the simple scene from Figure 3, a "Cornell box" with two spheres. Figure 4(a) shows a traditional ray traced image (direct illumination and specular reflection and transmission) of this scene. Figure 4(b) shows the photons in the photon map generated for this scene. The high concentration of photons under the glass sphere is caused by focusing of the photons by the glass sphere.

1.3.2 Data structure

Expressed in `C` the following structure is used for each photon [Jensen96b]:

```
struct photon {
  float x,y,z;      // position
  char p[4];        // power packed as 4 chars
  char phi, theta;  // compressed incident direction
  short flag;       // flag used in kdtree
}
```

The power of the photon is represented compactly as 4 bytes using Ward's packed rgb-format [Ward91]. If memory is not of concern one can use 3 floats to store the power in the red, green, and blue color band (or, in general, one float per color band if a spectral simulation is performed).

The incident direction is a mapping of the spherical coordinates of the photon direction to 65536 possible directions. They are computed as:

```
phi = 255 * (atan2(dy,dx)+PI) / (2*PI)
theta = 255 * acos(dx) / PI
```

where `atan2` is from the standard C library. The direction is used to compute the contribution for non-Lambertian surfaces [Jensen96a], and for Lambertian surfaces it is used to check if a photon arrived at the front of the surface. Since the photon direction is used often during rendering it pays to have a lookup table that maps the theta, phi direction to three floats directly instead of using the formula for spherical coordinates which involves the use of the costly `cos()` and `sin()` functions.

A minor note is that the flag in the structure is a short. Only 2 bits of this flag are used (this is for the splitting plane axis in the kd-tree), and it would be possible to use just one byte for the flag. However for alignment reasons it is preferable to have a 20 byte photon rather than a 19 byte photon — on some architectures it is even a necessity since the float-value in subsequent photons must be aligned on a 4 byte address.

We might be able to compress the information more by using the fact that we know the cube in which the photon is located. The position is, however, used very often when the photons are processed and by using standard float we avoid the overhead involved in extracting the true position from a specialized format.

During the photon tracing pass the photon map is arranged as a flat array of photons. For efficiency reasons this array is re-organized into a balanced kdtree before rendering as explained in section 2.

1.4 Extension to participating media

Up to this point, all photon interactions have been assumed to happen at object surfaces; all volumes were implicitly assumed to not affect the photons. However, it is simple to extend the photon map method to handle *participating media*, i.e. volumes that participate in the light transport. In scenes with participating media, the photons are stored within the media in a seperate *volume photon map* [Jensen98].

1.4.1 Photon emission, tracing, and storage

Photons can be emitted from volumes as well as from surfaces and points. For example, the light from a candle flame can be simulated by emitting photons from a flame-shaped volume.

When a photon travels through a participating medium, it has a certain probability of being scattered or absorped in the medium. The probability depends on the density of the medium and on the distance the photon travels through the medium: the denser the medium, the shorter the average distance before a photon interaction happens. Photons are stored at the positions where a scattering event happens. The exception is photons that come directly from the light source since direct illumination is evaluated using ray tracing. This separation was introduced in [Jensen98] and it allows us to compute the in-scattered radiance in a medium simply by a lookup in the photon map.

As an example, consider a glass sphere in fog illuminated by directional light. Figure 5(a) shows a schematic diagram of the photon paths as photons are being focused by refraction in the glass sphere. Figure 5(b) shows the caustic photons stored in the photon map.

1.4.2 Multiple scattering, anisotropic scattering, and non-homogeneous media

The simple example above only shows the photon interaction in the fog after refraction by the glass sphere, and the photon paths are terminated at the first scattering event. General multiple scattering is simulated simply by letting the photons scatter everywhere and continuously after the first interaction. The path can be terminated using Russian roulette.

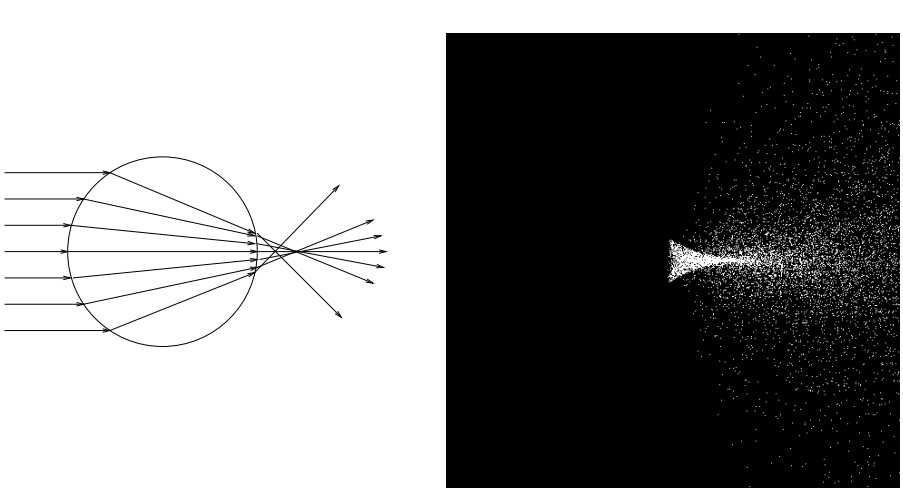


Figure 5: Sphere in fog: (a) schematic diagram of light paths, (b) the caustic photons in the photon map.

The fog in the example has uniform density, but it is not difficult to handle media with nonuniform density (aka. nonhomogeneous media), since we use ray marching to integrate the properties of the medium. A simple ray marcher works by dividing the medium into little steps [Ebert94]. The accumulated density (integrated extinction coefficient) is updated at each step, and based on a precomputed probability it is determined whether the photon should be absorbed, scattered, or whether another step is necessary.

For more complicated examples of scattering in participating media, including anisotropic and nonhomogeneous media and complex geometry, see [Jensen98].

1.5 Three photon maps

For efficiency reasons, it pays off to divide the stored photons into three photon maps:

Caustic photon map: contains photons that have been through at least one specular reflection before hitting a diffuse surface: LS^+D.

Global photon map: an approximate representation of the global illumination solution for the scene for all diffuse surfaces: $L\{S|D|V\}^*D$

Volume photon map: indirect illumination of a participating medium: $L\{S|D|V\}^+V$.

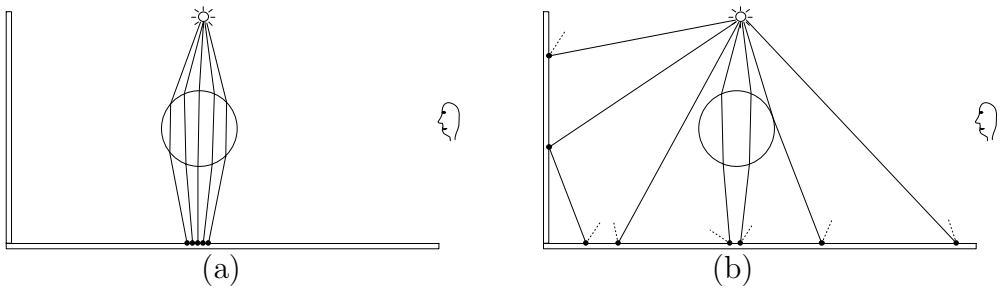


Figure 6: Building (a) the caustics photon map and (b) the global photon map.

Here, we used the grammar from [Heckbert90] to describe the photon paths: L means emission from the light source, S is specular reflection or transmission, D is diffuse (ie. non-specular) reflection or transmission, and V is volume scattering. The notation $\{x|y|z\}$ means "one of x, y, or z", x^+ means one or several repeats of x, and x^* means zero or several repeats of x.

The reason for keeping three separate photon maps will become clear in section 4. A separate photon tracing pass is performed for the caustic photon map since it should be of high quality and therefore often needs more photons than the global photon map and the volume photon map.

The construction of the photon maps is most easily achieved by using two separate photon tracing steps in order to build the caustics photon map and the global photon map (including the volume photon map). This is illustrated in Figure 6 for a simple test scene with a glass sphere and 2 diffuse walls. Figure 6(a) shows the construction of the caustics photon map with a dense distribution of photons, and Figure 6(b) shows the construction of the global photon map with a more coarse distribution of photons.

2 Preparing the photon map for rendering

Photons are only generated during the photon tracing pass — in the rendering pass the photon map is a static data structure that is used to compute estimates of the incoming flux and the reflected radiance at many points in the scene. To do this it is necessary to locate the nearest photons in the photon map. This is an operation that is done extremely often, and it is therefore a good idea to

optimize the representation of the photon map before the rendering pass such that finding the nearest photons is as fast as possible.

First, we need to select a good data structure for representing the photon map. The data structure should be compact and at the same time allow for fast nearest neighbor searching. It should also be able to handle highly non-uniform distributions — this is very often the case in the caustics photon map. A natural candidate that handles these requirements is *a balanced kd-tree* [Bentley75]. Examples of using a balanced versus an unbalanced kd-tree can be found in [Jensen96a].

2.1 The balanced kd-tree

The time it takes to locate one photon in a balanced kd-tree has a worst time performance of $O(\log N)$, where N is the number of photons in the tree. Since the photon map is created by tracing photons randomly through a model one might think that a dynamically built kd-tree would be quite well balanced already. However, the fact that the generation of the photons at the light source is based on the projection map combined with the fact that models often contain highly directional reflectance models easily results in a skewed tree. Since the tree is created only once and used many times during rendering it is quite natural to consider balancing the tree. Another argument that is perhaps even more important is the fact that a balanced kd-tree can be represented using a heap-like data-structure [Sedgewick92] which means that explicitly storing the pointers to the sub-trees at each node is no longer necessary. (Array element 1 is the tree root, and element i has element $2i$ as left child and element $2i+1$ as right child.) This can lead to considerable savings in memory when a large number of photons is used.

2.2 Balancing

Balancing a kd-tree is similar to balancing a binary tree. The main difference is the choice at each node of a splitting dimension. When a splitting dimension of a set is selected, the median of the points in that dimension is chosen as the root node of the tree representing the set and the left and right subtrees are constructed from the two sets separated by the median point. The choice of a splitting dimension is based on the distribution of points within the set. One might use either the variance or the maximum distance between the points as

```
kdtree *balance( points ) {
  Find the cube surrounding the points
  Select dimension dim in which the cube is largest
  Find median of the points in dim
  s1 = all points below median
  s2 = all points above median
  node = median
  node.left = balance( s1 )
  node.right = balance( s2 )
  return node
}
```

Figure 7: Pseudocode for balancing the photon map

a criterion. We prefer a choice based upon maximum distance since it can be computed very efficiently (even though a choice based upon variance might be slightly better). The splitting dimension is thus chosen as the one which has the largest maximum distance between the points.

Figure 7 contains a pseudocode outline for the balancing algorithm [Jensen96c].

To speed up the balancing process it is convenient to use an array of pointers to the photons. In this way only pointers needs to be shuffled during the median search. An efficient median search algorithm can be found in most textbooks on algorithms — see for example [Sedgewick92] or [Cormen89].

The complexity of the balancing algorithm is $O(N \log N)$ where N is the number of photons in the photon map. In practice, this step only takes a few seconds even for several million photons.

3 The radiance estimate

A fundamental component of the photon map method is the ability to compute radiance estimates at any non-specular surface point in any given direction.

3.1 Radiance estimate at a surface

The photon map can be seen as a representation of the incoming flux; to compute radiance we need to integrate this information. This can be done using the

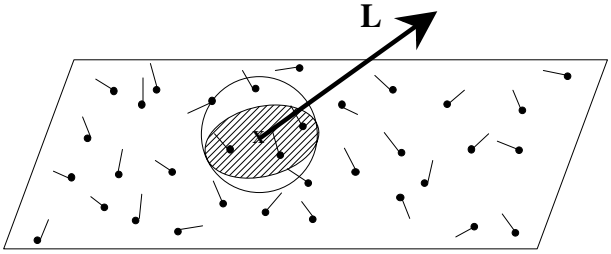


Figure 8: Radiance is estimated using the nearest photons in the photon map.

expression for reflected radiance:

$$L_r(x,\vec{\omega}) = \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') |\vec{n}_x \cdot \vec{\omega}'| \, d\omega_i' \,, \tag{3}$$

where L_r is the reflected radiance at x in direction $\vec{\omega}$. Ω_x is the (hemi)sphere of incoming directions, f_r is the BRDF (bidirectional reflectance distribution function) [Nicodemus77] at x and L_i is the incoming radiance. To evaluate this integral we need information about the incoming radiance. Since the photon map provides information about the incoming flux we need to rewrite this term. This can be done using the relationship between radiance and flux:

$$L_i(x,\vec{\omega}') = \frac{d^2\Phi_i(x,\vec{\omega}')}{\cos\theta_i \, d\omega_i' \, dA_i} \,, \tag{4}$$

and we can rewrite the integral as

$$\begin{aligned} L_r(x,\vec{\omega}) &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) \frac{d^2\Phi_i(x,\vec{\omega}')}{\cos\theta_i \, d\omega_i' \, dA_i} |\vec{n}_x \cdot \vec{\omega}'| \, d\omega_i' \\ &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) \frac{d^2\Phi_i(x,\vec{\omega}')}{dA_i} \,. \end{aligned} \tag{5}$$

The incoming flux Φ_i is approximated using the photon map by locating the n photons that has the shortest distance to x. Each photon p has the power $\Delta\Phi_p(\vec{\omega}_p)$ and by assuming that the photons intersects the surface at x we obtain

$$L_r(x,\vec{\omega}) \approx \sum_{p=1}^{n} f_r(x,\vec{\omega}_p,\vec{\omega}) \frac{\Delta\Phi_p(x,\vec{\omega}_p)}{\Delta A} \,. \tag{6}$$

The procedure can be imagined as expanding a sphere around x until it contains n photons (see Figure 8) and then using these n photons to estimate the radiance.

Equation 6 still contains ΔA which is related to the density of the photons around x. By assuming that the surface is locally flat around x we can compute this area by projecting the sphere onto the surface and use the area of the resulting circle. This is indicated by the hatched area in Figure 8 and equals:

$$\Delta A = \pi r^2 \,, \tag{7}$$

where r is the radius of the sphere – ie. the largest distance between x and each of the photons.

This results in the following equation for computing reflected radiance at a surface using the photon map:

$$L_r(x, \vec{\omega}) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) \,. \tag{8}$$

This estimate is based on many assumptions and the accuracy depends on the number of photons used in the photon map and in the formula. Since a sphere is used to locate the photons one might easily include wrong photons in the estimate in particular in corners and at sharp edges of objects. Edges and corners also causes the area estimate to be wrong. The size of those regions in which these errors occur depends largely on the number of photons in the photon map and in the estimate. As more photons are used in the estimate and in the photon map, formula 8 becomes more accurate. If we ignore the error due to limited accuracy of the representation of the position, direction and flux, then we can go to the limit and increase the number of photons to infinity. This gives the following interesting result where N is the number of photons in the photon map:

$$\lim_{N\to\infty} \frac{1}{\pi r^2} \sum_{p=1}^{\lfloor N^\alpha \rfloor} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) = L_r(x, \vec{\omega}) \quad \text{for } \alpha \in]0, 1[\,. \tag{9}$$

This formulation applies to all points x located on a locally flat part of a surface for which the BRDF, does not contain the Dirac delta function (this excludes perfect specular reflection). The principle in equation 9 is that not only will an infinite amount of photons be used to represent the flux within the model

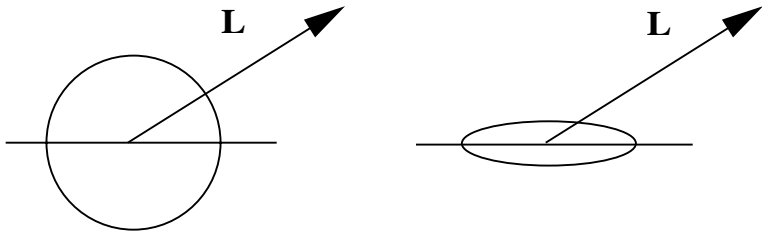


Figure 9: Using a sphere (left) and using a disc (right) to locate the photons.

but an infinite amount of photons will also be used to estimate radiance and the photons in the estimate will be located within an infinitesimal sphere. The different degrees of infinity are controlled by the term N^{α} where $\alpha \in]0,1[$. This ensures that the number of photons in the estimate will be infinitely fewer than the number of photons in the photon map.

Equation 9 means that we can obtain arbitrarily good radiance estimates by just using enough photons! In finite element based approaches it is more complicated to obtain arbitrary accuracy since the error depends on the resolution of the mesh, the resolution of the directional representation of radiance and the accuracy of the light simulation.

Figure 8 shows how locating the nearest photons is similar to expanding a sphere around x and using the photons within this sphere. It is possible to use other volumes than the sphere in this process. One might use a cube instead, a cylinder or perhaps a disc. This could be useful to either obtain an algorithm that is faster at locating the nearest photons or perhaps more accurate in the selection of photons. If a different volume is used then ΔA in equation 6 should be replaced by the area of the intersection between the volume and the tangent plane touching the surface at x. The sphere has the obvious advantage that the projected area and the distance computations are very simple and thus efficiently computed. A more accurate volume can be obtained by modifying the sphere into a disc (ellipsoid) by compressing the sphere in the direction of the surface normal at x (shown in Figure 9) [Jensen96c]. The advantage of using a disc would be that fewer "false photons" are used in the estimate at edges and in corners. This modification works pretty well at the edges in a room, for instance, since it prevents photons on the walls to leak down to the floor. One issue that still

occurs, however, is that the area estimate might be wrong or photons may leak into areas where they do not belong. This problem is handled primarily by the use of filtering.

3.2 Filtering

If the number of photons in the photon map is too low, the radiance estimates becomes blurry at the edges. This artifact can be pleasing when the photon map is used to estimate indirect illumination for a distribution ray tracer (see section 4 and Figure 15) but it is unwanted in situations where the radiance estimate represents caustics. Caustics often have sharp edges and it would be nice to preserve these edges without requiring too many photons.

To reduce the amount of blur at edges, the radiance estimate is filtered. The idea behind filtering is to increase the weight of photons that are close to the point of interest, x. Since we use a sphere to locate the photons it would be natural to assume that the filters should be three-dimensional. However, photons are stored at surfaces which are two-dimensional. The area estimate is also based on the assumption that photons are located on a surface. We therefore need a 2d-filter (similar to image filters) which is normalized over the region defined by the photons.

The idea of filtering caustics is not new. Collins [Collins94] has examined several filters in combination with illumination maps. The filters we have examined are two radially symmetric filters: the cone filter and the Gaussian filter [Jensen96c], and the specialized differential filter introduced in [Jensen95a]. For examples of more advanced filters see Myszkowski et al. [Myszkowski97].

3.2.1 The cone filter

The cone-filter [Jensen96c] assigns a weight, w_{pc}, to each photon based on the distance, d_p, between x and the photon p. This weight is:

$$w_{pc} = 1 - \frac{d_p}{k\,r}\,, \tag{10}$$

where $k \geq 1$ is a filter constant characterizing the filter and r is the maximum distance. The normalization of the filter based on a 2d-distribution of the photons

is $1 - \frac{2}{3k}$ and the filtered radiance estimate becomes:

$$L_r(x, \vec{\omega}) \approx \frac{\sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) w_{pc}}{(1 - \frac{2}{3k})\pi r^2} \,. \tag{11}$$

3.2.2 The Gaussian filter

The Gaussian filter [Jensen96c] has previously been reported to give good results when filtering caustics in illumination maps [Collins94]. It is easy to use the Gaussian filter with the photon map since we do not need to warp the filter to some surface function. Instead we use the assumption about the locally flat surfaces and we can use a simple image based Gaussian filter [Pavicic90] and the weight w_{pg} of each photon becomes

$$w_{pg} = \alpha \left[1 - \frac{1 - e^{-\beta \frac{d_p^2}{2r^2}}}{1 - e^{-\beta}} \right] , \tag{12}$$

where d_p is the distance between the photon p and x and $\alpha = 0.918$ and $\beta = 1.953$ (see [Pavicic90] for details). This filter is normalized and the only change to equation 8 is that each photon contribution is multiplied by w_{pg}:

$$L_r(x, \vec{\omega}) \approx \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) w_{pg} \,. \tag{13}$$

3.2.3 Differential checking

In [Jensen95a] it was suggested to use a filter based on differential checking. The idea is to detect regions near edges in the estimation process and use less photons in these regions. In this way we might get some noise in the estimate but that is often preferable to blurry edges.

The radiance estimate is modified based on the following observation: when adding photons to the estimate, near an edge the changes of the estimate will be monotonic. That is, if we are just outside a caustic and we begin to add photons to the estimate (by increasing the size of the sphere centered at x that contains the photons), then it can be observed that the value of the estimate is increasing as we add more photons; and vice versa when we are inside the caustic. Based on this observation, differential checking can be added to the estimate — we stop

adding photons and use the estimate available if we observe that the estimate is either constantly increasing or decreasing as more photons are added.

3.3 The radiance estimate in a participating medium

For the radiance estimate presented so far we have assumed that the photons are located on a surface. For photons in a participating medium the formula changes to [Jensen98]:

$$\begin{aligned} L_i(x,\vec{\omega}) &= \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, L(x,\vec{\omega}')\, d\omega' \\ &= \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, \frac{d^2\Phi(x,\vec{\omega}')}{\sigma(x)\, d\omega'\, dV}\, d\omega' \\ &= \frac{1}{\sigma(x)} \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, \frac{d^2\Phi(x,\vec{\omega}')}{dV} \\ &\approx \frac{1}{\sigma(x)} \sum_{p=1}^{n} f(x,\vec{\omega}_p',\vec{\omega})\, \frac{\Delta\Phi_p(x,\vec{\omega}_p')}{\frac{4}{3}\pi r^3}\,, \end{aligned} \tag{14}$$

where L_i is the in-scattered radiance, and the volume $dV = \frac{4}{3}\pi r^3$ is the volume of the sphere containing the photons. $\sigma(x)$ is the scattering coefficient at x and f is the phase-function.

3.4 Locating the nearest photons

Efficiently locating the nearest photons is critical for good performance of the photon map algorithm. In scenes with caustics, multiple diffuse reflections, and/or participating media there can be a large number of photon map queries.

Fortunately the simplicity of the kd-tree permits us to implement a simple but quite efficient search algorithm. This search algorithm is a straight forward extension of standard search algorithms for binary trees [Cormen89, Sedgewick92, Horowitz93]. It is also related to range searching where kd-trees are commonly used as they have optimal storage and good performance [Preparata85]. The nearest neighbors query for kd-trees has been described extensively in several publications by Bentley et al. including [Bentley75, Bentley79a, Bentley79b, Bentley80]. More recent publications include [Preparata85, Sedgewick92]. Some of these papers go beyond our description of a nearest neighbors query by adding modifications and extensions to the kd-tree to further reduce the cost of searching. We

do not implement these extensions because we want to maintain the low storage overhead of the kd-tree as this is an important aspect of the photon map.

Locating the nearest neighbors in a kd-tree is similar to range searching [Preparata85] in the sense that we want to locate photons within a give volume. For the photon map it makes sense to restrict the size of the initial search range since the contribution from a fixed number of photons becomes small for large regions. This simple observation is particularly important for caustics since they often are concentrated in a small region. A search algorithm that does not limit the search range will be slow in such situations since a large part of the kd-tree will be visited for regions with a sparse number of photons.

A generic nearest neighbors search algorithm begins at the root of the kd-tree, and adds photons to a list if they are within a certain distance. For the n nearest neighbors the list is sorted such that the photon that is furthest away can be deleted if the list contains n photons and a new closer photon is found. Instead of naive sorting of the full list it is better to use a max-heap [Preparata85, Sedgewick92, Horowitz93]. A max-heap (also known as a priority queue) is a very efficient way of keeping track of the element that is furthest away from the point of interest. When the max-heap is full, we can use the distance d to the root element (ie. the photon that is furthest away) to adjust the range of the query. Thus we skip parts of the kd-tree that are further away than d.

Another simple observation is that we can use squared distances — we do not need the real distance. This removes the need of a square root calculation per distance check.

The pseudo-code for the search algorithm is given in Figure 10. A simple implementation of this routine is available with source code at [MegaPov00].

For this search algorithm it is necessary to provide an initial maximum search radius. A well-chosen radius allows for good pruning of the search reducing the number of photons tested. A maximum radius that is too low will on the other hand introduce noise in the photon map estimates. The radius can be chosen based on an error metric or the size of the scene. The error metric could for example take the average energy of the stored photons into account and compute a maximum radius from that assuming some allowed error in the radiance estimate.

A few extra optimizations can be added to this routine, for example a delayed construction of the max heap to the time when the number of photons needed has

```
given the photon map, a position x and a max search distance d^2
this recursive function returns a heap h with the nearest photons.
Call with locate_photons(1) to initiate search at the root of the kd-tree

locate_photons( p ) {
  if ( 2p + 1 < number of photons ) {
    examine child nodes
            Compute distance to plane (just a subtract)
    δ = signed distance to splitting plane of node n
    if (δ < 0) {
              We are left of the plane - search left subtree first
      locate_photons( 2p )
      if ( δ^2 < d^2 )
        locate_photons( 2p + 1 )      check right subtree
    } else {
              We are right of the plane - search right subtree first
      locate_photons( 2p + 1 )
      if ( δ^2 < d^2 )
        locate_photons( 2p )      check left subtree
    }
  }
          Compute true squared distance to photon
  δ^2 = squared distance from photon p to x
  if ( δ^2 < d^2 ) {          Check if the photon is close enough?
    insert photon into max heap h
            Adjust maximum distance to prune the search
    d^2 = squared distance to photon in root node of h
  }
}
```

Figure 10: Pseudocode for locating the nearest photons in the photon map

been found. This is particularly useful when the requested number of photons is large.

Nathan Kopp has implemented a slightly different optimization in an extended version of the Persistence Of Vision Ray Tracer (POV) called `MegaPov` (available at [MegaPov00]). In his implementation the initial maximum search radius is set to a very low value. If this value turns out to be too low, another search is performed with a higher maximum radius. He reports good timings and results from this technique [Kopp99].

Another change to the search routine is to use the disc check as described

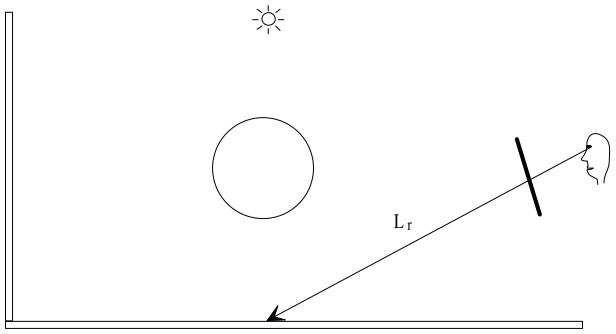


Figure 11: Tracing a ray through a pixel.

earlier. This is useful to avoid incorrect color bleeding and particularly helpful if the gathering step is not used and the photons are visualized directly.

4 Rendering

Given the photon map and the ability to compute a radiance estimate from it, we can proceed with the rendering pass. The photon map is view independent, and therefore a single photon map constructured for an environment can be utilized to render the scene from any desired view. There are several different ways in which the photon map can be visualized. A very fast visualization technique has been presented by Myszkowski et al. [Myszkowski97, Volevich99] where photons are used to compute radiosity values at the vertices of a mesh.

In this note we will focus on the full global illumination approach as presented in [Jensen96b]. Initially we will ignore the presence of participating media; at the end of the note we have added some notes for this case.

The final image is rendered using distribution ray tracing in which the pixel radiance is computed by averaging a number of sample estimates. Each sample consists of tracing a ray from the eye through a pixel into the scene (see Figure 11). The radiance returned by each ray equals the outgoing radiance in the direction of the ray leaving the point of intersection at the first surface intersected by the ray. The outgoing radiance, L_o, is the sum of the emitted, L_e, and the reflected radiance

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + L_r(x, \vec{\omega}) \,, \tag{15}$$

where the reflected radiance, L_r, is computed by integrating the contribution

from the incoming radiance, L_i,

$$L_r(x,\vec{\omega}) = \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') \cos\theta_i \, d\omega_i' \,, \tag{16}$$

where f_r is the bidirectional reflectance distribution function (BRDF), and Ω_x is the set of incoming directions around x.

L_r can be computed using Monte Carlo integration techniques like path tracing and distribution ray tracing. These methods are very costly in terms of rendering time and a more efficient approach can be obtained by using the photon map in combination with our knowledge of the BRDF and the incoming radiance.

The BRDF is separated into a sum of two components: A specular/glossy, $f_{r,s}$, and a diffuse, $f_{r,d}$

$$f_r(x,\vec{\omega}',\vec{\omega}) = f_{r,s}(x,\vec{\omega}',\vec{\omega}) + f_{r,d}(x,\vec{\omega}',\vec{\omega}) \,. \tag{17}$$

The incoming radiance is classified using three components:

- $L_{i,l}(x,\vec{\omega}')$ is direct illumination by light coming from the light sources.
- $L_{i,c}(x,\vec{\omega}')$ is caustics — indirect illumination from the light sources via specular reflection or transmission.
- $L_{i,d}(x,\vec{\omega}')$ is indirect illumination from the light sources which has been reflected diffusely at least once.

The incoming radiance is the sum of these three components:

$$L_i(x,\vec{\omega}') = L_{i,l}(x,\vec{\omega}') + L_{i,c}(x,\vec{\omega}') + L_{i,d}(x,\vec{\omega}') \,. \tag{18}$$

By using the classifications of the BRDF and the incoming radiance we can split the expression for reflected radiance into a sum of four integrals:

$$\begin{aligned} L_r(x,\vec{\omega}) &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') \cos\theta_i \, d\omega_i' \\ &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_{i,l}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' + \\ &\quad \int_{\Omega_x} f_{r,s}(x,\vec{\omega}',\vec{\omega}) (L_{i,c}(x,\vec{\omega}') + L_{i,d}(x,\vec{\omega}')) \cos\theta_i \, d\omega_i' + \\ &\quad \int_{\Omega_x} f_{r,d}(x,\vec{\omega}',\vec{\omega}) L_{i,c}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' + \\ &\quad \int_{\Omega_x} f_{r,d}(x,\vec{\omega}',\vec{\omega}) L_{i,d}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' \,. \end{aligned} \tag{19}$$

This is the equation used whenever we need to compute the reflected radiance from a surface. In the following sections we discuss the evaluation of each of the integrals in the equation in more detail. We distinguish between two different situations: an accurate and an approximate.

The accurate computation is used if the surface is seen directly by the eye or perhaps via a few specular reflections. It is also used if the distance between the ray origin and the intersection point is below a small threshold value — to eliminate potential inaccurate color bleeding effects in corners. The approximate evaluation is used if the ray intersecting the surface has been reflected diffusely since it left the eye or if the ray contributes only little to the pixel radiance.

4.1 Direct illumination

Direct illumination is given by the term

$$\int_{\Omega_x} f_r(x, \vec{\omega}', \vec{\omega}) L_{i,l}(x, \vec{\omega}') \cos\theta_i \, d\omega_i' \,,$$

and it represents the contribution to the reflected radiance due to direct illumination. This term is often the most important part of the reflected radiance and it has to be computed accurately since it determines light effects to which the eye is highly sensitive such as shadow edges.

Computing the contribution from the light sources is quite simple in ray tracing based methods. At the point of interest shadow rays are sent towards the light sources to test for possible occlusion by objects. This is illustrated in Figure 12. If a shadow ray does not hit an object the contribution from the light source is included in the integral otherwise it is neglected. For large area light sources several shadow rays are used to properly integrate the contribution and correctly render penumbra regions. This strategy can however be very costly since a large number of shadow rays is needed to properly integrate the direct illumination.

Using a derivative of the photon map method we can compute shadows more efficiently using shadow photons [Jensen95c]. This approach can lead to considerable speedups in scenes with large penumbra-regions that are normally very costly to render using standard ray tracing. The approach is stochastic though, so it might miss shadows from small objects in case these aren't intersected by any photons. This is a problem with all techniques that use stochastic evaluation of visibility.

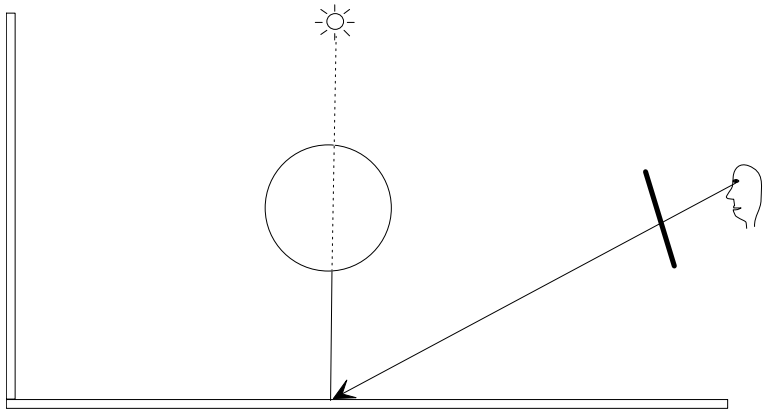


Figure 12: Accurate evaluation of the direct illumination.

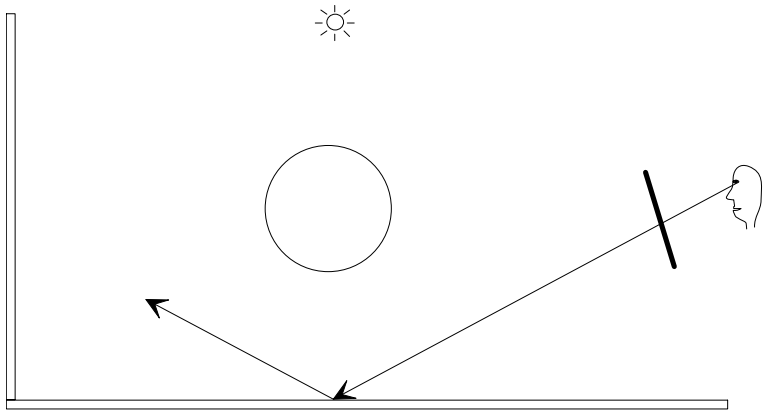


Figure 13: Rendering specular and glossy reflections.

The approximate evaluation is simply the radiance estimate obtained from the global photon map (no shadow rays or light source evaluations are used). This is seen in Figure 15 where the global photon map is used in the evaluation of the incoming light for the secondary diffuse reflection.

4.2 Specular and glossy reflection

Specular and glossy reflection is computed by evaluation of the term

$$\int_{\Omega_x} f_{r,s}(x, \vec{\omega}', \vec{\omega})(L_{i,c}(x, \vec{\omega}') + L_{i,d}(x, \vec{\omega}')) \cos\theta_i \, d\omega_i' \, .$$

The photon map is not used in the evaluation of this integral since it is strongly dominated by $f_{r,s}$ which has a narrow peak around the mirror direction. Using the

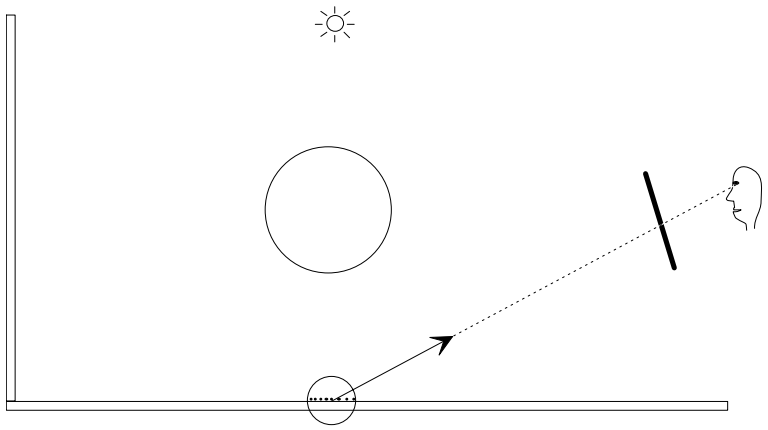


Figure 14: Rendering caustics.

photon map to optimize the integral would require a huge number of photons in order to make a useful classification of the different directions within the narrow peak of $f_{r,s}$. To save memory this strategy is not used and the integral is evaluated using standard Monte Carlo ray tracing optimized with importance sampling based on $f_{r,s}$. This is still quite efficient for glossy surfaces and the integral can in most situations be computed using only a small number of sample rays.

This is illustrated in Figure 13.

4.3 Caustics

Caustics are represented by the integral

$$\int_{\Omega_x} f_{r,d}(x, \vec{\omega}', \vec{\omega}) L_{i,c}(x, \vec{\omega}') \cos\theta_i \, d\omega_i' \, .$$

The evaluation of this term is dependent on whether an accurate or an approximate computation is required. In the accurate computation, the term is solved by using a radiance estimate from the caustics photon map. The number of photons in the caustics photon map is high and we can expect good quality of the estimate. Caustics are never computed using Monte Carlo ray tracing since this is a very inefficient method when it comes to rendering caustics. The approximate evaluation of the integral is included in the radiance estimate from the global photon map.

This is illustrated in Figure 14.

4.4 Multiple diffuse reflections

The last term in equation 19 is

$$\int_{\Omega_x} f_{r,d}(x, \vec{\omega}', \vec{\omega}) L_{i,d}(x, \vec{\omega}') \cos\theta_i \, d\omega_i' \, .$$

This term represents incoming light that has been reflected diffusely at least once since it left the light source. The light is then reflected diffusely by the surface (using $f_{r,d}$). Consequently the resulting illumination is very "soft".

The approximate evaluation of this integral is a part of the radiance estimate based on the global photon map.

The accurate evaluation of the integral is calculated using Monte Carlo ray tracing optimized using the BRDF with an estimate of the flux as described in [Jensen95b]. An important optimization at Lambertian surfaces is the use of Ward's irradiance gradient caching scheme [Ward88, Ward92]. This means that we only compute indirect illumination on Lambertian surfaces if we cannot interpolate with sufficient accuracy from previously computed values. The advantage of using the photon map compared to just using the irradiance gradient caching method is that we avoid having to trace multiple bounces of indirect illumination and we can use the information in the photon map to concentrate our samples into the important directions.

This is illustrated in Figure 15.

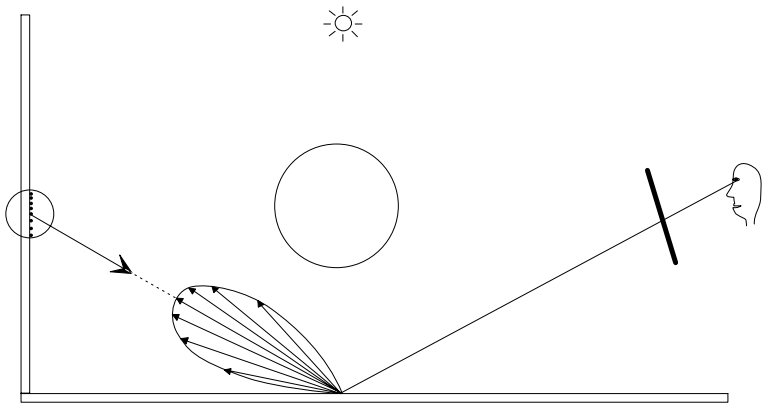


Figure 15: Computing indirect diffuse illumination with importance sampling.

4.5 Participating media

In the presence of participating media we can still use the framework as presented so far. The main difference is that we need to take the media into account as we trace rays through the scene. This can be done quite efficiently using ray marching and the volume radiance estimate as described in [Jensen98].

4.6 Why distribution ray tracing?

The rendering method presented here is a combination of many algorithms. In order to render accurate images without using too many photons a distribution ray tracer is used to compute illumination seen directly by the eye. One might consider visualizing the global photon map directly, and this would indeed be a full global illumination solution (it would be similar to the density estimation approach presented in [Shirley95]). The problem with this approach is that an accurate solution requires a large number of photons. Significantly fewer photons are necessary when a distribution ray tracer is used to evaluate the first diffuse reflection. If a blurry solution is not a problem (for example for previewing) then a direct visualization of the photon map can be used. For more accurate results it is often necessary to use more than 1000 photons in the radiance estimate (see the results section for some examples).

5 Examples

In this section we present some examples of scenes rendered using photon maps. Please see the photon map web-page at `http://www.gk.dtu.dk/photonmap` for the latest results. Also refer to the papers included in these notes for more examples.

All the images have been rendered using the `Dali` rendering program. `Dali` is an extremely flexible renderer that supports ray tracing with global illumination and participating media. The global illumination simulation code based on photon maps is a module in `Dali` that is loaded at runtime. All material and geometry code is also represented via modules that are loaded at runtime. `Dali` is multithreaded and all images have been rendered on a dual PentiumII-400 PC running Linux. The width of each image is 1024 pixels and 4 samples per pixel have been used.

5.1 The Cornell box

Most global illumination papers feature a simulation of the Cornell box, and so does this note. Since we do not use radiosity our version of the Cornell box is slightly different. It has a mirror sphere and a glass sphere instead of the two cubes featured in the original Cornell box (the original Cornell box can be found at `http://www.graphics.cornell.edu/online/box/`). Classic radiosity methods have difficulties handling curved specular objects, but ray tracing methods (including the photon map method) have no problems with these.

5.1.1 Ray tracing

The image in Figure 16 shows the *ray traced* version of the Cornell box. Notice the sharp shadows and the black ceiling of the box due to lack of area lights and global illumination. Rendering time was 3.5 seconds.

5.1.2 Ray tracing with soft shadows

In Figure 17 soft shadows have been added. It has been reported that some people associate soft shadows with global illumination, but in the Cornell box example it is still obvious that something is missing. The ceiling is still black. Rendering time was 21 seconds.

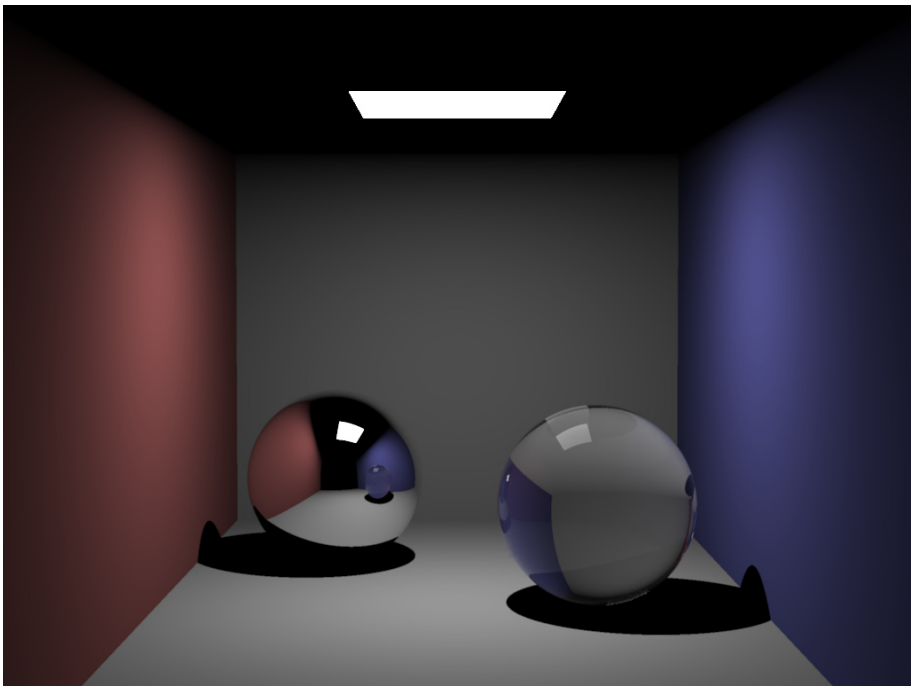


Figure 16: Ray traced Cornell box with sharp shadows.

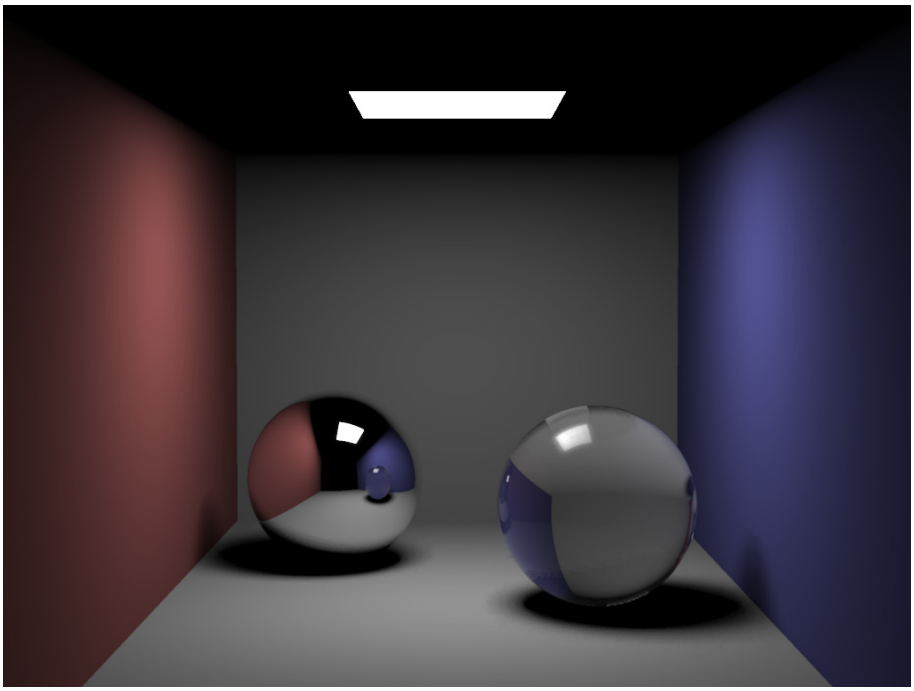


Figure 17: Ray traced Cornell box with soft shadows.

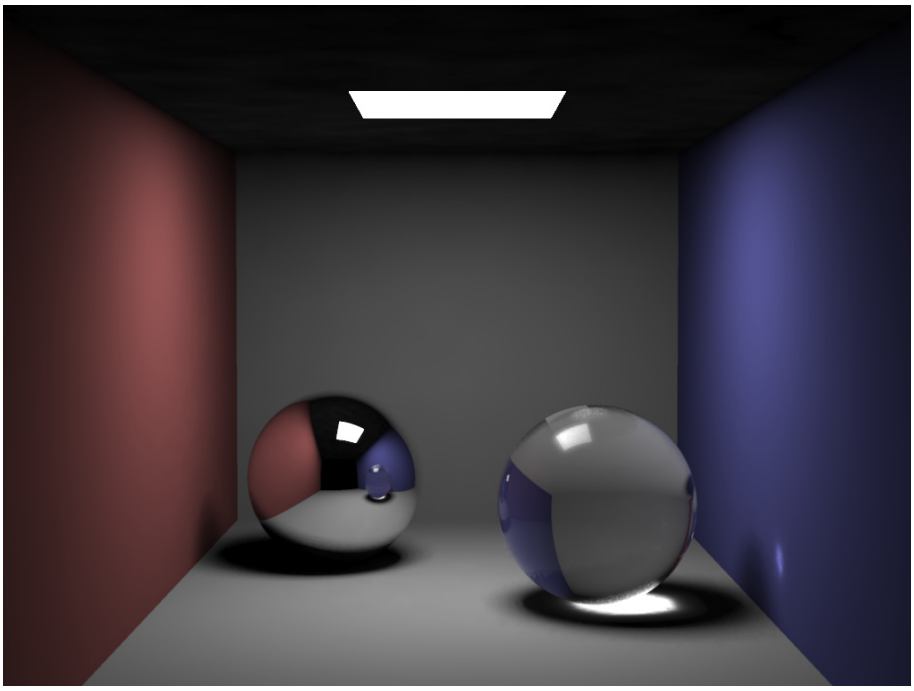


Figure 18: Cornell box with caustics.

5.1.3 Adding caustics

The image in Figure 18 includes the caustics photon map. Notice the bright spot below the glass sphere and on the right wall (due to light reflected of the mirror sphere and transmitted through the glass sphere). Also notice the faint illumination of the ceiling. The caustics photon map has 50000 photons and the estimate uses up to 60 photons. Photon tracing took 2 seconds. Rendering time was 34 seconds. We did not use any filtering of the caustics photons. A maximum search distance of 0.15 was used for the caustics photon map (the depth of the Cornell box is 5 units). Using a search distance of 0.5 increased the rendering time to 42 seconds. For an unlimited initial search radius the rendering time was 43 seconds. The computed images looked very similar. The faint illumination of the ceiling is a caustic (created by the bright caustic on the floor) — it becomes a little softer with the increased search radius. For a search radius of 0.01 the caustics became more noisy, and the rendering time was 25 seconds. For other scenes where the caustics are more localized the influence of the maximum search radius on the rendering time can be more dramatic than for the Cornell box.

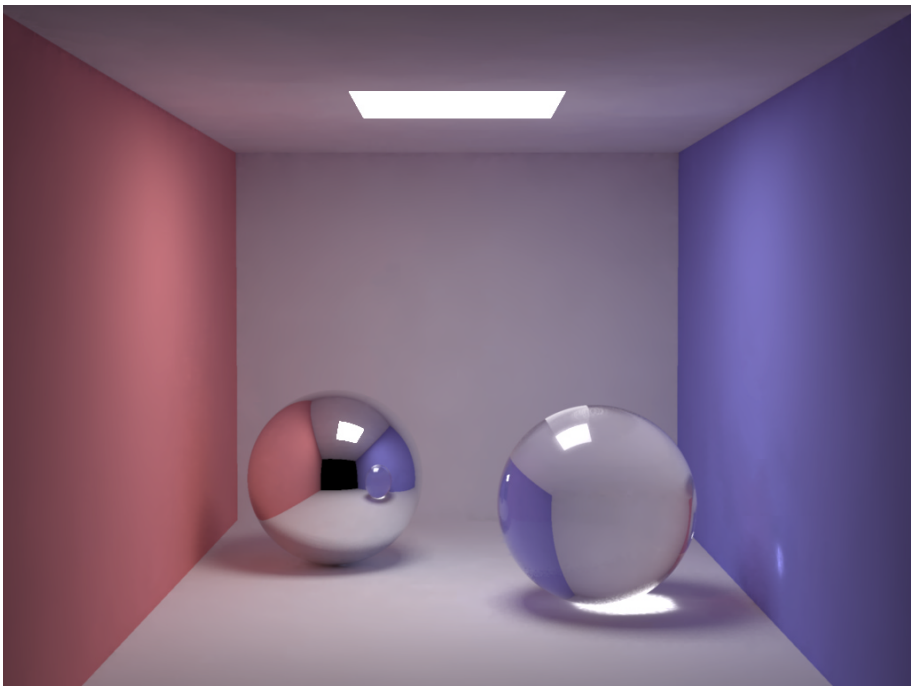


Figure 19: Cornell box with global illumination.

5.1.4 Global illumination

In Figure 19 global illumination has been computed. The image is much brighter and the ceiling is illuminated. 200000 photons were used in the global photon map and 100 photons in the estimate. The caustic photon map parameters are the same. Photon tracing took 4 seconds. Rendering time was 66 seconds.

5.1.5 The radiance estimate from the global photon map

Finally in Figure 20 we have visualized the radiance estimates from the global photon map directly. We have shown images with 100 and 500 photons in the estimate. Notice how the illumination becomes softer and more pleasing with more photons, but also more blurry and with more false color bleeding at the edges. The edge problem can be solved partially by using an ellipsoid or disc to locate the photons (see section 3) — with 500 photons in the estimate and the ellipsoid search activated we get the image in Figure 21 These images took 30–35 seconds to render. Notice how the quality of the direct visualization gives a reasonable estimate of the overall illumination in the scene. This is the information

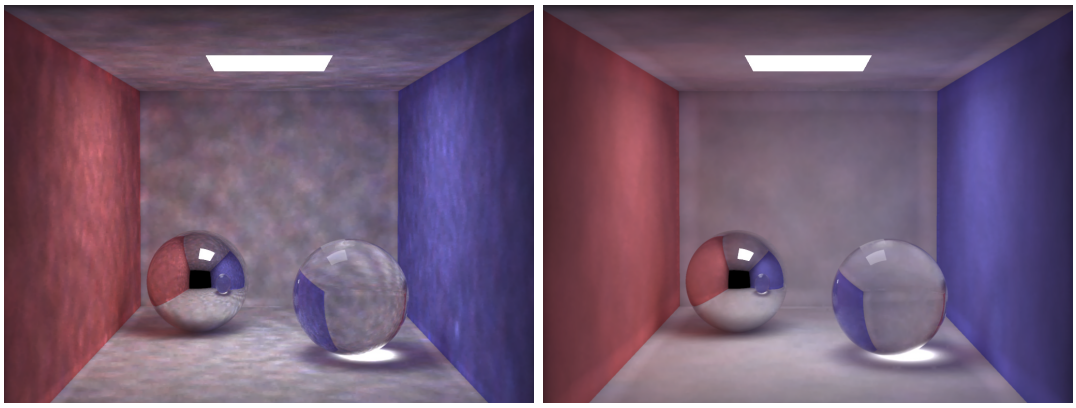


Figure 20: Global photon map radiance estimates visualized directly using 100 photons (left) and 500 photons (right) in the radiance estimate.

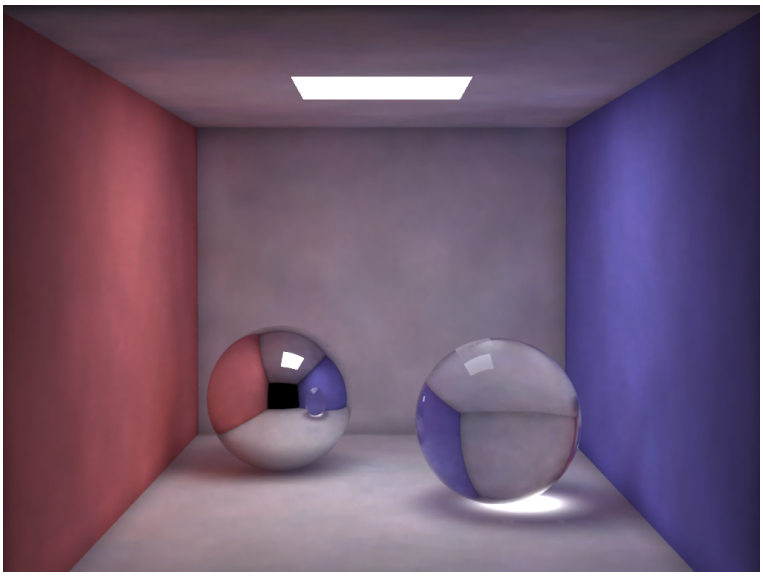


Figure 21: Global photon map radiance estimates visualized directly using 500 photons and a disc to locate the photons. Notice the reduced false color bleeding at the edges.

we benefit from in the full rendering step since we do not have to sample the incoming light recursively.

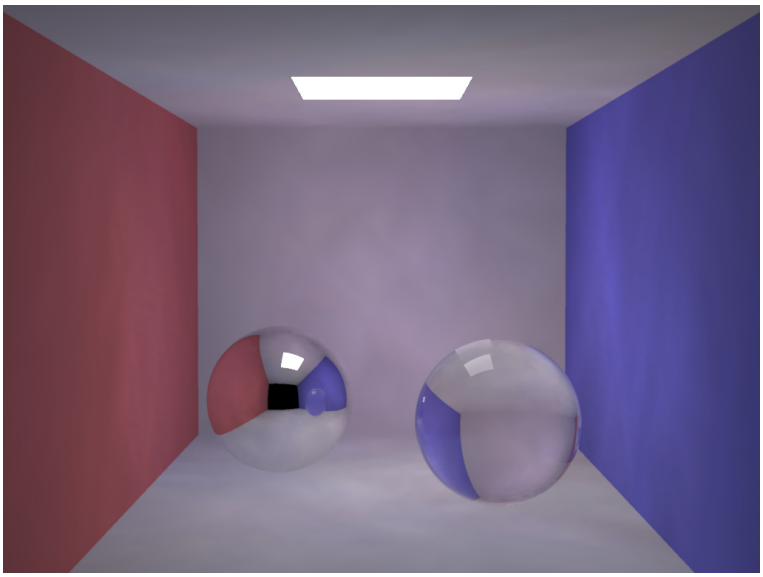


Figure 22: Fast visualization of the radiance estimate based on 50 photons and a global photon map with just 200 photons. Rendering time was 4 seconds.

5.1.6 Fast global illumination estimate

For fast visualization of global illumination one can use very few photons in the global photon map. In Figure 22 we have visualized the radiance estimate from a global photon map with just 200 photons! We used up to 50 photons in the radiance estimate. The illumination is very blurry and as a consequence the shadows and the caustics are missing, but the overall illumination is approximately correct, and this visualization is representative of the final rendering as shown in Figure 19. Photon tracing took 0.03 seconds and the rendering time for the image was 4 seconds. This is almost as fast as the simple ray tracing version, and the main reason is that we only used ray tracing to compute the first intersection and the mirror reflections and transmissions. The global photon map was used to estimate both indirect and direct light.

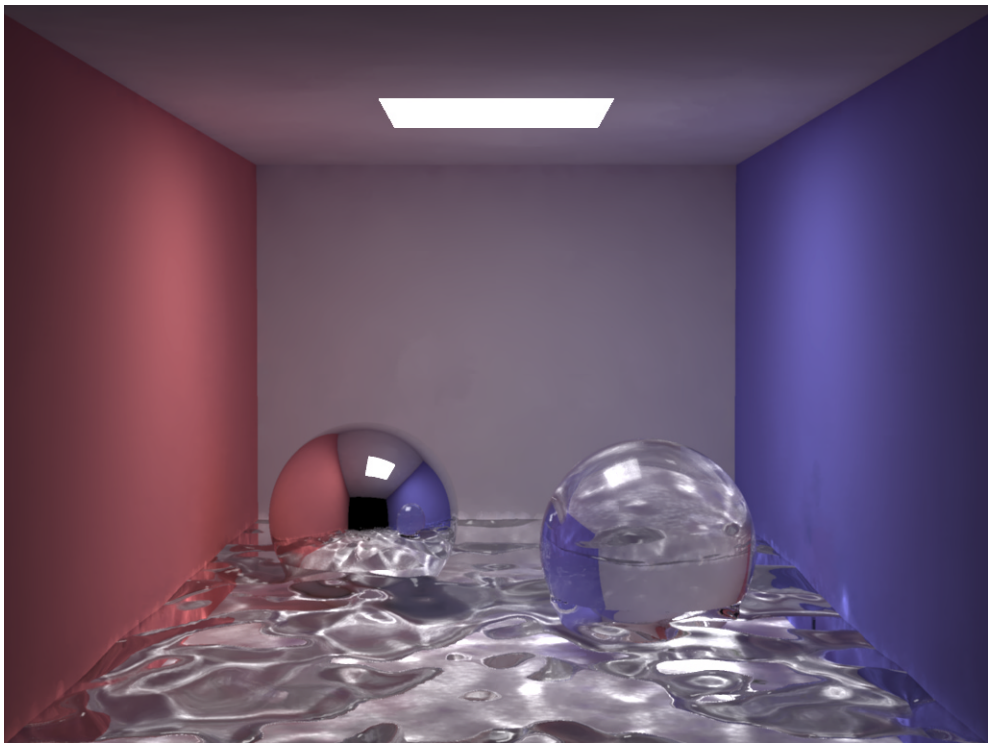


Figure 23: Cornell box with water.

5.2 Cornell box with water

In the Cornell box in Figure 23 we have inserted a displacement-mapped water surface. To render this scene we used 500000 photons in both the caustics and the global photon map, and up to 100 photons in the radiance estimate. We used a higher number of caustic photons due to the water surface which causes the entire floor to be illuminated by the photons in the caustics photon map. Also the number of photons in the global photon map have been increased to account for the more complex indirect illumination in the scene. The water surface is made of 20000 triangles. The rendering time for the image was 11 minutes.

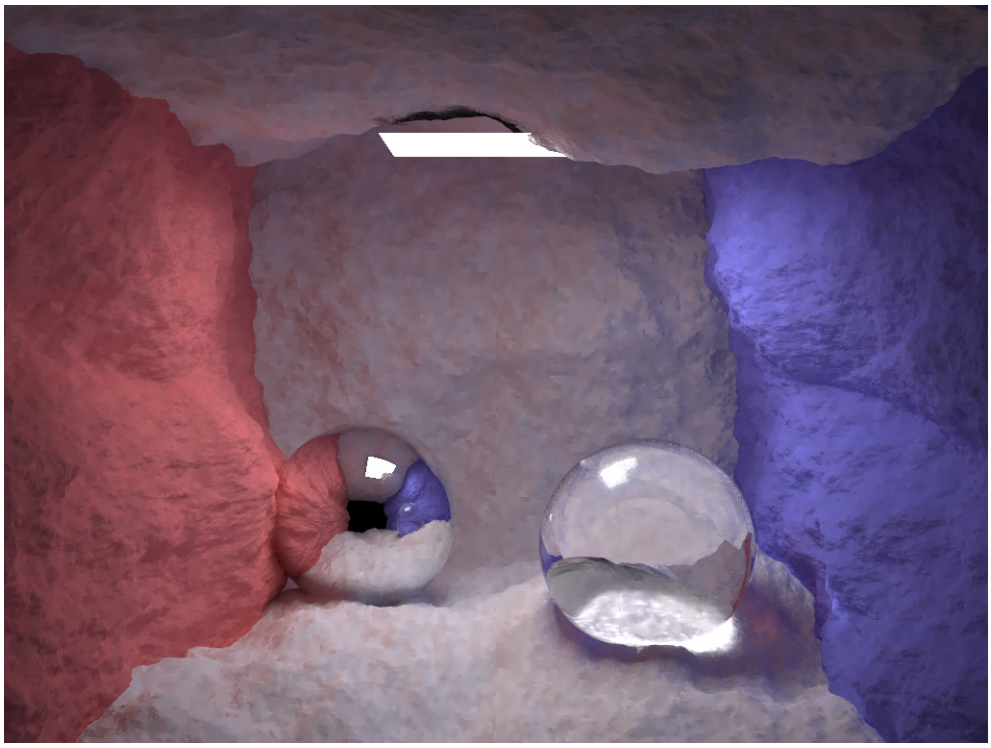


Figure 24: Fractal Cornell box.

5.3 Fractal Cornell box

An example of a more complex scene is shown in Figure 24. The walls have been replaced with displacement mapped surfaces (generated using a fractal midpoint subdivision algorithm) and the model contains a little more than 1.6 million elements. Notice that each wall segment is an instanced copy of the same fractal surface. With photon maps it is easy to take advantage of instancing and the geometry does not have to be explicitly represented. We used 200000 photons in the global photon map and 50000 in the caustics photon map. This is the same number of photons as in the simple Cornell box and our reasoning for choosing the same values are that the complexity of the illumination is more or less the same as in the simple Cornell box. We want to capture the color bleeding from the colored walls and the indirect illumination of the ceiling. All in all we used the same parameters for the photon map as in the simple Cornell box. We only changed the parameters for the acceleration structure to handle the larger amount of geometry. The rendering time for the scene was 14 minutes.

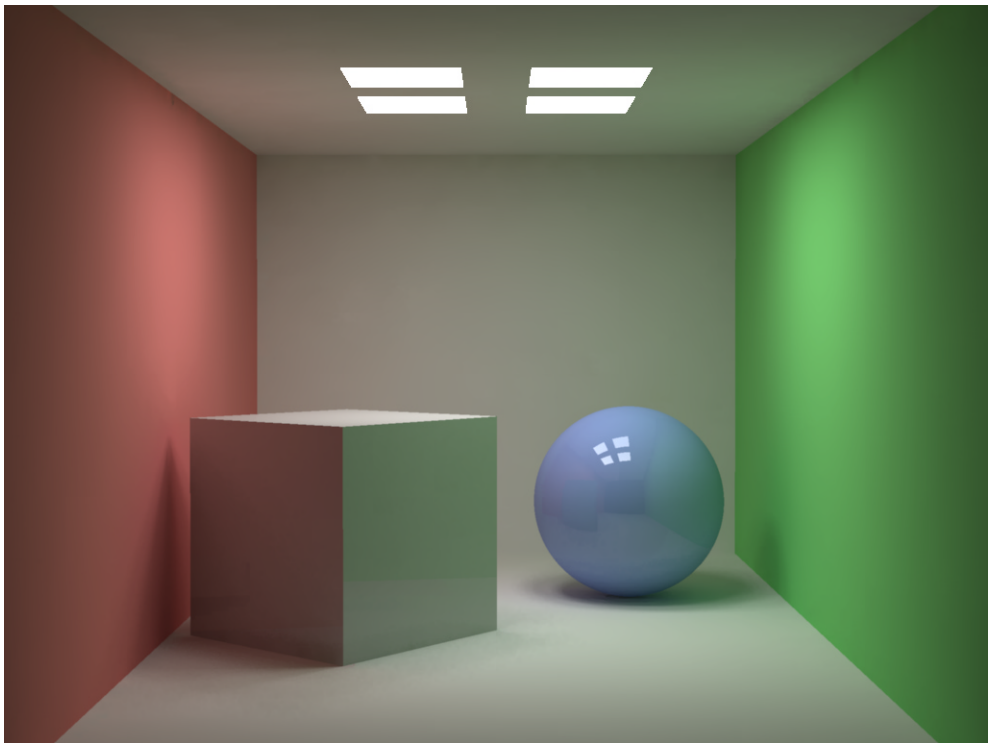


Figure 25: Cornell box variation with 4 light sources.

5.4 Cornell box with multiple lights

A simple example of a scene with multiple light sources is the variation of the Cornell box scene shown in Figure 25. We generated 100000 photons from each light source and the resulting global photon map has 400000 photons. Other than that the rendering parameters were the same as for the other Cornell box with 1 light source. The rendering time for this scene was 90 seconds.

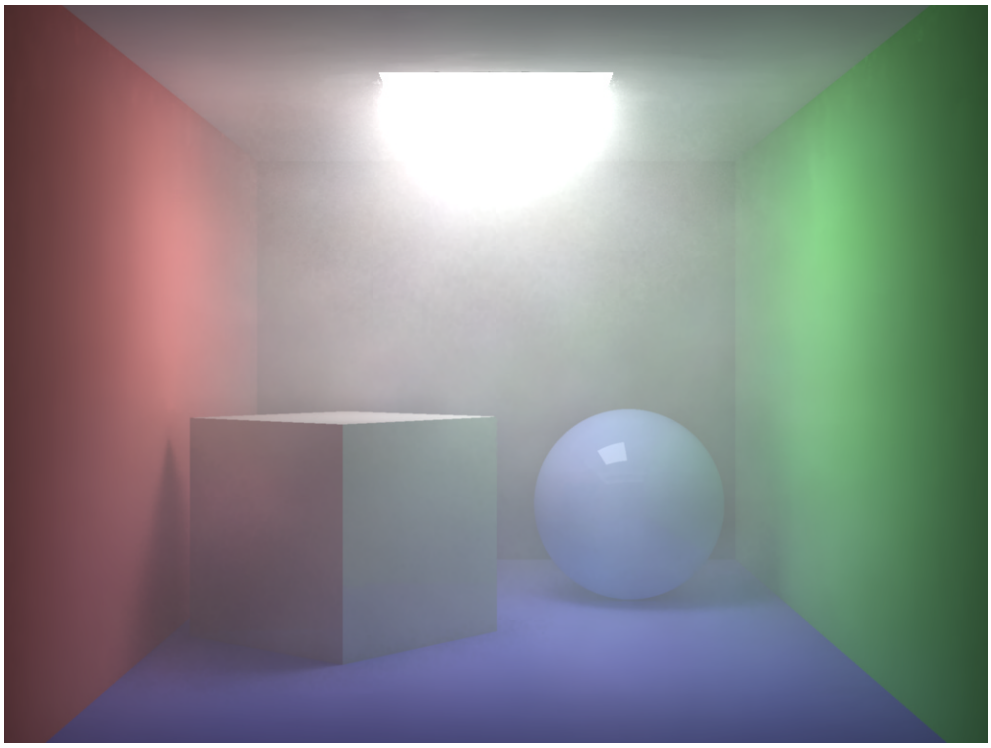


Figure 26: Cornell box with a participating medium.

5.5 Cornell box with smoke

The Cornell box scene shown in Figure 26 is an example of a scene with a uniform participating medium. To simulate this scene we used 100000 photons in the global photon map and 150000 photons in the volume photon map. A simple non-adaptive ray marcher has been implemented so the step size had to be set to a low value which is extra costly. The rendering time for the scene was 44 minutes.

Figure 27: A cognac glass with a caustic.

5.6 Cognac glass

Figure 27 shows an example of a caustic from a cognac glass. The glass is an object of revolution approximated with 12000 triangles. To generate the caustic we used 200000 photons. The radiance estimates for the caustic were computed using up to 40 photons. The rendering time for the image was 8 minutes and 10 seconds — part of this rendering time is due to the ray traced depth of field simulation.

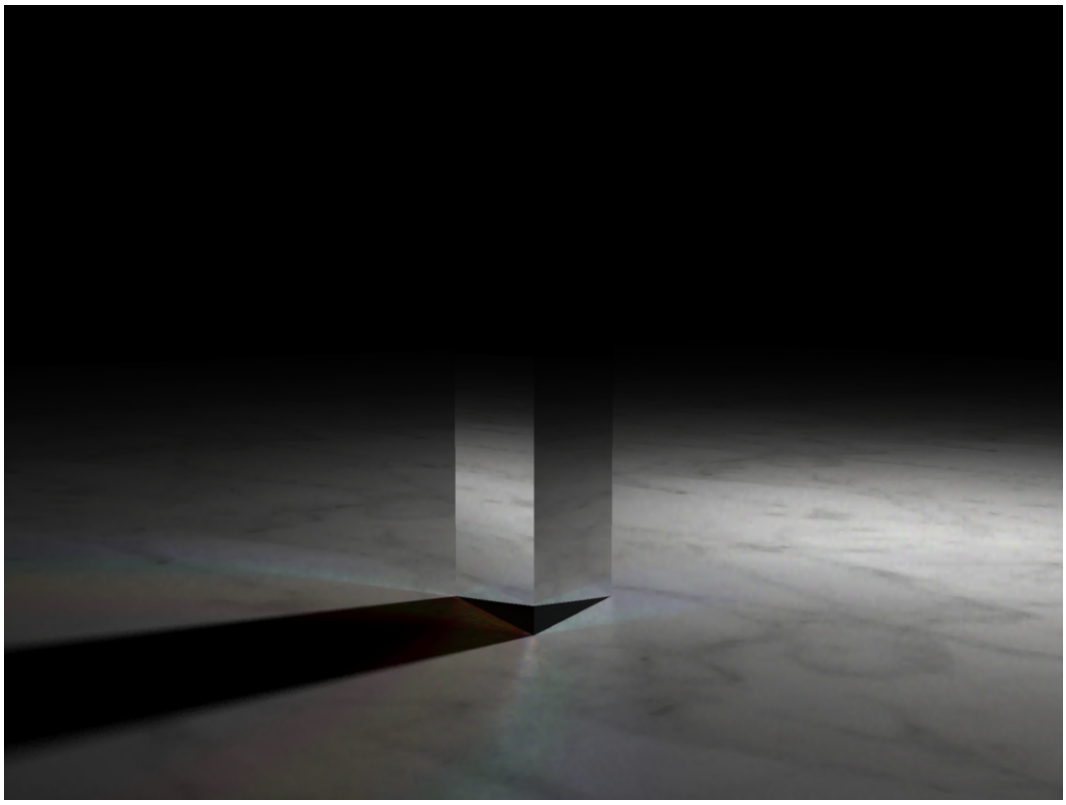


Figure 28: Caustics through a prism with dispersion.

5.7 Prism with dispersion

The classic example of dispersion with glass prism is shown in Figure 28. Even though only three separate wavelengths have been sampled, the color variations in the caustics are smooth. An accurate color representation would require more wavelength samples; such an extension to the photon map is easy to implement. 500000 photons were used in the caustics and 80 photons were used in the radiance estimate. The rendering time for the image was 32 seconds.

Figure 29: Granite bunny next to a marble bunny — both models are rendered using subsurface scattering. The photon map is used to compute multiple scattering inside the stone material.

5.8 Subsurface scattering

A recent addition to the photon map is the simulation of subsurface scattering [Jensen99, Dorsey99]. For subsurface scattering we use the photon map to compute the effect of multiple scattering within a given material. This is often very costly to compute and therefore mostly omitted from approaches dealing with subsurface scattering. This is unfortunate since multiple scattering leads to very nice and subtle color bleeding effects inside the material which improves the quality of the rendering.

Figure 29 shows a granite bunny next to a marble bunny. Both of these stone models are rendered using subsurface scattering with 100000 photons used to simulate multiple scattering. The rendering time for the picture was 21 minutes. This bunny is the original Stanford bunny and the scene contains 140000 triangles, and it is rendered with full global illumination and depth of field.

Figure 30 shows a bust of Diana the Huntress made of translucent marble. For this scene the light source was behind the bust to emphasize the effect of subsurface scattering. Notice the translucency of the hair and the nose region. This image was rendered in 21 minutes using 200000 photons.

Figure 30: Translucent marble bust illuminated from behind

Where to get programs with photon maps

Photon maps are already available on the Internet for downloading. We have collected the following links as of the writing of these notes.

Nathan Kopp has made a photon map extension to `MegaPov` [MegaPov00] (an extended version of POV ray). Free source code and executable can be found at: `www.nathan.kopp.com/patched.htm`

Larry Gritz has mentioned that photon maps will be available in the Blue Moon Rendering Tools (a free Renderman compliant ray tracer and global illu-

mination renderer). See `www.bmrt.org` for more information.

The following commercial products uses photon maps to render caustics and/or indirect illumination: Lightflow, LightWave, Luminaire and Twister. In addition the Inspirer rendering system uses a fast photon mapping approach — a mesh based view independent solution is computed based on the local density of photons.

Several other people have implemented photon maps, and a few research packages provides photon maps (these packages might not be publicly available).

Refer to `www.gk.dtu.dk/photonmap/` for an updated list of web-sites.

References and further reading

This section lists the references referenced in these course notes plus additional background material relevant to the photon map method. The material is divided into three groups: the photon map method, ray tracing and photon tracing and data-structures with focus on kd-trees. Each part is in chronological order with annotations. In addition we have listed a number of animations rendered with photon maps and finally we have provided a more detailed list of relevant background litterature.

The photon map method

[Jensen95a] Henrik Wann Jensen and Niels Jørgen Christensen.
"Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects".
Computers & Graphics **19** (2), pages 215–224, 1995.
The first paper describing the photon map. The paper suggested the use of a mixture of photon maps and illumination maps, where photon maps would be used for complex surfaces such as fractals.

[Jensen95b] Henrik Wann Jensen.
"Importance driven path tracing using the photon map".
Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering), pages 326–335. Springer Verlag, 1995.
Introduces the use of photons for importance sampling in path tracing. By combining the knowledge of the incoming flux with the BRDF it is possible to get better results using fewer sample rays.

[Jensen95c] Henrik Wann Jensen and Niels Jørgen Christensen.
"Efficiently Rendering Shadows using the Photon Maps".
In *Proceedings of Compugraphics'95*, pages 285–291, Alvor, December 1995.

Introduces the use of shadow photons for an approximate classification of the light source visibility in a scene.

[Jensen96a] Henrik Wann Jensen.
"Rendering caustics on non-Lambertian surfaces".
Proceedings of Graphics Interface'96, pages 116-121, Toronto, May 1996 (also selected for publication in Computer Graphics Forum, volume 16, number 1, pages 57–64, March 1997).
Extension of the photon map method to render caustics on non-Lambertian surfaces ranging from diffuse to glossy.

[Jensen96b] Henrik Wann Jensen.
"Global illumination using photon maps".
Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering), pages 21–30. Springer Verlag, 1996.
Presents the global illumination algorithm using photon maps. A caustic and a global photon map is used to optimize the rendering of global illumination including the simulation of caustics.

[Jensen96c] Henrik Wann Jensen.
The photon map in global illumination.
Ph.D. dissertation, Technical University of Denmark, September 1996.
An in-depth description of the photon map method based on the presentations in the published photon map papers.

[Christensen97] Per H. Christensen.
"Global illumination for professional 3D animation, visualization, and special effects" (invited paper).
Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering), pages 321–326. Springer Verlag, 1997.
Describes the requirements of a global illumination method in a commercial environment, and motivates the choice of the photon map method.

[Myszkowski97] Karol Myszkowski.
"Lighting reconstruction using fast and adaptive density estimation techniques".
Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering), pages 321–326. Springer Verlag, 1997.
Efficient techniques for filtering and visualizing photons.

[Slusallek98] Philipp Slusallek, Mark Stamminger, Wolfgang Heidrich, J.-C. Popp, and Hans-Peter Seidel.
"Composite Lighting Simulations with Lighting Network".
IEEE Computer Graphics & Applications, 18(2), pages 22-31, March/April 1998.
Describes a framework in which the photon map can be integrated into a radiosity simulation.

[Peter98] Ingmar Peter and Georg Pietrek.
"Importance driven construction of photon maps."
Rendering Techniques '98 (Proceedings of the Ninth Eurographics Workshop on Rendering), pages 269–280. Springer Verlag, 1998.
Use of importance to focus the photons where they contribute most to the visible solution. This requires an initial importance (or "importons") tracing pass from the camera before the photon tracing pass from the light sources.

[Jensen98] Henrik Wann Jensen and Per H. Christensen.
"Efficient simulation of light transport in scenes with participating media using photon maps".
Proceedings of SIGGRAPH 98, pages 311–320. ACM, 1998.
Extension of the photon map method to simulate global illumination in scenes with participating media.

[Lange98] Thorsten Lange and Georg Pietrek.
"Rendering Participating Media using the Photon Map".
Technical Report no. 678, University of Dortmund, 1998.

Also describes the extension of the photon map method to simulate global illumination in the presence of participating media.

[Jensen99] Henrik Wann Jensen, Justin Legakis and Julie Dorsey.
"Rendering of Wet Materials".
Proceedings of the Tenth Eurographics Workshop on Rendering, pages 281-290, Granada, June 1999.
Simulates subsurface scattering using the volume photon map in order to render wet materials.

[Dorsey99] Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis and Hans Køhling Pedersen.
"Modeling and Rendering of Weathered Stone".
Proceedings of SIGGRAPH 99.
Describes rendering of volumetric weathering effects in stone based on subsurface scattering optimized using the volume photon map.

[Jensen00] Henrik Wann Jensen.
"Parallel Global Illumination using Photon Mapping". In *SIGGRAPH'2000, Course 30*, New Orleans, July 2000.
Describes how to parallelize the photon map algorithm.

Ray tracing and photon tracing

[Whitted80] Turner Whitted.
"An improved illumination model for shaded display".
Communications of the ACM, volume 23, number 6, pages 343–349. ACM, June 1975.
The classic ray tracing paper.

[Arvo86] James Arvo.
"Backward ray tracing".
Developments in ray tracing, SIGGRAPH 96 seminar notes. ACM, August 1986.

Introduces light ray tracing and illumination maps for computing caustics.

[Glassner89] Andrew S. Glassner.
An introduction to ray tracing.
Academic Press, 1989.
The standard reference on ray tracing. Still a pleasure to read.

[Shirley91] Peter Shirley.
Physically Based Lightning Calculations for Computer Graphics.
Ph.d. thesis, University of Illinois at Urbana-Champaign, 1991.
Good overview of Monte Carlo ray tracing. Also presents one of the first practical multi-pass global illumination methods.

[Chen91] Eric Shenchang Chen, Holly E. Rushmeier, Gavin Miller, and Douglas Turner.
"A progressive multi-pass method for global illumination".
Proceedings of SIGGRAPH 91, pages 164–174. ACM, 1991.
One of the first multi-pass global illumination methods. Uses illumination maps for caustics, radiosity for indirect light and path tracing for rendering.

[Ward92] Gregory Ward and Paul Heckbert.
"Irradiance gradients".
Third Eurographics Workshop on Rendering, pages 85–98. Eurographics, 1992.
Describes the irradiance gradients method which is used for the final gathering step of the photon map method.

[Pattanaik93] Sumant N. Pattanaik.
"Computational Methods for Global Illumination and Visualisation of Complex 3D Environments".
Ph.d. Thesis, Birla Institute of Technology & Science, 1993

Introduces particle tracing where photons are emitted from the light sources and stored in a mesh.

[Rushmeier93] Holly Rushmeier, Ch. Patterson and A. Veerasamy.
"Geometric Simplification for Indirect Illumination Calculations".
Proceedings of Graphics Interface '93, pages 35-55, 1994.
Introduces the concept of geometry simplification for the radiosity step of multipass global illumination computations.

[Glassner95] Andrew S. Glassner.
Principles of digital image sythesis.
Morgan Kaufmann, 1995.
Gives an excellent overview of the entire field of image synthesis. Of particular interest here is the description of Monte Carlo photon tracing and Russian roulette.

[Lafortune96] Eric P. Lafortune.
Mathematical Models and Monte Carlo Algorithms for Physcially Based Rendering.
Ph.d. thesis, Katholieke University, Leuven, Belgium 1996.
Good overview of Monte Carlo ray tracing techniques including bidirectional path tracing.

[Dutre96] Philip Dutré and Yves D. Willems.
Mathematical Frameworks and Monte Carlo Algorithms for Global Illumination in Compute Graphics.
Ph.d. thesis, Katholieke Universiteit Leuven, 1996.
Another fine overview of Monte Carlo ray tracing and photon tracing.

[Ward98] Gregory Ward Larson and Rob Shakespeare.
Rendering with Radiance — the art and science of lighting visualization.
Morgan Kaufmann, 1998.
An entire book dedicated to the excellent Radiance renderer with many practical examples.

Datastructures

[Bentley75] Jon L. Bentley.
"Multidimensional binary search trees used for associative searching".
Communications of the ACM, volume 18, number 9, pages 509–517. ACM, 1975.
First paper on the kd-tree datastructure.

[Preparata85] Franco P. Preparata and Michael Ian Shamos.
Computational Geometry An Introduction, Springer-Verlag, 1985

[Cormen89] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest.
Introduction to algorithms.
MIT Press, 1989.
Good overview of algorithms including the heap-datastructure.

[Sedgewick92] Robert Sedgewick.
Algorithms in C++.
Addison-Wesley, 1992.
Also good description of the heap structure, and algorithms for the median search (used in the balancing algorithm).

Other references

These are additional useful references for ray tracing based rendering methods.

[Ansi86] American National Standard Institute.
"Nomenclature and Definitions for Illumination Engineering".
ANSI report, ANSI/IES RP-16-1986, 1986.

[Arvo90] James Arvo and David Kirk.
"Particle Transport and Image Synthesis".
Computer Graphics **24** (4), pages 53–66, 1990.

[Bentley79a] Jon Louis Bentley.
"Multidimensional Binary Search Trees in Database Applications".
IEEE Trans. on Soft. Eng. **5** (4), pages 333–340, July 1979.

[Bentley79b] Jon Louis Bentley and Jerome H. Friedman.
"Data Structures for Range Searching".
Computing Surveys **11** (4), pages 397–409, 1979.

[Bentley80] Jon Louis Bentley, Bruce W. Weide, and Andrew C. Yao.
"Optimal Expected-Time Algorithm for Closest Point Problems".
ACM Trans. on Math. Soft., **6** (4), pages 563–580, dec. 1980.

[Collins94] Steven Collins.
"Adaptive Splatting for Specular to Diffuse Light Transport".
In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 119–135, Darmstadt 1994.

[Cook84] Robert L. Cook.
"Distributed Ray Tracing".
Computer Graphics **18** (3), pages 137–145, 1984.

[Cook86] Robert L. Cook.
"Stochastic Sampling in Computer Graphics".
ACM Transactions on Graphics **5** (1), pages 51–72, Jan. 1986.

[Dutre94] Philip Dutré and Yves D. Willems.
"Importance-driven Monte Carlo Light Tracing".
In *proceedings of 5. Eurographics Workshop on Rendering*, pages 185–194, Darmstadt 1994.

[Ebert94] David Ebert, Ken Musgrave, Darwyn Peachey, Ken Perlin and Steve Worley.
Texturing and Modeling: A Procedural Approach.
Academic Press, October 1994.

[Gritz96] Larry Gritz and J. K. Hahn.
"BMRT: A Global Illumination Implementation of the RenderMan Standard".
Journal of Graphics Tools, Vol. 1, No. 3, pages 29-47, 1996.

[Hall88] Roy Hall.
Illumination and Color Computer Generated Imagery.
Springer-Verlag, 1988

[Heckbert90] Paul S. Heckbert.
"Adaptive Radiosity Textures for Bidirectional Ray Tracing".
Computer Graphics **24** (4), pages 145–154, 1990.

[Horowitz93] Ellis Horowitz, Sartaj Sahni and Susan Anderson-Freed.
Fundamentals of Data Structures in C, Computer Science Press, 1993

[Jensen93] Henrik Wann Jensen.
Global Illumination using Bidirectional Monte Carlo Ray Tracing.
M.Sc. thesis, Technical University of Denmark (in Danish), 1993.

[Jensen95f] Henrik Wann Jensen and Niels Jørgen Christensen.
"Optimizing Path Tracing using Noise Reduction Filters".
In *Proceedings of WSCG 95*, pages 134–142, Plzen 1995.

[Kajiya86] James T. Kajiya.
"The Rendering Equation".
Computer Graphics **20** (4), pages 143–149, 1986.

[Keller96] Alexander Keller.
"Quasi-Monte Carlo Radiosity".
In proceedings of *7th Eurographics Workshop on Rendering*, pages 102–111, Porto 1996.

[Keller98] Alexander Keller.
Quasi-Monte Monte Carlo Methods for Photorealistic Image

Synthesis.
Ph.d. thesis, University of Kaiserslautern, 1998.

[Kopp99] Nathan Kopp.
Personal communication.

[Lafortune93] Eric P. Lafortune and Yves D. Willems.
"Bidirectional Path Tracing".
In *Proceedings of CompuGraphics*, pages 95–104, 1993.

[MegaPov00] A free ray tracer that supports photon maps.
Source code and examples are available at:
`http://www.nathan.kopp.com/patched.htm`, Mar. 2000

[Nicodemus77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia. I. W. Ginsberg and T. Limperis: "Geometric Considerations and Nomenclature for Reflectance". *National Bureau of Standards*, 1977

[Niederreiter92] Harald Niederreiter.
Random Number Generation and Quasi-Monte Carlo Methods, SIAM, 1992.

[Pavicic90] Mark J. Pavicic.
"Convenient Anti-Aliasing Filters that Minimize Bumpy Sampling".
In *Graphics Gems I*, eds. Andrew S. Glassner, pages 144–146, 1990.

[Rubinstein81] Reuven Y. Rubinstein.
Simulation and the Monte Carlo Method.
John Wiley & Sons, 1981.

[Rushmeier88] Holly Rushmeier.
Realistic Image Synthesis for Scenes with Radiatively Participating Media.
Ph.d. thesis, Cornell University, 1988.

[Schlick93] Christophe Schlick.
"Customizable Reflectance Model for Everyday Rendering".

In *proceedings of 4. Eurographics Workshop on Rendering*, pages 73-84, Paris 1993.

[Shirley90] Peter Shirley.
"A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes". *Proceedings of Graphics Interface '90*, pages 205–212, 1990.

[Shirley92] Peter Shirley.
"Nonuniform Random Point Sets via Warping". *Graphics Gems III* (David Kirk ed.), Academic Press, pages 80–83, 1992.

[Shirley95] Peter Shirley; Bretton Wade; Phillip Hubbard; David Zareski; Bruce Walter and Donald P. Greenberg.
"Global Illumination via Density Estimation".
In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pages 219-230, 1995.

[Shirley96] Peter Shirley; C. Wang and Kurt. Zimmerman.
"Monte Carlo Techniques for Direct Lighting Calculations".
ACM Transactions on Graphics **15** (1), 1996.

[Tamstorf97] Rasmus Tamstorf and Henrik Wann Jensen.
"Adaptive Sampling and Bias Estimation in Path Tracing".
In "Rendering Techniques '97". Eds. J. Dorsey and Ph. Slusallek. *Springer-Verlag*, pages 285–295, 1997.

[Veach94] Eric Veach and Leonidas Guibas.
"Bidirectional Estimators for Light Transport".
In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 147–162, 1994.

[Veach95] Eric Veach and Leonidas Guibas.
"Optimally Combinig Sampling Techniques for Monte Carlo Rendering".
Computer Graphics **29** (4), pages 419–428, 1995.

[Veach97] Eric Veach and Leonidas Guibas.
"Metropolis Light Transport".
Computer Graphics **31** (3), pages 65–76, 1997.

[Volevich99] Vladimir Volevich, Karol Myszkowski, Andrei Khodulev and Edward A. Kopylov.
"Perceptually-Informed Progressive Global Illumination Solution".
Technical Report 99-1-002, University of Aizu, Japan, 1999.

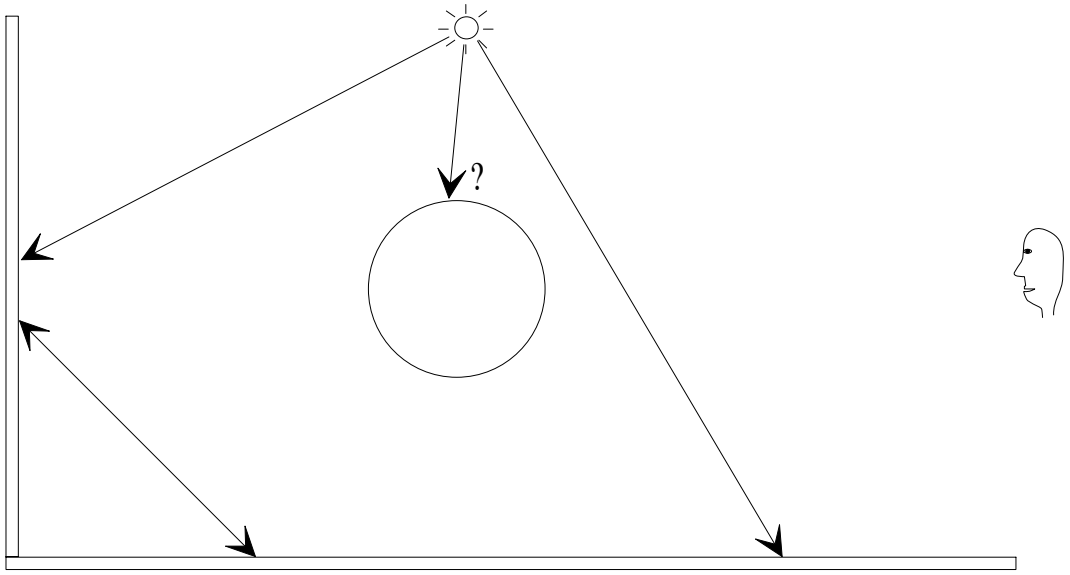
[Ward88] Greg Ward, Francis M. Rubinstein, and Robert D. Clear.
"A Ray Tracing Solution for Diffuse Interreflection".
Computer Graphics **22** (4), pages 85-92, 1988.

[Ward91] Greg Ward.
"Real pixels".
In *Graphics Gems II*, James Arvo (ed.), *Academic Press*, pages 80-83, 1991.

[Zimmerman98] Kurt Zimmerman.
Density Prediction for Importance Sampling in Realistic Image Synthesis.
Ph.d. thesis, Indiana University, 1998.

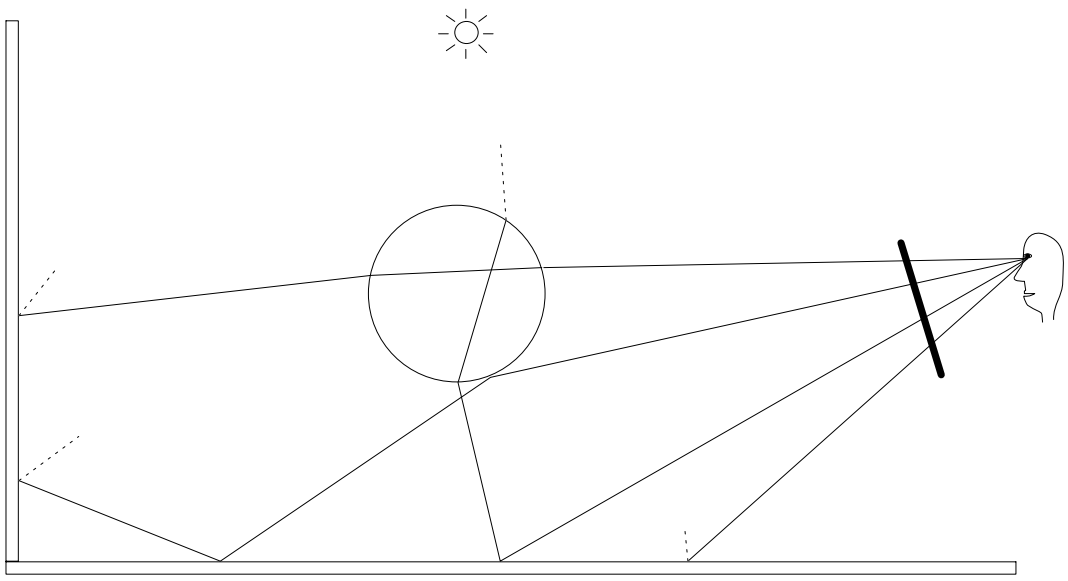
The following pages contains slides illustrating the photon map algorithm

Finite Element Methods (Radiosity)



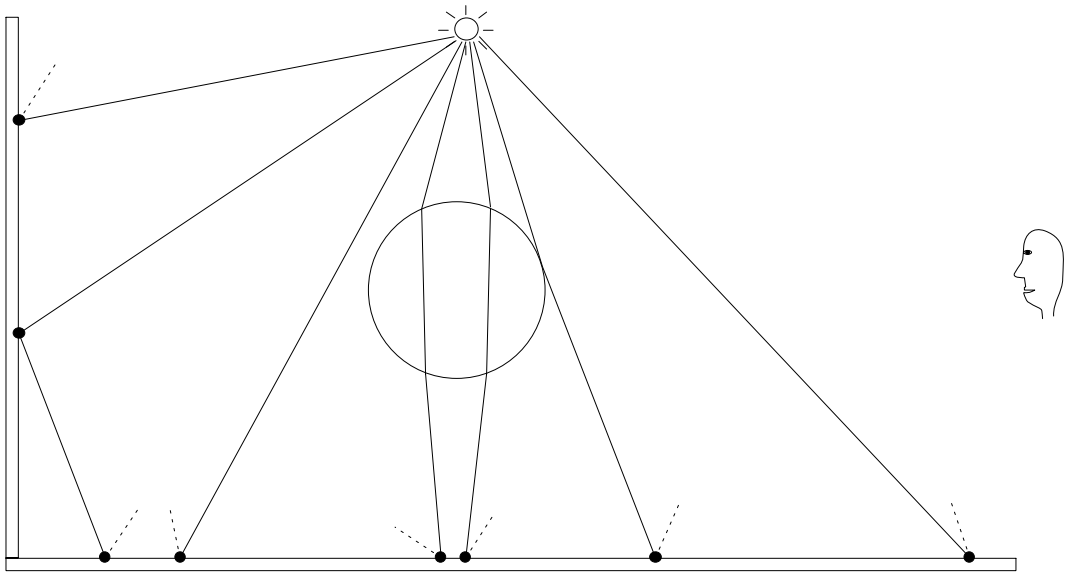
- Discretized Solution
- Non-diffuse reflection is costly
- Many objects are costly
- Specular reflection is difficult

Monte Carlo Ray Tracing

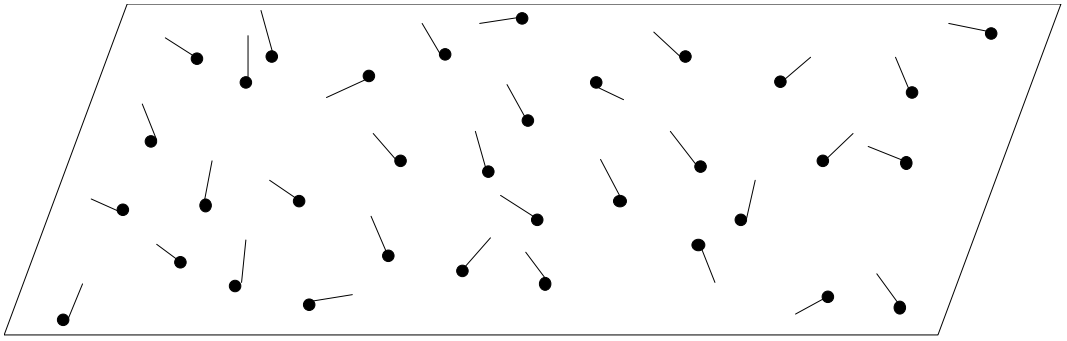


- Noisy Solution or
- Very long rendering times
- Caustics are difficult

The Photon Map



Close-up

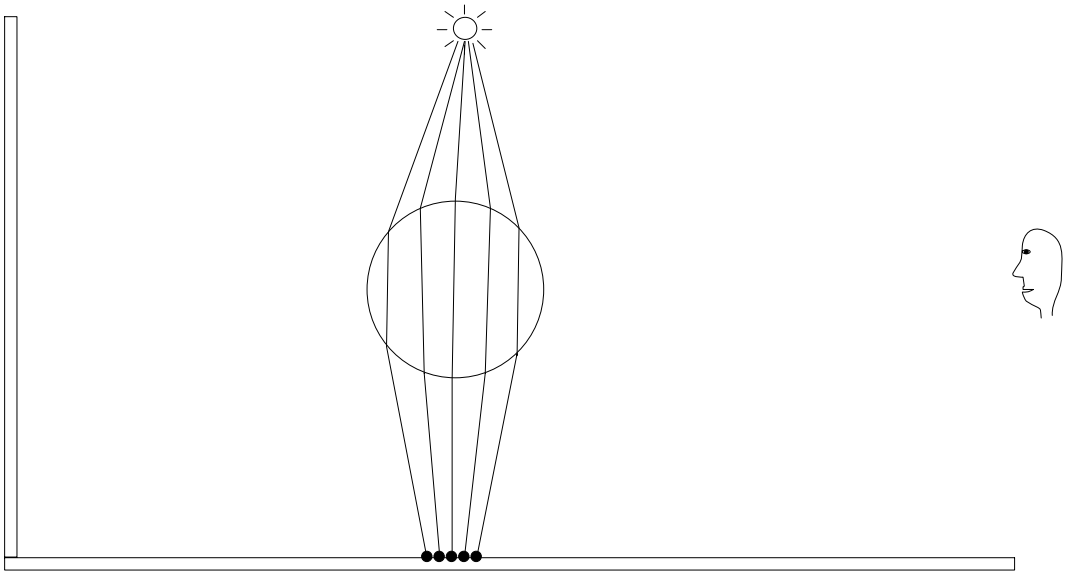


How do we render this?

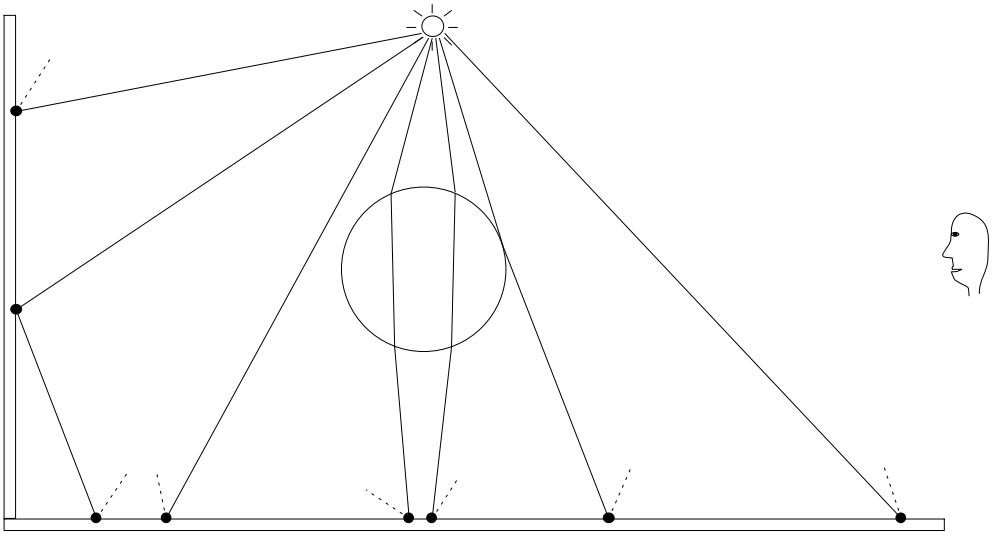
- Visualizing the photon map directly is too costly
- We cannot sample caustics using ray-tracing approaches
  (use the photons directly)
- Use two photon maps

Photon Tracing

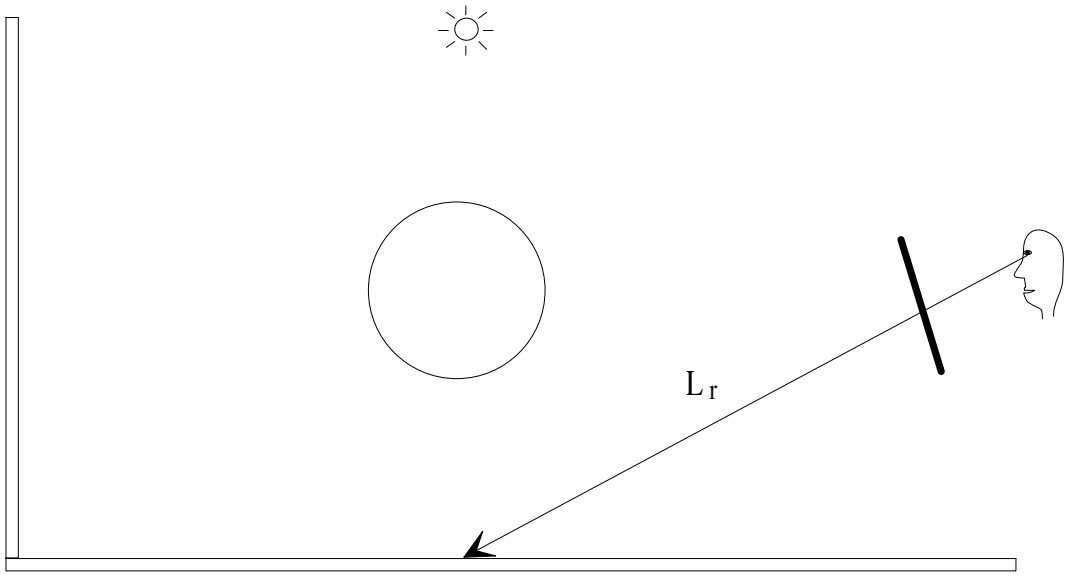
The Caustics Photon Map



The Global Photon Map



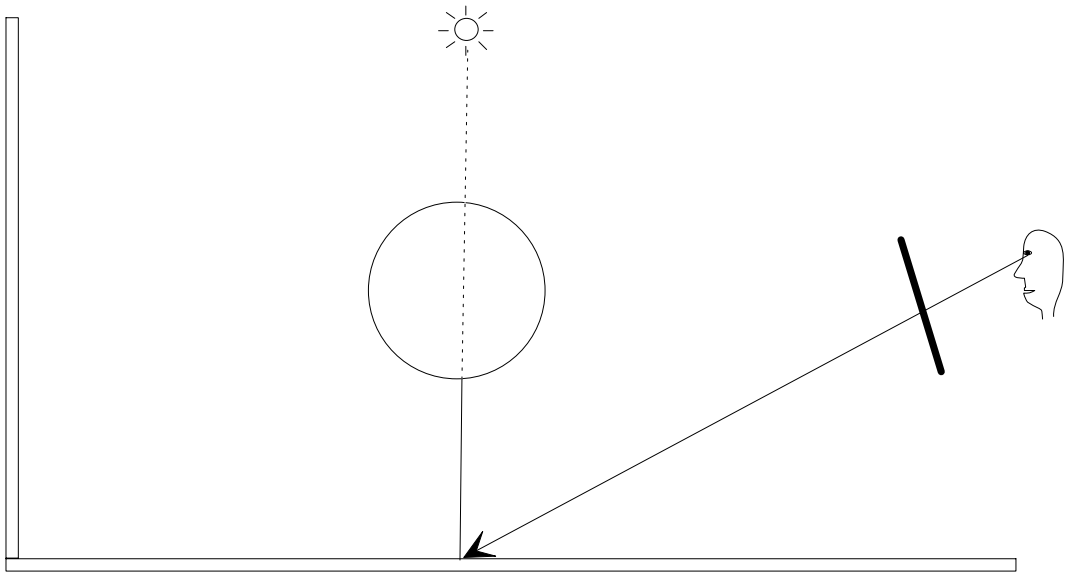
Rendering using Photon Maps



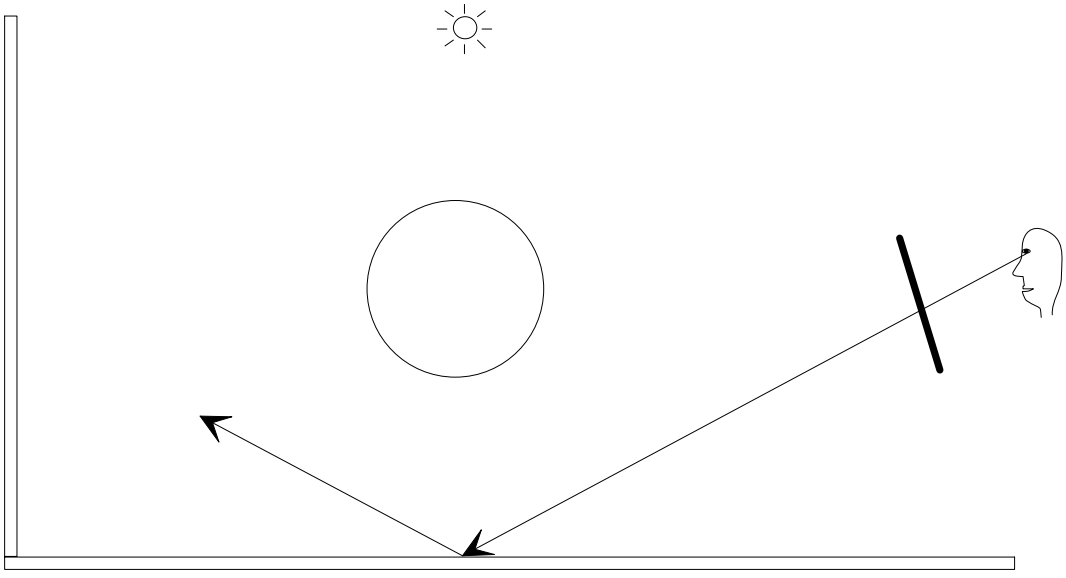
Radiance Components

- Direct Illumination
- Specular Reflection
- Caustics
- Indirect Illumination

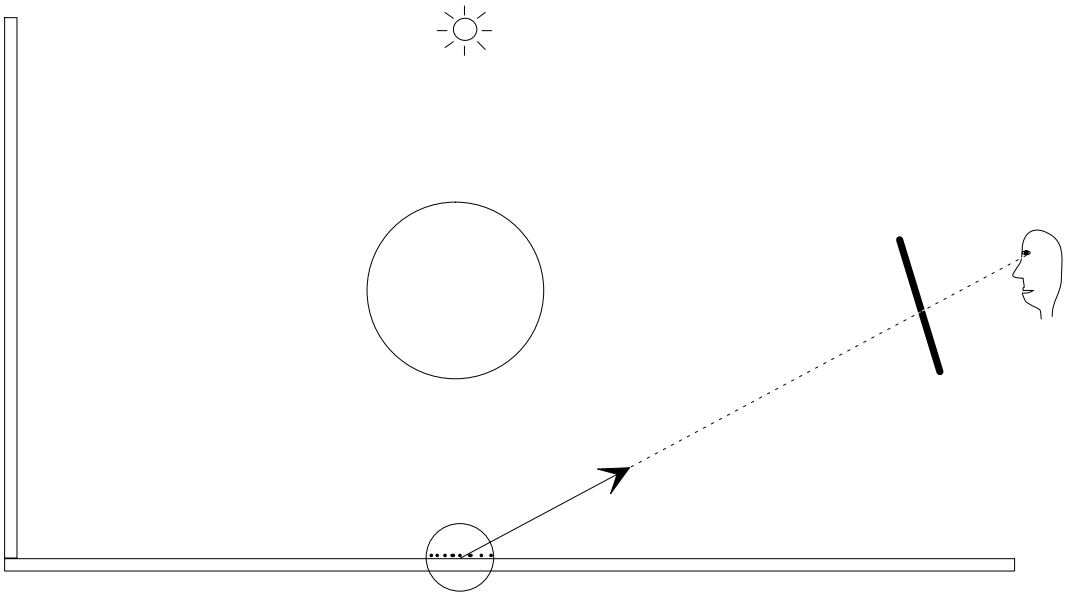
Direct Illumination



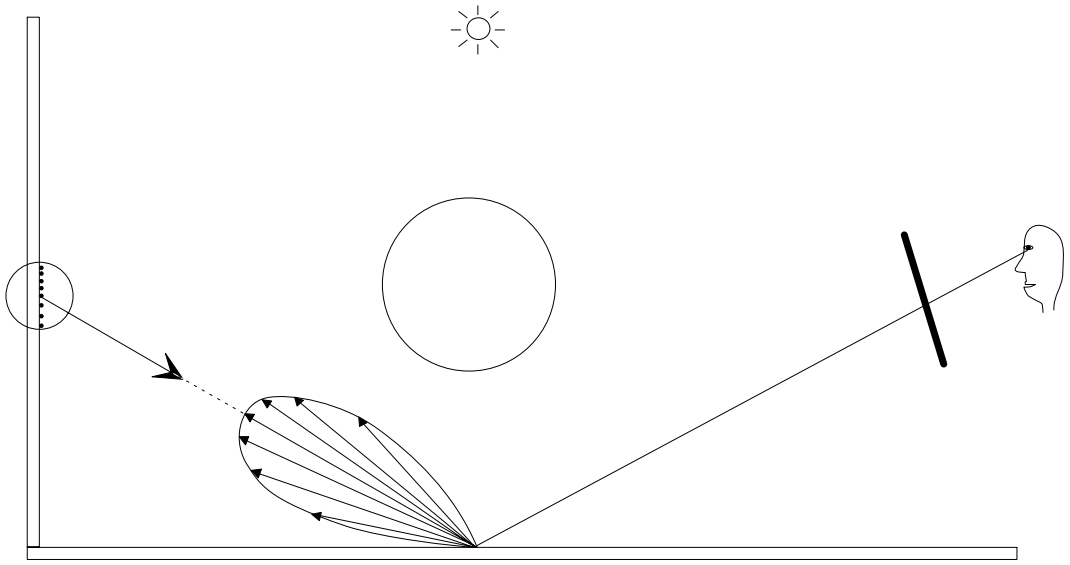
Specular Reflection



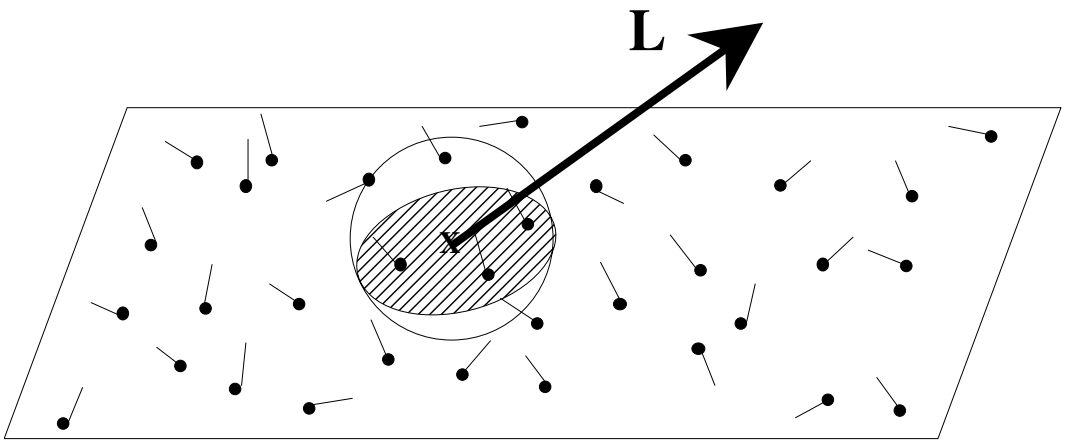
Caustics



Indirect Illumination



Estimating Radiance



$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{p=1}^{N} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p) \frac{\Delta\Phi_p(\mathbf{x}, \vec{\omega}_p)}{\pi r^2}$$