GLOBAL ILLUMINATION METHODS FOR ARCHITECTURAL SCENES

By

György Antal

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
EÖTVÖS LORÁND UNIVERSITY
BUDAPEST, HUNGARY
DECEMBER 2003

Contents

Abstract

Achieving realism has become one of the ultimate goals of computer graphics. The task of photorealistic image synthesis is generating images from a given description of the scene in such a way as to give the viewer the belief that he is looking at a real photograph. This dissertation contains research into image synthesis for obtaining high quality images.

Path tracing is a popular technique for solving the rendering equation. In order to reduce the rendering time of the algorithm, the light paths are terminated randomly using Russian roulette. We analyze aspects of the termination probability with an aim to reduce the variance of the algorithm.

Another very efficient global illumination technique uses bundles of rays for the light transport. It is presented that significant speed-up is achievable if – during light transfer – we separate the finite-element part of the radiance from the Monte-Carlo component.

Motivated by the parallel ray-bundle tracing method, a new algorithm, the perspective ray-bundle tracing is introduced, which is in some aspect superior to its predecessor.

The two algorithms complement each other, therefore we investigate the possible combination of them and conclude that the combination results a very robust algorithm.

Build upon the combined ray-bundle algorithm we propose a non-diffuse global illumination algorithm that is fast enough to be appropriate for interactive walkthroughs and general animations, which is the ultimate challenge of photorealistic rendering.

Acknowledgements

I would like to thank *László Szirmay-Kalos*, my supervisor, for his many suggestions and constant support during this research. Without him this work would never have come into existence. I am also thankful to *Ferenc Csonka*, my research fellow, for his persistent review of my papers and for his appropriate comments and good questions throughout the years. I would like to thank him his friendship too, which helped me many times.

I had the pleasure of working in the Hungarian Computer Graphics Research Group, which is a team of eager graphics enthusiasts in the Technical University of Budapest. I mention here just those who have the same interests and work with me on the rendering field of computer graphics: *Balázs Benedek*, *László Szécsi* and *Gábor Szíjártó*. Over the years, this group has grown up and doubled its size. The team members are wonderful people and their support made research like this possible. I really liked the time, when we worked together.

In the early years of this research I worked at Graphisoft R&D Rt. I am specially thankful to *László Sparing*, who as the Vice President of ArchiCAD Development in Graphisoft helped a lot with his support and care.

Of course, I am grateful to my parents for their patience and love and I also want to thank *Tündi* for her love, support and tolerance that she provided over the years.

I should also mention that this research has been supported by the National Scientific Research Fund (OTKA ref. No.: T042735), the IKTA-00101/2000 project and by Graphisoft R&D and by Intel Inc. Most of the scenes have been modelled by ArchiCAD or by Maya that were granted by Graphisoft and Alias.

Budapest, Hungary — György Antal
November 1, 2003

Chapter 1

Introduction

Computer graphics is the branch of computer science devoted to the creation and processing of images and animations by using computers. One of its subjects is the image synthesis, which is usually named as *rendering*. Since the mid-70s a vast amount of work has been published about rendering. Over the years the motivation has been photorealism, which has a very obscure definition. The essence of the expression was caught very precisely by Hall and Greenberg [HG83] in 1983:

"Our goal in realistic image synthesis is to generate an image that evokes from the visual system a response indistinguishable from that evoked by the actual environment."

It is a very ambitious goal that probably requires virtual reality techniques and brain-machine interfaces, which in the beginning of the XXI. century is still only a science-fiction topic. A less ambitious simplification allows the synthesized images to be displayed only in two dimensions, with a limited resolution and field of view and using only discrete colors. In this approach, the widely used adjective 'photo-realistic' means that the generated image *looks* – on a pixel by pixel basis – as real as a real photograph. This level of realism is needed by the film and the game industry and it stands high in the favour of product design (e.g. car industry).

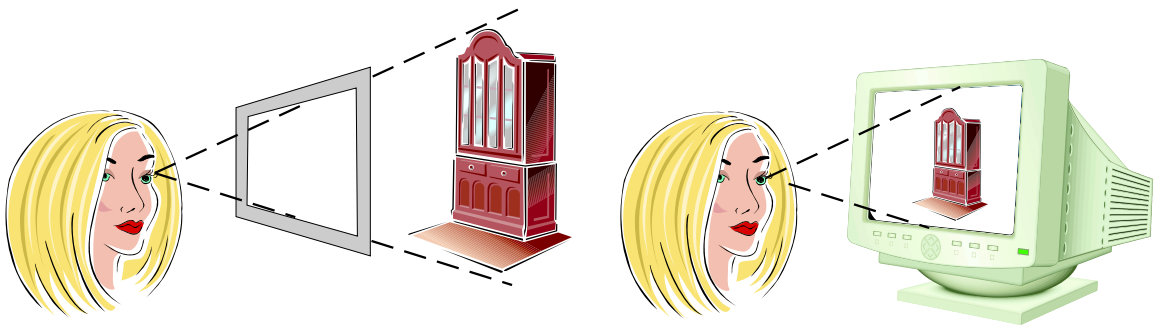


Figure 1.1: The goal of image synthesis

On the other hand, there are application areas where perfection is not such a mandatory requirement. For instance, a flight simulator needs a fairly believable output, but needs not be perfect in every detail. Here the challenge is real-time interactive control. However, these virtual reality applications always have hunger for more believable images at less computation cost.

Another application area is concerned to *architectural CAD*[1] programs. *Architecture* is the art that we walk amongst and live within. It defines our living environments, the buildings and the cities. It can also define virtual environments. There is an emerging need for interesting immersive virtual environments for visiting famous existing or existed sites, especially ones of cultural and historic importance like Taj Mahal in India, which is often called the most beautiful building in the world or the Buckingham Palace in London.

On the other hand, not only the virtual tourists meet architectural scenes. They are considered only as end users in the virtual model creation pipeline. On the other end of the workflow the architect creates the model. Not so many years ago, all phases of architectural design were done manually. Today, computer technology has definitely arrived to be integrated into the architectural practice.

In the pre-computer days, the term "modeling and rendering" would be referred to "presentation drawings and models", and would include an artist's rendering of the building, and for larger projects, a detailed physical model. However, creating a physical mock-up is a tedious work, not to mention the involved cost. Mostly, these are created for presentation purposes that is most useful to the client, rather than design purposes that would be useful to the architect. These days they have been replaced with computer-generated images created using sophisticated modeling and rendering softwares. In the meantime, animated walkthroughs are also being developed to produce the client a more realistic sense of navigation through the proposed building design. In the beginning of the informatics era it was accepted to visualize architectural sites by only drawing the surfaces with a single color and without the calculation of any illumination. More advanced applications considered virtual light sources and calculated only the direct illumination, which is called *local rendering methods*. Recently another term has appeared.

The *global illumination* is a computer graphics term that refers to the ability to accurately simulate and represent the physical characteristics of light within a digital model. The goal is not only to produce better quality renderings but also allow designers to better study the effects of real-world lighting and materials in their designs. Today, architects want to simply work with lighting as it would be in the real world, which is achievable only by the usage of global illumination techniques. The designer wants to place not virtual "never existed" point light sources but real light source fixtures, which has physical interpretations. They prefer to use more meaningful photometric units, for example lumens and candelas or they want to put

[1] Computer Aided Design

for example a 60 watt light bulb in the room, not a point light source which has some kind of unexplainable intensity value 70%. Only by using physical lights can lighting analysis and lighting design make sense.

Architectural scenes usually have the following properties:

- Architectural environments are rendered as interior images. If the building is rendered outdoor, usually it is in the focus, so there are no other buildings on the site.
- They can be quite complex models comparing with mechanical CAD data, where usually only the manufactured item is on the focus. Therefore robust methods are needed for rendering these scenes.
- The scenes consist of physical based light sources and in contrast to other applications not just artificial, but natural lights (e.g. Sun).
- The architect focuses mainly on the 2D section drawings, which is needed for the plan. The creation of the 3D model and the rendering is not crucial for him. Therefore the 3D models may contain errors on the edges or near the junctions, and they are not tesselated properly.

This dissertation was seriously motivated by the successful architectural CAD system called ArchiCAD, which is developed by the multinational Hungarian based software firm Graphisoft where the author of this dissertation worked. At first, the study was aimed to reveal the possibilities of applying a global illumination renderer for architectural scenes, especially for ArchiCAD and if possible, implementing a promising algorithm. In some measure, the title of this dissertation is also originated from this motivation that our algorithm is always targeted towards rendering architectural scenes.

However, modeling, rendering, and animation functionalities extend far beyond the realm of architectural design to be used in many other fields, most notably, in movies and in computer games.

1.1 Computer synthesis of images

The foundation of photo-realism is the calculation of light-object interaction. It proceeds by considering an existing physical model and simulating the behavior with a computer graphics algorithm. Since the computational power of the computers is limited, the original mathematical model for the physical process must be simplified for practical reasons.

Local models considers only the first reflection of light, so the generated *photograph* misses important light contributions. Most serious problems that arise are the inter-reflection and the soft shadows, which are not calculated by this model.

On the other hand, global methods handle the interaction of light between objects. They can produce phenomena as:

- **diffuse interaction**, which is often named as color bleeding (e.g. the diffuse red wall paints the nearby white ceiling to a light pink color)
- **specular interaction**, which can be differentiated into two categories. One type is the *mirror-like* interaction (e.g. a chrome spoon blur and distort the image of its surrounding). The ideal specular case is the perfect mirror. The other type of specular interaction is *caustics* (when the light is focused, e.g. the shimmering waves at the bottom of a swimming pool).
- **soft shadows**, which are displayed without hard edges.

The so-called global illumination algorithms attempt to solve the famous *rendering equation*. Two workable approaches are used in order to solve this equation: finite-element methods and random walk methods. Both kind of methods have some advantages and suffer from some disadvantages. Finite-element methods are good for sparse scenes, where the number of objects (therefore the number of finite-elements) is low. However, this method does not converge to the real solution, only to the solution, which is described by the finite-element tesselation. On the other hand, random walk methods are more robust and are not so sensitive to the complexity of the scene. Because of the different characteristic of the approaches, it is worth combining them. This consideration leads to many hybrid algorithms, which try to keep the advantages of the underlying applied algorithms.

1.2 Objective of the dissertation

This work aims at further improvement of the global illumination algorithms. The improvement means to research completely new algorithms or modify existing ones to gain speed, accuracy or reliability. Most of all, this dissertation is targeted towards the stochastic iteration solution of the rendering problem. The study investigates methods that transfer the light energy by the so called 'bundles' of rays, as it have been proven to be one of the most efficient global illumination algorithm. More specifically the contributions of this work are:

- **Analysis of Russian roulette variance**

  We study the survival probability of the followed ray in random walk global illumination algorithms. When the ray hits a surface, the survival probability is responsible for finishing the walk. We examine the variance of the Russian roulette random walk estimator of gathering type, give an explanation of the formula and show visually the behavior of the term. Based on variance analysis a

criterion for choosing the optimal survival probability is given, which is justified by empirical tests.

- **Extending the parallel ray-bundle iteration**

  We combine continuous and finite-element approaches, preserving the speed of finite-element based iteration and the accuracy of continuous random walks. The basic idea is to decompose the radiance function to a finite-element component that is only a rough estimate and to a difference component that is obtained by Monte-Carlo techniques. Iteration and random walk are handled uniformly in the framework of stochastic iteration. This uniform treatment allows the finite-element component to be built up adaptively aiming at minimizing the Monte-Carlo component. We develop a general technique and we apply it for the parallel ray-bundle iteration.

- **The perspective ray-bundle iteration**

  We generalize the hemicube approach of the diffuse radiosity to handle non-diffuse cases. This generalization is performed by applying stochastic iteration instead of the deterministic iteration used by the radiosity algorithm. Since our new algorithm uses ray-bundles originated from a specific point, we named the new method perspective ray-bundle iteration or stochastic hemicube shooting. We also propose extensions to the basic ray-bundle iteration, which eliminate the annoying artifacts produced by the brute-force implementation or assures better convergence of the method. These extensions are importance sampling, hemicube randomization, trading bias with noise and the constant radiance term.

- **Combination of parallel and perspective ray-bundles**

  We apply multiple importance sampling for combining different ray-bundle strategies into a new algorithm. At first we propose a modification of the multiple importance sampling scheme for being applicable to iteration algorithms. We develop a new balance heuristics, which is then applied for combining 3 stochastic iteration algorithms. They are the parallel ray-bundles with software z-buffer, the parallel ray-bundles with hardware z-buffer, and the perspective ray-bundles.

- **Animation with ray-bundles**

  We use our combined ray-bundle algorithm developed in the previous chapter for walkthrough and for general animation also. Our global illumination can achieve interactive frame rates even on a single computer, which was an unimaginable idea some years ago. To achieve this, we modify the radiance representation in the combined algorithm. Other novel idea concerns the radiance updates

in general animation, which is performed in a separate iteration phase of the algorithm.

1.3 Organization

This dissertation is organized as follow:

- **Chapter 2:**

  The used notation of optics and the used definitions are given. Then the global illumination problem is presented and an overview of useful methods can be found.

- **Chapter 3:**

  This section begins with the concept of Russian roulette, which is a widely used technique in random walk methods. This is followed by the analysis of the variance of the Russian roulette estimator. The analysis is separated into two cases, for both of them the results are then presented. Finally, the conclusion – as a proposal for the application of Russian roulette – is given.

- **Chapter 4:**

  The stochastic iteration is detailed in this section. Two types of random operator is presented which leads to the single ray based iteration and the parallel ray-bundle iteration. The later is then made more efficient by separating the finite-element main part and the Monte-Carlo part of the radiance.

- **Chapter 5:**

  This section introduces a new stochastic iteration approach based on hemicube shooting. This is followed by several improvements for the basic algorithm. The algorithm was implemented and the results are also presented here.

- **Chapter 6:**

  This section begins with the discussion of combining global illumination algorithms. Then the most efficient combination strategy, the multiple importance sampling is reviewed and extended for iteration algorithms. Then a combination of three ray-bundle methods is presented, which is followed by the results and the error analysis of the combined method.

- **Chapter 7:**

  This section is started by studying the recent developments and trends in global illumination animation. The algorithms are separated into offline animation and interactive animation. Then we study the possibility of interactive global

illumination. The combined ray-bundle strategy is then extended to achieve this task. After discussing the walkthrough animation, the modifications needed for the general animation is presented. Later, the implementation details are given and the results with the achievable frame rates are presented.

- **Chapter 8:**

  Conclusion of the dissertation is presented here, along with some recommendation for future research.

- **Appendix:**

  This section describes a general global illumination framework. It is not a scientific breakthrough in its own, however, it sketches the general architecture that every rendering systems (especially ones with global illumination support) must follow. It was developed in the context of this dissertation; therefore all of our algorithms in this thesis are implemented within. Other global illumination methods (not covered by this dissertation) are also developed and listed in the last part of this section.

Chapter 2

Background

This chapter introduces some important background material which underpins much of the work in this dissertation. We assume that energy is quantized into small, discrete pockets, which we often call particles, and describe the flow of energy by keeping track of the number of particles (photons). The technique for analyzing the flow of moving particles is known as *transport theory*. It was first developed by simulating the activity of neutrons in atomic reactors, but it is also used for gas and plasma dynamic and last but not least for simulating light.

A particle is a concentration of energy in space and time, whereas a wave is spread out over a larger region of space and time. Since 1925 we know, thanks to Louis de Broglie [dB25], that any moving particle (electron, mater) has wavelength associated with it, therefore light also has a dual nature, depending on the processes it is subjected to. It is both corpuscular (small discrete particle) and wave-like. The *wave model* is described by Maxwell's equations and captures effects such as diffraction, interference, polarization and dispersion. Interference makes the brilliant colors that can be seen e.g. on soap bubbles, and diffraction occurs for some soft shadows (not all) and some light bleeding around the edges of objects. The dispersion is the phenomenon that in dielectric medium whose refractive index varies with wavelength each component of the non-monochromatic incident wave is refracted through a different angle. This effect can be seen with a glass prism that is placed in a beam of white light. In this dissertation we consider light as photon particles and drop modeling those phenomena that are not specified by the particle model.

The field of *radiometry* defines a vocabulary of quantified light energy. This gives us the tools for specifying the rendering equation. There is another concept called *photometry*, which is the study of how human observer responds to light. Since it is a nonlinear response, we prefer to use the radiometric definitions.

The mathematical model for the interaction of the light with the scene is the *geometrical optics*, or ray optics. It uses the concept of rays, which have position and direction but no phase information, to model the way electromagnetic radiation travels. Optics approximates how the electromagnetic radiation behaves. It is a very

convenient model to use when the smallest dimension of the optical system is much larger than the wavelength of the incident radiation. But if the dimension of the optical components is about the size of a wavelength or smaller, a different model (either the *electromagnetic wave optics* model or the *quantum optics* model[1]) must be used. The geometrical optics model captures effects such as emission, reflection, refraction.

2.1 Nomenclature and definitions

Directional notation

We usually refer to a given direction relative to a point $\vec{x}$ on a surface. We use the local *spherical coordinate system* at the referred point, and parameterize the direction using $\vec{\omega} = (\theta, \phi)$, where θ is the angle of $\vec{\omega}$ and the surface normal, and ϕ is the angle of the projection of $\vec{\omega}$ and a reference axis defined on the tangent plane. By definition, $|\vec{\omega}| = 1$, so we think of direction vectors as identifying points on a unit-radius sphere around the origin.

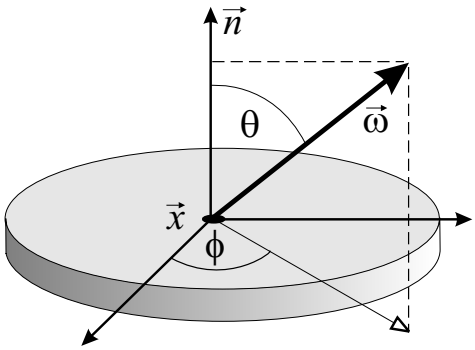


Figure 2.1: Direction parameterization

Solid angle

The solid angle is used to measure the portion of the space occupied by an object as seen from a point $\vec{x}$. According to the definition it is the area of the projection of the object onto the unit sphere around $\vec{x}$. It is measured in *steradians* $[sr]$. Note that the shape of the area does not matter at all. Any shape on the surface of the sphere that holds the same area will define a solid angle of the same size. The corollary of the definition is that a whole sphere consists of 4π sr, a half sphere 2π sr.

[1] It captures effects such as fluorescence and phosphorescence. Since this model is too detailed, it is generally not considered in computer graphics. Exception is given in the PhD dissertation of Alexander Wilkie [Wil01].

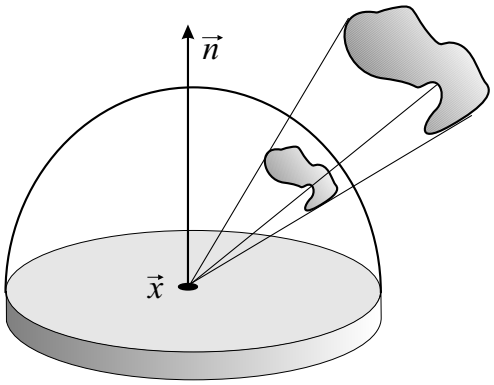


Figure 2.2: Solid angle of the surface point $\vec{x}$

The most important *radiometric terms* are:

Radiant energy

In the particle model of light, each photon has some energy, which depends on the wavelength of the matter. This quantum energy is defined by $E = \hbar f$, where f is the frequency of the incident energy of the particle (interpreted as a wave) and $\hbar$ denotes the Planck's constant. The radiant energy is measured in joules (J).

Radiant flux

The energy flowing through a surface per unit time is called *radiant power* or *radiant flux* at that surface:

$$\Phi = \frac{dQ}{dt}. \tag{2.1}$$

The radiant flux is measured in watts (W). In computer graphics, we often use the word energy, flux and power interchangeably. This is validated by the fact that we work with a steady-state scene with no time-dependent variables.

Radiant flux area density

It is the density of radiant energy flowing through a unit area per unit time:

$$u = \frac{d\Phi}{dA} = \frac{dQ}{dtdA}. \tag{2.2}$$

It is measured in watts per square meter (Wm^{-2}). Since this value is scalar, we do not know if the energy arrives to the surface or departs from it. To make a clear distinction, if the energy is arriving at the surface, we call it *irradiance* (denoted by E), and if the flux is leaving a surface, it is called *radiosity* (denoted by B).

Radiant intensity

The *radiant flux area density* requires that the source of energy has some (finite or differential) area. For point sources another quantities must be introduced. The measure of the radiant energy leaving a point in the direction $\vec{\omega}$, per unit solid angle is called *radiant* intensity:

$$I = \frac{d\Phi}{d\vec{\omega}}. \tag{2.3}$$

It is measured in watts per steradian (Wsr^{-1}).

Radiance

It might be the most important quantity in radiometry. *Radiance* is flux per unit projected area per unit solid angle:

$$L = \frac{d^2\Phi}{\cos\theta dA d\vec{\omega}}. \tag{2.4}$$

It is measured in watts per steradian per square meter ($Wsr^{-1}m^{-2}$). Intuitively, radiance expresses how much power arrives at (or leaves from) a certain point on a surface, per unit solid angle, and per unit area. Note that this definition does not consider the regular surface area, but it considers the area projected perpendicular to the incoming direction. The verification is presented in Figure 2.3.

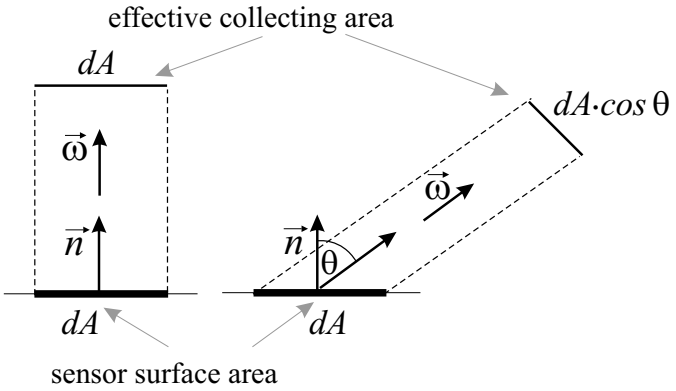


Figure 2.3: The origin of the cosine term

The effective collecting area of the sensor changes with the angle. If the incoming direction equals with the surface normal (left image), the sensor's effective collecting area is the same as its surface area. However, as the angle between $\vec{\omega}$ and $\vec{n}$ increases (right image), its effective collecting area decreases. The effective collecting area is also called the *projected area*, and it equals to the surface area times the cosine of the angle between the incoming direction $\vec{\omega}$ and the surface normal $\vec{n}$.

The radiance is the most useful quantity for most light detectors (e.g. cameras), since it is independent of the size of the surfaces or the aperture of a given solid angle. Therefore, to construct the image, it is enough to know the radiance leaving all surfaces in all directions.

Radiance function

The function which gives the radiance at any point on a surface is called *radiance function*. It is a high variate function, since it is (as other radiometric terms also) a function of position, direction, wavelength, time, and polarization:

$$L(\vec{x}, \vec{\omega}, \lambda, t, p).$$

We usually cannot consider all parameters, so some simplifications need to be introduced. The polarization is only considered in the wave model of light. Since computer graphics usually works with scenes in equilibrium, the time dependency can be also eliminated. Unfortunately, the wavelength dependency cannot be dropped so easily. The simplest solution is to evaluate the radiance function for at least three representative wavelengths (this can be the most common red, green, blue component). It reduces the dimensionality of the radiance function to a 5 (in the absence of participating media it is only 4). Computing the approximation to the radiance function ($L(\vec{x}, \vec{\omega})$) is the main goal in global illumination algorithms.

An important property of the radiance is that if participating media is not present, it is invariant along straight lines.

2.1.1 Rendering equation

The formal definition of the most popular mathematical model for light transport was presented in 1986 by Kajiya [Kaj86]. It describes what happens on a surface at a point $\vec{x}$. It is a very general mathematical statement, and global illumination algorithms can be categorized by the approach for solving this formalized problem. The *rendering equation* is a Fredholm type integral equation of the second kind:

$$L(\vec{x}, \vec{\omega}) = L^e(\vec{x}, \vec{\omega}) + \int_{\Omega_{4\pi}} L_i(\vec{x}, \vec{\omega}') \cdot f_s(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta' d\omega' \tag{2.5}$$

where L^e is the emitted radiance, $\Omega_{4\pi}$ is the bounding sphere at surface point $\vec{x}$, f_s is the Bidirectional Scattering Distribution Function (BSDF), and θ' is the angle between the incoming direction ω' and the surface normal at point $\vec{x}$.

The unknown incoming radiance L_i can be calculated using the outgoing radiance values. We have to trace back the path they arrive from. This means tracing a ray from $\vec{x}$ in the direction of the incoming radiance. This results in some surface point $\vec{y}$, which is given by the ray casting operator $\vec{y} = rc(\vec{x}, -\vec{\omega})$. The incoming radiance is

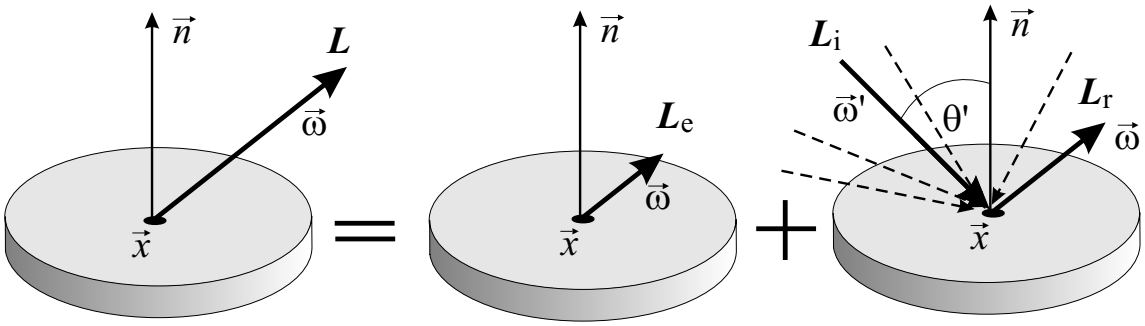


Figure 2.4: The outgoing radiance is the function of the emitted radiance and the reflected radiance, which is an integral over all incoming directions

then calculated from the outgoing radiance: $L_i(\vec{x}, \omega') = L(\vec{y}, \vec{\omega})$. The problem is then stated recursively, since this new outgoing radiance value is also described exactly by another rendering equation.

The BSDF is usually simplified by not taking into account the refractivity. In this case the Bidirectional Reflection Distribution Function (BRDF) can be used, and the integration domain can be also reduced to the hemisphere ($\Omega_{2\pi}$). Then the rendering equation can be written in the following form:

$$L(\vec{x}, \vec{\omega}) = L^e(\vec{x}, \vec{\omega}) + \int_{\Omega_{2\pi}} L(rc(\vec{x}, -\vec{\omega}'), \vec{\omega}') \cdot f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta' d\omega'. \tag{2.6}$$

Defining the following notation for the integral operator

$$(\mathcal{T}L)(\vec{x}, \vec{\omega}) = \int_{\Omega_{2\pi}} L(rc(\vec{x}, -\vec{\omega}'), \vec{\omega}') \cdot f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta' d\omega'$$

we get the sort form of the rendering equation:

$$L = L^e + \mathcal{T}L. \tag{2.7}$$

The simplest solution of the rendering equation considers only the direct illumination of the surfaces by the light sources. This eliminates the recursive nature of the rendering equation. It is called *local illumination*. *Global illumination* methods consider all light/surface interactions and solve a more general problem.

There is another very elegant mathematical model for simulating the light transport, the *path integral formulation* introduced by Veach [Vea97]. Given a path $\bar{z} = (\vec{z}_1, \ldots, \vec{z}_n)$ with length n, the radiance function can be calculated:

$$l(\bar{z}) = l(\vec{z}_1, \ldots, \vec{z}_n) =$$

$$L^e(\vec{z}_1, \vec{\omega}_1)\, G(\vec{z}_1, \vec{z}_2)\, f_s(\vec{\omega}_1, \vec{z}_2, \vec{\omega}_2)\, G(\vec{z}_2, \vec{z}_3) \ldots f_s(\vec{\omega}_{n-2}, \vec{z}_{n-1}, \vec{\omega}_{n-1})\, G(\vec{z}_{n-1}, \vec{z}_n), \tag{2.8}$$

where ω_i is the direction from point $\vec{z}_i$ to point $\vec{z}_{i+1}$ and G is the geometry term:

$$G(\vec{x},\vec{y}) = V(\vec{x},\vec{y}) \cdot \frac{\cos\theta_i \cos\theta_o}{|\vec{x}-\vec{y}|^2} \tag{2.9}$$

where $V(\vec{x},\vec{y})$ is the visibility function (it is 1 if the two point can see each other, and 0 otherwise), and the θ_i and θ_o are the angles between the $\vec{x} \to \vec{y}$ line and the surface normals at $\vec{x}$ and at $\vec{y}$ respectively. The geometry factor appears, because the integration over the projected solid angle is replaced by integration over surfaces. Note that the paths which have different length have different radiance function (the number of variables in the radiance function changes).

If $\bar{\Omega}_n$ denotes the domain of paths of length n, the domain of path space can be given by:

$$\bar{\Omega} = \bigcup_{n=1}^{\infty} \bar{\Omega}_n.$$

If $d\mu(\bar{z}) = dA_1 \times \ldots \times dA_n$ is the area product measure, then the integration of radiance over the domain of paths can be written as:

$$I = \int_{\bar{\Omega}} l(\bar{z}) d\mu(\bar{z}) = \sum_{n=0}^{\infty} \int_{\bar{\Omega}_n} l(\bar{z}) d\mu(\bar{z}) \tag{2.10}$$

2.2 Rendering

The original definition of the word *rendering* in the architecture literature says: rendering (*noun*) is an architects representation of the inside and outside of a finished building, drawn in perspective. Notice that this definition does not say anything about photo-realism or about the nowadays widely needed requirement that the rendered image must be as similar to the real world as possible. In these modern days men and machines developed some kind of symbiosis, and the rendering is not performed any more by human manual work (paint, brush, lead pencil), but via extensive usage of the computer. The first idea was to use the computers to replace the brush and the pencil. The advantage from using computers is the digital storagebility, the modifiability and reusability of the resulted image. The animators at Disney's studio used this type of rendering for long. Another technological leap was forcing the computers to automatically generate the corresponding snapshot of a three-dimensional world from the given viewpoint. Since this is the current conventional meaning, in this dissertation, the expression "rendering" corresponds to computer generated 3D-rendering.

When one imitates to photograph the real world, in the first place the precise 3D description of the scene is needed. The world representation must be stored in the memory of the computer. First of all, the scene around us consists of 3D objects which

have shapes of they own. Another input for the rendering process is the materials in the scene. For the physical simulation of the light, we also need the sources of light power. And finally the picture is taken from a specific viewpoint.

The rendering problem can be expressed as a quadruple [Kel97]:

$$\langle S, f_r(\vec{\omega}', \vec{x}, \vec{\omega}), L^e(\vec{x}, \vec{\omega}), W^e(\vec{x}, \vec{\omega}) \rangle, \tag{2.11}$$

where S denotes the geometry of surfaces, f_r concerns the material properties of the surfaces, L^e is the emitted radiance and W^e is the measuring function. We present brief descriptions for these components in the consecutive sections.

2.2.1 Geometry

Over the years many approaches have been developed, which can describe the geometry of the object. One of the first, and still the most popular one is the boundary representation. Other possible descriptions are the solid representation and the point set representation.

Boundary representation

The *B-rep* model – also called surface model – defines the boundary of the bodies. The surface is divided into basic elements (polygons, patches, spline surfaces) and the verge of the polygons is usually coincides with the discontinuity of the boundary. For algorithmical reasons the complex smooth surfaces is sometimes also divided into a more detailed representation.

An *implicit surface* is given by an implicit function:

$$f(\vec{x}) = 0.$$

The most popular types for this approach is the quadratic surfaces and the height fields.

The *polygon surface* is far the most widely used B-rep model. The patches are bounded by edges, and the edges attach each other in vertices. The most primitive polygon has just 3 vertices. The simplicity of the triangle validates the application of it on the graphics hardware. The term *mesh* is usually associated with a connected set of polygons (most of the time triangles). Notice that a mesh may not define a close body. Throughout this dissertation it is assumed that the scene is represented by polygons. Without loss of generality the triangle patches are assumed.

The *parametric surface* representation is defined by a two dimensional function:

$$\vec{r}(u, v), \qquad u, v \in [0, 1].$$

The most famous is the Non-Uniform Rational B-Spline (NURBS), which is the generalization of non-rational B-splines and non-rational and rational Bezier surfaces

[Far97]. However, since the graphics hardware cannot handle this surface type, it is usually approximated by polygon mesh.

The polygon models are quite faceted. Parametric surfaces are smooth and the derivative is also continuous. The *subdivision surface* stands between the two models. It is defined by a coarse mesh and a subdivision rule. The smooth surface is generated as a limit of a sequence of refinements.

For further discussion on surface definitions for forward and reverse engineering refer to [VMC97, RVW98, RV00], for transformations refer to [SKe95, Kra89, Her91, Lan91]. Beside the B-rep model, there are other techniques for storing the geometry of the scene such as *solid modeling*, *voxel representation* [PS85, Vid93, CSK98, CMSK97, PRE95] or the *point set* data [MA01, CS02].

2.2.2 Materials

The material gives the definition of how the surfaces reflect, refract or emit light.

"Reflection is the process by which electromagnetic flux, incident on a stationary surface or medium, leaves that surface or medium from the incident side without change in frequency. The reflectance (albedo) is the fraction of the incident flux that is reflected." (Nicodemus et al. [JJHL77])

The mathematical model for the light-surface interaction at point $\vec{x}$ is characterized by the Bidirectional Reflectance Distribution Function abbreviated as BRDF:

$$f_r(\vec{\omega}', \vec{x}, \vec{\omega}) = \frac{dL(\vec{x}, \vec{\omega})}{L_i(\vec{x}, \vec{\omega}')cos\theta' d\vec{\omega}'}, \tag{2.12}$$

where $\vec{\omega}'$ and $\vec{\omega}$ is the incoming and the outgoing direction respectively, L is the outgoing, L_i is the incoming radiance and θ' is the angle between the surface normal and the incoming direction $\vec{\omega}'$ (see figure 2.5).

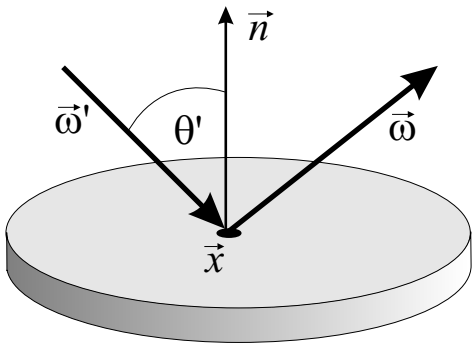


Figure 2.5: Interpretation of the BRDF

Helmholtz reciprocity

One reason for using the cosine factor in equation (2.12) is that for physically plausible materials the BRDF is symmetric. This property is referred as the Helmholtz reciprocity principle, which allows exchanging the incoming and outgoing directions:

$$f_r(\vec{\omega}', \vec{x}, \vec{\omega}) = f_r(\vec{\omega}, \vec{x}, \vec{\omega}').$$

The reciprocity is an important constraint, especially for those algorithms that compute the distribution of light by starting paths from the light sources and starting paths from the camera simultaneously. These algorithms assume that light paths can be reversed, and the BRDF holds the reciprocity principle. Denote that the BRDF is not bounded, and cannot be negative: $0 \leq f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \leq \infty$.

The definition of the BRDF can be easily generalized for refraction also. For transmissive surfaces, the Bidirectional Transmission Distribution Function (BTDF) is the appropriate measure, which contains only the hemisphere on the back side[2] of the surface as a domain for the outgoing direction $\vec{\omega}$. For general surfaces the BRDF and BTDF can be combined into the Bidirectional Scattering Distribution Function (BSDF) with the difference that the BSDF is defined over the full hemisphere for both incoming and outgoing directions[3].

Conservation of energy

Beside the BRDF, another useful measure is the fraction of the incoming radiance (from a single direction $\vec{\omega}'$) that is reflected back over the hemisphere. This is called the *reflectance* or the *albedo* and can be expressed as:

$$a_r(\vec{x}, \vec{\omega}') = \int_{\Omega_{2\pi}} f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta d\omega \tag{2.13}$$

Due to the law of energy conservation the a_r must be smaller than 1. If the transmittance is not neglected, the total scattering albedo $a = a_r + a_t$ should be less than 1. A noticeable corollary of the definition 2.13 that the albedo depends on the incoming direction $\vec{\omega}'$.

Most reflection models separates the materials into a small number of categories (see figure 2.6). In the context of computer graphics the following classes are typically considered:

Specular

Also called mirror reflection. Materials of this type reflect light only in one specific direction. The BRDF value of a perfectly specular surface is 0 everywhere except in one direction, hence the value for that direction is infinite. This BRDF is usually defined by a Dirac-delta (δ) function.

[2] in the direction of the inverted surface normal

[3] it can be considered as a combination of 2 BRDF and 2 BTDF, one for each side of the surface

Diffuse

These materials reflect the light in all directions with equal probability. The value of the BRDF is constant (independent of the incoming and the outgoing directions), and the relation of the BRDF and the albedo is given by:

$$a_r(\vec{x}, \vec{\omega}') = f_r(\vec{x}) \cdot \int_{\Omega_{2\pi}} \cos\theta d\omega = f_r(\vec{x}) \cdot \pi.$$

Glossy

Most materials are neither diffuse nor perfectly specular, but they are somewhere between them. They are responsible for the hazy mirror like appearance of the surfaces. It is difficult to model them with mathematical formulae. [Neu01, NN89, NNSK98, NNSK99]

Mixed reflections can be defined as a combination of the previous 3 BRDF types. The same kind of categories can be defined for materials given by BTDF, since the transmissive function can also behave as a diffuse, specular or glossy surface.

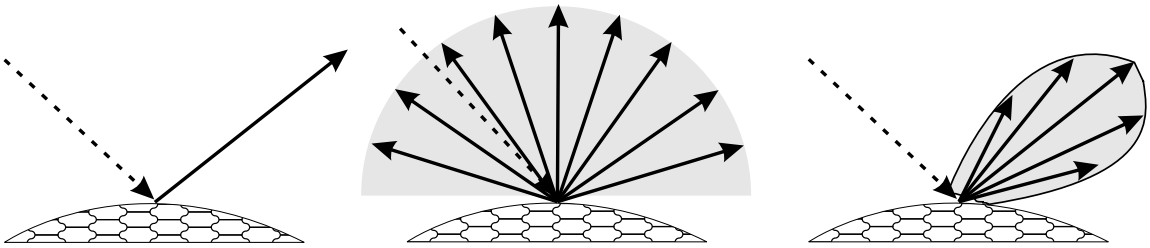


Figure 2.6: Different reflection types: specular, diffuse and glossy

2.2.3 Light sources

Light can be emitted in different ways: by thermal sources such as the sun, or by quantum effects such as fluorescence where materials absorb energy at some wavelength and emit it at some other wavelength. Since we use the optic model of light, we consider light sources as surfaces which produce a specific amount of flux. Manufacturers of lamps and lamp fittings issue diagrams that show the spectral distribution of the light and the distribution of the intensity in all directions[4]. These general light distribution functions are not favoured by all in graphics algorithms. They usually consider only *diffuse emitter*s, which produce light width equal radiance everywhere. The constant radiance implies that the power of this surfaces can be calculated as:

$$\Phi = \int_{Area} \int_{\Omega_{2\pi}} L(\vec{x}, \vec{\omega}) cos\theta d\vec{\omega} dA = L \cdot \int_{Area} dA \int_{\Omega_{2\pi}} cos\theta d\vec{\omega} = L \cdot Area \cdot \pi$$

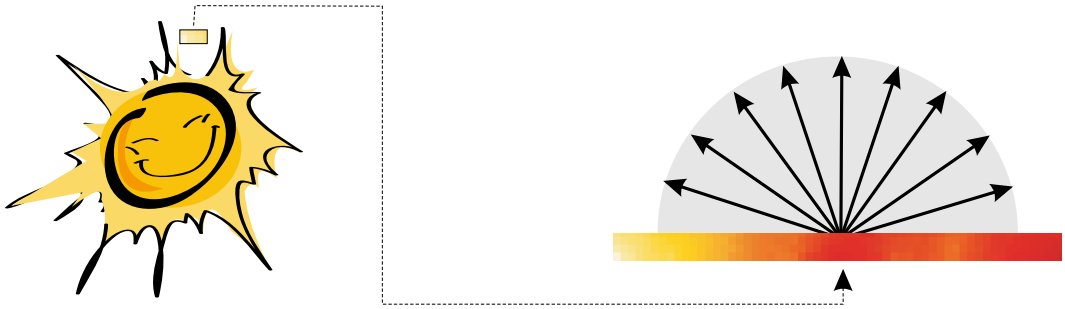


Figure 2.7: Most area light sources are modelled as diffuse emitters

For example the Sun generates $3.91 \cdot 10^{26}$ watt energy and has a surface area of $6.07 \cdot 10^{18} m^2$. This implies that the radiance L of the Sun is $2.05 \cdot 10^7$ watt ($\Phi = L \cdot Area \cdot \pi$). Considering the solid angle of the Sun, the power reaching earth on 1 m^2 square is 1373.5 watt.

The realistic model of light uses only area sources. However, for speed reasons, computer graphics algorithm uses abstract light sources without physical plausibility. The most common types are:

Point light: It has a well defined position in the 3D world and has no extent. The light intensity is attenuated proportional to the square of the distance of the illuminated object.

Directional light: The direction and the intensity of the light is the same everywhere. Light is considered to originate from the point of a surface that is placed infinitely far away.

Spot light: It has a cone of effect, e.g. a desk lamp or an electric torch that emits light from a specific point along a specific direction vector and constrained within a solid angle.

Sky light: It is modelled as a suitably large dome (hemisphere) above the scene. In ray tracers, the outdoor conditions are usually described by the combination of a sky light and a directional light imitating the sun.

Ambient light: The intensity of the light is the same in all points and in all directions. It is used as a rough estimate for multiple interreflections.

2.2.4 Measurements

After solving the rendering equation, the radiance of the surface points in any direction can be determined. A measurement is the response of a certain hypothetical

[4]IES, CIBSE, EULUMDAT

(radiance) sensor placed in the object space, which is defined by the *response function* $W_e(\vec{x}, \vec{\omega})$. The measurement C is the total response of the sensor, which is given by the *measurement equation*:

$$C = \int_A \int_\Omega W_e(\vec{x}, \vec{\omega}) L_i(\vec{x}, \vec{\omega}) cos\theta d\vec{\omega} dA = \mathcal{M}L, \tag{2.14}$$

where A is the total surface of the scene. For the sake of abbreviating the equation, the $\mathcal{M}$ operator is introduced, which is called *radiance measurement operator*.

To obtain the image, we must compute the specific measurement for each pixel. In computer graphics usually the simplest perspective camera model is used. It is the pinhole camera, which assumes that the aperture of the camera has a negligible size. In this context, the center of the perspective projection is referred as the eye point or the viewpoint or the position of the camera. The projection of a scene point $\vec{x}$ is then obtained as the intersection of a line – that is passing through this point and the center of projection – with the retinal plane. However in computer graphics, instead of the retinal plane, we use the image plane, which is placed not behind, but in front of the camera (see figure 2.8).

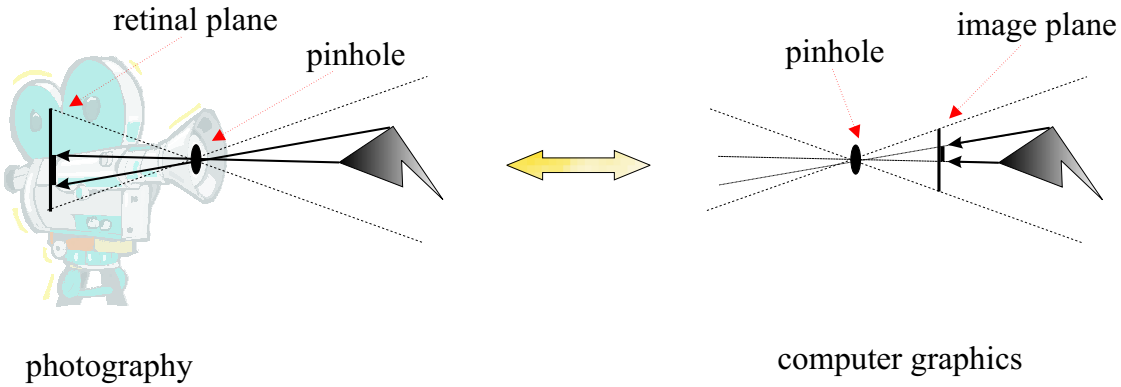


Figure 2.8: Projection of an object in photography and in computer graphics with the pinhole camera model

For the pinhole camera model the response function of a pixel is

$$W_e^{pix}(\vec{x}, \vec{\omega}) = \begin{cases} 1 \cdot \delta(\vec{x} - e\vec{y}e), & \text{if } \vec{\omega} \text{ goes through pixel} \\ 0, & \text{otherwise} \end{cases}$$

2.3 Global illumination

Global illumination algorithms are intended to solve the rendering equation (2.6) without ignoring the recursive nature of the model. They aimed to simulate all types of light paths. Unfortunately, the computational power of the computers cannot

afford to solve the general problem. For this reason over the years different methods have been developed, which simulate only a specific subset of the light paths domain. To be able to differentiate between them, we use a slightly extended form [Suy02] of the notation that was developed by Heckbert [Hec91]. This notation is originated from the formal languages theory and is based on regular expressions:

- E : eye point
- L : light source
- D_r, D_t : diffuse (incoherent) reflection and transmission
- G_r, G_t : glossy reflection and transmission
- S_r, S_t : specular reflection and transmission
- $|$: 'or' operator
- X^* : iteration, zero or more occurrences of X
- X^+ : iteration, one or more occurrences of X
- X^k : iteration, exactly k occurrences of X

To simplify the notation, the reflection and transmission can be grouped together:

- $D = D_r \mid D_t$
- $G = G_r \mid G_t$
- $S = S_r \mid S_t$
- $X = D \mid G \mid S$

We group the global illumination methods by the underlying mathematical model that they used for solving the rendering equation (2.7).

2.3.1 Inversion

A trivial solution is the inversion:

$$L = L^e + \mathcal{T}L \Rightarrow L = (1 - \mathcal{T})^{-1}L^e.$$

Since $\mathcal{T}$ cannot be inverted in a closed form, the solution is approximated by finite-element methods [Pop87]. The resulted system of linear equations is solved by the Gaussian method, which has cubic time complexity (according to the number of the finite-elements). The large number of surface elements made this approach unacceptable for real scenes but it was used in early radiosity algorithms. Therefore, this dissertation does not cover this approach.

2.3.2 Expansion

From mathematical point of view, these methods are based on the Neumann series expansion of the rendering equation, which is obtained by substituting L on the right side of the rendering equation (2.7) by the complete right side $L = L^e + \mathcal{T}L$. Considering that $\mathcal{T}$ is a contraction we repeat this process and get the Neumann series form of the rendering equation:

$$L = \sum_{i=0}^{\infty} \mathcal{T}^i L^e. \tag{2.15}$$

The measured power of the pixels are obtained as

$$C = \mathcal{M}L = \sum_{i=0}^{\infty} \mathcal{M}\mathcal{T}^i L^e.$$

The terms of this series are ever increasing high-dimensional integrals that are estimated by Monte-Carlo quadrature taking m random samples. Since Monte-Carlo methods have $\mathcal{O}(m^{-0.5})$ convergence independently of the dimension of the integration domain, they can avoid the exponential core of classical quadrature rules [SK99a].

Equation (2.15) creates the problem of calculating an infinite dimensional integral. Practical implementations usually truncate the infinite Neumann series, which introduces some bias, or stop the walks randomly, which reduces the samples of higher order inter-reflections.

These approaches are strongly view dependent. Since the radiance is estimated by random paths originated from the camera, if the camera changes, the complete calculation should be started from scratch.

However, expansion methods like path tracing have an important advantage. Namely, they do not require finite-element representations of the complete radiance function. Consequently, this approach can work with the original geometry without tessellating the surfaces to planar polygons.

2.3.3 Iteration

Iteration algorithms are based on the fact that the solution of the rendering equation is the fixed point of the following iteration scheme:

$$L(m) = L^e + \mathcal{T}L(m-1). \tag{2.16}$$

If this scheme is convergent, then the pixel colors can be obtained as a limiting value:

$$C = \mathcal{M}L = \lim_{m\to\infty} \mathcal{M}L(m).$$

Iteration converges with the speed of a geometric series, i.e. the error from the limiting value is in the order of $\mathcal{O}(a^m)$ where a is the contraction of integral operator $\mathcal{T}$. The

contraction is proportional to the average albedo of the surfaces and depends on how open the scene is. Note that iteration uses the estimate of the complete radiance function, thus it can potentially exploit coherence and reuse previous information. Since the complete radiance function is inserted into the iteration formula, parallelization is not as trivial as for random walks, and the error introduced in each step may accumulate to a large value [SK00]. To store the radiance estimates, finite-element approaches should be used.

2.3.4 Stochastic iteration

Stochastic iteration [SK99b] replaces the light-transport operator $\mathcal{T}$ by a random transport operator $\mathcal{T}^*$ that gives back the effect of the light-transport operator in the average case:

$$L(m) = L^e + \mathcal{T}^* L(m-1)$$

$$E[\mathcal{T}^* L] = \mathcal{T} L. \tag{2.17}$$

The pixel color is obtained as the average of the previous steps:

$$C = \mathcal{M} L = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \mathcal{M} L(i).$$

This method has a detailed description in section 2.6.

2.4 Random walk approach to the global illumination problem

Random walk approaches are based on random sampling to numerically estimate integrals. Since the expansion solution of the rendering equation requires the evaluation of very high dimensional integrals, and because the Monte-Carlo integration is very suitable for solving high dimensional problems, the Neumann series form (equation (2.15)) of the rendering equation can be successfully treated by random walk algorithms. This section presents a brief overview on Monte-Carlo methods and introduces the global illumination algorithms based on Monte-Carlo integration. The first Monte-Carlo random walk algorithm, the *distributed ray-tracing* was proposed by Cook et al. [CPC84]. The mathematical formulation of the rendering equation was published in 1986 by Kajiya [Kaj86]. Based on this formulation, he presented *path-tracing*, which is a variation of the more general *stochastic ray-tracing*. Ray-tracing techniques generate paths from the eye point. *Light tracing* [DLW93] follows light particle paths from the light sources. *Bi-directional path-tracing* [LW93] [VG95] generates an eye path and a light path and connects them to get the pixel estimate.

The weakness of the methods is that, in spite of many advancements, building a single path is computationally expensive.

2.4.1 Monte-Carlo methods

From the development of the nuclear bomb during World War II, the Monte-Carlo methods are used to solve wide variety of problems by simulating a suitable random process. They are based on probability theory.

Basic Monte-Carlo integration

The most common usage of Monte-Carlo methods is the computation of an integral

$$I = \int_{\Omega} f(x)dx,$$

where Ω is a possibly multi-dimensional domain. If $p(x)$ is a possible pdf (*probability density function*) on the domain Ω, and if we generate a sample x_0 according to this pdf, the *primary estimator* for integral I is given by

$$\langle I \rangle_{primary} = \frac{f(x_0)}{p(x_0)}.$$

In practice we usually take N independent samples to give the so-called *secondary estimator*

$$\langle I \rangle_{secondary} = \langle I \rangle_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}. \tag{2.18}$$

Beside estimator 2.18 other estimators $\langle I \rangle_N$ can be also defined for which the following estimator properties can be defined:

- *Error*: the error is given by $Error(\langle I \rangle_N) = I - \langle I \rangle_N$.
- *Unbiasedness*: the estimator $\langle I \rangle_N$ is unbiased if $\forall N : E[\langle I \rangle_N] = I$.
- *Bias*: bias is given by $[\langle I \rangle_N] = I - E[\langle I \rangle_N]$.
- *Consistency*: the estimator is consistent, if its bias vanishes in the limiting case

$$\lim_{N \to \infty} [\langle I \rangle_N] = 0.$$

It is straightforward that unbiasedness implies consistency. The secondary estimator 2.18 is an unbiased estimator. Unbiased (and also consistent) Monte-Carlo methods are useful, since they are provably converging to the good solution and for the unbiased estimators the variance can be estimated on the fly from the samples.

Analyzing the secondary estimator 2.18, it is quite straightforward to show that

$$D^2_{secondary} = \frac{D^2_{primary}}{N},$$

where D is the *standard deviation* (also called *RMS*[5] *error*). This equation is very important in Monte-Carlo methods. It shows that the RMS error (standard deviation) is inversely proportional to the square root of the number of samples of the secondary estimator. In practice it means – for example – that to halve the (RMS) error we must quadruple the number of samples. This is an unwanted property of the Monte-Carlo methods, therefore variance reduction strategies have utmost importance. The most common strategies for sample selection are *stratified sampling* and *importance sampling*.

Stratified sampling

Stratified sampling divides the integration domain Ω into M non-overlapping strata Ω_i and estimates each partial integral by N_i random samples:

$$I = \int_\Omega f(x)dx = \sum_{i=1}^{M} \int_{\Omega_i} f(x)dx = \sum_{i=1}^{M} \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{f(\xi_{ij})}{p(\xi_{ij})}. \tag{2.19}$$

The following properties of the stratified sampling can be proven:

- The variance of the estimate 2.19 is guaranteed to be lower than obtaining by distribution of random samples over the whole integration domain.
- Using one sample per strata has the most advantage, since it is easy to prove that it is always better to increase the number of strata than the number of samples per strata.

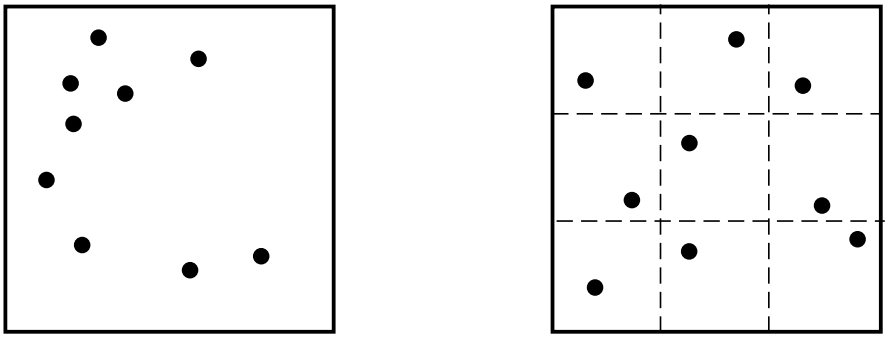


Figure 2.9: Regular and stratified pixel sampling

[5]Root Mean Square

The most frequently used example of stratified sampling is applied for pixel sampling. If the required number of samples is in the form of $N = k^2$, then the separation of the strata is straightforward (see figure 2.9).

In general, it is sometimes quite difficult divide the domain into N equal strata. The *Quasi-Monte Carlo* methods [Kel97b] provide an elegant solution for this problem. They do not produce samples randomly, but deterministically. They try to generate the samples as evenly and as distributed as possible.

Importance sampling

Importance sampling tries to select more samples in important region of the integrand. It needs prior knowledge of the function that we are going to estimate. We would choose samples where it is likely to return a high value. The best pdf is proportional to the integrand. Since any pdf must be non-negative and integrate to 1 over its domain, the optimal pdf is given by

$$p(x) = \frac{|f(x)|}{\int\limits_{\Omega} |f(x)|dx}.$$

For positive integrands this pdf results in a zero variance estimator, but unfortunately it is impossible to generate samples with this distribution, since the scaling factor contains the integral, which is not known.

In the rendering equation (2.6) the integrand is the product of two functions, one of which (the BRDF) is know a priori and can be used for importance sampling. Different material models use different *BRDF sampling* [Gla95] schemes. There are cases when sample generation according to the BRDF is not effective. E.g. when the light sources are very small, *light source sampling* [DW91] results a more advantageous algorithm.

2.4.2 Stochastic ray tracing

Stochastic ray tracing is based on the Neumann series 2.15 solution of the rendering equation. Theoretically, it can handle all the possible (LX^*E) paths, but unfortunately some types of light paths are sampled quite weakly. It is an image based algorithm, which recursively samples random directions $\vec{\omega}'_p, \vec{\omega}'_1, \vec{\omega}'_2, \ldots, \vec{\omega}'_n$ to follow the light paths backward and the emission of all visited points are gathered and transferred to the eye. Note that a single walk can be used to estimate the 1-bounce, the 2-bounce, etc. n-bounce transfer simultaneously.

If the light sources are small, then it is very unlikely to generate a direction that arrives at a light source and can contribute a non-zero value to the pixel estimate. To overcome the problem, *next event estimation* is applied, which samples the light sources and estimate the direct illumination of the visited point on each path vertex.

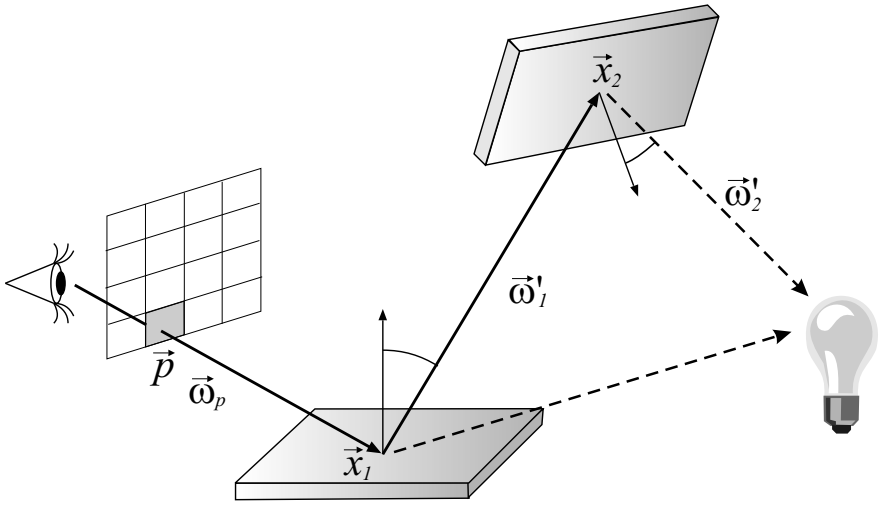


Figure 2.10: Path tracing

In the general case, stochastic ray tracing traces N scattered rays at each vertex to estimate the incoming radiance. A very important special case does not allow scattering, and uses only 1 ray sample for each vertex. This algorithm is called *path tracing* and it was presented by Kayija [Kaj86] as a generalization of distributed ray tracing. In his original paper he uses 40 ray samples per pixel to generate images.

Practical implementations usually truncate the infinite Neumann series, which introduces some bias, or stop the walks randomly, which reduces the samples of higher order inter-reflections. This method called Russian roulette [AK90] gives a still unbiased estimation. Russian roulette decides randomly (at hit point $\vec{x}_i$ with a probability s_i) whether or not the integrand is really evaluated for a domain point. If it is not evaluated, the integrand is assumed to be zero. If it is evaluated, the value is divided by the probability of this decision to compensate the cases when the integrand is not computed. It is straightforward that this random decision increases the variance of the estimator.

Since it starts from the eye and since it generates directions according to surface BRDF, the path tracing treats the specular or glossy effects quite efficiently. However, it is poor generating caustics.

2.4.3 Light tracing

Light tracing [DLW93] [DW94] [DW96] [Dut96] is a particle tracing technique (see figure 2.11). It is usually called the adjoint method of path tracing, since instead of generating rays from the eye through the pixel and progressing towards the light source, this technique generates rays from the light sources and proceeds towards the pixel.

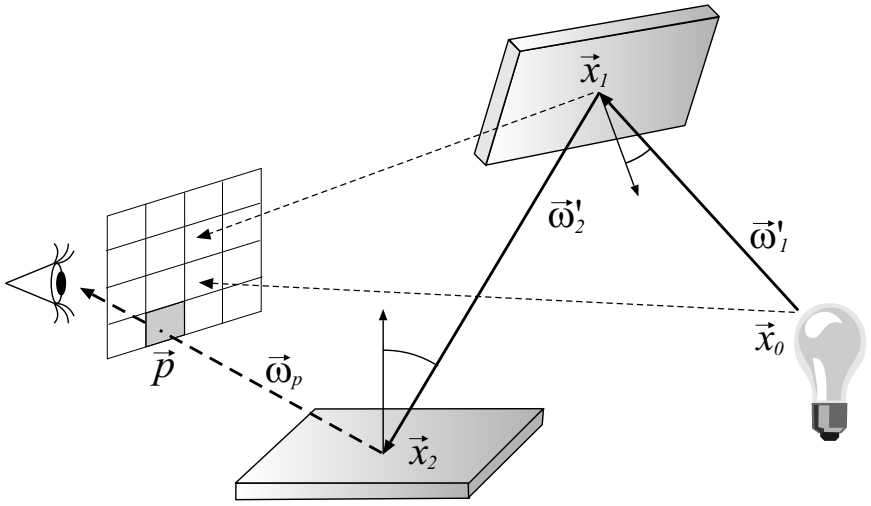


Figure 2.11: Light tracing

When the next direction is going to be generated, BRDF based importance sampling can be applied. This new direction determines (via ray casting) the intersection point on another surface. Then a ray is traced from the intersection point to the eye and light contribution is added to the affected pixel. If the camera is hidden from the intersection point or if the intersection with the rendering plane is outside of the rendering window, the contribution is zero which results high variance of the image estimate. Since light tracing (as any other shooting type global illumination algorithm) does not use BRDF sampling for the last step (which connects the path to the eye), it is very ineffective for generating mirror-like effects. However, caustics are handled very easily by light tracing.

2.4.4 Bidirectional path tracing

Since stochastic ray tracing is good for pinhole camera and mirror-like surfaces and light tracing is good for small light sources and caustics effects it would be a nice goal to combine them in a way that the advantages of the underlying algorithms are preserved.

Bidirectional ray tracing was proposed by Lafortune [LW93] and at the same time independently by Veach [VG94]. The bidirectional path $\bar{z}_{3,2} = \vec{x}_0\vec{x}_1\vec{x}_2\vec{y}_1\vec{y}_0$ is formed by connecting an eye and a light path. The notation of $\bar{z}_{st}$ means the eye sub-path contains s and the light sub-path contains t vertices. Using the path integral form of the rendering equation (2.10), the radiance contribution function $l(\bar{z})$ (equation (2.8)) and the pdf $p(\bar{z})$ of the bidirectional path must be determined. Since the direct connection of the sub-path is deterministic, it does not influence the pdf. The pdf of the path $\bar{z}$ that is presented in figure 2.12 is given by

$$p(\bar{z}) = p(\vec{y}_0)p(\vec{y}_1|\vec{y}_0) \cdot p(\vec{x}_0)p(\vec{x}_1|\vec{x}_0)p(\vec{x}_1|\vec{x}_1\vec{x}_0),$$

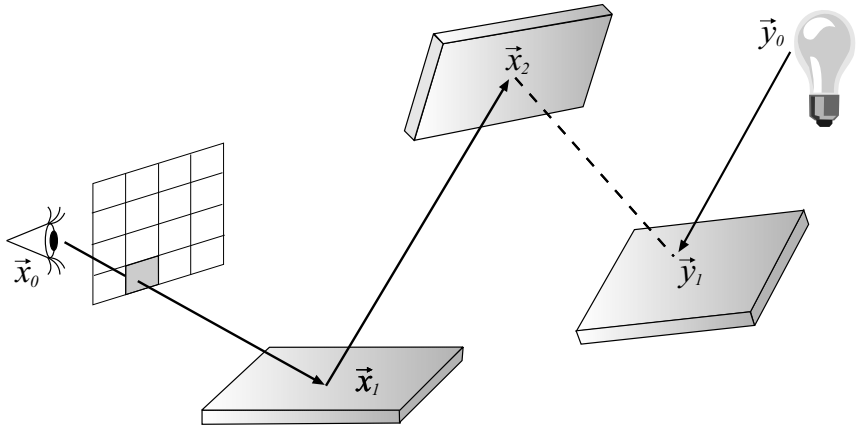


Figure 2.12: Bidirectional path is generated by connecting the eye and light sub-paths

where $\vec{y}_i$ is the i-th vertices of the light sub-path and $\vec{x}_i$ is the i-th vertices of the eye sub-path.

Note that the same path $\bar{z}$ can be generated by placing the deterministic connection in different edges of the path. In figure 2.13 the bidirectional paths $z_{3,0}$, $z_{2,1}$, $z_{1,2}$, $z_{0,1}$ define the same path and therefore have the same radiance contribution. Since a path of length n can be sampled $n+1$ different ways, care must be taken to avoid counting the same paths more than once [CW93]. Since all of them contribute to the final image, the contribution of them must be weighted. One possible solution corresponds to multiple importance sampling, as it was presented by Veach [Vea97].

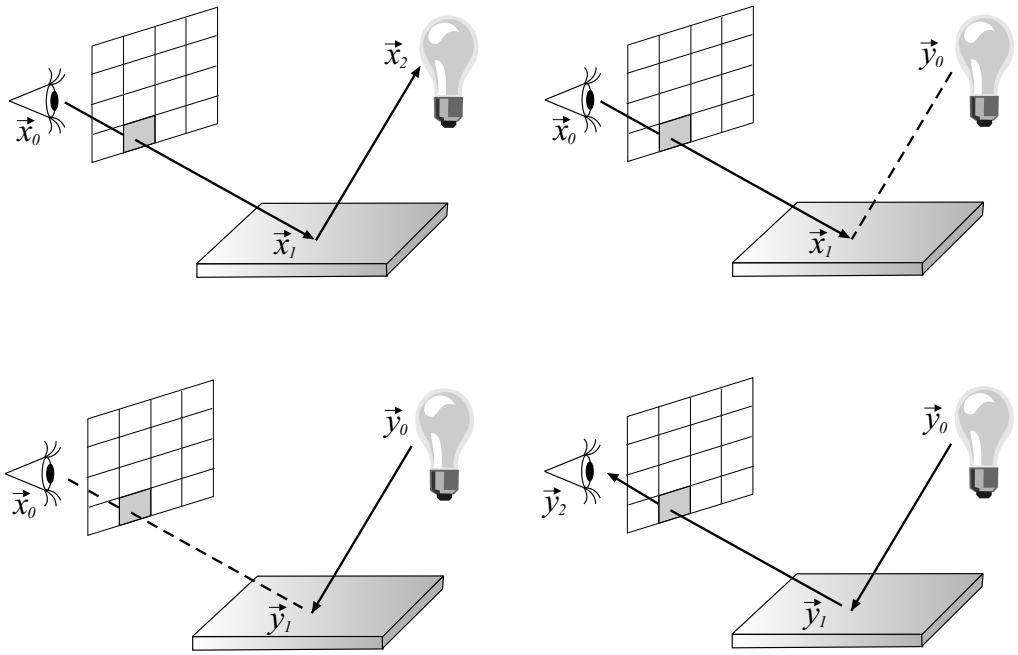


Figure 2.13: The same bidirectional path can be constructed in different ways

Observe the most important special cases for a bidirectional path $\bar{z}_{s,t}$:

- $s = 0$: The eye sub-path has no vertices at all. The light sub-path ended by chance on the aperture of the camera. For pinhole cameras this cannot happen.
- $s = 1$: The eye path has 1 vertex and the light path is connected deterministically to the camera. This case corresponds to light tracing.
- $t = 0$: The light path has no vertices at all. The eye path accidentally hits a light source surface.
- $t = 1$: The light path has 1 vertex, which was sampled by light source sampling. This case corresponds to path tracing with next event estimation.

Since bidirectional path tracing is a generalization of them, it is straightforward that the method is superior to both path tracing and light tracing.

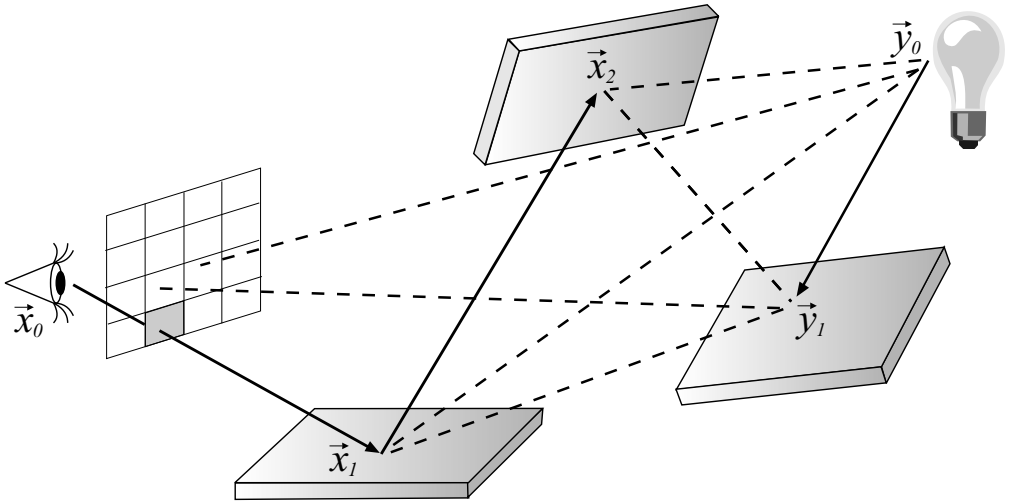


Figure 2.14: Bidirectional paths are generated by connecting the eye and light sub-paths in every possible way

Bidirectional path sampling generates one eye and one light sub-path randomly and connects them. The naive implementation of the sampler would generate the next bidirectional path by generating the sub-paths from the scratch. However, if a light sub-path and an eye sub-path is generated once, it is more effective to connect every vertex of the light sub-path to every vertex of the eye sub-path by shadow rays, and create all the possible bidirectional path. This construction is shown in figure 2.14. This efficient sampling was used both in Lafortune's and in Veach's methods.

2.4.5 Metropolis Monte-Carlo method

The Metropolis method was imported from physics [MRR+53] to computer graphics by Eric Veach in his SIGGRAPH'97 paper [VG97]. It is based on the path integral formulation of the rendering equation (see. equation (2.10)). This method generates stochastic samples which are distributed according to the desired distribution. In the global illumination case, the required probability is the importance function of the light intensities over the pixels:

$$p(\bar{z}) = \frac{I(\bar{z})}{b},$$

where b is the normalizing constant.

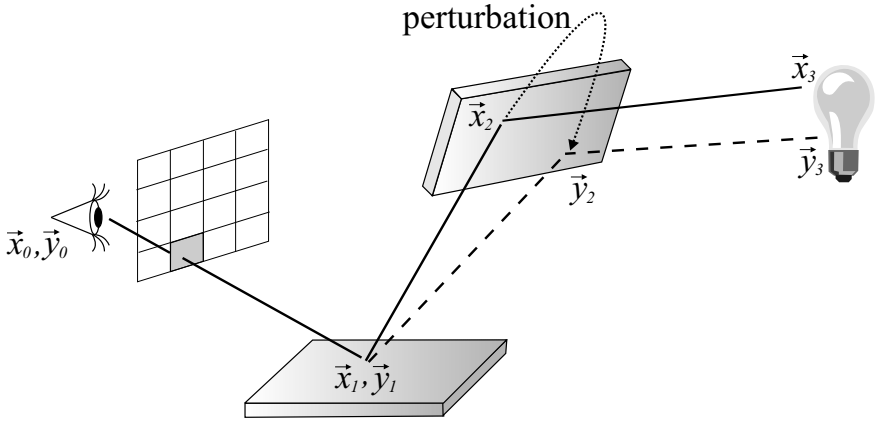


Figure 2.15: The mutation of path $\bar{x}$ to $\bar{y}$

However, instead of sampling light paths independently Metropolis method perturbs the previous light path $\bar{z}$, and creates a Markovian chain, in which each sample depends only on the previous one. It uses an almost arbitrary tentative transition function $T(\bar{z}_i \rightarrow \bar{z}_t)$ to generate a tentative sample $\bar{z}_t$ which is then accepted or rejected according to the acceptance probability $a(\bar{z}_i \rightarrow \bar{z}_t)$.

The generation of the Metropolis samples $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_n$, is given by

Find an initial sample $\bar{z}_1$
for $(i = 1; i < n; i{+}{+})$ {
    Based on $\bar{z}_i$, sample a tentative point $\bar{z}_t$ using $T(\bar{z}_i \rightarrow \bar{z}_t)$
    Generate random number $rand \in [0, 1]$
    if $rand < a(\bar{z}_i \rightarrow \bar{z}_t)$ // accept with probability $a(\bar{z}_i \rightarrow \bar{z}_t)$
        $\bar{z}_{i+1} = \bar{z}_t$
    else
        $\bar{z}_{i+1} = \bar{z}_i$
}

Theoretically, the Metropolis method achieves perfect importance sampling. However, some problems emerge which need to be treated carefully:

- The starting point of the Markovian chain must be selected with care.
- The generated samples are not independent, but correlated. If the correlation is too large, it produces annoying visible artifacts.
- If the mutation strategy makes too small perturbation, the method may violates the ergodicity condition, since it is possible that the it sticks some part of the domain and can never reach other important sub-set of the path domain.
- The mutation strategies are usually costly and difficult to implement.

Recently, Kelemen et al. [P14] proposed a solution for solving most of these problems based on mutation the samples not in the path space, but in the primary sample space.

2.4.6 Photon map

One of the most successful approach for solving the rendering equation is the *photon map* algorithm. Most of the commercial software packages, which support global illumination, are based on this method. Examples are Lightflow, LightWave, Luminaire, Maya, Twister and Mental Ray. The first paper on photon map was proposed in 1995 by Henrik Jensen [Jen95]. The idea behind this approach is that – unlike in bidirectional path tracing – light paths are not dropped when sampling a new eye path, but the light information is stored in a special data structure.

The photon map is the set of "virtual" photons[6] that are characterized by the energy on different wavelength, the position and the incoming direction of the photon. For calculating the outgoing direction faster, usually the surface normal at the hit-point is also stored. The photon hits are then stored in a kd-tree, which is balanced before using it for image rendering. The algorithm can be separated in three phases:

1. Photon shooting
2. Kd-tree balancing
3. Final gathering

The final gathering uses the following estimate of the rendering equation:

$$L(\vec{x},\vec{\omega}) = \int_{\Omega} L(rc(\vec{x},-\vec{\omega}'),\vec{\omega}') \cdot f_r(\vec{\omega}',\vec{x},\vec{\omega}) \cdot \cos\theta' \; d\vec{\omega}' =$$

$$\int_{\Omega} \frac{d\Phi(\vec{\omega}')}{dA\cos\theta' d\omega'} \cdot f_r(\vec{\omega}',\vec{x},\vec{\omega}) \cdot \cos\theta' \; d\omega' \approx \sum_{i=1}^{n} \frac{\Delta\Phi(\vec{\omega}_i')}{\Delta A} \cdot f_r(\vec{\omega}_i',\vec{x},\vec{\omega}), \qquad (2.20)$$

[6] real photons convey energy only on a single wavelength

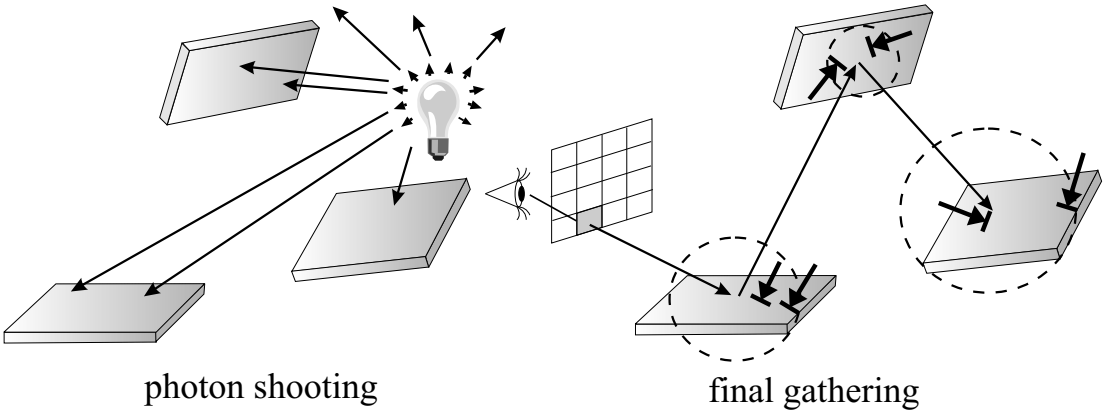


Figure 2.16: The first and the third steps of the photon map method

where $\Delta\Phi(\vec{\omega}_i')$ is the energy of the photon arriving to surface ΔA from incoming direction $\vec{\omega}_i'$. Value n is the parameter of the algorithm, which is set a–priori. Estimate 2.20 uses the contribution of n photons around the point $\vec{x}$. Practical implementations use 30–200 photons. The selection of the photons can be carried out by placing a sphere around the point $\vec{x}$ and enlarging the sphere until it contains exactly the required number of photons. If r denotes the radius of this sphere, then the ΔA is given by πr^2, since it is projected area of the sphere (a circle) to the surface. The previously presented basic algorithm can be extended and fine tuned further. Without the aim of completeness, we list some of the most important ideas:

1. The bounding sphere can be replaced by the bounding disc. The sphere is good basic candidate since the projected area is easy to calculate. However, since photon hits are not bind to surfaces, it occurs often (at the edges or in corners) that the estimate consists photons, which are placed on another surface. (e.g. the color of the wall leaks dawn to the floor). If we compress the sphere in the direction of the surface normal, we would use less "false" photons.

2. The caustics usually need more dense distribution of photons then it is presented in the *global photon map*. The caustics can be handled with a separate *caustics photon map*, which contains photons hits that hit at least one specular surface before hitting a diffuse surface. Using Heckbert notation these are LS^+D paths. Note that the photon shooting for caustics maps can be made faster, since we need to shoot photons only in the direction of specular surfaces.

3. By introducing the *volume photon map*, the method is capable to handle participating media [JC98].

2.4.7 Instant radiosity

Instant radiosity [Kel97b] is also a photon shooting technique, which is quite similar to photon map method. The shot photons considered as virtual point light sources. The contribution of point light sources then can be calculated by fast and hardware supported visibility and shadow algorithms.

Instant radiosity uses just small number of photons in the final gathering step. Usually about 40 point lights can be accounted for, which is rendered in e.g. 5 passes (if the maximal 8 light sources per pass of the OpenGL API are used). The used point light sources are selected from the virtual light sources by quasi-Monte Carlo technique. However, the dominant light sources must be selected with care, since if they are too close to a visible object (e.g. to a wall), the neighbor areas receive too much illumination, which causes highlighting artifacts.

The rendered images contribute to the accumulation buffer of the graphics hardware, which – at the end of the passes – contains the final image. Recently this method was combined with a fast, coherent ray tracer [IWS02], with which the authors report interactive global illumination rendering on the local area network of 20 computers.

2.5 Iteration methods for the global illumination problem

This section surveys the different radiosity algorithms, which originally was leaked to computer graphics from the field of heat transfer. They were the first solutions to the global illumination problem. These algorithms are all finite-element methods and based on deterministic iteration.

2.5.1 Classical radiosity

The classical radiosity was introduced by Goral et al. [GTG84] in 1984. The method subdivides the surfaces to small elemental *patch*es and supposes that the patches are small enough that the light intensity into a specific direction can be approximated by a constant value (the positional dependence of the radiance is eliminated). If only diffuse surfaces are allowed, the direction dependence can be also eliminated, since diffuse surfaces generate the same radiance in all direction. The corollary is that the method can handle only LD^*E type of light paths. The closed environment is also an assumption, in which the global illumination is simulated by the energy exchange of the patches.

The name of the algorithm refers to the radiometric quantity *radiosity*, which is the density of radiant energy flowing through a unit area per unit time (equation (2.2)). It includes the emitted energy and the energy reflected from other surfaces.

Using this measure, the rendering equation for a point $\vec{x}$ (equation (2.5)) takes the following form:

$$B(\vec{x}) = E(\vec{x}) + a_r(\vec{x}) \int_S B(\vec{y}) \frac{\cos\theta_{\vec{x}} \cos\theta_{\vec{y}}}{\pi r^2} V(\vec{x}, \vec{y}) dy, \tag{2.21}$$

where $B(\vec{x})$ is the radiosity and $E(\vec{x})$ is the emission at point $\vec{x}$, a_r is the diffuse albedo (or reflectance) of the surface, $\theta_{\vec{x}}$ is the angle between the surface normal at point $\vec{x}$ and the line that connects point $\vec{x}$ and $\vec{y}$, r is the length of the connecting line and $V(\vec{x}, \vec{y})$ is the visibility function (it is 1 if the two points see each other, and 0 otherwise).

In general, the finite-element method approximates the radiosity and the emittance functions in a function series form

$$B(\vec{x}) \approx \bar{B}(\vec{x}) = \sum_{i=1}^{n} B_i \cdot b_i(\vec{x}), \tag{2.22}$$

$$E(\vec{x}) \approx \bar{E}(\vec{x}) = \sum_{i=1}^{n} E_i \cdot b_i(\vec{x}),$$

where functions $b_i(\vec{x})$ are pre-defined basis functions and parameters B_i and E_i are scalar values. The classical radiosity uses only constant functions for the bases.

$$b_i(\vec{x}) = \begin{cases} 1, & \text{if point } \vec{x} \text{ is placed on surface } i \\ 0, & \text{otherwise} \end{cases}$$

Using the function series approximation, the radiosity equation (2.21) for points can be simplified for patches. This results a set of linear equations:

$$B_i = E_i + a_{r_i} \sum_{j=1}^{n} B_j F_{ij}, \qquad 1 \leq i \leq n \tag{2.23}$$

where F_{ij} is the geometric factor called *form factor* (see section 2.5.2) between patch i and patch j.

2.5.2 The form factor

The *form factor* F_{ij} describes the fraction of energy which leaves surface patch i and arrives at a surface patch j. It is a purely geometric relationship, independent of the viewpoint or the material properties of the surface. The form factor (see in figure 2.17) between differential areas dx and dy is given by:

$$F_{dxdy} = \frac{\cos\theta_{\vec{x}} \cos\theta_{\vec{y}}}{\pi r^2} dy.$$

The point-to-patch form is generated by integrating over surface area A_j:

$$F_{dxj} = \int_{A_j} \frac{\cos\theta_{\vec{x}} \cos\theta_{\vec{y}}}{\pi r^2} dy.$$

If patch A_j covers the whole hemisphere around point $\vec{x}$, the point-to-patch form factor is 1. This is what we expect, since the whole energy leaving dx arrives to surface A_j.

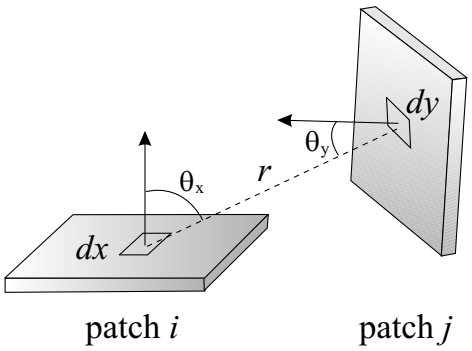


Figure 2.17: Form factor calculation

The patch-to-patch form factor that is the fraction of energy which leaves the whole surface patch i and arrives at a surface patch j is calculated by

$$F_{ij} = \frac{\int_{A_i} F_{dxj} dx}{\int_{A_i} dx} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_{\vec{x}} \cos\theta_{\vec{y}}}{\pi r^2} dy dx. \tag{2.24}$$

Instead of this estimate, the point-to-patch form factor can be used ($F_{ij} \approx F_{dxj}$) if the assumption is made that the single point $\vec{x}$ is a representative of all of the points on the surface.

Since the number of form factors is large (n^2, where n is the number of patches in the scene), the calculation of them is crucial in every radiosity algorithm. Over the years many techniques were developed to compute the form factor integral. The analytical solution in case of no occlusion is possible by applying Stokes theorem, but it is very inefficient. Other techniques uses hemisphere or hemicube projections or ray tracing approaches. Note that although the radiosity algorithm is a deterministic finite-element method, it can use Monte-Carlo methods to obtain the form factors [Bek99]. An elegant way uses integral geometry for casting a set of random global lines and estimating the form factor between two patches by the number of common line intersections as in [Sbe96]. A good overview of the form factor computation can be found in [CW93].

2.5.3 Solving the system

Two types of radiosity algorithms are exist.

The **full matrix radiosity** algorithms require the whole form factor matrix in each iteration step. It solves the following series of simultaneous linear equations:

$$\begin{bmatrix} 1-a_{r_1}F_{11} & -a_{r_1}F_{12} & \cdots & -a_{r_1}F_{1n} \\ -a_{r_2}F_{21} & 1-a_{r_2}F_{22} & \cdots & -a_{r_2}F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{r_n}F_{n1} & -a_{r_n}F_{n2} & \cdots & 1-a_{r_n}F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}.$$

The system of linear equations is solved by Jacobi or Gauss-Seidel [GTGB84] iteration . This approach requires storing the whole form factor matrix. In case of a moderately complex scene which contains for example one hundred thousand patches, the number of form factors reaches 10^{10}, which roughly needs 37 terabyte memory capacity[7]. Therefore this approach is suitable only for simple scenes.

In **progressive radiosity** (or Southwell iteration) each iteration requires the calculation of form factors between a point on a single surface and all other surfaces. Therefore just a row of form factor matrix needs to be stored in each step. Other advantage of this method that, without the costly form factor matrix computation, it will produce intermediate results, each of them is more accurate than the last. This intermediate result can be presented to the user and the algorithm can be halted if the desired approximation is reached. The overview of the algorithm is given by:

```
do (iteration) {
    Select a surface i according to the emitted radiosity
    Calculate F_ij for all surfaces j
    for (each surface j) {
        Shoot energy of surface i
        Update radiosity B_j
        Update emission of surface j
    }
}
```

2.5.4 Extensions and notes

The **hierarchical radiosity** [CCWG88] [HSA91] [CSSD96] [BNN+98] is one of the most efficient approach for radiosity calculations especially if it is combined with clustering [SDS95] [CLSS97]. The method tries to further decrease the number of form factor calculations by creating a multilevel hierarchy of surface patches. The fundamental idea is that the many interaction between a single patch and a set of

[7] if 4 byte float is used for each form factor

patches can be simulated by one interaction between a single patch and a large patch which averages the radiosity of the patches in the set. It can be carried out if a single patch is far away from a set of patches (e.g. it has small form factors with the patches in the set).

If a radiosity system uses other than the constant basis functions (see equation (2.22)) used by the classical radiosity algorithm, we refer it as **higher-order radiosity**. It is also often named as Galerkin radiosity [Hec91] [TM93] [Zat93]. The advantage of higher-order basis is that it allows using much coarser meshes compared to classical radiosity, while still maintaining the same error bound. In classical radiosity, the surfaces must be subdivided small enough to represent the same variation. One problem with higher-order methods is that they do not easily support discontinuities like sharp shadows, since the basis functions are usually continuous. In radiosity algorithm the basis functions $b_i(x)$ usually depend only on the position. However, the method can be extended if positional and **directional basis functions** are also considered which eliminates the restriction of diffuse surfaces, and the algorithm can handle specular effects. The positional basis functions may be either constant (as it is the case in classical radiosity) or linear on a patch, while the directional basis functions can also be piece-wise constant [ICG86], spherical harmonics [SAWG91] or Haar functions [SSG+99].

Wavelet radiosity was introduced by Gotler et al. [Gor93] and Schroder et al. [SGCH93], which combines the advantages of Galerkin radiosity and hierarchical radiosity.

Other types of radiosity extensions [Sil95] handles **participating media** – such as smoke, fog, or water vapor in the air – by considering the energy exchange between volumes, and defining a spatial hierarchy in 3D space.

The **advantage** of radiosity algorithms is that the radiosity estimates are completely view-independent, thus when the radiosity approximation is available, the image can be obtained from any viewpoint.

One **disadvantage** of radiosity methods is that it needs to divide the polygonal environments into small patches. The meshing is crucial in places where discontinuities of the light distribution occur. Such places are, for example the edges of shadows. Soft shadows can only be generated by radiosity algorithm if some kind of discontinuity meshing [Hec92] or adaptive subdivision is also used before or during the radiosity calculation. Other disadvantages are the large computational and storage cost, and the lack of simulating the non-diffuse component of light distribution. If the number of basis functions is less than necessary, light-leaks may occur and shadows and highlights may be placed incorrectly [Slu97]. Although, hierarchical or multiresolution methods and clustering can help, the memory requirements are still prohibitive for complex scenes. We will give partial solutions for these problems in the following section.

2.6 Stochastic iteration for the global illumination problem

Until now, we have surveyed the random walk and the iteration solutions for the global illumination problem. Note that a single iteration step requires much more computation than a single random light-path, but the $\mathcal{O}(a^m)$ convergence of iteration still seems to be far superior to the $\mathcal{O}(m^{-0.5})$ convergence of random walks. However, random walk converges to the real solution while iteration to the solution of the finite-element approximation of the original problem. In this section we present a third alternative for solving the global illumination equation.

Suppose that we stick to the finite-element approach. Since we are interested also in directional light distribution – unlike in classical radiosity algorithm – we must calculate the radiance – and not the radiosity – for each point and for each direction. Due to the fact that the radiance is a four variate function and it changes quickly, an accurate finite-element representation requires very many basis functions, which makes these algorithms both storage and time consuming.

Thus the radiance of each surface patch is described by a directional function, which can be obtained as the average of the directional radiance. If deterministic iteration was used, this directional radiance function should be approximated by many directional basis functions, (with constant, wavelet or spherical harmonics bases) which could easily lead to huge memory requirements.

In order to reduce the astronomical storage requirements of the directional radiosity, the iteration is randomized, which leads to *stochastic iteration* [SK99b]. Unlike classical iteration where all patches should be selected to gather the radiosity or to shoot their unshot radiosity, stochastic iteration can exploit that a randomly selected patch may represent its neighbours as well, thus accurate results can be obtained even if just a fraction of patches are selected at all. Since stochastic iteration requires just a random approximation of the patch radiance, it can use just a single variable per patch even if the general, non-diffuse problem is attacked.

Suppose that we have a random linear operator $\mathcal{T}^*$ (see equation (2.7)) so that

$$E[\mathcal{T}^* L] = \mathcal{T} L \tag{2.25}$$

for any integrable function L. During stochastic iteration a random sequence of operators $\mathcal{T}_1^*, \mathcal{T}_2^*, \ldots \mathcal{T}_i^* \ldots$ is generated, which are instantiations of $\mathcal{T}^*$, and this sequence is used in the iteration:

$$L(m) = L^e + \mathcal{T}_m^* L(m-1). \tag{2.26}$$

Suppose that at a given point of the iteration, an image estimate is computed from the actual radiance, that is, the measured value (see equation (2.14)) in each pixel is

$$C(m) = \mathcal{M} L(m). \tag{2.27}$$

This measured value will also be a random variable which does not converge but fluctuates around the real solution. However, this does not pose problem, since if the image estimates are computed after each iteration step, the final result can be obtained as an average of these estimates. Averaging the first m steps, we obtain:

$$\hat{C}(m) = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathcal{M}L(i) = \frac{1}{m} \cdot \mathcal{M}L(m) + \left(1 - \frac{1}{m}\right) \cdot \hat{C}(m-1).$$

If $\mathcal{T}^*$ is properly constructed, then it does not need the radiance function everywhere on its domain (just e.g. in a specific direction), which helps reducing the astronomical storage requirements of directional dependent finite-elements.

The conclusion is that stochastic iteration is theoretically an effective tool that can reduce the complexity of global illumination algorithms and can result in non-diffuse global illumination methods that require just a single variable per patch, that is, their memory requirement is the same as that of the diffuse radiosity case. The critical part of the construction of such an algorithm is to find an appropriate random transport operator, that can be efficiently computed, thus the computation time of a single step is small. On the other hand, the variance introduced by the randomization should be also small, to keep the number of required iteration steps acceptable.

There are two useful stochastic iteration algorithm published so far. *Parallel ray-bundle tracing* [SK00] transfers the radiance of all points parallel to a randomly selected global direction, while *single ray based shooting* [SK99b] – which is a continuous approach – transfers the radiance of a randomly selected point at a randomly selected direction. Both algorithms will be described in detail in chapter 4.

2.7 Summary

In this chapter we have introduced the most important radiometric definitions. Based on this notation the rendering equation – as a Fredholm type integral equation of the second kind – was developed. Later, the rendering problem was expressed as a quadruple, and its components (geometry, materials, light sources, measurement devices) were detailed. That was followed by a study considering the possible solution strategies for the global illumination equation. Many global illumination algorithms were also described. They can be differentiated into one of following three categories: random walk methods, iteration methods and stochastic iteration methods. Each approach has its advantages and disadvantages, which was also pointed out in this section.

Chapter 3

Analysis of Russian roulette

Russian roulette is a popular technique that ensures the unbiasedness of the Monte-Carlo integration in case of infinite dimensional integrals. This chapter examines the optimal selection of the termination probability in problems where the allowable computational time, i.e. the number of rays to be traced, is constant. The results of this chapter is the own work of the author and summarized as thesis 1 (see chapter 8).

3.1 Introduction

Random-walk algorithms (see section 2.4) evaluate the integrals in the rendering equation by Monte-Carlo quadrature. Monte-Carlo integration is justified by the fact that its complexity does not grow with the dimension of the domain, thus it can avoid the dimensional core of the classical quadrature rules. These algorithms are based on the Neumann series form of the rendering equation:

$$L = \sum_{i=0}^{\infty} \mathcal{T}^i L^e = L^e + \mathcal{T}(L^e + \mathcal{T}(L^e + \ldots)\ldots) = L^e + \mathcal{T}L^{in} = L^e + \mathcal{T}(L^e + L^{ind}),$$

where L^{in} is the incoming radiance that can be separated to L^e emittance and to L^{ind} radiance due to the indirect illumination. By restructuring the rendering equation (equation (2.6)), the integral that needs to be calculated is given by

$$\mathcal{T}L^{in} = \int_{\Omega} L^{in}(\vec{x}, \vec{\omega}') \cdot f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta' d\omega'. \tag{3.1}$$

For the sake of clarity, from now on we drop the notation of the function parameters (e.g. $\vec{\omega}$, $\vec{\omega}'$ and $\vec{x}$) in cases where it is straightforward. If the integral 3.1 is evaluated by Monte-Carlo quadrature, then it is converted to an expected value, which is estimated by an average:

$$\int_{\Omega} L^{in}(\vec{\omega}') \cdot f_r \cdot \cos\theta' \; d\omega' = \int_{\Omega} \frac{L^{in}(\vec{\omega}') \cdot f_r \cdot \cos\theta'}{p(\vec{\omega}')} \cdot p(\vec{\omega}') \; d\omega' = E\left[\frac{L^{in}(\vec{\omega}') \cdot f_r \cdot \cos\theta'}{p(\vec{\omega}')}\right],$$

where random direction $\vec{\omega}'$ is sampled from a probability density $p(\vec{\omega}')$.

Equation (3.1) requires the evaluation of infinite-dimensional integrals. One way of attacking the problem is **truncating the Neumann series**, but this introduces some bias. Note that

$$\mathcal{T}L^e(\vec{x},\vec{\omega}) \leq L^e_{\max} \cdot \mathcal{T}1 = L^e_{\max} \cdot a(\vec{x},\vec{\omega}), \tag{3.2}$$

where $a(\vec{x},\vec{\omega})$ is the albedo of the surface. Note also that

$$\mathcal{T}a \leq a_{max}\mathcal{T}1 \leq a^2_{max}$$

and similarly

$$\mathcal{T}^2 a \leq \mathcal{T}a^2_{max} \leq a^2_{max}\mathcal{T}1 \leq a^3_{max}. \tag{3.3}$$

It is straightforward to continue this process to estimate the upper bound for every $\mathcal{T}^i a$, where $i > 2$. If the truncation happens at step m ($m \geq 2$), then the truncation error can be upper bounded (by using inequality 3.2 and 3.3 and the sum of the geometric series) as follows:

$$\sum_{i=m}^{\infty} \mathcal{T}^i L^e \leq L^e_{\max} \cdot \sum_{i=m}^{\infty} \mathcal{T}^{i-1} a(\vec{x},\vec{\omega}) \leq$$

$$L^e_{\max} \cdot \sum_{i=m}^{\infty} a^i_{\max} = L^e_{\max} \cdot a^m_{\max} \cdot \sum_{i=0}^{\infty} a^i_{\max} = \frac{L^e_{\max} \cdot a^m_{\max}}{1 - a_{\max}}. \tag{3.4}$$

Therefore if the truncation depth m increases, the error can decrease with the factor $a^m_{\max}$. At first sight this looks promising, since for physically plausible materials the albedo is less than 1, and therefore increasing the truncation depth the error vanishes quickly. However, there exist physically valid scenes, where $a_{\max}$ is very close to 1 (trivial example is a room with 6 perfect mirror sides). In this case the truncation results in a significant error.

Fortunately, there is another approach that solves the infinite-dimensional integration problem through randomization. In the context of Monte-Carlo integration, this approach is called the Russian roulette [AK90]. Russian roulette decides randomly whether or not the integrand is really evaluated for a domain point. If it is not evaluated, then the integrand is assumed to be zero. If it is evaluated, the value is divided by the probability of this decision to compensate the cases when the integrand is not evaluated. Since this random decision is made for each integral in a recursive manner, the probability of really computing a sample of a higher order bounce goes to zero. This is good, because this reduces the computational efforts. On the other hand, this is bad, since the random decision increases the variance of the estimator.

3.2 Previous work

The continuation or the survival probability of the random walk is the probability that the walk is followed after bouncing at a surface. The termination probability is the likeliness of stopping the walk. A possible value for the survival probability of the Russian roulette is the local albedo. This is usually justified by the following way. Suppose that the probability density $p(\vec{\omega}')$ is proportional to the cosine weighted BRDF $f_r \cdot \cos\theta'$, which is called *BRDF sampling*. Since it must be guaranteed that $p(\vec{\omega}')$ integrates to 1, the proportionality ratio leads the probability density as the function of the albedo:

$$p(\vec{\omega}') = \frac{f_r \cdot \cos\theta'}{\int\limits_{\Omega} f_r \cdot \cos\theta' \; d\omega'} = \frac{f_r \cdot \cos\theta'}{a}.$$

In this case the expected value of equation (3.1) is computed for the following random variable

$$\frac{L^{in}(\vec{\omega}') \cdot f_r \cdot \cos\theta'}{p(\vec{\omega}')} = L^{in}(\vec{\omega}') \cdot a.$$

If Russian roulette is applied, then the estimate is divided by the continuation probability s, thus the new random variable is

$$L^{in}(\vec{\omega}') \cdot \frac{a}{s}.$$

Setting s to the albedo, only the incoming illumination remains in the formula, thus all transported rays have the same weight. This seems reasonable according to importance sampling. However, this is not exactly true. On the one hand, Russian roulette also inserts 0 values in the quadrature when it decides not to compute the integrand. Thus if the original integrand is large, this adds a lot of noise. On the other hand, this approach uses just the local albedo and ignores both the albedo at other points and the incoming illumination. Consider, for example, the following case: the scene has a single, planar light source surface that has high albedo, while all other surfaces are black. When the ray hits a light source during a gathering walk, the walk is continued almost surely according to the local albedo. However, the expected incoming illumination is zero since all other surfaces have 0 albedo, thus the optimal continuation probability would be zero.

The special case of the Russian roulette is when the survival probability equals to 1. This case is often called the *infinite path length estimator*. Assuring the termination of the algorithm, a particular depth is chosen which bounds the length of the walk. However, this introduces some bias in the estimator. In [SA97] it was proven that the best finite path length estimator was better than the biased infinite path length one.

Optimal absorption probabilities have been examined also in [Sbe00] where a condition for the existence of the variance was given. Another result of paper [Sbe00]

is that it examined the shooting type radiosity and optimal probabilities were found for the case when the random walk proceeds according to the discrete Form Factor probability transitions.

In this chapter we focus only on the gathering type random walk. As stated, the survival (or not absorption) probability of the ray on a surface is usually set equal to the albedo of the surface. Exceptions to this were given in [NNB+96] where the received importance was used instead of the albedo, and in [SK99b] where the survival probability is some kind of average of the albedo.

3.3 Variance analysis of the gathering random walk

This section focuses on giving an exact formula for the variance of the Russian roulette random walk estimator. In the first step, the estimates of the integral representing a single bounce are examined, and then the results are extended for multiple bounces.

3.3.1 Variance formulae for single bounce

Let us consider the case when the ray is followed until a single bounce:

$$L^{(1)} = \int_{\Omega} L^{in} f_r \cos\theta' d\omega' = \int_{\Omega} L^{in} w_1 d\omega', \tag{3.5}$$

where $w_1 = f_r \cos\theta'$.

This integral is approximated usually by a Monte-Carlo expansion with Russian roulette:

$$\int L^{in} w_1 d\omega' = E\left[L^{in}\frac{w_1}{p_1}\right] = E\left[L^{in}\frac{w_1}{p_1}\frac{\Theta_1}{s_1}\right], \tag{3.6}$$

where Θ_1 is a random variable which takes 1 value with probability s_1 (we continue the walk) and takes 0 with probability $1 - s_1$ (we stop the walk).

Denote the random variable, which appears in (3.6) without Russian roulette by α and with Russian roulette by α', namely $\alpha = L^{in}\frac{w_1}{p_1}$, $\alpha' = L^{in}\frac{w_1}{p_1}\frac{\Theta_1}{s_1}$.

The expected value of α' is equal to the expected value of α:

$$E[\alpha'] = E[\alpha'|continue]\cdot P(continue) + E[\alpha'|stop]\cdot P(stop)$$

$$= \frac{1}{s_1}E[\alpha]\cdot s_1 + 0\cdot(1-s_1) = E[\alpha]. \tag{3.7}$$

The variance of α' is:

$$D^2[\alpha'] = E[(\alpha' - E[\alpha'])^2] = E[\alpha'^2] - E^2[\alpha'] =$$

$$E[\alpha'^2|continue]\cdot s_1 + E[\alpha'^2|stop]\cdot(1-s_1) - E^2[\alpha] = \left(\frac{1}{s_1} - 1\right)E[\alpha^2] + D^2[\alpha].$$

This validates that the application of Russian roulette always increases the variance of the estimator by $(1/s_1 - 1)E[\alpha^2]$. If the survival probability s_1 is equal to 1, i.e. we always continue the walk, we get back the variance of the random variable without the Russian roulette. If it is not equal to 1, then the variance always increases.

If we restrict ourselves for maximum 2 bounces, the formulae can be calculated similarly:

$$L^{(2)} = \int_\Omega \int_\Omega L^{in} w_1 w_2 d\omega_1' d\omega_2'. \tag{3.8}$$

$L^{(2)}$ can be approximated by Monte-Carlo and by Russian roulette method:

$$\int_\Omega \int_\Omega L^{in} w_1 w_2 d\omega_1' d\omega_2' = E\left[L^{in}\frac{w_1}{p_1}\frac{w_2}{p_2}\right] = E\left[L^{in}\frac{w_1}{p_1}\frac{\Theta_1}{s_1}\frac{w_2}{p_2}\frac{\Theta_2}{s_2}\right], \tag{3.9}$$

where Θ_2 is 1 if we continue the walk at step 2 and 0 otherwise. Denote the random variable which appears in (3.9) without Russian roulette by α and with Russian roulette by α', namely $\alpha = L^{in}\frac{w_1}{p_1}\frac{w_2}{p_2}$, and $\alpha' = L^{in}\frac{w_1}{p_1}\frac{\Theta_1}{s_1}\frac{w_2}{p_2}\frac{\Theta_2}{s_2}$. The equality of the expected values of α and α' comes directly from the recursive application of equation (3.7).

The variance of (3.9) can be expressed in the following way:

$$D^2[\alpha'] = \left(\frac{1}{s_1 s_2} - 1\right) E[\alpha^2] + D^2[\alpha].$$

Similarly for n bounces:

$$D^2[\alpha'] = \left(\frac{1}{s_1 \cdot \ldots \cdot s_n} - 1\right) E[\alpha^2] + D^2[\alpha].$$

It holds that if all survival probabilities s_i are equal to 1, which means that the walk is always continued, we get back the variance of α which is the original estimator without Russian roulette. If they are not equal to 1, the variance increases.

3.3.2 Variance formulae for multiple bounces

Assume that we make a lot of walks. The total number of steps — i.e. the rays to be traced — denoted by N. In our first case, when we permit maximum two steps, N breaks down to N_1 and N_2 where N_1 is the number of the steps in the first bounce, N_2 in the second bounce. The running cost of the rendering algorithm is proportional to the number of rays, i.e. N. If we are restricted to a given computation time, that is, we can do only a limited number of steps, we want to find out the best value of the Russian roulette probability s where the variance gets to its minimum. More formally, if the walk is terminated after the first step by probability s, then:

$$N_1 + N_2 = N\ , \quad N_2 = N_1 s$$

Thus N_1 and N_2 are:

$$N_1 = \frac{N}{1+s}\,, \quad N_2 = \frac{sN}{1+s}. \tag{3.10}$$

In gathering type random walk algorithms, allowing maximum two bounces, the radiance can be calculated by:

$$L^r = \int w_1 L^e d\omega_1' + \int\int w_1 w_2 L^e d\omega_1' d\omega_2'. \tag{3.11}$$

In the first term we do not use Russian roulette, just in the second integral. Let us denote w_1/p_1 by a. Calculating the variance of (3.11), we assume that the two parts are independent random variables. It must be stated that it is just for the sake of simplicity, and it is not definitely true for the random walk. It helps us to express the variance of (3.11) by the variances of the parts. Thus the variance after performing N overall steps can be written in the following way:

$$D_s^2 = \frac{D^2[\alpha]}{N_1} + \frac{a^2((\frac{1}{s}-1)(E[\alpha^2]+D^2[\alpha])}{N_1} \tag{3.12}$$

Here, we search for the appropriate s, for which D_s^2 (as a function of s) is minimal.

Substituting equation (3.10) into (3.12) we obtain:

$$D_s^2 = \frac{(1+s)D^2[\alpha]}{N} + \frac{(1+s)a^2(\frac{1}{s}-1)(E[\alpha^2]+D^2[\alpha])}{N}$$

The optimum of the formula is the point where the first order derivative vanishes:

$$\frac{dD_s^2}{ds} = 0 \quad \Longrightarrow \quad s^2 = \frac{a^2 E[\alpha^2]}{D^2[\alpha] + a^2 D^2[\alpha] - a^2 E[\alpha^2]}.$$

To generalize this, we first examine the 2-bounce case. The appropriate numbers of steps N_1, N_2 and N_3 are

$$N_1 = \frac{N}{1+s_1+s_1s_2}, \quad N_2 = \frac{s_1 N}{1+s_1+s_1s_2}, \quad N_3 = \frac{s_1 s_2 N}{1+s_1+s_1s_2}. \tag{3.13}$$

For the sake of simplicity, all s_i probabilities and all a_i albedos are assumed to be same. Thus the variance can be written as:

$$\begin{aligned} N \cdot D_s^2 &= s^2(D^2[\alpha](1+a^2+a^4) - E[\alpha^2](a^2+a^4)) \\ &+ s(D^2[\alpha](1+a^2+a^4) - E[\alpha^2](a^4)) \\ &+ D^2[\alpha](1+a^2+a^4) \\ &+ \frac{1}{s}(E[\alpha^2](a^2+a^4)) \\ &+ \frac{1}{s^2}E[\alpha^2](a^4). \end{aligned} \tag{3.14}$$

This formula can be generalized for arbitrary natural number n. We calculate the reflected radiance of a patch until n bounces. Thus we use the biased form of the rendering equation. As in equation (3.11) the reflected radiance is calculated by:

$$L^r = \int w_1 L^e + \int\int w_1 w_2 L^e + \ldots + \int \ldots \int w_1 \ldots w_n L^e \tag{3.15}$$

Calculating this by Russian roulette and expressing the appropriate N_1 term similarly as in equation (3.13), the variance of the formula can be obtained as a generalization of (3.14). After performing simplifications in notation, we get:

$$N \cdot D_s^2 = D^2[\alpha]\frac{1-a^{2n}}{1-a^2} + \sum_{i=1}^{n-1} s^i \left(\frac{1-a^{2n}}{1-a^2} D^2[\alpha] - \frac{a^{2(n-i)} - a^{2n}}{1-a^2} E[\alpha^2] \right) + \sum_{i=1}^{n-1} \frac{1}{s^{n-i}} \frac{a^{2(n-i)} - a^{2n}}{1-a^2} E[\alpha^2]. \tag{3.16}$$

Formula (3.16) breaks down into three parts: a constant, a hyperbolic (determined by $1/s^i$) and a parabolic part (determined by s^{n-i}). Considering this decomposition, two cases can be separated. In the first case the coefficients of the s^i are positive. In this case the hyperbolic curve is accelerated and the sum of the constant, hyperbolic and parabolic parts may have a valley shape. On the other hand, when the coefficients of s^i are negative (i.e. decreasing), the hyperbolic shape is warped to decrease faster.

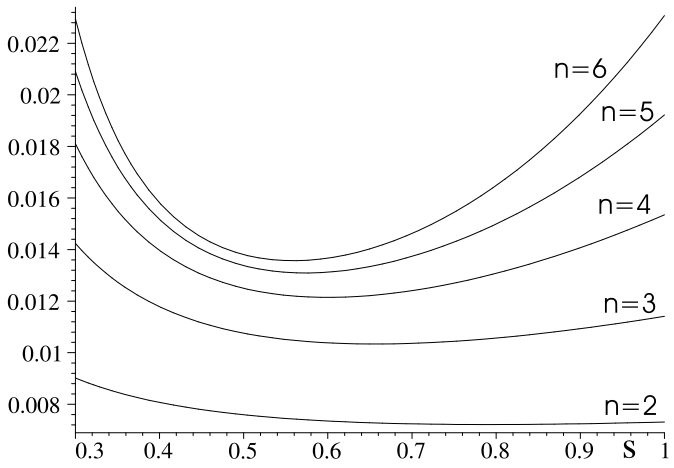


Figure 3.1: Behavior of the variance functions for the positive coefficients case

To show this, we plot these functions for the $n = 2, 3, 4, 5, 6$. In the first case when we want to achieve positive coefficients, we have chosen $E[\alpha^2] = 0.29$, $D^2[\alpha] = 0.137$, $a = 0.15\pi \approx 0.471$, $N = 1500000/(181 \cdot 181)$, which are approximately the values

with which the measurements took place (see section 3.4). The plots of the functions can be seen in figure 3.1.

On the other hand, as the coefficients of s^i go to negative, the functions undergo a dramatic change. To pose this, we have chosen $E[\alpha^2] = 0.29$, $D^2[\alpha] = 0.02$, $a = 0.15\pi \approx 0.471$, $N = 1500000/(181 \cdot 181)$. Note that compared to the previous case we have not changed anything but the value of the variance. The graphs of the variances for $n = 2, 3, 4, 5, 6$ are shown in figure 3.2.

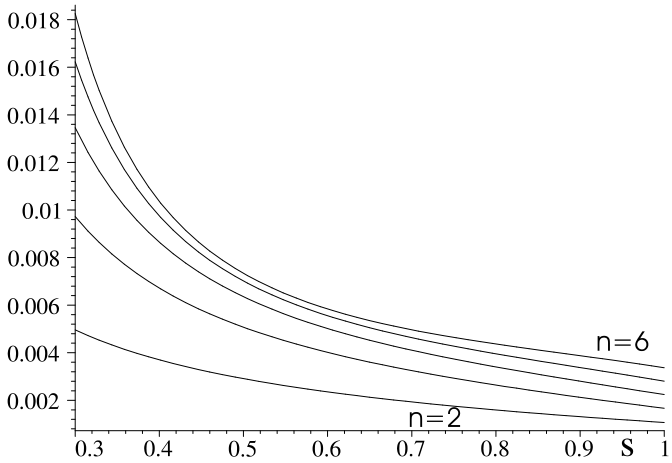


Figure 3.2: Behavior of the variance functions for the negative coefficients case

Supposing that the number of steps is limited to N, we want to find the optimal solution of the survival probability of the gathering random walk. The optimal value s can be determined by finding where the variance of the Russian roulette estimation is minimal. We have a constraint for choosing s, since it must fall into the interval $[0, 1]$.

Looking at equation (3.16) and at the figures 3.1 and 3.2, we find that if the coefficients are negative, then the best value of s equals to 1. If they are all positive, the optimal s is somewhere below 1. Let us consider for example the coefficient of the $n-1$ order term:

$$D^2[\alpha]\frac{1-a^{2n}}{1-a^2} - E[\alpha^2]\frac{a^2-a^{2n}}{1-a^2}. \tag{3.17}$$

Equation (3.17) could be negative if the variance of the original estimator is less than a specific value (which comes true in the case of a flat, homogeneous scene). If the coefficients of the polinom are negative, the optimal survival probability is 1. Considering the first order coefficient ($i = 1$)

$$D^2[\alpha]\frac{1-a^{2n}}{1-a^2} - E[\alpha^2](a^{2(n-1)}),$$

we can note that if this value is negative, all higher order coefficients get negative values, resulting the optimal s to be 1. This comes from the monotone increasing property of $(a^{2(n-i)} - a^{2n})/(1-a^2)$ as a function of i ($a \leq 1$).

Note that expression (3.17) can also be made negative by increasing the albedo a to a specific level.

In contrast to the previous case, the coefficients can be positive, which induces that the optimal value for the survival probability s could be less than 1. Looking at equation (3.16) and especially at (3.17), we can state that if expression (3.17) is positive, which means that

$$D^2[\alpha] > E[\alpha^2]\frac{a^2 - a^{2n}}{1 - a^{2n}},$$

then all positive order coefficients in formula (3.16) are positive. If the integrand of the rendering equation is very heterogeneous, the variance is large enough to fulfill this requirement.

If we consider the positivity of the coefficients, another property of equation (3.16) can be found. As the albedo increases, the location of the minimum point of the variance increases. This justifies the correctness of the widely used algorithms which choose the survival probability equal to the local albedo.

The relationship of the albedo and the optimal survival probability is also an important question. To see their correspondence, the

$$\frac{dD^2_{s,a}}{ds} = 0$$

implicit functions (for $n = 2, 3, 4, 5, 6, 7$) were plotted for various $D^2[\alpha]$ and $E[\alpha^2]$. In figure 3.3 $D^2[\alpha]$ was chosen to be 0.2 and $E[\alpha^2]$ to be 0.29. In this high variance case the optimal survival probability was found to be equal to the albedo. As n increases, the curve gets closer to a linear function.

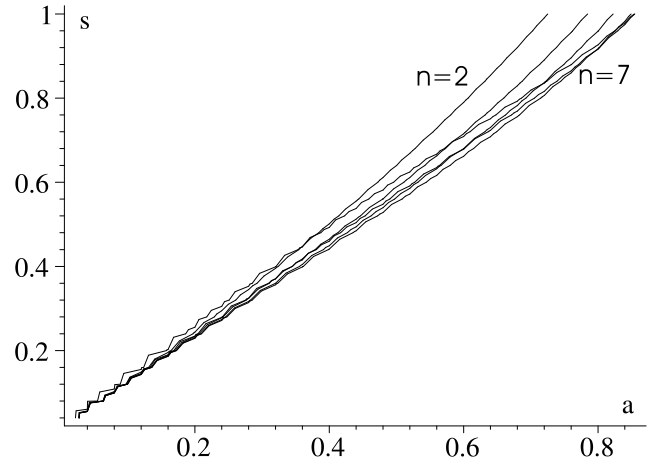


Figure 3.3: The optimal survival probability s for the case of $D^2[\alpha] = 0.2$ and $E[\alpha^2] = 0.29$

However, as the variance decreases, the optimal probability will be greater than the albedo, and will be 1 for higher albedo values, as can be seen in figure 3.4 (e.g. if $a > 0.4$, then the optimal s is 1), where $D^2[\alpha]$ is 0.02 and $E[\alpha^2]$ is 0.29.

The conclusion of examining figures 3.3 and 3.4 is that the optimal survival probability is greater or equal to the albedo. The smaller the variance, the greater the optimal value of s.

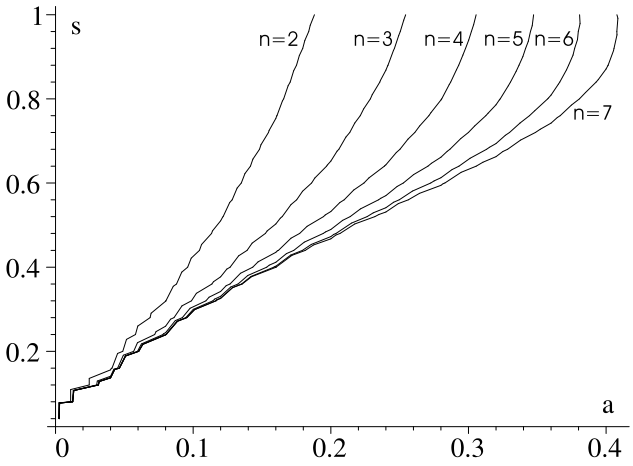


Figure 3.4: The optimal survival probability s for the case of $D^2[\alpha] = 0.02$ and $E[\alpha^2] = 0.29$. If $a > 0.4$, then the optimal s is 1

A particular case is when $a = 0$ for the whole scene, which means that we are about to render a black room, for which equation (3.16) gets the following form:

$$N \cdot D_s^2 = \sum_{i=0}^{n-1} s^i D^2[\alpha]$$

Since s must be in interval $[0, 1]$, thus the minimum of this formula is at $s = 0$. It means that in a completely black room the best survival probability of the walk is zero, as we expected.

3.4 Results

We have made some experiments to demonstrate the theoretical results with a Monte-Carlo global illumination renderer proposed in [SK99a]. The survival probability of the random walk was set to $0.1, 0.2, ..., 0.9$. For each of these values, rendering was performed with 1 500 000 rays, and the variance was calculated.

First we measured the variance in a scene with constant radiance. The internal sphere surface was chosen as it was proposed in [HMF98], since for this scenes analytical solutions are available that can be used as a reference. Then a not constant

radiance function was taken, where the albedo was chosen constant, and the measurement was performed with two different values of the albedo. Finally, a usual scene with no special constraints was examined. To obtain a reference for variance analysis, the reference image was rendered with 150 000 000 rays.

3.4.1 Constant scene

The internal surface of a sphere is used as a test scene. Three images are presented here as can be seen in figure 3.5.

Figure 3.5: Internal surface of a sphere, the images were taken by setting the survival probability to $0.1, 0.2, 0.9$ respectively

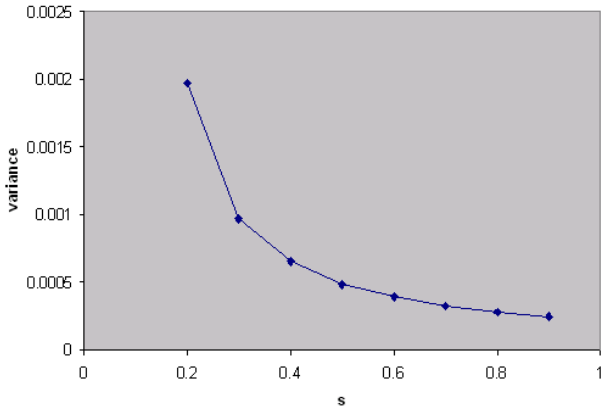


Figure 3.6: The measured variance of the rendering in an internal sphere surface scene

The albedo is chosen to $0.1\pi \approx 0.314$. The total internal sphere surface is a light source. It is known that this rendering should result in a constant image [HMF98]. This is a flat integrand, so the variance is small. Because of this flatness of the integrand, we expect to find that the optimal survival probability of this scene is 1

(as shown by figure 3.2). The empirical variance of the estimation after 1 500 000 steps can be seen in figure 3.6.

3.4.2 Constant albedo

To demonstrate empirically that increasing the albedo, the optimal survival probability goes to 1, we created a heterogeneous scene. An area light source with high emittance value was placed in a sphere, which was chosen as a bounding geometry. The closed space was necessary to forbid rays to escape to outer space, thus assuring that the termination of the walk can be caused only by the Russian roulette. We have placed 3 spheres in the scene, that are different in size and emittance power. At first we set all albedos to $0.1\pi \approx 0.314$, then to $0.15\pi \approx 0.471$. Looking at figures 3.7, we can notice the shift of the curve to the right, indicating the optimal s getting closer to 1. This meets exactly our expectations that the optimal survival probability is proportional to the average albedo.

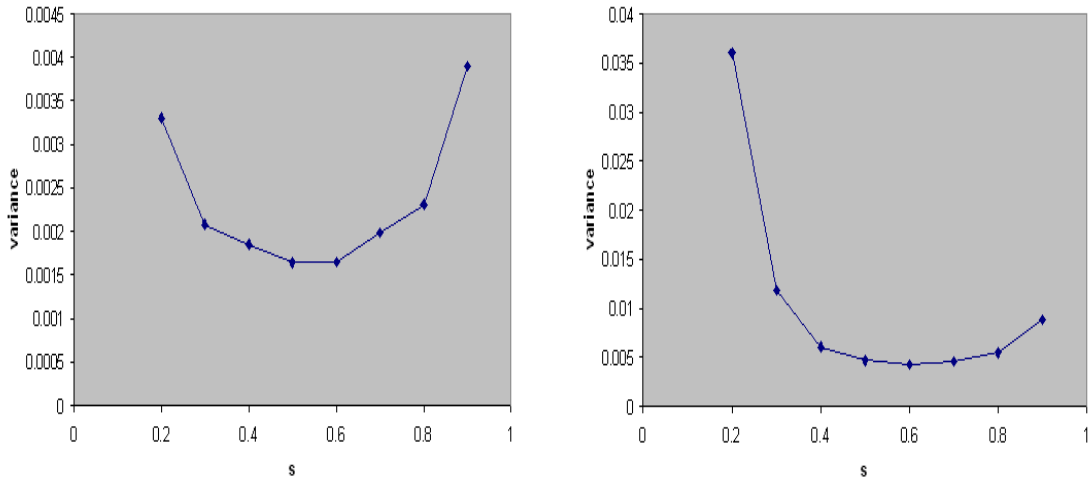


Figure 3.7: The measured variance of the rendering by choosing the albedo to $0.1\pi \approx 0.314$ and $0.15\pi \approx 0.471$ respectively

3.4.3 General scene

This scene has neither homogeneous illumination, nor constant BRDF. The results of rendering with 1 500 000 rays have been compared to a reference image computed with $9 \cdot 10^8$ number of rays. The minimum of the variance curve is about at $s = 0.5$. Looking at figure 3.8, it can be noticed that the second ($s = 0.5$) image approximates the end result better than the first ($s = 0.1$) or third ($s = 0.9$) images.

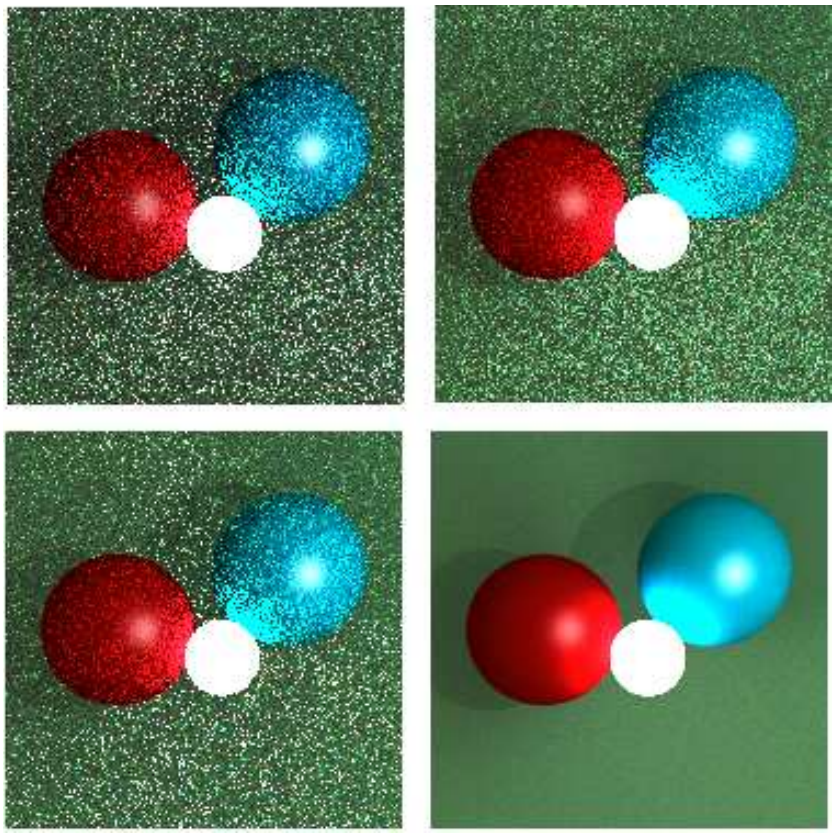


Figure 3.8: Rendered images by setting the survival probability to 0.1 (top left), 0.5 (top right), 0.9 (bottom left) and the reference image (bottom right)

3.5 Summary

In this chapter it was theoretically and empirically shown that the survival probability of the Russian roulette random walk algorithms should be selected carefully. In addition to the local albedo, which is used in the majority of random walk algorithms, we must also consider the variance (as a measure of flatness) of the scene radiance. The flatter the scene radiance, the most probably the optimal s is close to 1, and the optimal survival probability increases with the average albedo. This means that in homogeneous environments it is not suggested to use Russian roulette, or if it is applied, the survival probability should be set to a value that is greater than the albedo.

Chapter 4

Stochastic iteration algorithms

The concept of stochastic iteration has been proposed and applied for the diffuse radiosity problem in [Neu95, NFKP94, NPT+95, SKFNC97], that is for the solution of finite-dimensional linear equations. In this chapter we generalize the fundamental concepts to solve integral equations [SK98, SK99b], then the generalized method will be used for attacking non-diffuse global illumination problems. The extended stochastic iteration presented in section 4.4 is the own work of the author and concerns to thesis 2 (see chapter 8).

4.1 Introduction

Suppose that we have a random linear transport operator $\mathcal{T}^*$ so that

$$E[\mathcal{T}^* L] = \mathcal{T} L, \tag{4.1}$$

that is, this random transport operator gives back the effect of the real light transport operator in the average case. During stochastic iteration a random sequence of operators $\mathcal{T}_1^*, \mathcal{T}_2^*, \ldots, \mathcal{T}_i^*, \ldots$ is generated, which are instantiations of $\mathcal{T}^*$, and this sequence is used in the iteration:

$$L(m) = L^e + \mathcal{T}_m^* L(m-1). \tag{4.2}$$

In order to apply stochastic iteration in practice, the key problem is the definition of the random transport operator. This operator should meet the requirement of equation (4.1) and should be easy to compute. For the continuous case, a single application of the transport operator contains a directional integral. For the finite-element case, the transport operator also includes the projection to the adjoint base which requires additional integration in the domain of basis functions.

When moving towards the non-diffuse case, another requirement must be imposed upon the random transport operator. It must not only meet the requirement of equation (4.1) and be easy to compute, but it must also allow the compact representation

of the $\mathcal{T}_i^* L$ functions. This extra requirement is evident if we take into account that unlike in the diffuse case, the domain of L is a 4-dimensional continuous space, so is the domain of $\mathcal{T}_i^* L$. From the point of view of compactness, we have to avoid the representation of these functions over the complete domain.

Thus those transport operators are preferred, which require the value of L just in a few "domain points" (e.g. in a single "domain point"). As we shall see, this means that we need just one variable per patch even for the non-diffuse case.

4.2 Single ray based iteration

In this section we present a random transport operator that uses a single ray having random origin $\vec{y}_i$ and direction $\vec{\omega}_i$ generated with a probability that is proportional to the cosine weighted radiance of this point at the given direction. This ray transports the whole power of the scene

$$\Phi = \int_S \int_\Omega L(\vec{y}, \vec{\omega}') \cos\theta_{\vec{y}} \; d\omega' \; dy$$

to that point $\vec{x}$ which is hit by the ray. Formally, the random transport operator is

$$(\mathcal{T}^* L)(\vec{x}, \vec{\omega}) = \Phi \cdot \delta(\vec{x} - rc(\vec{y}, \vec{\omega}_i)) \cdot f_r(\vec{\omega}_i, \vec{x}, \vec{\omega}), \tag{4.3}$$

where rc is the ray-casting operator and δ is the Dirac-delta function.

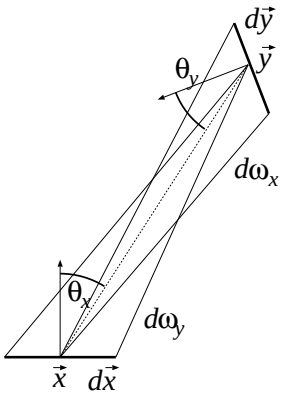


Figure 4.1: Symmetry of solid angles of shooting and gathering

We prove that this random operator complies with equation (4.1). The probability density of selecting surface point $\vec{y}$ and direction $\vec{\omega}'$ is

$$\frac{d\Pr\{\vec{y}, \vec{\omega}'\}}{dy \; d\omega_{\vec{y}}} = \frac{L(\vec{y}, \vec{\omega}') \cdot \cos\theta_{\vec{y}}}{\Phi}. \tag{4.4}$$

Since the definition of the solid angle is

$$d\omega_{\vec{y}} = \frac{dx \cdot \cos\theta'_{\vec{x}}}{|\vec{y} - \vec{x}|^2},$$

we can write a symmetry relation (figure 4.1) for the shooting and gathering solid angles:

$$dy \cdot d\omega_{\vec{y}} \cdot \cos\theta_{\vec{y}} = dy \cdot \frac{dx \cdot \cos\theta'_{\vec{x}}}{|\vec{y} - \vec{x}|^2} \cdot \cos\theta_{\vec{y}} =$$

$$dx \cdot \frac{dy \cdot \cos\theta_{\vec{y}}}{|\vec{y} - \vec{x}|^2} \cdot \cos\theta'_{\vec{x}} = dx \cdot d\omega'_{\vec{x}} \cdot \cos\theta'_{\vec{x}}. \tag{4.5}$$

Thus the probability of selecting $\vec{y}, \vec{\omega}'$ can also be expressed by:

$$d\Pr\{\vec{y}, \vec{\omega}'\} = \frac{L(\vec{y}, \vec{\omega}') \cdot \cos\theta_{\vec{y}}}{\Phi} \cdot dy \; d\omega_{\vec{y}} = \frac{L(rc(\vec{x}, -\vec{\omega}'), \vec{\omega}') \cdot \cos\theta_{\vec{x}}}{\Phi} \cdot dx \; d\omega'_{\vec{x}}.$$

After that it is easy to prove that the random transport operator meets the requirement of equation (4.1) since

$$E[(\mathcal{T}^* L)(\vec{x}, \vec{\omega})] = \int_S \int_\Omega \Phi \cdot \delta(\vec{x} - rc(\vec{y}, \vec{\omega}')) \cdot f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \; d\Pr\{\vec{y}, \vec{\omega}'\} =$$

$$\int_\Omega L(rc(\vec{x}, -\vec{\omega}'), \vec{\omega}') \cdot \cos\theta'_{\vec{x}} \cdot f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \; d\omega'_{\vec{x}} = (\mathcal{T} L)(\vec{x}, \vec{\omega}).$$

There can be two possible outcome when applying the random operator. On the one hand the result can be a single point that receives all the power and reflects some part of it back to the scene. On the other hand it is possible that the ray casting operator find no point at all in which case the ray leaves the scene.

Suppose that the first random operator $\mathcal{T}_1^*$ is applied to L^e which may transfer all power

$$\Phi_1 = \int_S \int_\Omega L^e(\vec{y}_1, \vec{\omega}_1) \cos\theta_{\vec{y}_1} \; d\omega_1 \; dy_1,$$

to a single point $\vec{x}_1 = rc(\vec{y}_1, \vec{\omega}_1)$ using probability density

$$\frac{d\Pr_1\{\vec{y}_1, \vec{\omega}_1\}}{dy_1 d\omega_1} = \frac{L^e(\vec{y}_1, \vec{\omega}_1) \cdot \cos\theta_{\vec{y}_1}}{\Phi}.$$

Before continuing with the second step of the iteration, the radiance should be measured, that is, an image estimate should be computed from $L^e + \mathcal{T}_1^* L^e$. We can separately calculate the effect of the light sources on the image and then add the effect of $\mathcal{T}_1^* L^e$. Note that $\mathcal{T}_1^* L^e$ is concentrated in a single point, thus its contribution can be computed by tracing a ray from the eye to this point, and if this point is not occluded, then evaluating the $f_r(\vec{\omega}_1, \vec{x}, \vec{\omega}_{eye}) \cdot \Phi$ expression.

The second operator $\mathcal{T}_2^*$ should be applied to

$$L_1 = L^e + \mathcal{T}_1^* L^e,$$

thus both the total power Φ and the probability density have been modified:

$$\Phi_2 = \int_S \int_\Omega L_1(\vec{y}_2, \vec{\omega}_2) \cos\theta_{\vec{y}_2} \; d\omega_2 \; dy_2 = \Phi_1 \cdot (1 + a_{\vec{x}_1}(\vec{\omega}_1))$$

where $a_{\vec{x}_1}$ is the albedo at point $\vec{x}_1$ defined by

$$a_{\vec{x}}(\vec{\omega}) = \int_\Omega f_r(\vec{\omega}, \vec{x}, \vec{\omega}') \cos\theta'_{\vec{x}} \; d\omega',$$

and the new probability density is

$$\frac{d\mathrm{Pr}_2\{\vec{y}_2, \vec{\omega}_2\}}{dy_2 d\omega_2} = \frac{L_1(\vec{y}_2, \vec{\omega}_2) \cdot \cos\theta_{\vec{y}_2}}{\Phi} =$$

$$\frac{L^e(\vec{y}_2, \vec{\omega}_2) \cdot \cos\theta_{\vec{y}_2} + f_r(\vec{\omega}_1, \vec{y}_2, \vec{\omega}_2) \cdot \cos\theta_{\vec{y}_2} \cdot \delta(\vec{y}_2 - \vec{x}_1)}{\Phi_1(1 + a_{\vec{x}_1}(\vec{\omega}_1))}.$$

Sampling according to this mixed, discrete-continuous probability density can be realized in the following way. First it is decided randomly whether we sample L^e or the newly generated point using probabilities $1/(1+a_{\vec{x}_1}(\vec{\omega}_1))$ and $a_{\vec{x}_1}(\vec{\omega}_1)/(1+a_{\vec{x}_1}(\vec{\omega}_1))$, respectively. If L^e is selected, then the sampling process is the same as before, i.e. a random point and a random direction are found with probability density

$$\frac{L^e(\vec{y}_2, \vec{\omega}_2) \cos\theta_{\vec{y}_2}}{\Phi_1}.$$

However, if the new point is chosen, then the direction $\vec{\omega}_2$ of the next transfer is found with probability density

$$\frac{f_r(\vec{\omega}_1, \vec{y}_2, \vec{\omega}_2) \cos\theta_{\vec{y}_2}}{a_{\vec{x}_1}(\vec{\omega}_1)}.$$

In either case, a ray defined by the selected point and direction is traced, and the complete power $\Phi_2 = \Phi_1 \cdot (1 + a_{\vec{x}_1}(\vec{\omega}'_1))$ is transferred to that point which is hit by the ray. The subsequent steps of the iteration are similar.

This iteration can be also considered as a sequence of variable length random walks, since at each step the point that is last hit by the ray is only selected with a given probability as the starting point of the next ray. The algorithm initiates a random walk by selecting a point from a light source. After each step the walk can be finished with probability $1/(1+a_{\vec{x}_i}(\vec{\omega}_i))$ and also when the ray hits no object. In case of finishing, another walk is started from the light source. In case of continuation, the transferred power is weighted by $(1 + a_{\vec{x}_i}(\vec{\omega}_i))$. This is exactly the so called Russian roulette [AK90, SP94b] technique.

4.3 Parallel ray-bundle iteration

In this section a specific algorithm is discussed that transfers the radiance of all patches to a randomly selected global direction in each iteration cycle. If the algorithm transfers the radiance into a randomly selected direction $\vec{\omega}'$, the random transport operator is

$$L^r(\vec{x},\vec{\omega}) = \mathcal{T}^* L = 4\pi \cdot L(rc(\vec{x},-\vec{\omega}'),\vec{\omega}') \cdot f_r(\vec{\omega}',\vec{x},\vec{\omega}) \cdot \cos\theta'.$$

Indeed, if the direction is sampled uniformly, then its probability density is $1/4\pi$, thus the expectation of the random transport operator gives back the effect of the light transport operator $\mathcal{T}L$, as required by equation (4.1):

$$E[\mathcal{T}^* L] = \int_{\Omega'} 4\pi \cdot L(rc(\vec{x},-\vec{\omega}'),\vec{\omega}') \cdot f_r(\vec{\omega}',\vec{x},\vec{\omega}) \cdot \cos\theta' \cdot \frac{d\omega'}{4\pi} = \mathcal{T}L.$$

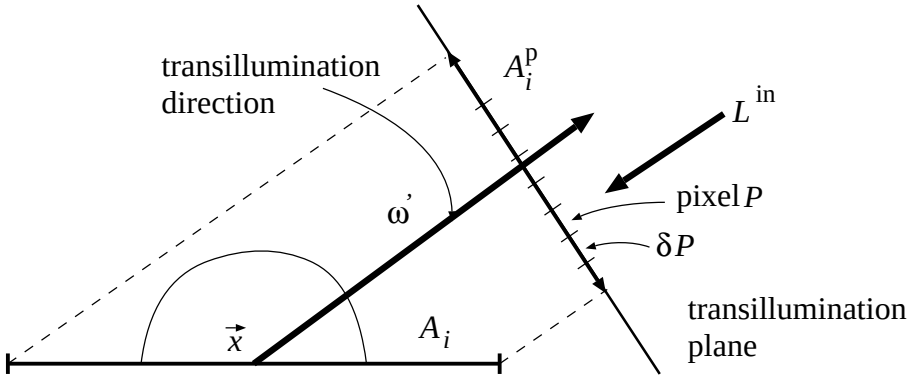


Figure 4.2: Integration on the transillumination plane

From the reflected radiance the average radiance of a patch can be obtained by a simple averaging operation:

$$\tilde{L}(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}L(m-1)\ dx.$$

The direction $\vec{\omega}'$ defines the flow of the radiance, thus it is also called the *transillumination direction* (figure 4.2). Let us place a plane that is perpendicular to this direction and define a window on it, which can include the projections of all objects. The window is decomposed into pixels. The window is also called the *transillumination window* and its plane is the *transillumination plane*. Note that integral of the average radiance can also be evaluated on the window, when the cosine factor is compensated:

$$\tilde{L}(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}^* L(m-1)\ dx \approx \frac{4\pi \cdot \delta P}{A_i} \cdot \sum_P L^{in}(P) \cdot f_r(\vec{\omega}', P, \vec{\omega}),$$

where P runs on the pixels covering the projection of patch i, $L^{in}(P)$ is the radiance of the surface point visible in pixel P, $f_r(\vec{\omega}', P, \vec{\omega})$ is the BRDF of that point which receives this radiance coming through pixel P and δP is the area of the pixels.

In order to avoid the complete representation of the radiance function during iteration, we store the irradiance

$$I(m)|_i = \frac{4\pi \cdot \delta P}{A_i} \cdot \sum_P L^{in}(P)$$

on each patch. Note that this is independent of the direction $\vec{\omega}$ in the next iteration. Thus when carrying out the next step and $\vec{\omega}$ is known, the irradiance is multiplied by $f_r(\vec{\omega}', P, \vec{\omega})$ to obtain the patch radiance $\tilde{L}(m)|_i$ from $I(m)|_i$. In this way, it is enough to store just one variable in each patch to hold the irradiance.

It is straightforward to extend the method to be bi-directional, which transfers the radiance not only into direction $\vec{\omega}'$ but also to $-\vec{\omega}'$. Note that this does not even require additional visibility computation. If the global directions are sampled from a uniform distribution and the radiance is transferred into two opposite directions, then the directional density is:

$$p(\vec{\omega}') = \frac{1}{2\pi}.$$

Thus the random transport operator changes to

$$L^r(\vec{x}, \vec{\omega}) = \mathcal{T}^* L = 2\pi \cdot L(rc(\vec{x}, -\vec{\omega}'), \vec{\omega}') \cdot f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta'.$$

Let us suppose that the initial finite-element structure is a family of triangles obtained from a tessellation process and a single radiance value is used for all directions.

4.3.1 Computation of the radiance transfer in a single direction

The radiance transfer needs the identification of those points that are mutually visible in the global direction. In order to solve this global visibility problem, three methods are presented.

1. Radiance transfer with the painter's algorithm

If the patches are sorted in the transillumination direction and processed in this order, the computation of the radiance transfer requires the determination of the pixel values inside the projection of patch i. Then, to proceed with the next patch in the given order, the pixels covered by patch i are filled with i if patch i is not front facing and 0 otherwise. The two steps can be done simultaneously by a modified scan-conversion algorithm that reads the value of the image buffer before modifying it.

This is summarized in the following algorithm [SKF97]:

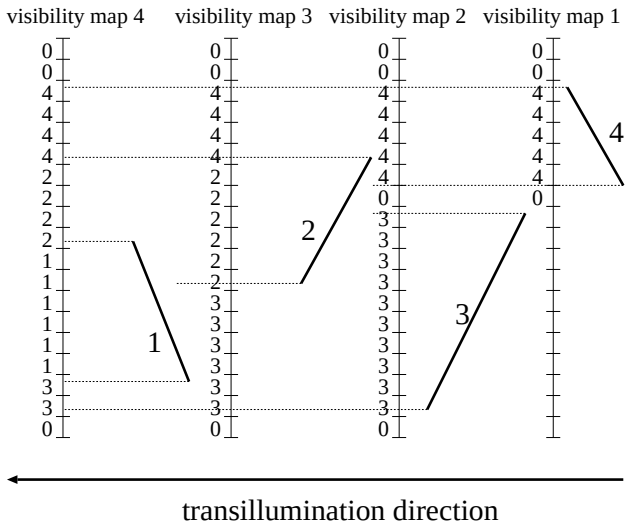


Figure 4.3: Application of painter's algorithm

```
Sort patches in direction ω⃗′ (painter's algorithm)
Clear image
for each patch i in the sorted order {
    if (patch i is front facing)
        for each pixel of patch i {
            j = Read pixel
            Transfer radiance between patches i and j
            Write 0 to the pixel
        }
    else
        Render patch i with color i
}
```

Sorting a data set is known to have $(n \log n)$ time complexity (quick sort is used), so does the painter's algorithm in the average case. A single cycle of the second `for` loop contains only instructions that work with a single patch and an "image", thus the time required for a single cycle is independent of the number of patches. Since the `for` loop executed n number of times, the time complexity of the `for` loop is (n). Consequently the algorithm requires $(n \log n)$ time.

However, using the painter's algorithm can result difficulties. The most straightforward is that sometimes it is not possible to unambiguously decide which patch lays before the other in the sorting order. The classical examples are when patch A hides partially patch B and patch B hides partially patch A. But there are cases when there can be difficulties due to limited floating point precision also. Usually the quick sort algorithm sort patches according to the coordinates of the center of the patch (or the

minimum coordinate of the patch in the transillumination direction, or whatever). This can result in an error in the final image. Imagine an architectural model with a very thin wall (see figure 4.4), where one side of the wall (surface A) is illuminated by a bright light source and the other side of the wall (surface B) stays in shadow. Their normal vectors are opposite to each other.

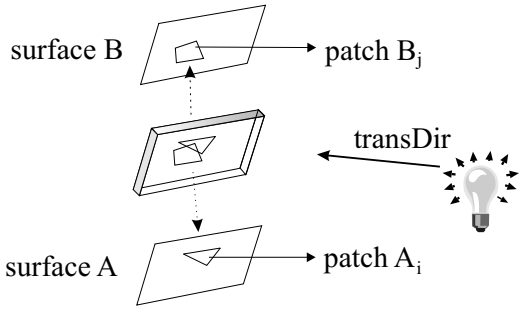


Figure 4.4: The problem with the painter's algorithm in parallel ray-bundle iteration

The two sides of the wall are so close together that it is possible that the quick sort algorithm arranges them in a wrong order (since it considers only the minimum Z, the maximum Z or the Z of the center for the patch). If it happens that patch A_i is the first in the order, then it radiates light energy to patch B_j (since the normal vector of patch A_i and patch B_j are opposite). This is of course an error, and this extra light can increase the total light power in the scene. Since it is an iteration algorithm, if this extra light cannot be compensated by the absorption of the materials, the scene has more and more flux in each iteration, the scene is overlit and finally this algorithm fails to render the correct image.

Radiance transfer with a software z-buffer algorithm

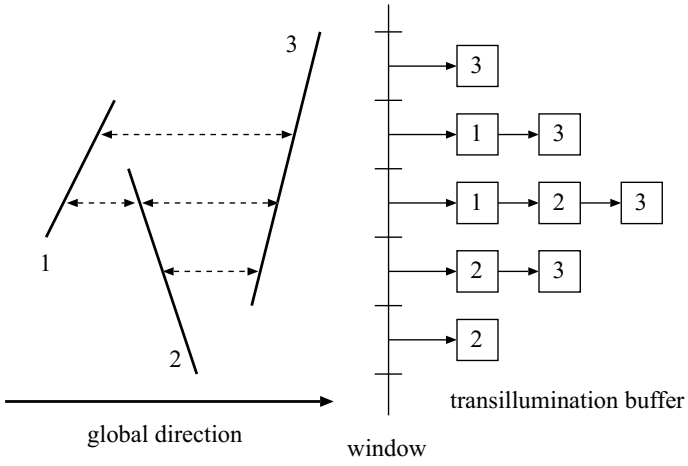


Figure 4.5: Organization of the transillumination buffer with software z-buffer

The second method is an extension of the z-buffer algorithm. The main difference is that now a pixel is capable to store a list of patch indices and z-values. The lists are sorted according to the z-values. The patches are rendered one after the other into the buffer using a modified z-buffer algorithm which keeps all visible points not just the nearest one. Traversing the generated lists the pairs of mutually visible points can be obtained. For each pair of points, the radiance transfer is computed and the transferred radiance is multiplied by the BRDF, resulting in the reflected radiance L^r.

Radiance transfer with the hardware z-buffer

The drawback of the previous algorithm is that it cannot exploit the hardware z-buffer that can store only a single value per pixel, because the algorithm requires all patches that are projected onto this pixel. Fortunately, this requirement can be eliminated, thus the algorithm can be executed on the hardware, if the visibility algorithm is further randomized in the following way.

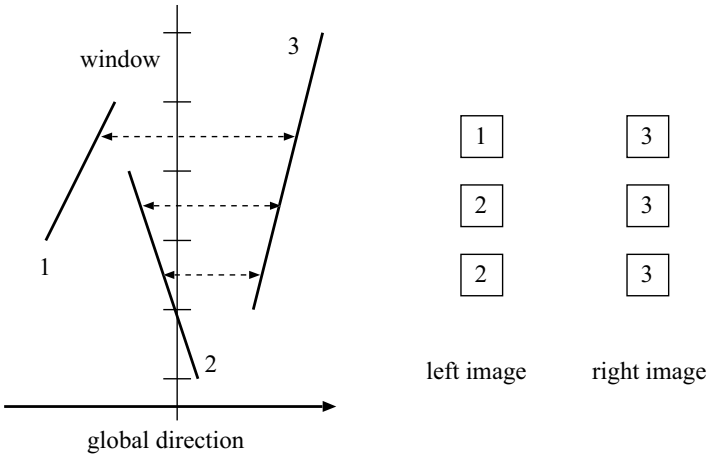


Figure 4.6: Transferring the radiance through a single plane with hardware z-buffer

Let us find randomly a point on the line of the transillumination direction and place the transillumination window at this point. The scene is rendered from the two sides of the window supposing that the color of patch i is i. Having read the two images from the frame buffer, the patches that see each other from the opposite sides of the window can be identified, and the radiance can be transferred. Of course, this method finds two points that see each other in the transillumination direction only with some probability. This probability is proportional to the distance between the two points. If the distance of the front and back clipping planes is R, then the probability is $|\vec{x} - \vec{y}|/R$. When the scene is rendered, the z coordinates are transformed in a way that they fit in the [0,1] range for the whole scene. It means that this probability equals to the sum of the z values of the two visible points, as read out from the z-buffer. In order to compensate those cases when the two points

are not on the opposite sides of the transillumination window, when the radiance is transferred, it is divided by the selection probability, i.e. by the normalized distance of the two points. Formally, the random transport operator is:

$$\mathcal{T}_2^* L = 2\pi \cdot L(\vec{y}, \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}} \cdot \xi(\vec{x}, \vec{y}) \cdot \frac{R}{|\vec{x} - \vec{y}|},$$

where $\xi(\vec{x}, \vec{y})$ is the indicator function, which is 1 if and only if the transillumination plane is between $\vec{x}$ and $\vec{y}$.

When a single plane is set for each direction, this method introduces too much additional noise. The noise can be reduced if not only a single plane, but a few (for example 4) of them are used.

Incoming first-shot

The proposed method selects bundles of parallel lines to transfer the radiance blindly without considering where the important sources are. On the one hand, this is good, since very many rays can be traced simultaneously in a single step (note that if the resolution of the transillumination buffer is 1000×1000, then a single transfer corresponds to tracing a million bi-directional global lines where all intersections are used, i.e. at least to 2 million rays). On the other hand, this becomes ineffective if the initial radiance is very heterogeneous due to the small bright light sources. These small light sources need special treatment that is generally called as the *first-shot* or *direct light source computation*.

First-shot can be formulated as decomposing the radiance into emission L^e and reflection L^r and deriving an appropriate form of the rendering equation for the reflection term. Substituting the $L = L^e + L^r$ decomposition into the rendering equation, we can obtain the following formulae:

$$L^e + L^r = L^e + \mathcal{T}(L^e + L^r) \Longrightarrow L^r = (\mathcal{T}L^e) + \mathcal{T}L^r = L^d + \mathcal{T}L^r.$$

This equation is similar to the original rendering equation. The only difference is that the original emission function L^e is replaced by its single reflection $L^d = \mathcal{T}L^e$.

In order to compute and store the first reflection of the emission function, point samples are defined on the small light sources and hemicubes are placed above these point samples [SKSMT00]. Running z-buffer/constant shading visibility algorithms for the sides of the hemicubes, the visible triangles and their visible portions can be identified. When the radiances of the points of a triangle are computed, first it is determined whether or not the triangle is seen through some hemicube face from the light source. Then those light vectors [ZBP99] are calculated on each patch, which can represent the illumination coming from the light sources. Note that unlike in other first-shot algorithms developed for diffuse radiosity, here the incoming radiance is stored, from which the outgoing radiance can be obtained by multiplying it with the BRDF taking into account the incoming and outgoing directions.

Note that the area lightsources is decomposed into a finite number of point lightsources. However, due to the fact that the first-shot of l point lightsources requires l additional variables per patch, this approach becomes very memory demanding.

4.4 Extended stochastic iteration with parallel ray-bundles

The stochastic iteration method proposed so far computes a single color value for each patch. In the final rendering these colors are smoothed according to Gouraud shading. However, Gouraud shading is not very nice if the surfaces are specular, thus it is worth replacing this operation by Phong shading. On the other hand, Phong shading is much more time consuming, thus we apply Phong shading only for the direct reflection components, the indirect reflection is still visualized with Gouraud shading. The other possibility of improvement is to apply the concept of the *separation of the main part* introduced in the theory of Monte-Carlo methods. It means that a rough approximation is also represented in object space. This speeds up the convergence and can be very useful when animations are rendered.

Let us decompose the radiance function L to emission L^e and to reflected component L^r:

$$L = L^e + L^r.$$

The reflected component is further subdivided into direct reflection L^d for which efficient and hardware supported algorithms are available, and to an indirect reflection L^{id}, which consists of its main part $\tilde{L}^{id}$ (called the finite-element component), and an indirect residuum $\Delta L^{id}(\vec{\omega})$ (called the Monte-Carlo component) that is estimated by Monte-Carlo simulation:

$$L^r = L^d + L^{id} = L^d + \tilde{L}^{id} + \Delta L^{id}. \tag{4.6}$$

In order to keep the storage requirements low, the finite-element part will be constant on each patch and will represent the average radiance. The directional average is obtained as:

$$\tilde{L}^{id} = \frac{1}{\pi} \cdot \int_{\Omega} L^{id}(\vec{\omega}) \cdot \cos\theta \; d\omega.$$

If the patch receives illumination just from direction $\vec{\omega}'$ and the irradiance is $I(\vec{\omega}')$ — this will be the usual case in the proposed algorithm — then the average radiance can be obtained from the albedo $a(\vec{x}, \vec{\omega}')$:

$$\tilde{L}^{id} = I(\vec{\omega}') \cdot \frac{1}{\pi} \cdot \int_{\Omega} f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cdot \cos\theta \; d\omega = I(\vec{\omega}') \cdot \frac{a(\vec{x}, \vec{\omega}')}{\pi}.$$

The albedo can be computed in the preprocessing phase for each possible material and stored in tables or can be estimated on the fly.

Let us substitute this decomposition into the stochastic iteration formula (equation (4.2)). Subtracting L^e from both sides and using $L^d = \mathcal{T}L^e$, we can obtain:

$$L^r(m) = L^d + L^{id}(m) = L^d + \mathcal{T}^*(L^d + \tilde{L}^{id}(m-1) + \Delta L^{id}(m-1)).$$

To obtain the radiance value of patch i, the radiances of its points are averaged:

$$L^{id}(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}^* L^r(m-1)\, dx. \tag{4.7}$$

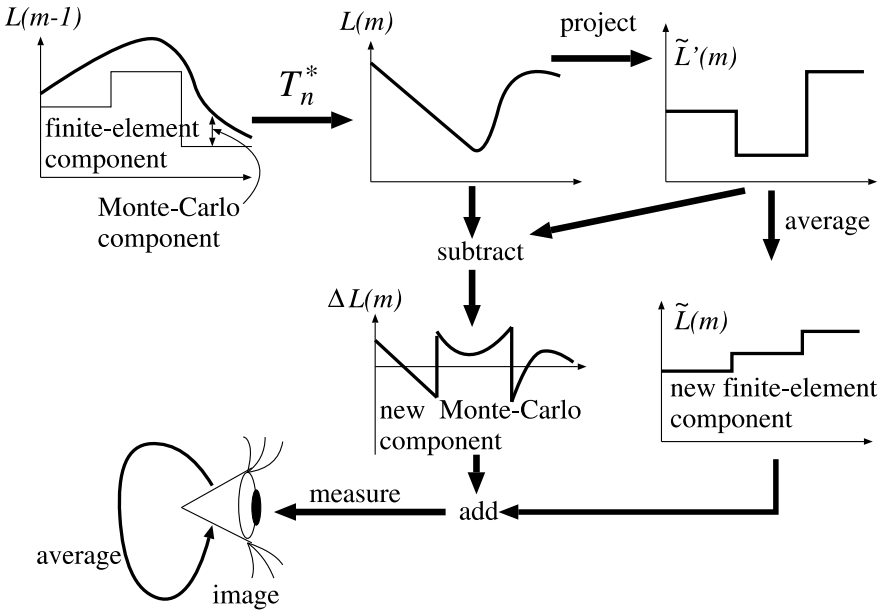


Figure 4.7: Dataflow in the new algorithm

In each iteration step the radiance average is obtained, and an image estimate is computed from the actual radiance. Note that the image estimates and the finite-element components obtained in an iteration step — as stochastic iteration in general — will not converge, but they will fluctuate around the real solution. Thus the final image is obtained as the average of these image estimates, and the finite-element component as the average of the finite-element components of different iteration steps. If the finite-element projection of the indirect reflection at step m is $\tilde{L}'(m)$, then the finite-element part may be derived as follows:

$$\tilde{L}^{id}(m) = \frac{1}{m} \cdot \sum_{n=1}^{m} \tilde{L}'(n) = \frac{1}{m} \cdot \tilde{L}'(m) + \left(1 - \frac{1}{m}\right) \cdot \tilde{L}^{id}(m-1). \tag{4.8}$$

The Monte-Carlo component, which is obtained as a difference between the actual radiance estimate and its finite-element projection, is used to correct the finite-element approximation.

The complete algorithm is:

StohasticIterationExtended
$\tilde{L}^{id}(0) = 0$, $\Delta L^{id}(0) = 0$
$L^d = \mathcal{T} L^e$
`for` $(m = 1; m \leq M; m{+}{+})$ {
$L^{id}(m) = \mathcal{T}^*(L^d + \tilde{L}^{id}(m-1) + \Delta L^{id}(m-1))$
$\tilde{L}'(m) =$ average of $L^{id}(m)$
$\Delta L^{id}(m) = L^{id} - \tilde{L}'(m)$
$\tilde{L}^{id}(m) = 1/m \cdot \tilde{L}'(m) + (1 - 1/m) \cdot \tilde{L}^{id}(m-1)$
$C'(m) = \mathcal{M}(\tilde{L}^{id}(m) + \Delta L^{id}(m))$
$C^{id}(m) = 1/m \cdot C'(m) + (1 - 1/m) \cdot C^{id}(m-1)$
}
$C(M) = \mathcal{M}(L^e + L^d) + C^{id}(M)$
Display $C(M)$ colors
}

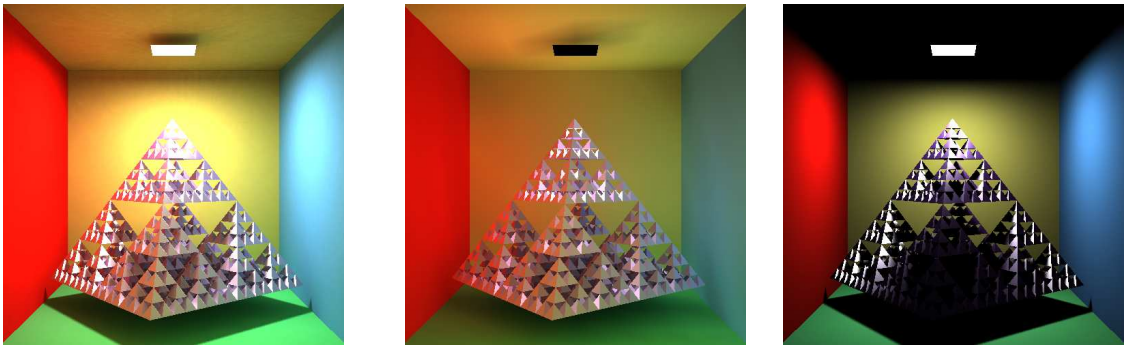
The dataflow of the algorithm is shown in figure 4.7. Note that the new reflected radiance $\tilde{L}^{id}(m) + \Delta L^{id}(m)$ is computed from the radiance generated by the random transport operator as first subtracting its finite-element projection then adding the average of these finite-element projections. At the beginning of the execution of the algorithm, this replaces the high-variance main part by its estimated average, which is responsible for good initial convergence. Later, when the algorithm converges, the expected finite-element component gets close to its average, thus subtraction and addition compensate each other and the finite-element approximation does not distort the final result. We could have the speed of the iteration together with the asymptotic accuracy of random walks.

This is a generic algorithm from which different specific versions can be built by inserting the random transport operator. The algorithm will be fast if the application of the random transport operator $\mathcal{T}^* L$ results in a low variance random variable. Note that this depends not only on the random transport operator but also on the current radiance function. With other words, for a different actual radiance function, a different transport operator can be the winner of this game.

4.5 Results

We have implemented the proposed method. The rendered architectural scenes are presented in figure 4.9. The images have been rendered with 500×500 resolution. The transillumination buffer contained 1000×1000 pixels. The running times given in the following sections are measured on a laptop with 500 MHz Pentium®III processor and with no graphics accelerator.

Figure 4.8 shows a scene of a 3D Sierpinsky set, that has 4479 patches. The diffuse albedo of the patches in this set is $(0.09, 0.03, 0.06)$ on the wavelengths 400 nm, 552



L global illumination solution $\tilde{L}^r$ finite-element component L_{direct} direct illumination

Figure 4.8: Sierpinsky set rendered with the extended ray bundle algorithm

nm and on 700 nm, respectively. The specular albedo is wavelength independent and is between 0.8 and 0.4 depending on the viewing angle. The "shine" parameter of the max-Phong reflection model is 3. The walls are diffuse. The area light source is sampled at 18 discrete points. The image was rendered with the proposed method in 6 minutes. In addition to the rendered scene, the finite-element reflected component $\tilde{L}^r$ visualized by Gouraud shading and the direct illumination L_{direct} are also shown. Note that these two components really represent the major part of the radiance. On the other hand, the incorrect shadow smearing of the finite-element component in the ceiling is completely removed by the Monte-Carlo component in the final image.

Figure 4.9: House modelled in ArchiCAD. The images are rendered with parallel ray-bundles.

Figure 4.9 contains snap-shots of a virtual house. The walls have 0.22 diffuse and 0.36 specular albedo, the shine parameter is 3. The frame of the doors and the windows have 0.4 diffuse albedo and 0.58 specular albedo with a shine parameter of 35. The model consists of 31.000 polygons that are tessellated to 241.000 patches. A single iteration requires about 1.5 second.

4.6 Summary

In this chapter we have described two stochastic iteration algorithms: the single ray shooting and the parallel ray-bundle iteration. After the mathematical background we presented three approaches for implementing parallel ray-bundle iteration. The most straightforward is based on the painter's algorithm. More advanced methods are based on software z-buffer and the hardware z-buffer. Then the parallel ray-bundle iteration was extended. The basic idea is to decompose the radiance function to a finite-element component that is only a rough estimate and to a difference component that is obtained by Monte-Carlo techniques. The classical iteration using finite-elements and random walks are handled uniformly in the framework of stochastic iteration. This uniform treatment allows the finite-element component to be built up adaptively aiming at minimizing the Monte-Carlo component.

Chapter 5

Stochastic iteration with perspective ray-bundles

The hemicube is a classical tool to transfer the light power in diffuse radiosity algorithms. The main advantage of the hemicube based light transfer is that the visible patches can easily be identified by the graphics hardware. This chapter extends the hemicube approach to solve the non-diffuse global illumination problem. In order to get rid of the quadratic complexity of classical radiosity algorithms and to allow using specular surfaces without storing directional finite-elements, the original radiosity iteration is replaced by stochastic iteration. Random selection, however, may introduce noise that is particularly significant where the source and receiver patches are close. We also propose a solution strategy to eliminate these artifacts. The chapter also discusses further improvements by applying constant radiance step and by the randomization of the hemicube. The perspective ray-bundles with all of its improvements is the own work of the author and summarized in thesis 3 (see chapter 8).

5.1 The new algorithm

Parallel ray-bundles (see section 4.3) transferred the radiance of all points in a single random direction. An orthogonal approach would select a single random point and transfer its radiance into all possible direction. This approach is called *perspective ray-bundle* algorithm.

Suppose that patch j is selected with probability p_j and point $\vec{y}$ on this patch with uniform $1/A_j$ probability density. According to importance sampling, it is worth setting the selection probability p_i proportional to the powers of the patches.

Let us define the random transport operator as transferring the radiance $L(\vec{y}, \vec{\omega}'_{\vec{y}\to\vec{x}})$ of this point, divided by its selection probability p_j/A_j, to all other visible points $\vec{x}$. When the radiance arrives at point $\vec{x}$, it is reflected according to the BRDF of the material here, and thus results in a new radiance value. Formally, the random transport

operator is

$$(\mathcal{T}^* L)(\vec{x}, \omega) = \frac{A_j}{p_j} \cdot V(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2}, \tag{5.1}$$

where $V(\vec{x}, \vec{y})$ is the mutual visibility indicator, which is 1 if the two points are visible from each other.

In order to show that this random transport operator is appropriate, we have to prove that the expected value of its effect gives back the application of the original light transport operator. The expected value of the random radiance after the transfer is:

$$E[\mathcal{T}^* L] = \sum_j p_j \cdot \int_{A_j} (\mathcal{T}^* L)(\vec{x}, \omega) \ \frac{dy}{A_j} =$$

$$\sum_j \int_{A_j} V(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2} \ dy.$$

Using the formula of solid angles $dy \cdot \cos\theta_{\vec{y}} / |\vec{x} - \vec{y}|^2 = d\omega_{\vec{x}}$ and assuming that illumination can only come from surfaces — i.e. there is no external sky light illumination — the integration over all surfaces can be replaced by an integration over all incoming solid angles:

$$E[\mathcal{T}^* L] = \int_{\Omega'} L(rc(\vec{x}, -\omega'), \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}} \ d\omega'_{\vec{x}}.$$

Thus we could prove that the expectation of the application of the random operator really gives back the effect of the real transport operator, thus the requirement of equation (4.1) is met.

To obtain the radiance on the receiver patch, the radiances of its points, which have been computed according to equation (5.1), are averaged:

$$L(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}^* L(m-1) \ dx =$$

$$\frac{A_j}{p_j A_i} \int_{A_i} V(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2} \ dx. \tag{5.2}$$

Let us interpret equation (5.2). The new radiance of patch i depends on the probability p_j of selecting the shooting patch, the radiance of the shooting point towards the receiver points $L(\vec{y}, \omega'_{\vec{y}\to\vec{x}})$, on a geometric factor

$$G_{\vec{y}\to A_i} = \frac{A_j}{A_i} \int_{A_i} V(\vec{x}, \vec{y}) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{\pi |\vec{x} - \vec{y}|^2} \ dx,$$

and on the BRDF at the receiving point from the direction of the shooting point

$$\rho(\vec{y} \to \vec{x}) = f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \pi.$$

Conceptually, this is very similar to the diffuse case except for the facts that we used the direction dependent radiance and the BRDF instead of the direction independent radiosity and diffuse albedo. Note that the formula has been divided by π in the geometric factor and multiplied by π in the BRDF, in order to give back the classical radiosity interpretation in the special case. The geometric factor can also be given a classical interpretation. Note that the integral in the geometric term is the point-to-patch form factor $F_{\vec{y}\to A_i}$. If patch j is small, then this point-to-patch form factor approximates well the patch-to-patch form factor $F_{A_j\to A_i}$, thus

$$G_{\vec{y}\to A_i} = \frac{A_j}{A_i} \cdot F_{\vec{y}\to A_i} \approx \frac{A_j}{A_i} \cdot F_{A_j\to A_i} = F_{A_i\to A_j}$$

according to the symmetry relation of the form factors. Despite to the conceptual similarities, the formulation of equation (5.2) is more complicated formally, since these factors cannot be decomposed and the radiance cannot be obtained as their simple product. The reason of this notational complexity is that now the radiance and the BRDFs depend on the direction as well. In fact, the product form is valid only for differential surface elements on patch i. However, if patch i is small compared with its distance to point $\vec{y}$, we can still use the following approximation:

$$L(m)|_i \approx \frac{L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot F_{A_i\to A_j} \cdot \rho(\vec{y} \to \vec{x})}{p_j}, \tag{5.3}$$

where $\vec{x}$ is the center of patch A_i. We should emphasize that the implementation of the algorithm does not use this approximation. This formula, however, will be useful to understand the heuristic variance reduction technique, which is presented later in subsection 5.4.

5.1.1 Representation of the radiance function

The radiance function now depends on both the surface point and the direction, thus its accurate finite-element representation would be too expensive. Fortunately, the radiance is needed only for computing the random transfer from a single patch and its image contribution. Note that these tasks require the radiance function just in a small subdomain compared to the set of all points and directions.

In order to compute the radiance transfers and the image contribution without explicitly storing the direction dependent radiance function itself, instead of the outgoing radiance L, the incoming radiance I is associated with each patch. If the sender and receiver patches are small compared to their distances, then we can assume that a patch may receive radiance only from a single direction, thus I is non-zero only

for the direction pointing from the previously selected patch to the currently selected patch. Thus the incoming radiance is represented by two variables per patch, the intensity I and the direction of the last transfer ω'. From the incoming radiance, the outgoing radiance can be obtained by a multiplication with the local BRDF:

$$L(\vec{x}, \omega) = I(\omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}}.$$

5.1.2 Computation of the radiance transfer by hemicubes

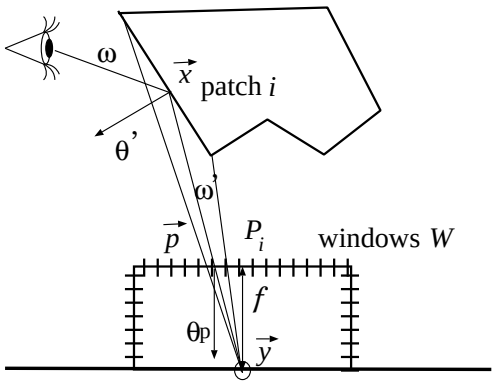


Figure 5.1: Hemicube shooting

In order to efficiently determine those $\vec{x}$ points that are visible from $\vec{y}$, the classical hemicube method can be used [CG85]. We can note that the integral in equation (5.2) can be evaluated on the five window surfaces (W) that form a hemicube around the source $\vec{y}$ (figure 5.1). In the remaining part of this section, we re-derive the basic formulae to show that the hemicubes can also be used in cases when the reflection is non-diffuse.

To find formal expressions, let us express the solid angle $d\Omega_p$, in which a differential surface area dx is seen through pixel area dp, both from the surface area and from the pixel area:

$$d\Omega_p = \frac{dx \cdot \cos\theta'_{\vec{x}}}{|\vec{y} - \vec{x}|^2} = \frac{dp \cdot \cos\theta_p}{|\vec{y} - \vec{p}|^2}, \tag{5.4}$$

where θ_p is the angle between the direction pointing to $\vec{x}$ from $\vec{y}$ and the normal of the window (figure 5.1). The distance $|\vec{y} - \vec{p}|$ between pixel point $\vec{p}$ and the radiance source $\vec{y}$ equals to $f/\cos\theta_p$ where f is the distance from $\vec{y}$ to the window plane, that is also called the focal distance. Using this and equation (5.4), differential area dx can be expressed and substituted into equation (5.2), thus we can obtain:

$$L(m)|_i = \frac{A_j}{p_j A_i f^2} \cdot \int_W V(\vec{y}, \vec{x}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{p}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \cos\theta_{\vec{y}} \cdot \cos^3\theta_p \; dp.$$

Let P_i be the set of those pixels in which patch i is visible from $\vec{y}$. P_i is computed by running a z-buffer/constant shading rendering step for each sides of the window surface, assuming that the color of patch i is i, then reading back the "images". The reflected radiance on patch i is approximated by a discrete sum as follows:

$$L(m)|_i \approx \frac{A_j \delta P}{p_j A_i f^2} \cdot \sum_{P_i} L(\vec{y}, \omega'_{\vec{y}\to\vec{p}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}(\vec{p}), \omega) \cdot \cos\theta_{\vec{y}} \cdot \cos\theta_p^3, \tag{5.5}$$

where δP is the area of a single pixel in the image. If R is the resolution of the image — i.e. the top of the hemicube contains $R \times R$ pixels, while the side faces contain $R \times R/2$ pixels – then $\delta P = 4f^2/R^2$.

5.2 Randomization of the hemicube

It is easy to see that when computing the point-to-patch form factors by hemicubes, sometimes it is not efficient to render the scene across all sides of the hemicube. This is the situation when the power distributed through a specific side of the hemicube is negligible. This can be caused by the low incoming radiance or by the anti-symmetry of the outgoing radiance. The main cause of this anti-symmetry is the specular characteristic of the surface, which transfers most of the radiance towards the ideal reflection direction. If the ratio of specular albedo and diffuse albedo and the *shine* parameter of the Phong illumination formula is large, most of the light power is transferred through that specific side of the hemicube, which lies in the direction of the ideal reflection. In these cases it is useless to use the other sides of the hemicube.

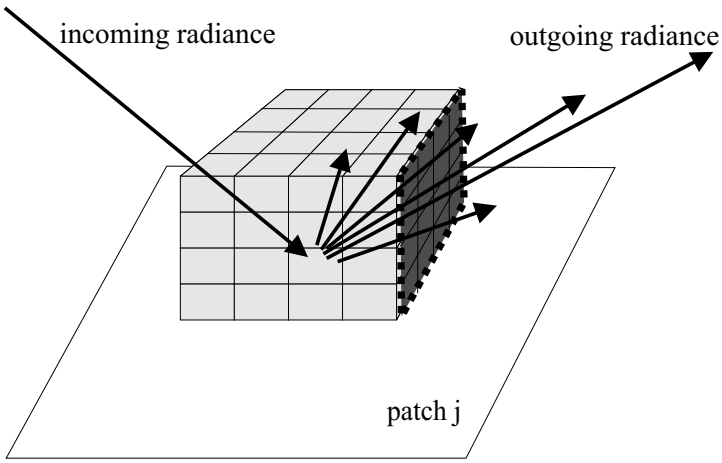


Figure 5.2: On specular surfaces most of the illumination goes through just 1 side of the hemicube.

Since we do not want to loose the unbiasedness characteristics of the method, but we want to save computation time when it is possible, we introduce randomization into the process. When the selected surface has quite strong specular characteristics, we randomly select one side of the hemicube and propagate the radiance through just the selected side. In that cases, according to Russian roulette, the transported radiance is divided by the selection probability.

5.3 Importance sampling

The stochastic iteration will converge quickly if the random noise added by a single iteration step is small. This means that the randomization of the light transfer should not be too strong (the optimal level is determined by the efficiency of transferring the radiance by the random operator and the variance caused by the randomization).

The variance of the random transfer can be decreased by the variance reduction techniques of the Monte-Carlo literature, and particularly by importance sampling. According to importance sampling, the selection probability is good if it mimics the original integrand, thus the total transferred power, which is computed as the real power divided by the selection probability, should be roughly constant. This means that the patches are worth selecting with a probability that is proportional to their output power, and the sides of the hemicube according to the power transferred through them.

Recall that before a given iteration step, we store the emission and the incoming radiance of each patch. If a patch is selected, then it will shoot the following power:

$$\Phi_j = \Phi_j^e + A_j \cdot \cos\theta' \cdot I(\omega'_{m-1}) \cdot a(\omega'_{m-1}),$$

where Φ_j^e is the emission power and $a(\omega'_{m-1})$ is the local albedo. Thus in each iteration step, power Φ_j is computed for each patch, and the patch selection is realized with $\Phi_j / \sum_k \Phi_k$ probability.

The second level of randomization controls the identification of those hemicube sides through which the transfer is computed. This should be proportional to the represented solid angle and the average radiance in the directions of the solid angle. We used a very simple heuristic scheme. If the surface is highly specular, the algorithm selects that side which is intersected by the ideal reflection direction by 0.6 probability and all the other sides by 0.1 probability. If the ratio of the specular and diffuse albedos is smaller then 1, more than half of the power is distributed by diffuse light transfer. Since there is a cosine term in the transferred radiance formula, in these cases most of the radiance goes through the top of the hemicube. Thus it is worth selecting the top deterministically and using one from the four sides randomly.

5.4 Variance reduction by trading bias with noise

If we implement and run the algorithm described so far, we can realize that the general convergence of the image will be very fast, but embarrassing noise occurs at corners and at object boundaries (figure 5.3). This problem is mentioned by the Monte-Carlo radiosity literature, but so far it has not been solved.

Figure 5.3: An office scene rendered without the biased variance reduction.

The explanation of these irritating artifacts is the following. Darker patches are very seldomly selected by the algorithm. However, when they are selected, the transferred power is divided by the small probability, thus even dark patches can result in large power transfers. When the receiver patches are very close to the shooting patch, then the point-to-patch form factor is large, thus the receiving patch gets too much power and tends to be too bright in the image. As stochastic iteration proceeds this annoying artifact slowly disappears. Theoretically there is nothing bad or unusual with these too bright patches, this behavior is caused by the random noise, which is inherent in all Monte-Carlo methods. As the iteration number goes to infinity, the radiance values will converge to the mean of these random variables.

However, when we want to have accurate and nice images quickly, i.e. after just a few hundred iterations, the fact that the irritating bright patches would disappear if we were running the algorithm for much longer time, is not acceptable. Fortunately a solution exists that can successfully attack this problem, which trades bias for noise in a way that for a given iteration number the result will not be unbiased, but the total

error of the bias and the Monte-Carlo noise will be still smaller than the Monte-Carlo noise of the original algorithm.

Let us return to the approximation of the radiance of patch i after an iteration step (equation (5.3)):

$$L(m)|_i \approx \frac{L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot F_{A_i\to A_j} \cdot \rho(\vec{y}\to\vec{x})}{p_j}.$$

The expected value of this random variable is

$$E[L(m)|_i] \approx E[L(\vec{y}, \omega'_{\vec{y}\to x_j})] \cdot F_{A_i\to A_j} \cdot \rho(\vec{y}\to\vec{x}).$$

If M iteration steps are computed altogether, then the probability of selecting patch j for shooting at least once is $1-(1-p_j)^M$. If p_j is really small, because patch j is not a light source, its size is also small and it does not receive significant illumination, then the selection probability is reasonably smaller than one and can be approximated as

$$1-(1-p_j)^M \approx p_j M.$$

The fact that these patches are not selected at all is not a problem in itself since according to the fundamental assumption of hierarchical and Monte-Carlo methods, patches form homogeneous groups and using one patch in these groups can also simulate the radiance transfer of other elements of the group. Suppose that the selected patch is a member of such a homogeneous group consisting of k similar patches. Then the probability of selecting at least one member of this group is $1-(1-kp_j)^M$. The real problem happens when even this group selection probability is much smaller than one. In this case, this probability is roughly kp_jM. Statistically, such a group should not be used for radiance transfer in the M step long iteration, but the random selection might find elements also in this group. If this patch group is selected $n>0$ times, the random estimator is:

$$\hat{L}(m)|_i = \tilde{L}(\vec{y}, \omega'_{\vec{y}\to A_j}) \cdot F_{\vec{y}\to A_i} \cdot \rho(\vec{y}\to\vec{x}) \cdot \frac{n}{kp_jM},$$

where $\tilde{L}$ is the average of the radiances in these transfers. The distance of this estimator from the expected value can be bigger than the distance between the expected value and zero, when it is worth replacing the transferred radiance by zero. If we assume that the average radiance $\tilde{L}$ is approximately equal to its expected value, then the criterion of replacing the transfer by zero is:

$$\hat{L}(m)|_i - E[L(m)|_i] > E[L(m)|_i] - 0 \Longrightarrow p_j < \frac{1}{2kM}.$$

In order to find an upper-bound for k, notice that the geometric parameters of the transfer are characterized by form factor $F_{\vec{y}\to A_i}$. Different patches behave similarly in

this transfer if their respective form factors are also similar. The sum of form factors is at most one (exactly 1 in closed and less than 1 in open scenes), that is

$$\sum_j F_{A_i \to A_j} \leq 1.$$

It means that the number of patches that have roughly this form factor with patch i is bounded by the inverse of the form factor, thus for k we obtain:

$$k \leq \frac{1}{F_{A_i \to A_j}}.$$

This allows to establish the limit of probability where the radiosity transfer is not worth executing:

$$p_j < \frac{F_{A_i \to A_j}}{2M}. \tag{5.6}$$

The modified algorithm works similarly as the previous one, it selects patches randomly and computes the radiance transfer from the selected patch towards those patches that are visible from here. In order to compute the new radiance value of the receivers, the form-factor is also computed. However, when it turns out that selection probability of the shooting patch is smaller as the limit of inequality 5.6, then this particular receiving patch is assumed to get zero radiance in this iteration step.

Note that this trick steals energy from the system, thus for a fixed iteration number M the result will be biased. However, the error is still less than in the unbiased estimate. On the other hand, the bias disappears as M goes to infinity, thus the method is still unbiased in the asymptotic case.

For very small M values, the missing energy becomes noticeable at the corners since they are darker than expected. Although this is still much better than the too bright patches, this problem can be further reduced by a special type of mean value substitution. When it turns out that the random estimator of the current transfer is too large, then instead of replacing it by zero, it can be replaced by its approximated mean $L(\vec{y}, \omega'_{\vec{y} \to A_j}) \cdot F_{\vec{y} \to A_i} \cdot \rho(\vec{y} \to \vec{x})$. This is as accurate as the radiance L is close to its converged value.

5.5 Application of the constant radiance term

Usually just a fraction of the patches belong to light sources. Importance sampling on the other hand will probably select the shooting patches from the light sources. One alternative for making it better is the first-shot technique, but since it selects the center of the light source patches deterministically, it introduces bias.

Anyway, the hemicube shooting by nature is very good at performing first-shot, so doing a first-shot before starting the algorithm seems unnecessary. On the other hand, the selection probability of the non light source patches can be significantly

improved by transforming the radiance function to a function with smaller amplitude [NNPP98].

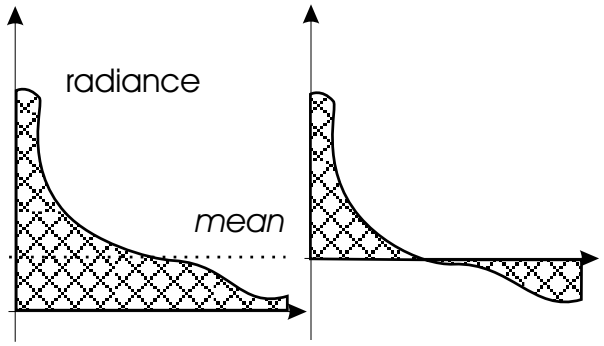


Figure 5.4: When subtracting the mean, the importance sampling more probably selects the non light sources.

A constant radiance value is extracted from the solution in every surface point and direction. However, we should be careful, when choosing this constant value. The optimal constant value is hard to find for each patch separately, but even a conservative estimate can improve the convergence. The average radiance $\overline{L}$ can be computed by the assumption that all points have the same BRDF and albedo, that are computed as the average [NNPP98].

The average radiance is determined by:

$$\overline{L} = \frac{\int_{\Omega}\int_{S} L^e(\vec{x},\omega)\cos\theta d\vec{x}d\omega}{S\pi(1-a_{mean})},$$

where a_{mean} is the average albedo of the scene, calculated by:

$$a_{mean} = \frac{1}{S\pi}\cdot\int_{\Omega}\int_{S} a(\vec{x},\omega)\cos\theta d\vec{x}d\omega.$$

Formally, let us decompose the radiance function into this average $\overline{L}$ and a distance from the average ΔL. Substituting $L = \overline{L} + \Delta L$ into the rendering equation, we can obtain:

$$\Delta L(\vec{x},\omega) = L^e(\vec{x},\omega) + (a(\vec{x},\omega)-1)\cdot\overline{L} + \mathcal{T}\Delta L.$$

Note that we obtained a rendering equation for the ΔL term, having modified the emission function with $(a(\vec{x},\omega)-1)\overline{L}$. This new light source term is negative for physically valid scenes, which means that the non light source patches emit negative power. Therefore, when doing importance sampling we should use the absolute values of power for computing the selection probability of the patches. After the iteration

finishes and $\Delta L(\vec{x}, \omega)$ is obtained, we should add the average radiance $\overline{L}$ to the final result.

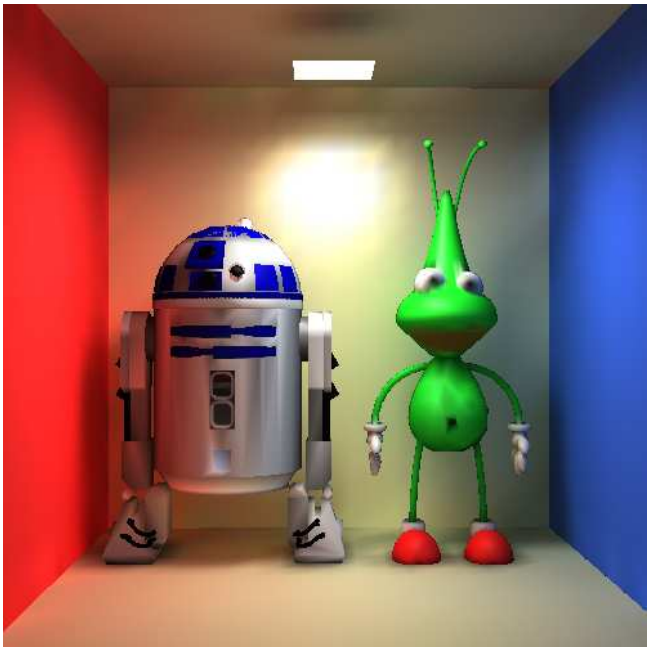


Figure 5.5: R2D2 meets an alien in the Cornell-box. The image was rendered with the new method using the biased variance reduction and the constant radiance step.

5.6 Results

The presented algorithm has been implemented in C++ in OpenGL environment. The images have been rendered with 500×500 resolution. The faces of the hemicube had 600×600 pixels. The algorithm can render moderately complex scenes within a minute.

We tried the hemicube randomization with scenes of specular reflectance, and according to our experience the hemicube randomization resulted in 5-15 percent speed-up. Without using the constant radiance step, importance sampling selected the light sources very frequently, i.e. in about 50 percents of iterations. When applying the constant radiance step, this decreased to 32 percent. This trick increased the speed by another ten percent.

When the Cornell box scene with R2D2 and the alien (figure 5.5) was rendered, the 500 iterations needed 50 seconds on a Pentium®III 1Ghz computer using GeForce2 MX graphics hardware. Note the specular highlight on the back wall and on the body of R2D2. The specular albedos were set to 0.27 and 0.2 and the shine parameters

Figure 5.6: An office scene rendered with the new method using biased variance reduction.

of the Phong BRDF to 28 and 10, respectively. Other surfaces also have specular albedos, usually in the range of $0.05 - 0.0$. Rendering the office room (figure 5.6), which contains a specular vase, took about 60 seconds and needed 600 iterations.

5.7 Summary

In this chapter we presented a new stochastic rendering technique that is based on the randomization of the classical hemicube approach. This randomization speeds up the convergence of the algorithm and ensures rendering non-diffuse scenes with the same storage space as required by the radiosity method. In order to get rid of the artifacts of the randomization, we proposed trading of noise with bias in a way that the error gets smaller, but the algorithm is still unbiased asymptotically. We also discussed some improvements for the basic algorithm, which included the random selection of the hemicube faces and the application of the constant radiance step.

Chapter 6

The combination of ray-bundle based strategies

Global illumination algorithms can be classified as local and global transfer methods. Local methods find a single point (or patch) in a given step and transfer its radiance towards other point(s). Global methods, on the other hand, select the source and the target of the transfer simultaneously. In the context of stochastic iteration, parallel ray-bundle iteration belongs to the global group, and the perspective ray-bundle algorithm is a local method. In this chapter we propose the combination of global and local global illumination algorithms in the sense of multiple importance sampling. In this way, the combined method can eliminate the higher noise at the corners produced by local methods and the need for first-shot for global techniques. The modified multiple importance sampling and the combined algorithm are the own work of the author and concerns to thesis 4 (see chapter 8).

6.1 Introduction

Local line methods find the starting point of the half-line first, then they obtain the direction of the line, which will identify the intersection point or the other point of the transfer. An alternative is the global line approach which samples the two points simultaneously. Local methods are better if the radiance distribution is heterogeneous and the scene is sparse, while global methods can win for dense scenes of homogeneous radiance. There have been many discussions about the comparative advantages of these algorithms, but no method can be claimed to be the best. This is not surprising since each method has advantages and disadvantages in certain situations. Thus instead of insisting to a given technique, it is worth combining several of them, in a way that the advantages are preserved. Such quasi-optimal combination of Monte-Carlo sampling techniques is offered by multiple importance sampling [Vea97]. Assume that integral $L = \int_{\mathcal{P}} l(z)dz$ needs to be evaluated. Monte-Carlo quadratures generate

samples with certain probability density. Suppose that we have N different sampling techniques. Sampling method i uses probability density $p_i(z)$, thus the primary estimator of this method is $l(z)/p_i(z)$. Assume also that with method i we obtain N_i samples $z_{i1}, \dots z_{iN_i}$. The combined estimator is computed from the samples of all sampling techniques, applying appropriate weighting functions $w_i(z)$, and summing the results:

$$\langle L \rangle_c = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(z_{ij}) \cdot \frac{l(z_{ij})}{p_i(z_{ij})} = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \frac{l(z_{ij})}{d_i(z_{ij})} \tag{6.1}$$

where the divider is

$$d_i(z) = \frac{N_i p_i(z)}{w_i(z)}.$$

The combined estimator is unbiased, i.e. the expected value of this estimator gives back the original integral, if for all z values $\sum_{i=1}^{N} w_i(z) = 1$. In order to find an optimal weighting, the variance of the combined estimator $\langle L \rangle_c$ should be minimized by setting the weights appropriately and also taking into account the constraint of unbiasedness. Unfortunately, this optimization problem cannot be solved analytically, but different quasi-optimal solutions can be obtained. One such approximate solution is called the *balance heuristic* [VG95]:

$$w_i(z) = \frac{N_i p_i(z)}{\sum_{k=1}^{N} N_k p_k(z)}. \tag{6.2}$$

Substituting these weights into equation (6.1), we can conclude that balance heuristic divides with the total density

$$d_i(z) = \sum_{k=1}^{N} N_k p_k(z) = d(z)$$

instead of the original densities $N_i p_i(z)$ of the individual methods.

6.2 Multiple importance sampling in iteration

Multiple importance sampling can be efficiently used in random walk algorithms that obtain samples independently [SW99]. However, the application of multiple importance sampling in iteration like algorithms requires further considerations. We could, for example, use all sampling techniques to obtain a tentative value in the next iteration step then find the real value as the weighted average of the results of the individual methods, but this method would slow down the progress of the iteration and thus the introduction of higher order terms. Thus we propose to randomly select just a single technique in each iteration step, compute just a single sample, and apply the other techniques to the already iterated value.

To consider the random selection formally, let us assume that the sample is computed with method i with probability P_i.

The modified estimator uses the indicator functions ξ_i, which are 1 if the respective method generates a sample:

$$\langle L \rangle_c = \sum_{i=1}^{N} w_i(z_i) \cdot \frac{l(z_i)}{p_i(z_i)} \cdot \xi_i. \tag{6.3}$$

The requirement of the unbiasedness becomes:

$$\sum_{i=1}^{N} P_i \cdot w_i(z) = 1.$$

The modified formulae of balanced heuristics is the following:

$$w_i(z) = \frac{p_i(z)}{\sum_{k=1}^{N} P_k \cdot p_k(z)}.$$

Thus when a sample is computed, its contribution is always divided by

$$d(z) = \sum_{k=1}^{N} P_k \cdot p_k(z)$$

no matter which sample strategy is used.

This general approach can be used for different global illumination algorithms. We have applied it to the combination of local and global ray-shooting as it was proposed in [P7]. In the context of this dissertation we present here the other combination, the mixture of parallel and perspective ray-bundle based transfers.

6.3 Combination of methods using ray-bundles

So far, we introduced three different random radiance transfer methods that use different sampling probabilities. Parallel ray-bundle tracing samples the direction from point $\vec{x}$ with a uniform density, i.e. the probability of generating a direction in $d\omega$ is

$$\frac{d\omega}{2\pi}.$$

Note that we use 2π due to the bi-directionality of the algorithm.

When just a single plane is used for parallel radiance transfer, contribution to point $\vec{x}$ is possible only if the plane is between point $\vec{x}$ and that point $\vec{y}$ which is visible from here. If the maximum size of the scene is R, then probability that a contributing direction is in $d\omega$ is

$$\frac{|\vec{x} - \vec{y}| \cdot d\omega}{2\pi R}.$$

For perspective ray-bundle shooting, the probability that shooting point is in differential area dy of patch j is

$$\frac{p_j \cdot dy}{A_j} = \frac{\Phi_j \cdot dy}{A_j \sum_i \Phi_i}.$$

Before applying the concept of multiple importance sampling, we have to solve the problem that different methods formulate the light transport problem with different integrals. Parallel ray-bundles use directional itegrals while perspective ray-bundle shooting applies surface integrals. According to the formula of differential solid angles

$$d\omega = \frac{dy \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2},$$

directional integrals can also be converted to surface integrals, thus the probability densities used by the discussed methods are the following:

$$p_1(\vec{y}) = \frac{\cos\theta_{\vec{y}}}{2\pi \cdot |\vec{x} - \vec{y}|^2},$$

$$p_2(\vec{y}) = \frac{\cos\theta_{\vec{y}}}{2\pi R \cdot |\vec{x} - \vec{y}|},$$

$$p_3(\vec{y}) = \frac{p_j}{A_j}.$$

Each of them is good for a particular part of the scene. The parallel ray-bundles are effective if the scene consists of patches of similar radiance, while the perspective ray bundles are effective if one or several patches are much brighter than the others (note that these bright points are selected with much higher probability by perspective ray-bundle shooting). Thus perspective ray-bundle shooting is the best method if the scene contains small light sources. It is thus highly intuitive why parallel ray-bundle algorithms always apply a first shot to distribute the illumination of the light sources, letting the algorithm compute only the indirect illumination.

On the other hand, the transfer of nearby points is better coped by parallel transfers then by perspective transfers. Close points are obtained by parallel ray-bundle tracing with the higher probability, this probability is smaller if just a single plane is used and the smallest for perspective ray-bundle shooting. Thus in homogeneous environment, corners (close patches) can be rendered in a better way by *parallel ray-bundle transfers using software z-buffer*, and radiance transfer between distant patches can be rendered better by *parallel ray-bundle transfers using hardware z-buffer*.

In order to obtain a method that does not require first shot and can nicely render corners and close objects, the presented techniques are combined according to multiple importance sampling.

Suppose that each of the three methods is used with probability P_1, P_2 and P_3, respectively. Since one method is applied in each step $P_1 + P_2 + P_3 = 1$. These

probabilities can be specified by the user, taking into account the features of the scene and the time cost of the application of the methods. A simple way is setting these parameters to be inversely proportional to their computation time.

The divider of balanced heuristic becomes:

$$d(\vec{y}) = P_1 \frac{\cos\theta_{\vec{y}}}{2\pi|\vec{x}-\vec{y}|^2} + P_2 \frac{\cos\theta_{\vec{y}}}{2\pi R|\vec{x}-\vec{y}|} + P_3 \frac{p_j}{A_j}.$$

When parallel ray-bundles are used, this weight should be multiplied by $dy/d\omega = |\vec{x}-\vec{y}|^2/\cos\theta_{\vec{y}}$ in order to replace surface points $\vec{y}$ by directions $d\omega$:

$$d(\vec{\omega}) = P_1 \frac{1}{2\pi} + P_2 \frac{|\vec{x}-\vec{y}|}{2\pi R} + P_3 \frac{p_j}{A_j} \frac{|\vec{x}-\vec{y}|^2}{\cos\theta_{\vec{y}}}.$$

Let us interpret these results. When a perspective ray-bundle transfers the light in the combined method, the integrand of the rendering equation,

$$v(\vec{x},\vec{y}) \cdot L(\vec{y},\vec{\omega}') \cdot f_r(\vec{\omega}',\vec{x},\vec{\omega}) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2}$$

is divided by $d(\vec{y})$ instead of its own sampling density $p_3(\vec{y})$. The integrand can be very large if the two points $\vec{x}$ and $\vec{y}$ are close, which is not compensated by the original density $p_3(\vec{y})$, resulting in high variance around the corners. However, thanks to parallel transfers, the combined density includes a similar $|\vec{x}-\vec{y}|^2$ factor, thus the corner spikes can be eliminated. On the other hand, when parallel bundles are used alone, variance is caused by the variation of the source radiance. This error is also reduced in the combined method, since we divide the transfer by $d(\vec{\omega})$, which includes the source radiance thanks to the probability density of perspective transfers.

The optimal selection of P_1, P_2 and P_3 depends on how homogeneous the radiance in the scene. It is worth setting the probability of perspective bundles high at the beginning of the algorithm and letting parallel ray-bundles refine the roughly distributed light energy.

Note that all the three methods require just partial information about the radiance function, parallel ray-bundle transfer needs the radiance values just in a single direction, perspective ray-bundle transfer requires the radiance distribution of a single patch only. The combined algorithm should be able to serve the needs of the underlying algorithms. Therefore in an iteration step let us thus compute only the irradiance on each patch, which is independent of the transfer direction of the next step. With the irradiance information we also store the incoming direction. This determines the memory requirement of the combined algorithm, which is 2 variables per patch. In the next iteration step, when the output radiance of a patch in a given direction is needed, it is obtained on the fly, multiplying the irradiance by the BRDF of the patch taking into account the previous and current directions.

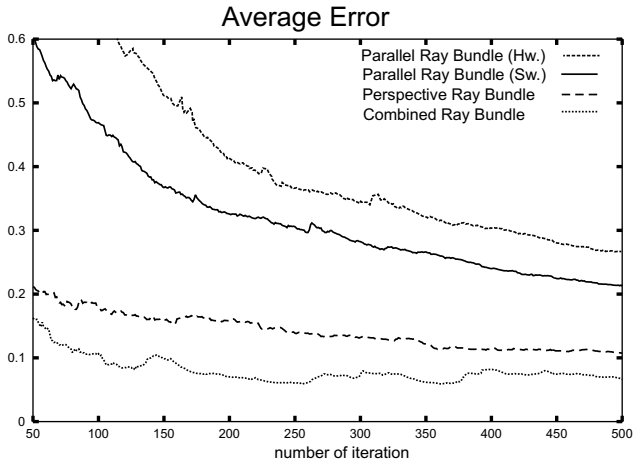


Figure 6.1: Error of the software z-buffer parallel, hardware z-buffer parallel, perspective and combined ray-bundle shooting algorithms for the Cornell box

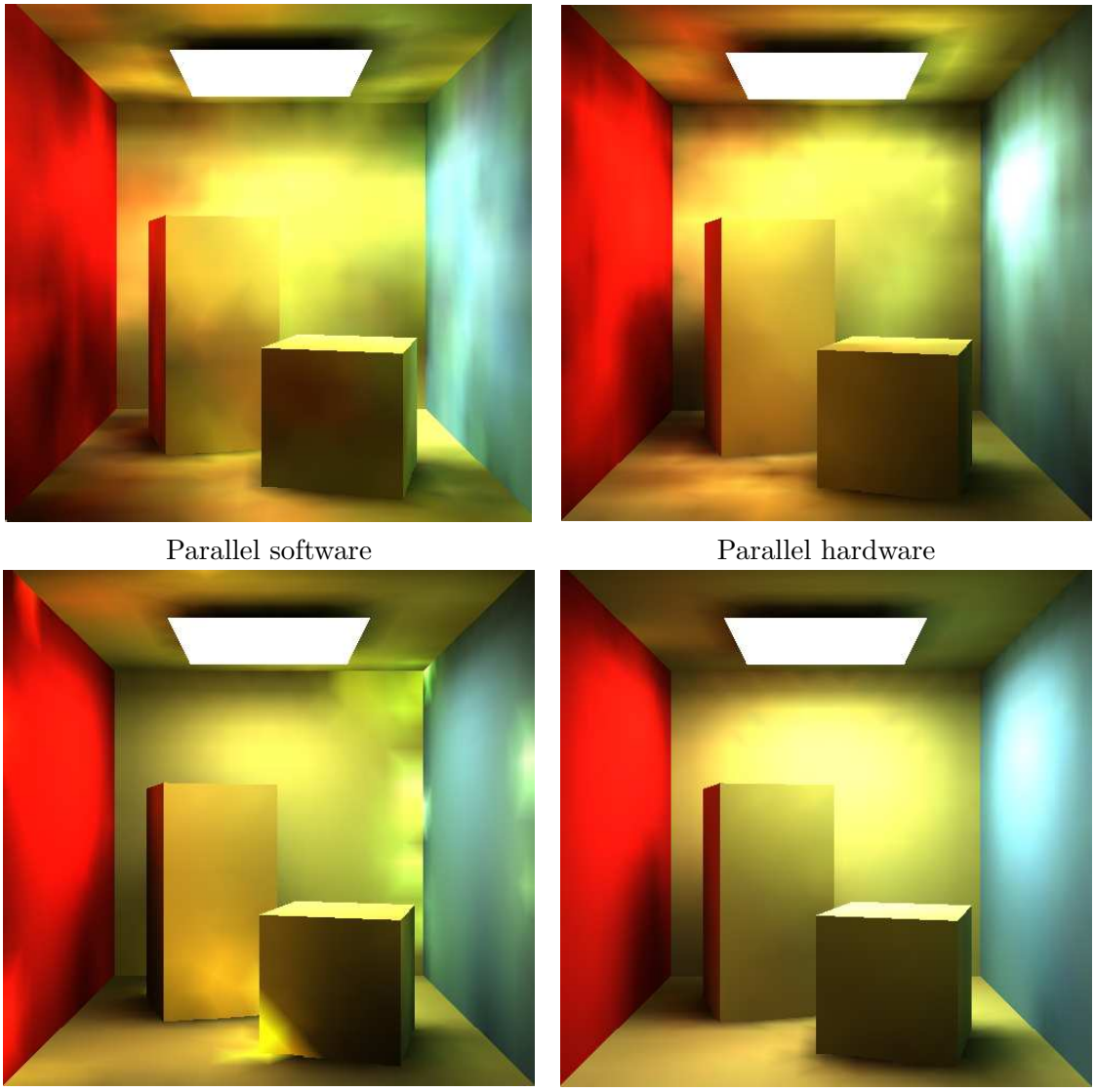
6.4 Results

In order to test the proposed method we have selected the standard Cornell Box scene. The images have been rendered with 500 x 500 resolution. The transillumination buffer contained 1000 x 1000 pixels. Figure 6.2 shows the Cornell box rendered by parallel ray bundles with software z-buffer and with the hardware z-buffer method, by perspective ray bundles and by the combined method. The computation time was 7 seconds in all cases. Note that parallel bundles distribute the energy with higher noise generally, but are good in rendering corners. The result of perspective bundles is better except for the annoying spikes at the corners. The combined method preserves the advantages of each technique and results is much more pleasing image than its parents.

In figure 6.1 we have plot the average error for the 4 algorithms. As it is expected, the combined method gives the lowest error.

6.5 Summary

We proposed the modification of multiple importance sampling, which is a technique for combining different sampling algorithms. We have applied our mathematical framework to ray bundle iteration methods. We proposed a stochastic iteration algorithm that dynamically combines three random radiance transport methods based on the actual radiance approximation. The method is able to render complex glossy scenes in about a minute and is particularly effective if the surfaces are not highly specular.



Parallel software | Parallel hardware

Perspective | Combined

Figure 6.2: Comparison of stochastic iteration using parallel ray bundles with software z-buffer, parallel ray-bundles with hardware z-buffer, perspective ray-bundles and the combination of the three methods respectively using the same computation time (7 seconds on a Pentium®4/1.4GHz computer)

Chapter 7

Animation

This chapter proposes a non-diffuse global illumination algorithm that is fast enough to be appropriate for interactive walkthroughs and general animations. To meet the severe performance requirements, we heavily exploit coherence both in time and space, and use randomization to reduce the time and storage complexity. To speed up convergence and to support animation, the approximation of the radiance is stored in the object space as well. However, in order to reduce the high memory requirements of such representations and to reduce finite-element artifacts, we use just a random approximation, which fluctuates around the real radiance function. The direction dependent radiance approximation is represented in a compact way, by four random variables per patch. The key of performance is then to make the error – i.e. the variance of this compact approximation – as small as possible. In addition to main part separation, we apply a novel sampling scheme inspired by the Metropolis method to achieve this goal. In this algorithm light transfers are computed by both local and global methods using ray bundles and with the support of the graphics hardware. The animation algorithm with the special iteration phase concept concerns to thesis 5 (see chapter 8).

7.1 Introduction

Of late years there was significant advancement in animation techniques. The movie and game industry requires more believable human motion and photo-realistic facial animation techniques [SK03, RN00, BW97]. However, this dissertation is concerned for animating architectural models which pose other problems that do not appear in human animation.

The recent growth of computing power and available memory made it possible to use global illumination algorithms for complex scenes even on PC class computers. It is usual that these rendering systems render a complex scene with high quality in a couple of minutes. They are designed for rendering static scenes. If they are applied for dynamic scenes, every illumination computation has to be repeated for

each frame, which makes them impractical for animation sequences.

However, this approach is not only impractical, but results in typical errors like *temporal aliasing*. Stochastic methods suffer from noise both in the spatial and in the temporal dimensions. The temporal anti-aliasing tries to reduce the little differences in lighting reconstruction which occurs in the successive frames. If it is not considered, the result is unpleasant flickering and shimmering, which can be perceived easily by the human observer.

Recently, new algorithms have been proposed, which tries to remedy these problems. These dynamic algorithms can be classified as:

- **Offline global illumination animation:**

  These methods are intended for final production of high-quality animations, where the dynamic behavior of the scene is known a-priori. Usually an animation script is given that contains this information. The objective here is to exploit the temporal coherence in order to reduce the vast computation need for rendering e.g. 300 frames for a 10 second long TV-advertisement. Other goal of these algorithms is the reduction of temporal aliasing. By exploiting the coherence between successive frames, they typically increase the speed of the rendering by 5-10 times, compared with the blind algorithm, which renders all frames from the scratch. Notice that this speed increase is still far enough from the need of real-time applications.

- **Interactive global illumination animation:**

  These methods are designed for speed. They usually make simplifications and give up to achieve good quality of illumination reconstruction. They have to provide fast response for changes in the scene properties.

7.1.1 Offline global illumination animation

It is straightforward that interactive algorithms can be used for offline rendering; however, the same is not true in the opposite direction. In the context of this dissertation we are interested only in interactive global illumination, thus we omit to discuss the offline methods. We just mention the most important approaches, which are the *space-time hierarchical radiosity* by Damez [DS99] [DSH01], the *multi frame lighting method* by Besuievsky [BP01a] [BS96], the *time dependent photon mapping* by Cammarano [CJ02], the image based methods for animation by Nimeroff [NDR96b], two perception-guided animation rendering (the *Animation Quality Metric* (AQM) by Myszkowski [KM99] and the *visual attention modeling* by Yee [Yee00] and *frame-to-frame coherent animation with two pass radiosity* by Martin et al. [MPT99].

7.1.2 Interactive global illumination animation

The interactive global illumination animation is an area of intensive research, and as we are intending to show that in this introduction, – in spite of great progress has been made recently – this problem has not been fully solved yet. The first initiatives for interactivity were connected the radiosity algorithm. The *incremental progressive radiosity* was published by Chen [Che90] [XSG00] and George [GSG90] independently. The algorithm selects the dynamic objects (one after the other) and shoots the energy of this object toward all other static objects. Then it selects one static object and shoots the energy towards the dynamic objects, and shoots the negative energy to surfaces which are now in the shadow volume of the dynamic object. The selection of static objects is repeated until the approximated error decreases under a certain threshold.

However, progressive radiosity is slightly outdated and excelled by hierarchical radiosity. The first *dynamic hierarchical radiosity* was proposed by Hanrahan [HSA91]. One problem with his approach is the lack of control the trade-off between accuracy and speed.

The dynamic hierarchical radiosity was later extended by shaft that represents a set of lines between two elements in the hierarchy. Therefore this algorithm – which was published by Drettakis [DS97] – uses *line-space hierarchy for hierarchical radiosity.*

However, two main problems appear when using interactive radiosity solutions. The first is that it can never result in a full global illumination solution, since they are targeted only for calculating the diffuse radiosity. Other problem is that they do not scale well with the complexity of the scene, and therefore they are restricted to very simple scenes. The papers listed before usually present examples with using only a few hundred polygons.

The glossy illumination is also covered by the algorithm presented by Granier [GDW00], where the diffuse part of the illumination is handled by classical hierarchical radiosity and the glossy part is computed by stochastic particle shooting. They report 3–5 seconds per frame, which is hardly qualify that the *unified hierarchical algorithm* can be interactive. However, for calculating the first frame of the animation is also slow (according to their proposal, 35 minutes is spend for the initial frame).

The *hybrid hardware radiosity and ray tracing* was proposed by Udeshi and Hansen [UH99] uses the video card hardware for direct illumination. The shadow computation is also done in hardware by the shadow volume algorithm. The implementation uses 64 processors and 8 graphics pipelines for moderate complexity scenes. The ray tracing part of the algorithm involves calculating the color of that pixel in which specular or refractive objects are seen.

The *instant radiosity* published by Keller [Kel97a] uses at first a photon tracing stage, which approximates the diffuse radiance in the scene (it is very similar to the photon map method of Jensen). The resulted photon hits become virtual light sources,

which can be used in the rendering pass as OpenGL point light sources. Thus, this algorithm can exploit the graphics hardware. If 40–100 point light sources is selected the resulting image is quite free from shadowing artifacts. Considering that OpenGL can use maximum 8 light sources, the hardware rendering is also broken down into phases. The result of rendering phases is piled up in the accumulation buffer of the graphics card. However, the active virtual light sources must be selected with care, since in the neighborhood of a virtual light source the illumination of the surfaces tends to be too bright. This is an annoying side effect of this method. Other problems concern dynamic environments. If the objects or a light source changes its position, the approximation of the diffuse radiance by photon hits becomes invalid. However, the recomputation of the light distribution is not fast.

Recently, a very promising approach was presented by Dmitriev. The *selective photon tracing* [DBMS02] is similar to the hybrid hardware radiosity and ray tracing method in the calculation of the direct illumination. Both of them use graphics hardware for rendering direct light and use shadow volumes for shadow calculation. The indirect lighting is handled by a quasi-random photon tracing, where density estimation is used for calculating the radiance at the vertices of the mesh. This information can be used later by Gouraud or Phong illumination calculation that is implemented on the graphics card. As before, in dynamic scenes, the recomputation of the illumination at the vertices must be done from the scratch, which takes for a while.

We point out that the hardware calculation of direct illumination can use only directional and point light sources, which is hardly enough for simulating the most common area light sources that is the main criterion for global illumination algorithms.

The *interactive global illumination using a distributed ray-tracing engine* was proposed by Wald [IWS02] and it exploits the fact that the ray-tracing can be parallelized quite easily. Using SIMD[1] CPU instructions, which can process 4 computations at a time on a cluster of 16 PCs (Athlon 1800+) they report 1.5 fps and 1.84 fps for quite complex scenes. The method scales well with the complexity of the scene, since ray-tracing with space partitioning is an $O(log_n)$ algorithm (after $O(nlog_n)$ preprocessing), where n is the number of surfaces in the scene. Similarly as in the instant radiosity a coarse representation of the radiance is generated by shooting particles in the first pass. These particles are then treated as virtual light sources. The caustics are treated separately by a method very similar to the caustics map used in Jensen's photon map algorithm. The rendering pass uses ray-tracing, which is fast concerning that the computation is distributed on a local network of PCs. Notice that the same problems occurs as it was discussed by the preceding algorithms (e.g. selection of the virtual light sources, dynamic environments).

[1]SIMD: Single Instruction Multiple Data

Other approaches that are important to mention are not really global illumination algorithms on their own. They are intended to complement the rendering by specifying what part of illumination needs to be recomputed by the global illumination algorithm in the successive frame. With the help of these methods it is always possible to generate fast feedback for the user by invalidating less portion of the current approximation of the illumination. Naturally, this produces less accurate image, but can result real interactivity. If the underlying global illumination supports it, they can control the trade-off between speed and accuracy. These algorithms fall into two categories.

The *reconstruction* can be made *in image space* as it was proposed by Walter [BW99] [WDG02]. They generate images by reprojecting samples stored in a container called *Render Cache*. As the dynamic environment changes, the samples stored in the Render Cache age, and based on the age of them a gray-scale image, the priority image is generated. Using this image, new samples are requested from the underlying global illumination algorithm. Notice that it is worth requesting those samples which are most likely to change in the next frame. However, these approach cannot compensate the drastic changes on the image plane. For example when the camera changes rapidly, all samples (the full image) become invalid. If it is desirable to maintain interactivity, the assembly of the image have to use both invalid and valid samples from the Render Cache, which distorts the final image.

The *reconstruction* can be also made *in object space* similarly as in image space. In the algorithm presented by Tole [TPWG02] the Render Cache is replaced by the *Shading Cache*, which stores samples of shading information for patch vertices in the 3D mesh. In this method, not points, but patches are selected for update. This suggests that this method is suited for diffuse lighting only, however view dependent specular characteristics of the surfaces can be also treated with a special care. According to the authors the Shading Cache needs only 10 percent of that samples that is required by using the Render Cache.

7.1.3 Discussion

Animations can be classified by the changes that could happen in the dynamic environment, which can be:

- **Camera** – camera parameters can change, which includes the position, the orientation and the viewing angle.
- **Geometry** – objects can change their position and orientation.
- **Light conditions** – point and area light sources can change their position orientation and goniometric properties.

- **Material** – the surface attributes can change, which includes modifying the BRDF or BSDF (e.g. changing the color, the refractive index or the texture generated by e.g. procedural textures)

The *camera animations*, also called walkthroughs [BP95], correspond those cases when only the camera moves. We refer the animation as *general animation* when even objects are allowed to change their properties.

Walkthroughs are simpler to compute since if we had the radiance function, they would only require to identify the points visible from the new eye position and to obtain their radiance. However, the radiance is also a function of the viewing direction if the surfaces are non-diffuse, thus the explicit representation of this radiance function is usually not feasible [BS96]. General animations are even more difficult to render, since all properties, even light source intensity may change in time. In architectural walkthroughs, images should be generated usually close to real-time, but when we stop to look at small details, we have the time to wait for more accurate images.

Making global illumination fast enough to be appropriate for walkthrough and general animations is one of the most important challenges of rendering. To reach this goal, we can either try to increase the computation speed to a level that rendering from scratch takes just a fraction of a second, or we may exploit not only object space and view space coherence [CLSS97, Chr00] but also time coherence, and recompute only those parts of the illumination, which became invalid [Che90, DS97, TPWG02, BS96, BP01b].

Taking into account the enormous computation required by the global illumination solution, the first approach is feasible only if we have huge computational power provided by a parallel system and/or we use simplifications [WBS02, FSZ98]. On the other hand, coherence allows interactive rendering even on a single computer. This chapter proposes an algorithm that falls into this second category.

Coherence methods make the errors correlated. Sometimes it is an advantage since it can reduce dot noise and flickering. However, coherence can also have disadvantages, and can result in artifacts such as light leaks, for example. Due to time coherence the highlights and shadows may follow the movement of the objects with a noticeable delay. Such problems should be avoided by the smart application of the coherence, such as by good quality or adaptively subdivided meshes, continuous directional functions [SP94a] and by elegant heuristic strategies to locate discontinuities [WBS02]. Concerning the problems of time coherence, we need a mechanism that quickly updates the changed illumination.

If we use random walks to transfer the light, we face the problem of slow convergence and of the task to figure out which walks are affected by object movements. A brute force approach would regenerate all paths from scratch as it was presented by Wald [WBS02]. Alternatively, pioneer paths can also be selected to find the changes. Then the algorithm should recompute only those walks which are close to the pioneer paths reporting changes as in selective photon tracing [DBMS02].

Adapting the solution to the continuously evolving environment is somehow natural in iteration approaches [Bek99, SK99b]. In iteration the solution of the previous frame is supposed to be the initial value of the iteration, which will converge to the required solution with the speed of a geometric series. Considering the better initial convergence and the view independence, iteration seems to be the better alternative for animation than random walk methods. However, we should pay a high price for this remarkable convergence in terms of storage space, which becomes really prohibitive if the surfaces are non-diffuse, not to mention the visible artifacts of finite-element approximations. These were one of the main problems in the radiosity based interactive global illumination described before.

To attack the problems, we propose an iteration algorithm with a novel random radiance approximation scheme that uses finite-element decomposition just in the spatial domain. The directional variation of the radiance is represented randomly, which requires just a few variables per patch, but provides a low variance estimate. Thus the proposed method is mesh based with continuous (not finite-element) but random directional radiance representation. Due to the low variance random representation, the convergence rate of iteration is preserved in the initial phase of the computation, and just the smaller random variations should be eliminated by the slower Monte-Carlo quadrature computing the radiance only for the view directions of the patches. Even the geometric convergence is too slow at parts of the scene where the radiance changes considerably during an animation sequence. Thus in our approach we follow a combined strategy, which is basically an iteration, but when objects move, it switches itself to a special mode, which removes previous transfers that having become invalid and introduces new ones as fast as possible.

We use our combined ray-bundle iteration algorithm developed in chapter 6. As it was introduced before, the combined algorithm selects one method from the 3 ray-bundle strategies in each iteration cycle. These strategies are the parallel ray-bundle with software z-buffer, the parallel ray-bundle with hardware z-buffer and the perspective ray-bundle iteration. The optimal selection probabilities P_1, P_2 and P_3 (see section 6.3) depend on how homogeneous the radiance in the scene. It is worth setting the probability of perspective bundles high at the beginning of the algorithm and letting parallel ray-bundles refine the roughly distributed light energy. On the other hand, parallel ray-bundles force all patches to communicate, thus they can be efficiently used to detect changes during the animation.

7.2 Random representation of the radiance

The discussed methods sample the radiance function in each step and obtain a new function. The radiance is a four variate function and usually has high variation. Our goal is to avoid the complete representation of this function, because that would pose prohibitive memory requirements.

As it was presented in section 6.3, the combined ray-bundle algorithm stores the irradiance (i.e. the incoming radiance estimate multiplied by the cosine of the incoming angle) and the direction of the last light transfer on each patch. For those patches that are not hit by rays, the irradiance of this iteration step is zero. From irradiance I_n and incoming direction $\vec{\omega}_n^{in}$ of iteration step n, the reflected radiance of the patch in an arbitrary direction $\vec{\omega}$ can be obtained as

$$L_n^{rr}(\vec{\omega}) = I_n \cdot f_r(\vec{\omega}_n^{in}, \vec{\omega}).$$

Examining the $L_n^{rr}(\vec{\omega})$ sequence, we can note that it has a high fluctuation since its elements are zero or very small when the patch is not the target of a transfer or the incoming direction is not the preferred direction of the BRDF, but when it is lucky enough to be hit by rays from the preferred direction, then it gets a larger contribution.

The variance of the whole method can be reduced if the fluctuation of this sequence is decreased. The general idea is to replace sequence I_n by another sequence, which is smoother but still results in the correct reflected radiance when averages are calculated. We use a combination of two techniques. The first is based on the *main part separation* [P2] [Kel99]. The second technique – which was proposed in [P9] – applies random acceptance and rejection according to Metropolis Sampling [MRR+53]. We should note that Metropolis sampling is used differently than in the Metropolis Light Transport algorithm [VG97]. Instead of sampling light paths proportional to their carried luminance, our objective is to develop a random representation of the directional radiance, which fluctuates around the real radiance maintaining a constant luminance.

The first method separates the constant main part of the reflected radiance, which is replaced by its average. Let us store the directional average of the reflected radiance in variable L_n^d in each patch computed as

$$L_n^d = \frac{1}{n} \cdot \sum_{k=1}^{n} I_k \cdot \frac{a(\vec{\omega}_k^{in})}{\pi} = \frac{1}{n} \cdot I_n \cdot \frac{a(\vec{\omega}_n^{in})}{\pi} + \left(1 - \frac{1}{n}\right) \cdot L_{n-1}^d,$$

where $a(\vec{\omega})$ is the albedo of the material. Note that this main part is computed not only from the last transfer but from the average of all transfers happened so far. We could take advantage of the fact that the main part is independent of the outgoing direction, thus an average could be computed that is valid for all directions. Thus a better (i.e. lower variance) sequence of the reflected radiance is

$$L_n^{rr}(\vec{\omega}) = L_n^d + I_n \cdot \Delta f_r(\vec{\omega}_n^{in}, \vec{\omega}).$$

where Δf_r is the difference BRDF

$$\Delta f_r(\vec{\omega}_n^{in}, \vec{\omega}) = f_r(\vec{\omega}_n^{in}, \vec{\omega}) - \frac{a(\vec{\omega}_n^{in})}{\pi}.$$

This technique reduces the general fluctuation, but the variation of the transfers represented by the difference BRDF still remains high in the sequence. Unfortunately, we cannot use the same trick of averaging here, since this term does depend on the outgoing direction $\vec{\omega}$, which will change from iteration cycle to iteration cycle. Therefore, either a finite-element representation of the reflected radiance is needed, or we should store all incoming directions and irradiance values. Both approaches have prohibitive memory requirements.

The second variance reduction technique solves this problem without requiring additional variables. We shall still store a single incoming direction and irradiance per patch in addition to the main part, but the incoming direction and the irradiance will not necessarily come from the last transfer (figure 7.1).

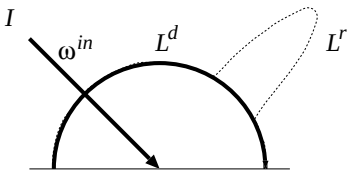


Figure 7.1: Random representation of the radiance

This method reduces the fluctuation by replacing a random sequence by another sequence of ,,similar samples". During the transformation zero samples are ignored, large samples of the original sequence will be scaled down and small samples will be scaled up. This transformation should not distort the expected values computed from the sequence, thus a scaled down value will appear more times in the new sequence. Denoting the albedo of the difference BRDF by $\Delta a(\vec{\omega}^{in})$, an appropriate scaling is:

$$\frac{I_n}{\mathcal{L}(I_n \Delta a(\vec{\omega}_n^{in}))} \cdot C_n, \quad \text{where} \quad C_n = \frac{1}{n} \cdot \sum_{k=1}^{n} \mathcal{L}(I_k \Delta a(\vec{\omega}_k^{in})),$$

since it makes the luminance of the reflected radiance estimates similar.

The average computed from the transformed sequence will be correct if we can guarantee that I_m is expected to appear $\mathcal{L}(I_m \Delta a(\vec{\omega}_m^{in}))/C_m$ times. A sampling scheme that can produce samples proportional to $\mathcal{L}(I_m \Delta a(\vec{\omega}_m^{in}))$ is based on random acceptance and rejection similar to Metropolis sampling [KBSK03]. At each iteration step the new irradiance I_n is compared with the stored irradiance I_m. If $\mathcal{L}(I_n \Delta a(\vec{\omega}_n^{in}))$ is greater or equal than $\mathcal{L}(I_m \Delta a(\vec{\omega}_m^{in}))$, then the new irradiance and its incoming direction will replace I_m and the stored incoming direction in the random representation of the radiance. However, when $\mathcal{L}(I_n \Delta a(\vec{\omega}_n^{in}))$ is smaller than $\mathcal{L}(I_m \Delta a(\vec{\omega}_m^{in}))$, the new irradiance is accepted randomly with probability $\mathcal{L}(I_n \Delta a(\vec{\omega}_n^{in}))/\mathcal{L}(I_m \Delta a(\vec{\omega}_m^{in}))$.

When combined with the separation of the main part, the improved sequence of reflected radiance estimates is

$$L_n^{rr}(\vec{\omega}) = L_n^d + \frac{I_m \cdot \Delta f_r(\vec{\omega}_m^{in}, \vec{\omega})}{\mathcal{L}(I_m \Delta a(\vec{\omega}_m^{in}))} \cdot C_n,$$

where I_m is the irradiance accepted most recently.

In order to establish importance sampling for perspective ray-bundles, the luminance of the patches should also be known. The computation of the powers from the irradiance values is also straightforward, the irradiances should be multiplied by the albedos $a(\vec{\omega}^{in})$ of the patches. The luminance of a patch of area A is

$$\mathcal{L}(\Phi) = \big(\mathcal{L}(L^e)\pi + \mathcal{L}(L^d)\pi + C_n\big) \cdot A.$$

Finally, we emphasize that only the main part converges, but sequence $L_n^{rr}(\vec{\omega})$ will fluctuate around the main part forever. However, this does not pose any problem since the image is obtained as the average of the image estimates of subsequent iteration steps. Thus Monte-Carlo integration happens in the image space, while we maintain a random, but low variance radiance estimate in the object space. This random radiance estimate speeds up the iteration and supports animation as well.

7.3 Radiance updates in walkthrough animation

We proposed a random representation of the object space radiance. Since these values, including main part radiance L^d, irradiance I, incoming direction $\vec{\omega}^{in}$ and scaling value C, are independent of the camera, and they remain valid when the camera moves. When the camera moves, the new visible radiance values of the patches are set to

$$L^{eye}(\vec{\omega}) = L^e(\vec{\omega}) + L^d + \frac{I \cdot \Delta f_r(\vec{\omega}^{in}, \vec{\omega})}{\mathcal{L}(I \Delta a(\vec{\omega}^{in}))} \cdot C.$$

This is a low variance estimator, thus even this initial value is quite close to the real visible radiance. Then, iterating further, a new image is computed as an average of the random estimates. Initial flickering can be reduced if the iteration is started from a weighted average of the previous and the new visible radiance values.

7.3.1 Results

Figure 7.2 shows two displayed images and a temporary result of a walkthrough animation. This scene consists of 27 thousand patches having both diffuse and specular reflections. The wardrobe, which is the most specular object in this scene, has the following material properties: $(0.3, 0.3, 0.4)$ diffuse albedo on the wavelengths of red, green and blue respectively, 0.45 wavelength independent specular albedo, and the shininess of the Phong-like BRDF is 28. The probabilities of the 2 parallel and perspective transfers were 0.15, 0.15 and 0.7, respectively. Note the slight difference between the middle image obtained after changing the camera and making a single iteration, and the right image taken after performing more iterations to get a converged image. This small difference shows that the proposed random radiance representation is quite accurate in glossy scenes. This approach allows 3 frames per

old camera position | after moving the camera | iterating further

Figure 7.2: The left and right images show two frames of a walkthrough animation. The middle image is not seen by the user, but demonstrates the effect of just changing the camera location but not allowing time for the iteration to adapt to the new situation. Note that the algorithm needed a few iterations to correct the highlights.

second walkthrough on a 2GHz Pentium®4 computer if only three iterations are performed in each frame. Since the errors in subsequent frames are highly correlated, the error due to the small iteration number is not noticeable for the user. Thanks to the spatial finite-element representation, there is no dot noise, and the frame rate is practically independent of the image resolution (we rendered the images at 800×800 resolution). When the user stops, the algorithm needs only one second to converge to the final image.

7.4 Radiance updates in general animation

In general animations objects may move and the emission of the light sources may change, which modifies the rendering equation. At the beginning of a frame the scene is represented by rendering equation $L = L^e + \mathcal{T}L$, and the approximation of its solution is available. Because of the changes of object properties, the new situation is described by a new light transfer operator $\mathcal{T}_{new}$ and a new emission function L^e_{new} in the next frame. The new radiance function L_{new} will be the solution of the updated rendering equation:

$$L_{new} = L^e_{new} + \mathcal{T}_{new} L_{new}.$$

Theoretically, we could continue the iteration with the new light transfer operator supposing the previous solution as the initial value, and the radiance will converge to the new solution. However, this is often not fast enough in animation sequences. Shadows may be visible in their old position for a few seconds. In order to avoid

this, when objects move, we switch to a special iteration mode to quickly correct the radiance where it changed significantly.

Let us denote the difference of the new and the old radiance functions by $\Delta L = L_{new} - L$. Subtracting the old version of the rendering equation from the new one, we obtain:

$$\Delta L = (L^e_{new} - L^e + \mathcal{T}_{new}L - \mathcal{T}L) + \mathcal{T}_{new}\Delta L.$$

We got an equation for ΔL, which is formally similar to the original rendering equation with the following light source term

$$L^{e*} = L^e_{new} - L^e + \mathcal{T}_{new}L - \mathcal{T}L.$$

It means that the same iteration algorithm can be continued to compute the change of the radiance function with this modified light source term. In order to work with the new light source term, the radiance transfer of each iteration cycle should be computed twice. First, placing objects at their original positions, the original radiance is transferred with negative sign (i.e. term $-\mathcal{T}L$ of L^{e*} is computed). Then, having moved the objects to their new positions, the new radiance is transferred with positive sign (i.e. $\mathcal{T}_{new}L + \mathcal{T}_{new}\Delta L = \mathcal{T}_{new}(L + \Delta L)$ is calculated).

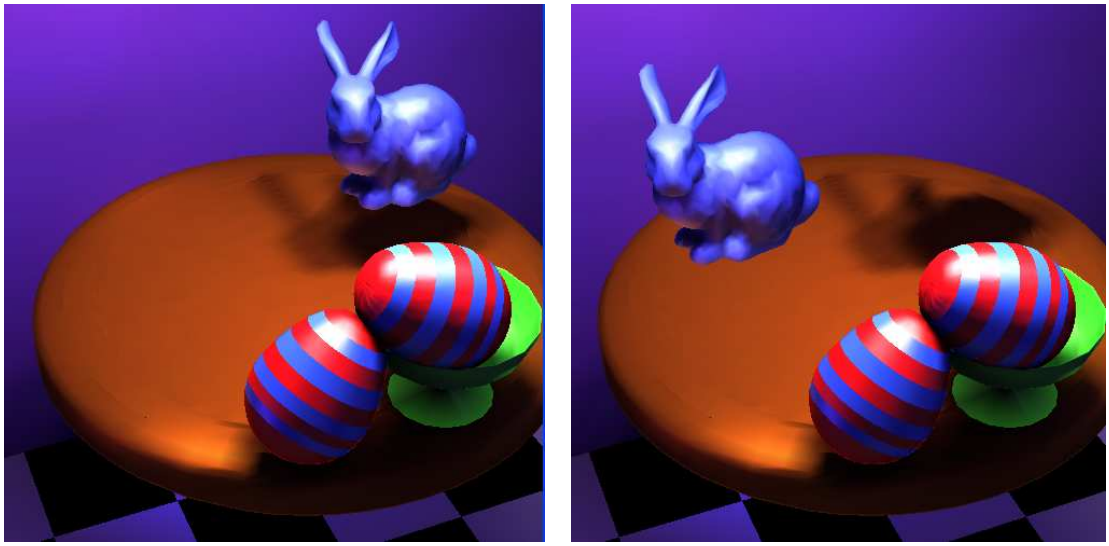
These double transfers quickly update the illumination according to the new situation and after a few iterations, the shadows and highlights are moved to their new positions. At the end of this special iteration phase, the computed ΔL increments are added to the stored radiance representation (i.e. to the main part and to the scaling factor). In order to further refine the results, the algorithm switches back to the normal stochastic iteration scheme and iterates according to formula $L^e_{new} + \mathcal{T}_{new}L_{new}$.

7.4.1 Results

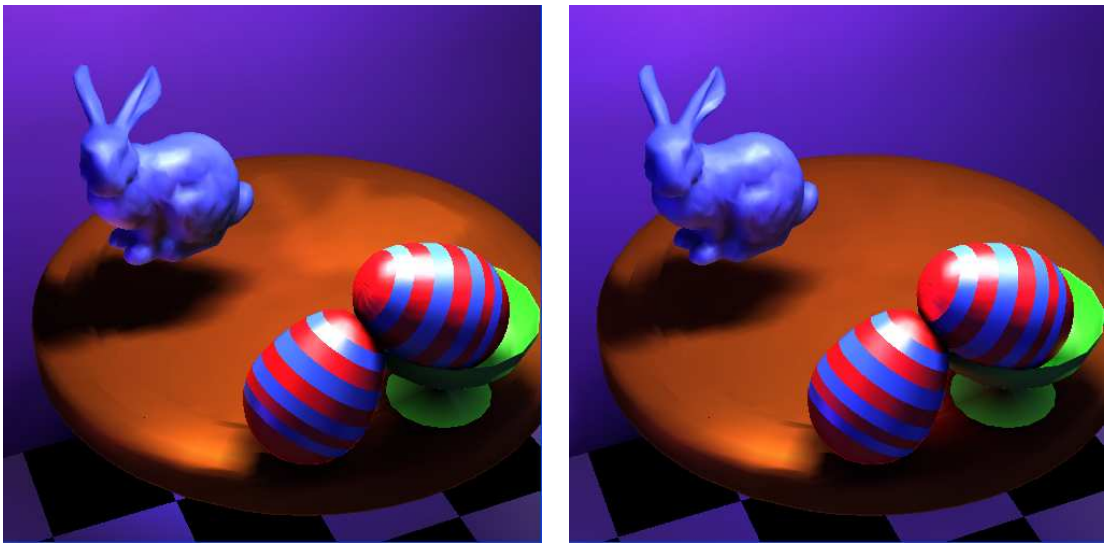
Figure 7.3 shows two displayed images and temporary results of an object animation. The scene consists of 20 thousand patches. The stripes of the egg have $(0.1, 0.2, 0.7)$ and $(0.8, 0.04, 0.04)$ diffuse albedos, 0.14 specular albedo, and the shininess values are 9 and 11, respectively. The rabbit's diffuse albedo is $(0.16, 0.19, 0.63)$, the specular albedo is 0.15, and the shininess is 9. The animation speed depends on the number of special iterations made to update the radiance. We have found that 10 iterations provide good images, which results in 1 frames per second. In interactive applications, however, users require a prompt response from the system, thus accuracy should be traded for speed. This is possible if the iteration number in the update cycles is reduced.

7.5 Summary

In this chapter we propose a random radiance representation scheme and an animation approach that can exploit both space and time coherence. This representation



old | after moving the object



end of adaptation | continuing the iteration

Figure 7.3: The first and last images show two frames of an object animation rendered at 1 frame per second on a Pentium®4 2GHz computer. The two other images are not seen by the user, but demonstrate the roles of the adaptation phase.

requires just a few values per patch, thus the storage requirement is modest, it is close to the storage need of a diffuse radiosity algorithm, although the proposed method is also good for glossy scenes. If the surfaces are not highly specular, the variance caused by the randomization is small due to the applied main part separation and the application of Metropolis sampling to maintain a constant luminance. Thus we can get fast initial convergence of finite-element based iteration methods without their prohibitive memory requirements. The used stochastic iteration algorithm combines three random radiance transport methods based on multiple importance sampling. This novel combined strategy preserves the advantages of local and global light transfers, and eliminates the corner problem of local shooting and the necessity of first shot of global sampling. The combined method is able to render moderately complex glossy scenes with the speed required by interactive systems. The application of parallel and perspective ray-bundles not only resulted in an effective global illumination algorithm, but proved to be really powerful to detect where the radiance function should be updated in an animation sequence.

Highly specular surfaces pose problems for this approach since they increase the variance of the random radiance representation and require higher tessellation levels to reconstruct the quickly changing radiance in the highlights. Fortunately, stochastic iteration applying bundles performs well on scenes containing a lot of patches. The rasterization and the radiance transfer through the pixels of the buffers are the bottlenecks of the computation, and not the geometric transformations (the rasterization time does not change if the patches are tessellated further, and the number of required iterations depends on the variation of the radiance function and not on the number of patches). On the other hand, the spatial finite-element representation eliminates the objectionable dot-noises and reduces the flickering of other Monte-Carlo algorithms and makes the algorithms practically independent of the image resolution.

Chapter 8

Conclusion

This dissertation studies a novel global illumination approach – namely the stochastic iteration – which is so new that it is not yet investigated frequently in the computer graphics literature. However, this gives good opportunity to refine the basic algorithm and fill the gaps that was left behind since the first proposal of the algorithm in [SK99b].

8.1 Contribution of this thesis

The main contribution of this dissertation is the interactive global illumination animation. According to the Chinese proverb that "The journey of a thousand miles, starts with a single step.", we achieve our big main goal by making separate steps.

The first thesis – that is detailed in chapter 3 – can be considered as a sidetrack, since it deals width one detail of the random walk global illumination approach, namely the optimal survival probability of the Russian roulette. The study can be used also in other scientific areas, because Russian roulette is a general technique used in Monte-Carlo methods. The rest of the dissertation focuses strictly on stochastic iteration methods. I summarize the contribution of this dissertation in the following:

Thesis 1: Russian roulette optimization ([P1]):

I have analyzed the possibility of optimizing Russian roulette that is used in random walk algorithms for calculating the infinite Neumann series. I have proved that the local albedo that is usually used for the surviving probability is not optimal, but the optimal probability lies between the local albedo and 1. The more homogenous the radiance of the scene, the optimal probability is closer to 1.

Thesis 2: Extended parallel ray-bundle iteration ([P2]):

I have worked out a method for improving parallel ray-bundle iteration. The new algorithm subdivides the radiance function into a finite-element component and into a Monte-Carlo component, therefore the global illumination problem is decomposed into a simple finite-element problem and into a low variance Monte-Carlo problem.

The Monte-Carlo part is responsible for building the finite-elements, on the other hand, the finite element part reduces the variance of the Monte-Carlo component. The method is similar to the classic main part separation. The difference in this case is that the main part is not known and it cannot be integrated in a closed form. Therefore the main part is generated adaptively by Monte-Carlo quadrature. I applied the general mathematical scheme to the ray-bundle stochastic iteration algorithm and came to the conclusion that it improves the rendering speed.

Thesis 3: Perspective ray-bundle iteration ([P3]):

I generalized the hemicube based diffuse radiosity for specular scenes. The new algorithm suits into the stochastic iteration framework. Furthermore, I extended this basic perspective ray-bundle iteration by importance sampling, random selection of hemicube sides, constant radiance term and by introducing a special controllable bias that results significant variance reduction.

Thesis 4: Combination of ray-bundle based strategies ([P4, P5, P6, P7]):

I combined different methods based on stochastic iteration. At first I modified the method of multiple importance sampling for being applicable to iteration algorithms. I combined three algorithms: parallel ray bundles with software z-buffer, parallel ray-bundles with hardware z-buffer and perspective ray-bundles. The combined method preserves the advantages and eliminates the drawbacks of the underlying techniques. It is not only effective if the scene consists of patches of similar radiance but also if one or several patches are much brighter than the others.

Thesis 5: Ray-bundle based global illumination animation ([P8, P9]):

I extended the combined algorithm and achieved walkthrough and general animation with interactive speed on a single computer. I worked out a special iteration phase for updating the radiance after moving dynamic objects.

8.2 New research directions

As with any other work there are always things and ideas that are left untried or unimplemented. Some possible directions – in which the work in this thesis can be continued – are gathered together here.

It would be possible to extend the ray-bundle iteration by using BTF (Bidirectional Texture Function) instead of BRDF. In this way the points across the surface of the patches can have different colors, which could not only improve the direct illumination in the rendering, but the textures should play important rule in inter-reflection calculations also, which would result nice effects like the reflection of a texture painting on a glossy surface.

Other enhancements for the parallel ray-bundle iteration are also possible. The

transillumination directions are sampled uniformly, which is not a necessity. Directions can be generated by any probability distribution function. It would be advantageous to consider importance sampling which prefer those directions that convey more energy. For improving the rendering of visible specular surfaces, it should prefer those directions which would possible generate the perfect mirror directions when reflected towards the eye. This *view dependent importance sampling* for transillumination direction should improve the rendering of nearly perfect visible mirrors.

The rendering equation proposed by Kajiya considers only surfaces and assumes that the light intensity is not attenuated in the transport between two surfaces. However, there is a recent trend to extend current global illumination algorithms with participating media. The parallel ray-bundle iteration can be extended by rendering partly transparent particle points onto the transillumination buffer. The perspective ray-bundle and also the combined algorithm require other treatments that we would like to present in the future.

As with any other global illumination animation, which is based on stochastic approaches, the temporal aliasing may be annoying to the human observer. This sometimes occurs as flickering, darkening and lightening the colors of the surfaces in the successive frames. The offline animation algorithms try to handle this by using the same random numbers or low discrepancy sequences for each frame. In our algorithm the main part separation is effective for reducing the temporal aliasing. However, other anti-aliasing techniques would further improve the rendering.

It is a general technique to decompose the lighting distribution into direct and indirect parts. The radiance caused by the indirect illumination should be computed by ray-bundle based stochastic iteration and the direct illuminations should be generated using the graphics card by placing point light sources in the scene. We successfully used this decomposition in the parallel ray-bundle iteration in static environments. However, enhancing the combined iteration proposed for dynamic environments is straightforward and could result faster generation of image sequences.

The parallel approach – even with a small number of computers – is an effective way to produce speedup. It is very likely that the first real-time global illumination methods will be distributed applications. Since parallel ray-bundle iteration can be modified for running on more than one CPU, we should consider implementing our combined ray-bundle method to run in distributed environment.

8.3 A final word

We think that stochastic iteration has a bright future and with the help of this work it is getting wider publicity among the computer graphics scientist. On the user side, there are high demands – especially among the people working in the CAD industry or in architectural design – for very fast and very realistic rendering. Our belief is that computer graphics is marching on the road which eventually leads to real-time global

illumination for complex scenes. In the future we will surely get there. However, they are still many snags and challenges to overcome. This dissertation is aimed to step forward on this road.

Appendix A

Framework for global illumination algorithms

This appendix describes a rendering framework called RenderX for testing and implementing global illumination methods. We started to build the rendering framework 4 years ago by supporting the implementation needs of our computer graphics research. We named the system RenderX. The system has been developed by Ferenc Csonka, Balázs Benedek, László Szécsi, Gábor Szíjártó and László Kovács and supervised by the author of this dissertation. This architecture facilitates in the realization of existing algorithms and in the development of new techniques, since it frees programmers and researchers from implementing an entire system, thus enables them to focus only on those areas that they are interested in. The system is built on an object oriented basis, therefore it is flexible and can be extended easily. Compared to other frameworks, a unique feature of our system is that it supports both types of global illumination techniques, namely the random walk and the finite-element approaches. To illustrate the strength of the framework, we implemented and compared existing and newly developed rendering methods for global illumination. The pseudo-code for global illumination algorithms are briefly revisited and the implementation issues together with the rendering results are presented in this chapter.

A.1 Introduction

In recent years computer graphics researchers have paid a lot of efforts for trying the usefulness of their algorithms in practice. Most of the authors implement a complete software package for their published paper. However, this has two major drawbacks. At first, this requires unnecessarily too much work for the researcher, since he spends a lot of time for debugging his source code. On the other hand, comparing the results of the different rendering strategies is not possible.

Using a framework, which is a test-bed for the realization of many global illumination algorithms is inevitable. The most important feature of such a rendering

framework is the open-source nature. Thus rendering systems which are given as a compiled binary (DLL) and accessible through just an API are not feasible.

A.2 Previous work

There are several existing systems that enable researchers to construct their algorithms and compare it to other methods. Usually the implementation is started by a single researcher and after the first initial steps the system was extended by a team.

The **Vision** system [Slu95] was developed in the context of a PhD thesis in 1995. It contains software components, which communicate by the CORBA protocol. Vision considers a physically based object-oriented decomposition of the rendering process.

The Abstract Rendering Toolkit [Tob98] (**ART**) is the product of the Vienna University of Technology. The group promised to release the first public beta in 2001, but until now the source code is not accessible on the Internet, only for the researchers of the Vienna group. Therefore we do not have a chance to inspect their system or compare it with our framework.

The state-of-the-art is definitely the **RenderPark** [Bek93] system. The development was started by Philippe Bekaert as a test-bed for his research in 1997. Since then it has become famous, and it is used mostly in the Max Planck Institute in Saarbrücken. The strength of RenderPark is its extensive usage of radiosity algorithms.

The **EFFIGI** [LW00] (An Efficient Framework For Implementing Global Illumination) system is new a player on the field. Breaking the tradition, this framework is built on a completely new concept. It is constructed by the Image Synthesis Group of the Trinity College Dublin. The system is built on COM (Component Object Model), which is a robust protocol for connecting software components. Because of that, the unique feature of EFFIGI is that it is not restricted to a single programming language.

In spite of the limitation as a robust rendering framework, the open-source **PovRay** [Pov] should be also mentioned here. It is useful only for implementing random walk type global illumination algorithms.

A.3 The component architecture

This section presents a general overview of the structure of the RenderX framework. We identify the basic subsystems that are needed for the rendering process. Note that we just study the core of the system here, and we do not consider, for example, the user interface classes. The user interface subsystem can contain more classes then all other subsystems in our framework. They are usually reached via an API (Motif, QT, MFC).

The framework presented in this paper tries to achieve the following goals:

- enable code sharing and reuse,
- as generic and flexible as possible,
- easy to understand,
- there is no fixed rendering algorithm,
- efficient,
- open source,
- the user is not required to learn the entire core, just the part of interest.

A.3.1 Geometry subsystem

When designing the components in the rendering architecture, at first the basic building blocks of the system must be identified.

Some utility classes were obviously needed: points, vectors of different lengths, colors, matrices. It is possible to use off-the-self solutions for these basic entities, but for flexibility issues, in our framework we realized our own classes. The geometry subsystem is responsible for defining surfaces (or volumes also). Since there are many different techniques for representing surfaces (poligons, NURBSs, etc.), we should derive many surface classes from an abstract surface parent class. However, for efficiency reasons (hardware acceleration) we currently support only the triangle primitive, which is realized in the CPatch class. On the other hand, introducing other surface types into the system is not difficult.

Another basic utility class is the ray class, represented by a unit-vector and the point of origin. The aim is to choose the interfaces in a way that the rendering methods can be designed independently from the representation of the scene (type of geometrical primitives) and the underlying algorithms. Heckbert observed that geometrical primitives share a common interface for ray-tracing systems. Kirk and Arvo has generalized this idea [AK89], and observed that the ray-object intersection structures (regular grid, octree, BSP-tree, kd-tree) can also be hidden into a common interface. In RenderX we support both the octree and the kd-tree data structures. Because of the speed advantage, the kd-tree is the default data structure.

A.3.2 Shader subsystem

This subsystem describes the reflection of light at a point of the surface. This description is independent from the actual representation of the geometric objects. Usually, the shader is responsible to store the BSDF (Bidirectional Scattering Distribution

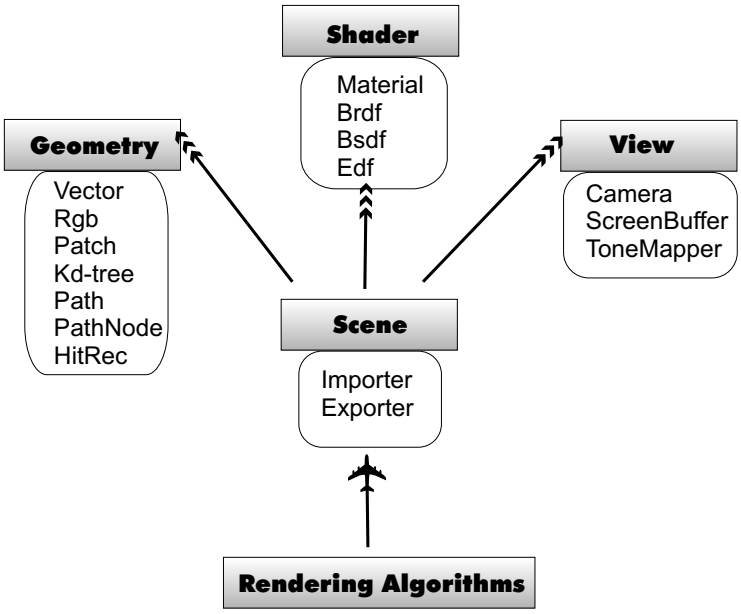


Figure A.1: The component diagram of the RenderX architecture and the most important classes

Function). The BRDF (Bidirectional Reflectance Distribution Function) is a simplified BSDF, which does not include transmittance. The subsystem is built on the *CMaterial* class that represents an abstract material type. The material consists of a *CBrdf* and a *CEdf* (Emittance Distribution Function) class. The abstract *CBrdf* can hide any type of BRDFs from its realization. For example, we usually use the *CPhongBrdf*, which describes a Phong type specular shader. Other shaders (like Cook-Torrance) can be easily added to the framework. We consider the area ligh-sources as common surfaces with materials, which has a *CEdf* component. The most feasible *Edf* uses cosine distribution for emitting light. In spite of the fact that the point light sources are physically impossible to exist, they are also allowed in the system. They are stored separately and can be considered by the rendering method.

A.3.3 View subsystem

In the geometry and shader subsystems we determined what is to be rendered. In this section we specify what kind of image generation we need. The view subsystem defines one particular view of the scene, which corresponds to one particular image. A single scene can also have multiple *Camera* objects, in which case it is possible to switch between them. This subsystem is mainly defined by the *Camera* object. In our system we assume the pinhole camera model, so the Camera object contains the position, viewing and up direction, focal distance, field of view angles, distances of the two cutting planes and image resolution.

The *ScreenBuffer* corresponds to the final stage of the image generation process. The concept of *ScreenBuffer* in RenderX is a rectangular array of pixels with a given width and height and 3 channels for each pixels. It contains a radiance map and color map. Radiance map contains radiance values, which are the result of a rendering algorithm. Color map contains the quantized [0..255] values of the radiance map. The quantization is made by the *ToneMapper* object before displaying the image. Different tonemapping techniques can be applied by deriving a new tonemapper from the abstract *ToneMapper* class.

A.3.4 Scene subsystem

In contrast to the previous subsystems, this subsystem does not describe parts of the scene, rather it helps organizing the scene description. This subsystem contains helper classes, which are cardinal in the system. One corresponds to building up the scene from a scene description file. These are the so called *Importer* classes. They build the geometry, shader and view subsystems. In our framework there are 3 types of them. The *VRMLReader* can read VRML2.0 scene files, the *3DSReader* reads 3D Studio files and the *SCEReader* reads scenes files saved by ArchiCAD.

The output of the rendering is the final image, so the exporters are also important in a rendering architecture. The RenderX currently supports BMP, TGA and PNG file formats.

A.3.5 Rendering algorithm subsystem

If there is a hierarchy, this subsystem is on the top of the others. Different rendering algorithms can be implemented by deriving a subclass from the *RenderingAlg* abstract class and call the *StartRender*() method of the instance. Since the RenderX support multi-threading, it is desirable – and it is the default – that the *StartRender*() is called in a different thread of the process. At first it allows running different algorithms parallel and another advantage is that the user does not loose the feedback and interactivity when starting a single algorithm.

A.3.6 Implementation

The implemented user interface consists a preview window (drawn by OpenGL) always open, where the user can interactively navigate through the scene. When he selects a rendering algorithm another sub-window opens and the global illumination algorithm starts in a different thread. The user can still navigate in the preview window or select other menus with other rendering algorithms.

The complete RenderX architecture is written in C++ [Sou97] and currently consists about 50 classes. The scene walkthrough mode allows the user to interactively

move in the scene, which is implemented using the OpenGL 1.3 API. The user interface is implemented using the MFC (Microsoft Foundation Classes) API.

A.4 Algorithms for the RenderX architecture

In order to show the strength of the framework, we implemented the most common global illumination algorithms [SK99a]. We introduce briefly the pseudo-code here, present implementation issues and show the final images. The images were usually rendered in 600x600 resolution using Pentium®III 1.4 Ghz computer.

A.4.1 Gathering algorithms

In the next sections, a number of practical random walk algorithms are reviewed. The primary classification of random walk algorithms is based on the direction of generated rays. If the walks are started at the eye and go opposite to the light, then the algorithm is called *gathering*. On the other hand, if the walks originate at the light sources and go in the direction of the light, then the algorithm is called *shooting*. The general structure of all random walk type gathering algorithms is:

```
for (each pixel p) {
    color = 0
    for (i = 1 to N) {
        ray = sample ray randomly from the eye through pixel p
        samplecolor = c · Trace(ray)
        color += samplecolor/N
    }
    SetPixel(p, color)
}
```

A.4.2 Ray-casting

Ray-casting is a local-illumination algorithm of type LDE (see section 2.3), which replaces the unknown radiance inside the integral of the rendering equation by the emission function. The "Trace" function of ray-casting is:

```
Trace(ray) {
    (object, x⃗) = FirstIntersect(ray)
    if (no intersection)
        return L_sky
    color = L^e(x⃗, -ray.dir) + DirectLight(x⃗, -ray.dir) + Ambient()
    return color
}
```

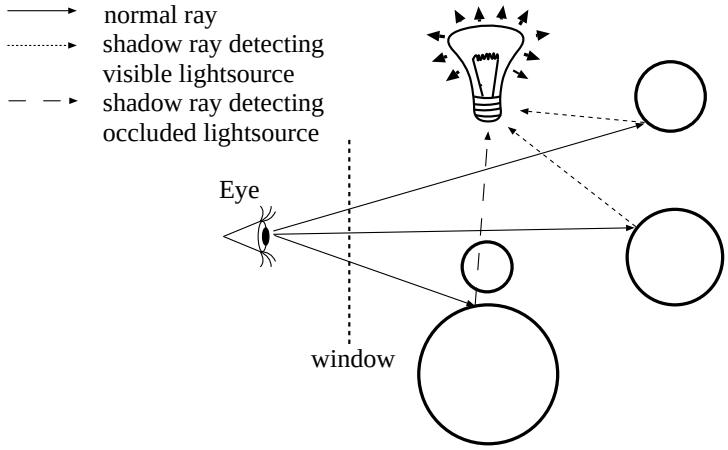


Figure A.2: Ray-casting

In this algorithm L_{sky} is the radiance of background illumination (e.g. sky), "FirstIntersect" is responsible for finding that object which is first intersected by the ray and also the intersection point.

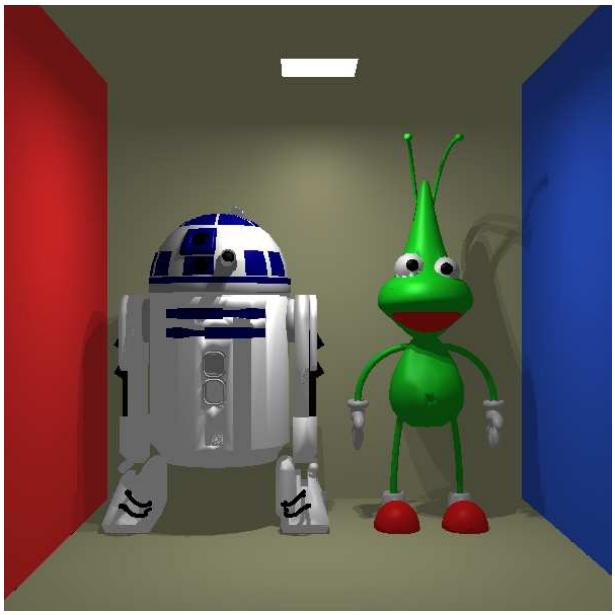


Figure A.3: Alien meets R2D2 in the Cornell-box

"DirectLight", on the other hand, computes an estimate of the single reflection of the light originated from the light sources, which happens at point $\vec{x}$ into the given direction. It is achieved by following shadow rays (see in figure A.2), and it is easy to perform, when the scene contains only point light sources. In order to handle area light

sources, Monte-Carlo integration can be used, which selects N uniformly distributed $\vec{y_i}$ samples on the light source area and accumulates the radiance contribution of the samples.

Without the "Ambient" term, the directly not illuminated surfaces are completely black. This is not suitable for final images, so a simple constant estimation (ambient) for the indirect illumination is added to the final color.

The R2D2 image rendered with this method can be seen in figure A.3. The scene consists of 22K patches and the rendering time was 8 seconds.

A.4.3 Path tracing

The path tracing is a special case of stochastic ray tracing (see section 2.4.2 for details). It is a traditional Monte-Carlo approach for global illumination, and it is based on the multi-dimensional integral formulation of the rendering equation.

This method creates a path history for a single particle interacting with the environment until absorption using BRDF sampling and Russian roulette. Rather than spawning new rays at each intersection, it chooses a random direction according to a density p_i which is approximately proportional to the BRDF of the surface. The walk is continued with a probability of the albedo (Russian roulette).

The measured value of a single path is

$$P = c \cdot (L_1 + L_2 + L_3 + \ldots).$$

where L_i is the radiance estimate of the subpath of length i.

This estimate has very high variation if the light sources are small. This problem can be solved if light source sampling is combined with the gathering walk, which means that at each visited point the effects of the light sources are estimated.

The implementation of the "Trace" function of path tracing is:

```
Trace(ray) {
    (object, x⃗) = FirstIntersect(ray)
    if (no intersection)
        return L_sky
    color = L^e(x⃗, -ray.dir)+ DirectLight(x⃗, -ray.dir)
    prob = BRDFSampling(-ray.dir, normal, newRay)
    if (prob = 0)
        return color
    color +=Trace(newRay)·w(newRay.dir, normal, -ray.dir)/prob
    return color
}
```

In this program "BRDFSampling" finds a new direction or if it returns 0, then it has decided that the walk has to be stopped because of Russian-roulette. Note that

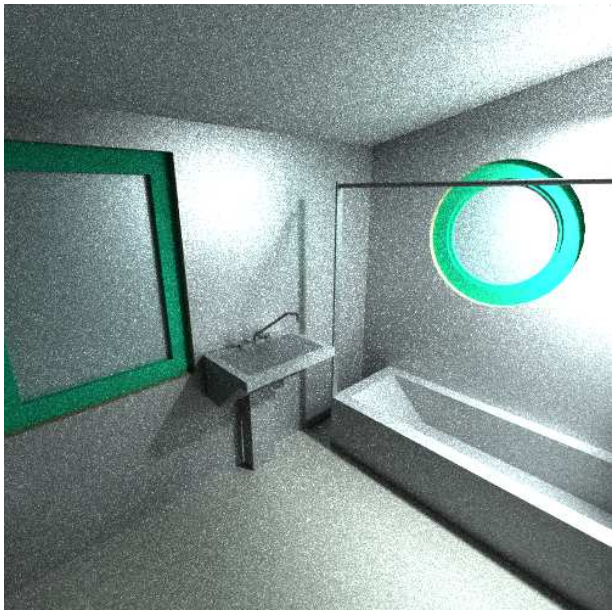


Figure A.4: Image rendered with path tracing

this algorithm generates all but the last directions of the path by BRDF sampling and the last is obtained by light source sampling. Thus if the surface is shiny (close to ideal mirror or ideal refractor), then the quality of the light source importance sampling can be quite bad. Since almost ideal surfaces close to the light sources are responsible for caustics, path tracing — as other gathering type algorithms — is poor in rendering caustics effects.

An architectural scene rendered with path tracing can be seen in figure A.4. The scene consists of 15K patches and the rendering time was 1.5 hours.

A.4.4 Bidirectional path tracing

Bidirectional path tracing (see section 2.4.4 for details) initiates paths at the same time from a selected light source and from the camera. After some steps, either a single deterministic shadow ray is used to connect the two types of walks, or all points of the gathering walk are connected to all points of the shooting walk using deterministic rays. If the deterministic shadow ray detects that the two points are occluded from each other, then the contribution of this path is zero.

An architectural scene rendered with an extended type of bidirectional path tracing which uses dependent tests for filtering [P20] can be seen in figure A.5. The scene consists of 20K patches and the rendering time was 30 minutes.

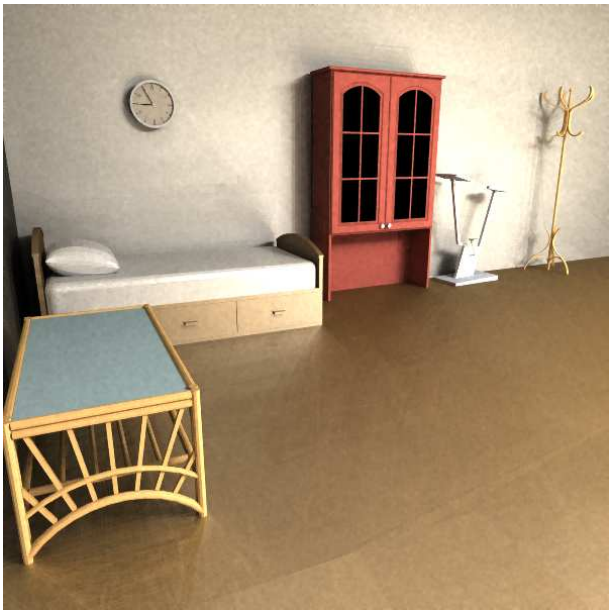


Figure A.5: Image rendered with bidirectional path tracing

A.4.5 Metropolis light transport

Metropolis method (see section 2.4.5 for details), generates samples by perturbing the previous path, thus this method expected to be better than blind bidirectional path tracing for difficult lighting conditions.

Figure A.6 was generated by the Metropolis algorithm. The average computation time of a single mutation took 0.005 msec on a Pentium®III 1.4 Ghz computer.

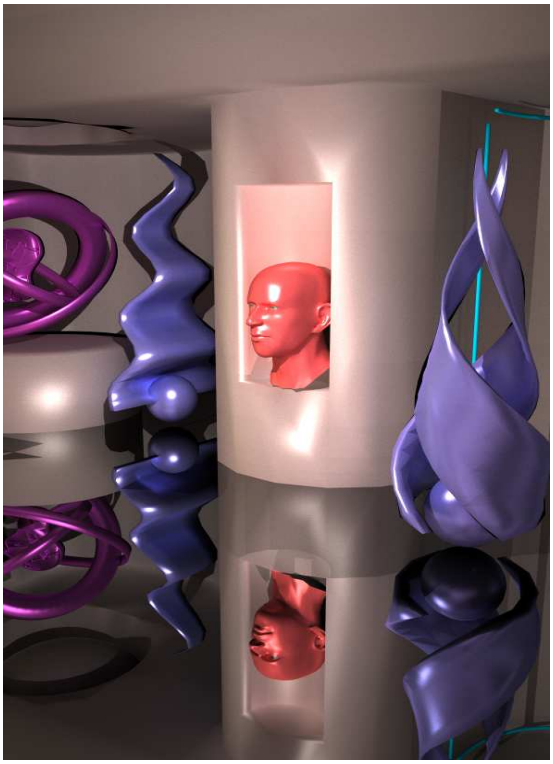


Figure A.6: Image rendered with the Metropolis Light Transport

A.4.6 Photon map

Bidirectional path tracing connects a single gathering walk to a single shooting walk. However, if the effects of all shooting walks could be stored, then when a new gathering walk is computed, it could be connected to all of the shooting walks simultaneously, which can significantly increase the number of samples in the integral quadrature. This is exactly what Jensen [Jen96] proposed, also giving the definition of a data structure, called the photon-map (see section 2.4.6 for details) which can efficiently store the effects of many shooting walks.

A photon map is a collection of photon hits at the visited points of the paths generated in the shooting phase of the algorithm.

The photon-map is organized in a kd-tree to support efficient retrieval. A photon hit is stored with the power of the photon on different wavelengths, position, direction of arrival and with the surface normal.

The algorithm has thus three phases:

- Photon tracing.
- Sorting the photon map and balancing the kd-tree.
- Final gathering, which uses a standard ray-tracing to collect information from the photon map.

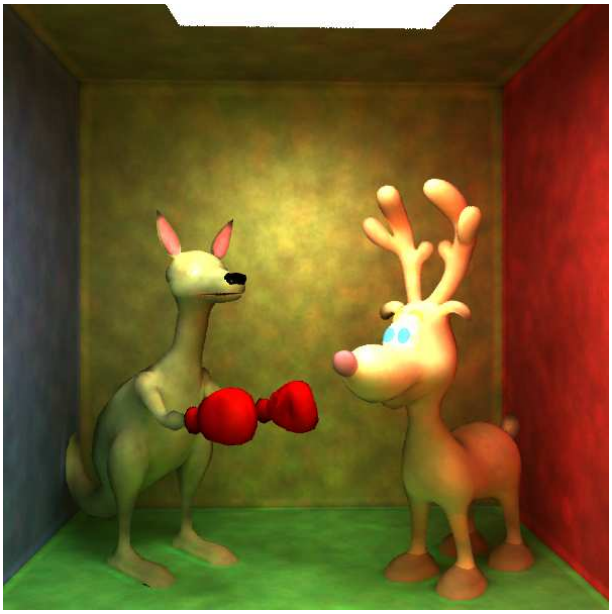


Figure A.7: Image rendered with the Photon Map method

In figure A.7 the "fighting animals" scene can be seen. This image is the result of the direct visualization of the photon map. Note the bumping artifacts on the image. A better image quality can be achieved by performing direct light source computation first, and using the photon map just for the indirect illumination. The rendering time of the scene (21K patches) took 50 minutes.

A.4.7 Parallel ray-bundle iteration

This dissertation is based on the careful implementation of stochastic iteration algorithms. The parallel ray-bundle iteration was detailed in chapter 4. For the sake of completeness we mention this type of algorithm also here.

An architectural scene is presented in figure A.8. The scene has 15K patches. As all types of object space rendering algorithm strongly exploits coherence, it is no surprise that the rendering time was 150 seconds for this scene. Note that the rendering time is not affected by the resolution of the image.

Figure A.8: Architectural scene rendered with parallel ray-bundle iteration

A.4.8 Perspective ray-bundle iteration

The list of implemented stochastic iteration algorithms is continued. The perspective ray-bundle iteration – as one of the results of this thesis – was described in chapter 5.

The architectural scene presented in figure A.9 consists 30K patches. The direct illumination is calculated separately by sampling points on area light sources. This is a costly process, so the rendering time is 3 minutes. The Soutpark scene has the same properties. As in the previous algorithm, the rendering time is not affected by the resolution of the image.

A.4.9 Others

Besides the presented algorithms, many other global illumination methods were also tried and tested within this framework. Most of them is provably unbiased combination of these methods. Some of them are extensions of the presented methods.

Like in other software package developments, there are always ideas to be left unimplemented giving ammunition for future work. The system does not yet give full support for texture mapping. First of all this should be implemented. It would be also desirable to introduce a Sampler abstract class, therefore the many types of stratification methods could be tried easily.

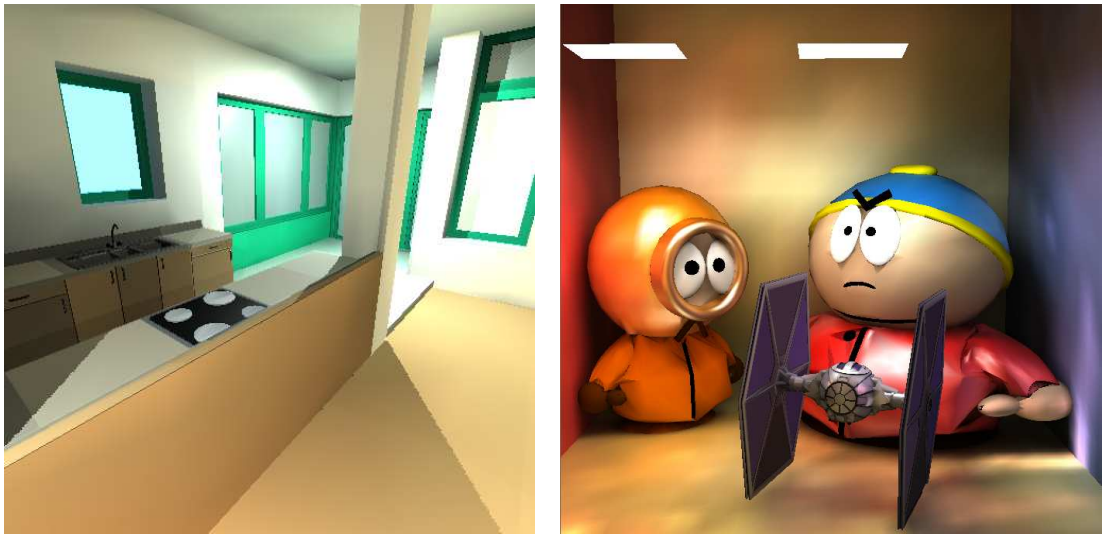


Figure A.9: Architectural scene and Eric Cartmen from Southpark rendered with Perspective Ray-bundle Iteration

A.5 Summary

In this appendix we have presented a rendering framework which is successful for implementing many of the latest photorealistic image synthesis algorithms. The programmer can experiment with various ideas (different BRDFs, different ray-object intersection methods), without significantly alter the source code. Since the framework has great flexibility and the components support code reuse, the RenderX package is definitely a powerful rendering environment.

Publications

Directly related to theses

[P1] György Antal, László Szirmay-Kalos: Finding Good Termination Probability for Random-Walks. *Proceedings of Spring Conference on Computer Graphics*, pp. 205–211, 2000

[P2] László Szirmay-Kalos, Ferenc Csonka, György Antal: Global Illumination as a Combination of Continuous Random Walk and Finite Element Based Iteration. *Computer Graphics Forum (Eurographics 2001)*, Vol. 20, No. 3, pp. 288–298, 2001

[P3] György Antal, László Szirmay-Kalos, Ferenc Csonka: Hemicube Shooting for Non-Diffuse Global Illumination. *Proceedings of Spring Conference on Computer Graphics, IEEE Computer Society*, pp. 81–88, 2002 (best paper award: first prize)

[P4] György Antal, László Szirmay-Kalos, Ferenc Csonka: Multiple Strategy Stochastic Iteration For Architectural Walkthroughs. *Computers & Graphics*, Elsevier Journal, Vol. 26, No. 3., pp. 192–198, 2003.

[P5] György Antal, László Szirmay-Kalos, Ferenc Csonka, Csaba Kelemen: Multiple Strategy Stochastic Iteration for Architectural Walkthroughs. *Dagstuhl Seminar on Monte-Carlo Methods in Rendering*, 2001

[P6] György Antal, László Szirmay-Kalos, Ferenc Csonka, Csaba Kelemen: Multiple Strategy Stochastic Iteration for Architectural Walkthroughs. *Technical Report, TR-186-2-01-17*, Vienna, 2001, http://www.cg.tuwien.ac.at/research/TR/

[P7] György Antal, Ferenc Csonka: Combining Global and Local Global--Illumination Algorithms. *Proceedings of Spring Conference on Computer Graphics*, pp. 199–206, 2003, (best paper award: third prize)

[P8] György Antal, Balázs Benedek, Ferenc Csonka: Global Illumination Animation with Ray Bundle Iteration. *Proceedings of the 2nd Hungarian Conference on Computer Graphics*, pp. 205–213, 2003

[P9] László Szirmay-Kalos, György Antal, Balázs Benedek: Global Illumination Animation with Random Radiance Representation. *Rendering Techniques'03, (Eurographics Symposium on Rendering)*, Editors : P. Christensen, D. Cohen-Or, pp. 64–73, 2003

Books or part of books

[P10] László Szirmay-Kalos, György Antal: Metropolis Sampling in Random Walk Global Illumination Algorithms. *appears in book "Graphics Programming Methods"*, (edited by Jeff Lander), Charles River Media, 2003.

[P11] Szirmay-Kalos László, Antal György, Csonka Ferenc: Háromdimenziós grafika, animáció és játékfejlesztés. (book in Hungarian) *ComputerBooks*, 2003

[P12] Antal György: Java3D. (book chapter) *Nyéky J. et al. ed. Java 2 útikalauz programozóknak*, ELTE TTK Hallgatói Alapítvány, Budapest, Hungary, 1999. pp. (II)364–422, (III)362-494.

Other publications in the topic of the dissertation

[P13] György Antal, Ferenc Csonka: An Efficient And Robust Framework for Global Illumination Algorithms. *Proceedings of 1st Hungarian Conference on Computer Graphics*, pp. 198–206, 2002

[P14] Csaba Kelemen, László Szirmay-Kalos, György Antal, Ferenc Csonka: Simple and Robust Mutation Strategy for Metropolis Light Transport Algorithm. *Computer Graphics Forum (Eurographics 2002)*, Vol. 21., No. 3, 2002

[P15] Ferenc Csonka, László Szirmay-Kalos, György Antal: Generalized Multiple Importance Sampling for Monte-Carlo Global Illumination. *Machine Graphics and Vision*, Vol. 11, No. 4, 2002

[P16] György Antal, László Szirmay-Kalos, Ferenc Csonka: Weighted Multipass Method Based on Stochastic Iteration and Random Walk Methods. *Proceedings of Winter School Of Computer Graphics Conference*, pp. 24–31, 2002

[P17] Balázs Benedek, László Szirmay-Kalos, György Antal: Weighted Importance Sampling in Shooting Algorithms. *Proceedings of Spring Conference on Computer Graphics*, pp. 192–198, 2003

[P18] László Szirmay-Kalos, György Antal, Mateu Sbert: Progressive Light Path Development. *Proceedings of Winter School of Computer Graphics Conf.*, pp. 411–418, 2001

[P19] Ferenc Csonka, László Szirmay-Kalos György Antal: Cost-Driven Multiple Importance Sampling for Monte-Carlo Rendering. *MCM Conference*, Salzburg, 2001

[P20] Ferenc Csonka, László Szirmay-Kalos, Csaba Kelemen, György Antal: Dependent Tests driven Filtering in Monte-Carlo Global Illumination. *Eurographics Conference*, 2002

[P21] Ferenc Csonka, György Antal: A Multi-Phase Energy Preserving Filtering Method for Architectural Scenes. *Proceedings of 1st Hungarian Conference on Computer Graphics*, pp. 189–197, 2002

[P22] Csonka Ferenc, Antal György: Sztochasztikus iteráció fényelnyelő közegben. *Proceedings of 2nd Hungarian Conference on Computer Graphics*, pp. 214–219, 2003

[P23] Ferenc Csonka, László Szirmay-Kalos, György Antal: Cost-Driven Multiple Importance Sampling for Monte-Carlo Rendering. *Technical Report*, TR-186-2-01-19, Vienna, 2001

Bibliography

[AK89] J. Arvo and D. Kirk. A survey of ray tracing acceleration techniques. In Andrew S. Glassner, editor, *An Introduction to Ray Tracing*, pages 201–262. Academic Press, London, 1989.

[AK90] J. Arvo and D. Kirk. Particle transport and image synthesis. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 63–66, 1990.

[Bek93] Ph. et al. Bekaert. Renderpark. Technical report, 1993. http://www.cs.kuleuven.ac.be/cwis/research/graphics/RENDERPARK.

[Bek97] Ph. Bekaert. Error control for radiosity. In *Rendering Techniques '97 (8th Eurographics Workshop on Rendering)*, Porto, Portugal, 1997.

[Bek99] Ph. Bekaert. *Hierarchical and stochastic algorithms for radiosity.* PhD thesis, University of Leuven, 1999. http://www.cs.leuven.ac.be/cwis/research/graphics/ CGRG.PUBLICATIONS/PHBPPHD.

[BNN+98] Ph. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. Willems. Hierarchical Monte-Carlo radiosity. In *Rendering Techniques '98*, pages 259–268, 1998.

[BP95] Kadi Bouatouch and Sumanta N. Pattanaik. Interactive Walkthrough Using Particle Tracing. In Rae E. Earnshaw and John A. Vince, editors, *Computer Graphics Developments in Virtual Environments (CG International '95 Proceedings)*, Boston, MA, 1995. Academic Press.

[BP01a] Gonzalo Besuievsky and Xavier Pueyo. Animating radiosity environments through the multi-frame lighting method. *Journal of Visualization and Computer Graphics*, 12:93–106, 2001.

[BP01b] Gonzalo Besuievsky and Xavier Pueyo. A monte carlo method for accelerating the computation of animated radiosity sequences. In *Proceedings of Computer Graphics International 2001*, pages 201–208, 2001.

[BS96] Gonzalo Besuievsky and Mateu Sbert. The multi-frame lighting method - a Monte-Carlo based solution for radiosity in dynamic environments. In *Rendering Techniques '96*, pages 185–194, 1996.

[BW97] Takács B. and H. Wechsler. A dynamic and multiresolution model of visual attention and its application to facial landmark detection. In *Computer Vision and Image Understanding*, volume 71, 1997.

[BW99] S. Parker B. Walter, G. Drettakis. Interactive rendering using the render cache. In *Rendering Techniques'99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, pages 235–246, 1999.

[CCWG88] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, pages 75–84, 1988.

[CG85] M. Cohen and D. Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, pages 31–40, 1985.

[Che90] Shenchang Eric Chen. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 135–144, August 1990.

[Chr00] P. Christensen. Faster photon map global illumination. *Journal of Graphics Tools*, 4(3):1–10, 2000.

[CJ02] Mike Cammarano and Henrik Wann Jensen. Time dependent photon mapping. In *Rendering Techniques 2002 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)*, June 2002.

[CLSS97] P. H. Christensen, D. Lischinski, E. J. Stollnitz, and D. H. Salesin. Clustering for glossy global illumination. *ACM Transactions on Graphics*, 16(1):3–33, 1997.

[CMSK97] B. Csébfalvi, G. Márton, and L. Szirmay-Kalos. Fast opacity control of volumetric CT data. In *Winter School of Computer Graphics '97*, pages 79–87, Plzen, Czech Republic, 1997.

[CPC84] R. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, pages 137–145, 1984.

[CS02] D. Chetverikov and D. Stepanov. Robust euclidean alignment of 3d point sets. In *Proc. First Hungarian Conference on Computer Graphics and Geometry*, pages 70–75, Budapest, 2002.

[CSK98] B. Csébfalvi and L. Szirmay-Kalos. Interactive volume rotation. *Machine Graphics and Vision*, 7(4):793–806, 1998.

[CSSD96] P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics*, 15(1):37–71, 1996.

[CW93] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. CA. Academic Press Professional, San Diego, 1993.

[dB25] Louis de Broglie. Recherches sur la théorie des quanta. *Annales de Physique*, 10:22–128, 1925.

[DBMS02] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Rendering Techniques 2002 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)*, June 2002.

[DDM02] Cyrille Damez, Kirill Dmitriev, and Karl Myszkowski. Global illumination for interactive applications and high-quality animations. Eurographics, September 2002. STAR - State of the Art Report.

[DLW93] Ph. Dutre, E. Lafortune, and Y. D. Willems. Monte Carlo light tracing with direct computation of pixel intensities. In *Compugraphics '93*, pages 128–137, Alvor, 1993.

[DS97] George Drettakis and Francois X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 57–64, 1997.

[DS99] Cyrille Damez and Francois Sillion. Space-time hierarchical radiosity. In *Rendering Techniques '99*, pages 235–246, New York, NY, 1999. Springer Wien.

[DSH01] Cyrille Damez, Francois X. Sillion, and Nicolas Holzschuch. Space-time hierarchical radiosity with clustering and higher-order wavelets. In *Proceedings of Eurographics 2001*, September 2001. Available on Computer Graphics Forum Volume 20 CD-ROM.

[Dut96] Ph. Dutre. *Mathematical Frameworks and Monte Carlo Algorithms for Global Illumination in Computer Graphics*. PhD thesis, Dept. Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven, Leuven, 1996.

[DW91] Mark A. Z. Dippe and Erling Henry Wold. Stochastic Sampling: Theory and Application. In George W. Zobrist, editor, *Progress in Computer Graphics*. Ablex Publishing, Norwood, NJ, 1991.

[DW94] Ph. Dutre and Y. D. Willems. Importance-driven Monte Carlo light tracing. In *Rendering Techniques'94 (Proceedings of the 5th Eurographics Workshop on Rendering)*, 1994.

[DW96] Ph. Dutre and Y. D. Willems. Potential-driven Monte Carlo particle tracing for diffuse environments with adaptive probability functions. In *Rendering Techniques '96*, pages 306–315, 1996.

[Far97] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, San Diego, 1997.

[FSZ98] Dieter Fellner, Stephan Schaefer, and Marco Zens. *Parallel Computing: Fundamentals, Applications and New Directions*, volume 12 of *Advances in Parallel Computing*, chapter Photorealistic Rendering in Heterogeneous Networks. Elsevier Science, 1998. Proceedings of Parallel Computing '97.

[FvDFH90] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, Mass., 1990.

[GD01] Xavier Granier and George Drettakis. Incremental updates for rapid glossy global illumination. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, volume 20, pages 268–277, September 2001.

[GDW00] Xavier Granier, George Drettakis, and Bruce Walter. Fast global illumination including specular effects. In B. Peroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 47–58, New York, NY, 2000. Springer Wien.

[Gla95] Andrew Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, Inc., San Francisco, 1995.

[Gor93] P. Cohen M. Hanrahan P. Gortler, S. Schröder. Wavelet radiosity. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 221–230, 1993.

[GSG90] David W. George, Francois X. Sillion, and Donald P. Greenberg. Radiosity Redistribution for Dynamic Environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990.

[GTG84] Cindy M. Goral, Kenneth E. Torrance, and Donald P. Greenberg. Modeling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, pages 213–222, 1984.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 212–222, July 1984.

[Hec91] P. S. Heckbert. *Simulating Global Illumination Using Adaptive Meshing.* PhD thesis, University of California, Berkeley, 1991.

[Hec92] P. S. Heckbert. Discontinuity meshing for radiosity. In *Third Eurographics Workshop on Rendering*, pages 203–226, 1992.

[Her91] Ivan Herman. *The Use of Projective Geometry in Computer Graphics.* Springer-Verlag, Berlin, 1991.

[HG83] Roy Hall and Donald P. Greenberg. A testbed for realistic image synthesis. *IEEE Computer Graphics and Applications*, 3(8):10–20, 1983.

[HMF98] M. Hyben, I. Martisovits, and A. Ferko. Scene complexity for rendering in flatland. In L. Szirmay-Kalos, editor, *Spring Conference on Computer Graphics*, pages 112–120, 1998.

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.

[ICG86] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 133–142, 1986.

[IWS02] Carsten Benthin Alexander Keller Ingo Wald, Thomas Kollig and Philipp Slusallek. Interactive global illumination. In P. Shirley G. Sakas and S. Müller, editors, *Rendering Techniques 2002 (Proceedings of the 13th Eurographics Workshop on Rendering)*, pages 9–20, 2002.

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. *ACM Computers and Graphics (SIGGRAPH '98 Proceedings)*, pages 311–320, 1998.

[Jen95] H. W. Jensen. Importance driven path tracing using the photon maps. In *Rendering Techniques '95*, pages 326–335, 1995.

[Jen96] H. W. Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30, 1996.

[JJHL77] F. E. Nicodemus J. C. Richmond J. J. Hsia, I. W. Ginsber and T. Limperis. Geometrical considerations and nomenclature for reflectance. *NBS Monograph (U. S. Dept. of Commerce)*, 1977.

[Kaj86] J. T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 143–150, 1986.

[KBSK03] Cs. Kelemen, B. Benedek, and L. Szirmay-Kalos. Bi-directional rays in global illumination. In *WSCG 2003 Conference, Posters, Plzen*, 2003.

[Kel97a] Alexander Keller. Instant radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 49–56, 1997.

[Kel97b] Alexander Keller. *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*. PhD thesis, University of Kaiserlautern, ISBN 3-8265-3330-5, 1997.

[Kel99] Alexander Keller. Hierarchical Monte Carlo image synthesis. Technical Report 298/99, Universität Kaiserslautern, AG Numerische Algorithmen, 1999. to appear in Mathematics and Computers in Simulation.

[KM99] T. Tawara K. Myszkowski, P. Rokita. Perceptually-informed accelerated rendering of high quality walkthrough sequences. In *Rendering Techniques'99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, pages 5–18, 1999.

[Kra89] G. Krammer. Notes on the mathematics of the PHIGS output pipeline. *Computer Graphics Forum*, 8(8):219–226, 1989.

[Kra03] G. Krammer. *Bevezetés a Számítógépi grafikába - Jegyzet*. ELTE, 2003. http://krammer.web.elte.hu/eltettk/grafika/jegyzet/.

[Lan91] B. Lantos. *Robotok Irányitása*. Akadémiai Kiadó, Budapest, Hungary, 1991. (in Hungarian).

[LW93] E. Lafortune and Y. D. Willems. Bi-directional path-tracing. In *Compugraphics '93*, pages 145–153, Alvor, 1993.

[LW00] Collins S. Leeson W., O'Sullivan C. EFFIGI. an efficient framework for implementing global illumination. In *Eighth International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2000)*, 2000.

[MA01] D. Cohen-Or S. Fleishman D. Levin C. Silva M. Alexa, J. Behr. Point set surfaces. In *IEEE Visualization*, pages 21–28, 2001.

[MPT99] Igancio Martin, Xavier Pueyo, and Dani Tost. Frame-to-frame coherent animation with two-pass radiosity. Technical Report IIiA 99-08-RR, Institut d'Informatica i Aplicacions, Universitat de Girona, Girona, Spain, June 1999.

[MRR+53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[NDR96a] J. Nimeroff, J. Dorsey, and H. Rushmeier. Implementation and analysis of a global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics*, 2(4), 1996.

[NDR96b] Jeffry Nimeroff, Julie Dorsey, and Holly Rushmeier. Implementation and analysis of an image-based global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):283–298, December 1996.

[Neu95] L. Neumann. Monte Carlo radiosity. *Computing*, 55:23–42, 1995.

[Neu01] A. Neumann. *Constructions of Bidirectional Reflection Distribution Functions*. PhD thesis, Institute of Computer Graphics and Algorithms, Technical University of Vienna, Wien, Austria, June 2001. Available from http://www.cg.tuwien.ac.at/research/theses.

[NFKP94] L. Neumann, M. Feda, M. Kopp, and W. Purgathofer. A new stochastic radiosity method for highly complex scenes. In *Proc. of the 5th. EG Workshop on Rendering*, 1994.

[NN89] L. Neumann and A. Neumann. Photosimulation: Interreflection with arbitrary reflectance models and illumination. *Computer Graphics Forum*, (8):21–34, 1989.

[NNB+96] A. Neumann, L. Neumann, P. Bekaert, Y. Willems, and W. Purgathofer. Importance-driven stochastic ray radiosity. In *Rendering Techniques '96*, 1996.

[NNPP98] L. Neumann, A. Neumann, J. Prikryl, and W. Purgathofer. The constant radiance term. *Machine Graphics Vision*, 7(3):535–549, 1998.

[NNSK98] L. Neumann, A. Neumann, and L. Szirmay-Kalos. New simple reflectance models for metals and other specular materials. Technical Report TR-186-2-98-17, Institute of Computer Graphics, Vienna University of Technology, 1998.

[NNSK99] L. Neumann, A. Neumann, and L. Szirmay-Kalos. Reflectance models with fast importance sampling. *Computer Graphics Forum*, 18(4):249–265, 1999.

[NPT+95] L. Neumann, W. Purgathofer, R. F. Tobler, A. Neumann, P. Elias, M. Feda, and X. Pueyo. The stochastic ray method for radiosity. In *Rendering Techniques '95*, pages 206–218, 1995.

[Pop87] Gy. Popper. *Bevezetés a végeselem-módszer matematikai elméletébe*. BME Mérnöktovábbképző Intézet, Budapest, 1987.

[Pov] PovRay. The persistence of vision raytracer. http://www.povray.org.

[PRE95] D. Play, J.F. Rigal, and T. Endrődy. Feature based geometric modelling and analysis of multibody mechanical system behaviour. *Periodica Polytechnica (UTBudapest, Ser.Mech.Eng)*, 39(2):131–15, 1995.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[RN00] Zs. Ruttkay and H. Noot. Solution strategies to produce facial animations. In *Proceedings of the ERCIM/CompulogWorkshop on Constraints*, San Padova, Italy, 2000.

[RV00] G. Renner and J. Vida. Reconstruction of free-from features by genetic algorithms. In *2000 International CIPRP Design Seminar*, pages 411–416, Haifa, Israel, 2000.

[RVW98] G. Renner, T. Várady, and V. Weiss. Reverse engineering of free-form features. In *PROLAMAT 98, CD proceedings*, Trento, 1998.

[SA97] M. Sbert and Brusi A. Comparing finite and biased infinite path length shooting random walk estimators for radiosity. In *Proc. Spring Conference on Computer Graphics (SCCG '97)*, Budmerice, Slovakia, 1997. Comenius University Press.

[SAWG91] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):187–198, 1991.

[Sbe96] Mateu Sbert. *The Use of Global Directions to Compute Radiosity*. PhD thesis, Catalan Technical University, Barcelona, 1996.

[Sbe00] M. Sbert. Optimal absorption probabilities for random walk radiosity. *Graphical Models*, 62:56–70, 2000.

[SDS95] F. Sillion, G. Drettakis, and C. Soler. Clustering algorithm for radiance calculation in general environments. In *Rendering Techniques '95*, pages 197–205, 1995.

[SGCH93] Peter Schroder, Steven J. Gortler, Michael F. Cohen, and Pat Hanrahan. Wavelet Projections for Radiosity. In *Fourth Eurographics Workshop on Rendering*, number Series EG 93 RW, pages 105–114, Paris, France, June 1993.

[Sil95] François X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 3(1), 1995.

[SK98] L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. Technical Report TR-186-2-98-21, Institute of Computer Graphics, Vienna University of Technology, 1998. http://www.cg.tuwien.ac.at.

[SK99a] L. Szirmay-Kalos. *Monte-Carlo Methods in Global Illumination*. Institute of Computer Graphics, Vienna University of Technology, Vienna, 1999. http: //www.iit.bme.hu/~szirmay/script.pdf.

[SK99b] L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum (Eurographics'99)*, 18(3):233–244, 1999.

[SK00] L. Szirmay-Kalos. *Photorealistic Image Synthesis with Ray-Bundles*. Hungarian Academy of Sciences, D.Sc. Dissertation, Budapest, 2000. http//www.iit.bme.hu/~szirmay.

[SK03] N. Magnenat-Thalmann S. Kshirsagar. Visyllable based speech animation. In *Computer Graphics Forum (Proceedings of Eurographics 2003)*, 2003.

[SKe95] L. Szirmay-Kalos (editor). *Theory of Three Dimensional Computer Graphics*. Akadémia Kiadó, Budapest, 1995. http://www.iit.bme.hu/~szirmay.

[SKF97] L. Szirmay-Kalos and T. Fóris. Radiosity algorithms running in subquadratic time. In *Winter School of Computer Graphics '97*, pages 562–571, Plzen, Czech Republic, 1997.

[SKFNC97] L. Szirmay-Kalos, T. Fóris, L. Neumann, and B. Csébfalvi. An analysis to quasi-Monte Carlo integration applied to the transillumination radiosity method. *Computer Graphics Forum (Eurographics'97)*, 16(3):271–281, 1997.

[SKSMT00] L. Szirmay-Kalos, M. Sbert, R. Martinez, and R.F. Tobler. Incoming first-shot for non-diffuse global illumination. In *Spring Conference of Computer Graphics '00*, 2000.

[Slu95] P. Slussalek. *Vision. An Architecture for Physically-Based Rendering*. PhD thesis, Universitat Erlangen-Nürnberg, Nürnberg,, 1995.

[Slu97] P. Slussalek. Photo-realistic rendering — recent trends and developments. In *Eurographics '97, STAR reports*, pages 35–57, 1997.

[Sou97] Bjarne Soustroup. *The C++ Programming Language*. Addison-Wesley, 3rd edition, 1997.

[SP94a] F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, Inc., San Francisco, 1994.

[SP94b] Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994.

[SSG+99] M. Stamminger, A. Scheel, A. Granier, F. Perez-Cazorla, G. Drettakis, and F. Sillion. Efficient glossy global illumination with interactive viewing. In *Graphics Interface'99, Kingston, Ontario*, 1999.

[SSG+00] Marc Stamminger, Annette Scheel, Xavier Granier, Frederic Perez-Cazorla, George Drettakis, and Francois Sillion. Efficient glossy global illumination with interactive viewing. *Computer Graphics Forum*, 19(1):13–25, 2000.

[Suy02] Frank Suykens. *On robust Monte Carlo algorithms for multi-pass global illumination*. PhD thesis, Dept. Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven, Leuven, 2002.

[SW99] F. Suykens and Y. D. Willems. Weighted multipass methods for global illumination. *Computer Graphics Forum*, 18(3):209–220, 1999.

[Sza95] Zs. Szalavári. Rendering natürlicher atmosphärisher lichteffecte. Technical report, http://www.cg.tuwien.ac.at/research/rendering/halos, 1995.

[TM93] R. Troutman and N. L. Max. Radiosity algorithms using higher order finite element methods. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 209–212, 1993.

[Tob98] R. F. Tobler. ART — Advanced Rendering Toolkit. Technical report, 1998. http://www.cg.tuwien.ac.at/ reseach/ rendering/ ART.

[TPWG02] Parag Tole, Fabio Pellicini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002 Annual Conference)*, 21(3):537–546, 2002.

[UH99] Tushar Udeshi and Charles D. Hansen. Towards interactive photorealistic rendering of indoor scenes: A hybrid approach. In D. Lischinski and G. W. Larson, editors, *Rendering Techniques '99*, pages 63–76, New York, NY, 1999. Springer Wien.

[Vea97] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, December 1997. Available from http://graphics.stanford.edu/papers/veach_thesis.

[VG94] E. Veach and L. Guibas. Bidirectional estimators for light transport. In P. Shirley G. Sakas and S. Müller, editors, *Photorealistic Rendering Techniques (Proceedings of the fifth Eurographics Workshop on Rendering*, pages 147–162, 1994.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.

[VG97] E. Veach and L. Guibas. Metropolis light transport. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 65–76, 1997.

[Vid93] J. Vida. *Integration of Blends into a Solid Modeller*. PhD thesis, Computer and Automation Institute (SZTAKI), Budapest, 1993.

[VMC97] T. Várady, R. R. Martin, and J. Cox. Reverse engineering of geometric models - an introduction. *Computer-Aided Design*, 29(4):255–269, 1997.

[Wat99] Alan Watt. *3D Computer Graphics*. Addision-Wesley, 3rd edition, 1999.

[WBS02] I. Wald, C. Benthin, and P. Slussalek. Interactive global illumination using fast ray tracing. In *13th Eurographics Workshop on Rendering*, 2002.

[WDG02] Bruce Walter, George Drettakis, and Donald P. Greenberg. Enhancing and optimizing the render cache. In *Rendering Techniques 2002 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)*, June 2002.

[Wil01] Alexander Wilkie. *Photon Tracing for Complex Environments*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Wien, Austria, April 2001. Available from http://www.cg.tuwien.ac.at/research/theses/.

[XSG00] Changyu Xing, Jizhou Sun, and R. L. Grimsdale. Accelerated incremental radiosity algorithm. *Journal of Computer Science and Technology*, 15(1):47–55, 2000.

[Yee00] Yang Li Hector Yee. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, August 2000.

[Zat93] Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. In *Computer Graphics, Annual Conference Series*, pages 213–220, 1993.

[ZBP99] J. Zaninetti, P. Boy, and B. Peroche. An adaptive method for area light sources and daylight in ray tracing. *Computer Graphics Forum*, 18(3):139–150, 1999.