2366 S

Stellingen

behorende bij het proefschrift

Ray Tracing and Radiosity Algorithms for Photorealistic Image Synthesis

A.J.F. Kok

3 mei 1994

1. Hoewel nauwkeurige modellering en texture mapping op het eerste gezicht belangrijker lijken voor het verkrijgen van realisme, mag globale belichting niet genegeerd worden.
2. De voor hardware implementatie noodzakelijke algoritmische vereenvoudiging van radiosity kan uiteindelijk leiden tot een lagere overall performance dan van een software implementatie.
3. Voor het maken van fotorealistische afbeeldingen van bestaande scenes is de camera nog altijd sneller dan de computer.
4. Gezien het belang van een goede belichting in veel toepassingen, zou men meer interesse voor radiosity methoden mogen verwachten.
5. Voor realistische beeldsynthese zijn twee zaken van groot belang: de modellering van scenes en de berekening van de belichting daarin. Er is nog weinig onderzoek gedaan naar de interactie tussen beide.
6. Gezien de toename van het aantal dubbeldeksrijtuigen en de beenruimte daarin, lijken de Nederlandse Spoorwegen te anticiperen op een contract met het Ministerie van Onderwijs en Wetenschappen over een OV-kleuterkaart.
7. Door de verbetering van de infrastructuur in en naar natuurgebieden worden deze gebieden steeds minder aantrekkelijk voor echte wandelaars.
8. Zolang er sporten op het programma staan waarbij de uitslag bepaald wordt door een jury, zullen de Olympische Spelen nooit een eerlijk verloop hebben.
9. De verhoging van de tarieven van het openbaar vervoer zijn wel enigszins te rechtvaardigen. Door alle vertragingen maakt men steeds langer gebruik van bus en trein.
10. Bugs in computer graphics programmatuur leiden tot de meest fantastische plaatjes.

[illegible]
[illegible]
TR diss [illegible]

**TR diss
2366**

RAY TRACING AND RADIOSITY ALGORITHMS
FOR PHOTOREALISTIC IMAGE SYNTHESIS

RAY TRACING AND RADIOSITY ALGORITHMS FOR PHOTOREALISTIC IMAGE SYNTHESIS

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op dinsdag 3 mei 1994 te 13.30 uur
door

Adrianus Johannes Franciscus KOK

informatica ingenieur
geboren te Breda

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. F.W. Jansen

Published and distributed by:

Delft University Press
Stevinweg 1
2628 CN Delft
The Netherlands

Telephone +31 15 783254
Fax +31 15 781661

ISBN 90-6275-981-5 / CIP

Copyright © 1994 by A.J.F. Kok

All rights reserved.
No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without permission from the publisher: Delft University Press, Stevinweg 1, 2628 CN Delft, The Netherlands.

Printed in the Netherlands

Preface

Realistic image synthesis is an important research direction in computer graphics. The goal is to generate images that not only look realistic, but that are also physically correct. It is a challenging task to design and implement a system for realistic image synthesis.

This thesis is the result of a Ph.D. research project carried out by the author at the Faculty of Technical Mathematics and Informatics of the Delft University of Technology. The research was directed at developing methods for realistic image synthesis. Physical correctness, accuracy, and efficiency were the keywords during the developments of these methods. This thesis reports the results of this research.

During this project several people have contributed directly or indirectly to my work. I would like to thank all these people.

First I would like to thank my promotor, professor Erik Jansen, for giving me the opportunity to do this research, and for the ideas and advise he gave me during this research.

Further I would like to thank the Ph.D. students of the sections Computer Graphics and CAD/CAM, for the discussions we had concerning research and other matters. Theo van Walsum and Wim de Leeuw contributed to the software. I would also like to thank the technical support personnel. They helped when assistance was needed.

I would also like to thank the participants in the "Radiosity Engine" project at the Network Theory section at the faculty of Electrical Engineering: Ed Deprettere, Theo Verelst, Li-Shen Sheng and Gerben Heckstra. We had several discussions about radiosity and ray tracing methods. They tested programs, and made several test models.

Last but not least, I would like to thank my parents for their care and support.

Arjan Kok

Contents

1 Introduction 1
    1.1 Realistic image synthesis 1
    1.2 Global illumination 2
    1.3 Solution methods 3
    1.4 Goals 5
    1.5 Overview thesis 5
2 Global illumination 7
    2.1 Introduction 7
    2.2 Terminology for global illumination 7
        2.2.1 Diffuse and specular reflection 9
    2.3 The rendering equation 10
    2.4 Ray tracing 12
    2.5 Radiosity 17
        2.5.1 Full-Matrix radiosity 19
        2.5.2 Progressive radiosity 20
        2.5.3 Form factor computation 21
        2.5.4 Meshing 23
        2.5.5 Particle tracing 24
    2.6 Hybrid global illumination methods 25
    2.7 Conclusions 29
3 A global illumination method with source selection 33
    3.1 Introduction 33
    3.2 Source selection in a hybrid global illumination method 34
        3.2.1 Low-frequency pass 35
        3.2.2 High-frequency pass 38
    3.3 Selection criteria 39
        3.3.1 Global selection criteria 39
        3.3.2 Local selection criteria 40
    3.4 Results and conclusions 44

4 The shadow image buffer 49
- 4.1 Introduction 49
- 4.2 Shadow ray culling methods 49
- 4.3 Shadow coherence 51
- 4.4 Shadow image buffer 51
- 4.5 Results and conclusions 55

5 Adaptive light source sampling 61
- 5.1 Introduction 61
- 5.2 Sampling area light sources 61
- 5.3 An implementation of adaptive stochastic source sampling 64
- 5.4 Sampling pattern coherence 71
- 5.5 Results and conclusions 73

6 Hierarchy within the radiosity process 77
- 6.1 Introduction 77
- 6.2 Hierarchical meshing 77
- 6.3 A hierarchy for the radiosity method 80
- 6.4 Implementation of adaptive receiving levels 80
- 6.5 Results and conclusions 83

7 Grouping of patches 87
- 7.1 Introduction 87
- 7.2 Motivation for grouping 87
- 7.3 Grouping patches 88
- 7.4 Transfer functions 90
  - 7.4.1 Transfer functions for undirected shooting 91
  - 7.4.2 Transfer functions for directed shooting 94
- 7.5 Results and conclusions 96

8 Conclusions and future research 101
- 8.1 Introduction 101
- 8.2 A global illumination system for realistic image synthesis 101
  - 8.2.1 Improvement rendering pass 102
  - 8.2.2 Improvement radiosity pass 104
- 8.3 Ideas for future research 105
  - 8.3.1 Improvement of algorithms 105
  - 8.3.2 Multiprocessor and specialized hardware implementations 106

Bibliography 109
Appendix A Description of test models 115
    A.1 Model A, triangle 115
    A.2 Model B, small room 116
    A.3 Model C, computer room 116
Summary 123
Samenvatting 125
Curriculum vitae 127

1 Introduction

1.1 Realistic image synthesis

One of the most interesting areas in computer graphics is high-quality rendering of complex scenes, also known as realistic image synthesis. The goal of realistic image synthesis is to generate images that look real, are physically correct, and represent all light reflection effects. Ideally, when an environment is built in real life, and an image is made of a model of this environment, then the image and the view on the real-life environment should be the same. Another often used definition of realism is photorealism: a photo of the real scene and a computer generated image of a model of the scene should be the same. It is a very complicated task to design systems that are capable of generating images that achieve this goal. However, a lot of research has been carried out, and better and better images are generated.

Realistic image synthesis can be used for several applications. The most important ones can be found in architecture and lighting design. Architects and lighting designers have a strong interest in modelling and simulating the illumination in the environment they have in mind. Based on the results of these simulations, they can decide where windows could be placed best in a building, or where and which lights and light fixtures should be used to get an optimal lighting in the building. It is also important to decide which materials and colors are used for wall, floors and ceilings. The Radiance Imaging System (Ward 92) is an example of a global illumination system that is developed for lighting design. Another application of realistic rendering is the simulation of the visibility at roads and of road signs at different parts of the day and season, and in different weather conditions (Clavé and Gross 91). Realistic image synthesis can also be used to create special effects for movies, for computer art, flight simulators, computer aided design, etc.

Two aspects are important for generating realistic images:

- Modelling.
  The scene of which an image must be generated should be modelled very accurately. It is impossible to make a realistic looking image from a too simply modelled scene. One of the requirements on the modelling is that it is not restricted to polygons only. It should also be possible to use curved surfaces. Modelling for realistic image synthesis is mainly an unsolved problem yet.

- Illumination.
  Illumination is important for the appearance of the objects. Some (parts of) objects are illuminated, others are not. Moreover, when an object receives light, it does not receive an equal amount of illumination everywhere. In real life several optical

effects can occur that influence the illumination of an environment. A realistic imaging system should be capable of simulating all these optical effects. The computation of illumination is the main topic of this thesis.

1.2 Global illumination

Illumination can be split into two parts: local illumination and global illumination. These two types of illumination are closely related.

Light can travel a long path before it reaches a surface and eventually the eye. It may be reflected by many other surfaces after it was emitted by a light source. Light that reaches a surface from a light source without being reflected by other surfaces is called direct illumination. Light that reaches a surface after one or more reflections is called indirect illumination (figure 1.1).

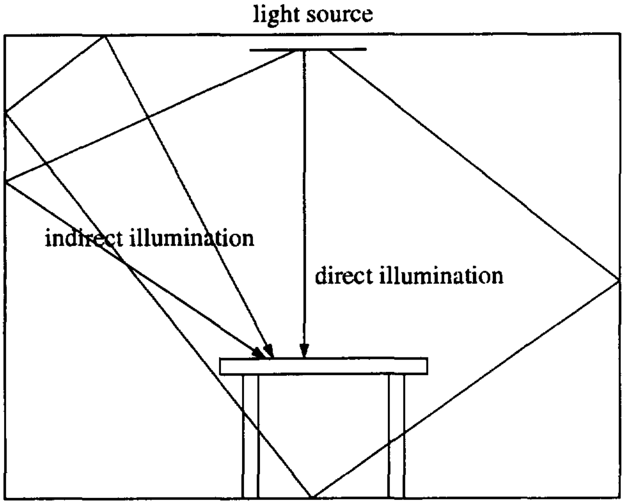


Figure 1.1 Direct and indirect illumination

Local illumination models describe how light that impinges on a surface is reflected. It describes how much light is reflected and/or transmitted and in what directions. It ignores where the light is coming from. Local illumination does only take into account the direct illumination. Moreover it assumes that surfaces are illuminated with a few point light sources and it ignores that light from these sources can be blocked by other surfaces. Rendering algorithms that only compute local illumination can generate images of realistic looking objects, but not of complex environments with many objects.

Global illumination takes into account where the illumination comes from, and that the shading of a surface is dependent on the illumination of all other surfaces that can be seen from the surface.

Global illumination models enable us to simulate the visual effects, that occur in real life, and that can not be simulated with local illumination models. Examples of these visual effects are:

- Soft shadows.
  Area light sources generate penumbras (soft shadows), and not the sharp shadows that were generated before global illumination was used.
- Indirect illumination.
  Not all parts of an environment are illuminated directly by light sources. Often these areas are not totally dark. These areas are illuminated by indirect illumination. Light is reflected by other parts of the scene before it reaches these areas. In closed environments the effect of indirect illumination is often considerable.
- Color bleeding.
  Color bleeding is one of the effects of indirect illumination. The color of an illuminated object is visible on other objects, when these objects receive reflected light from the illuminated object.
- Caustics.
  Caustics are generated when light is reflected in a mirror or metal object or when it is transmitted (refracted) through a transparent object. An example of a caustic is a highlight that occurs when light is refracted in a glass of water.
- Participating media.
  The environment between objects in a scene is not always a vacuum. Often some kind of medium or atmosphere exists. Such a medium may contain many small particles (molecules) that can absorb or scatter the light that traverses the medium. Examples of media are air, smoke and water. Usually air contains very little particles so it can often be considered as a vacuum, but water and smoke contain many particles that influence the overall shading of the scene.

1.3 Solution methods

Standard projective algorithms (depth-buffer algorithms, scanline algorithms) only account for the local (direct) illumination, and they usually ignore shadows. Only light that is reflected once before it reaches the viewer is computed. These algorithms are therefore not suited for realistic image synthesis.

The global illumination problem can be solved with two methods: radiosity and ray tracing.

The radiosity method (Goral et al 84; Nishita and Nakamae 85; Cohen and Greenberg 85) computes the illumination within a scene by calculating the power exchange between all surfaces in the scene. The surfaces are subdivided into patches. The radiosity method computes the illumination (radiance) for each patch. The radiance of a patch is assumed to be constant. The radiances are computed by determining relations between all pairs of patches. Such a relation (called form factor) describes the amount of energy that is exchanged between the two patches. All the relations together form a system of equations. Solving these equations gives the radiance values for all patches. All patches in the scene can now be projected on the screen with their computed radiance value as color. Solving all interreflections between the patches is an expensive process. For scenes with a moderate complexity the radiosity process may take hours on a standard workstation. The radiosity method is only applicable for patches that have a matte (diffuse) reflection. It cannot handle adequately mirroring reflections. Moreover the result is dependent on the subdivision into patches: the more patches, the better the result.

Ray tracing (Whitted 80; Glassner 89) is a rendering method that computes the pixel intensity by following one or a few rays from the viewpoint through the pixel into the scene. The first object that is intersected by the ray is visible on the pixel. The radiance at the intersection point is computed by following rays from this point to all light sources. These rays to light sources are called shadow rays. If no objects are intersected by such a shadow ray, then the point is illuminated by the light source and the radiance of the point caused by the light source can be computed using a shading model. When the intersected point was mirroring or transparent, one or more secondary rays are traced into the reflection or transmission direction and the tracing process is continued. The main disadvantage of standard ray tracing is that it does not account for the diffuse interreflection between the surfaces in the scene. This method can be extended to account for this interreflection, but only at very high costs.

Both radiosity and ray tracing solve parts of the global illumination, but neither of them is able to solve the global illumination problem efficiently. Hybrid methods, that combine radiosity and ray tracing, are able to simulate all types of interreflection (Wallace et al 87; Sillion and Puech 89). A standard two-pass method consists of a radiosity pass and a rendering pass that is based on ray tracing. The radiosity pass computes the interreflections between the patches. The results are the radiance values for all patches. During the rendering pass the illumination of a visible object is computed from the radiance values for that object that are calculated during the radiosity pass. Mirroring reflection is added by tracing secondary rays in the reflection direction.

The main disadvantage of the standard two-pass method is that the surfaces must be subdivided into patches during the radiosity pass. The quality of the image strongly depends on the subdivision that is applied. With a coarse subdivision the shadows in the image will be very soft, although sharp shadows are expected. A fine subdivision makes the radiosity process give better results but is expensive, as for each generated patch many form factors must be computed.

1.4 Goals

The main goal of this research is to improve methods for realistic image synthesis. Such methods must comply with the following requirements (in order of descending importance):

- The ability to solve the complete global illumination.
  All possible paths that light can travel before it reaches the eye should be covered. All optical effects must be accounted for. This is the most important requirement to be able to make realistic images.
- The accuracy of the solution.
  Sometimes it is possible to simulate all light paths, but is it impossible to compute them very accurately. This will for example result in shadows that are not totally correct. The shape of the shadow is not exactly what is expected. The goal is to be as accurate as possible.
- The efficiency.
  Ideally, images should be generated in real-time or at an interactive speed. However, because of the complex nature of global illumination, this is not possible. We can strive to generate images as fast as possible. Efficiency techniques can be applied as long as they do not decrease image quality .

An optimal combination of radiosity and ray tracing methods should be developed that will give the most accurate image possible. Both radiosity and ray tracing are computationally expensive processes. Therefore, after developing a new hybrid radiosity and ray tracing method, efficiency methods will be developed that speed up the realistic image generation problem.

1.5 Overview thesis

This thesis is organised as follows.

The theory of global illumination is described in chapter 2. Several methods to compute the global illumination in a scene and to make images of this scene are described and compared to each other based on how well they solve the total global illumination problem (the ability to compute all possible light paths), the shading accuracy, and the processing time.

Chapter 3 describes a hybrid radiosity and ray tracing method that separates the high-frequency from the low-frequency illumination. High-frequency illumination is responsible for perceptible shading contrast. Low-frequency illumination is computed during a radiosity pass. The high-frequency illumination is computed by shadow-ray casting during the (ray tracing based) rendering pass. To identify sources that are responsible for high-frequency illumination a method called source selection has been developed. Several selection criteria can be applied. Based on how strict these criteria are applied, this method can perform like all hybrid global illumination methods, ranging from a standard two-pass method, that can generate reasonable accurate images in a limited amount of time, to a Monte-Carlo sampling approach, that generates very accurate images in a large amount of time.

The method described in chapter 3 can be rather expensive concerning computation times, both for the rendering pass as for the radiosity pass. The most expensive part of the rendering pass is casting shadow rays to important sources to capture the illumination of sources to a point. A first method to reduce the number of shadow rays is the use of shadow coherence: shadows are visible on coherent parts of the screen. The shadow image buffer method that takes advantage of shadow coherence is described in chapter 4. Another method to reduce the number of shadow rays is to sample area light sources adaptively with shadow rays. Chapter 5 describes an adaptive stochastic source sampling method. This method is further improved by sampling pattern coherence.

The main difficulty in the radiosity process is the determination of the meshing on the surfaces (the subdivision of surfaces into patches). The size of the patches determines the accuracy of the solution. The smaller the patches, the better are the results, but also the longer is the computation time. Therefore it is important to find an optimum meshing for a pair of surfaces when the interreflections between these two surfaces must be computed. When surfaces are very small and form together one object, it is sometimes useful to compute the interreflections between objects instead of between the individual patches. A hierarchy of objects and patches can strongly improve the efficiency of the radiosity computations. Chapter 6 describes such a hierarchy and describes methods to automatically subdivide patches to an appropriate size when they receive energy from a source. Chapter 7 describes another part of the hierarchy: groups. Groups are clusters of patches that receive and shoot energy as one item instead of as individual patches.

Chapter 8 evaluates the issues explored in this thesis. Conclusions are drawn and ideas for future developments are given.

2 Global illumination

2.1 Introduction

A very important factor in generating realistic images is the ability to accurately simulate the global illumination in a scene. In contrast to local illumination, that is only concerned with the direct illumination from light sources, global illumination also includes indirect illumination. Indirect illumination is the illumination after one or more reflections or transmissions.

Methods that solve the global illumination problem should take into account all possible paths that light (or photons) can travel before it reaches the viewer. While travelling through the scene light may be scattered when it traverses a participating medium, e.g. smoke. However, in this thesis only non-participating media are considered, and so scattering is ignored. When light reaches a surface, it can be absorbed, reflected or transmitted, depending on the physical properties of the surface. It may take many reflections and transmissions before light reaches the viewer.

This chapter describes methods to make realistic images of scenes by computing the global illumination within these scenes. Before describing methods that can tackle the global illumination problem, some background will be given. Section 2.2 describes the most important quantities that are used in global illumination. In section 2.3 the formulas are given that are the basis for all global illumination methods. Also a number of important light transport mechanisms are explained. Ray tracing methods, as described in section 2.4, and radiosity methods, as described in section 2.5, are algorithms that are capable of simulating most of the light paths. However, combinations of radiosity and ray tracing perform best. These hybrid methods are described in section 2.6. Finally, section 2.7 summarizes and compares all methods.

2.2 Terminology for global illumination

To describe the physics of global illumination methods, the illuminating engineering terminology (ANSI 86) will be used. A good survey of the use of this terminology in computer graphics can be found in (Shirley 90b). A summary of the most important quantities for computer graphics is given below. Note that all quantities are a function of wavelength λ. However, for conciseness this wavelength is ignored in the following description.

- Radiant flux or radiant power Φ [W].
  Radiant power is defined as the radiant energy per unit time.

- Radiance L [W sr^{1} m^{-2}].

  Radiance is the basic unit for describing the radiation of a point on a surface. In computer graphics literature radiance is often called intensity. Radiance is defined as the radiant power leaving, passing through, or arriving at an element of the surface surrounding the point, per solid angle and per area of the orthogonal projection of the element of the surface on a plane perpendicular to the given direction (ANSI 86):

$$L = \frac{d^2\Phi}{d\omega dA \cos\theta} \tag{2.1}$$

- Irradiance E [W m^{-2}].

  Irradiance is defined as the density of radiant power incident on a surface:

$$E = \frac{d\Phi}{dA} \tag{2.2}$$

- Bidirectional reflectance distribution function (BRDF) f_r [sr^{-1}].

  The bidirectional reflection distribution function is defined as the ratio of differential radiance reflected in a given direction to the differential power density incident from a given direction:

$$f_r(\Theta_i, \Theta_r) = \frac{dL_r(\Theta_r)}{dE_i(\Theta_i)} = \frac{dL_r(\Theta_r)}{dL_i(\Theta_i) d\omega_i \cos\theta_i} \tag{2.3}$$

where $\Theta_i = (\theta_i, \phi_i)$[1] is the incident direction, $\Theta_r = (\theta_r, \phi_r)$ the reflection (outgoing) direction, and $d\omega_i$ the solid angle of the incident radiance (figure 2.1).

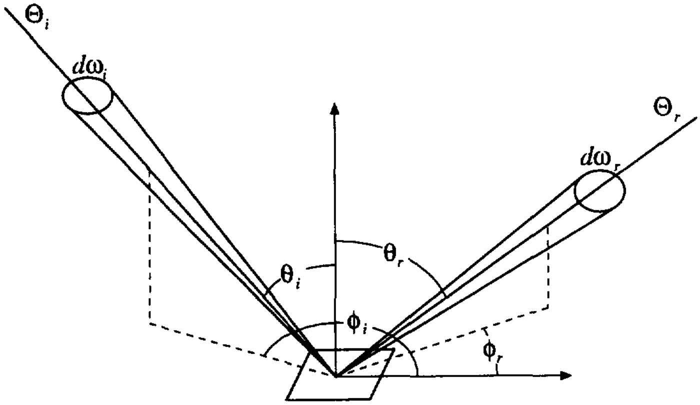


Figure 2.1 Geometry for bidirectional reflection distribution function

---

[1] A direction Θ can be defined with two angles, polar angle θ, and azimuth angle ϕ. For conciseness, mostly Θ is used instead of these two angles.

- Reflectance ρ [dimensionless].
  Reflectance is defined as the ratio of reflected flux to incident flux:

  $$\rho = \frac{\Phi_r}{\Phi_i} \tag{2.4}$$

  It is often easier to use reflectance instead of BRDF.

- Like the bidirectional reflection distribution function f_r and reflectance ρ to describe reflections, there exist the corresponding bidirectional transmittance distribution function f_t and transmittance τ to describe transmission. In the description of global illumination methods transmission is further ignored. However, transmission can be computed in almost the same way as reflection.

2.2.1 Diffuse and specular reflection

The BRDF of a surface can be very complicated (figure 2.2) (Nicodemus et al 77). Therefore often reflection is divided into two types: diffuse and specular reflection.

For a perfectly diffuse (Lambertian) surface the reflected radiance is constant in all directions (figure 2.3). A diffusely reflecting surface looks the same under all viewing directions. A diffuse reflector looks matte. For a diffuse reflector the relation between BRDF and reflectance is given by:

$$f_r(\theta_i,\phi_i;\theta_r,\phi_r) = \frac{\rho_d}{\pi} \tag{2.5}$$

For an ideal specular surface radiance is reflected in only one (specular) direction (figure 2.4). Examples of specular reflectors are mirrors. The following relations exist between the incident and reflected direction:

$$\theta_r = \theta_i \, , \, \phi_r = \phi_i + \pi \tag{2.6}$$

For other reflection directions the BRDF is zero. Often, however, light is not reflected in one direction, but in a small cone of directions (figure 2.5). This kind of reflection is called rough-specular reflection.

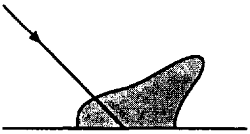


Figure 2.2 General reflection

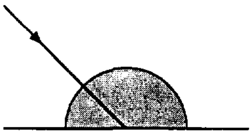


Figure 2.3 Diffuse reflection

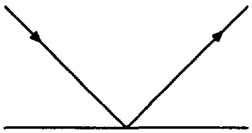


Figure 2.4 Specular reflection

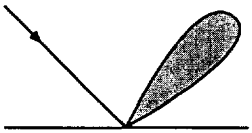


Figure 2.5 Rough specular reflection

Several methods to evaluate the BRDF and to separate the general BRDF into a diffuse and a specular component have been used in computer graphics (Phong 75; Blinn 77; Cook and Torrance 82; He et al 91). However, not all of them are physically correct. A good survey of computing reflections in computer graphics is given in (Hall 89).

2.3 The rendering equation

The theoretical background for all global illumination methods is given by the rendering equation. The outgoing radiance L_o at a surface point **x** in a certain outgoing direction Θ_o is equal to the emitted radiance L_e plus the reflected radiance L_r in that direction:

$$L_o(\mathbf{x},\Theta_o) = L_e(\mathbf{x},\Theta_o) + L_r(\mathbf{x},\Theta_o) \tag{2.7}$$

Using the definition of the BRDF (equation 2.3), the reflected radiance can be written as:

$$L_r(\mathbf{x},\Theta_r) = \int_{\Omega_i} f_r(\mathbf{x},\Theta_i,\Theta_r) L_i(\mathbf{x},\Theta_i) \cos\theta_i \, d\omega_i \tag{2.8}$$

where Ω_i is the hemisphere of all incoming directions. Combining equations 2.7 and 2.8 gives an equation that forms the basis of all global illumination methods, the *rendering equation*:

$$L_o(\mathbf{x},\Theta_o) = L_e(\mathbf{x},\Theta_o) + \int_{\Omega_i} f_r(\mathbf{x},\Theta_i,\Theta_o) L_i(\mathbf{x},\Theta_i) \cos\theta_i \, d\omega_i \tag{2.9}$$

The incoming radiance at point **x** from the direction Θ_i is equal to the outgoing radiance at point $\mathbf{x}'$, the point that is seen from **x** into direction Θ_i, into the outgoing direction Θ'_o ($\Theta_i = \Theta'_o$). This will give the following form of the rendering equation:

$$L_o(\mathbf{x},\Theta_o) = L_e(\mathbf{x},\Theta_o) + \int_{\Omega_i} f_r(\mathbf{x},\Theta_i,\Theta_o) L_o(\mathbf{x}',\Theta_i) \cos\theta_i \, d\omega_i \tag{2.10}$$

Instead of integrating over the hemisphere around **x**, it also possible to integrate over all surfaces in the scene, as was done for the well known rendering equation of (Kajiya 86). Using the relation for the projected solid angle:

$$d\omega_i = \frac{\cos\theta'_o \, dA'}{\|\mathbf{x}' - \mathbf{x}\|^2} \tag{2.11}$$

where θ'_o is the angle between the outgoing direction and the normal at point $\mathbf{x}'$, and dA' is a differential area around $\mathbf{x}'$, the rendering equation can be rewritten as:

$$L_o(\mathbf{x},\Theta_o) = L_e(\mathbf{x},\Theta_o) + \int_{all\,\mathbf{x}'} v(\mathbf{x},\mathbf{x}') f_r(\mathbf{x},\Theta_i,\Theta_o) L_o(\mathbf{x}',\Theta_i) \cos\theta_i \frac{\cos\theta'_o \, dA'}{\|\mathbf{x}' - \mathbf{x}\|^2} \tag{2.12}$$

or when written with only outgoing directions using $\Theta_i = \Theta'_o$:

$$L_o(\mathbf{x},\Theta_o) = L_e(\mathbf{x},\Theta_o) + \int_{all\,\mathbf{x}'} v(\mathbf{x},\mathbf{x}') f_r(\mathbf{x},\Theta_o',\Theta_o) L_o(\mathbf{x}',\Theta_o') \cos\theta_i \frac{\cos\theta_o' dA'}{\|\mathbf{x}'-\mathbf{x}\|^2} \quad (2.13)$$

In these equations $v(\mathbf{x},\mathbf{x}')$ is a visibility term that has the value one when $\mathbf{x}$ is visible to $\mathbf{x}'$, and zero otherwise.

These rendering equations should be the basis for all methods that solve the global illumination problem.

The rendering equation can be evaluated recursively until a light emitting source is reached. Therefore the radiance of a point can be computed by following the light backwards via reflections until a light source is reached. However, it is also possible to follow the light into the opposite direction, so from the light source to the surfaces in the scene. No matter what direction is followed, a system for realistic image synthesis should be able to simulate all possible paths light can travel between a light source and the viewer. Figure 2.6 shows several of these light paths. In this case an ideal situation is assumed, where a surface is either ideal diffusely or ideal specularly reflecting.

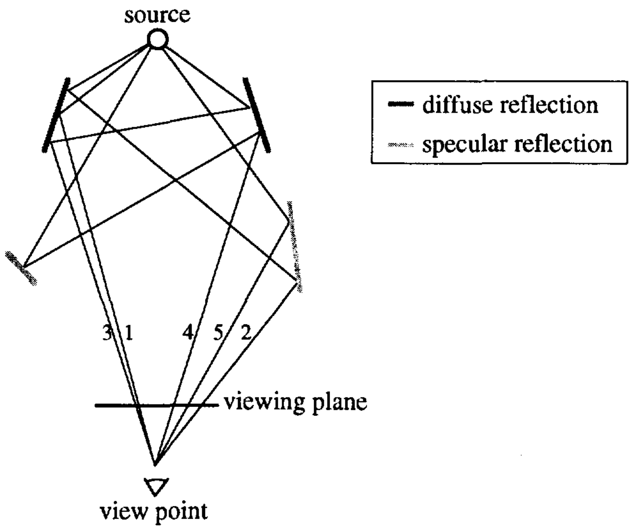


Figure 2.6 Some light transport paths

Several light transport mechanisms are shown in this figure. Path 1 represents the light that reaches the eye after only one diffuse reflection. This direct diffuse light transport is very important because the direct contribution of a light source to a surface will normally be the main contribution. This contribution is responsible for the most obvious shadows. Path 2 represents the diffuse-specular light transport. The light is first diffusely reflected and, before it reaches the eye, it is specularly reflected. An example of this light path is when one sees a direct illuminated object in a mirror. Light path 3 shows the diffuse-interreflection light transport. Light that is diffusely reflected by a surface is again diffusely reflected by (another) surface before it reaches the eye. This diffuse interreflection is responsible for the color bleeding. Although this diffuse interreflection light transport is often ignored, it is very important, because it may be responsible for a significant part of the total illumination in a scene. Path 4 represents the specular-diffuse light transport. Light is specularly reflected, for example by a mirror, onto a diffusely reflecting surface. This path is responsible for caustics, highlights on a diffuse surface because of specular reflection. Path 5 represents light that is only specularly reflected. This path may result in highlights.

2.4 Ray tracing

Ray tracing (Whitted 80) was the first rendering technique that did not only account for the local illumination, but also included some global illumination effects. An image is generated by following the light backwards into the scene (figure 2.7). Rays are traced from the eye through the viewing plane into the scene to find the first surface intersected by the ray. If this surface is (partially) specularly reflecting or transmitting, then secondary rays are traced in the reflected and transmitted directions. To calculate the direct diffuse reflection contribution at the intersection point, shadow rays are cast to all light sources in the scene. If no intersection of a shadow ray and the objects in the scene is found, then the intersection point is lit by the light source, and so the source contributes to the shading at the intersection point. An extensive description of ray tracing and methods to speed up ray tracing can be found in (Glassner 89). This section will further concentrate on the possibilities of the different ray tracing methods to compute the global illumination.

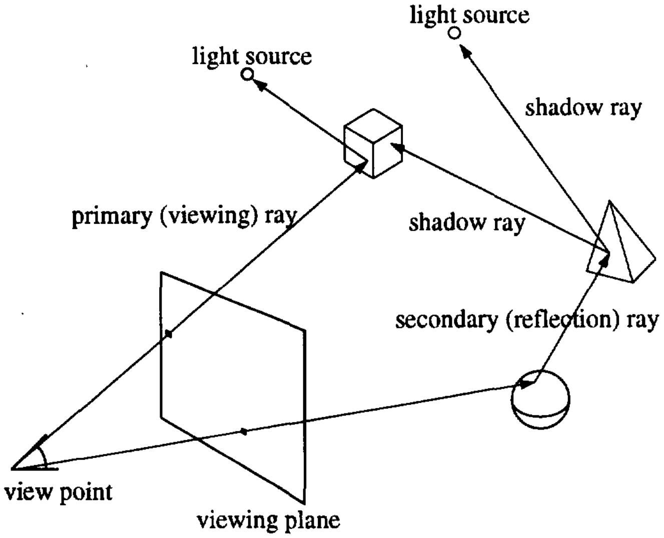


Figure 2.7 Ray tracing

It is possible to describe the ray tracing process in terms of the rendering equation. The radiance $L(\mathbf{p},\Theta_p)$ at pixel **p** is computed by tracing a ray from the eye into the viewing direction from the eye to the pixel. This ray will hit a patch at a point **x**. So the radiance of the pixel equals the outgoing radiance at point **x** into the opposite viewing direction:

$$L(\mathbf{p},\Theta_p) = L_o(\mathbf{x},\Theta_o)$$

$$\Theta_p = -\Theta_o \tag{2.14}$$

This outgoing radiance is computed by a simplified rendering equation. Instead of sampling all directions to get the incoming radiance, as indicated by equation 2.9, only the directions to the light sources and the specular directions (if the hit patch is specular) are sampled with rays, giving the following ray tracing equation:

$$\begin{aligned} L_o(\mathbf{x},\Theta_o) = {} & L_e(\mathbf{x},\Theta_o) + \\ & \sum_L \int_{all\,\mathbf{x}_l \in L} v(\mathbf{x},\mathbf{x}_l) f_{r,d}(\mathbf{x}) L_e(\mathbf{x}_l,\Theta'_o) \cos\theta_l \, d\omega_l + \\ & \int_{\theta_s \in \Omega_s} f_{r,s}(\mathbf{x},\Theta_s,\Theta_o) L(\mathbf{x}_s,\Theta_s) \cos\theta_s \, d\omega_s + \\ & \rho_d(\mathbf{x}) L_a(\mathbf{x}) \end{aligned} \tag{2.15}$$

The second term at the right hand side of this equation computes the direct diffuse contribution of the light sources L. Shadow rays are cast to evaluate the visibility term v. The third term accounts for the specular contribution. Secondary (reflection) rays

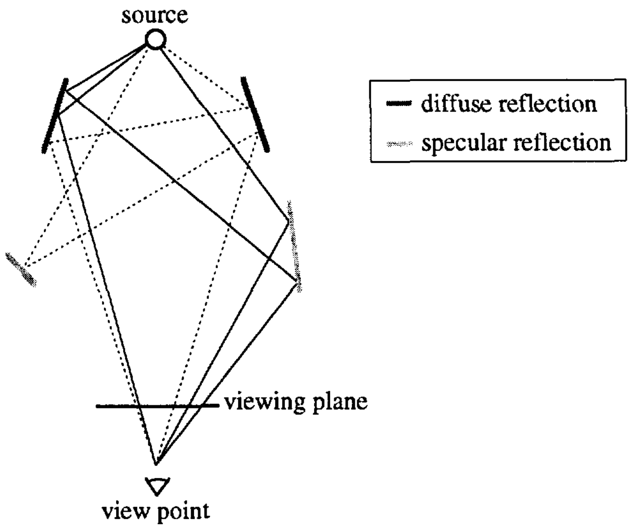


Figure 2.8 Light paths for 'standard' recursive ray tracing

are traced into all directions when the specular part of the BRDF at point **x** is not equal to zero. When a patch is hit, equation 2.15 is evaluated recursively by tracing shadow and reflection rays. As no indirect diffuse illumination is captured during ray tracing, usually an ambient term is used to give a crude estimation of this diffuse interreflection. The fourth term in the equation gives this ambient illumination.

Recursive ray tracing, as described above, takes only into account the direct diffuse, the diffuse-specular, and the direct specular light transport (figure 2.8). Although this method has shown to give nice looking images, fully realistic and physically correct images cannot be generated with this method.

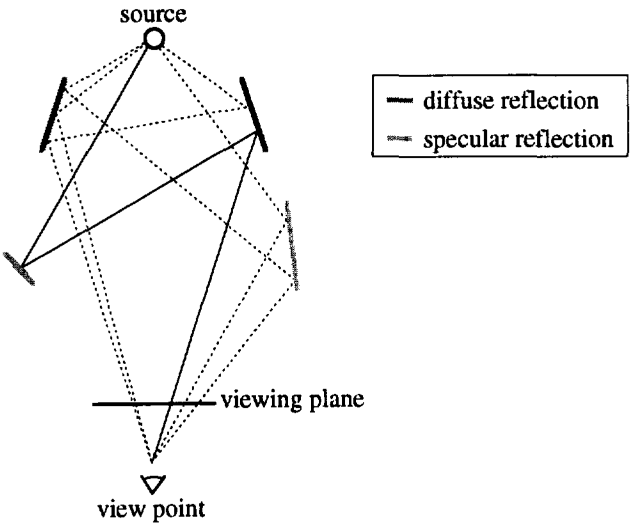


Figure 2.9 Extra light path of light ray tracing

Recursive ray tracing does not take into account diffuse interreflection and the specular-diffuse light transport that causes caustics. To capture the latter component a preprocessing step, light ray tracing[2], can be used (Arvo 86). Light ray tracing computes the radiance values caused by illumination via specular reflections. These radiances are stored on the surfaces in illumination maps. Rays are shot from the light source into the scene. Each ray represents some amount of power. When a ray hits a diffusely reflecting surface after being reflected by one or more specular reflecting patches, the power of the ray is recorded on this diffusely reflecting surface. When the ray hits a diffusely reflecting surface without being specularly reflected before, the ray is further neglected. When the scene is now rendered with standard ray tracing to generate an image, and a ray hits a diffusely reflecting surface, then the radiance at the intersection point is calculated by adding the value of the illumination map at the intersection point to the direct illumination component that is computed by shadow

---

[2] Originally this method was called backward ray tracing, but this name caused too much confusion.

ray casting. Light ray tracing accounts for the specular-diffuse light transport as depicted in figure 2.9. The quality of the caustics is dependent on the resolution of the illumination map. The higher this resolution the more accurate the shape of the caustics will be.

Distribution ray tracing (Cook et al 84; Cook 86) extends standard recursive ray tracing by adding stochastic methods to simulate all kinds of optical effects. Rays are distributed over several domains (pixel positions, lens positions, area sampling position, etc.). This will allow anti-aliased images with depth of field, soft shadows, etc.

In ray tracing algorithms, the diffuse interreflection contribution to the illumination is usually approximated with an ambient term. It is, however possible to calculate the diffuse interreflection accurately within a distribution ray tracer, and solve the complete rendering equation (Kajiya 86). Instead of only sending shadow rays to light sources and reflection rays into specular reflection directions, rays are traced into all directions to sample all incoming light. Because the light reflected by the other surfaces that are hit by these rays is not known, the sampling must be performed recursively (figure 2.10). General BRDFs can be used with this method, because all incoming directions can be sampled with rays, and therefore all light paths can be computed (figure 2.11).

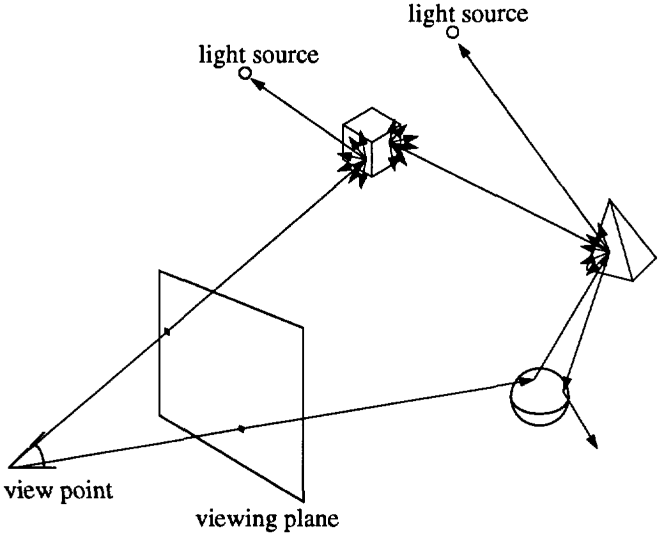


Figure 2.10 Diffuse interreflection calculation with ray tracing

To sample all directions for the complete hemisphere above the intersection point a ray distribution over the hemisphere can be determined with Monte-Carlo methods (Kajiya 86; Kalos and Whitlock 86). Random numbers are generated from which these directions can be computed. However, many rays are needed to get a good representation of all incoming directions, and so to get an accurate irradiance.

Methods to reduce the number of sampling rays are path tracing in combination with importance sampling (Kajiya 86), and irradiance caching (Ward et al 88).

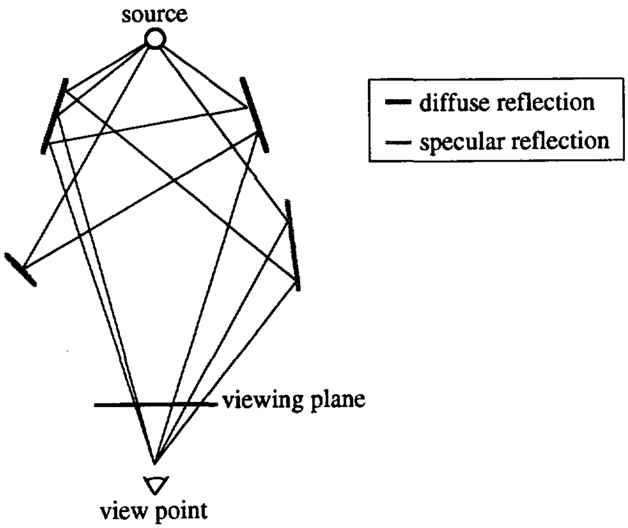


Figure 2.11 Light paths for ray tracing with diffuse interreflection

In path tracing the number of rays is reduced by not tracing all secondary (reflection) and shadow rays, but by selecting only one or a few light sources to which shadow rays are cast and by selecting only one or a few directions in which rays are traced recursively to sample the diffuse interreflection. The selection of light sources and sampling directions can be done randomly, but better results are achieved with importance sampling (Kajiya 86; Shirley 90b; Shirley and Wang 91). Shadow rays and rays that sample the diffuse interreflection are distributed over the different light sources and reflection directions in such a way that the number of rays to each light source and in each direction is proportional to the importance of these light sources and directions. A probability factor, proportional to the estimated contribution of a light source or direction, is therefore assigned to each light source and sampling direction. This probability factor ensures that shadow rays are sent at least to the brightest and nearest light sources and patches. A serious drawback of these stochastic approaches is that although they reduce the number of secondary and higher generation rays, they require a large number of primary (viewing) rays for each pixel. They work well for images that are heavily supersampled, but generate a great deal of noise when the sampling density is low.

Irradiance caching (Ward et al 88; Ward and Heckbert 92) exploits shading coherence and is based on the assumption that the indirect diffuse component varies only smoothly over a surface. Therefore the indirect diffuse irradiance calculated for one point, can also be used when the indirect diffuse irradiance must be calculated for neighbouring points. This is implemented by storing the calculated irradiance values in a datastructure. When an indirect interreflection must be calculated for a point in the scene, and irradiance values are already calculated for nearby points (for rays that were traced earlier), the indirect irradiance value can be interpolated instead of calculated by sampling the environment. For the first rays tracing is done to full depth, but for later viewing rays the number of rays needed to compute indirect irradiance values diminishes because enough information is calculated already. Coherence criteria are used to determine whether earlier calculations provide enough information. Extra samples can be required for areas with a high surface curvature and near edges of objects, while interpolation is possible for points on flat patches. An advantage of the irradiance caching method is that only those areas are sampled that are involved in the illumination calculation for a given view.

2.5 Radiosity

Another method to solve the global illumination problem is the radiosity method. This method, that finds its roots in thermal engineering (Sparrow and Cess 78; Siegel and Howell 81), can also be applied for solving the global illumination problem for the generation of realistic images (Goral et al 84; Nishita and Nakamae 85; Cohen and Greenberg 85).

In the 'standard' radiosity method all surfaces are assumed to be perfectly diffuse. All surfaces emit diffusely and reflect diffusely. The consequence is that in this case the radiance is independent of outgoing direction, and is only a function of position:

$$L_{out}(\mathbf{x},\Theta_{out}) = L(\mathbf{x}) \tag{2.16}$$

Also, the diffuse reflectance is independent of in- and outgoing directions, and is constant in all directions. Using equation 2.5 for diffuse reflectance, equation 2.13 can be simplified for purely diffuse scenes:

$$L(\mathbf{x}) = L^e(\mathbf{x}) + \rho^d(\mathbf{x}) \int_{all\,\mathbf{x}'} L(\mathbf{x}') \frac{\cos\theta_i \cos\theta'_o}{\pi \|\mathbf{x}' - \mathbf{x}\|^2} v(\mathbf{x},\mathbf{x}') dA' \tag{2.17}$$

It is not possible to solve this equation for each infinitesimal small point in the scene. Therefore the surfaces in the scene are subdivided into sufficiently small patches and the radiance and reflectance are assumed to be constant for each patch. Consider point **x** on patch *i* and **x**' on patch *j*, then the radiance for patch *i* is given by:

$$L_i = L_i^e + \rho_i^d \sum_j \frac{L_j}{A_i} \int\!\!\int_{A_i A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \delta_{ij}\, dA_j\, dA_i \tag{2.18}$$

where r is the distance between the patches *i* and *j*, and δ_{ij} gives the mutual visibility between the delta areas of the patches *i* and *j*. If no occlusions exist between these delta areas, then δ_{ij} is one, otherwise it is zero.

Equation 2.18 can be rewritten as:

$$L_i = L_i^e + \rho_i^d \sum_j L_j f_{i \to j} \tag{2.19}$$

with:

$$f_{i \to j} = \frac{1}{A_i} \iint\limits_{A_i A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \delta_{ij} \, dA_j \, dA_i \tag{2.20}$$

Using the relation:

$$A_i f_{i \to j} = A_j f_{j \to i} \tag{2.21}$$

and writing formula 2.19 in terms of power instead of radiance gives:

$$\Phi_i = \Phi_i^e + \rho_i^d \sum_j \Phi_j f_{j \to i} \tag{2.22}$$

From this equation it is clear that $f_{i \to j}$ is the fraction of the power leaving patch *i* that will reach patch *j*. This fraction is called the form factor. It is based on the geometry of the two patches and their relative position.

Equation 2.19 gives the radiance of a patch by gathering incoming radiance coming from all other patches in the scene. It is the basis for the radiosity method (Goral et al 84; Nishita and Nakamae 85). This method received its name from the quantity in which it was first described: radiosity. Radiosity (B) is a non-standard name for radiant exitance. It is defined as the density of radiant flux leaving an area. The relation between radiosity and radiance is given by the following equation:

$$B = \int\limits_{\Omega_o} L_o \cos\theta_o \, d\omega_o \tag{2.23}$$

For diffuse surfaces, radiosity and radiance are therefore related by:

$$B(\mathbf{x}) = \pi L(\mathbf{x}) \tag{2.24}$$

Radiosity is not part of the standard terminology, so will not be used here further. Radiosity methods can be defined as methods that solve the global illumination problem by partitioning the scene into small patches (or zones, the radiosity method is sometimes also called zonal method) and computing the radiances of these patches by solving a system of equations.

The radiosity method computes radiance values for each patch in the scene. This method simulates direct and indirect diffuse light transport (figure 2.12), but does not account for the specular reflection. It is not a rendering method, but only an illumination computation method. Any rendering technique can be used to make images of the scene using the results of the radiosity method. The results of the radiosity method are view-independent, because only diffuse surfaces are used. Therefore many images from different view-points can be made with the results of one radiosity processing.

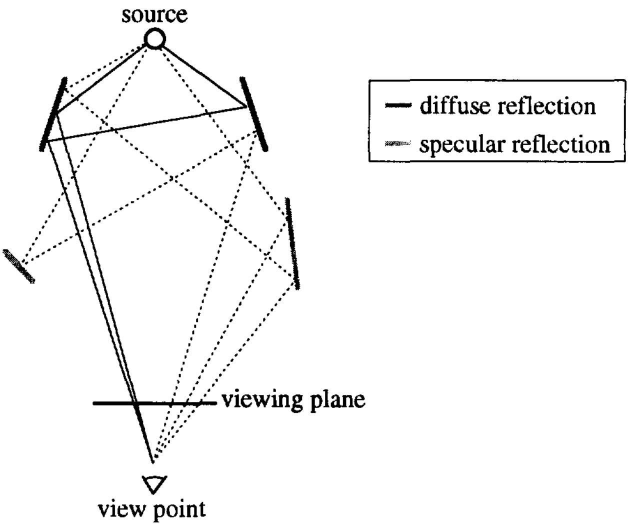


Figure 2.12 Light paths of radiosity method with simple rendering algorithm

The attempt to generalize the view-independent radiosity method to include also specular light reflection (Immel et al 86) was not successful. It required an enormous amount of memory and an enormous amount of computations, because for the specular component form factors are not only a function of the position and geometry of two patches, but also of the reflection directions.

2.5.1 Full-Matrix radiosity

If the scene consists of n patches, then n linear equations with n unknown values exist. Each unknown represents a radiance value for a patch. The radiance of each patch can now be calculated by solving the resulting system of n equations:

$$\begin{pmatrix} 1-\rho_1^d f_{1\to 1} & -\rho_1^d f_{1\to 2} & \cdots & -\rho_1^d f_{1\to n} \\ -\rho_2^d f_{2\to 1} & 1-\rho_2^d f_{2\to 2} & \cdots & -\rho_2^d f_{2\to n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n^d f_{n\to 1} & -\rho_n^d f_{n\to 2} & \cdots & 1-\rho_n^d f_{n\to n} \end{pmatrix} \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_n \end{pmatrix} = \begin{pmatrix} L_1^e \\ L_2^e \\ \vdots \\ L_n^e \end{pmatrix} \qquad (2.25)$$

This system is normally diagonally dominant, so a Gauss-Seidel iterative solution method can be used to rapidly get accurate patch radiances (Cohen and Greenberg 85). Because this method requires solving a matrix equation, and the full matrix must be known before a solution can be computed, it is often called Full Matrix (FM) radiosity.

To be able to solve this system of equations, first all form factors must be known. The storage and computation requirements for these form factors are very high, namely of $O(n^2)$. For very complex scenes this is unacceptable.

2.5.2 Progressive radiosity

An alternative for the full-matrix radiosity method was given in (Cohen et al 88). In this method, progressive radiosity (PR), it is not necessary to compute and store all n^2 form factors before the radiances can be derived.

From equation 2.19 it can be seen that the contribution of a patch j to the radiance of a patch i is given by:

$$L_{i\leftarrow j} = \rho_i^d L_j f_{i\rightarrow j} \tag{2.26}$$

Written in terms of power this equation becomes:

$$\Phi_{i\leftarrow j} = \rho_i^d \Phi_j f_{j\rightarrow i} \tag{2.27}$$

If the radiance or power of patch j is known, then the contribution to all other patches i in the scene can be computed. Patch j shoots its radiant power into the scene to all other patches. Thus instead of gathering the radiance at a patch from the other patches in the scene, now radiance or power is shot from a patch to all other patches in the scene (figure 2.13). By selecting one of the patches in the environment being the 'shooting' patch, the contributions of this shooting patch to all the patches in the scene can be calculated using the form factors from this patch to all other patches. The form factors need not to be stored for later use, saving an enormous amount of storage.

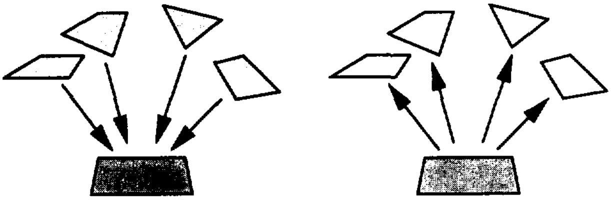


Figure 2.13 Gathering (FM) versus shooting (PR)

The shooting algorithm is as follows. The unshot radiant power and the 'normal' radiant power of all emitting patches are initialized to the emitting power. For all other patches these values are initialized to zero. In each iteration step of the iterative progressive radiosity method, the most radiant patch (the patch with the highest unshot power) is chosen to be the shooting patch. This patch shoots its unshot power to the environment by computing the form factors to the other patches. The unshot power and the normal power of each patch in the environment are updated using formula 2.27. At the end of the iteration step the unshot power of the shooting patch is set to zero. Then again the patch with the highest unshot power is selected to shoot its power. This process stops when the remaining unshot power in the scene is sufficiently small. After each iteration step an image can be made (for example with depth buffering) using the radiances calculated so far, so one can see the image converge to the actual solution. The remaining unshot power in the scene can be used during display to make a reasonable approximation for an 'ambient' term.

The main advantage to use progressive radiosity instead of full-matrix radiosity is the reduced memory cost, so more complex scenes can be handled.

2.5.3 Form factor computation

The computation of the form factors is the computationally most expensive part of the radiosity method, both in the full-matrix and in the progressive radiosity method. It is impossible to solve analytically the double integral while taking into account the mutual visibility of the two surfaces. Stoke's theorem to solve the double integral as a double contour integral is only applicable when no occlusions occur (Goral et al 84). Therefore methods, such as the hemi-cube and ray tracing, that approximate the form factors are used instead. These methods are often based on Nusselt's analog that states that the form factor is equal to the fraction of the base of the hemisphere covered by projection of a surface onto the hemisphere (figure 2.14).

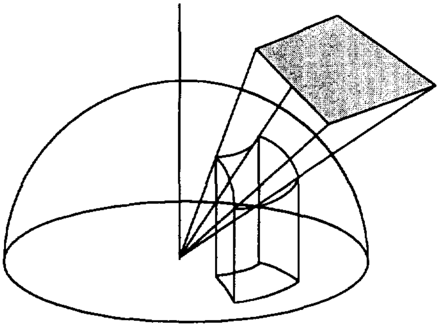


Figure 2.14 Nusselt's analog to compute the form factor

The Hemicube

The hemicube method (Cohen and Greenberg 85) is based on Nusselt's analog, but instead of projection on the hemisphere, it uses projection onto a hemicube. The hemicube consists of five planes, each subdivided into small elements. Each element of the hemicube represents a delta form factor. The delta form factor of each cell of the hemicube is determined in advance. The hemicube is placed on the patch for which the form factors must be computed. The form factors are determined by projecting all other patches onto the planes of the hemicube. The advantage of the hemi-cube method is that depth-buffering can be done with the specialized hardware that is available for several graphics workstations. However, the method is prone to aliasing, it requires all surfaces to be approximated with polygons, and is less suitable for including specular light transport within the form factor computation.

Form factor computation by ray tracing

Form factors can also be computed using ray tracing methods. Ray tracing has a number of advantages compared with the hemicube method. Arbitrarily shaped surfaces can be used; surfaces need not to be approximated with polygons anymore. Directions of rays can be determined in a flexible way, allowing adaptive sampling and stochastic methods to avoid aliasing. The main disadvantage is that up to now no specialized hardware is available to speed up the ray object intersection calculations.

Ray tracing based form factor computation methods can be classified into two groups, the undirected form factor computation, and the directed form factor computation (figure 2.15).

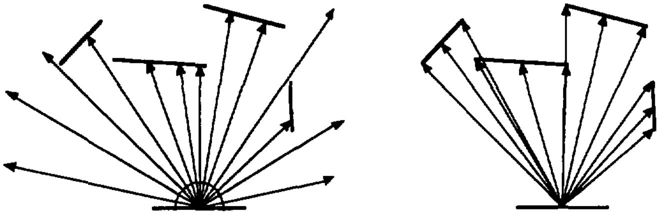


Figure 2.15 Undirected versus directed ray tracing based form factor computations

Undirected form factor computation

Form factors can be computed by sampling the hemisphere with rays. Rays are traced from the base point of the hemisphere through points on the hemisphere. Each ray then represents a delta form factor. The form factor of the patch that is hit and the patch from which the ray originated is then updated with this delta form factor. Because rays are not shot aimed at hitting a specific patch, this kind of form factor computation is called undirected.

The most straightforward method is to distribute the points uniformly over the hemisphere. A disadvantage of this method is that the delta form factors of the rays are not equal, because the delta form factors are a function (cosine) of the angle of the ray direction and the normal of the hemisphere. Also a considerable amount of work is done for directions where the delta form factor is almost zero (along the base of the hemisphere). A third problem is that a uniform distribution can result in aliasing.

A non-uniform Monte-Carlo based method will therefore have a better performance. A good distribution of ray directions is achieved when the base of the hemisphere is sampled with jittered points (Malley 88). To determine a ray direction, such a point is projected onto the hemisphere, and the ray direction is now from the center point of the hemisphere to the projected point. When the total base is sampled evenly, because of Nusselt's analog each ray represents the same delta form factor, that is equal to one divided by the number of points.

A third method for undirected shooting uses a plane to determine ray directions (Sillion and Puech 89). One plane is placed above the point for which form factors should be calculated. This plane is subdivided non-uniformly into 'proxels' (projection elements) that all represent about the same delta form factor, and rays are sent through the proxels into the scene.

Undirected form factor computation has one important disadvantage. When the resolution of the rays is not high enough, aliasing effects may occur (figure 2.16). Although small patches or elements are visible, they are easily missed by the rays, especially when they are far away of the origin of the rays. Also it is not guaranteed that two neighboring elements receive (approximately) the same amount of rays. Especially when the number of rays is small this may become visible. In figure 2.16 a surface that is subdivided into 256 patches is illuminated by a source. The number of rays that is shot from the source is increased. At least 100000 rays (figure 2.16.f) are needed to avoid aliasing, while for directed shooting only 289 rays are needed (figure 2.16h)

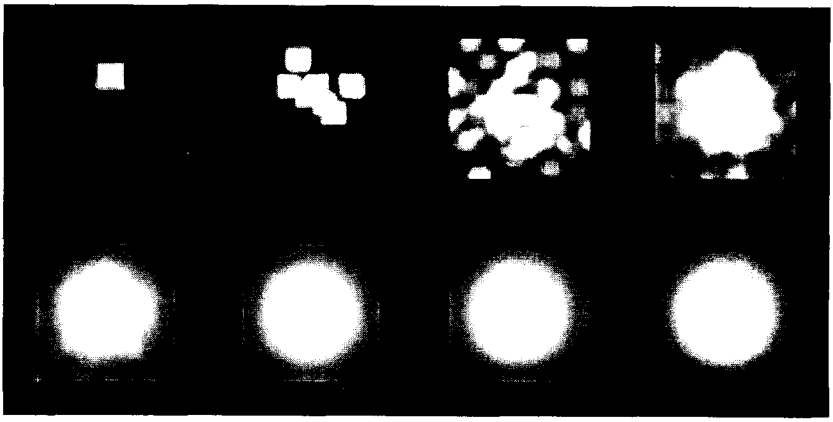


Figure 2.16 Results radiosity computations for a surface, divided into 256 patches, lit by a small light source. Undirected shooting: 1 (a), 10 (b), 100 (c), 1000 (d), 10000 (e), 100000 (f), 1000000 (g) rays, and directed shooting: 289 (h) rays

Directed form factor computation

Directed shooting does not have this problem (Wallace et al 89). At least one ray is traced from the patch for which the form factors must be computed to each other patch or element. So each patch that should be visible is known to be visible. If no intersections are found, then the form factor can be computed, otherwise it is zero. If necessary, patches can be adaptively subdivided to find their visible parts more exactly, and to get more accurate form factors. When the visibility is solved, the form factor can be computed. Equation 2.20 cannot be used directly to compute the form factor. Therefore often the integrals in this equation are replaced by summations. However, this can only be done if the distance between the two patches is much larger than the delta areas of the patches. Therefore alternative formulas are used that use finite areas instead of delta areas. One solution is to approximate the area with a circular disk (Wallace et al 89). A better solution, but also more expensive to compute, is to use an analytical solution based on Stoke's theorem (Baum et al 89).

2.5.4 Meshing

The radiosity method computes only one radiance value for each patch (or for a number of elements or vertices of elements of a patch). To represent all shading transitions properly, the patches or elements should be infinitesimal small. However, then the radiosity method will take an infinite amount of time to compute the radiances. Therefore a meshing scheme is necessary that subdivides the surfaces into (as few as possible) patches and elements in such a way that patch and element borders follow the shading transitions. If it creates patches that are too big, then a very inaccurate shading will result. Shadow areas will be too large and the shadows will be blurred and jagged. For example, a very small light source is expected to generate a sharp shadow on a surface. If the shading is represented in a mesh that is too coarse, then this shadow will be highly blurred. It is very difficult to generate a good mesh. Many problems will arise when the meshing is not done properly (Haines 91). A good mesh must satisfy a number of requirements (Baum et al 91). An automatic meshing scheme should take care that all these requirements are fulfilled.

When an initial mesh is made, this mesh can be adaptively refined (Cohen et al 86) to achieve a better solution. Only those regions of a surface are refined where a large shading gradient exists (so where important shadow boundaries are). Although this method can improve the radiosity results, it will never be able to represent the shading variations accurately, for example when zooming in on a patch.

The goal of the meshing is to be able to represent the shading gradients. If it is known a priori where these transitions will occur, then the mesh can be made to fit the shading transitions exactly. Exact meshing or *discontinuity meshing* schemes (Campbell and Fussell 90; Heckbert 91; Heckbert 92a; Lischinski et al 92) compute in advance where discontinuities in the shading will occur. Patches are subdivided along these discontinuities. The radiosity results with these meshes are very good. However, determining where the discontinuities will be is a complex and expensive process. The projection of surface contours onto other patches, that is used for that, is very difficult and expensive, and storing the mesh requires complex datastructures, especially when curved surfaces are involved. Also, it is very difficult to include the discontinuities that are caused by specular reflections.

2.5.5 Particle tracing

In progressive radiosity using undirected or Monte-Carlo shooting (see section 2.5.4), when a ray hits a patch, the power is stored on this patch, and later, when this patch is selected to be the source patch, this power is sent further into the scene. It is, however, also possible to reflect the power and send it back into the scene immediately. In this case, packets of power (particles or photons) are followed on their way through the scene until they are absorbed by a surface. This method is called particle tracing (Shirley 90b; Pattanaik and Mudur 92; Pattanaik 93). A particle is sent out of an emitter into the scene. When this particle hits a surface, it is either reflected or absorbed. When it is reflected, the outgoing flux of the surface is updated and the particle is followed further on its way through the scene. The reflected flux can be stored by recording the position of the hit on the surface, but this will require much memory as many reflection points will be computed during the simulation. An alternative is to store this flux in a mesh as in the radiosity method.

A number of decisions have to be made at different stages of this algorithm: the wavelength of the particles, their origin (from which light source and from which position on this light source they are cast), reflection or absorption when they hit a surface, and their reflection direction. All these decisions are taken using Monte Carlo methods by sampling probability density functions (Kalos and Whitlock 86). For example, the direction in which a particle is reflected is determined by a probability distribution function that is based on the BRDF of the surface.

Particle tracing is able to simulate all light paths. Even very complicated BRDFs can be used easily. Because of the stochastic nature of this method, particle tracing must use many rays to avoid noise.

2.6 Hybrid global illumination methods

From the previous discussion it is clear that ray tracing methods are very well suited to simulate specular reflections and that radiosity methods are mainly suited to simulate diffuse reflections. Both methods can be extended to include all light transfer paths, but only at high costs. However, ray tracing and radiosity complement each other well. Therefore several hybrid radiosity and ray tracing methods have been proposed.

One of the first hybrid methods is the two-pass method (Wallace et al 87; Shao et al 88; Sillion and Puech 89). In the first pass, the radiosity pass, the diffuse component of the radiances is computed with a radiosity method. In the second pass, the rendering pass, an image, including specular reflection, is rendered by ray tracing using the results of the first pass. Figure 2.17 shows the different parts of the two-pass method.

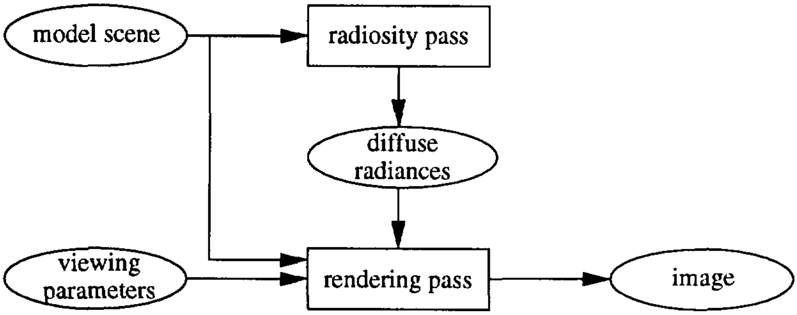


Figure 2.17 Two-pass method

The radiosity pass computes the diffuse component of the radiances. This diffuse component can also be caused by a specular reflection, for example when light reaches a diffuse surface via a specular surface (mirror). It is therefore necessary to include the specular to diffuse transport in the radiosity pass. To account for this transport the extended form factor is introduced. The extended form factor $f_{i \rightarrow j}$ is defined as the fraction of power leaving surface i that reaches surface j directly or by one or more specular reflections. To compute the extended form factor, in (Wallace et al 87) the concept of mirror form factors is used (Rushmeier and Torrance 90). The scene is flipped about the plane of the specular patch, and this new (virtual) scene is used to compute the extended form factors. The specular patch is now an opening to the mirrored scene. This method, however, is only applicable to planar specular surfaces and for scenes with a very limited number of specular surfaces. However, extended form factors can be computed very well with an undirected ray tracing based form factor calculation method (Malley 86; Sillion and Puech 89; Kok et al 90). When a form factor ray hits a specular surface, the ray (or a frustum of rays, to simulate rough specularity) is propagated into the reflection direction. When a (partly) diffuse surface is hit, the form factor of the surface at the origin of the ray and the hit surface is updated (figure 2.18). Figure 2.19a gives the light paths that are computed with a radiosity pass using extended form factors.

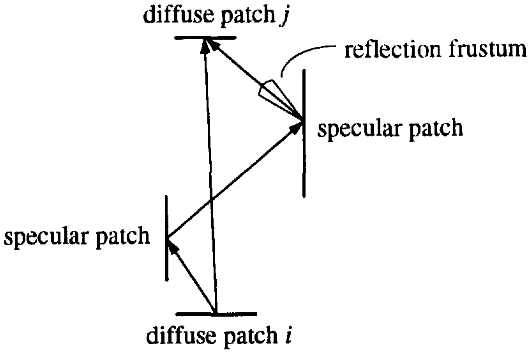


Figure 2.18 Extended form factor computation by ray tracing

In the second pass, the rendering pass, an image is generated by ray tracing. However, here ray tracing is simpler than standard ray tracing (Whitted 80). No shadow rays are cast to compute the shading of a point. When a viewing ray hits a (partly) diffuse patch, instead of casting shadow rays as in standard ray tracing, the diffuse component of the radiance value at the intersection point is derived from the radiance values computed during the radiosity pass. Only when the patch is (partly) specular, reflection rays are sent in the reflection direction, as in standard recursive ray tracing, and the process in repeated. In figure 2.19b the paths are given that are followed during the rendering pass. The ray tracer uses the radiance values computed during the radiosity pass at all points indicated with x.

This two-pass method is able to simulate all light paths. The resulting image quality, however, strongly depends on the meshing used during the radiosity pass, as in all radiosity based methods. The resolution of the meshing determines the accuracy of the shading of the surfaces in the scene (see section 2.5.4). It is possible to make the meshing dependent on the viewing position (Heckbert 90). An extra ray tracing preprocessing pass determines which surfaces are visible in the image and what meshing density is required to achieve good shading results. An important disadvantage of this approach is that the radiosity solution is not longer viewpoint independent, although this is only a disadvantage when more than one image has to be made of the scene.

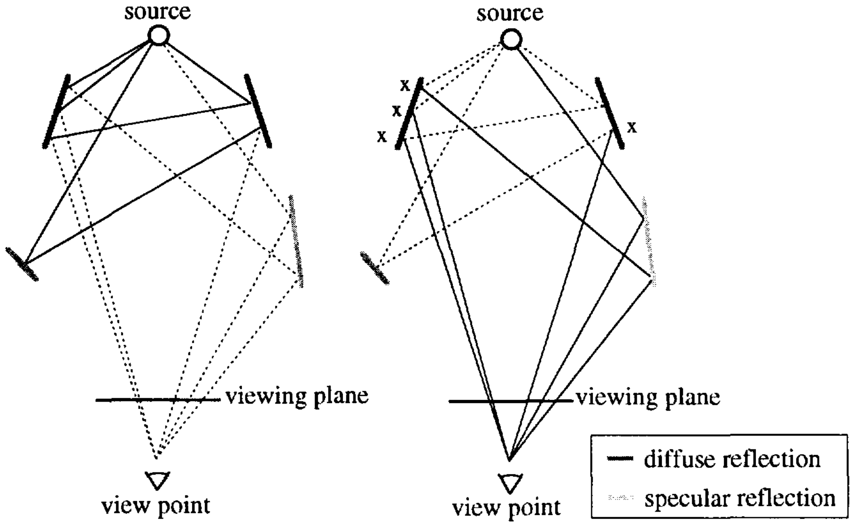


Figure 2.19 Light paths for two-pass method (Wallace et al 87; Sillion and Puech 89): radiosity pass (a), and rendering pass (b)

Another solution to the meshing problem is to make a distinction between light that a surface receives directly and light that it receives indirectly from light sources (Ward et al 88; Shirley 90a). Direct light is the light that a patch receives from a light source without being reflected by other patches. Indirect light is the light that is received after one or more (diffuse or specular) reflections. The direct light is responsible for the larger shading gradients (shadow boundaries) on surfaces, so it is better not to store this light on a mesh. The indirect lighting usually shows minor variations, and can therefore be stored in a relatively coarse mesh. The indirectly received light is calculated by light ray tracing and radiosity (Shirley 90a). First, light ray tracing (Arvo 86) computes the indirectly specularly reflected illumination and stores it in illumination maps (figure 2.20a). It accounts for the hard lighting component via specular reflections. Second, a radiosity computation, using extended form factors, is performed for the whole scene. Hereafter, for each patch, the light received directly from light sources is subtracted from the patch radiance value by following the light again from the sources until a patch is met. The contributions of the light rays are subtraced from the patch radiance values (figure 2.20b). The resulting indirect shading will show only minor variations. The patch refinement can therefore be relatively coarse. Finally, during the rendering pass, light received directly from light sources is calculated by casting shadow rays to the light sources as in standard ray tracing, whereas the indirect component is calculated by taking the sum of the radiance value and the illumination map value at the intersection point (figure 2.20c).

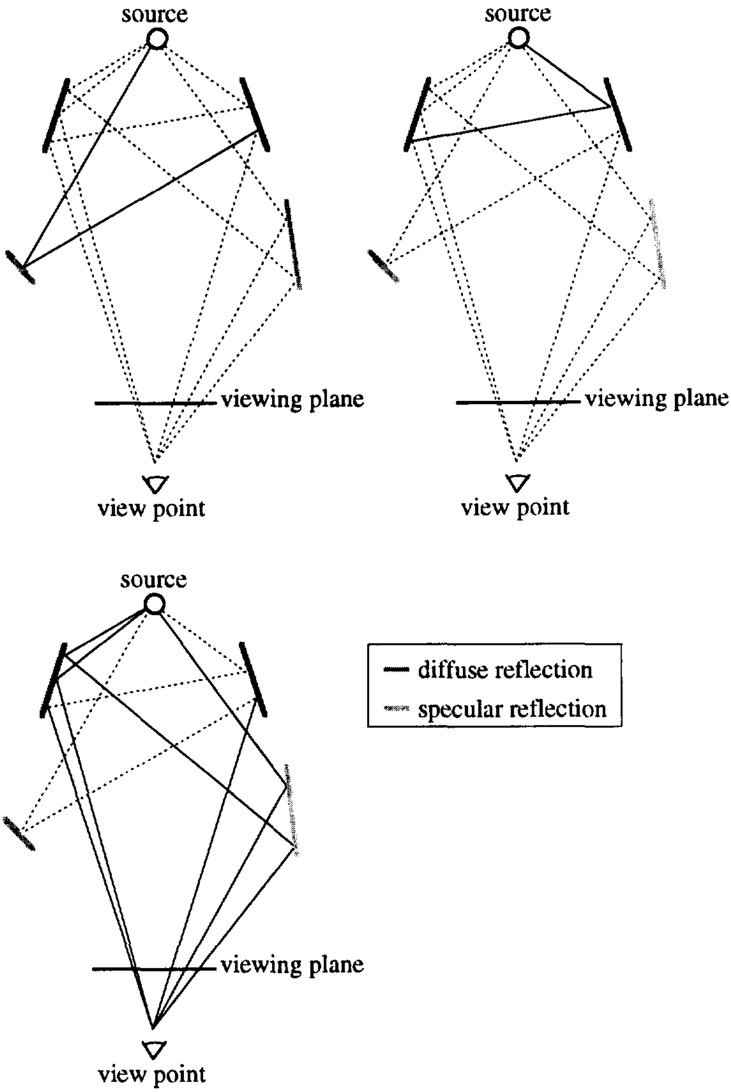


Figure 2.20 Light paths for hybrid method with distinction between directly and indirectly received light: light ray tracing pass (a), radiosity pass (b), and rendering pass with shadow rays (c)

Another hybrid method is one-level Monte-Carlo sampling. This method starts with a radiosity pass to compute the radiances of all patches as a normal radiosity pass. When, during rendering, the illumination of a point must be computed, this illumination is not interpolated from the radiances of the patch that is intersected. Instead, a gathering step is performed by sampling all incoming illumination with sampling rays. Each sampling ray represents a delta form factor. When a sampling ray hits a patch, the illumination represented by that ray is computed from the radiance of the hit patch and the delta form factor of the ray. The results of the radiosity pass are not used for display purposes, only to compute the contributions of the patches to visible points in the image. Therefore the meshing can be very coarse during the radiosity pass. This method may give a good shadow accuracy. However, for each visible point many sampling rays should be cast to capture the illumination. Otherwise the image will contain much noise. Instead of sending multiple sampling rays per viewing ray, it is also possible to cast only one sampling ray per viewing ray and send many viewing rays per pixel (Rushmeier 88). The direction of the sampling ray is then determined as in Monte Carlo path tracing. The length of the paths is limited to one. To avoid noise, this method requires many sampling rays at each intersection point or many viewing rays per pixel. This noise is primarily caused by direct illumination. Sometimes a light source (or strong radiant patch) is hit by a sampling ray, sometimes not. To reduce the noise it is also possible to extend the method to capture the indirect illuminations by undirected sampling (gathering), and to capture the direct illumination from light sources by directed sampling (by shadow rays). Then the methods of (Shirley 90a) and (Rushmeier 88) only differ in the way they compute indirect interreflection: by interpolation from radiosity results, or by sampling the environment. Computing the indirect illumination also by directed sampling may be quite expensive, as it requires sending shadow rays from each visible point to all patches (and for large patches to several points on these patches).

2.7 Conclusions

This chapter contains a survey of methods to solve the global illumination problem. To be able to make a decision what method is considered to be the best, the methods can be compared on the following characteristics:

- The ability to simulate all possible light paths.
- The accuracy of the shading (the ability to represent all shading gradients).
- The time to generate images.

Table 2.1 gives a summary of the methods described before, based on the above characteristics.

To compare the methods in their capability to solve the global illumination the regular expression notation given in (Heckbert 90) is used. Table 2.2 describes the symbols and operators in such a regular expresssion.

All light paths can be indicated with L(S|D)*E; light leaves the light source and reaches the eye after an arbitrary number of specular and/or diffuse reflections. For each of the methods described in this chapter such an expression can be given. These expressions are given in the second column of table 2.1. Except for standard recursive ray tracing, with or without light ray tracing, and the standard radiosity approach, all methods are capable of simulating all paths.

The third column of table 2.1 gives the shading accuracy that can be reached. Ray tracing methods give a very good accuracy. A light ray tracing pass uses a mesh (illumination map) to store the caustics, so the accuracy of these caustics are dependent on the resolution of the used mesh. The same is true for radiosity methods and particle tracing that store the diffuse illumination in meshes. When a hybrid method is used that makes a separation into direct and indirect illumination, the resolution of the meshing is of less importance, because the highly varying shading is computed with mesh independent shadow ray casting. The one-level Monte-Carlo sampling method is strongly dependent on the resolution of the shading rays. When the number of rays is not high enough, the generated image may contain a lot of noise, especially if there are some small important light sources in the scene.

An indication of the time that is needed to generate images is given in columns 4 and 5 of table 2.1. Computation times are separated into a view-independent (pre-) processing time and a view-dependent image generation time. The view-independent processing has to be performed only once for a scene. After that, many images can be generated using the same results. The image generation time is therefore of more importance. Times are given relative to each other, and are dependent on the complexity of the scene and the resolution of the image. For an image of moderate complexity, short is in the order of minutes, long of hours, and very long of days.

The ability to simulate all light paths and the shading accuracy that can be achieved are the most important issues. Considering these issues, two methods are preferable: ray tracing with diffuse interreflection and a direct-indirect hybrid method. They both simulate all light paths, and have no problems with shading accuracy of direct illumination, and only slight problems with shading accuracy for indirect illumination. Ray tracing with diffuse interreflection has the advantage that no meshing is needed, and so it is better suited for generally shaped objects. The advantage of the hybrid method is that the image generation time is shorter. Therefore this method will be explored further.

The hybrid method uses two passes, a radiosity and a ray tracing based rendering pass. The radiosity pass should use the progressive refinement approach, because the full-matrix approach cannot cope with scenes with many patches.

A number of problems still remain unsolved. Both passes can take a considerable amount of time to run. Therefore we will try to find methods to reduce the amount of computations in both passes. Following chapters will try to provide solutions for these problems:

- During rendering, the number of shadow rays necessary to sample the light sources can still be very high. A mechanism to find the sources that are really important must be invented. Chapter 3 describes a hybrid method that selects the really important sources for shadow ray casting. It might also be useful to try to use coherence during shadow ray casting (chapter 4). Besides it is necessary to find a way to limit the number of shadow rays necessary to sample large area light sources (chapter 5).

- Although shadow rays are cast to capture important shading discontinuities, surfaces still need to be subdivided into patches and elements to be able to do a proper radiosity pass. A good adaptive subdivision scheme is necessary to obtain an optimum between accuracy and computational cost. It might even be necessary to combine surfaces and to consider these as one 'macro' patch to get optimal results. A solution for this problem is given in chapters 6 and 7.

method	light paths	shading accuracy	time view-independent processing	time image generation
standard ray tracing	L(S*|DS*)E	good	no	long
ray tracing with light ray tracing	L(S*DS*|S*)E	good/(mesh)	moderate	long
ray tracing with diffuse interreflection	L(S|D)*E	good/(noise)	no	very long
radiosity + depth-buffering	LD*E	mesh	long	very short (real-time)
two-pass method	L(S|D)*E	mesh	long	short
direct-indirect hybrid method	L(S|D)*E	good/(mesh)	moderate/long	long
one-level Monte-Carlo sampling	L(S|D)*E	good/noise	moderate	very long
particle tracing	L(S|D)*E	good/mesh/noise	very long	depends on rendering

Table 2.1 Summary global illumination methods

symbol		operation	
L	light source	*	repeat zero or more times
E	eye	|	or
D	diffuse reflection	( )	group sub expressions
S	specular reflection		

Table 2.2 Regular expression symbols and operators for light paths

3 A global illumination method with source selection

3.1 Introduction

Improved shadow accuracy can be obtained by a separate treatment of the direct and indirect illumination. The indirect component is computed during the radiosity pass and the direct component during rendering by casting shadow rays. The shading accuracy is improved compared to 'standard' two-pass methods, because the meshing that is needed in the radiosity pass does not have to represent the perceptible shadow transitions caused by obstructions of light sources. Only the indirect illumination must be represented in the mesh. Therefore this mesh can be relatively coarse (Shirley 90a).

The number of shadow rays that is necessary to sample all light sources during rendering may be very high. A scene may contain many light sources, or the present light sources may have a large area, which means that many shadow rays are required to sample them properly. To reduce the number of shadow rays it is possible to rely on a Monte-Carlo integration method (Shirley and Wang 91). Only one shadow ray is cast, regardless of the number of light sources. Probabilities assigned to the light sources determine to which source the ray is cast. This method, like all stochastic methods, has the disadvantage that many samples, in this case viewing rays per pixel, must be used to suppress perceptible noise.

Moreover, the separate treatment of directly and indirectly received light from light sources is rather arbitrary. Indirectly received light from a very strong light source may be stronger than the directly received light from a very weak light source. Consider for instance a strong spot light that is directed onto a nearby white wall. Although most of the other patches will not receive direct light from this spot light, the light received indirectly via the wall may still create perceptible shading transitions on these patches. Therefore it must be possible to sample not only the light from light-emitting patches, but also the light from other highly radiant patches. It is then possible to sample also the important indirectly received light.

This chapter presents a new hybrid global illumination method. It gives a very flexible way of determining which contributions, and not only those of light-emitting patches, should be computed very accurately during rendering. Section 3.2 describes this method. It uses source selection to classify which patches should be considered to be important contributors. Section 3.3 gives some criteria to select these patches. Finally, section 3.4 describes the results of some experiments and presents conclusions.

3.2 Source selection in a hybrid global illumination method

The separation of direct illumination from indirect illumination does not always provide the best results in the most efficient way. As a result a new and better separation of illumination must be used. Instead of separating the direct illumination from the indirect illumination, we propose to separate high-frequency illumination from low-frequency illumination:

- High-frequency illumination is the illumination from highly radiant patches that generate high shading gradients. Often these highly radiant patches are the real light sources, so often high-frequency illumination corresponds with direct illumination, but not always. High frequency illumination must be computed very accurately.
- Low-frequency illumination causes low shading gradients. If it is computed less accurately, it will not be so perceptible in the resulting image. As a consequence, low-frequency illumination can be stored in a mesh.

A classification method is needed that localizes high-frequency components in the shading and the light sources or other strong radiant patches that are responsible for these high frequencies. This mechanism is called source selection or source classification (Kok and Jansen 91; Chen et al 91). Source selection classifies patches to whether they give important radiance contributions or not. When a patch gives an important contribution, the patch is further considered to be a source. Shadow rays will then be cast to this source to compute its direct contribution to points in the scene.

A hybrid system with source selection can be regarded as a generalization for the different types of hybrid methods:

- If none of the patches is selected for shadow ray casting, because the meshing is very fine or because we are just interested in an image of moderate quality at a reasonable speed, the method will be a conventional two-pass method (Wallace et al 87; Sillion and Puech 89).
- If a limited number of patches is selected to be responsible for the high-frequency shading, then the method converts to a method in which shadow rays are cast to the most important patches, and where low-frequency illumination is derived from the results of the radiosity pass (Chen et al 91; Kok and Jansen 91). The method of (Shirley 90a) is a special case of this method in which only all light emitters are regarded to be important.
- If all patches are selected to be sampled with shadow rays, then the algorithm converts to a one-level Monte-Carlo sampling method (Rushmeier 88). In this case the preprocessed radiance value is not used directly for display. Instead, the illumination at a visible point is calculated by sampling the incoming light by sending rays into all directions within the hemisphere placed on this point.

The separation of high-frequency and low-frequency illumination lends itself well to an adaptive refinement strategy (Chen et al 91). This strategy combines the three previously described methods. Initially an approximated image is computed from the results of a progressive radiosity pass at low cost for previewing reasons (first method). Later, this image is gradually refined to obtain a more accurate image by casting shadow rays to the most important sources (high-frequency pass, second method). This second step uses Monte Carlo path tracing, but rays are only directed to directions with selected patches. In this pass also a light ray tracing step is performed to compute the high-frequency caustics. Finally the image can be refined further by re-computing the diffuse interreflections with a one-level Monte-Carlo sampling method (low-frequency pass, third method).

Modifying the method that separates direct from indirect illumination (Shirley 90a) into a method that separates high-frequency from low-frequency illumination, and combining this method with the notion of source selection, a new global illumination method, that consists of a low-frequency pass and a high-frequency pass, is proposed.

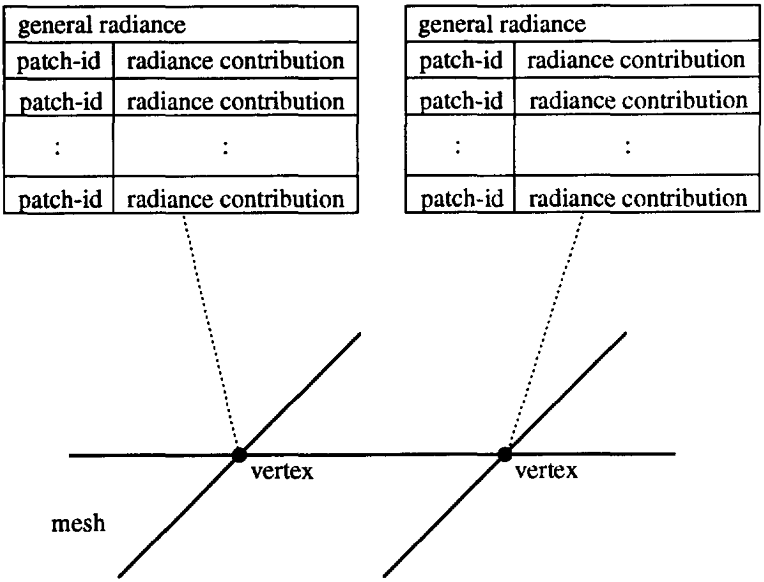
3.2.1 Low-frequency pass

The first pass of the method, the low-frequency pass, makes use of a progressive radiosity algorithm (Cohen et al 88). The meshing of the surfaces consists of patches. Patches are in turn subdivided into elements (Cohen and Greenberg 86). Radiances are computed for the vertices of these elements. However, a distinction must be made between the high- and low-frequency illumination. Although the definitive source selection is performed before the rendering (high-frequency-pass), the low-frequency pass is already adapted to allow this definitive selection. This selection needs the contributions of possible sources to patches. Therefore, during the low-frequency pass a vertex contains, along with the general radiance value, also a list of radiance contributions (figure 3.1). Because it is impossible to store the contributions of all patches to all vertices (this can be compared with storing all form factors in full-matrix radiosity), only a limited number of radiance contributions is stored at each vertex. These possible source patches, the source candidates, are selected with a pre-selection during the low-frequency pass. We call this pre-selection the source candidate selection. After the low-frequency pass each vertex contains a list with contributions of the source candidates and a general radiance value that contains the sum of the contributions of all other patches.

The n most important contributors to a patch can be determined with several strategies.

- An obvious choice is to take the n patches that have shot their power into the environment during the first n iterations of progressive refinement process. These patches have the most power. This is a global selection because each patch in the scene gets the same selection of source candidates.

- Another strategy is to select the sources for each patch separately. Each patch gets its own selection of the candidate source set. Two examples of these local candidate source selection methods are:
  - For each patch the source candidate list contains the *n* patches that provide the highest radiance contributions on the patch.
  - For each patch the source candidate list contains the first *n* patches that contribute to it. Often these first contributors provide the highest radiance values.



general radiance	
patch-id	radiance contribution
patch-id	radiance contribution
:	:
patch-id	radiance contribution

general radiance	
patch-id	radiance contribution
patch-id	radiance contribution
:	:
patch-id	radiance contribution

Figure 3.1 Candidate source list of radiances

A drawback of a local candidate selection is that an important candidate source may not be detected, because its contribution is not recorded on the patch. When the meshing is too coarse, local candidate source selection strategies might miss small details. They might ignore a patch as candidate source although it may give perceptible shading contrasts. Consider a strong light source that illuminates a patch through a small gap. Because of the coarse meshing of the patch, this light is not detected (figure 3.2). Thus the source will not appear in the candidate source list of the patch. A global source candidate selection does not have this problem, it is independent of the meshing.

On the other hand, for very complex scenes with many strong radiant patches, a local candidate selection is preferred. Consider a model of a complete house. It is not likely that a light source in a room at the second floor will contribute to the illumination of rooms on the ground floor, let alone that it will be responsible for perceptible shadows in these rooms. It would be best to have a separate candidate selection for each room. To be able to account for all high-frequency illumination during rendering when a global candidate selection is used and to allow a local definitive source selection before rendering, the number of candidate sources in the radiosity pass should be large, namely all patches that can give shadow gradients somewhere in the scene. This will require much memory to store all contributions.

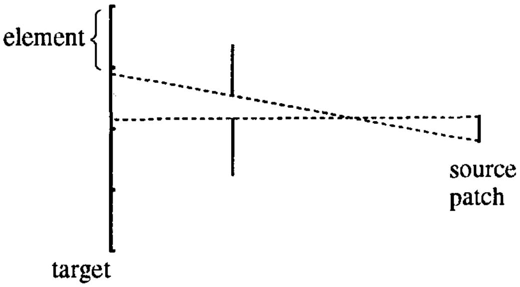


Figure 3.2 Important light is missed, so a local candidate source selection will fail

Whether a global or a local selection is the best strategy is an open question. It depends strongly on the complexity of the scene, and the resolution of the meshing that is used.

When, during the radiosity process, a patch shoots its unshot power into the scene, and the radiance of a vertex must be updated, one of the following situations arises:

- The shooting patch is not in the candidate set but the candidate list contains empty entries. The shooting patch is now added to the candidate set and its radiance contribution is stored in the list.

- The shooting patch is already in the candidate set. The new radiance contribution is added to the entry in the radiance list that corresponds to the shooting patch.

- A local source candidate selection that selects the patches with the highest contributions is used, the shooting patch is not a member of the selected candidate set, and the candidate set contains no more empty entries. The source candidate list is searched for the smallest radiance contribution already stored. If the new radiance contribution is larger than this smallest contribution already stored in the list, this smallest contribution is added to the general radiance value. The new shooting patch is added to the source candidate set and its contribution is stored in the list. If, however, the new contribution is smaller than the smallest contribution in the list, then the new contribution is added to the general radiance value.

- Another candidate selection is used (not a local highest contribution selection), the shooting patch is not in the candidate set, and the candidate set contains no more empty entries. In this case the radiance contribution is added to the general radiance value.

- In all previous situations it is assumed that the vertex is illuminated by the shooting patch without specular reflections. If this is not the case, then the contribution is stored in the general radiance value. The reason is that it is impossible to trace shadow rays via specular reflecting patches. The quality of the caustics is therefore dependent on the mesh resolution used during the radiosity pass. An alternative would be to use an extra light ray tracing step as part of the high-frequency pass (Shirley 90a; Chen et al 91) and to store the caustics in very fine illumination maps. In that case the fact whether a vertex is illuminated via specular reflection or not can be ignored.

The output of the low-frequency pass consists of a mesh of vertices for each patch. Each vertex contains a general radiance value and a list of source candidates with their contributions. The sum of the general radiance and all radiance contributions from the list gives the total illumination at the vertex.

3.2.2 High-frequency pass

The high-frequency pass starts with the definitive source selection. Before the image generation, for each patch in the scene the selection of the n candidate sources is further reduced to m sources ($m \leq n$; m is not constant for all patches). It is again possible to choose between a global and a local selection. However, a global selection ignores the fact that a candidate source may strongly affect some patches, but may have a negligible influence on other patches. Therefore the definitive source selection is usually done locally. Source selection criteria determine which candidate sources will be treated as real sources. A survey of some selection criteria is given in section 3.3.

Applying the selection criteria results in one of the following actions:

- The candidate source is selected as source.
  The contribution from this source will be recalculated more accurately during the rendering. The contribution that was stored in the candidate list is discarded. Sometimes it is useful to keep the sum of the direct contributions of the selected sources for other purposes (see section 4.5).

- The candidate source is not selected as source.
  The contribution from this candidate source is added to the general radiosity value. After that, the contribution that was stored in the candidate list can be discarded.

After the source selection each patch in the scene has a source list that contains references to all patches that will be sampled with shadow rays.

The rendering is done using a ray tracing method. Rays are traced from the eye through the viewing plane into the scene to determine which patches are visible. When a patch is specular, rays are traced into the reflection direction just like conventional ray tracing methods. When a diffuse patch is hit, the radiance at the intersection point must be computed. This radiance is formed of two components:

- Radiance caused by low-frequency illumination.
  This radiance was already computed during the low-frequency pass and is stored in the general radiance values of the vertices on the hit patch. The low-frequency

illumination at this point is computed by a reconstruction (e.g. bilinear interpolation) from the vertex radiances.

- Radiance caused by high-frequency illumination.
  To compute this radiance accurately shadow rays are traced to the important sources: the sources in the source list of the hit patch. The resulting high-frequency illumination is no longer dependent on the meshing.

The sum of these two components gives an accurate radiance at the intersection point.

3.3 Selection criteria

The main goal of source selection is to find for a patch all those patches that contribute to the high-frequency shading. Selection criteria determine which patches are responsible for the high-frequency components in the shading. As mentioned before, selection can be done locally and globally. Both types of selection have their own selection criteria. Various selection criteria can be used. In the following two sections a number of these criteria will be described.

3.3.1 Global selection criteria

A global selection criterion selects the same sources for all patches in the scene. Therefore it only depends on the characteristics of the candidate source. Let patch j be a candidate source, then the following global selection criteria can be used.

Emission criterion

Often light-emitting patches are responsible for the most perceptible shading gradients, for example shadows. Thus by selecting these light-emitting patches many of the perceptible shading gradients are accounted for. The emission criterion selects patch j as source if it is a light emitter, so if $L_j^e > 0$. The direct-indirect hybrid method (Shirley 90a) uses the emission criterion as the only criterion to select sources.

Power criterion

Not only the fact whether a patch is an emitter, but also the total power that a patch radiates is important. Some strong reflecting surfaces may also generate perceptible shading gradients. On the other hand some light emitting patches may be too weak to be responsible for perceptible shading gradients. The power criterion decides to select patch j as source if the total power shot from this patch during the low-frequency pass is larger than a certain threshold power value, so if $\Phi_j > \Phi_{threshold}$. During the radiosity pass this criterion can be used to decide whether a patch is a candidate source, when the n patches with the most shot power are selected as candidate sources.

Size criterion

Small patches will give 'sharp' shadow transitions, so they are responsible for high-frequency shading. Large patches, however, will give smooth shading transitions and can therefore be represented well with a coarse meshing. If the size of patch j is smaller than a given size-threshold value, so if $A_j < A_{threshold}$, then the size criterion decides to regard this patch as a source.

Both the emission and the power criterion can be used as the only criterion. The size criterion should only be used in combination with at least one of the two other criteria. The size criterion favours small patches, because they need only a limited number of shadow rays for sampling.

3.3.2 Local selection criteria

A local selection criterion does a selection of sources depending on the expected interaction between a patch for which a selection of sources must be made and a candidate source. For each patch in the scene the local selection criteria are applied to see which candidate sources are important. Important terms in determining the contribution of a patch to another patch are the irradiance and the radiance. The irradiance at a point on a patch *i* caused by a candidate source *j* is (figure 3.3):

$$E_{di\leftarrow j} = \int_{A_j} L_j \cos\theta_i \delta_{ij} d\omega_j \tag{3.1}$$

The diffuse radiance contribution from a candidate source *j* to a point on a patch *i* is given by:

$$L_{di\leftarrow j} = \frac{\rho_i^d}{\pi} \int_{A_j} L_j \cos\theta_i \delta_{ij} d\omega_j \tag{3.2}$$

In both equations the delta solid angle can be expressed as:

$$d\omega_j = \frac{\cos\theta_j}{r^2} dA_j \tag{3.3}$$

If we can find those patches *j* that are responsible for large differences in irradiance and so possible large differences in radiance on patch *i*, then a good selection can be made. The following criteria attempt to do that.

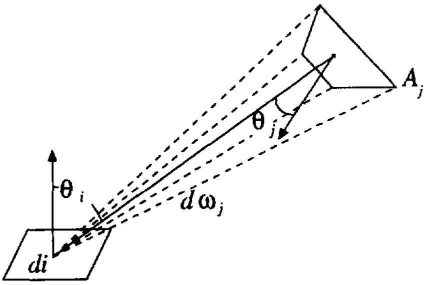


Figure 3.3 Geometry for local source selection

Irradiance criterion

The unoccluded irradiance is the irradiance that would reach a patch if there were no occlusions between patch *i* and source *j* (so when δ_{ij} is set to 1 in formula 3.1). The irradiance criterion decides to select a candidate source *j* if the unoccluded irradiance on patch *i* is larger then some threshold irradiance value, so if $E_{di\leftarrow j} > E_{threshold}$. If the unoccluded irradiance is high, then there is a good chance that patch *j* can give perceptible shadows on patch *i* when the light source is partly occluded. If the unoccluded irradiance is low, then the possible shadow transitions will be less obvious. Therefore the pre-computed (meshed) radiance values can be used. This local criterion should be performed for at least one point on each patch. If, however, a patch is large, then it is advisable to apply this criterion for several points on the patch, as the irradiance may vary a lot over the patch. If for one of these points the criterion decides to select the source, then the source should be selected for the complete patch.

An alternative irradiance criterion is to sort the source candidates on their unoccluded irradiance. The first few candidate sources (that have the highest contribution to the receiving patch) are selected. This method has been used in (Ward 91).

Radiance criterion

It is often useful to take into account the diffuse reflectance of the receiving patch when selecting its important sources, and to use the unoccluded radiance value for selection. When this diffuse reflectance is very small or zero, the radiance, and so the possible differences of radiance on a patch, are very small although the irradiance values may be large. It is therefore better to use a radiance criterion. If the unoccluded radiance at a patch caused by a candidate source is larger than a threshold radiance value, so if $L_{di\leftarrow j} > L_{threshold}$, then this candidate source is selected as source.

Gradient criterion

Although the value of the radiance of a patch is important for finding possible high-frequencies on a patch, the distribution of the radiance on a patch is of more importance. If the light coming from a candidate source gives large radiance variations over a patch, then this candidate source should be considered as source for the patch. On the other hand, if it can be detected a priori that light from a strong candidate source illuminates the whole patch or does not reach this patch at all, then this candidate source is not responsible for high-frequencies on the patch. The gradient criterion compares the pre-computed candidate source radiance contributions at the neighbouring vertices of a patch. If the difference of the radiance contribution of candidate source *j* to vertex $v_{i,v}$ of patch *i* and the radiance contribution to its neighbouring vertex $v_{i,n}$ is larger then a given gradient threshold value, so if equation 3.4 is true, then candidate source *j* will be selected as source for patch *i*.

$$\left\| L_{v_{i,v}\leftarrow j} - L_{v_{i,n}\leftarrow j} \right\| > L^{gradient}_{threshold} \tag{3.4}$$

The efficiency of this criterion depends strongly on the meshing. If the meshing is very coarse, then shadows, and so gradients, can be missed easily. Also, gradients can become large because of a large difference in orientation between neighbouring sample points on a coarsely subdivided patch, although they are both completely illuminated.

Radiance-difference criterion.

The source contribution list of a vertex contains the radiance contribution of a candidate source. This contribution can be compared to the unoccluded radiance value for the vertex. First, the difference in radiance between the contribution from the list and the unoccluded radiance is computed:

$$\Delta L_{v_i \leftarrow j} = \left\| L^{list}_{v_i \leftarrow j} - L^{unoccluded}_{v_i \leftarrow j} \right\| \tag{3.5}$$

If this difference is almost zero, then the vertex is completely illuminated by the candidate source. Otherwise at least a part of the light source is not visible for the vertex, and some shadow occurs on the patch. The radiance difference criterion decides to select source candidate j as source for patch i if for one of the vertices on patch i the radiance difference is larger than some threshold value, so if $\Delta L_{v_i \leftarrow j} > \Delta L_{threshold}$. This criterion should only be applied when the contributions of path j to the vertices of patch i that are stored in the list are not all equal to zero. In the case that all contributions are zero, patch i is completely in shadow, and it is not necessary to select source j.

This criterion does not perform so well for a patch with very smooth penumbra regions. One of the vertices may then have a large radiance difference and so the source is selected. However, the smooth penumbra can be represented appropriately with a coarse meshing. Therefore it is not necessary to select the source.

Solid-angle criterion

If the integral of equation 3.1 is approximated by ray casting, then the accuracy of the results will depend on the resolution of the rays, i.e. the number of rays per solid angle. If a candidate source is small (for example a point light source) or is seen with a small solid angle from the patch, then only one or a few shadow rays are needed. Shadow rays are then effective, and it is worthwhile to select the patch as a light source. If, however, the candidate source is an area light source and the solid angle is large, then many rays are needed and so it is not so attractive to select the candidate source. Also, the soft shadow that may be expected from an area light source can often very well be represented by the meshing.

Thus, if the solid angle is below a threshold value, so if $\omega_j < \omega_{threshold}$, then candidate source j will be selected for patch i. Just as the irradiance criterion for large patches, this criterion must be applied for several points on the patch, because the solid angle may vary over the patch.

When a source is selected, the solid-angle criterion also decides how many shadow rays must be cast to this source. This number is equal to the solid angle of the source divided by a maximum solid-angle-size value $\omega_{\max}$. A good value for this maximum size is 0.005 steradians (Campbell and Fussell 90). When source selection is applied in combination with adaptive source sampling (see chapter 5), the solid angle criterion can determine the initial source sampling rate. Of coarse, in this case a larger maximum solid-angle size is used.

View criterion

Patches farther away from the view point are not so critical for the perceived image quality. The projection of such a patch onto the screen can be computed with the viewing parameters. If the number of pixels that the patch will occupy is known, the accuracy of the meshing can be determined. If each mesh element occupies only one or a few pixels, then interpolation of the pre-computed radiance contributions gives a good representation of possible shading gradients. No source needs to be sampled with shadow rays. When the projected mesh elements are too large, the shadow rays are cast to the sources selected by the other criteria. If the maximum projected element sizes in x and y direction (figure 3.4) are both smaller than a threshold value, so if $p_{x,e_i} < p_{threshold}$ and $p_{y,e_i} < p_{threshold}$, then no shadow rays are cast at all.

When images are generated for animation purposes, the view criterion should rather not be used. It might be possible that this criterion gives different selections for two successive frames, resulting in sudden changes in shadows. Another problem is that the view criterion does not take into account the fact that surfaces can also be seen after one or more specular reflections. This may also cause problems when curved specular surfaces (mirrors) exist in the scene. If a patch is seen magnified in a mirror, then also the size of the elements is magnified. Sometimes the size of the element may become too big, although the direct projection on the screen is not.

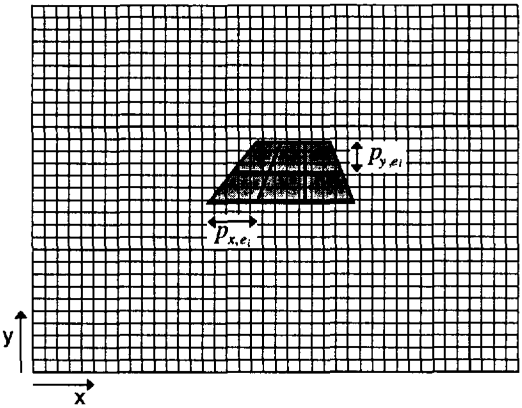


Figure 3.4 View criterion

Each of these criteria can be applied to make a selection. Some of them, however, do not perform well in all circumstances or do not limit the number of sources sufficiently. Therefore a combination of these criteria should be used.

A final remark about local source selection is that two neighbouring patches may get a different selection of sources. This may lead to discontinuities in the shading, although the shading should be continuous. It is therefore advisable to make source selections not for individual patches, but for clusters of patches (i.e. objects).

3.4 Results and conclusions

Results

The proposed hybrid radiosity method with source selection is applied to three models with different complexity. A description of these models is given in Appendix A. A radiosity pass that uses directed shooting is applied. Statistics for this pass are given in table 3.1.

The number of iterations for the progressive radiosity for models A and B is determined by an ambient threshold value. After each iteration a remaining ambient value, representing the contribution of the unshot power that still has to be shot, can be calculated (Cohen et al 88). When this value is larger than the threshold value, the process continues. For model C the process is stopped after 8000 iterations although the threshold value was not reached yet. To reach the same threshold value as for models A and B many iterations are necessary because many very small patches exist. Although the meshing for these models is not very fine (because direct illumination is recomputed during rendering), still the number of rays, and therefore the execution times, are high. This shows that radiosity is an expensive process.

model	A triangle	B small room	C computer room
iterations	3	1703	8000
rays	19840	68524556	212840688
time[1]	00:00:15	09:00:27	26:28:00

Table 3.1 Statistics radiosity pass

During the radiosity a global source candidate selection is performed. The first *n* patches that shot their unshot power during the radiosity pass are selected. Model A has only one source candidate: the square light source. Model B has four source candidates: two spot light sources, and two area sources. For Model C all twelve area sources at the ceiling are candidates.

Several images are generated of the models. For each model one image is made as a 'standard' two-pass method. Radiances for visible points are interpolated from mesh radiances. One image is made with global source selection. Shadow rays are cast to all source candidates. The number of shadow rays that is cast to each light source is fixed:

- For model A 64 shadow rays are cast to the light source.
- For model B 1 shadow ray is cast to each of the two spot light sources, 64 shadow rays are cast to the area source on the ceiling and the area source (window) on the

---

[1] All timings are done on a Sillicon Graphics Iris Indigo.

left wall. Experiments have shown that these numbers are necessary to insure smooth shadows everywhere (Kok and Jansen 91).

- For model C 16 shadow rays are cast to each of the twelve area light sources on the ceiling.

Also one image of each model is made using a local source selection. For each patch in the model it is determined independently whether shadow rays need to be cast to a light source, and if so how many. The threshold values used by the selection are chosen to ensure good shading accuracy. Statistics are listed in table 3.2.

model	A triangle	B small room 1	B small room 2	C computer room
picture size	240x240	400x400	400x400	800x800
viewing rays	38125	176989	193251	641601
interpolation				
shadow rays	0	0	0	0
sampling time	00:00:06	00:00:50	00:01:49	00:03:21
global selection				
shadow rays	2440000	16654885	19525916	70639928
sampling time	00:06:00	01:28:28	02:53:32	06:48:45
local selection				
shadow rays	2331328	6394347	7968714	15923128
selection time	00:00:00	00:00:01	00:00:01	00:00:33
sampling time	00:05:46	00:35:07	01:12:52	01:28:28

Table 3.2 Statistics rendering pass

Several remarks can be made:

- A 'standard' two-pass method (indicated by interpolation in table 3.2) renders the fastest. However, the shading is inaccurate, because the meshing that was applied was not fine enough.

- For model A, local selection does not much improve efficiency. The ground plane, that covers the largest part of the image, contains important shading gradients and therefore the light source was selected.

- For more complex scenes local selection reduces the number of shadow rays and therefore execution times considerably. The following reductions of shadow rays are obtained: model B view 1: 62%, model B view 2: 59%, and model C: 77 %. A few examples of the reduction due to local selection are given now. No shadow rays are cast from the books in model B and the walls in model C. The gradient criterion detects that no shading boundaries are expected on these surfaces. The view criterion decides not to cast shadow rays from the lid of the teapot in model B view 1. The lid covers only a few pixels, and therefore the mesh elements on the lid are small enough and cover only one or two pixels. The radiance gradients are therefore represented properly by the mesh. In view 2 of this model, however, the projection on screen is larger. Therefore shadow rays are cast to compute the contributions of the sources.

- The efficiency gained with local selection depends on the selection criterion threshold values. The values used for the tests in table 3.2 are determined by experiments with several models. These values give good image quality for all scenes tested. Limiting the range for which a criterion decides to select a source may result in a greater speed up of the rendering.

An example of the influence of the value of a selection criterion value is given. In the solid angle criterion the maximum allowed solid angle ω_{max} determines how many shadow rays are cast to a source. A small value means many rays, a large value few rays. Several values are applied to model C. Figure 3.5 shows the number of shadow rays for several values of ω_{max}. Figure 3.6 shows what errors are made for these values.

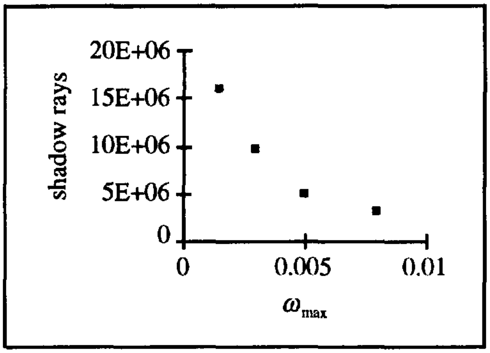


Figure 3.5 Number of shadow rays for several values of ω_{max}

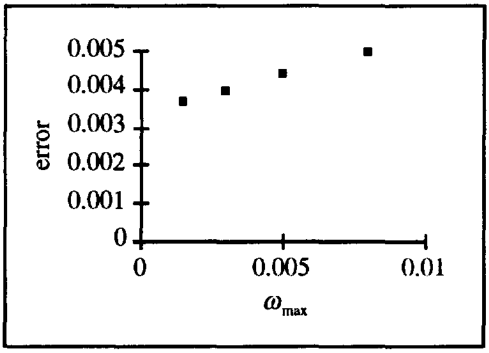


Figure 3.6 Errors made for several values of ω_{max}

The errors are measured by comparing the images with the image generated with global source selection. The error metric used is:

$$\varepsilon = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{(p_r(i)-q_r(i))^2+(p_g(i)-q_g(i))^2+(p_b(i)-q_b(i))^2}{3}} \tag{3.6}$$

where n is the number of pixels in the image, $p_{rgb}(i)\in[0,1]$ and $q_{rgb}(i)\in[0,1]$ are the pixel RGB values of pixel i of the two pictures p and q that are compared. The result is an error $\varepsilon\in[0,1]$. Extrapolation of the graph of figure 3.6 shows that always an error will be made, no matter what $\omega_{\max}$ is used. There are several reasons for that. Firstly, during rendering random numbers are used (for example to jitter in screen space and to jitter on sources). Secondly, except for the solid angle criterion, other criteria are also responsible for errors. Some shadows are not detected by these other criteria (e.g. gradient criterion), because the meshing that was used during the radiosity pass was rather coarse for some patches.

- The selection time is limited compared to the sampling time. It might therefore be a good idea to investigate whether more complicated selection criteria, that take some more time, can make better selections.

Conclusions

A global illumination method that separates the high-frequency shading computations from the low-frequency computations generates images with a high accuracy at reasonable computational costs. The low-frequency shading is computed with a radiosity method. The meshing can be rather coarse. The high-frequency shading is computed during the rendering by tracing shadow rays. This high-frequency component is only computed for parts of the scene that are actually visible in the image.

Source selection detects where high-frequency shading transitions might occur. Combining the different selection criteria and varying threshold values used in these criteria make a very flexible global illumination method. This method can simulate all methods within the range from a standard two-pass method to a one-level Monte-Carlo sampling approach.

A disadvantage of the presented and implemented method is the large memory requirement. Instead of storing one radiance value for each vertex, also a list of radiance contributions from source candidates is stored. However, these contributions are only needed to do source selection. If it is known in advance (before the radiosity pass) which light sources will be selected anyhow, then it is not necessary to store their contributions.

Although source selection limits the number of shadow rays by selecting carefully the sources to sample, this number may still be very high, especially when rather large area sources are selected. Shadow coherence and adaptive source sampling methods, as described in chapters 4 and 5, will be needed to further reduce the number of shadow rays.

4 The shadow image buffer

4.1 Introduction

Computation of ray-object intersections is the most expensive part within ray tracing. For good quality images the number of rays to be traced, and therefore the number of ray-object intersections, can be very large. Much research is done to accelerate the ray tracing process. Ray-object intersections are optimized, and (hierarchical) bounding volumes and spatial subdivision techniques are used to reduce the number of ray-object intersections. A good survey of these methods is given in (Arvo and Kirk 89).

Often shadow rays are responsible for a considerable part of the computational costs. Shadow rays determine how much light from a light source is received by a point in the scene. A ray is traced from the point to the light source, and if an intersection is found between the point and the source, then the point lies in shadow, otherwise the point is lit by the light source. The number of shadow rays can be enormous. Source selection methods as described in chapter 3 may limit the number of light sources to be sampled and hence the number of shadow rays to be traced. However, the number of rays can still be very large, especially when large area light sources are involved. Conventional ray tracing efficiency techniques, such as bounding volume and spatial subdivision techniques, are also applicable for shadow rays, but because of the special character of shadow rays, also specialized shadow ray acceleration methods can be used.

This chapter describes methods for accelerating shadow ray casting. In section 4.2 previous shadow ray culling methods are discussed. Section 4.3 describes shadow coherence. All the methods described in section 4.2 exploit shadow coherence in object space. A method that exploits shadow coherence in image space, the shadow image buffer, introduced in (Woodward 90), is described in section 4.4. This method is extended so that it will be more generally applicable. In section 4.5 the efficiency and applicability of the shadow image buffer are tested and evaluated.

4.2 Shadow ray culling methods

Several culling methods, specialized for shadow rays, have been designed. These methods attempt to limit the number of shadow rays or attempt to reduce the number of intersection tests for a shadow ray.

The light-buffer method (Haines and Greenberg 86) uses a cube around a light source: the light buffer. The planes of this cube are subdivided into small square cells. Each cell contains a depth sorted list of the surfaces enclosed by the frustum formed by the source origin and the cell. This list is constructed in a preprocessing step by projecting the surfaces in the scene onto the cells of the planes. Each cell contains also a depth value of total occlusion. This is the distance after which all further surfaces are in shadow. When a shadow ray should be traced from a point to the source, first the cell is determined through which the ray will go. If the distance of the point to the source (the shadow-ray distance) is larger than the total-occlusion distance of the cell, then the point is in shadow. If this is not the case, then intersection computations of the ray with the surfaces with a smaller depth than the shadow-ray distance are performed until an intersection is found (point in shadow) or until all surfaces of the cell are processed (point in light). The light buffer may give an important improvement in shadow ray casting cost, but the memory requirements (of $O(LN^2m)$, where L is the number of light sources, NxN the cell resolution of a cube plane, and m the number of surfaces in the scene) and the preprocessing time are large.

Many ray tracers are equipped with a voxel-traversal algorithm. This will also improve the efficiency for shadow ray casting. But in addition, the voxels can be extended to reduce shadow ray casting even more.

The hybrid shadow testing method (Eo and Kyung 89) uses shadow volumes in combination with a voxel grid structure. The shadow polygons of the shadow volumes are stored in the voxel grid structure. During viewing-ray traversal to find the visible point a count is kept that indicates whether the ray traverses a shadow volume or not. This method has several disadvantages. The number of shadow polygons and so the memory requirements are large. The method is only applicable for point light sources. It is impossible to account for curved surfaces that will generate arbitrarily shaped shadow volumes.

In the voxel occlusion testing method (Woo and Amanatides 90) each voxel is assigned a 2-bit field for each light source. This fields indicates the occlusion type: full, null, or complicated occlusion. The assignment is done in a preprocessing by projecting shadow umbra on the voxels. When one wants to know whether a point is in shadow or not, the occlusion value of the voxel in which the point is situated is examined. If this value is full or null occlusion, no other actions have to be done. Either the voxel (and therefore all points in the voxel) is totally visible by the light source or the voxel is completely in shadow. However, when a complicated occlusion is found, the voxels are traversed to the light source. In this case intersection tests are performed with the objects in the voxel. The voxel traversal continues until an intersection with an object is found or until a voxel is traversed that has full or null occlusion. The performance of this algorithm depends highly on the resolution of the voxel grid. It performs well for high resolutions. The extra storage requirements are $2LN^3$ bits, where L is the number of light sources, and NxNxN the voxel resolution of the uniform voxel subdivision. The preprocessing time for each light source is rather high, especially when curved surfaces exist in the scene. This method can be extended to be suited for area light sources, by identifying all penumbra regions as complicated shadow and always tracing rays for these areas. However, in this case the method is less efficient.

All three given methods require a preprocessing. In this preprocessing, a projection method is used to identify shadow areas in the scene. The time this preprocessing takes depends on the number of surfaces and the number of light sources in the scene. The number of surfaces is usually very high, especially because for all these methods curved surfaces need to be subdivided into many small polygons. In the hybrid shadow testing method, the necessary subdivision into polygons will also effect the accuracy of the shadows.

4.3 Shadow coherence

Computer graphics methods often use the concept of coherence. The definition of coherence given in (Foley et al 90) is:

> Coherence is the degree to which parts of an environment or its projection exhibit local similarities.

Many types of coherence are used within computer graphics. Roughly speaking, coherence methods can be divided into methods that exploit coherence in object space and methods that exploit coherence in image space.

Coherence also applies to shadows. Although shadow patterns caused by several light sources may be quite complex, the shadow shapes caused by each individual light source tend to be very homogeneous, both on the surfaces in the scene (in object space) and projected on the screen (in image space). We call this shadow coherence.

The light buffer, the hybrid shadow testing, and the voxel occlusion testing are methods that use shadow coherence in object space. The shadow image buffer (Woodward 90; Kok et al 91), as described in the following section, uses shadow coherence in image space.

4.4 Shadow image buffer

Unlike earlier methods, the shadow image buffer exploits shadow coherence directly in image space in combination with an adaptive image refinement technique (Whitted 80). Consequently, shadow information is only computed for parts of the scene that are visible in the image. This information is not computed in a separate preprocessing step, as for the object space methods of section 4.2, but during the rendering.

According to the method presented in (Woodward 90), a two-bit deep shadow buffer (the SIB) with a size equal to the screen resolution is attached to each light source. This shadow buffer can be added to the frame buffer. Each entry in the SIB corresponds with a pixel in the image. An entry in the SIB can have one of the three following values:

- UNKNOWN: It is not known whether the visible point is in shadow or not. The initial value of all entries.

- UMBRA: The point that is visible at the pixel is in shadow.
- LIGHT: The point that is visible at the pixel is illuminated and not in shadow.

After a first low resolution ray trace pass, in which rays are traced on a certain refinement level, a pixel on a not evaluated refinement level keeps the value UNKNOWN if its neighbouring pixels, for which rays already have been traced have different SIB values. This indicates that a shadow boundary exists in this region and that additional shadow sampling will improve the shadow. However, if all its neighbouring values are the same and are unequal to UNKNOWN, then the pixel gets the SIB value of its neighbours. No extra shadow sampling is necessary.

However, it is not necessary to have the value UNKNOWN, when the SIB values are evaluated during the ray tracing instead of as a separate step. In that case, it can be detected on the fly that an entry was not evaluated before. Therefore a one-bit deep shadow buffer for each light source suffices, with two values: UMBRA or LIGHT. The total storage requirements for the shadow image buffer are therefore at most *LPQ* bits for a picture of size *P*x*Q* with *L* light sources. Note that the memory requirements are independent of the number of surfaces in the scene and that no preprocessing is needed.

First the image is sampled at a low resolution. All shadow rays are computed as usual and the shadow results are stored in the SIB. Figure 4.1 gives the algorithm for the initial (low resolution) sampling.

```
for all viewing samples (i,j) at level 0 do
   TraceRay(i,j,p)                /* p is intersection point */
   for each light source L do
      SIB(L,i,j) = CastShadowRay(p,L)
      if SIB(L,i,j) == LIGHT then
         Shade(i,j,p,L)
```

Figure 4.1 Low resolution sampling algorithm for shadow image buffer method

TraceRay(*i,j,p*) traces a viewing ray. The indices (*i,j*) identify the viewing ray (viewing sample). They determine the point in the viewing plane through which the viewing ray passes. If an intersection is found then this point is stored in *p*. CastShadowRay(*p,L*) casts a shadow ray from point *p* to light source *L* and returns the result UMBRA when the point *p* is in shadow or LIGHT if the point is lit by light source *L*. Shade(*i,j,p,L*) computes the radiance of point *p* due to light source *L*, knowing that no occlusions exist, and adds this value to sample (*i,j*).

When the image is refined, the evaluation of a new sample on refinement level *l* ($l > 0$) is performed with the algorithm of figure 4.2.

The effect of this algorithm is that during image refinement only shadow rays are cast at shadow boundaries where the neighbouring SIB values differ. The obtained shadow values are stored in the SIB for the next pass. Figure 4.3 shows an example of how the shadows of a sphere are refined in successive passes.

```
for all viewing samples (i,j) at level l  do
  TraceRay(i,j,p)
  for each light source L do
    if all neighbouring SIB(L,i,j) at level l-1 are equal then
      shadow_result = value of neighbouring SIB
    else
      shadow_result = CastShadowRay(p,L)
    if shadow_result == LIGHT then
      Shade(i,j,p,L)
    if l not antialiasing level then
      SIB(L,i,j) = shadow_result
```

Figure 4.2 Shadow image buffer refinement algorithm

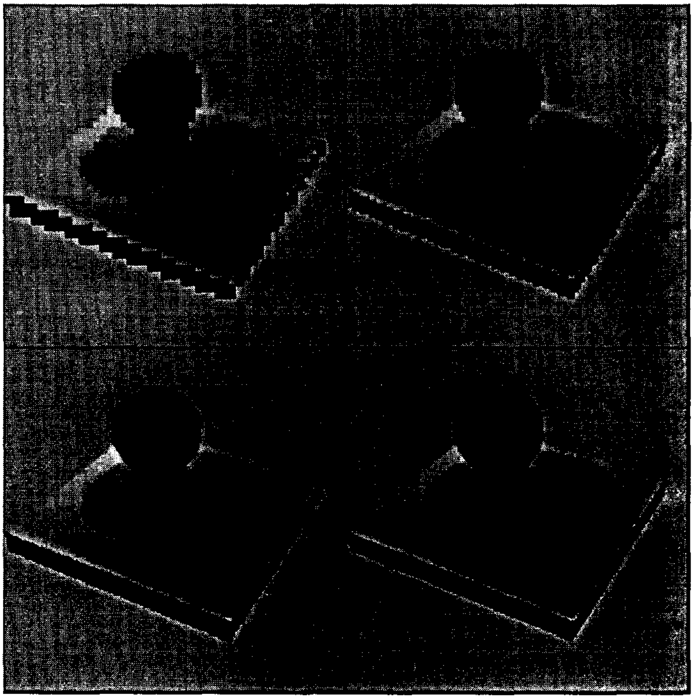


Figure 4.3 Shadow refinement

The shadow image buffer as described above is applicable for light sources that are small enough to be sampled with only one shadow ray, e.g. point light sources. The shadow image buffer can be extended to be applicable for area light sources in one of the following ways:

- Area light sources can be represented as a number of point light sources. Each point light source then gets one SIB entry. This method is only applicable if it is known in advance in how many point sources the light source must be subdivided. It can not be used in combination with an adaptive source sampling strategy. Besides, when the subdivision of the source is very fine, the memory requirements may be high.
- Each area light source gets one SIB entry. Three different shadow values are now possible:
  - UMBRA: The visible point is totally in shadow.
  - LIGHT: The visible point is totally illuminated by the area light source.
  - PENUMBRA: The visible point is partly in shadow of the light source and partly illuminated by the light source.

  A two-bit deep shadow image buffer is now necessary. The algorithm is given in figure 4.4.

```
for all viewing samples (i,j) at level l  do
  TraceRay(i,j,p)
  for each light source L do
    if all neighbouring SIB(L,i,j) at level l-1 are UMBRA then
      shadow_result = UMBRA
    else if all neighbouring SIB(L,i,j) at level l-1 LIGHT then
      shadow_result = LIGHT
      ShadeAreaSource(i,j,p,L)
    else    /* PENUMBRA */
      shadow_result = CastAreaShadowRaysAndShade(i,j,p,L)
    if l not antialiasing level then
      SIB(L,i,j) = shadow_result
```

Figure 4.4 Shadow buffer image refinement algorithm for area light sources

CastAreaShadowRaysAndShade(*i,j,p,L*) casts shadow rays from point *p* to one or more points on the light source *L*. It returns UMBRA if all rays are blocked, LIGHT if no rays are blocked, and PENUMBRA otherwise. This routine also computes the radiance of point *p* using the information of the shadow ray casting, and adds this radiance to sample (*i,j*). A shadow image buffer with one (2-bit) entry for an area light source will save fewer rays than subdividing the area sources into point sources, but it requires less memory and can be used well in combination with an adaptive source sampling algorithm as described in chapter 5.

Instead of storing the SIB in the frame buffer attached to the light sources, as proposed in (Woodward 90), the SIB can also be stored with the sampling data, e.g. the primary (viewing) rays.

Usually, the sampling datastructure contains information about the screen-sampling process. For each viewing ray this datastructure contains information about which object(s) is/are hit, what the sampled radiance value is, etc. This information is used to filter pixel radiance values and to decide whether to sample adaptively in certain areas of the screen.

The SIB can also be added to this sampling datastructure. For each light source one SIB entry is added to each record in this datastructure. Storing the SIB with the sampling data has the following advantages:

- The SIB-information can also be stored for samples at sub-pixel (anti-aliasing) level. This gives more accurate and efficient use of the shadow image buffer at these levels.
- The SIB-information can be used as an extra criterion to sample adaptively. The SIB indicates for which light sources the shadows should be refined and for which light sources not. If SIB-values for one or more light sources of neighbouring samples show differences, then one can decide to sample the screen in this area more accurately. From the intersection point shadow rays are cast to the source(s) that were responsible for the differences in SIB-values. The contribution from the other sources can be found by inspecting the SIB-values corresponding with these sources.

  An alternative, that reduces the computations even more, is not to trace the primary ray for the new sample to find the point that is visible, but to interpolate this point from the visible points of the surrounding samples. From this interpolated point shadow rays are traced to the source that was responsible for the differences in SIB-values. Compared with previous method it is now not necessary to trace primary rays to anti-aliase shadow boundaries.

4.5 Results and conclusions

Results

The effectiveness of the shadow image buffer is tested. The shadow image buffer is applied in combination with adaptive screen sampling. The following criteria are used to determine whether the screen is refined:

- Object criterion.
  If neighbouring samples (viewing rays) do not all hit the same object, then extra samples are taken to discover the exact object boundaries.
- Shadow criterion.
  If neighbouring samples do not all hit the same shadow property (all LIGHT or all UMBRA, which can be seen from the SIB), then a shadow boundary exists and

extra samples are taken in this region to find the exact shadow boundaries. This criterion is only applied to real point-light sources, because only these sources give sharp shadows that need to be antialiased.

- Difference (gradient) criterion.
  If the radiance of neighbouring samples differs more than a certain threshold, then extra samples are taken. Because of the use of the shadow image buffer, no shadow rays are needed for these samples.
- Texture criterion.
  Textures are sources of aliasing (Heckbert 86). Therefore textures should be sampled with many rays. If the neighbouring samples hit a texture, then extra samples are taken. As in the difference criterion, extra samples generated by this criterion also do not need shadow rays.

The shadow image buffer is applied to all three models described in appendix A. The number of shadow rays that is shot to each light source is fixed, and is the same as for the experiments in chapter 3. For each delta area on an area source a SIB-entry of one byte is used. The results are given in tables 4.1 to 4.4. For all images screen sampling starts at level -1, which means that one viewing ray is shot for 4 pixels. If started with fewer viewing rays per pixel, small shadow details can be missed initially, resulting in missed or distorted shadows in the image. At this initial level all shadow rays are shot. When refining at level 0 (1 viewing ray per pixel), all viewing rays are shot (non-adaptively to avoid missing small objects). Shadow rays are only cast when necessary, as indicated by the SIB. At levels beyond level 0, the anti-aliasing levels, the screen is sampled adaptively with the criteria described before. Adaptive sampling is done until level 2 (16 samples per pixel).

The results show that the shadow image buffer method can reduce the number of rays significantly. At levels greater than 0 (adaptive image refinement for anti-aliasing), the shadow-ray reduction of the SIB decreases. The adaptively refined samples are mainly caused by object boundaries, for which the SIB is not used. The SIB is only effective if the decision to shoot extra viewing rays is made due to difference in sample values, due to shadow boundaries of point light sources, or due to textures.

For model B the SIB is not effective for diffuse hits after specular reflection in the teapot. This teapot is partly specular and partly diffuse. For each sample only one entry can be stored: the information of the first diffuse hit. For the diffuse hits of secondary rays the SIB cannot be used, and all shadow rays are cast. This affects the results of model B negatively. If the teapot was only diffuse or only specular, the reduction of rays would be larger.

The SIB can also be used in combination with adaptive sampling with the routine of figure 4.4, with a two-bit deep SIB. The results for the different models is given in tables 4.5 to 4.8.

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	9616	615424	0	0	0
0	28509	93548	1731028	94	71
1	4815	130470	177690	58	69
2	4702	241471	59457	20	65

Table 4.1 Shadow buffer results for model A triangle

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	44431	4159252	0	0	0
0	132558	2744784	9750968	78	59
1	74513	5452214	1258498	19	47
2	160195	12002881	2637616	18	36

Table 4.2 Shadow buffer results for model B small room, view 1

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	48613	4894622	0	0	0
0	144638	2799838	11831541	81	61
1	72403	5141718	1417365	22	51
2	157674	11232727	3024989	21	40

Table 4.3 Shadow buffer results for model B small room, view 2

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	160801	17702213	0	0	0
0	480800	8591020	443444836	84	63
1	220197	18762195	2684043	13	51
2	552536	41639110	7055930	14	38

Table 4.4 Shadow buffer results for model C computer room

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	9616	96650	0	0	0
0	28509	145444	143746	50	37
1	18351	325020	0	0	20
2	50213	932678	0	0	9

Table 4.5 Shadow buffer results for model A triangle, adaptive sampling

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	44431	3292228	0	0	0
0	132558	5589163	4284521	43	33
1	74814	4846647	327670	6	25
2	160879	10541496	737659	7	18

Table 4.6 Shadow buffer results for model B small room, view 1, adaptive sampling

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	48613	1541570	0	0	0
0	144638	3001201	160860	35	26
1	74629	2152775	331443	13	22
2	163895	4685253	766175	14	19

Table 4.7 Shadow buffer results for model B small room, view 2, adaptive sampling

level	diffuse hits	shadow rays	shadow rays saved	% shadow rays saved	% saved cumulatively
-1	160801	7957232	0	0	0
0	480800	7899202	15891736	67	50
1	226335	9181019	1469503	14	40
2	561073	19875268	4047295	17	32

Table 4.8 Shadow buffer results for model C computer room, adaptive sampling

As expected, the efficiency improvement of the SIB is in this case less than the efficiency improvement of the one-bit deep SIB. If only one of the shadow rays that are cast to an area light source has a different shadow type than the shadow types of corresponding shadow rays of neighbouring samples, then, during image refinement, all shadow rays have to be cast, while for the one-bit deep SIB only one shadow ray is cast. For the triangle model the use of the SIB does not improve efficiency at all while anti-aliasing. Image refinement is applied because of the object criterion, in which case the shadow image buffer is not used, or because of the difference criterion. When the difference criterion is applied, the samples are in a penumbra region, in which case shadow rays are always shot. In the other models, other criteria also decide to refine the image, and so the shadow image buffer can sometimes be applied.

Conclusions

The shadow image buffer is a shadow ray culling method that exploits shadow coherence in image space. It is able to reduce the number of shadow rays during the image refinement process. The shadow image buffer is an extension to standard image coherence methods. Its efficiency is based on a better discrimination that can be made to refine the image:

- It gives an extra criterion to decide to refine sampling, namely the existence of shadow boundaries that should be anti-aliased.
- When the scene contains multiple light sources, the SIB indicates for which light sources(s) the shadow boundary should be antialiased and therefore be sampled with shadow rays. The contributions of the other light sources can be computed with the information in the SIB.

This method has several advantages compared to other shadow ray culling methods:

- The shadow image buffer imposes practically no time overhead to the basic ray tracing algorithm. It is also simple to implement.
- Unlike other shadow acceleration methods that typically rely on polygonal approximations, the shadow image buffer is accurate also with curved objects inasmuch the shadow boundaries are traced by rays using exact geometries.
- Because of its compactness and the independence of the number of surfaces in the scene, the shadow image buffer is particularly useful for scenes with many surfaces and multiple light sources.
- The shadow image buffer requires no preprocessing.
- The shadow image buffer enables us to adaptively refine the image space sampling to anti-aliase shadow boundaries.

The shadow image buffer method has also some disadvantages:

- The accuracy of the shadow image buffer is dependent on the initial sampling level and the homogeneity of the shadows. When the initial level is too coarse, small shadows are easily missed. At refinement levels these shadows are further ignored. It is true that for the examples given in (Woodward 90) the shadow image buffer gives savings of 44% up to 90%, but that is because the shadow areas are very coherent. Therefore adaptive image refinement can be started at a coarse level. In real scenes often small, less well shaped shadows occur. Therefore the initial level should be not too low. The shadow image buffer should mainly be used for anti-aliasing. However, then the efficiency of the shadow image buffer will be not so good.

- The shadow image buffer uses one entry for each light source for each primary (viewing) ray. This raises a problem when partly diffuse and partly specular surfaces exist. If such a surface is hit, the radiance is computed and the shadow result is stored in the SIB. However, one or more secondary rays are traced along the specular reflection direction. Each of these secondary rays can again require to store shadow information. Therefore a tree of shadow information should be stored for each primary ray. It is too expensive to store a multilevel shadow image buffer (with an entry for each light source for each node in the tracing tree). We therefore chose to store only the shadow information for the first (partly) diffuse intersected surface. For diffuse surfaces found at higher levels in the tree, we always cast shadow rays. This reduces the effectiveness of the shadow image buffer. If the rendering is part of a two-pass rendering method it is also possible to interpolate radiance contributions computed in the radiosity pass instead of casting shadow rays for these higher level intersections, because often the contributions of higher level intersections are less important for the accuracy of the pixel value, so they do not need to be computed very accurately.

Concluding, the shadow image buffer can save shadow ray casting, but for complex scenes (with small objects and general reflectance functions), the efficiency is limited. Nevertheless, it is useful to apply the shadow image buffer to save shadow rays whenever it is possible, because the shadow image buffer involves almost no overhead.

5 Adaptive light source sampling

5.1 Introduction

Area light sources do not only create umbra and full-light regions, but also penumbra regions. For these regions, only a part of the light source is visible. An area source must be sampled with a large number of shadow rays to determine the exact contribution of this source to the shading of a point. This can be an expensive process. Therefore a good light source sampling strategy, that minimizes the sampling effort, is desirable.

This chapter describes sampling of area light sources. In section 5.2 several source sampling methods are discussed. This discussion shows that an adaptive sampling method will perform best. Section 5.3 describes an adaptive stochastic source sampling method. This method can be extended with the use of sampling pattern coherence as described in section 5.4. In section 5.5, the methods are compared with non-adaptive sampling.

5.2 Sampling area light sources

The diffusely reflected radiance on a point $\mathbf{x}$ caused by a light source S is given by the following integral:

$$L(\mathbf{x}) = \frac{\rho^d(\mathbf{x})}{\pi} \int_{x' \in S} v(\mathbf{x},\mathbf{x}')L(\mathbf{x}')\cos\theta \frac{\cos\theta'}{\|\mathbf{x}'-\mathbf{x}\|^2} dA' \tag{5.1}$$

In this formula, $\mathbf{x}'$ is a point on the light source, $\rho^d(\mathbf{x})$ is the diffuse reflectance at point $\mathbf{x}$, $v(\mathbf{x},\mathbf{x}')$ is the visibility term indicating whether points $\mathbf{x}$ and $\mathbf{x}'$ are mutual visible, θ the angle between the normal at $\mathbf{x}$ and the direction of the incoming light, θ' the angle between the normal at $\mathbf{x}'$ and the direction of the outgoing light, and dA' the differential area around $\mathbf{x}'$ on S (figure 5.1).

An exact analytic solution for this integral is not feasible. Ray casting is the best way to determine the visibility term $v(\mathbf{x},\mathbf{x}')$. If a ray from $\mathbf{x}$ to $\mathbf{x}'$ does not intersect other surfaces, then this term is one, otherwise zero. Source S contains an infinite number of points $\mathbf{x}'$, and thus a proper way to sample the source (determining to which points $\mathbf{x}'$ rays will be cast) is necessary. Several source sampling strategies are possible.

The simplest form of sampling is uniform or regular sampling. The source is subdivided into equal-sized areas and one shadow ray is cast to the center of each of these areas (figure 5.2). A problem in this approach is how to find a good sampling density. A coarse density results in aliasing and can miss occlusions. A high density

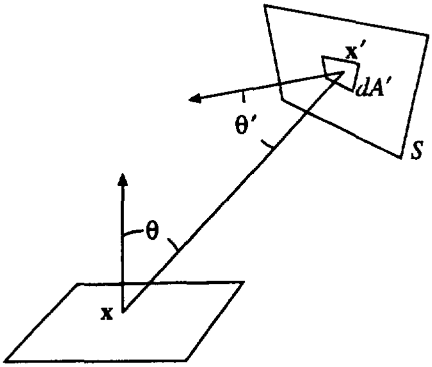


Figure 5.1 Source sampling geometry

may be too expensive. In general, the sampling density should be based on the solid angle formed by the point and the source and on the properties of the scene within this solid angle. If it contains many small surfaces then a high sampling density is necessary. If it does not contain any surface at all, then a less high density is sufficient. However, normally this information is not available and is expensive to compute.

A second approach is to sample adaptively. During sampling, information is gathered where occlusion transitions occur. An adaptive sampling strategy takes advantage of the sampling information by subdividing the source locally into more areas where occlusion transitions occur. An occlusion transition is a difference of mutual visibility between two neighbouring areas on the source and the point to be shaded. The advantage of this adaptive sampling compared to uniform sampling is that the sampling effort is directed to regions where it is really necessary: where shadow transitions occur. An accurate fraction of occlusion, and therefore an accurate estimate of the energy the point receives, is obtained in this way with a reasonable number of shadow rays. A disadvantage of regular adaptive sampling is that the initial sampling rate can miss small shadow details if the sampling rate is chosen too coarsely. An example of adaptive source sampling is given in figure 5.3. Instead of casting 96 rays for uniform sampling (figure 5.2c), only 42 rays are cast for adaptive sampling (figure 5.3c), while the resulting accuracy is the same. In (Wallace et al 89; Tampieri and Lischinski 91) this adaptive sampling of sources is used to estimate the form factor between a point and a shooting patch during radiosity processing.

Another approach is stochastic source sampling. Instead of sending rays to regularly determined points on the source, shadow rays are cast to random positions on the source (figure 5.4). Stochastic methods suffer less from aliasing effects. They trade aliasing for noise. After a number of shadow rays is cast, the variance of the samples can be examined. If the variance is high a decision can be made to take more random samples (figure 5.4b and 5.4c). A disadvantage of this pure random sampling is that it may take many rays, and so a long time, before the variance reduces.

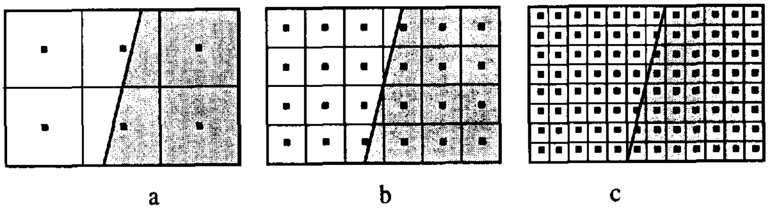


Figure 5.2 Uniform sampling

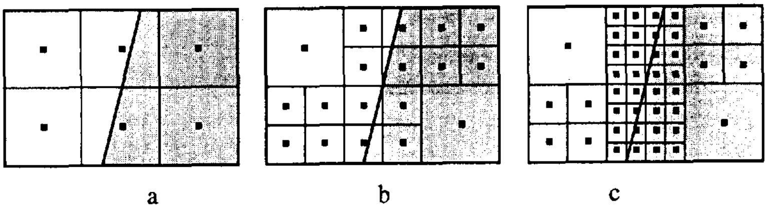


Figure 5.3 Adaptive sampling

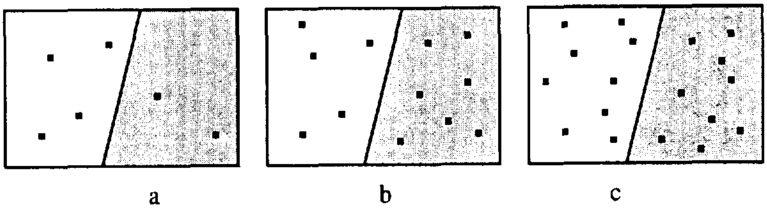


Figure 5.4 Stochastic sampling

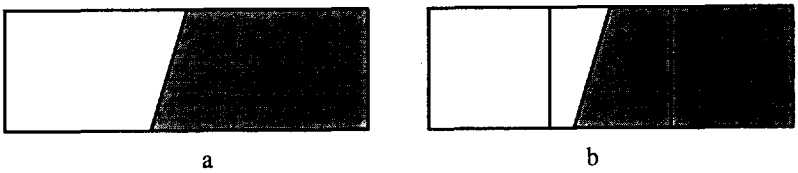


Figure 5.5 Stratification

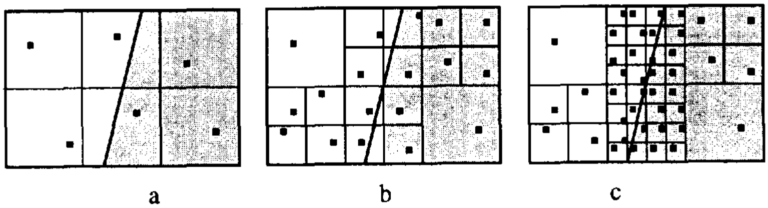


Figure 5.6 Adaptive stochastic sampling

Several methods can be used to improve stochastic sampling.

In a pure random determination of sampling positions on the source these positions may not be distributed evenly over the source. This may result in high variances. A better distribution can be obtained with Poisson disk sampling (Dippé and Wold 85). However, it is expensive to generate such a sampling pattern. Therefore less expensive sampling methods like jittered sampling (Cook et al 84) and n-rooks sampling (Shirley 90b) are often used instead. For example, in jittered sampling, the source is subdivided into areas, and one random position is determined in each of these areas.

Another method to improve stochastic sampling is stratification. The sampling domain (here the source) is subdivided into several subdomains. Each of these subdomains is then sampled independently. This may reduce the variance of the total domain in cases where the subdivision is done in such a way that several subdomains will be homogeneous, i.e. do not contain shadow transitions. This can be illustrated by the following example from (Lee et al 85). If the rectangle is treated as one domain (figure 5.5a), then with random sampling within this domain it may take a while before the variance is below a given threshold. If the domain is subdivided (figure 5.5b), the variance in the uniform areas will be zero, which will reduce the variance of the total domain.

This process can be repeated if the sampling is done adaptively. The non-uniform area can be subdivided again. The old samples are reclassified or thrown away, and new samples are added (figure 5.6). Adaptive sampling done in this way can optimally take advantage of the variance reduction qualities of stratification.

Applying these notions to the sampling of light sources, it then seems advantageous to subdivide the light source into separate subdomains. Some of these domains may be completely unoccluded as seen from the sample point, some may be completely occluded by obstructing objects, and some may be partly occluded (as in figure 5.6b). For the first two categories the variance will be small and the sampling effort can be directed to the last category. This can be done dynamically by directing the sampling to the subdomain with the highest variance. If the number of samples in a subdomain is above a certain level and there still is a high variance, then the subdomain can be split again (as in figure 5.6c), and so on, until the overall variance satisfies the image quality criteria.

The discussion above clearly indicates that adaptive stochastic sampling will give the most optimal results taking into account speed and quality. Therefore this sampling strategy is used to sample area light sources.

5.3 An implementation of adaptive stochastic source sampling

A light source can be parametrized in u and v direction, with $(u,v) \in [0,1] \times [0,1]$. This parametrization is used to subdivide the source into smaller elements. A bintree (the sampling tree) is used to construct and store the source sampling pattern. The source is alternately subdivided in *u* and *v* direction. Each leaf node in the resulting binary sampling tree represents a small area of the light source to which one shadow ray is cast. An example of the subdivision of a source and the corresponding bintree is given in figure 5.7.

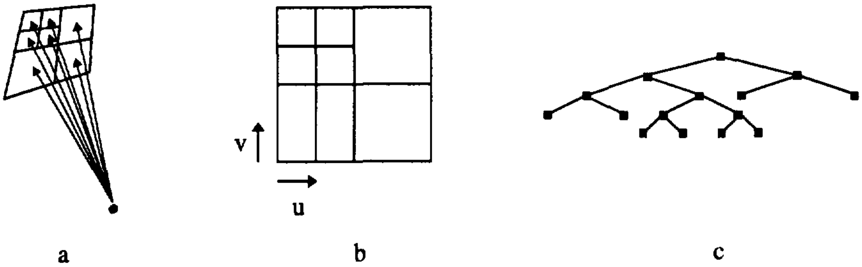


Figure 5.7 Area source sampling (a), subdivision of source (b), and corresponding bintree (c)

Two types of nodes can be distinguished in the tree: leaf nodes and internal nodes. A leaf node represents only one sample (one shadow ray). An internal node contains gathered information of all leaf nodes in its subtrees. A node contains the following entities:

- Shadow flag.
  The shadow flag indicates what type of shadow is generated by this part of the light source. Tree values are possible:
  - UMBRA. If the node is a leaf node, then the sample to the corresponding area of the source showed that no light of the source reaches the point to be shaded. If the node is an internal node, then all samples in its subtrees have shadow flag UMBRA.
  - LIGHT. If the node is a leaf node, then the sample to the corresponding area of the source showed that light of the source reaches the point to be shaded. If the node is an internal node, then all samples in its subtrees have shadow flag LIGHT.
  - PENUMBRA. This shadow type is only valid for internal nodes. Some but not all samples in its subtrees have shadow flag UMBRA, and some have shadow flag LIGHT.

  An example of a sampling pattern and the shadow flags at each node is given in figure 5.8.
- Sampling mean.
  The sampling mean contains the weighted average of the contributions of all samples in the subtrees of the node. If the node is a leaf node, then the sampling mean contains the contribution of one shadow ray to the corresponding part of the source.

- Sampling variance.
  The sampling variance is the variance of the samples of the subtrees of the node. For a leaf node this variance is zero.
- Number of samples.
  It contains the number of samples already cast in the corresponding area, so the number of samples in the subtrees of the node.
- Position.
  For a leaf node, position contains the *u* and *v* coordinates of the position of the sample. When the area corresponding with a node is split, the position parameters determine in which one of the two new areas the existing sample must be stored.
- Level.
  Level is the level in the sampling tree at which the node occurs.
- Depth.
  It contains the longest distance from the node to one of the leaf nodes in its subtrees, measured in subdivision levels.

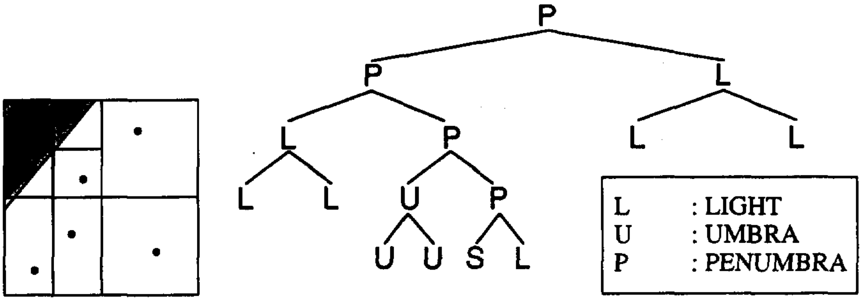


Figure 5.8 Source subdivision and shadow flags

When the contribution of a light source *S* to an (intersection) point *P* has to be calculated, first an initial number of shadow rays is cast. This initial number can be chosen based on the information derived during the radiosity pass (chapter 3), or can be made dependent on some parameter, for instance the solid angle with which *S* is seen by *P*. The source is regularly subdivided and a shadow ray is cast to a jittered point in each of these areas.

The results of the sampling are stored in the leaf nodes of the sampling tree. The results of the sampling are also passed to the ancestors of the leafs, so that each node in the tree will have the mean, the variance, the type of shadow (UMBRA, LIGHT, or PENUMBRA) it represents, and the number of rays (leafs) in the subtrees. Then the source is further subdivided in regions where shadow transitions occur or where the variance is high. When an area to be subdivided is found, the leaf node that represents this area becomes an internal node and two new leaf nodes are created. The old sample is reclassified and stored in one of the new leaf nodes. For the other new leaf node a shadow ray is cast to the corresponding part of the source, and the resulting sampling information is stored within this node. Then the sampling tree is updated with the new information. Refinement is continued until a stop criterion is reached.

To find the area to be subdivided a recursive algorithm is used that descends the sampling tree from the root. At each node a decision is made which subtree is descended first. It is important to concentrate the sampling on areas with shadow transitions and with high variances. A simple approach is just to compare the variances of the two subnodes of the node and to search in the node with the highest variance. A high variance can, for example, be caused by shadow transitions, or can occur when the number of samples is small. However, because of the randomness of the method, a node can have a small variance because the distribution of the samples in its subtree was not appropriate. Therefore it is better to use an approach that searches more directly for the areas that are expected to have a high variance. Subtrees that contain a shadow transition or represent only a limited number of samples are searched first. The recursive procedure that searches the area to be refined is given in figure 5.9.

```
boolean FindAreaToSubdivide(n,n1,n2)
/* Find area on source to subdivide further. Return TRUE when */
/* an area has been found, else return FALSE                  */
node    n           /* input node */
node    n1, n2      /* output nodes (to be subdivided) */

  node      s1, s2
  node      try1, try2   /* prefer try1 over try2 */
  boolean   found        /* areas are found ? */
  boolean   s1nb, s2nb   /* have neighbours same shadow ? */

  SubNodes(n,s1,s2)      /* s1 and s2 are children of n */
  if s1 and s2 not subdivided then
    if s1->level == maximum level then
      return(FALSE)      /* no more subdivision possible */
    else
      n1 = s1, n2 = s2   /* areas to be subdivided, CASE 1 */
      return(TRUE)

  /* s1 and s2 were already subdivided */

  if s1->shadow != PENUMBRA && s2->shadow == PENUMBRA then
    try1 = s2, try2 = s1             /* prefer s2, CASE 2 */
  else if s1->shadow == PENUMBRA && s2->shadow != PENUMBRA then
    try1 = s1, try2 = s2             /* prefer s1, CASE 2 */
  else if s1->shadow == PENUMBRA && s2->shadow == PENUMBRA then
  if s1->depth < s2->depth || s1->nr < s2->nr ||
     (s1->nr == s2->nr) && random < 0.5) then
    try1 = s1, try2 = s2          /* prefer s1, CASE 3 */
  else
    try1 = s2, try2 = s1          /* prefer s2, CASE 3 */
else /* s1->shadow != PENUMBRA && s2->shadow != PENUMBRA */
  s1nb = CompareNeighbourSamples(s1)     /* TRUE if same */
  s2nb = CompareNeighbourSamples(s2)     /* shadow types */
  if s1nb && !s2nb then
    try1 = s2                     /* choose s2, CASE 4 */
    if s1->variance is high then
      try2 = s1
    else
      try2 = NULL
  else if !s1nb && s2nb then
    try1 = s1                     /* choose s1, CASE 4 */
    if s2->variance is high then
      try2 = s2
    else
      try2 = NULL
  else if !s1nb && !s2nb then
    if s1->depth < s2->depth || s1->nr < s2->nr ||
       (s1->nr == s2->nr && random < 0.5) then
      try1 = s1, try2 = s2        /* prefer s1, CASE 5 */
    else
      try1 = s2, try2 = s1        /* prefer s2, CASE 5 */
  else /* s1nb && s2nb */
    if n->variance is high then
      if s1->nr < s2->nr || random < 0.5 then
        try1 = s1, try2 = s2      /* prefer s1, CASE 6 */
      else
        try1 = s2, try2 = s1      /* prefer s2, CASE 6 */
    else
      n1 = NULL, n2 = NULL        /* no subdivision */
      return(FALSE)

found = FindAreaToSubdivide(try1,n1,n2)
if !found && try2 != NULL then
  found = FindAreaToSubdivide(try2,n1,n2)
return(found)
```

Figure 5.9 Algorithm to find area of source to be subdivided

The function CompareNeighbourSamples compares the shadow type of a node with the shadow types of nodes of neighbouring areas on the source. If the shadow type is equal to the shadow types of all neighbours and, then this function returns TRUE, otherwise FALSE.

When a decision has to be made at node *n* whether to refine its subnodes *s1* or *s2*, the following cases can be distinguished (in decreasing order of importance):

- Case 1

  s1 and *s2* are not subdivided. In this case the areas to be subdivided are found. Both *s1* and *s2* are subdivided.

- Case 2

  The shadow type of only one of the subtrees *s1* and *s2* is PENUMBRA. The subtree that contains the shadow value PENUMBRA certainly contains a shadow transition. It is selected first. If recursive traversal of this subtree does not result in new samples, because sampling in this subtree was already performed until maximum level, then the other subtree is chosen to be searched.

- Case 3

  Both subtrees *s1* and *s2* have shadow type PENUMBRA. Extra samples should be traced in the area where the confidence in the sampling is the smallest. This will be in the area that contains the least number of samples. So the subtree that contains the smallest number of samples is traversed first. However, sometimes both subtrees contain the same amount of samples. In this case the subtree with the smallest depth value, thus subdivided less fine, is chosen first to be selected. If however the depth value of both subtrees is equal than the subtree to be selected first is chosen randomly. Again, when no area to be sampled is found in this subtree, the other one is chosen.

- Case 4

  None of the subtrees has shadow type PENUMBRA. The shadow types of both *s1* and *s2* are compared with the shadow types of the nodes of neighbouring areas at the same level in the tree. If only *s1* has a different shadow type, then one of its neighbours, and *s1* is traversed to find the area to be sampled. If only *s2* has a different shadow type, then *s2* is traversed further. The other subtree is not searched for areas to be sampled if in the chosen subtree no areas can be found to be sampled, because no shadow transitions are expected, unless the variance in this subtree is very high (see also case 6).

- Case 5

  None of the subtrees has shadow type PENUMBRA and both subtrees have other shadow types than their neighbours. In this case the subtree that contains the smallest number of samples, or the smallest depth value is chosen first. If both the numbers of samples and depth values of the subtrees are equal, the subtree is chosen randomly. If in the selected subtree no areas to be sampled can be found then the other one is selected.

- Case 6
  None of the subtrees has shadow type PENUMBRA and both subtrees have the same shadow types as their neighbours. Probably the areas that both subtrees represent contain no shadow transitions. The search for areas to be sampled in subtree *s* can be stopped. However, the variance in one of the subtrees *s1* or *s2* is very high, it might be wise still to take one or a few extra samples. It is possible that the source was very large and that the shadow rays are situated far away of each other. Obstructors may be situated in between those rays. It is wise to take a few extra samples to reduce the possibility of missing these obstructors in case of a high variance.

When the area to be subdivided is found, it is subdivided in *u* or *v* direction, depending on the level of subdivision. The sample that was taken already in the area is put in the new node that covers its position. A new sample position is determined in the other new node by jittering over its area. After this position is sampled with a shadow ray the results are stored at the node. The tree is ascended to the root and each node is updated (new mean, variance, etc.).

It is possible that small shadow details will be missed. Therefore the bintree is restricted. The subdivision level of two neighbouring not further subdivided areas is not allowed to differ more than a given maximum value. In this way areas around shadow boundaries are sampled more densely and shadow gradients are detected earlier.

An example of the sampling process is given in figure 5.10. After an initial sampling (5.10a), more samples are taken along the shadow boundary (5.10b-e). Now the level difference between the lower left area and the samples to the right is too large. Therefore the left part is subdivided further (5.10f). Subdivision is continued along the shadow boundary (5.10g-h). In (5.10i) again a subdivision is done because of level difference.

The adaptive source sampling is stopped when the mean seems to be a good estimate. That is when the standard deviation is within a certain percentage (user defined) of the computed mean. If the variance (and therefore the standard deviation) of the samples stays high, then probably part of the source is obstructed by some objects, or the solid angle with which the area source is seen is very large (then each sample represents a rather different energy contribution). In these cases more samples are needed to get a good approximation of the amount of light that is received. Sometimes, however, the variance will reduce very slowly, because of a major intensity gradient (for example a sharp shadow), but the mean of the samples does not change very much. Therefore a stop criterion based on the change in the mean sample value after a few iteration steps may be used in addition. If this mean shows only minor changes, then the shadow sampling can be stopped also. This stop criterion based on difference in average is only applied if enough shadow samples have been cast already.

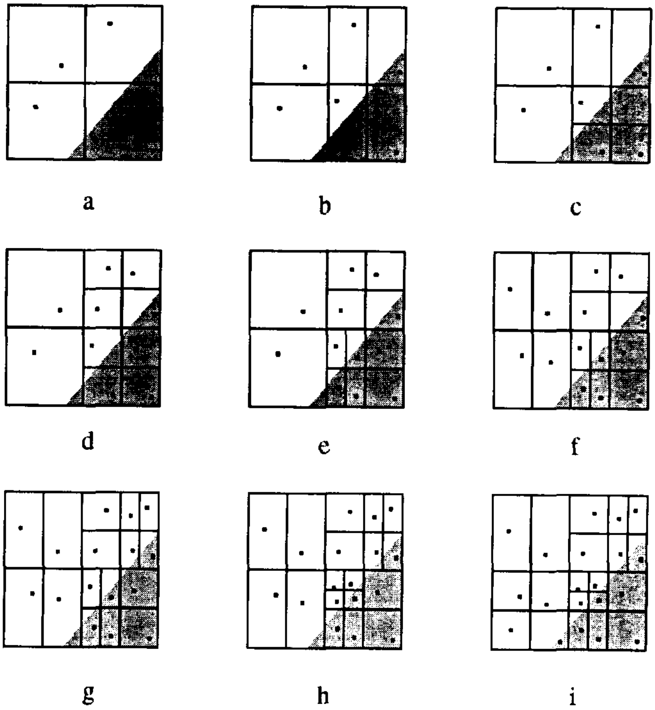


Figure 5.10 An example of area sampling

5.4 Sampling pattern coherence

A light source *S* is sampled from an intersection point *P* of a viewing ray with an object. The intersection point *Q* of the next (neighbouring) viewing ray with the same object will probably have a similar sampling pattern for light source *S*. If there is a shadow boundary for *P*, then it will be moved only slightly for *Q* (figure 5.11).

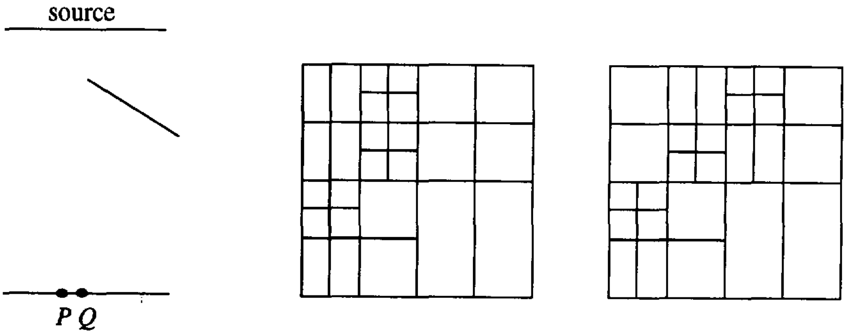


Figure 5.11 Similarity source sampling patterns for neighbouring points P and Q

This similarity between source sampling patterns for neighbouring viewing rays shows a special case of shadow coherence: shadow pattern coherence. This shadow pattern coherence can be used in two ways:

- Sampling pattern coherence during adaptive image refinement.
- Sampling pattern coherence during sampling in scanline order.

Sampling pattern coherence can be applied within an adaptive image refinement strategy by storing the sampling pattern for each ray. After an initial low resolution sampling with viewing rays, the image is locally refined by casting additional viewing rays. For a new viewing ray the sampling patterns of neighbouring rays can be compared, and only new shadow rays have to be cast for areas where the sampling patterns differ. In figure 5.12, *P* and *Q* are viewing rays evaluated during low resolution screen sampling. Source sampling patterns were generated to compute the contribution of a source to *P* and *Q*. *R* represents a viewing ray generated during image refinement in between *P* and *Q*. If *R* hits the same object as *P* and *Q* then the sampling pattern for *R* can be made by identifying the differences in the patterns for *P* and *Q*. Only for areas where the sampling patterns differ, shadow rays must be traced to compute the radiance contribution to *R*. Only the highlighted parts in the sampling pattern of *R* in figure 5.12 need sampling with shadow rays. This method can save many rays, but the memory requirements are rather high, because for each viewing ray a sampling pattern has to be stored for each source.

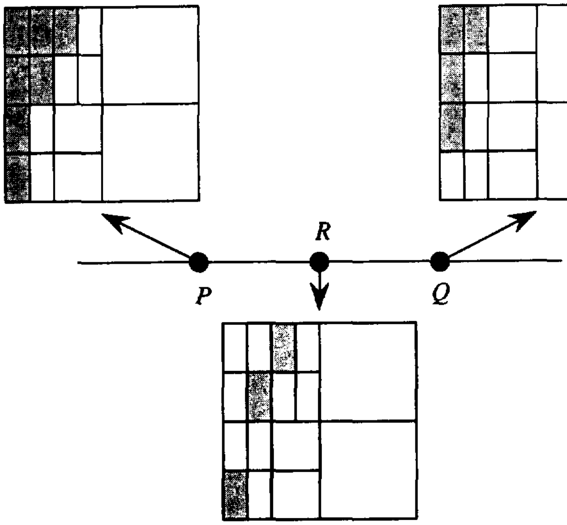


Figure 5.12 Sampling pattern coherence during adaptive image refinement

The storage of the sampling patterns for each viewing ray is not needed when sampling pattern coherence is exploited when the screen is sampled in scanline order. For consecutive source samplings on the scanline the shadow transition moves only slightly, and so the sampling pattern will show only minor changes (figure 5.13). A sampling pattern for a new point can therefore be generated by adapting the sampling pattern from the previous point. Given a sampling pattern from a previous sampling P, the new sampling pattern Q can be generated by re-using pattern P, and re-sampling the source with the same pattern. However, there may be areas in the pattern that are uniform but subdivided because of previous tests. To avoid unnecessary sampling, the pattern has to be reduced. All areas where no shadow transitions exist, but that were nevertheless subdivided are combined to be one area. The reduction of the sampling tree is done as follows (figure 5.14). If the shadow flag for subtree s is LIGHT or UMBRA, then the subtrees of $s1$ and $s2$ gave no new shadow information and so can be eliminated. The reduction should not be continued to a level below the initial sampling rate to avoid missing shadows. The reduced sampling pattern is used as an initial pattern for the new source sampling. First the reduced pattern is resampled. A shadow ray is cast for each leaf node. Next, extra adaptive sampling is done to account for the change in shadow transitions. The sampling pattern coherence used in scanline order sampling, will not necessarily reduce the number of rays for shadow sampling, but will reduce the noise in the image, because rather good initial sampling patterns are used.

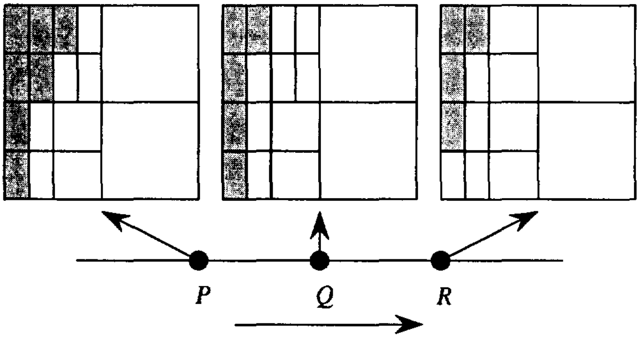


Figure 5.13 Sampling pattern coherence in scanline order

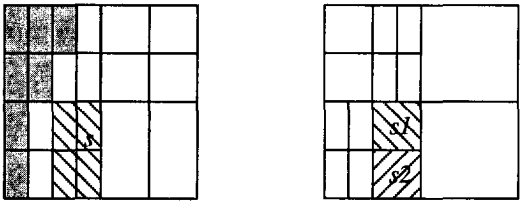


Figure 5.14 Reduction, original pattern (left) and pattern after reduction (right)

5.5 Results and conclusions

Results

The effectiveness of the presented adaptive source sampling method is tested with model A. This model contains large penumbra areas that need to be sampled properly.

Firstly, the routine to find the area to be subdivided of figure 5.9 is compared with a more simple method. This simple method does not search for the shadow transitions, but when a decision has to be made which node to refine, always the node with the highest variance is chosen. In figure 5.15 the results of several tests with different allowed standard deviations for the stop criterion are shown. The error is computed with formula 3.6. The results show that for the same number of rays the error made with the new approach is better. The shadow rays are more often sent to really important areas on the source.

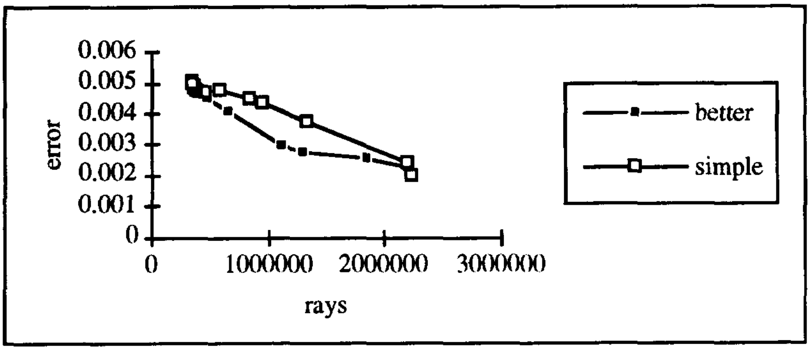


Figure 5.15 Results 'better' adaptive source sampling method compared with a 'simple' method

Often the variance of the sampling stays high, but the mean of the sampling shows only minor changes during the sampling. In this case an extra stop criterion based on the mean will further reduce the number of rays. To prove this assumption, tests were done with the extra stop criterion, that says that, when the mean does not change much, sampling can be stopped. The results of figure 5.16 show that in this case fewer rays are necessary to get the same error. It can be concluded that this extra criterion performs well.

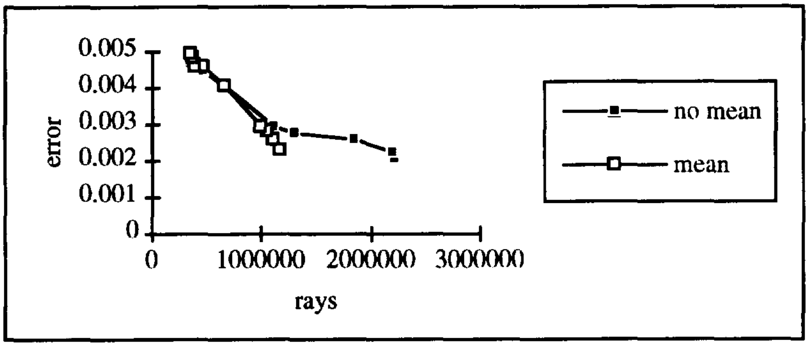


Figure 5.16 Results adaptive source sampling with and without mean stop criterion

Figure 5.17 shows the results, when sampling pattern coherence is applied by re-using sampling patterns during sampling in scanline order. The number of shadow rays used for a certain maximum allowed standard deviation is higher. However, the error is much smaller. The maximum allowed standard deviation can be set much larger, and still good results will be obtained.

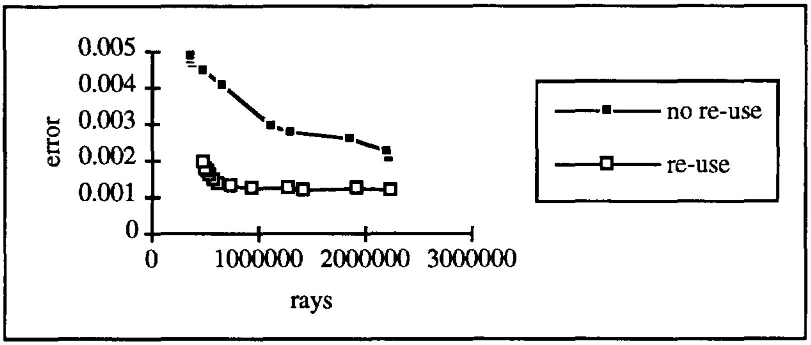


Figure 5.17 Results of use of sampling pattern coherence in scanline order

Adaptive area light source sampling (with and without pattern coherence) is also applied during generation of images from the other models. A maximum standard deviation of 0.05 was allowed. The results are given in table 5.1. The errors in table 5.1 are mainly caused by the randomness of the total ray tracing process (jittering in screen space, etc.). The errors due to the adaptive source sampling process are very small. When the images are compared visually, the shadows look the same.

model	B small room 1	B small room 2	C computer room
shadow rays fixed	16654885	19525916	70636928
without coherence			
shadow rays	13167972	3931885	22701674
% saved	21	80	68
error	0.0017	0.0023	0.0025
with coherence			
shadow rays	13368684	5122521	24806964
% saved	20	74	65
error	0.0015	0.0016	0.0020

Table 5.1 Results adaptive source sampling

Conclusions

Methods are presented to reduce the large number of shadow rays that are needed for sampling large area light sources. The area light sources are sampled in an adaptive way using statistics derived during the sampling process. With this method most of the sampling effort is directed to areas where it is needed, for example in penumbra regions or in areas close to the source.

The method that is used to find the areas to be subdivided provides better results than a more simple approach. However, the time to find the area is larger because more comparisons are done. However, the total time to generate images is usually smaller than with the simple approach because the reduction of execution time because of the reduction of rays is larger than the extra time necessary for the more complicated search algorithm.

Using sampling pattern coherence, a better sampling quality can be obtained with the same number of shadow rays because we have an estimation of where shadow boundaries might occur. When a small shadow is found, this information is passed to the next sampling point, avoiding spatial aliasing. Often sampling pattern coherence will reduce the number of rays. This is caused by the fact that fewer shadow rays are cast to areas where no shadow transitions exist, because a larger standard deviation threshold can be used for the stop criterion to obtain the same image quality.

6 Hierarchy within the radiosity process

6.1 Introduction

The quality of the results of the radiosity process strongly depends on the discretization (meshing) that is applied to the surfaces in the scene. To obtain accurately represented shading gradients, the surfaces must be discretized into very small patches to obtain accurate shading transitions.

However, the number of form factor computations depends strongly on the number of patches in the scene. It is therefore worthwhile to investigate whether adaptive meshing schemes can reduce these computations while providing the same accuracy for the radiosity processing. Adaptive meshing schemes may lead to a hierarchy of discretization levels. For each receiving patch during shooting, an optimal level may be chosen at which it will receive power, based on the expected interaction with the source patch.

Chapters 6 and 7 describe the use of hierarchy within progressive radiosity. Section 6.2 describes why some kind of hierarchy is necessary during the shooting in the radiosity process. Examples of existing methods that use some kind of hierarchy are given. A combination of these methods may lead to an optimal hierarchical radiosity method, that is described in section 6.3. Section 6.4 describes the implementation of one part of the hierarchy: adaptive receiving levels. Section 6.5 gives results and conclusions. Chapter 7 describes another part of the hierarchy: grouping.

6.2 Hierarchical meshing

Radiosity methods compute the global illumination of a scene by calculating the interreflections in the scene. Ideally, the interreflection between all points must be computed. This is not possible. Instead, the power exchange between patches with a finite area is calculated, assuming that these patches have a constant radiance value. To approximate the exact solution these patches should be as small as possible.

However, very small patches may cause some problems:

- In undirected or Monte-Carlo shooting (Malley 86), there may be a mismatch between the density of the rays and the resolution of the mesh. Neighbouring patches that should receive (approximately) the same power are not guaranteed to receive the same number of rays. Because of the randomness of the rays, one patch may receive more rays than its neighbouring patch. Thus the power received by the patches will not be equal, resulting in aliasing artefacts. A solution to this problem is to cast a very large amount of rays from the source patch, resulting in a large computation time.

- In directed shooting (Wallace et al 89) a ray is traced from the source to each patch (or sample point in the meshing) in the scene. When the scene contains many patches (and so many sample points), many rays must be traced. The time for each iteration will be large.

However, it is not necessary to use the same mesh density during the total radiosity process. Sometimes small patches are necessary, sometimes large patches can be used, because none or only unimportant shading transitions result from the interaction between two patches. For example, the exchange of power between two rather large patches that are situated far away from each other is usually constant over these patches. Seen from one patch, the other patch is already small, and need not to be subdivided into smaller patches anymore. Power can be exchanged while they are large. Only when a large patch must exchange power with another patch nearby and occluding objects are expected between these patches, the large patch should be subdivided into many smaller patches. Also, a large patch can be responsible for a very small form factor when it is far away from another patch. In this case it can even be useful to combine patches and compute only one form factor for the whole group of patches. Once again such a group can be small and so several of these groups can be again grouped into larger groups.

Adaptive sampling in combination with some kind of hierarchy results in an accurate and efficient radiosity process. For each iteration in the progressive radiosity process the level of interaction between the selected source and the other objects in the scene should be based on the expected interaction. When a high interaction is expected, the interaction takes place on a fine level, and when almost no interaction is expected, interaction takes place on a coarse level.

The first effort to use some kind of mesh hierarchy was the use of patches and elements (Cohen et al 86). The scene is subdivided into patches. In turn the patches are (adaptively) subdivided into elements. Elements are only used to receive illumination. Power is sent out at patch level. This method strongly reduces the number of form factor computations, while still a good shading resolution is obtained. Elements can be adaptively subdivided when neighbouring elements show large differences in computed radiance. For the new elements form factors are computed and the system of equations is solved again. This can be repeated several times, giving more accurate shading transitions in each iteration.

Another method to use hierarchy in radiosity was presented in (Hanrahan et al 91). In this hierarchical meshing method all surfaces are recursively subdivided. This subdivision is stored in a quadtree, in which each node represents a part of the surface. When power is exchanged between two surfaces, the level in the quadtree is determined at which the interaction should take place. This is done both for the source patch as well as for the receiving patch. Form factor estimates determine the interaction levels for the source and receiver patch. If an estimate is too large, then a higher interaction level is used. For this new interaction level again form factors are estimated, and when necessary, higher levels are selected. When the interaction levels of the source and receiver surface are determined, power can be exchanged. When a surface receives power at some interaction level, then all nodes at levels beyond this interaction level will receive a contribution from their ancestor node at the interaction level.

An alternative method was presented in (Languénou et al 92). This method is applicable in a progressive radiosity algorithm that uses undirected shooting. Polygons are subdivided dynamically. At each iteration, rays, carrying some amount of power, are sent from the source patch into the environment. The intersection points with the polygons in the environment are stored with the intersected polygon. The first intersection point of a ray is marked as a lit point. This point is illuminated by the sources. Rays are traced further. All further intersection points are marked as shadow points. These points are in shadow of the source. When all intersections of the rays with the scene are computed, for each patch the subdivision (or interaction level) that was necessary for this source is reconstructed from the intersection points. The subdivision is chosen to be so fine that at most a user specified number of rays intersects each subpatch. This user specified threshold value determines the accuracy of the solution. With this threshold value, for each number of intersection points on a polygon an optimum reconstruction is found. The reconstruction results in an interaction tree. This tree contains the subdivision that was necessary at the current iteration. After creation of the interaction tree, this tree is combined with the subdivision tree that stores the subdivision until the maximum subdivision level that is reached during previous iterations. Because this method generates lit points and shadow points, it is possible to detect shadow boundaries on surfaces. Along these shadow boundaries, subdivision can be increased, although it is not clear how this should be done exactly. This method has some small disadvantages. Rays are traced further than the first intersection point. So more intersection computations are necessary than in 'normal' undirected shooting. Another problem might be that the number of intersection points to be stored at the polygons may become large, when the number of rays that is traced from a source is very large.

All methods described so far ignore the fact that a single surface may be so small that it is wasteful to treat it as a shooting or receiving patch. The form factor with such a surface may be much smaller than the form factor estimate threshold used in (Hanrahan et al 91). The number of intersections with such a surface may be very small or zero, because the resolution of rays is not sufficient (Languénou et al 92). In these cases it is useful to take several small neighbouring surfaces together, and treat them as one item during the radiosity computations. This grouping or clustering process (Kok 93; Rushmeier et al 93) will be described in chapter 7.

Another solution to reduce the number of form factor computations is to divide the scene into subscenes (Xu et al 89; Liere 91). Virtual walls are placed in between these subscenes. These virtual walls can be subdivided into several patches and act as power transmitters between the subscenes. The method starts with computing local radiosity solutions for each of the subscenes. Power is also computed for the virtual wall patches. When a local solution is known for all the subscenes, power is exchanged between the subscenes by exchanging the energies on both sides of the virtual walls. After the power exchange between the subscenes, new local solutions can be computed. After that, power can again be exchanged through the virtual walls and the process can be repeated, until the global solution converges.

6.3 A hierarchy for the radiosity method

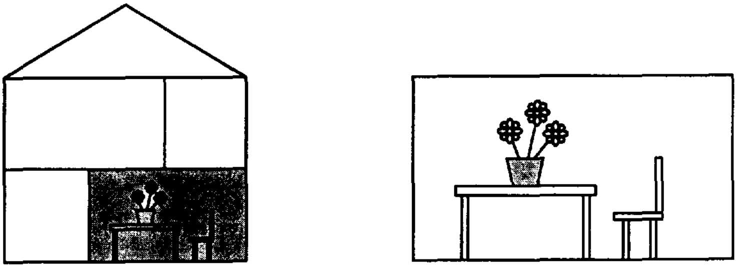
A method that combines some of the previously described methods may give an accurate, yet efficient, radiosity solution. The following hierarchy seems to be useful (figure 6.1). On top of the hierarchy is the scene. The scene consists of several subscenes. Often a scene consists already of logical subscenes, for example rooms in a house, but it is also possible to make the separation into subscenes by placing virtual walls in the scene. A subscene contains a number of objects or groups. An object or group may again consist of a number of smaller objects that are grouped together. For example, a plant may consist of leafs, stems and flowers. Several levels of groups (a hierarchy of groups) should be available. The groups are subdivided into basic surfaces (polygons, curved surfaces, etc.). Surfaces are subdivided into patches because these surfaces may sometimes be too large to be handled properly in a radiosity algorithm. As with groups, a hierarchy of patches is possible. An alternative to a hierarchy of patches is one level of patches and a hierarchy of elements. Elements are small parts of a patch. Elements are only used as receiving entities, not as source. Shooting is done from the patch level.

Several parts of the proposed hierarchy match with adaptive and hierarchical methods already used in radiosity algorithms, in particular the subdivision of surfaces into patches and elements (Cohen et al 86), a hierarchy of patches (Hanrahan et al 91; Languénou et al 92), the subdivision of the scenes into subscenes (Xu et al 89; Liere 91), and grouping (Kok 93; Rushmeier et al 93).

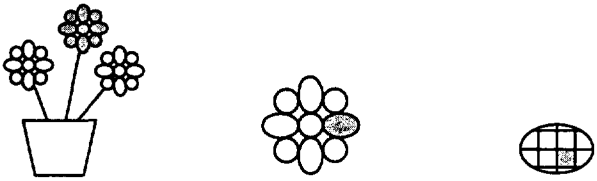
Concluding we can say that several existing methods cover parts of the proposed hierarchy. However, these methods are still not integrated into one hierarchical method, and these methods can not be applied fully automatically at this moment.

6.4 Implementation of adaptive receiving levels

A hierarchical meshing (Hanrahan 91) scheme is an important improvement to the radiosity process. Therefore a hierarchical meshing scheme has been implemented in our progressive radiosity method. This scheme is a combination of the methods given by (Hanrahan et al 91) and (Languénou et al 92). It can be applied with both directed and undirected shooting. The element level at which a surface receives power is determined automatically based on the estimation of interaction between source and receiver. Unlike the method of (Hanrahan et al 91), the interaction level of the source is not determined automatically.



scene->subscene subscene->group/object



group->(sub)group (sub)group->patch patch->patches/elements

Figure 6.1 An example of a hierarchy

The element subdivision is stored in a quadtree. Radiances are computed for the vertices of the elements and stored at these vertices. Two levels can be distinguished with respect to the quadtree: the maximum subdivision level and the interaction level.

When a patch is selected to be the source patch, for each other patch that will receive power from the source it is determined at which element level interaction with this patch should take place.

An estimation of the unoccluded form factor (ignoring occlusions by other objects) is used to compute the element interaction level, as in (Hanrahan 91):

$$f_{est} = \max\left(0, \frac{\cos\theta}{\pi}\omega'\right) = \max\left(0, \frac{\cos\theta\cos\theta'}{\pi r^2}A'\right) \tag{6.1}$$

where θ is the angle between the source normal and the direction to the receiver, ω' is the solid angle of the receiver seen from the source, θ' is the angle between the normal of the receiver and the direction to the source, r is the distance from source to receiver, and A' is the area of the receiver. If f_{est} is equal to zero, then the patch and source are not mutual visible, so there will be no power exchange between them.

The way the interaction level is determined depends on the method that is used in progressive radiosity. Undirected shooting requires a different approach from directed shooting.

In undirected or Monte-Carlo shooting rays are sent out in a cosine distribution. The number of rays in a direction is proportional to the form factor. Each ray represents the same form factor and therefore carries the same amount of power. In contrast to the method of (Languénou et al 92) in which the interaction level is determined after shooting the hemisphere rays, in our method the interaction level of all surfaces is determined before the rays are shot from the source. The expected number of intersections of the rays of the hemisphere and the surface determines the interaction level. Given the number of rays that is shot from the source R_s, and the form factor estimate f_{est}, the number of rays that probably will hit the receiving patch when there are no occlusions, R_r, can be estimated by:

$$R_r = f_{est} R_s \tag{6.2}$$

Because of the quadtree structure level l contains 4^l elements. The user can specify how many rays at least must hit each element to get accurate results. With this number R_h, the element level at which interaction must take place can be computed assuming that the distribution of hits on the receiver is almost uniform. The following relation is true:

$$4^l \leq \frac{R_r}{R_h} \tag{6.3}$$

Therefore the maximum allowed interaction level l is given by the following equation:

$$l = \left\lfloor {}^4\log \frac{R_r}{R_h} \right\rfloor = \left\lfloor {}^4\log \frac{f_{est} R_s}{R_h} \right\rfloor \tag{6.4}$$

For directed shooting the interaction level for a receiving patch can be determined in a comparable way. The receiving level is derived based on a user specified form factor accuracy f_{acc}. If the form factor estimate is much larger than this value then the results will not be accurate enough. If the form factor estimate is much smaller then too many computations will be performed. The form factors are approximately equal to f_{acc} if the interaction level l is computed with the following equation:

$$l = \left\lfloor {}^4\log \frac{f_{est}}{f_{acc}} \right\rfloor \tag{6.5}$$

If the resulting level l is negative, the patch is strictly speaking too small. It should not be subdivided, but grouped with other patches (see chapter 7).

An alternative for using the form factor estimate to determine the interaction level of the receiver is to use the solid angle estimate. Sometimes the solid angle estimate will give better results. When, for example, rays are sent out uniformly over the hemisphere (carrying different amounts of power), the solid angle gives a better estimate of how many rays will hit a patch.

Now the interaction levels of the receiving patches are known, the source will send its unshot power into the scene. For each vertex of the elements at the chosen interaction level the radiance contribution is computed. After all radiances for a receiving patch are computed, the radiance contributions for vertices at interaction levels larger than l can be computed by interpolation.

6.5 Results and conclusions

Results

Figure 2.16 shows what happens when in undirected shooting the resolution of the rays does not agree with the meshing. Aliasing occurs.

Adaptive receiving levels should reduce the aliasing. For model B, the small room, several radiosity runs (250 progressive radiosity iterations) have been performed, varying the number of rays per shooting patch R_s, and varying the number of rays that should hit a patch R_h (that determines the interaction level). Results are given in figure 6.2. Results are compared with a radiosity process in which 200000 rays are traced from each shooting patch. The results show that the error decreases when the number of rays per shooting patch is increased. Increasing the number of rays that should hit a patch will result in coarser interaction levels, and therefore larger errors. However, when the images are compared visually, images made with larger numbers of rays per element are visually more attractive. The shadows are somewhat worse, because they are larger than expected, but the annoying aliasing disappears.

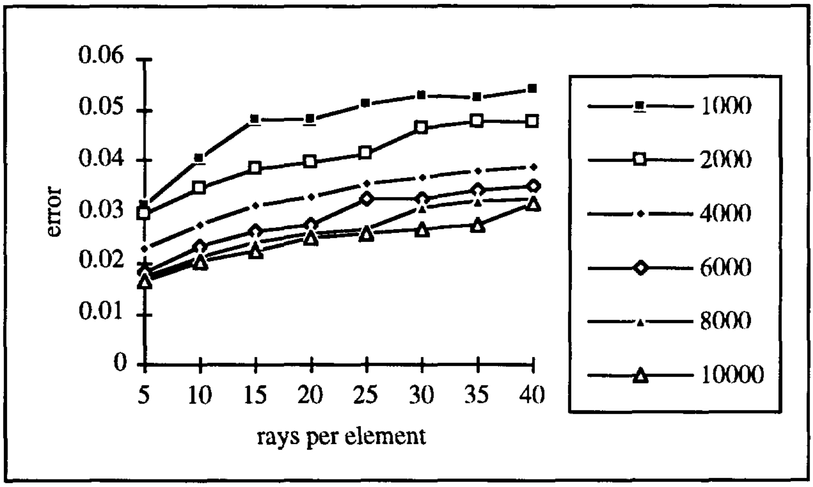


Figure 6.2 Results adaptive receiving levels for undirected shooting for several values of the number of rays that should hit an element R_h and the number of rays per shooting patch R_s

Several runs are also performed for directed shooting with varying value for the form factor accuracy f_{acc}. Results are shown in figure 6.3. Decreasing this value results in finer interaction between patches, and therefore in more rays and smaller errors.

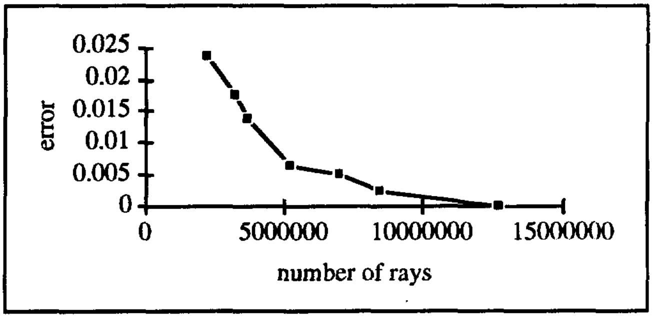


Figure 6.3 Results adaptive receiving levels for directed shooting for several values of f_{acc}

Conclusions

A radiosity method that is accurate and efficient should use a multi-level hierarchy. Up to now, no method is presented that uses a complete hierarchy. For some parts of the hierarchy methods exist, although not all methods do subdivisions or groupings automatically.

The hierarchical meshing that was presented by (Hanrahan et al 91) provides a good solution very efficiently. We combined parts of this method with the method presented in (Languénou et al 92) into a method in which the discretization level, at which a surface receives its power, is determined automatically, based on the expected exchange of power during an iteration of progressive refinement.

Using hierarchical methods during the radiosity process has the following advantages:

- The user does not have to specify exactly how fine patches should be subdivided into elements. The algorithm will decide what is the best interaction level.
- The interaction level is computed for each iteration independently. As a consequence, subdivision is coarse when possible. Many computations can be saved.
- Interaction is guided by a user specified threshold. With this value the user can influence the time the computations will take. Given the effort the user wants to spend, the best possible radiosity solution is computed. For example, when the user knows that the results of the radiosity pass are used in a Monte-Carlo sampling based rendering pass, the user can choose the threshold value in such a way that subdivision is coarse. If, however, the user knows that the results of the radiosity pass are interpolated to compute the complete shading, then the user can choose the threshold in such a way that subdivision is fine (and consequently execution time is large).

Our implementation has the following advantages:

- It is applicable for both directed and undirected shooting.
- The adaptive receiving levels method does not need the extra memory that is necessary to store the intersection points of the rays with the polygons, as in (Languénou et al 92). Also it is not necessary to trace rays beyond the first intersection point.

The disadvantages of the given method are:

- The method only provides a method to compute the interaction level of the illumination receiving patch. No interaction level is determined for the shooting patch, as is done in (Hanrahan et al 91).
- While determining the interaction level, it is not known whether shadow boundaries will occur. Therefore it is not possible to choose a higher interaction level for surfaces that contain shadow boundaries. However, because in our case the radiosity pass is part of a hybrid method in which important shadow boundaries are computed again and more accurately during the rendering, this will not strongly affect the image quality.

Next chapter will describe another part of the hierarchy: grouping.

7 Grouping of patches

7.1 Introduction

To make realistic images, scenes must be modelled accurately. Most objects are very detailed. Modelling them requires many, often small, surfaces. Examples of complex objects with many details are plants, keyboards, etc. It is the accurate modelling of such detailed objects that makes scenes look realistic.

Radiosity programs usually have problems handling very small surfaces or patches. Most programs ignore the fact that a single surface can be too small to be treated as a patch. Artefacts like aliasing may occur or else very expensive computations have to be performed.

The radiosity method can be improved by (adaptively) grouping small neighbouring patches into groups. Computations normally done for the individual patches are now applied to these groups. Groups receive power from the environment, and can shoot power into the environment. The radiances of the patches in the group are derived from the power the group receives. Grouping small patches reduces the number of form factor computations, reduces aliasing effects, and improves the convergence in progressive radiosity.

This chapter presents methods to use grouping of surfaces in ray tracing based progressive radiosity algorithms. Section 7.2 shows the need for grouping. Section 7.3 describes how to group patches and section 7.4 describes transfer functions. These functions transfer the power that a group receives to radiance values for the individual patches within the group and vice versa. Results and conclusions are presented in section 7.5.

7.2 Motivation for grouping

In chapter 6, methods are given to determine the interaction level of a patch. The result of applying formula 6.4 or formula 6.5 to a patch may be a negative interaction level l. If this interaction level l is negative, then the patch was too small to be treated as a single patch during the radiosity pass. It is possible to ignore such a negative level and to set the level in this case to zero. Then the patch is not subdivided into smaller elements. However, the progressive radiosity algorithm will not perform optimally. For example, consider the scene of figure 7.1. A light source illuminates a computer from a reasonable distance. The solid angle of the computer seen from the source is small. The computer consists of 441 quadrilateral patches, mainly constituting the keys of the keyboard.

If the undirected shooting method is used, an enormous amount of rays must be traced from the light source to avoid aliasing effects caused by inappropriate ray density over the keys of the keyboard (figure 7.1, right). The distribution of the ray hits on the patches is not correct (because of the random effect of undirected shooting). Some of the patches are hit with zero rays, others with 3 rays although they should be illuminated approximately the same. A solution is to increase the number of rays that originate at the source in such a way that all patches are expected to receive a comparable amount of ray hits. However, this may require an increase in the number of rays of several orders. This solution is therefore not feasible.

If the directed shooting method is used, and radiances are calculated for the vertices of the patches, then 1348 rays (= number of vertices) must be traced to calculate the form factors for all sample points that are facing the light source, although the mutual visibility for all these sample points will be the same. This is an enormous waste of effort.

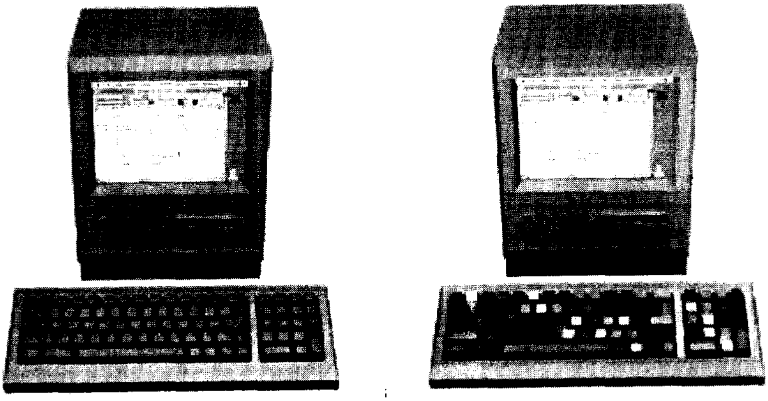


Figure 7.1 Object with many small patches in directed shooting (left). Aliasing on the keyboard due to insufficient sampling density in undirected shooting (right)

Another problem with small patches is that they are almost never selected to be source patch, because their unshot power is always less than the unshot power of larger patches in the scene. However, the total unshot power of a number of neighbouring small patches can be quite considerable. It is therefore useful to consider patches, that are close to each other, as one shooting group, so that they are selected together as a source during the progressive radiosity.

7.3 Grouping patches

The previous discussion showed the need of a method that handles small patches. One way to solve this problem is grouping or clustering. Grouping can be defined as follows:

> Grouping is the process of regarding a number of patches that are close to each other as one 'macro-patch' during the radiosity process. This 'macro-patch' is called group.

A grouping method consists of two parts:

- A method to cluster small patches into meaningful groups. This clustering method identifies the patches that can be taken together.
- A transfer function for the illumination computation of the group with the world outside the group. This part consists of the computation of the radiances of the group caused by illumination from other parts of the scene, and computation of the radiances of other parts of the scene caused by illumination from the group.

First the decision must be made which patches should be grouped and which patches can stay separately. Preferably, this decision is made at each iteration of the progressive radiosity algorithm. Given the relative positions and sizes (the solid angles or the form factors) of patches with regard to the source the interaction level of the patches can be determined. Patches with negative interaction levels that are situated close to each other regarding the distance to the source are grouped. However, it can be rather expensive to do all these computations at each iteration. An alternative is to do the grouping in a preprocess before the radiosity process starts (for example during the modelling). The scene for which the illumination must be calculated can be scanned for clusters of small patches that are close to each other, where 'small' and 'close to each other' are defined relatively to the size of the total scene. Small patches in such a cluster are grouped. Now the determination of groups is performed only once, but is not well adapted to each individual processing step. A better solution would be to make a hierarchy of groups in a preprocess. At which level in the hierarchy of groups the calculations should be performed can then be determined for each iteration separately. This decision can be taken on basis of the solid angles of the groups as seen from the source.

The easiest way of grouping patches is to replace a group of patches by its bounding box. The planes of the bounding box act as patches to the outside world. This method shows some resemblance with the local environment or virtual wall techniques (Xu et al 89). During the shooting process, the patches within the group are ignored. The plane patches of the box receive power from the environment and send out their power into the environment if necessary. The reflectance of the bounding box planes, that is necessary to be able to interact with the environment, can be computed by averaging the reflectance of the patches in the group projected onto the planes. At the end of the progressive radiosity process, the bounding box plane radiances are converted to patch radiances for the patches within the group. This method can be implemented easily in a radiosity algorithm.

A number of disadvantages, however, makes this method less attractive. The accuracy of the method is affected by the fact that the radiances are stored for a limited number of directions (corresponding with the bounding box planes) although the number of orientations within the group may be much larger. So when reconstructing the patch radiances from the radiances of the plane patches of the group, large errors may occur. Also, when other patches, that do not belong to the group, are partially within the bounding box of the group, the radiances for these patches may be incorrect, because sample points may be hidden by the group planes, although not hidden by the group patches themselves. Another disadvantage is that the bounding box exists in the scene. When the visibility between two not grouped patches is determined, and a group is situated between these patches, the shadow will have the shape of the bounding box instead of the group, if no special actions are performed when a group bounding box patch is hit by a ray.

An improvement to this method is given in (Rushmeier et al 93). As before, small isolated patches are clustered and replaced with a box. The faces of the box around the cluster are subdivided into patches. The reflectance of such a patch is determined by sending a number of rays perpendicular to the patch through the box. The percentage of rays that do not hit any surface before leaving the box gives the transmittance of the patch. The box is shrunken in size according to this percentage. The average reflectance of patches that are hit within the box is used as the reflectance of the patch. The result of the clustering process is that there are now two geometries for the model: the original geometry, and the simplified geometry in which small groups of small patches are replaced with boxes. The radiosity method is now applied to the simplified geometry. When the radiosity pass is completed, the rendering starts. For viewing rays the original geometry is used. Direct illumination for each visible point is computed by sending shadow rays to the light sources through the original geometry. Indirect illumination is computed by Monte-Carlo path tracing. The original geometry is now used when intersections are found close to the origin of a ray and the simplified geometry is used when the intersection point is farther away.

Experiments show a good efficiency improvement without degradation of image quality for this method. However, the method as described in (Rushmeier et al 93), combines well with a one-level Monte-Carlo path tracing rendering algorithm in which indirect illumination is computed for each visible point in the scene. Other hybrid methods require radiances for each of the patches in the scene, also for the clustered patches. To calculate the radiances of the clustered patches we need a transfer function that transfers the box-plane patch radiances into radiances for the individual patches that were clustered within the group.

7.4 Transfer functions

The grouping method, presented in this chapter, does not use the approximation with a box enclosure. Patches in a group are not replaced by some larger patches, but are only marked as belonging to a specific group. Illumination computed for the group is directly converted into radiance values for the patches in the group.

A group receives power from the environment and sends power into the environment. When a group receives power from a source patch, this power must be distributed in some way over the individual patches in the group, taking into account the orientation of each patch. When a group is selected to shoot its unshot power, because the unshot power of the group is larger than the unshot power of all patches and other groups in the scene, the unshot power of the different patches in the group must be combined to form the unshot power of the group.

To simplify the transformation from group illumination to the illumination for the individual patches in the group, a few assumptions are made:

- Patches are only clustered in a group when the distance of the group to the source patch is large compared to the size of the group; otherwise it is better to use the individual patches. This means that the distance and the direction of the patches to the source patch can be assumed to be constant.
- A group has a constant visibility, so all patches in the group have the same occlusion. This assumption has the consequence that shadow boundaries are not visible on a group.
- There is no self occlusion, so patches within a group will not generate shadows on other patches within the group.

The choice of shooting method (directed or undirected) determines the way grouping is used. Therefore different transfer functions are presented for undirected and directed shooting.

7.4.1 Transfer functions for undirected shooting

In undirected shooting (Malley 88) a number of rays is traced from a source patch into the environment. A distribution function (e.g. cosine) determines the directions of the rays. Each ray leaving the source represents some power $\Delta\Phi_r$. Normally, when a ray hits a patch, this patch receives all the power, and the radiance of this patch is updated. The larger the solid angle of the patch with respect to the source point, the more rays will hit the patch, and so the more power the patch will receive.

A group as receiver of illumination

When a ray hits a patch of a group, the power $\Delta\Phi_r$ is not assigned only to the patch that is hit by the rays, but this power is distributed over all patches in the group. A weight w_p is calculated for each patch p in the group. This weight indicates which fraction of the total power is assigned to the patch. The power $\Delta\Phi_p$ that a patch p in the group receives is given by:

$$\Delta\Phi_p = w_p \Delta\Phi_r \qquad (7.1)$$

The weight should be proportional to the power the patch should receive. An obvious choice is therefore the estimation of the form factor of formula 6.1. The weight of a patch p can then be expressed as the form factor estimation of patch p, $f_{est,p}$, divided by the form factor estimation of the group $f_{est,g}$:

$$w_p = \frac{f_{est,p}}{f_{est,g}} \tag{7.2}$$

At first sight it seems that it is expensive to compute the weights. However, the angles between the normal at the source and the directions to the different patches of the group can be considered constant because of the assumption that the group is far away from the source. The weight can then be expressed as a fraction of solid angles:

$$w_p = \frac{\max(0, \omega_p)}{\omega_g} \tag{7.3}$$

where ω_p is the solid angle formed by source point and patch p, and ω_g is the solid angle formed by source point and group (figure 7.2). Patches not facing the source do not receive any power contribution.

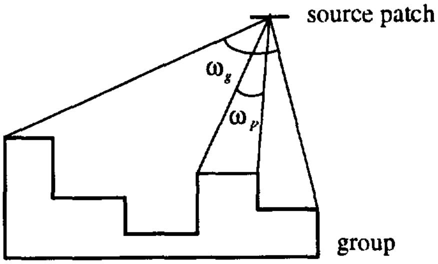


Figure 7.2 Solid angles of group

The solid angle ω_p from a source point and a patch p is given by:

$$\omega_p = \frac{A_p \cos\theta_p}{r^2} = \frac{A_p(\mathbf{d}_g \cdot \mathbf{n}_p)}{r^2} \tag{7.4}$$

where θ_p is the angle between patch normal (= $\mathbf{n}_p$) and direction to source (= $\mathbf{d}_g$), A_p is the area of the patch, and r is the distance between the source point and the patch. The direction to the source is equal to the negative ray direction.

The total solid angle for the group can be calculated in two ways. The first possible solution is to calculate the 'exact' geometrical solid angle for the entire group. The second solution is to calculate the sum of the solid angles of all patches in the group.

When the geometrical solid angle is used, the illumination of the illuminated patches is calculated correctly. However, two facts make the use of the geometrical solid angle less attractive. Firstly, it is very expensive to exactly compute the exact geometrical solid angle. It is also possible to use the solid angle of the bounding volume around the group, but this is not very accurate. Secondly, the patches that should not be illuminated because of occlusion, caused by patches within the group, also receive a power contribution, because self-occlusion within a group is not taken into account. Too much power will be calculated because of overlapping solid angles (figure 7.3). Therefore the sum of the power received by all patches in the group will be larger than the power of the ray $\Delta\Phi_r$. When later on the group is selected to shoot its power, too much power is shot into the environment.

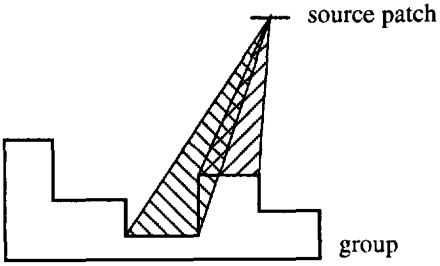


Figure 7.3 Overlapping solid angles

When the sum of the solid angles of all patches in the group is used, the total power that is assigned to the group is correct. It is taken into account that some parts of the group are not visible because of overlapping solid angles, but it is not known which parts. However, the total power will be spread over all patches, so the power of the really illuminated patches will be lower then expected, resulting in darker patches when rendering these patches. On the hand, patches that should really not be illuminated because of obstruction within the group get also a part of the illumination.

Both calculation methods for the solid angle of the group are not optimal. The following solution is therefore proposed instead. The geometrical solid angle is used for calculation of the radiances (that are used for display), so the shading for the really illuminated patches is correct. The sum of solid angles is used for calculation of the unshot power of the patches in the group, so that the power that is shot from the group in later iterations is correct.

Lets return to the formulas to compute the weight of a patch within a group. Using the assumption of the constant distance the weight is given by:

$$w_p = \frac{\max(0, A_p(\mathbf{d}_g \cdot \mathbf{n}_p))}{w_g} \tag{7.5}$$

where w_g is the total weight of the group. This value can be based on the exact geometrical solid angle of the group:

$$w_g = r\,\omega_g \tag{7.6}$$

or can be based on the sum of the solid angles within the group. In this case w_g can be computed with:

$$w_g = \sum_p \max\left(0, A_p\left(\mathbf{d}_g \cdot \mathbf{n}_p\right)\right) \tag{7.7}$$

A group as source of illumination

A group can also be selected to be the source during one of the iterations of the progressive radiosity process. A method is then necessary to compute the contribution from the group to the rest of the scene.

From the selected group as many rays are shot as normally is done when a patch is selected to shoot its unshot power. This number of rays can, for example, be dependent on the amount of unshot power that will be shot. A decision now has to be made from which patches in the group the rays will be shot into the environment. A proper choice is to use a method in which the number of rays that originate from a patch is proportional to the unshot power of this patch. The larger the unshot power of the patch, the more rays originate from it. The number of rays shot from a patch in the group R_p is therefore:

$$R_p = \frac{\Phi_p}{\Phi_g} R \tag{7.8}$$

where R is the number of rays normally shot from a non-grouped patch, Φ_p is the unshot power of the patch in group, and Φ_g is the total unshot power of the group:

$$\Phi_g = \sum_p \Phi_p \tag{7.9}$$

With this method, most rays are shot from the most radiant parts of the group. It allows diffuse interreflection within a group.

7.4.2 Transfer functions for directed shooting

Directed shooting (Wallace et al 89) from a source starts with subdividing the source patch into delta areas. The form factor of each delta area of the source with respect to each of the receiving vertices of the patches in the environment is computed. The visibility is tested by casting a ray from the center of the delta area to a vertex.

A group as receiver of illumination

When the contribution of a source to a group must be calculated, the following method is applied. For each point (center of delta area) on the source patch one or a few random points are determined on the group. Points are preferably only chosen on patches that face the source. Rays are traced to determine the visibility between the source point and these points on the group (intersections of the ray with patches of the group can be ignored). The visibility of the group from a source point is given by the fraction of rays that reach the patches of the group (figure 7.4). The radiances of each vertex in the group can now be calculated using the form factor from the source patch to the vertex given the estimated visibility of the group. Note that the radiances of the individual patches are now calculated correctly, because they are not based on $\Delta\Phi_r$ contributions as in undirected shooting, but on an exact form factor calculation. Only the number of visibility tests is reduced. Instead of casting a ray from each receiving point to each delta area on the source, now rays are cast from only one or a few points on the group to each delta area on the source.

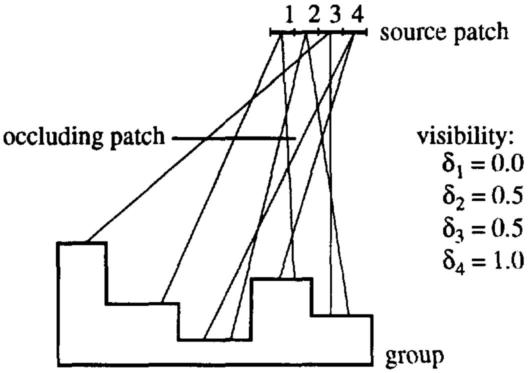


Figure 7.4 Visibility determination between source points and group

A group as source of illumination

When a group is selected to shoot its unshot power in directed shooting, a number of source points must be determined on the group, just as is usually done for a source patch. Points are chosen on the boundary of the group. Each of these points has a direction pointing out of the group. The directions must have a good distribution, so that all directions that exist in the group can be represented with the chosen directions.

First, the unshot power of the source points is calculated from the unshot power of the patches of the group. The unshot power $\Delta\Phi_{ip}$ of source point i, due to a patch p, is given by:

$$\Delta\Phi_{ip} = \frac{\max(0,\cos\theta_{ip})}{\sum_{j=1}^{n}\max(0,\cos\theta_{jp})}\Delta\Phi_p = \frac{\max(0,(\mathbf{n}_i\cdot\mathbf{n}_p))}{\sum_{j=1}^{n}\max(0,(\mathbf{n}_j\cdot\mathbf{n}_p))}\Delta\Phi_p \qquad (7.10)$$

where θ_{ip} is the angle between shooting direction ($\mathbf{n}_i$) and patch normal ($\mathbf{n}_p$) (figure 7.5), $\Delta\Phi_p$ is the unshot power of patch p, and n is the number of chosen source points.

Instead of many source patches representing the group, now only n shooting points exist. Each source point sends its unshot power into the environment as a regular source patch.

A drawback of this method is that no interreflection between the patches within a group is calculated, because power is shot from the boundaries of the group, so no power ends up within the group. This can be solved by performing a normal directed shooting for patches within the group, where each patch in the group is successively selected to shoot its unshot power, but only to the other patches in the group.

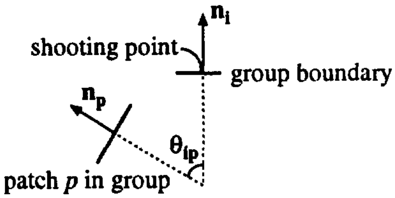


Figure 7.5 Calculation of patch contribution to source point power

7.5 Results and conclusions

Results

The described grouping methods are implemented in a progressive radiosity program. The grouping of surfaces is done manually during the modelling process. However, it is possible to do the grouping adaptively. When a patch in a group is large enough with respect to the position of the source, this patch will be treated as a separate patch and temporarily not be included in the group.

Consider the scene of figure 7.1, showing a computer with keyboard. Grouping methods were tested both for undirected and undirected shooting. The results of directed shooting are shown in figure 7.6. The left picture is made without grouping. The number of rays to be traced to determine the form factors for the computer is more than 1348 (= the number of vertices on the computer that face the shooting patch). For the right picture, the patches of the monitor (except the screen patch) and the patches of the keyboard were grouped into two groups. For these groups the number of visibility tests is chosen by the user. For the tests only one visibility test was used. This did not affect the quality of the image, except for small shadow details caused by occlusion within a group, e.g. near the lower right corner of the screen.

Figure 7.7 and 7.8 show the results for undirected shooting. Although the number of rays originating from the light source that hit the keyboard is only 691 (for figure 7.7) and 7464 (for figure 7.8) no aliasing effects are visible when grouping is used (right pictures). Even for a smaller number of rays the shading of the keyboard will be smooth, because the power of a ray that hits the group is distributed over all patches in the group, instead of only over the patch that is hit. When the patches are grouped, the probability that the group is hit, and so that all patches receive an 'equal' contribution, increases considerably.

Model C contains several computers that are modelled accurately. For this scene grouping gives an enormous reduction of computations. Each computer consists of three groups: the computer itself, the keyboard, and the mouse. 8000 progressive radiosity iterations are done. Results for directed shooting are given in table 7.1. The reduction of rays to compute the solution with grouping is 53%. The execution time reduction is 50%. We can conclude that the overhead of the grouping (computation of transfer functions) is rather small.

	time	number of rays
without grouping	26:28:00	212840688
with grouping	13:16:56	99492269

Table 7.1 Results grouping for fixed number of iterations

Although the number of iterations in table 7.1 is the same for both test runs, the convergence of these two runs differs. With grouping the process converges much more, because energy is not shot from separate patches, but from groups. To obtain the same convergence, radiosity without grouping needs more iterations than radiosity with grouping. Table 7.2 gives the results when the radiosity process is stopped when the same convergence is reached (the same amount of unshot power remains left over in the scene). Apart from the fact that the number of form factor computations is reduced because of the grouping, now also the number of iterations is reduced with 17%, resulting in a total execution time reduction of almost 58%.

	iterations	time	number of rays
without grouping	8000	26:28:00	212840688
with grouping	6639	10:59:41	83194449

Table 7.2 Results grouping for same convergence

Conclusions

Grouping of small patches is a useful extension for the progressive radiosity method. It reduces the number of rays for ray tracing based form factor computation. The use of groups in an undirected shooting method reduces the aliasing caused by an inappropriate resolution of the shooting rays. The number of shooting rays can be reduced.

Grouping fits well in an adaptive hierarchical radiosity process. When surfaces are too small to discretize, and even too small to act as individual patches, for instance because they are too far away from the source patch and will not be hit by enough rays, they can be combined into a group.

Using a group as a source for an iteration in the progressive radiosity algorithm makes the radiosity process converge faster.

The method presented here has the major advantage compared to the method described in (Rushmeier et al 93) that it is applicable for all radiosity methods. It is not limited to a two-pass method that computes the indirect interreflection during the rendering pass. Because transfer functions are available, it is possible to compute the radiances for all patches in the group. These radiances can be used directly for display.

However, a problem is that shadows generated by obstruction of other patches as well as from patches within the group are not taken into account because a constant visibility for the whole group is assumed. Areas that should be in shadow are illuminated. The total power that the group receives and reflects, however, is correct. Thus, although for display the shading of the group is not completely correct, the error for the indirect illumination of other patches is negligible. When the direct illumination is computed accurately during the rendering with ray casting with the method described in chapter 3, the fact that occlusions within a group are ignored does not reduce the image quality. These occlusions, when caused by important sources, are then computed during the rendering.

Another problem is that clustering is now done manually. A method is needed to automatically cluster patches into groups. It would be worthwhile to investigate whether the spatial subdivision that is available in almost each ray tracing program can be used to identify patches that are situated in each other's neighbourhood.

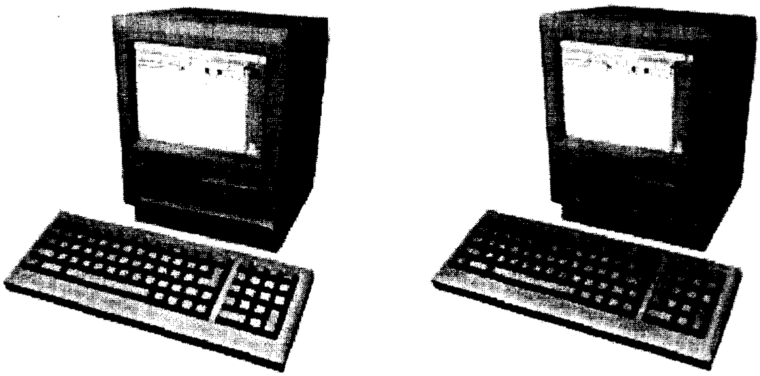


Figure 7.6 Directed shooting; without grouping (left), with grouping (right)

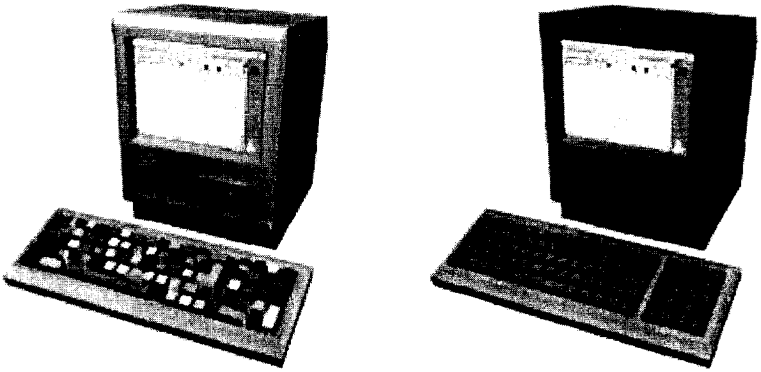


Figure 7.7 Undirected shooting, 10^5 rays per shooting point; without grouping (left), with grouping (right)

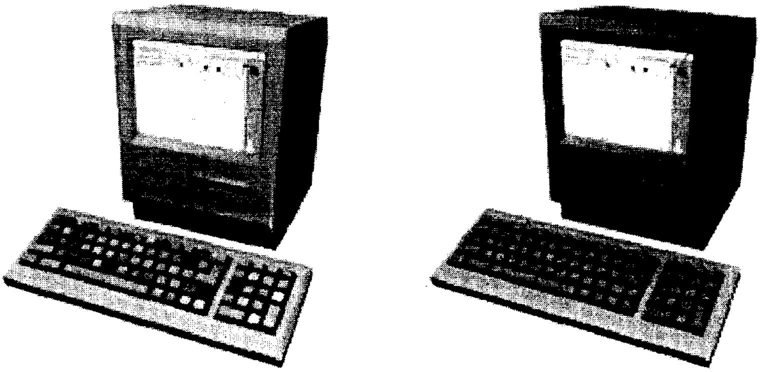


Figure 7.8 Undirected shooting, 10^6 rays per shooting point; without grouping (left), with grouping (right)

8 Conclusions and future research

8.1 Introduction

The research described in this thesis was aimed at the development of algorithms for realistic image synthesis, to provide means to generate images that look real and are physically correct, and that are capable of solving the complete global illumination of a scene, and preferably as efficient as possible.

In chapters 2 and 3 several radiosity, ray tracing, and hybrid radiosity and ray tracing methods were reviewed. Based on the observations made there, a new hybrid method was proposed. This new hybrid method consists of two computational expensive processes: radiosity and ray tracing. Several methods were developed to increase the efficiency of both processes. Ray tracing acceleration methods were given in chapters 4 and 5. Acceleration techniques for the radiosity process were described in chapters 6 and 7.

This chapter summarizes the conclusions from previous chapters, combines the results of the different methods, and presents some directions for further research. Section 8.2 brings together all results and conclusions. The presented methods are evaluated on quality and efficiency. Section 8.3 gives some ideas for further research. Primarily, global illumination methods can be improved by better algorithms. However, the applicability of the methods can also be improved by speeding up the computations with multiprocessor systems or specialized hardware.

8.2 A global illumination system for realistic image synthesis

To generate realistic images it is necessary to compute the global illumination of the scene to be displayed. All possible paths that light can travel before it reaches the eye of the viewer must be taken into account. In chapter 2 methods for realistic image synthesis were compared on their ability to simulate all light paths, the accuracy of the shading they compute, and their efficiency. Ray tracing methods that include the computation of the diffuse interreflection may generate correct images, but the time to generate them will be very high. Diffuse interreflections are preferably computed during a separate pass in a hybrid method. A standard two-pass method can compute all light paths, but the shading accuracy is very much dependent on the meshing that is applied during the first (radiosity) pass. Adequate results can only be guaranteed if exact meshing schemes are used, that try to find the shading discontinuities before the radiosity process is performed. However, this is a very complicated task. The hybrid method that separates the direct from the indirect illumination makes the results less dependent of the meshing resolution. Important perceptible shading gradients are often caused by direct illumination. Indirect illumination is only responsible for minor, often not perceptible, shading gradients. Therefore, while the indirect illumination can be stored in the meshing, and can be computed during the radiosity pass, the direct illumination is computed accurately during the rendering pass by casting shadow rays to the light sources.

The method presented in chapter 3 is a generalization of the hybrid direct-indirect method. Instead of separating the direct from the indirect illumination, the method separates the high-frequency illumination from the low-frequency illumination. High frequency illumination is illumination that causes important shading gradients. Although high-frequency illumination often corresponds with direct illumination, this is not generally the case. Low-frequency illumination causes low shading gradients. Low-frequency illumination is computed with a radiosity method. The meshing can be rather coarse, so the execution time of this pass will be limited. High-frequency illumination is computed during the rendering by casting shadow rays to the important sources (light emitting and strong reflecting) patches. The source selection process is used to detect the high-frequency illumination. Source selection detects which patches generate perceptible shading gradients on a patch. By applying different selection criteria (using different threshold values), the new hybrid method can simulate all kinds of hybrid methods within the range from a standard two-pass method (select no sources) to a one-level Monte-Carlo method (select all patches as sources). Simple selection criteria already provide good results (savings of 60% of the shadow rays), while the application of these criteria takes almost no time. More complex criteria may lead to even better results.

The presented method tries to find an optimum between what should be computed during the radiosity pass and what should be computed during the rendering pass to get a solution, that is as accurately and efficiently as possible. However, both passes require still too much time. Therefore methods were investigated to further speed up both passes.

8.2.1 Improvement rendering pass

Although source selection reduces the number of shadow rays to be cast during the rendering process compared to the direct-indirect hybrid method, the number of shadow rays to be cast can still be large. Therefore during rendering, two methods were applied to reduce the number of shadow rays further: The shadow image buffer, as described in chapter 4, and adaptive source sampling, as described in chapter 5.

The shadow image buffer method reduces the number of shadow rays by exploiting shadow coherence. In contrast to other methods to cull shadow rays, for example the light buffer, it exploits shadow coherence in image space. During adaptive image refinement, shadow rays are only cast to a light source when a shadow boundary is expected. In other cases the shadow information of neighbouring samples indicates whether a point is in shadow or not. The shadow image buffer is very simple: it requires almost no overhead, almost no extra memory (only one or two bits per light source per viewing ray), and no extra preprocessing. The method is applicable for arbitrarily shaped objects, it is not restricted to point light sources, and gives an extra image refinement criterion (to anti-aliase shadow boundaries). Experiments showed that a good reduction of shadow rays can be obtained, especially when shadows are coherent and well shaped. However, the shadow image buffer is yet not well suited for scenes with many partly diffusely and partly specularly reflecting surfaces. An extension to solve this problem will require much more memory.

Large area light sources must be sampled with many shadow rays. An adaptive stochastic sampling method seems the best method to obtain reliable results as efficient as possible. In our implementation, rays are semi-randomly cast to the source. Rays are cast to areas on the source where a shadow transition is expected or where the confidence in the sampling up to now is small. The sampling is stopped if the standard deviation of the samples of the source sampling is small enough. Adaptive stochastic sampling can be improved with sampling pattern coherence. The shadow patterns of two consecutive samplings tend to be almost the same. Therefore a previous sampling pattern can be used for the next source sampling. It provides a good initial sampling pattern (a good initial estimation of where shadow transitions are), avoiding aliasing that can be caused when the initial sampling misses small shadow details.

The three methods to avoid casting shadow rays (source selection, the shadow image buffer, and adaptive source sampling using sampling pattern coherence) can be combined. Results for anti-aliased pictures are given in table 8.1. Using all efficiency methods reduces the number of shadow rays up to 94%. The culling methods are not independent. Therefore sometimes the reduction of the three methods together is not as good as what is expected from the test results when the methods were applied separately. The methods give the best reduction in the same circumstances. For example, when no shadows are cast on a patch, source selection will decide to cast no shadow rays, the shadow image buffer decides that no shadow rays are needed for refinement levels, and with adaptive source sampling a limited number of shadow rays will be cast.

Also the savings expressed in shadow rays are higher than the savings expressed in time. The reason is that casting shadow rays adaptively (which is not implemented optimally) takes more time per ray than casting a fixed number of rays, because sampling statistics have to be computed, and because for each ray a search is done to find the area that can be sampled best. The fact that this difference in savings is larger for model B is caused by the partly specularly partly diffusely reflecting teapot. Because this teapot is not purely diffusely reflecting, the shadow image buffer cannot be exploited optimally. Therefore in this region of the image more rays are cast. Because of the fact that intersection of a ray with a curved surface takes much more time than intersection with a polygon, much time is spent for casting rays to and from the teapot.

	B small room 1	B small room 2	C computer room
no culling methods			
shadow rays	35871743	37214548	140948622
time	03:53:14	07:11:46	14:34:57
with culling methods			
shadow rays	7014358	6701509	9079356
time	01:12:07	02:38:05	01:12:20
% improvement	80 (rays)	82 (rays)	94 (rays)
	69 (time)	64 (time)	92 (time)

Table 8.1 Overall efficiency rendering pass

8.2.2 Improvement radiosity pass

An efficient radiosity process needs some kind of hierarchical description of the scene. Interaction should take place on a level such that the number of form factor computations is small but sufficiently.

To improve the radiosity pass two methods were proposed: adaptive receiving levels, as described in chapter 6, and grouping of small patches, as described in chapter 7.

The adaptive receiving levels method is based on the methods of (Hanrahan et al 91; Languénou et al 92). During progressive refinement, the resolution of the meshing that is needed depends on which patch is chosen to be the source patch. So at each iteration for each patch the optimum level of interaction can be determined. This determination is done automatically based on the expected interaction between the patch and the source. Using this automatic interaction level determination has several advantages. Because it is done automatically, the user does not have to define the meshing so explicitly, making the radiosity program easier to use. In directed shooting it may save many form factor computations and therefore rays to be traced. In undirected shooting, aliasing due to the mismatch of the ray resolution and the meshing is eliminated. Therefore the ray resolution can be smaller, while acceptable results are still obtained.

The other method, grouping, was presented in chapter 7. Small patches are clustered to one entity, the group, and instead of interaction with the individual patches, now interaction is performed with the group. A group can be a receiver of power and a source of power. Transfer functions transfer group radiances to patch radiances and vice versa. Grouping small patches has several advantages. In undirected shooting it reduces aliasing effect, that can occur when the resolution of rays shot from the source is not sufficient. Therefore the number of rays that is shot from each source can be limited. In directed shooting, grouping has the advantage that it reduces the number of visibility tests for the form factor calculation and therefore the number of rays. Also the progressive radiosity process may converge faster, because the unshot power of several grouped patches (often patches that would not be chosen when handled separately because they are too small) is shot into the environment during one iteration. Experiments showed that grouping can reduce the execution time with more than 50%.

8.3 Ideas for future research

The hybrid method described in chapter 3 is able to generate accurate pictures. However, the time to generate images is still far too long. The methods of chapter 4 to 7 reduce the time to generate the images, but times are still measured in minutes, and often hours, even for models of moderate complexity. In the end it should be possible to generate images at interactive times or, preferably, in real-time. Moreover, the current state of the technique does not allow to compute all possible optical effects. The methods are still not general enough.

Improvement of functionality and efficiency can be obtained in two ways: by better algorithms, and by multiprocessor and specialized hardware implementations.

8.3.1 Improvement of algorithms

Algorithmic improvement can be done in two directions: extension of the functionality and accuracy of the methods, and improving the efficiency.

Standard radiosity algorithms are only suited for completely diffusely reflecting and transmitting patches. The use of extended form factors makes it possible to use specularity. However, only perfect specular, and sometimes rough specular, surfaces can be used. It is still not possible to use general reflection functions. The use of general reflection and emission functions will further improve the realism of the images.

As shown in chapter 3, the source selection takes only a very tiny part of the total processing time. The selection criteria that are applied now are rather simple. The use of more complex criteria may improve the selection. Better image quality might be obtained, and/or the efficiency might be improved.

As discussed in chapter 6, a hierarchy should be used during the radiosity pass. Although some parts of this hierarchy have been applied, a complete solution is not available yet. Research in the area of a complete multiple level algorithm is therefore one of the most promising directions for future developments of radiosity methods.

Grouping methods are now applied to groups that are determined during modelling (by the user). Clustering methods should be available to cluster patches into groups automatically.

The hierarchy that is used now in the radiosity pass, could also be applied during the rendering, when a hybrid method is used in which shadow rays are traced during this rendering (low/high-frequency method, one-level Monte-Carlo sampling). For example, in a one-level Monte-Carlo sampling normally small strong radiant patches can be missed easily by shadow rays. A group, that represents the radiance of all patches in it, will be hit more often. Therefore, when an illumination sampling ray is at a certain distance of the point for which the illumination is computed, and therefore the resolution is low, it will be better to sample the group as a whole instead of the individual patches. This may help to reduce the number of rays that should be cast to prevent aliasing. Groups may also be used during rendering in other circumstances.

Participating media have been applied to radiosity algorithms (Rushmeier and Torrance 87; Bhate and Tokuta 92). The effects of media can be computed by extending the meshing to a subdivision of the world in volumes, and apply the radiosity method to these volumes instead of to patches. However, a radiosity solution with participating media is very expensive to compute. It is worthwhile to investigate how this method can be combined with hierarchical methods.

8.3.2 Multiprocessor and specialized hardware implementations

Although efficiency improvement techniques are developed for radiosity and ray tracing, still the processing times on standard workstations are in the order of minutes or hours, and not in seconds. To obtain interactive or real-time response, processing times should go to seconds or fractions of seconds. To achieve this goal, multiprocessor implementations or specialized hardware implementations could be considered.

Several attempts have been made to parallelize radiosity and ray tracing processes. Both networks of workstations and general purpose parallel computers, such as transputer networks, have been applied. When transputer networks are used, large patch databases cannot be duplicated completely on all transputers. The database should be divided over the transputers. As a consequence, the transputers will have to communicate this data. This might become a bottleneck. Therefore it is important to find methods that reduce the communication that is necessary. Solutions to this problem are finding a good configuration and trying to explore coherence (Green and Paddon 89; Jansen and Chalmers 93). However, it is still not clear how coherence can be exploited optimally.

Another approach is to develop specialized hardware for the most computational expensive processes. Not all possible software implementations for realistic image synthesis can be implemented in hardware because of the complexity of most of these methods (Jansen et al 92). A hardware implementation, in the form of a plug in board, the 'radiosity engine', to enhance the performance of standard workstations for realistic image synthesis is now being developed (Shen and Deprettere 92; Shen 93). This 'radiosity engine' will be used to accelerate both the radiosity process and the rendering process by speeding up the ray intersection calculations. The basic primitive in the radiosity engine is the (undirected) shooting of rays in a frustum. This frustum is a part of the hemisphere during the progressive radiosity process, or represents a number of viewing rays during rendering. In both processes a frustum of rays is also used when a bundle of rays needs to be traced after (rough) specular reflection. The radiosity engine consists of a number of processor elements connected by a mesh network. The engine is connected to a host, probably a workstation or a cluster of workstations. Because each processor will be equipped with only a limited amount of memory, each processor element will be able to contain only a small portion of the patch database. When a processor needs a patch that is not in its memory, it sends a request to the other processor elements or to the host. In (Shen 93) strategies are presented to fill the memories. However, still much communication may be needed. Therefore also for this hardware implementation it is worthwhile to investigate whether coherence may reduce the communication. Hierarchical methods, as described in chapter 6, and especially grouping methods, as discussed in chapter 7, will be required in order to improve the performance of the radiosity engine and the quality of the images generated.

Bibliography

ANSI (1986), Nomenclatura and Definitions for Illuminating Engineering. Technical Report, American National Standards Institute. ANSI/IES RP-16-1986.

Arvo, J. (1986), Backward Ray Tracing, *SIGGRAPH '86 Developments in Ray Tracing Course Notes.*

Arvo, J., Kirk, D. (1989), A Survey of Ray Tracing Accelaration Techniques, in Glassner, A.S. (ed), *An Introduction to Ray Tracing*, Academic Press, London.

Baum, D.R., Rushmeier, H.E., Winget, J.M. (1989), Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors, *Computer Graphics (SIGGRAPH '89 Proceedings)*, vol 23 no 3, pp 325-334.

Baum, D.R., Mann, S., Smith, K.P., Winget, J.M. (1991), Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions, *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol 25 no 4, pp 51-60.

Bhate, N., Tokuta A. (1992), Photorealistic Volume Rendering of Media with Directional Scattering, *Proceedings Third Eurographics Workshop on Rendering*, Bristol UK, pp 227-245.

Blinn J.F. (1977), Models of Light Reflection for Computer Synthesized Pictures, *Computer Graphics (SIGGRAPH '77 Proceedings)*, vol 11 no 2, pp 192-198.

Campbell III, A.T., Fussell, D.S. (1990), Adaptive Mesh Generation for Global Diffuse Illumination, *Computer Graphics (SIGGRAPH '90 Proceedings)*, vol 24 no 4, pp 155-164.

Chen, S.E., Rushmeier, H.E., Miller, G., Turner, D. (1991), A Progressive Multi-Pass Method for Global Illumination, *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol 25 no 4, pp 164-174.

Clavé, S., Gross, M. (1991), A Rendering Pipeline for Street Lighting Simulation, *Proceedings Second Eurographics Workshop on Rendering*, Barcelona Spain.

Cohen, M.F., Greenberg, D.P. (1985), The Hemi-Cube: A Radiosity Solution for Complex Environments, *Computes Graphics (SIGGRAPH '85 Proceedings)*, vol 19 no 3, pp 31-40.

Cohen, M.F., Greenberg, D.P., Immel, D.S., Brock, P.J. (1986), An Efficient Radiosity Approach for Realistic Image Synthesis, *IEEE Computer Graphics and Applications*, vol 6 no 3, pp 26-35.

Cohen, M.F., Chen, S.E., Wallace, J.R., Greenberg, D.P. (1988), A Progressive Refinement Approach to Fast Radiosity Image Generation, *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol 22 no 4, pp 75-84.

Cook, R.L., Torrance K.E. (1982), A Reflectance Model for Computer Graphics, *ACM Transactions on Graphics*, vol 1 no 1, pp 7-24.

Cook, R.L., Porter, T., Carpenter, L. (1984), Distributed Ray Tracing, *Computer Graphics (Proceedings SIGGRAPH '84)*, vol 18 no 3, pp 137-145.

Cook, R.L. (1986), Stochastic Sampling in Computer Graphics, *ACM Transactions on Graphics*, vol 5 no 1, pp 51-72.

Dippé, M.A., Wold, E.H. (1985), Antialiasing Through Stochastic Sampling, *Computer Graphics (Proceedings SIGGRAPH '85)*, vol 19 no 3, pp 69-78.

Eo, K.S., Kyung, C.M. (1989), Hybrid Shadow Testing Scheme for Ray Tracing, *Computer Aided Design*, vol 21 no 1, pp 38-48.

Foley, J.D., Dam van, A., Feiner, S.K., Hughes, J.F. (1990), Computer Graphics, Principles and Practice, Second Edition, Addison-Wesley, Reading Massachusetts.

Glassner, A.S. (ed.) (1989), An Introduction to Ray Tracing, Academic Press, London.

Goral, C.M., Torrance, K.E., Greenberg, D.P., Battaile, B. (1984), Modelling the Interaction of Light Between Diffuse Surfaces, *Computer Graphics (SIGGRAPH '84 Proceedings)*, vol 18, no 3, pp 212-222.

Green, S.A., Paddon, D.J. (1989), Exploiting Coherence for Multiprocessor Ray Tracing, *IEEE Computer Graphics and Applications*, vol 9 no 6, pp 12-26.

Haines, E.A., Greenberg, D.P. (1986), The Light Buffer: A Ray Tracer Shadow Testing Accelerator, *IEEE Computer Graphics and Applications*, vol 6 no 9, pp 6-16.

Haines, E.A. (1991), Ronchamp: A Case Study for Radiosity, *SIGGRAPH '91 Frontiers in Rendering Course Notes*.

Hall, R. (1989), Illumination and Color in Computer Generated Imagery, Springer Verlag, New York.

Hanrahan, P., Salzman, D., Aupperle, L. (1991), A Rapid Hierarchical Radiosity Algorithm, *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol 25 no 4, pp 197-206.

He, X.D., Torrance K.E., Sillion F.X., Greenberg, D.P. (1991), A Comprehensive Physical Model for Light Reflection, *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol 25 no 4, pp 175-186.

Heckbert. P. (1986), Survey of Texture Mapping, *IEEE Computer Graphics and Applications*, vol 6 no 11, pp 56-67.

Heckbert, P. (1990), Adaptive Radiosity Textures for Bidirectional Ray Tracing, *Computer Graphics (SIGGRAPH '90 Proceedings)*, vol 24 no 4, pp 145-154.

Heckbert, P. (1991), Simulating Global Illumination Using Adaptive Meshing, PhD Thesis, Report No. UCB/CSD 91/636, CS Dept, University of California Berkeley.

Heckbert, P. (1992a), Discontinuity Meshing for Radiosity, *Proceedings Third Eurographics Workshop on Rendering*, Bristol UK, pp 203-216.

Heckbert, P. (1992b), Radiosity in Flatland, *Computer Graphics Forum (Eurographics '92 Proceedings)*, vol 11 no 3, pp 181-192.

Immel, D.S., Cohen, M.F., Greenberg, D.P. (1986), A Radiosity Method for Non-Diffuse Environments, *Computer Graphics (SIGGRAPH '86 Proceedings)*, vol 20 no 4, pp 132-142.

Jansen, F.W, Kok, A.J.F., Verelst, T. (1992), Hardware Challenges for Ray Tracing and Radiosity Algorithms, Seventh Workshop on Graphics Hardware, *Eurographics 92 Technical Report*, Cambridge England, September 1992, pp 123-134.

Jansen, F.W., Chalmers, A. (1993), Realism in Real Time ?, *Proceedings Fourth Eurographics Workshop on Rendering*, pp 27-46, Paris France.

Kajiya, J.T. (1986), The Rendering Equation, *Computer Graphics (SIGGRAPH '86 Proceedings)*, vol 20 no 4, pp 143-150.

Kalos, M.H., Whitlock, P.A. (1986), Monte Carlo Methods, John Wiley & Sons.

Kok, A.J.F., Yilmaz, C., Bierens, L.H.J. (1990), A Two-Pass Radiosity Method for Bézier Patches, in: Bouatouch, K., Bouville, C. (eds), *Photorealism in Computer Graphics (Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics, Rennes, France, 1990)*, Springer Verlag 1992, pp 117-126.

Kok, A.J.F., Jansen, F.W. (1991), Source Selection for the Direct Lighting Computation in Global Illumination, *Proceedings Second Eurographics Workshop on Rendering*, Barcelona Spain.

Kok, A.J.F., Jansen, F.W., Woodward C. (1991), Efficient Complete Radiosity Ray Tracing using a Shadow Coherence Method, Reports of the Faculty of Technical Mathematics and Informatics no 91-63, Delft University of Technology. Also appeared in: *The Visual Computer*, vol 10 no 1 (1993), pp 19-33.

Kok, A.J.F., Jansen, F.W. (1992), Adaptive Sampling of Area Light Sources in Ray Tracing Including Diffuse Interreflection, *Computer Graphics Forum (Eurographics '92 Proceedings)*, vol 11 no 3, pp 289-298.

Kok, A.J.F (1993), Grouping Patches in Progressive Radiosity, *Proceedings Fourth Eurographics Workshop on Rendering*, Paris France, pp 221-231.

Languénou, E., Bouatouch, K., Tellier, P. (1992), An Adaptive Discretization Method for Radiosity, *Computer Graphics Forum (Eurographics '92 Proceedings)*, vol 11 no 3, 205-216.

Lee, M.E., Redner, R.A., Uselton, S.P. (1985), Statistically Optimized Sampling for Distributed Ray Tracing, *Computer Graphics (SIGGRAPH '85 Proceedings)*, vol 19 no 3, pp 61-67.

Liere, R. van (1991), Divide and Conquer Radiosity. *Proceedings Second Eurographics Workshop on Rendering*, Barcelona Spain.

Lischinski, D., Tampieri, F., Greenberg, D.P. (1992), Discontinuity Meshing for Accurate Radiosity, *IEEE Computer Graphics and Applications*, vol 12 no 6, pp 25-39.

Malley, T.J.V. (1988), A Shading Method for Computer Generated Images, Master's thesis, Dept. of Computer Science, University of Utah.

Moon, P. (1936), The Scientific Basis of Illuminating Engineering, MacGraw-Hill, New York.

Nicodemus, F.E., Richmond, J.C., Hsia, J.J., Ginsberg, I.W., Limperis, T. (1977), Geometrical Considerations and Nomenclature for Reflectance, NBS Monograph 160.

Nishita, T., Nakamae, E. (1985), Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection, *Computer Graphics (SIGGRAPH '85 Proceedings)*, vol 19 no 3, pp 23-30.

Pattanaik, S.N., Mudur, S.P. (1992), Computation of Global Illumination by Monte Carlo Simulation of the Particle Model of Light, *Proceedings Third Eurographics Workshop on Rendering*, Bristol UK, pp 71-83.

Pattanaik, S.N. (1993), Computational Methods for Global Illumination and Visualisation of Complex 3D Environments, PhD Thesis, Birla Institute of Technology & Science, Pilani India.

Phong, B.T. (1975), Illumination for Computer Generated Pictures, *Communications of the ACM*, vol 18 no 6, pp 311-317.

Rushmeier, H.E., Torrance K.E. (1987), The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium, *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol 21 no 4, pp 293-302.

Rushmeier, H.E. (1988), Realistic Image Synthesis for Scenes with Radiatively Participating Media, PhD Thesis, Cornel University.

Rushmeier, H.E., Torrance, K.E. (1990), Extending the Radiosity Method to Include Specularly Reflecting and Translucent Materials, *ACM Transactions on Graphics*, vol 9 no 1, pp 1-27.

Rushmeier, H., Patterson, C., Veerasamy, A. (1993), Geometric Simplification for Indirect Illumination Calculations, *Proceedings of Graphics Interface '93*, pp 227-236, Toronto Canada.

Shao, M.Z., Peng, Q.S., Liang, Q.D. (1988), A New Radiosity Approach by Procedural Refinements for Realistic Image Synthesis, *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol 22 no 4, pp 93-101.

She, L.S., Deprettere, E.F. (1992), A Parallel-Pipelined Multiprocessor System for the Radiosity Method, Seventh Workshop on Graphics Hardware, *Eurographics 92 Technical Report*, Cambridge England, September 1992, pp 106-122.

Shen, L.S. (1993), A Parallel Image Rendering Algorithm and Architecture Based on Ray Tracing and Radiosity Shading, PhD Thesis, Delft University of Technology, The Netherlands.

Shirley, P. (1990a), A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes, *Proceedings of Graphics Interface '90*, pp 205-212.

Shirley, P. (1990b), Physically Based Lighting Calculations for Computer Graphics, PhD Thesis, Dept. of Computer Science, University of Illinois, Urbana-Champaign.

Shirley, P., Wang, C. (1991), Direct Lighting Calculations by Monte Carlo Integration, *Proceedings Second Eurographics Workshop on Rendering*, Barcelona Spain.

Siegel, R., Howell, J.R. (1981), Thermal Radiation Heat Transfer, Hemisphere Publishing Corporation, New York.

Sillion, F., Puech, C. (1989), A General Two-Pass Method Integrating Specular and Diffuse Reflection, *Computer Graphics (SIGGRAPH '89 Proceedings)*, vol 25 no 4, pp 187-196.

Sparrow, E.M., Cess, R.D. (1978), Radiation Heat Transfer, Hemisphere Publishing Company, Washington.

Tampieri, F., Lischinski, D. (1991), The Constant Radiosity Assumption Syndrome, *Proceedings Second Eurographics Workshop on Rendering*, Barcelona Spain.

Wallace, J.R., Cohen, M.F., Greenberg, D.P. (1987), A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods, *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol 21 no 4, pp 311-320.

Wallace, J.R., Elmquist, K.A., Haines, E.A. (1989), A Ray Tracing Algorithm for Progressive Radiosity, *Computer Graphics (SIGGRAPH '89 Proceedings)*, vol 23 no 3, pp 315-324.

Ward, G.J., Rubinstein, F.M., Clear, R.D. (1988), A Ray Tracing Solution for Diffuse Interreflection, *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol 22 no 4, pp 85-92.

Ward, G.J. (1991), Adaptive Shadow testing for Ray Tracing, *Proceedings Second Eurographics Workshop on Rendering*, Barcelona Spain.

Ward, G.J., Heckbert P.S. (1992), Irradiance Gradients, *Proceedings Third Eurographics Workshop on Rendering*, Bristol UK, pp 85-98.

Ward, G.J. (1992), The Radiance Lighting Simulation System, *SIGGRAPH '92 Global Illumination Course Notes*.

Whitted, T. (1980), An Improved Illumination Model for Shaded Display, *Communications of the ACM*, vol 23 no 6, pp 343-349.

Woo, A., Amanatides, J. (1990), Voxel Occlusion Testing: A Shadow Determination Accelerator for Ray Tracing, *Proceedings of Graphics Interface '90*, pp 213-220.

Woodward, C. (1990), Methods for Computer-Aided Design of Free-Form Objects, *Acta Polytechnica Scandinavica*, Mathematics and Computer Science Series no 56, Finnish Academy of Technology, Helsinki.

Xu, H., Peng, Q.S., Liang, Y.D. (1989), Accelerated Radiosity Method for Complex Environments, *Eurographics '89*, Elsevier Science Publishers, Amsterdam, pp 51-61.

Appendix A Description of test models

The algorithms presented in this thesis are tested with three different models: triangle, small room, and computer room. These models have a different complexity.

Table A.1 summarizes the main characteristics of the test models. The surfaces that are used for describing the scenes are often too large to be handled correctly by radiosity programs. Therefore large surfaces are automatically subdivided into smaller patches. These patches are the basic entities for the radiosity computations. Sample points are placed on these patches. The radiosity program computes radiances for these sample points.

model	surfaces	patches	sample points[1]
A triangle	3	3	314
B small room	477	522	16378
C computer room	8613	10596	51542

Table A.1 Description of test models

A.1 Model A, triangle

Model A is a very simple model consisting of only three polygons: a square area light source, a triangular surface, and a square surface. Figure A.1 shows the geometry of the triangle scene. This model can very well be used to test accuracy of shadows that are generated by the different algorithms. The light source illuminates the triangular and the square surface. The triangular surface occludes (parts of) the light source for some parts of the square surface. Therefore umbra and penumbra regions should be visible on the square surface. The meshing that is applied is shown in figure A.2.

All illumination in this scene is direct illumination. The light source and the triangle are mutual visible, and the light source and the square are mutual visible. The triangle and the square cannot see each other. The light source only emits light, it does not reflect. Therefore there is no other light interaction than direct illumination from the light source to the two other patches.

---

[1] Default value. When another number of sample points is used for an experiment, then it is mentioned in the description of the experiment.

A.2 Model B, small room

Model B is a small room with several objects. It has a moderate complexity. Figure A.3 shows the geometry of the small room scene. The scene contains a teapot, modelled with curved surfaces. In our rendering system it is possible to use rational bicubic Bézier patches (Kok et al 90). All other furniture, and the walls and the floors are modelled with polygons. Several surfaces in the scene are (partly) specular. The mirror is specularly reflecting. The surfaces of the teapot are partly diffusely and partly specularly reflecting. Two surfaces of the cabinet are specularly transmitting. The scene is illuminated by two spot light sources on the corners on the right. The spot sources are pointed to the teapot on the table, and are responsible for sharp shadows. The scene is also illuminated by two area light sources: a light source on the ceiling and a light source (window) on the left wall.

This scene is closed, and large parts of it have a high diffuse reflectivity. Therefore a lot of diffuse interreflection occurs. Rendering without computing the diffuse interreflection will result in a rather dark picture.

For this model two different view points are used. View 1 gives an overview of the scene. View 2 zooms in on the teapot on the table. The meshing that is applied is shown in figures A.4 and A.5.

A.3 Model C, computer room

Model C is a model of a room with computers. It describes a rather complex scene. Figure A.6 shows the geometry of the computer room scene. The room contains nine desks with chairs. On eight of these desks two computers are placed. These computers are modelled very accurately. They each consist of many polygons. The monitor is modelled with 24 polygons, the keyboard with 417 polygons (each key is modelled with 5 polygons), and the mouse with 17 polygons. The computer room is illuminated with 12 area light sources on the ceiling. The screens of the computers that are turned on also emit light. However, the power emitted by these screens is small. Therefore these screens do not contribute largely to the illumination of the scene. Texture mapping is used (only for rendering purposes) to improve the reality of the images. The meshing that is applied is shown in figure A.7.

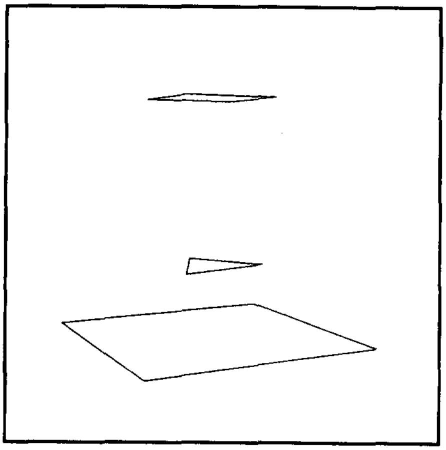


Fig A.1 Geometry of triangle model

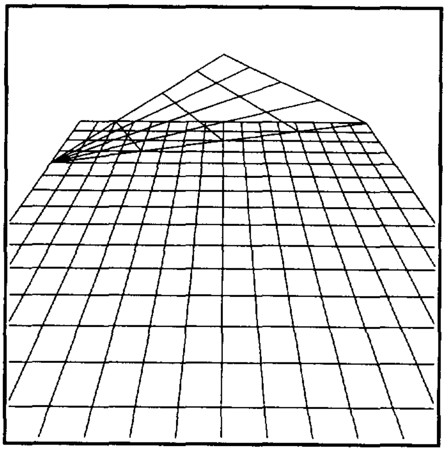


Fig A.2 Meshing for triangle model

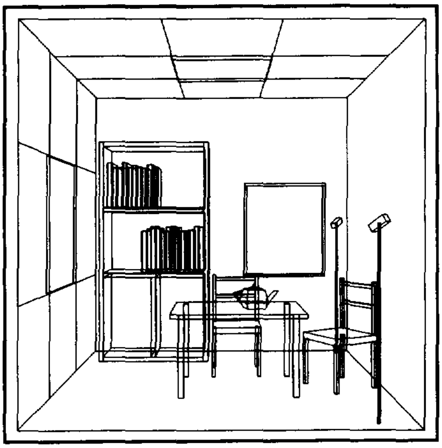


Fig A.3 Geometry of small room model

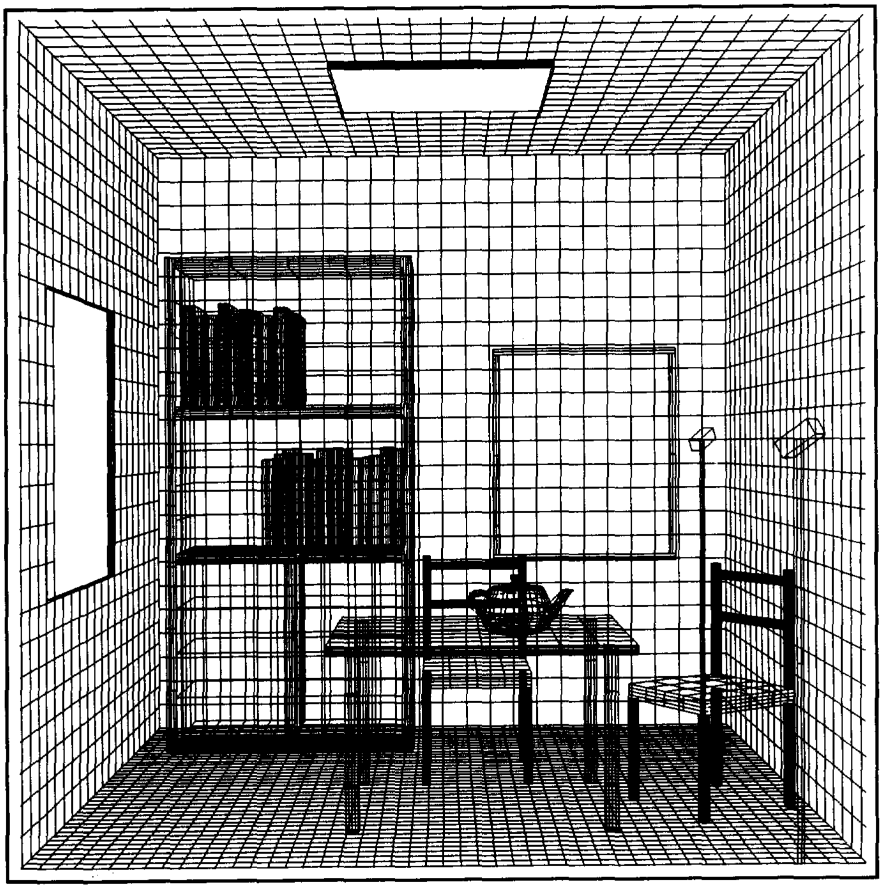


Fig A.4 Meshing for small room model, view 1

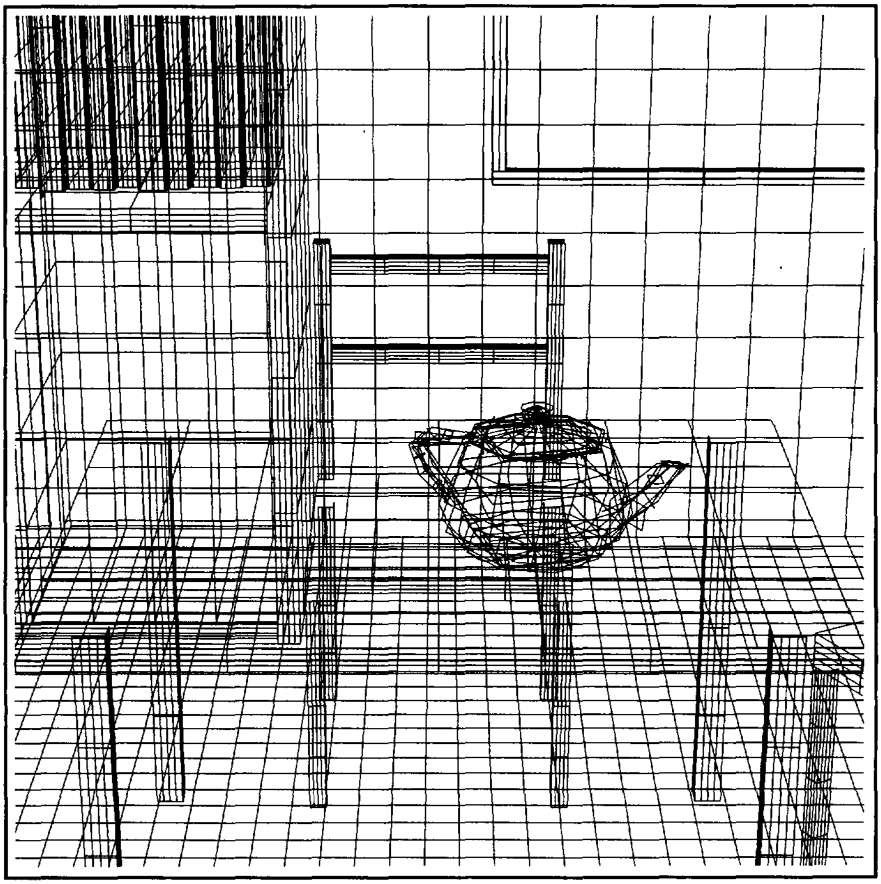


Fig A.5 Meshing for small room model, view 2

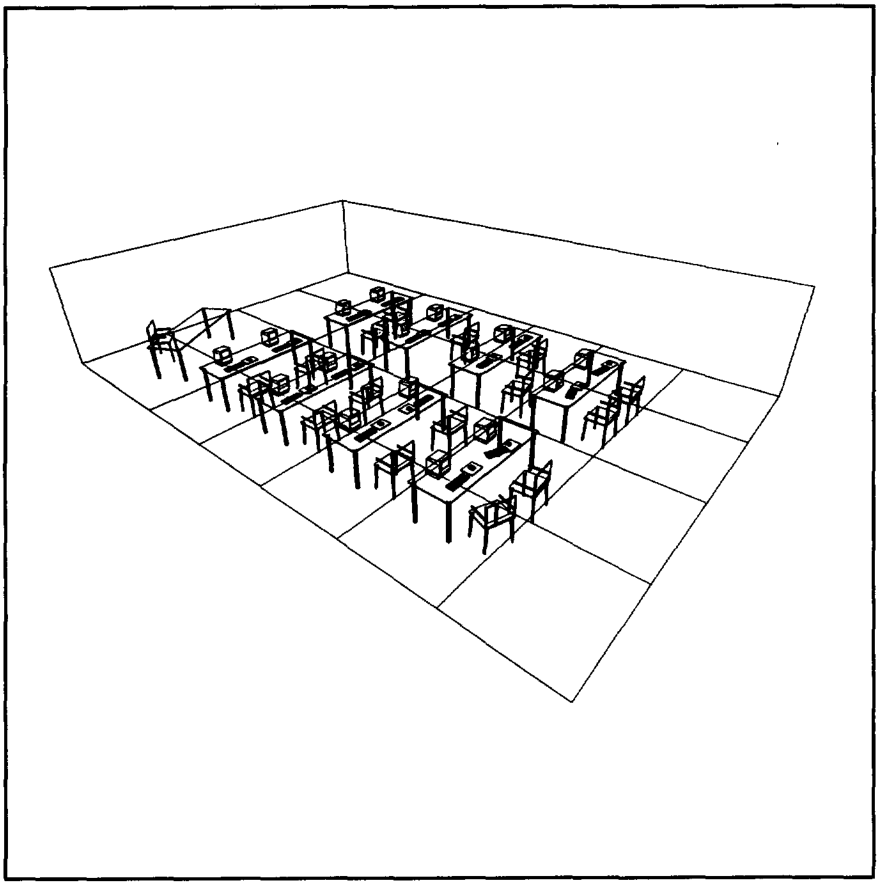


Fig A.6 Geometry of computer room model

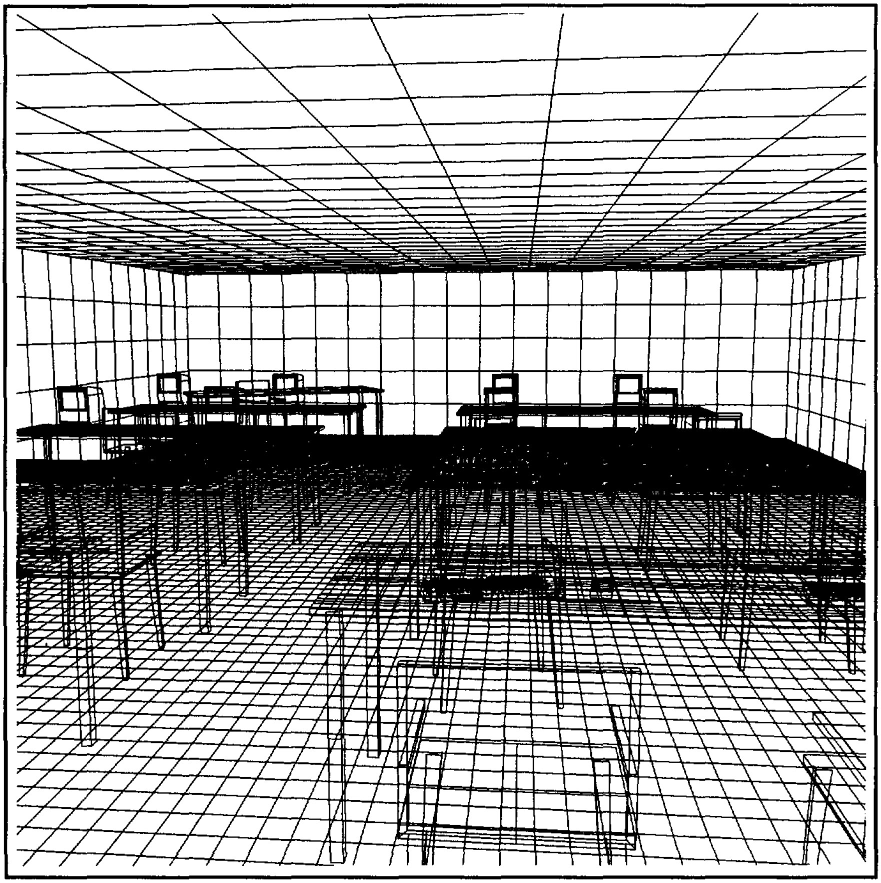


Fig A.7 Meshing for computer room model

Summary

Ray tracing and radiosity methods for photorealistic image synthesis

Realistic image synthesis strives to generate images that not only look realistic, but that are also physically correct. All kinds of optical effects must be computed to reach this goal. Examples of these effects are indirect illumination, soft shadows, caustics, etc. Realistic image synthesis can be used for several applications, for example in architecture, in lighting design, for special effects in film, for (flight) simulators, etc.

A combination of radiosity and ray tracing methods will be able to compute all optical effects accurately. The radiosity process computes the illumination for a scene. Because it is impossible to compute an illumination value for each point in a scene, only a limited number of points is selected by subdivision of the surfaces in the scene into a mesh of patches. For each point in the mesh an illumination value is computed. During the ray tracing based rendering process, these illumination values are used to display the visible points. A disadvantage of such a hybrid method is that the quality of the resulting images strongly depends on the meshing resolution that is applied.

In this thesis a new hybrid method is presented, that is less dependent on the meshing. To avoid inaccurate shadows due to inappropriate meshing, during the radiosity pass only the illumination that is responsible for minor shading gradients is computed. The shading that is responsible for important shading gradients (perceptible shading contrasts) is added during rendering by recomputing the illumination of the important sources for all the visible points. A source selection mechanism is used to detect which sources are responsible for the important shading contrasts. This new method is able to generate very accurate images. The method is also very flexible. Most of the known hybrid radiosity and ray tracing methods can be simulated with this algorithm, depending on how strictly selection criteria are applied.

However, this new method still requires much computation time. Therefore the efficiency of both the radiosity and the ray tracing process must be improved. The radiosity method can be improved by using a hierarchical meshing of the scene. This meshing should not only include an adaptive subdivision of surfaces, but also the grouping of several surfaces into (a hierarchy of) groups. Interaction of light between two objects is computed at the most appropriate level in the mesh hierarchy, considering accuracy and efficiency.

For the rendering pass, methods are developed to reduce the number of shadow rays that are used to compute the most important illumination contributions. The number of shadow rays can be reduced by exploiting shadow coherence, because shadows tend to be very coherent and homogeneous. Also, for sampling large area light sources, an adaptive stochastic sampling method may reduce the number of shadow rays compared to other sampling methods.

In future, further speed-up may also be gained by parallel and specialized hardware implementations.

Samenvatting

Ray tracing en radiosity methoden voor foto-realistische beeldgeneratie

Realistische beeldgeneratie is gericht op de generatie van afbeeldingen die er niet alleen realistisch uitzien, maar die bovendien fysisch correct zijn. Allerlei optische effecten moeten berekend worden om dit doel te bereiken. Voorbeelden zijn indirecte belichting, zachte schaduwen, enz. Er zijn verschillende toepassingen voor realistische beeldgeneratie te bedenken: in de architectuur, voor het ontwerp van belichtingsoplossingen, voor speciale effecten in film, in (vlucht)simulatoren, enz.

Een combinatie van radiosity en ray tracing methoden is in staat om alle optische effecten nauwkeurig te berekenen. Het radiosity proces berekent de belichting voor een scène. Omdat het onmogelijk is om een belichtingswaarde voor ieder punt in een scène te bepalen, wordt een beperkt aantal punten gekozen door alle oppervlakken in de scène op te delen in een netwerk van kleine oppervlakjes. Voor ieder hoekpunt in het netwerk wordt een belichtingswaarde berekend. Gedurende de generatie van de afbeelding door middel van ray tracing worden deze belichtingswaarden gebruikt om de zichtbare punten af te beelden. Een nadeel van de zo'n gecombineerde methode is dat de kwaliteit van de resulterende afbeeldingen sterk afhankelijk is van de resolutie van de gebruikte opdeling van de scène.

In dit proefschrift wordt een nieuwe gecombineerde methode gepresenteerd, die minder afhankelijk is van de resolutie van de opdeling. Om onnauwkeurige schaduwen als gevolg van een onvoldoende opdeling te voorkomen, wordt tijdens het radiosity proces alleen de belichting berekend die verantwoordelijk is voor de minder duidelijke schakeringen in de belichting. De belichting die verantwoordelijke is voor duidelijke schakeringen (duidelijke schaduw overgangen) wordt gedurende het afbeeldingsproces toegevoegd door de belichting door belangrijke lichtbronnen voor alle zichtbare punten opnieuw te berekenen. Een lichtbronselectie-mechanisme wordt gebruikt om te detecteren welke bronnen verantwoordelijk zijn voor de belangrijke contrasten in de belichting. Met deze nieuwe methode kunnen zeer nauwkeurige afbeeldingen gemaakt worden. Bovendien is de methode erg flexibel. De meeste bekende gecombineerde radiosity en ray tracing methoden kunnen worden gesimuleerd met dit algoritme, afhankelijk van hoe streng de selectiecriteria worden toegepast.

De nieuwe methode vereist echter nog steeds lange rekentijden. Daarom moeten zowel het radiosity als het ray tracing proces worden verbeterd. De radiosity methode kan worden verbeterd door het gebruik van een hiërarchische opdeling van de scène. Deze opdeling moet niet alleen bestaan uit een adaptieve opdeling van de oppervlakken, maar moet ook de groepering van verschillende oppervlakken in (een hiërarchie van) groepen bevatten. De interactie van licht tussen twee objecten wordt berekend op het meest geschikte niveau in de hiërarchie, waarbij nauwkeurigheid en efficiency tegen elkaar afgewogen worden.

Voor het beeldgeneratieproces zijn methoden ontwikkeld om het aantal schaduwstralen, die gebruikt worden om de belangrijkste belichtingsbijdragen te berekenen, te verminderen. Dit aantal kan verminderd worden door gebruik te maken van schaduwcoherentie. Schaduwen zijn namelijk over het algemeen coherent en homogeen. Bovendien kan een adaptieve stochastische sampling methode, in vergelijking met andere sampling methoden, het aantal schaduwstralen om grote oppervlaktelichtbronnen te samplen verminderen.

In de toekomst zullen de methoden verder versneld kunnen worden door parallelle en gespecialiseerde hardware implementaties.

Curriculum vitae

Adrianus Johannes Franciscus (Arjan) Kok werd op 25 november 1965 geboren te Breda. In 1984 behaalde hij het VWO diploma aan het Stedelijk Gymnasium in Breda.

In 1984 begon hij aan de studie Informatica aan de Technische Universiteit Delft. In 1989 studeerde hij af bij de sectie Technische Toepassingen van de Informatica, met als afstudeerproject het ontwerp en de implementatie van een systeem voor kinematische simulatie van mechanismen.

Van december 1989 tot december 1993 was hij werkzaam als assistent in opleiding (AIO) bij de Faculteit der Technische Wiskunde en Informatica aan de Technische Universiteit Delft. Bij de leerstoel Computer Graphics heeft hij promotie-onderzoek gedaan naar methoden voor de hoogwaardige (realistische) visualisatie van scènes.