Advanced Global Illumination

Philip Dutré (co-organizer)
Departement of Computer Science
Katholieke Universiteit Leuven
BELGIUM

Kavita Bala (co-organizer)
Program of Computer Graphics
Cornell University
USA

Philippe Bekaert
Max-Planck-Institut für Informatik
Saarbrücken
GERMANY

SIGGRAPH 2002 Course 2 (Half Day)

Course Abstract

In this course, we describe the fundamentals of light transport and techniques for computing the global distribution of light in a scene. The main focus will be on the light transport simulation since the quality and efficiency of a photo-realistic renderer is determined by its global illumination algorithm. We explain the basic principles and fundamentals behind algorithms such as stochastic ray tracing, path tracing, light tracing and stochastic radiosity.

Presenters

Philip Dutré
Assistant Professor
Department of Computer Science, University of Leuven
Celestijnenlaan 200A
B-3001 Leuven
BELGIUM
Email: phil@cs.kuleuven.ac.be
URL: http://www.cs.kuleuven.ac.be/~phil/

Philip Dutré is an assistant professor at the Department of Computer Science at the Katholieke Universiteit Leuven, Belgium. Previously, he was a post-doctoral research associate in the Program of Computer Graphics at Cornell University. He received a Ph.D. in Computer Science from the Katholieke Universiteit Leuven, Belgium in 1996, titled “Mathematical Frame-works and Monte Carlo Algorithms for Global Illumination in Computer Graphics” under supervision of Prof. Dr. Y. Willems. His current research interests include real-time global illumination, probabilistic visibility, accurate shadows and augmented reality. He is involved in teaching undergraduate and graduate courses covering computer graphics and rendering algorithms. He is the author of several papers on global illumination algorithms and the light transport problem.

Kavita Bala
Postdoctoral Associate
Program of Computer Graphics - Cornell University
580 Rhodes Hall
Cornell University
Ithaca NY 14853-3801
Email: kb@graphics.cornell.edu
URL: http://www.graphics.cornell.edu/~kb/

Kavita Bala is a post-doctoral research associate in the Program of Computer Graphics at Cornell University. She received her doctorate degree from the Massachusetts Institute of Technology in 1999. Her thesis, titled “Radiance Interpolants for Interactive Scene Editing and Ray Tracing”, was supervised by Profs. Julie Dorsey and Seth Teller. Her current research interests include interactive rendering, 4D radiance representations, global illumination algorithms, and image-based rendering and she has published several papers in these fields. At Cornell University, she has been involved in teaching the introductory undergraduate computer graphics course and the graduate advanced rendering course.

Philippe Bekaert
Postdoctoral Research Fellow
Max Planck Institut fuer Informatik (Computer Graphics Unit), Saarbrücken, Germany
Im Stadtwald 46.1
66123 Saarbrücken, Germany
Email: philippe@mpi-sb.mpg.de
URL: http://www.mpi-sb.mpg.de/~philippe/

Philippe Bekaert is a post-doctoral research fellow at the Max Planck Institut fuer Informatik in Saarbruecken, Germany. He received a Ph.D. in Computer Science, titled "Hierarchical and Stochastic Algorithms for Radiosity", from the *Katholieke Universiteit Leuve*n, Belgium, in December 1999 under the supervision of Prof. Dr. Y. Willems. He is author or co-author of numerous papers on stochastic radiosity. His current research interests include interactive global illumination and the Monte Carlo method.

Syllabus

The course syllabus assumes a half-day course (2 modules of 1.75 hours each). The 1st module focuses on the fundamentals of all global illumination algorithms, and can serve as an introduction for designing you own global illumination program. The 2nd module is more of a case study for a specific family of global illumination algorithms: stochastic radiosity. The 2nd module also concludes with an overview of recent advances and trends in global illumination research.

1st module (105 minutes)

Introduction (5 minutes): Philip Dutré

Radiometry (40 minutes): Kavita Bala

Nature of light, and how to model light in computer graphics
Radiometric quantities (radiance, irradiance, flux)
Bidirectional reflectance distribution functions
Transport equations and rendering equation; particle model of light

General Strategies for Solving the Rendering Equation (60 minutes): Philip Dutré

Introduction to the Monte Carlo method
Monte Carlo integration applied to the rendering equation: path tracing
Next Event Estimation (explicit light source sampling)
General path generation strategies

2nd module (105 minutes)

Stochastic Radiosity (65 minutes): Philippe Bekaert

Radiosity equations – how to derive them from the rendering equation
Stochastic relaxation methods for radiosity
Random walk methods (particle tracing, density estimation)
Variance reduction techniques and hierarchical refinement

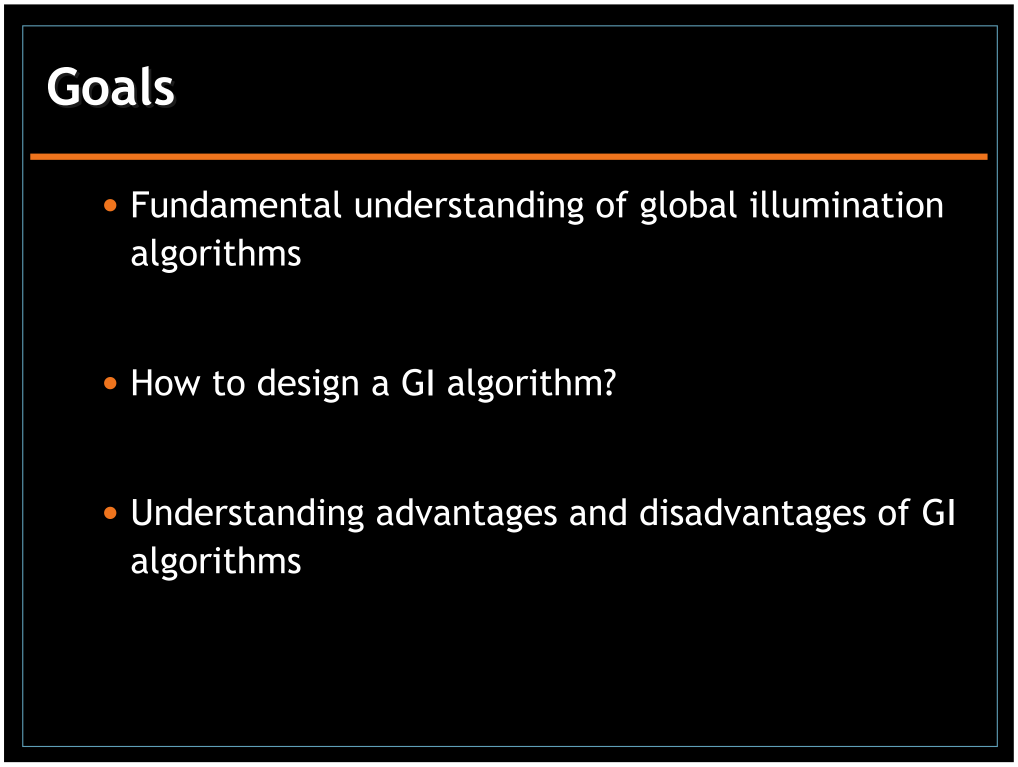
Future Trends (30 minutes): Kavita Bala

Summary Statement (10 minutes): Philip Dutré

Prerequisites

A basic understanding of classic photo-realistic rendering algorithms, such as basic ray tracing and radiosity is assumed. Knowledge of probability theory will be very helpful. Some familiarity with transport equations and radiometry will be useful, but is not necessary.

Table Of Contents



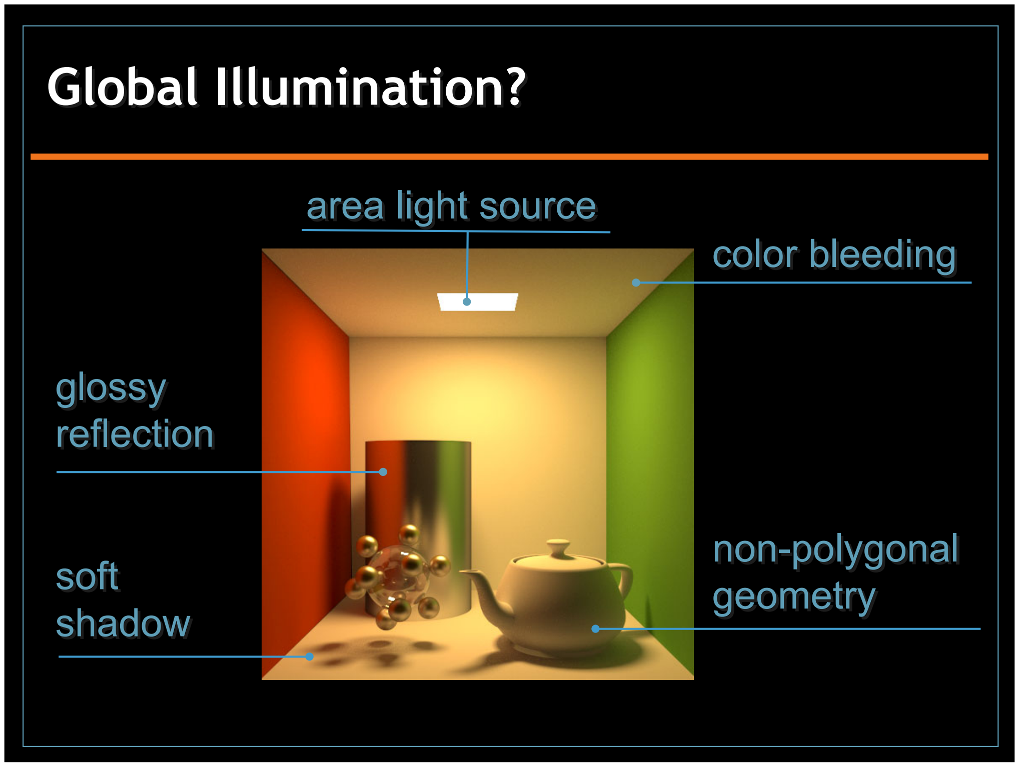
Goals

- Fundamental understanding of global illumination algorithms
- How to design a GI algorithm?
- Understanding advantages and disadvantages of GI algorithms

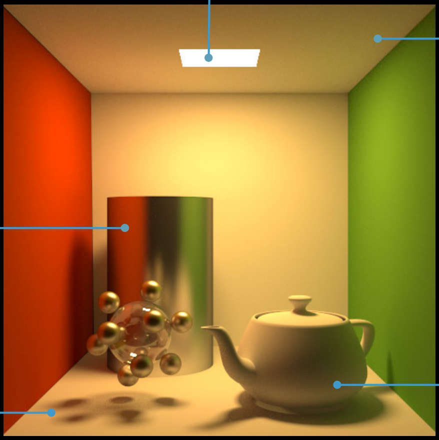
The goal of this course is to offer a fundamental understanding of global illumination algorithms.

More specifically, we will explain how one can design a global illumination algorithm, starting from the fundamental light transport equations. This approach differs from the more traditional ad-hoc approaches, where people start from a basic ray tracer, and then add more effects on a 'menu' basis.

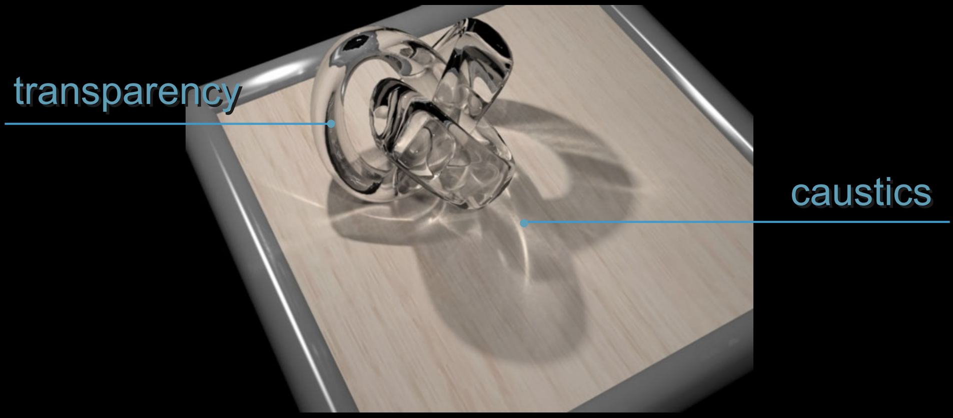
Looking at global illumination algorithms this way, it is easier to look at specific advantages and disadvantages of different approaches.



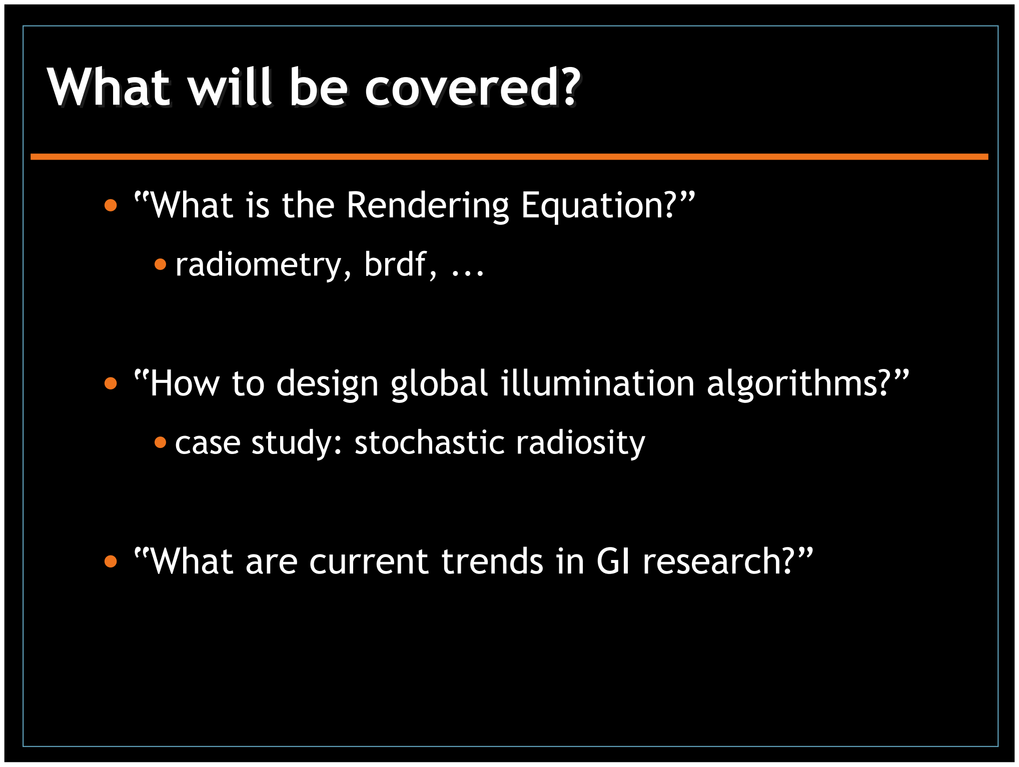
Global Illumination?



Global Illumination?



(F. Suykens 2001 - rendered with RenderPark)



What will be covered?

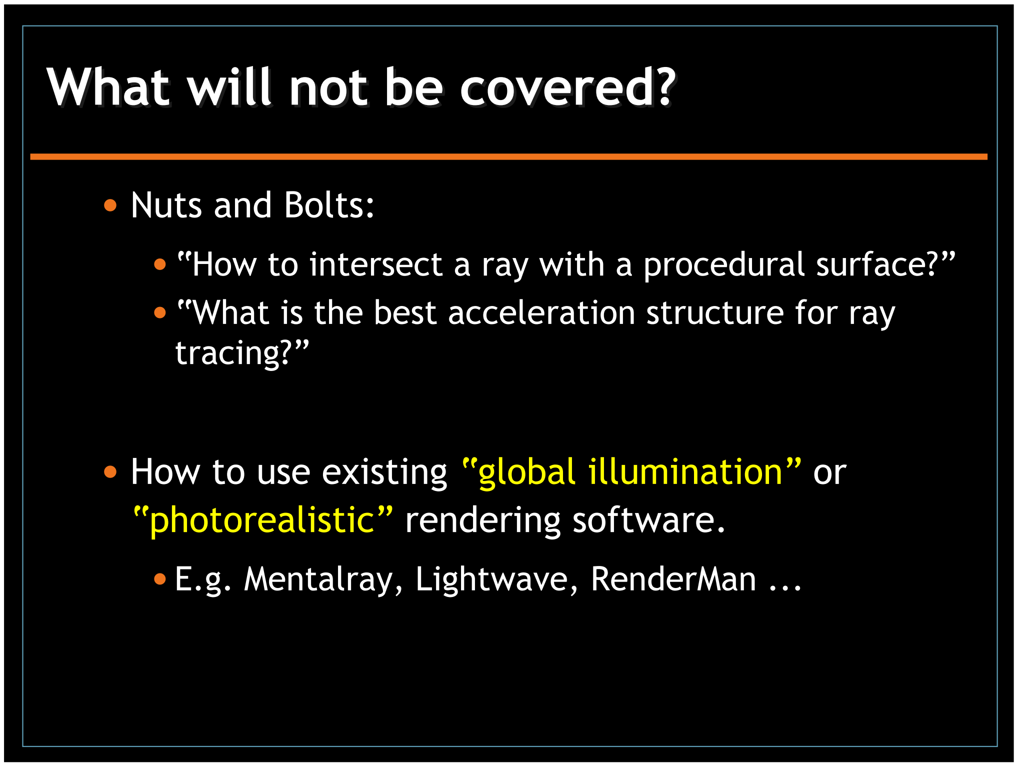
- "What is the Rendering Equation?"
  - radiometry, brdf, ...
- "How to design global illumination algorithms?"
  - case study: stochastic radiosity
- "What are current trends in GI research?"

This course will cover the following topics:

- Radiometry and the rendering equation. The rendering equation is the fundamental transport equation that describes light transport in a three-dimensional scene. It is a recursive integral equation, which is difficult to solve analytically. So numerical techniques need to be used.

- Starting from the rendering equation, we can design global illumination algorithms based on the general notion of path transport. Different choices give rise to different algorithms, each with their own error characteristics.

- An important case study is the class of stochastic radiosity methods. Employing stochastic relaxation methods or random walks, they can provide much better convergence than the more classic finite element techniques



What will not be covered?

- Nuts and Bolts:
  - "How to intersect a ray with a procedural surface?"
  - "What is the best acceleration structure for ray tracing?"
- How to use existing "global illumination" or "photorealistic" rendering software.
  - E.g. Mentalray, Lightwave, RenderMan ...

We will not cover any lower-level functionality needed to write and design global illumination algorithms. Basic operations such as ray-geometry intersection tests, or acceleration structures for speeding up ray-casting, are not the focus of this course.

Also, we will not focus on how to use existing software packages that can render photo-realistic images.

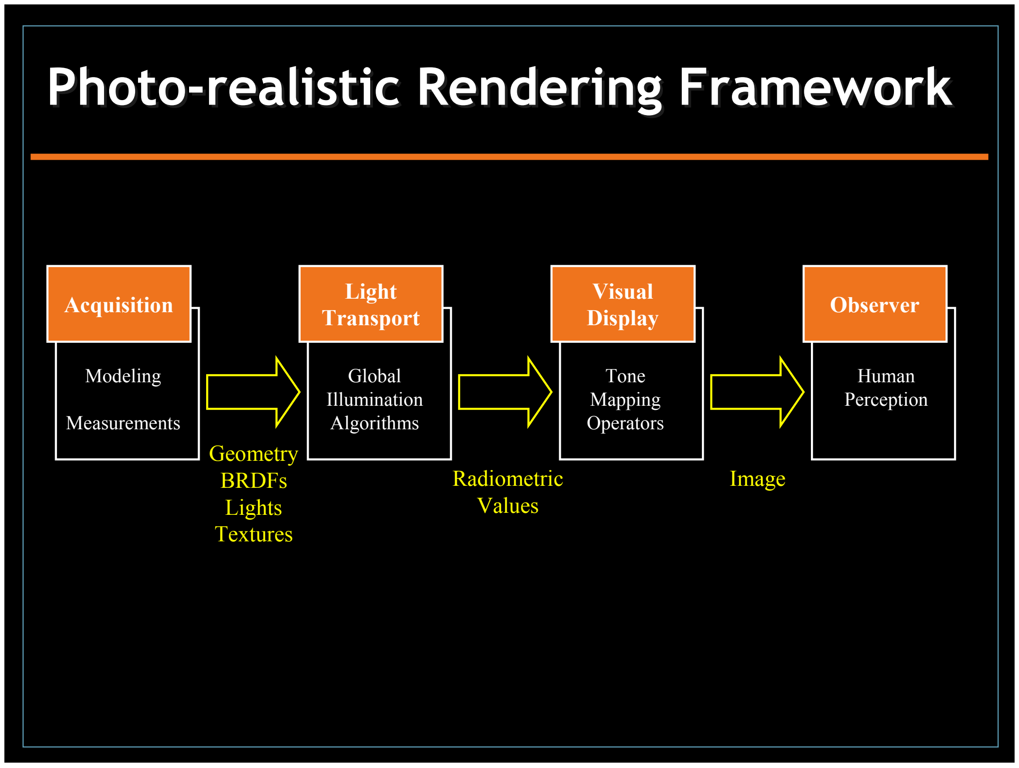


Photo-realistic Rendering Framework

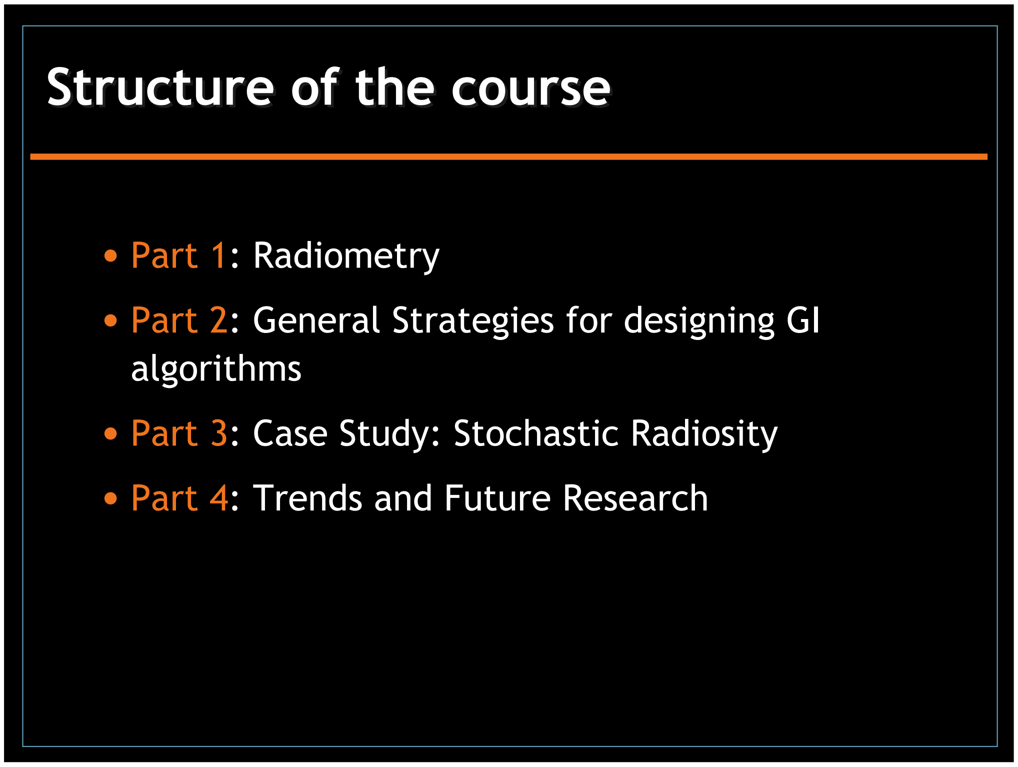
The general framework for a photo-realistic rendering system can be composed of four parts:

- Acquisition of data (measurements of geometry and materials)
- Light transport simulation (global illumination algorithms)
- Visual Display (tone mapping operators)
- Human observer

This course focuses on the global illumination part. We will touch on some of the other parts, but will not go into any detail.

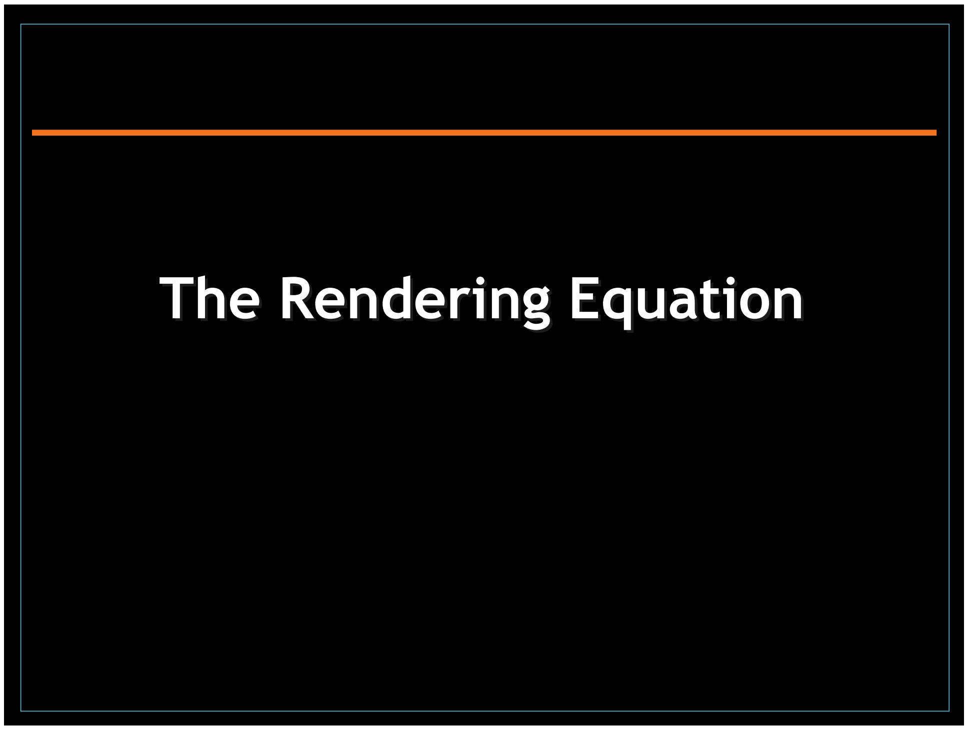
This framework for photorealistic rendering is described in more detail in:

Donald P. Greenberg, Kenneth Torrance, Peter Shirley, James Arvo, James Ferwerda, Sumanta Pattanaik, Eric Lafortune, Bruce Walter, Sing-Choong Foo, and Ben Trumbore. A framework for realistic image synthesis. In Turner Whitted, editor, SIGGRAPH 97 ConferenceProceedings, Annual Conference Series, pages 477--494. ACM SIGGRAPH, Addison Wesley, August 1997.

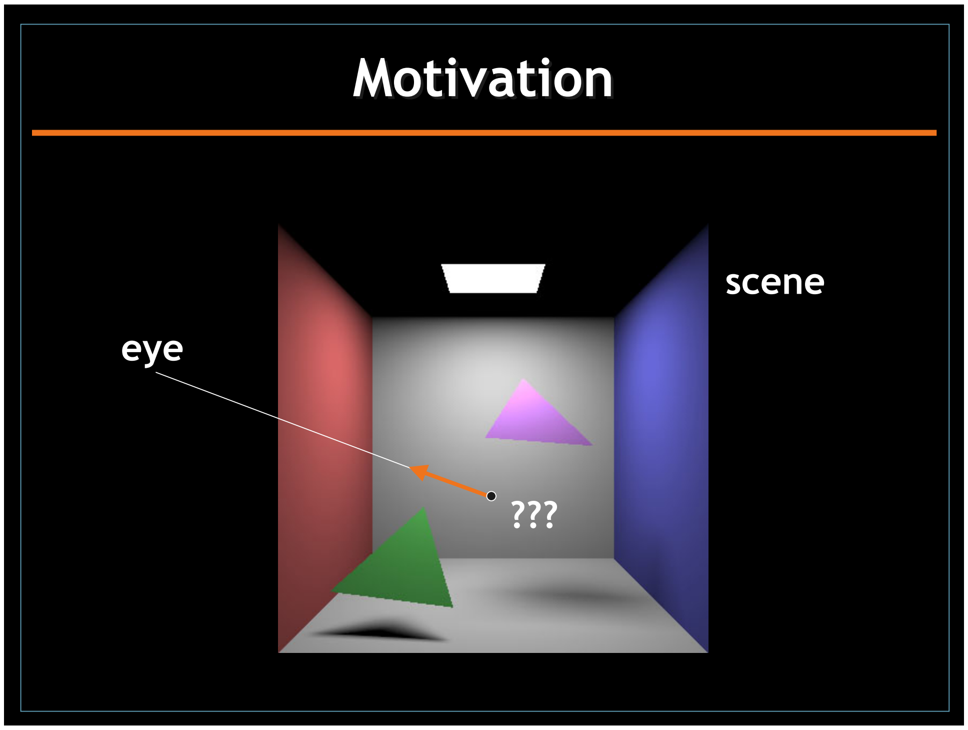


Structure of the course

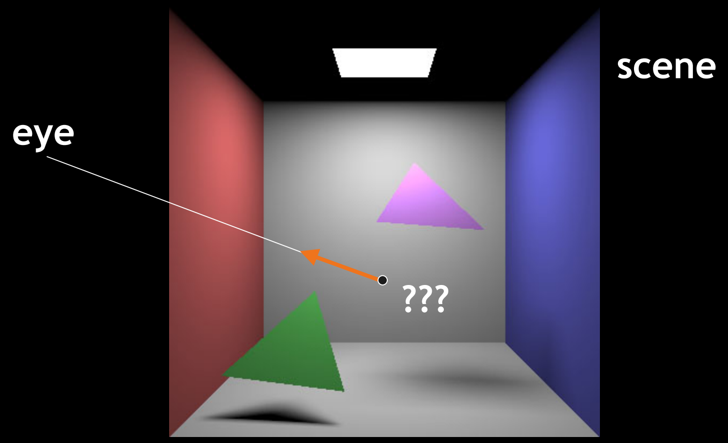
- Part 1: Radiometry
- Part 2: General Strategies for designing GI algorithms
- Part 3: Case Study: Stochastic Radiosity
- Part 4: Trends and Future Research



The Rendering Equation

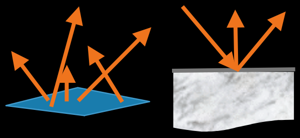


Motivation

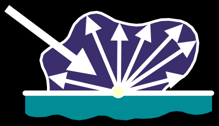


Outline

- Light Model



- Radiometry
- Materials: Interaction with light
- Rendering equation



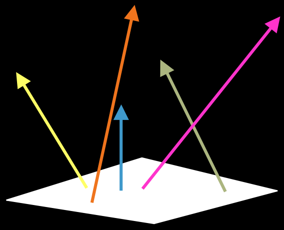
Light Models

- Geometric Optics
  - Emission, Reflection / Refraction, Absorption
- Wave Model

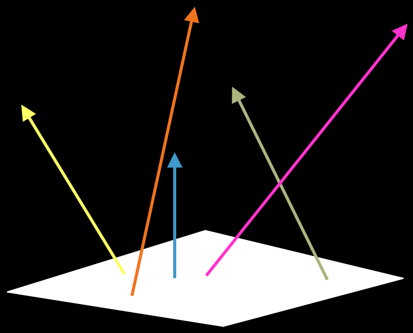
  - Maxwell’s Equations
  - Object size comparable to wavelength
  - Diffraction & Interference, Polarization
- Quantum Model
  - Fluorescence, Phosphorescence

Geometric Optics: Properties

- Light travels in straight lines
- Rays do not interact with each other
- Rays have color(wavelength), intensity

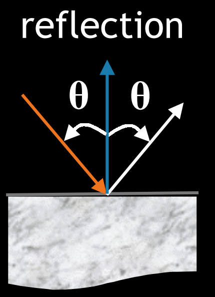


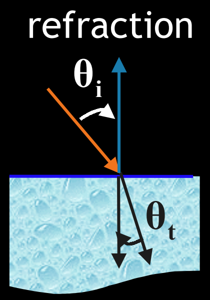
Emission



Reflections/Refractions

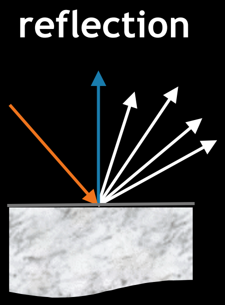
- Interface between 2 materials

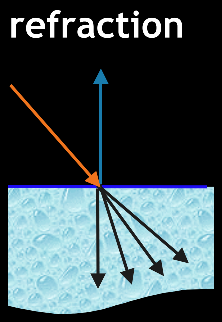




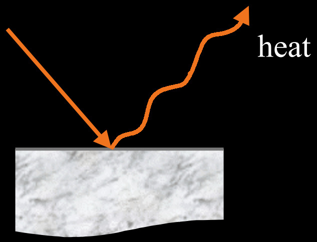
- Specular reflections and refractions
  - One direction

Realistic Reflections/Refractions





Absorption



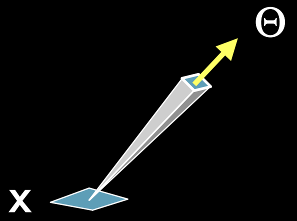
Geometric Optics: other effects

- Participating Media

- Varying index of refraction

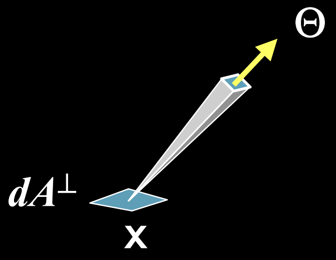
Radiometry: radiance

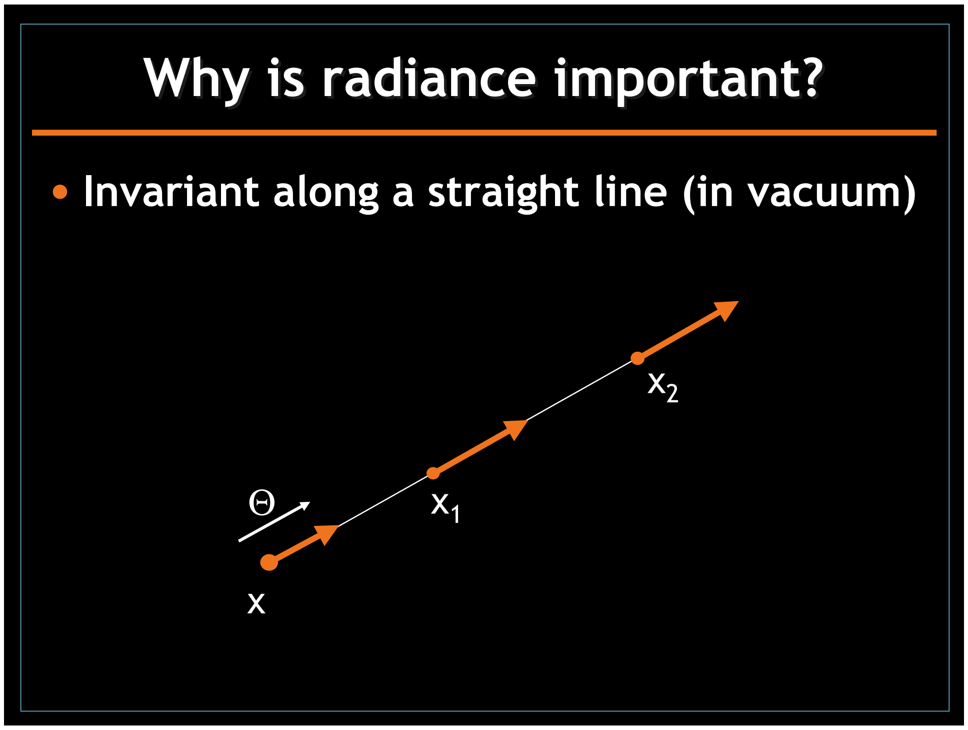
- Radiometry: Measurement of light energy
- Radiance: *radiant energy density*
  - *x is position, Θ is direction*
  - Varies with position and direction: 5D function



Radiance

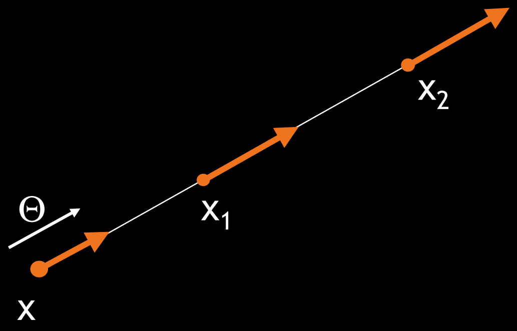
- Radiance $L(x \rightarrow \Theta)$ is the power
  - Per unit projected surface area
  - Per unit solid angle
  - $$L(x \rightarrow \Theta) = \frac{d^2 P}{dA^{\perp} d\omega_{\Theta}}$$
  - units: Watt / m^2.sr
  - Wavelength dependence
- Important quantity

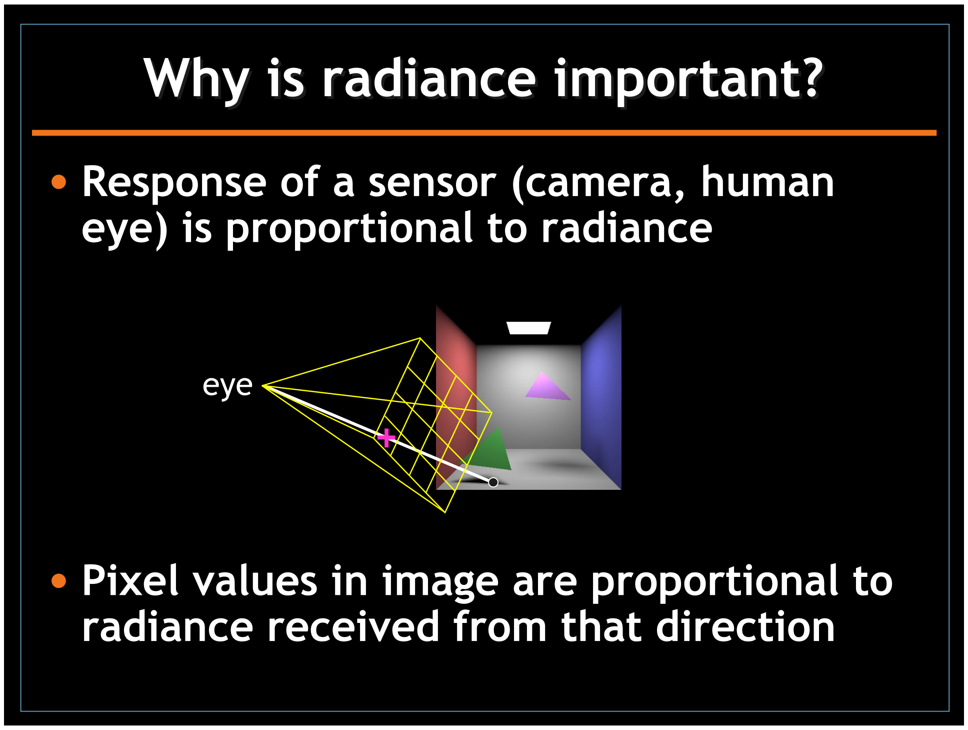




Why is radiance important?

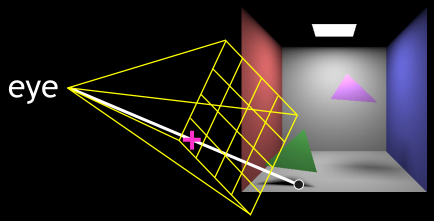
- Invariant along a straight line (in vacuum)



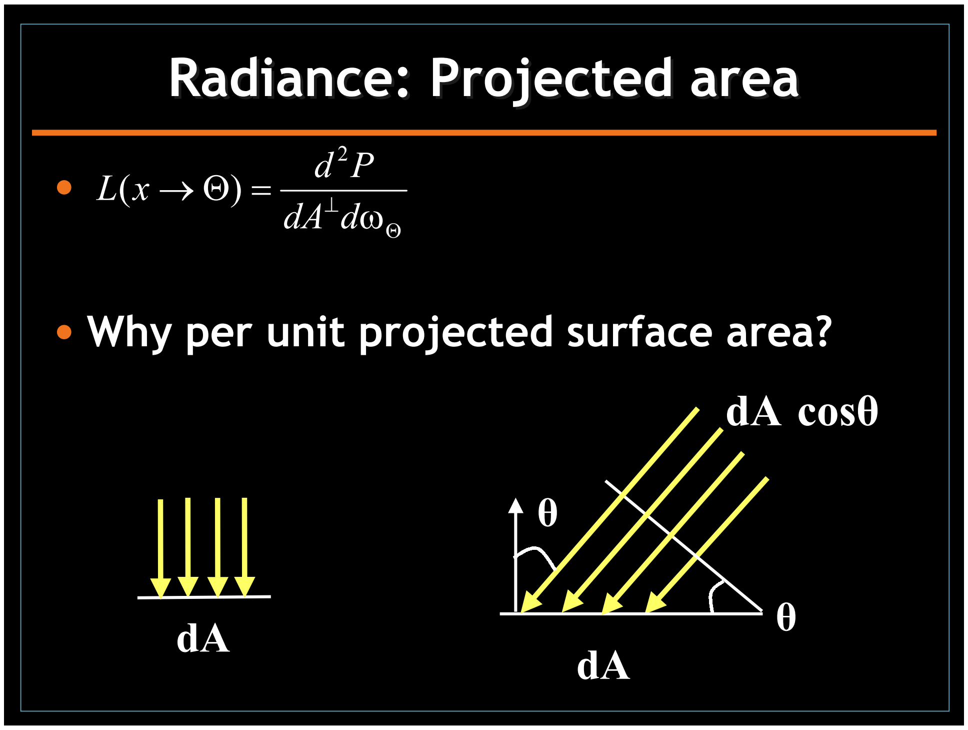


Why is radiance important?

- Response of a sensor (camera, human eye) is proportional to radiance



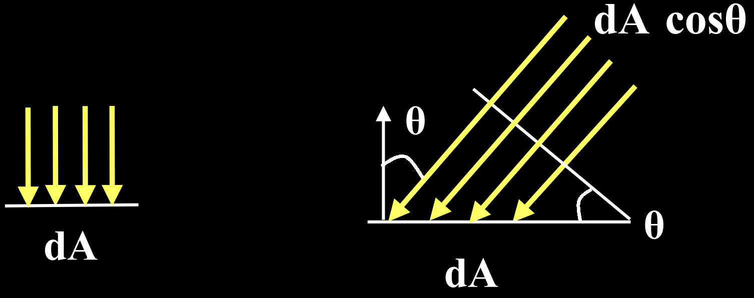
- Pixel values in image are proportional to radiance received from that direction

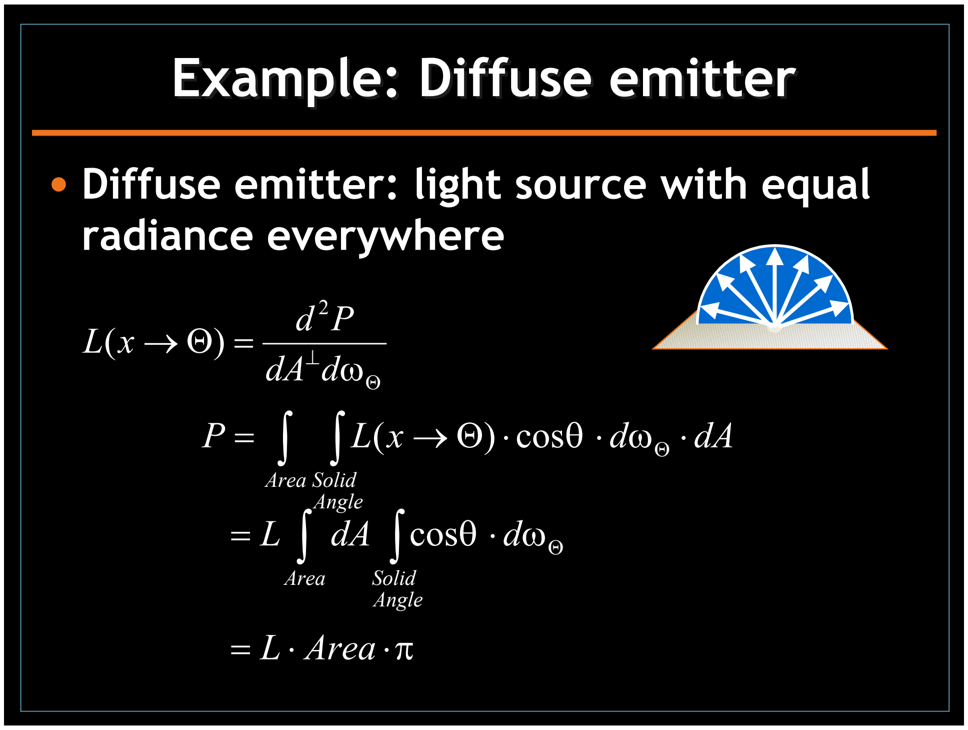


Radiance: Projected area

- $L(x \to \Theta) = \dfrac{d^2P}{dA^{\perp} d\omega_{\Theta}}$

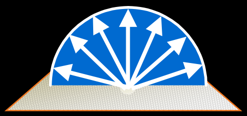
- Why per unit projected surface area?





Example: Diffuse emitter

- Diffuse emitter: light source with equal radiance everywhere

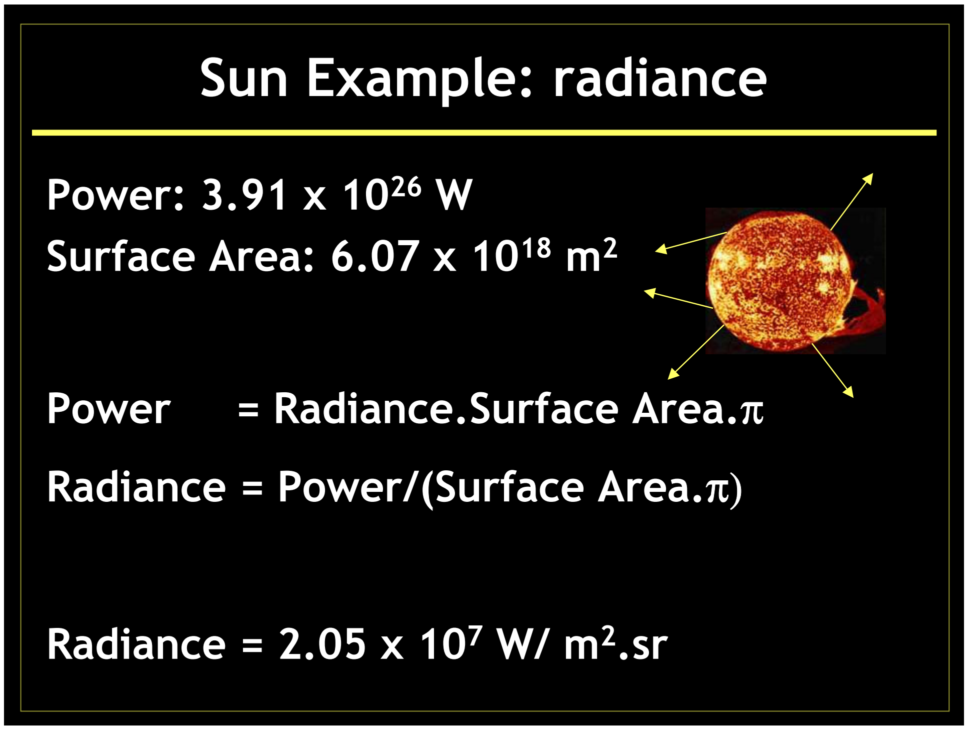


$$L(x \to \Theta) = \frac{d^2P}{dA^{\perp} d\omega_{\Theta}}$$

$$P = \int_{Area} \int_{\substack{Solid\\Angle}} L(x \to \Theta) \cdot \cos\theta \cdot d\omega_{\Theta} \cdot dA$$

$$= L \int_{Area} dA \int_{\substack{Solid\\Angle}} \cos\theta \cdot d\omega_{\Theta}$$

$$= L \cdot Area \cdot \pi$$



Sun Example: radiance

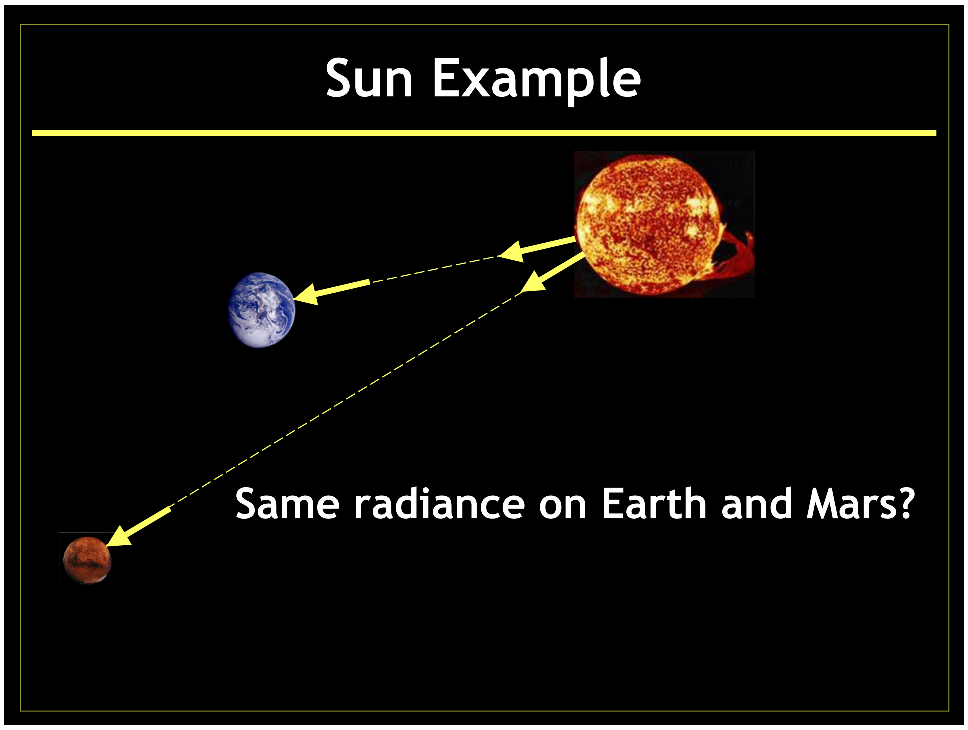
Power: 3.91×10^{26} W

Surface Area: 6.07×10^{18} m^2

Power = Radiance.Surface Area.π

Radiance = Power/(Surface Area.π)

Radiance = 2.05×10^{7} W/ m^2.sr



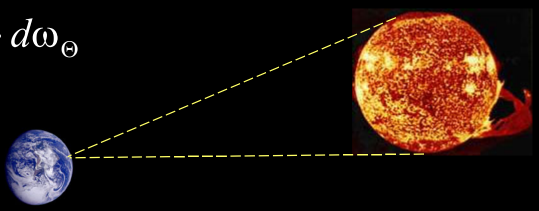
Sun Example

Same radiance on Earth and Mars?

Sun Example: Power on Earth

Power reaching earth on a 1m^2 square:

$$P = L \underbrace{\int}_{Area} dA \underbrace{\int}_{\substack{Solid\\Angle}} \cos\theta \cdot d\omega_{\Theta}$$

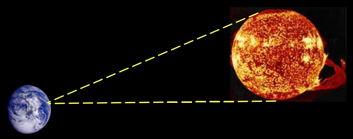


Assume $\cos\theta$ = 1 (sun in zenith)

$$P = L \underbrace{\int}_{Area} dA \underbrace{\int}_{\substack{Solid\\Angle}} d\omega_{\Theta}$$

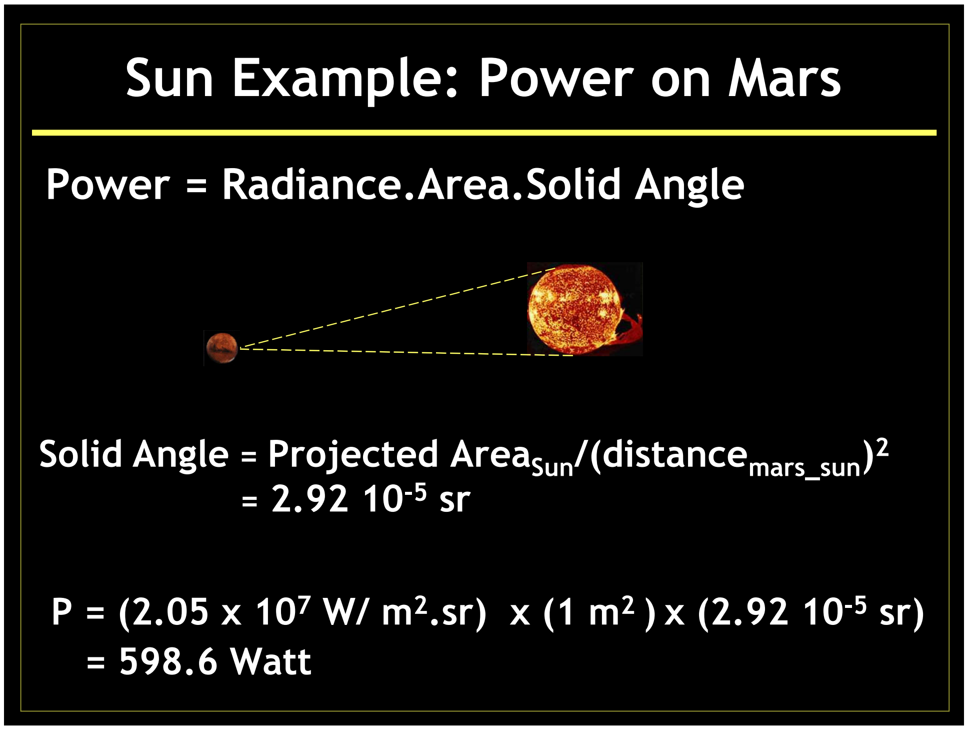
Sun Example: Power on Earth

Power = Radiance.Area.Solid Angle



Solid Angle = Projected Area$_{Sun}$/(distance$_{earth\_sun}$)^2
= 6.7 10^{-5} sr

P = (2.05 x 10^7 W/ m^2.sr) x (1 m^2) x (6.7 10^{-5} sr)
= 1373.5 Watt

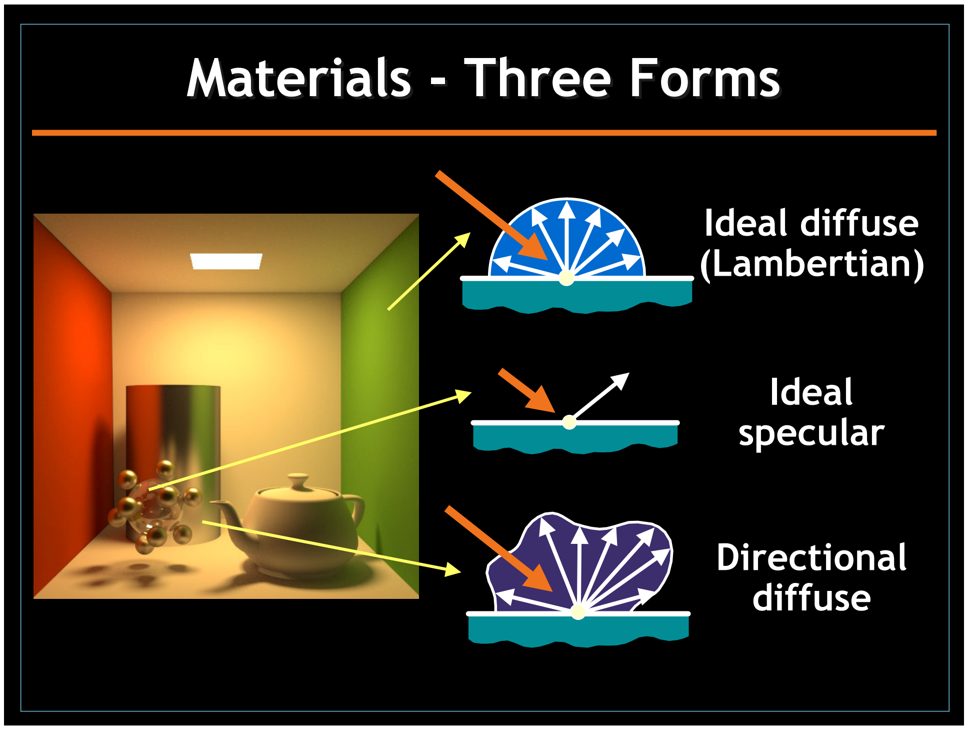


Sun Example: Power on Mars

Power = Radiance.Area.Solid Angle

Solid Angle = Projected Area$_{Sun}$/(distance$_{mars\_sun}$)^2
= 2.92 10^{-5} sr

P = (2.05 x 10^7 W/ m^2.sr) x (1 m^2) x (2.92 10^{-5} sr)
= 598.6 Watt

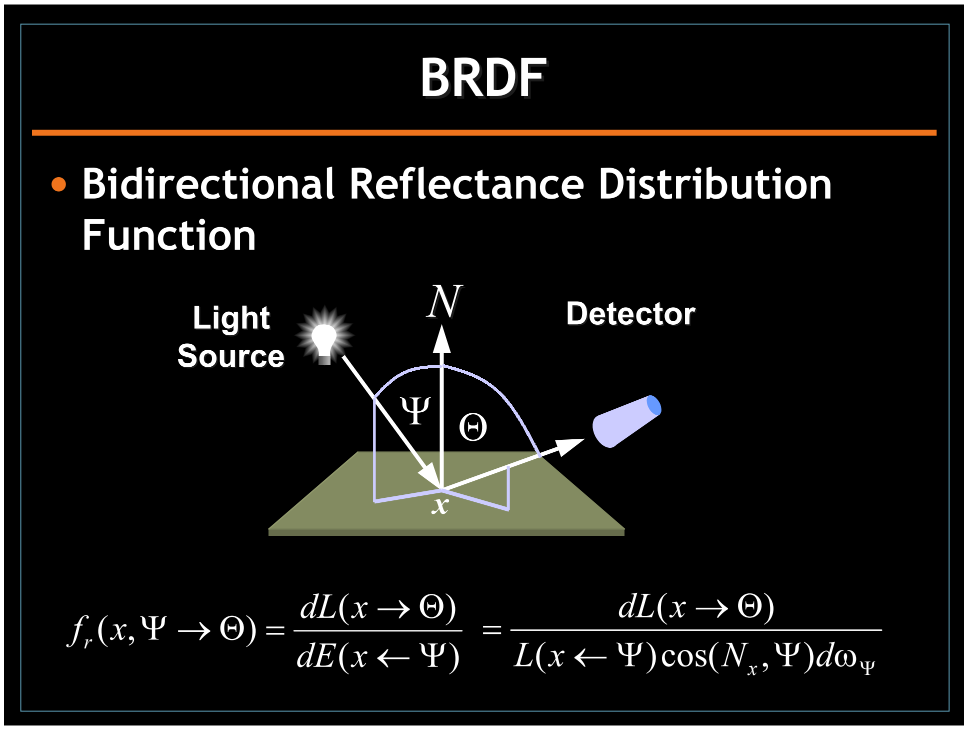


Materials - Three Forms

Ideal diffuse (Lambertian)

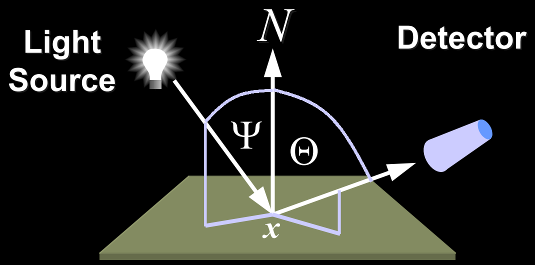
Ideal specular

Directional diffuse

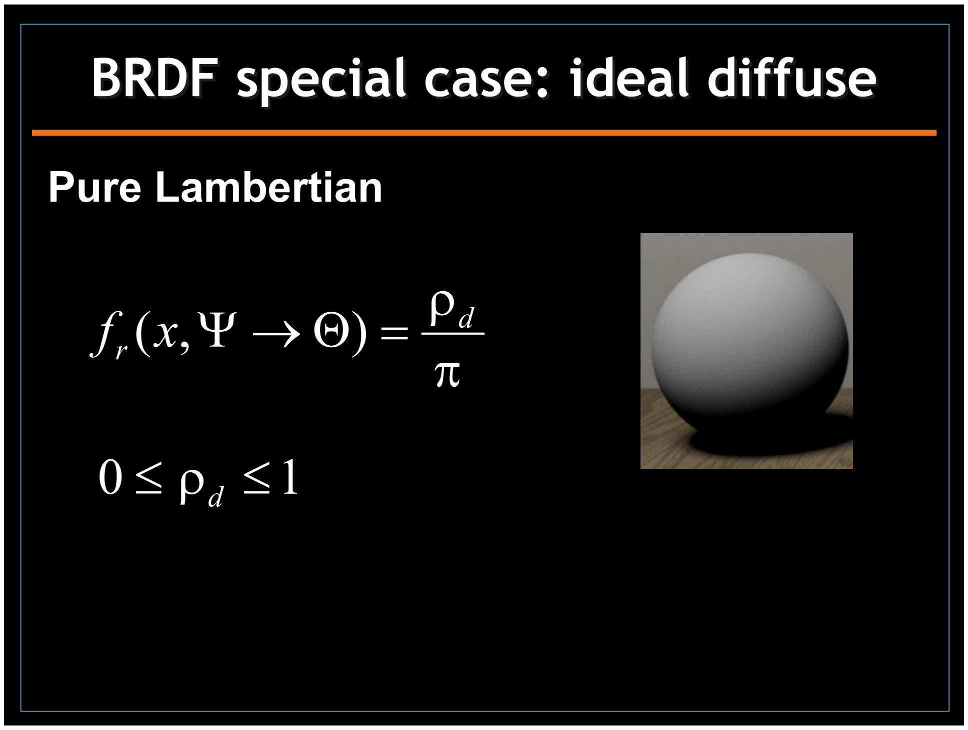


BRDF

- Bidirectional Reflectance Distribution Function



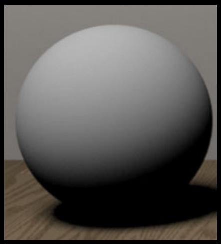
$$f_r(x,\Psi\to\Theta)=\frac{dL(x\to\Theta)}{dE(x\leftarrow\Psi)}=\frac{dL(x\to\Theta)}{L(x\leftarrow\Psi)\cos(N_x,\Psi)d\omega_\Psi}$$



BRDF special case: ideal diffuse

Pure Lambertian

$$f_r(x,\Psi\to\Theta)=\frac{\rho_d}{\pi}$$



$$0\le\rho_d\le1$$

Properties of the BRDF

- Reciprocity:

$$f_r(x,\Psi \to \Theta) = f_r(x,\Theta \to \Psi)$$

- Therefore, notation: $f_r(x,\Psi \leftrightarrow \Theta)$

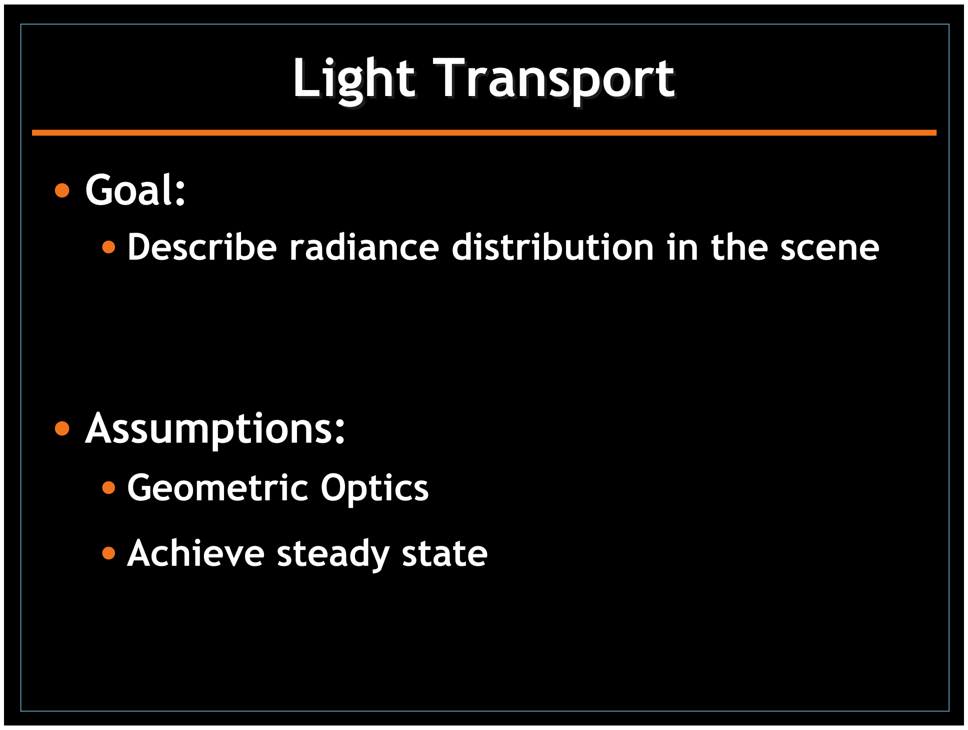
Properties of the BRDF

- Bounds:

$$0 \le f_r(x,\Psi \leftrightarrow \Theta) \le \infty$$

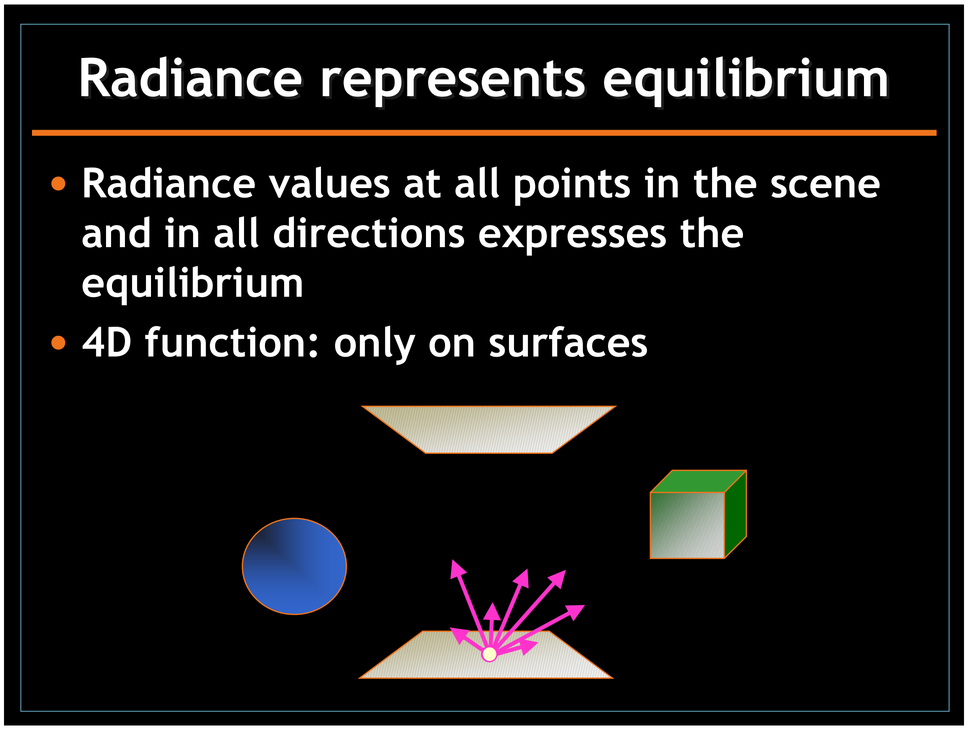
- Energy conservation:

$$\forall\Psi \quad \int_{\Theta} f_r(x,\Psi \leftrightarrow \Theta)\cos(N_x,\Theta)d\omega_{\Theta} \le 1$$



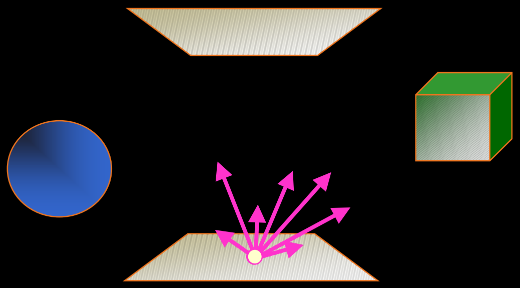
Light Transport

- Goal:
  - Describe radiance distribution in the scene
- Assumptions:
  - Geometric Optics
  - Achieve steady state



Radiance represents equilibrium

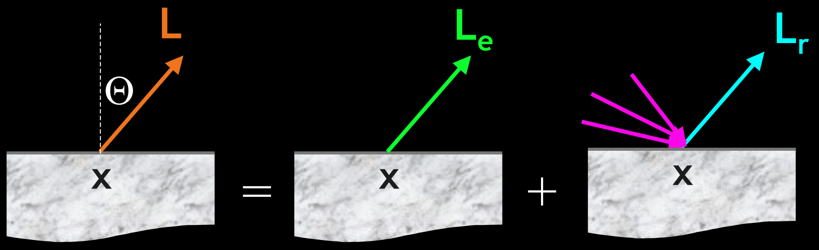
- Radiance values at all points in the scene and in all directions expresses the equilibrium
- 4D function: only on surfaces



Rendering Equation (RE)

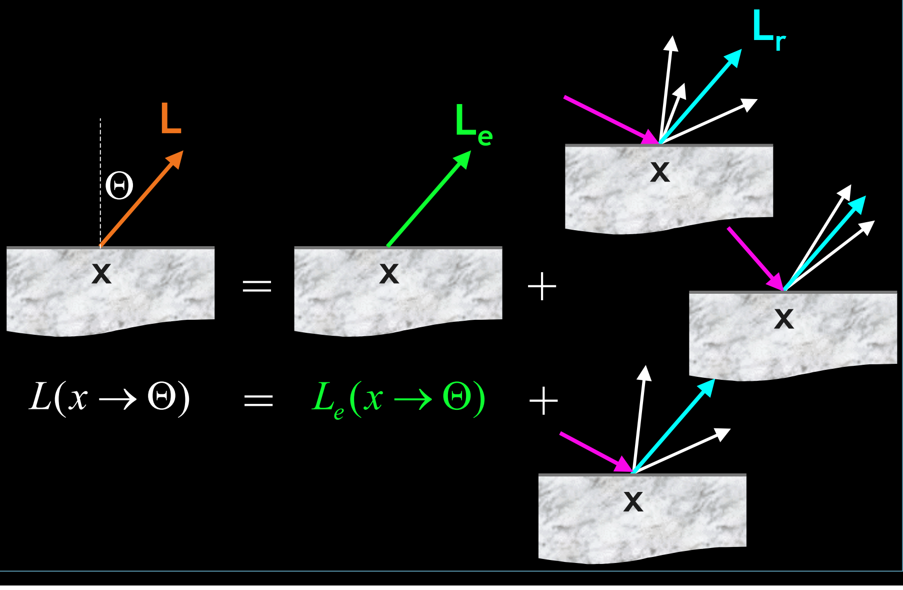
- RE describes energy transport in a scene
- Input:
  - light sources
  - geometry of surfaces
  - reflectance characteristics of surfaces
- Output: value of radiance at all surface points and in all directions

Rendering Equation



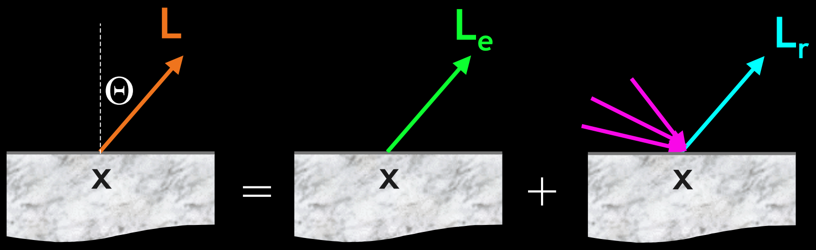
$$L(x \to \Theta) = L_e(x \to \Theta) + L_r(x \to \Theta)$$

Rendering Equation



$$L(x \to \Theta) = L_e(x \to \Theta) + $$

Rendering Equation



$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{hemisphere} L(x \leftarrow \Psi) \cdots$$

Rendering Equation

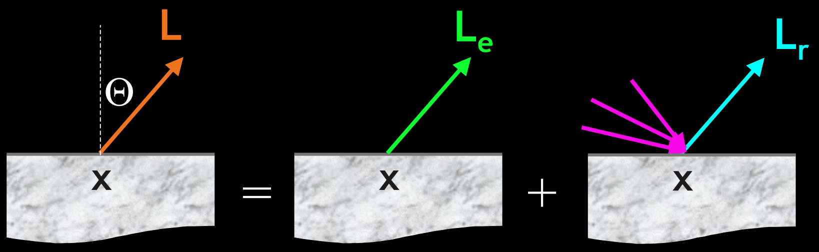
$$f_r(x,\Psi \leftrightarrow \Theta) = \frac{dL(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)}$$

$$dL(x \rightarrow \Theta) = f_r(x,\Psi \leftrightarrow \Theta)\,dE(x \leftarrow \Psi)$$

$$dL(x \rightarrow \Theta) = f_r(x,\Psi \leftrightarrow \Theta)\,L(x \leftarrow \Psi)\cos(N_x,\Psi)\,d\omega_\Psi$$

$$L_r(x \rightarrow \Theta) = \int_{hemisphere} f_r(x,\Psi \leftrightarrow \Theta)\,L(x \leftarrow \Psi)\cos(N_x,\Psi)\,d\omega_\Psi$$

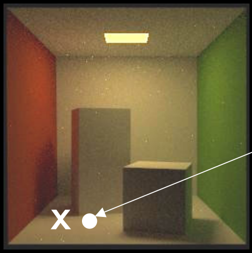
Rendering Equation



$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{hemisphere} L(x \leftarrow \Psi)\; f_r(x,\Psi \leftrightarrow \Theta)\cos(\mathbf{N_x},\Psi)\,d\omega_\Psi$$

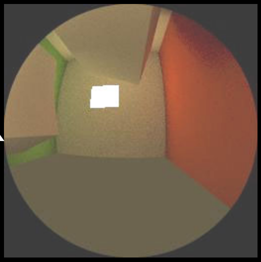
- Applicable for each wavelength

Rendering Equation



$$\underline{L(x \to \Theta)} = L_e(x \to \Theta) + \int_{hemisphere} \underline{L(x \leftarrow \Psi)} f_r(x, \Psi \leftrightarrow \Theta) \cos(\mathbf{N_x}, \Psi) d\omega_\Psi$$

incoming radiance



Summary

- Geometric Optics
- Goal:
  - to compute steady-state radiance values in scene
- Rendering equation:
  - mathematical formulation of problem

Radiometry & The Rendering Equation

Global illumination algorithms solve for the equilibrium distribution of light energy in a scene. This chapter presents key concepts and definitions required to formulate the global illumination problem. First, we give the basic assumptions that rendering algorithms make about the behavior of light. Then, we present radiometric terms and define the bidirectional reflectance distribution function (BRDF), which captures the interaction of light with surfaces. Finally, we present the rendering equation, a mathematical formulation of the problem that global illumination algorithms must solve.

1. Model of light

Light is electromagnetic radiation produced by accelerating a charge. Light can be produced in different ways; for example, by thermal sources such as the sun, or by quantum effects such as fluorescence where materials absorb energy at some wavelength and emit it at some other wavelength. There are several models that attempt to explain the behavior of light:

- Geometric Optics Model

  In this model, light is assumed to travel through transparent media along rays. This model captures effects such as emission, reflection, transmission (or refraction). This is the most commonly used model in computer graphics.

- Wave model

  The wave model is described by Maxwell's equations and captures effects that arise because light interacts with objects of size comparable to the wavelength of light. This model explains effects such as diffraction, interference, polarization and dispersion. However, these effects are typically too detailed for the purposes of image generation in computer graphics and are generally ignored.

- Quantum model

  The quantum mechanics model is the fundamental model of light that captures effects such as fluorescence and phosphorescence. However, this model is also too detailed and is generally not considered in computer graphics.

The geometric optics model is the most commonly used model in computer graphics and the model we are going to use in this course. We deal with a subset of the behavior exhibited by light: emission, reflection and transmission.

In this chapter we make several assumptions to formulate the rendering equation. We ignore the light energy that is absorbed at surfaces and dissipated as heat. We also ignore effects due to the transmission of light through participating media and media with varying indices of refraction. Additionally, we assume that light propagates instantaneously through vacuum.

2. Radiometry

Radiometry is the area of study involved in the physical measurement of light. This section gives a brief overview of the radiometric units that will be used in this course.

2.1 Radiometric Terms

Radiance Power or Flux

The fundamental radiometric quantity is **Radiant Power**, also called **Flux**. **Radiant Power**, often denoted as Φ, is expressed in *Watt* (*Joule/sec*), and expresses how much total energy flows from/to/through a surface per unit time. For example, we can say that a light source emits 100 Watts radiant power, or that 50 Watts radiant power is incident on a desk. Note that we do not specify how large the surface of the light source or desk is, nor do we specify the distance to/from the source.

Irradiance

Irradiance (E) is the *incident* radiant power on a surface, per unit surface area. It is expressed as *Watt/m*^{2}.

$$E = \frac{\Phi}{A}$$

For example, if 50 Watt radiant power is incident on a surface which has an area of 1.25 m^2, the irradiance at each surface point is 40 Watt/m^2 (if the incident power is uniformly distributed over the surface).

Radiant Exitance or Radiosity

Radiant Exitance (M), also called **Radiosity** (B), is the *exitant* radiant power per unit surface area, and is also expressed as *Watt/m*^{2}.

$$M = B = \frac{\Phi}{A}$$

For example, a light source emitting 100 Watt, which has an area of 0.1 m^2, has a radiant exitance of 1000 Watt/m^2 in each point of its surface (if the emitting power is uniform over the area of the light source).

Radiance

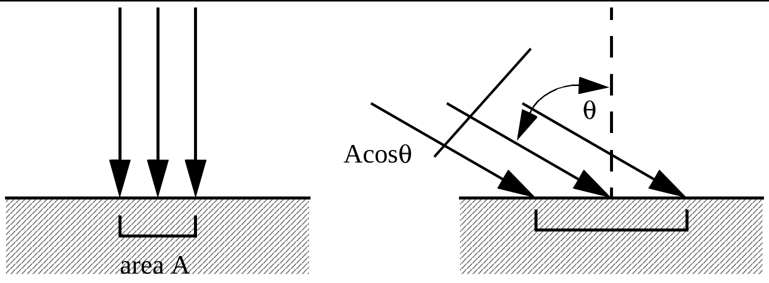
Radiance (L) is the most important quantity in radiometry. Radiance is flux per unit projected area per unit solid angle (*Watt/sr.m*^{2})

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

Intuitively, radiance expresses how much power arrives at (or leaves from) a certain point on a surface, per unit solid angle, and per unit area. But this definition does not consider the regular notion of area; rather it considers the area projected perpendicular to the direction we are interested in. Intuitively, this can be understood by considering the power that arrives at a normal incident angle. If that power were now to arrive at a grazing angle, the energy is 'smeared out' over a larger surface. Therefore, we must take the larger area into account, and that is where the cosine term comes from.

Notations:

$L(x \rightarrow \Theta)$: radiance leaving point x in direction Θ
$L(x \leftarrow \Theta)$: radiance arriving at point x from direction Θ



Energy incident on a surface: perpendicular and under angle θ.

Relationships between radiometric units

In the above definitions, we assumed finite surfaces and finite solid angles. However, we can also define radiometric quantities as continuous functions defined for each point in space and each direction (where appropriate):

Flux: $\Phi(x \rightarrow \Theta)$

Irradiance: $E(x \leftarrow \Theta) = \dfrac{d\Phi(x \leftarrow \Theta)}{dA}$

Radiant exitance or radiosity: $B(x \rightarrow \Theta) = \dfrac{d\Phi(x \rightarrow \Theta)}{dA}$

Radiance: $L(x \rightarrow \Theta) = \dfrac{d^2\Phi(x \rightarrow \Theta)}{d\omega dA^{\perp}} = \dfrac{d^2\Phi(x \rightarrow \Theta)}{d\omega dA \cos\theta}$

Reversing the above differentiations:

$$\Phi = \int_A \int_\Omega L(x \rightarrow \Theta)\cos\theta d\omega_\Theta dA$$

$$E(x) = \int_\Omega L(x \leftarrow \Theta)\cos\theta d\omega_\Theta$$

$$B(x) = \int_\Omega L(x \rightarrow \Theta)\cos\theta d\omega_\Theta$$

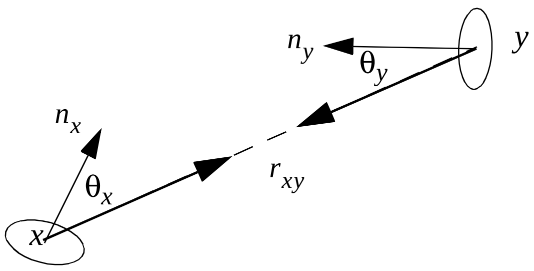
2.2 Properties of radiance

An important property of radiance is its invariance along straight paths (in vacuum). The radiance leaving point *x* directed towards point *y* is equal to the radiance arriving at point *y* from the direction in which point *x* is observed. In other words: $L(x \rightarrow y) = L(y \leftarrow x)$. This can be proved by computing the energy transport between two differential surface areas.

From the definition of radiance, the total (differential) power which is leaving differential surface area dA_x, and which is arriving at dA_y, can be written as:

$$d^2\Phi = L(x \rightarrow y)\cos\theta_x d\omega_{\overline{xy}} dA_x$$

where $\overline{xy}$ is the direction pointing from *x* to *y*, and $d\omega_{\overline{xy}}$ is the solid angle under which dA_y is seen from *x*. The power that arrives at area dA_y from area dA_x can be expressed in a similar way:



Energy transport between two differential surfaces.

$$d^2\Phi = L(y \leftarrow x)\cos\theta_y d\omega_{\overline{yx}} dA_y$$

We can also write the differential solid angles as follows:

$$d\omega_{\overline{xy}} = \frac{\cos\theta_y dA_y}{r_{xy}^2} \qquad d\omega_{\overline{yx}} = \frac{\cos\theta_x dA_x}{r_{xy}^2}$$

If we assume that no energy loss occurs between the two differential surfaces (as is the case in vacuum), and that there are no external light sources adding to the power arriving at dA_y, then from the conservation of energy, all energy that leaves the surface dA_x in the direction of the surface dA_y must arrive at the surface dA_y:

$$L(x \rightarrow y)\cos\theta_x d\omega_{\overline{xy}} dA_x = L(y \leftarrow x)\cos\theta_y d\omega_{\overline{yx}} dA_y$$

$$L(x \rightarrow y)\cos\theta_x \frac{\cos\theta_y dA_y}{r_{xy}^2} dA_x = L(y \leftarrow x)\cos\theta_y \frac{\cos\theta_x dA_x}{r_{xy}^2} dA_y$$

and thus:

$$L(x \rightarrow y) = L(y \leftarrow x)$$

So, radiance does not attenuate with distance, and is invariable along straight paths of travel. If we allow a participating medium that can absorb and scatter energy to be present between the surfaces, the above property of radiance is no longer valid.

From the above observation, it follows that once incident or exitant radiance at all surface points is known, the radiance distribution for all points in a three-dimensional scene is also known. Almost all algorithms used in global illumination limit themselves to computing the radiance values at surface points (still assuming the absence of any participating media). Radiance at surface points is referred to as surface radiance by some authors, whereas radiance for general points in three-dimensional space sometimes is called field radiance.

Another important property of radiance is that most light receivers, such as cameras or the human eye, are sensitive to radiance. The response of these sensors is proportional to the radiance incident upon them; the constant of proportionality depends on the geometry of the sensor.

Together, these two properties of radiance explain why the perceived color or brightness of an object does not change with distance.

Wavelength dependency

All of the above measures and quantities are not only dependent on position and direction, but are also dependent on the wavelength of the light energy under consideration. Thus, radiance values are normally specified for all possible wavelength values. The measures defined above are to be considered as integrated functions over the wavelength domain covering visible light. However, in papers and publications, it is often implicitly assumed that the wavelength dependency is part of the equations, and is not mentioned explicitly.

2.3 Examples

Example 1: Diffuse Emitter

A diffuse emitter, by definition, emits equal radiance in all directions from all its surface points: $L(x \to \Theta) = L$

$$\begin{aligned}
\Phi &= \int_A \int_\Omega L(x \to \Theta) \cos\theta \, d\omega_\Theta dA_x \\
&= \int_A \int_\Omega L \cos\theta \, d\omega_\Theta dA_x \\
&= L\left(\int_A dA_x\right)\left(\int_\Omega \cos\theta \, d\omega_\Theta\right) \\
&= \pi L A
\end{aligned}$$

Thus, for a diffuse surface, the radiance equals the flux divided by the area, divided by π. Using the above equations, it is straightforward to write down a relationship between the power, radiance and radiosity of a diffuse surface:

$$\Phi = LA\pi = BA$$

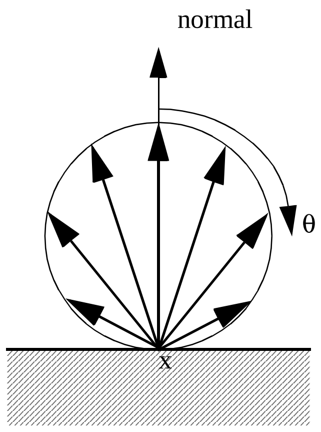
Example 2: Non-diffuse emitter

Suppose we have a square area light source with area 10cm by 10cm. Each point on the light source emits radiance according to the following distribution over its hemisphere:

$$L(x \to \Theta) = 6000 \cos\theta \quad (Watt/sr.m^2)$$

Remember that the radiance function is defined for all directions on the hemisphere, and all points on a surface. This specific distribution is equal for all points on the light source, but there is a fall-off as the direction is further away from the normal at each surface point. This figure gives a plot of the radiance in one plane, perpendicular to the surface:

The radiosity for each point can be computed as follows:



Two-dimensional plot of emitted radiance according to cosine distribution.

$$
\begin{aligned}
B(x) &= \int_\Omega L(x \to \Theta)\cos\theta \, d\omega_\Theta \\
&= \int_\Omega 6000\cos^2\theta \; d\omega_\Theta \\
&= 6000\int \cos^2\theta \, d\omega_\Theta \\
&= 6000\int_0^{2\pi}\int_0^{\pi/2} \cos^2\theta \; \sin\theta \, d\theta \, d\varphi \\
&= 6000 \cdot 2\pi \cdot \left[-\frac{\cos^3\theta}{3}\right]_0^{\pi/2} \\
&= 4000\pi \text{ Watt/m}^2 = 12566 \text{ Watt/m}^2
\end{aligned}
$$

The power for the whole light source can be computed as follows:

$$
\begin{aligned}
\Phi &= \int_A\int_\Omega L(x \to \Theta)\cos\theta \, d\omega_\Theta \, dA_x \\
&= \int_A\left(\int_\Omega L(x \to \Theta)\cos\theta \, d\omega_\Theta\right) dA_x \\
&= \int B(x) \, dA_x \\
&= 4000\pi \text{ Watt/m}^2 \cdot 0.1 \text{ m}^2 \cdot 0.1 \text{ m}^2 \\
&= 125.66 \text{ Watt}
\end{aligned}
$$

Example 3

Consider the radiance output from the Sun arriving at the Earth and Mars. Assume the sun is a uniform diffuse emitter. Using the equation from Example 1:

$$\Phi = LA\pi$$

The total power emitted by the sun is 3.91(10^{26}) Watt, and the surface area of the sun is 6.07(10^{18}) m^2. Therefore, the radiance equals:

$$L = \frac{\Phi}{A\pi} = \frac{3.91(10^{26})}{\pi 6.07(10^{18})} = (2.05(10^7))$$

Given a 1m x 1m patch on the surface of the earth, the power arriving at that patch is given as:

$$P = \int_A \int_\Omega L \cos\theta d\omega dA$$

Assume that the sun is at its zenith, therefore, $\cos\theta = 1$

$$P = AL\omega_{solidangle}$$

The solid angle subtended by the sun as seen from Earth is:

$$\omega_{solidangle} = \frac{A}{dis\tan ce} = 10^{-5}(6.7)\text{ sr}$$

So the total power incident on the patch equals:

$$P = (1x1)(2.05(10^7))(6.7(10^{-5})) = 1373.5W$$

Given a 1m x 1m patch on the surface of Mars, the power arriving at that patch can be computed in the same way. The solid angle subtended by the Sun as seen from Mars equals:

$$\omega_{solidangle} = \frac{A}{dis\tan ce} = 10^{-5}(2.92)\text{ sr}$$

Therefore the power incident on a patch on Mars is given by:

$$P = (1x1)(2.05(10^7))(2.92(10^{-5})) = 598.6W$$

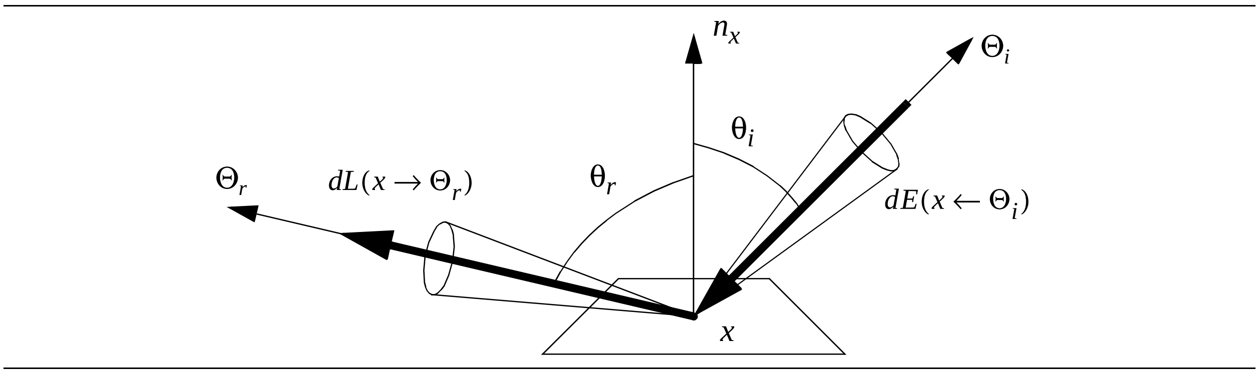
Thus, even though the radiance of the sun is invariant along rays and does not drop off with distance, the solid angle measure ensures that the power arriving at the Earth and Mars *does* drop off with distance squared.

3. The Bidirectional Reflectance Distribution Function (BRDF)

Materials interact with light in different ways, and different materials have different appearances given the same lighting conditions. Some materials appear as mirrors, others appear as diffuse surfaces. The reflectance properties of a surface are described by a reflectance function, which models the interaction of light reflecting at a surface.

The bidirectional reflectance distribution function (BRDF) is the most general expression of reflectance of a material, at least at the level of detail we wish to consider. The BRDF is defined as the ratio between differential radiance reflected in an exitant direction, and incident irradiance through a differential solid angle; or more precisely, the BRDF is defined as the derivative of reflected radiance to incident irradiance.

$$f_r(x, \Theta_i \to \Theta_r) = \frac{dL(x \to \Theta_r)}{dE(x \leftarrow \Theta_i)} = \frac{dL(x \to \Theta_r)}{L(x \leftarrow \Theta_i)\cos\theta_i d\omega_{\Theta_i}}$$



Geometry for the BRDF

The BRDF has some interesting properties:

1. The BRDF can take any positive value, and varies with wavelength.

   The value of the BRDF will remain unchanged if the incident and exitant directions are interchanged. This property is also called Helmholtz reciprocity, a principle which says that paths followed by light can be reversed.

   $$f_r(x, \Theta_i \to \Theta_r) = f_r(x, \Theta_r \to \Theta_i)$$

   Because of the reciprocity property, we will use a double arrow to indicate the fact that the two directions may be freely interchanged: $f_r(x, \Theta_i \leftrightarrow \Theta_r)$.

2. Generally, the BRDF is anisotropic. That is, if the surface is rotated about the surface normal, the value of f_r will change. However, there are many materials which are isotropic and where the value of f_r does not depend on the specific orientation of the underlying surface.

3. The value of the BRDF for a specific incident direction is not dependent on the possible presence of irradiance along other incident angles. Therefore, the BRDF, as defined above, behaves as a linear function with respect to all incident directions. In order to know the total reflected radiance due to some irradiance distribution over the hemisphere around an opaque, non-emissive surface point, we have to integrate the BRDF equation over the surrounding hemisphere. This provides us with the following equation, referred to as the reflectance equation:

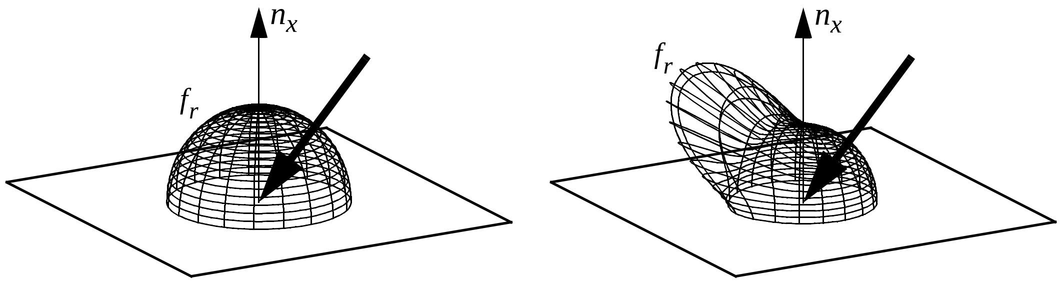
   $$dL(x \to \Theta_r) = f_r(x, \Theta_i \to \Theta_r) dE(x \leftarrow \Theta_i)$$

   $$L(x \to \Theta_r) = \int_{\Omega_x} f_r(x, \Theta \leftrightarrow \Theta_r) dE(x \leftarrow \Theta)$$

   $$L(x \to \Theta_r) = \int_{\Omega_x} f_r(x, \Theta \leftrightarrow \Theta_r) L(x \leftarrow \Theta) \cos(n_x, \Theta) d\omega_\Theta$$

   where $\cos(n_x, \Theta)$ is the cosine of the angle formed by the vectors n_x and Θ.

Depending on the nature of the BRDF, the material will appear as a diffuse surface, or a mirror, or a glossy surface, or it may exhibit any behaviour described by the BRDF. The more common encountered types of BRDF, as used in photo-realistic rendering, are listed below:



Diffuse and glossy surfaces

Diffuse surfaces

Some materials reflect light in a uniform way over the entire reflecting hemisphere. That is, given an irradiance distribution, the reflected radiance is independent of the exitant direction. Such materials are called diffuse reflectors, and the value of their BRDF is constant for all values of Θ_r. To an observer, a diffuse material looks the same from all possible directions. For a pure Lambertian surface:

$$f_r(x, \Theta_i \to \Theta_r) = \frac{\rho_d}{\pi}$$

The reflectance ρ_d represents the fraction of incident energy that is reflected at the surface. For physically based materials, ρ_d varies from 0 to 1.

Specular surfaces

Other materials can be considered as perfectly specular surfaces and only reflect light in one specific direction. According to Snell's law the incident and exitant direction make equal angles to the surface's normal. The BRDF of a perfectly specular surface can be described with the proper use of δ-functions. A perfect specular surface has only one exitant direction for which the BRDF is different from 0, which implies that the value of the BRDF along that direction is infinite.

Glossy surfaces

Most surfaces, however, are neither ideally diffuse nor ideally specular, but exhibit a combination of both reflectance behaviors; these surfaces are called glossy surfaces. Their BRDF is often difficult to model with analytical formulae.

Transparent surfaces

Strictly speaking, the BRDF is defined over the entire sphere of directions (4π steradians) around a surface point. This is important for transparent surfaces, since these surfaces can 'reflect' light over the entire sphere. The 'transparent' side of the BRDF can also behave as a diffuse, specular or glossy surface, depending on the transparency characteristics of the material. In this text, we will limit ourselves to non-transparent surfaces, and to BRDFs defined only over the reflecting side of the sphere. However, one has to be careful when assuming properties about the transparent side of the BRDF. Some characteristics, such as the reciprocity condition specified earlier, may not be true with transparent surfaces. In most texts, the term BSDF (Bidirectional Scattering Function) is used, to denote the reflection and transparent parts together.

In global illumination algorithms, one often uses empirical models to characterize the BRDF. Great care must be taken to make certain that these empirical models indeed make up a good and acceptable BRDF. More specifically, the following conditions must be met to make the empirical model physically plausible:

- Due to the conservation of energy, the total amount of power reflected over all directions must be less than or equal to the total amount of power incident on the surface (excess power is transformed into heat or other forms of energy). For any distribution of incident radiance $L(x \leftarrow \Psi)$ over the hemisphere, the total incident power per surface area is the total irradiance over the hemisphere:

  $$E = \int_{\Omega_x} L(x \leftarrow \Psi) \cos(n_x, \Psi) d\omega_\Psi$$

  The total reflected power M is a double integral over the hemisphere: suppose we have a distribution of exitant radiance $L(x \to \Theta)$ at a surface. The total power per unit surface area leaving the surface is:

  $$M = \int_{\Omega_x} L(x \to \Theta) \cos(n_x, \Theta) d\omega_\Theta$$

  From the definition of the BRDF we know:

  $$dL(x \to \Theta) = f_r(x, \Psi \leftrightarrow \Theta) L(x \leftarrow \Psi) \cos(n_x, \Psi) d\omega_\Psi$$

  Integrating this equation to find the value for $L(x \to \Theta)$ and combining it with the expression for M gives us:

  $$M = \int_{\Omega_x}\int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) L(x \leftarrow \Psi) \cos(n_x, \Theta) \cos(n_x, \Psi) d\omega_\Psi d\omega_\Theta$$

  The BRDF satisfies the constraint of energy conservation for reflectance at a surface point if, for all possible incident radiance distributions $L(x \leftarrow \Psi)$, the following inequality holds:

  $$M \le E \quad \text{or} \quad \frac{\int_{\Omega_x}\int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) L(x \leftarrow \Psi) \cos(n_x, \Theta) \cos(n_x, \Psi) d\omega_\Psi d\omega_\Theta}{\int_{\Omega_x} L(x \leftarrow \Psi) \cos(n_x, \Psi) d\omega_\Psi} \le 1$$

  This inequality must be true for any incident radiance function. Suppose we take an appropriate δ-function for the incident radiance distribution, such that the integrals become simple expressions:

  $$L(x \leftarrow \Psi) = L_{in} \delta(\Psi - \Theta)$$

  then the above equation can be simplified to:

  $$\forall \Psi: \quad \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) \cos(n_x, \Theta) d\omega_\Theta \le 1$$

  This is a necessary condition for energy conservation, since it expresses the inequality for a specific incident radiance distribution. It is also a sufficient condition, since incident radiance from two different directions does not influence the value of the BRDF, and thus conservation of energy is valid for any combination of incident radiance values. If the value of the BRDF is dependent on the intensity of the incoming light, one has to check the more elaborate inequality. A BRDF which is dependent on the value of incident radiance is not uncommon at all. Light striking a surface changes the temperature of that surface, and the perceived color of an object depends on its temperature. Such behavior of BRDFs is usually not considered in photo-realistic rendering.

- The empirical model for the BRDF must also obey Helmholtz reciprocity. This is an important constraint for some algorithms, especially those that compute the distribution of light energy by considering paths starting from the light sources and paths starting from the observer at the same time. Such algorithms explicitly assume that light paths can be reversed, and therefore the model for the BRDF should reflect this property.

Glassner[Glas95] presents an overview of several BRDF models used in computer graphics. The most commonly used model is the Phong model which is computationally efficient, but is not physically based since it does not satisfy the energy conservation property described above. To date the most comprehensive model is the He[He92] model which includes effects such as subsurface scattering and surface anisotropy; however, it is computationally very inefficient. Instead, people use models such as the Cook-Torrance[Cook81] which is physically based and uses microfacets to explain the reflectance behavior of light, or the Ward model [Ward92] which is a popular empirically based model.

4. The Rendering Equation (RE)

This section presents the rendering equation, a mathematical formulation of the steady-state distribution of energy in a scene with no participating media. As mentioned before, we assume that this steady-state equilibrium distribution of light energy is achieved instatantaneously.

The rendering equation specifies the outgoing radiance at a point x in a direction Θ: $L(x \to \Theta)$ in terms of the emitted radiance at that point, $L_e(x \to \Theta)$, and the reflected radiance at that point $L_r(x \to \Theta)$. From the conservation of energy,

$$L(x \to \Theta) = L_e(x \to \Theta) + L_r(x \to \Theta)$$

From the definition of the BRDF we have

$$\begin{aligned} dL_r(x \to \Theta) &= f_r(x, \Psi \leftrightarrow \Theta)dE(x \leftarrow \Psi) \\ L_r(x \to \Theta) &= \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta)dE(x \leftarrow \Psi) \\ L_r(x \to \Theta) &= \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta)L(x \leftarrow \Psi)\cos(n_x, \Psi)d\omega_\Psi \end{aligned}$$

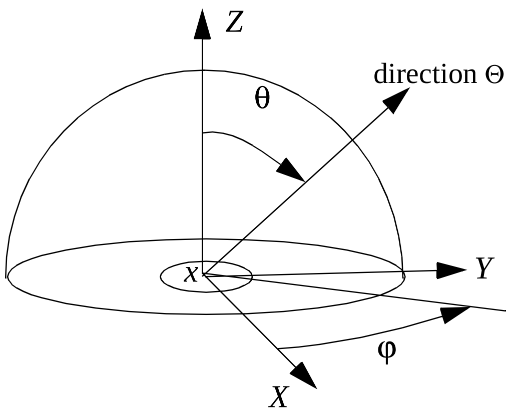
Therefore, one formulation of the rendering equation is:

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta)L(x \leftarrow \Psi)\cos(n_x, \Psi)d\omega_\Psi$$

Spherical Geometry & Coordinates

1. General Concepts

In rendering, we often want to work with functions defined over a hemisphere (one-half of a sphere). A hemisphere is a two-dimensional surface, where each point on the surface defines a direction. Spherical coordinates parametrize the hemisphere such that mathematical operations become possible. In spherical coordinates, each direction is represented by two angles. The first angle, φ, indicates the azimuth, the second angle, θ, indicates the elevation. Using the notation that capital Greek letters represent directions, we can say that direction $\Theta = (\varphi, \theta)$.



Hemispherical coordinates.

The range of the angles is given by:

$$\varphi \in [0, 2\pi]$$
$$\theta \in [0, \pi/2]$$

θ is measured starting from the normal vector at point *x* (the direction perpendicular to the surface on which *x* is located), φ is measured w.r.t an arbitrary axis located in the tangent plane to the surface.

So far we have defined points - or directions - on the hemisphere. If we want full spherical coordinates, where we can specify every point in space (and not only points on the hemisphere), we not only need a direction, but also a distance *r* along this direction. A point is then defined by three coordinates (φ, θ, r). Transforming between Cartesian and full spherical coordinates is straightforward using elementary trigonometry:

$$x = r\cos\varphi\sin\theta$$
$$y = r\sin\varphi\sin\theta$$
$$z = r\cos\theta$$

or:

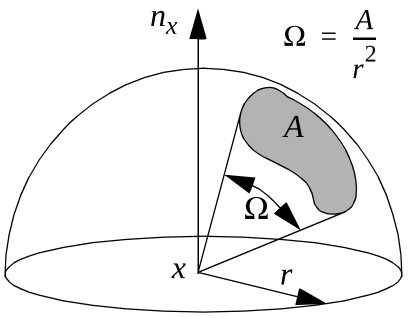
$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\tan\varphi = y/x$$
$$\tan\theta = \frac{\sqrt{x^2 + y^2}}{z}$$

Rendering algorithms typically integrate functions over directions incident at a given surface point; therefore, these algorithms usually use hemispherical coordinates without the distance parameter.

2. Solid Angle

If we want to integrate functions over the hemisphere (e.g. we want to compute all incident light at a surface point), we need a measure on the hemisphere. This measure is the solid angle.

A finite solid angle is defined as follows: measure the given area on the sphere, and divide by the radius squared. In the figure below, solid angle Ω equals its subtended area *A* divided by the radius squared. If $r = 1$, the solid angle is simply the surface area on the hemisphere.



Finite solid angle.

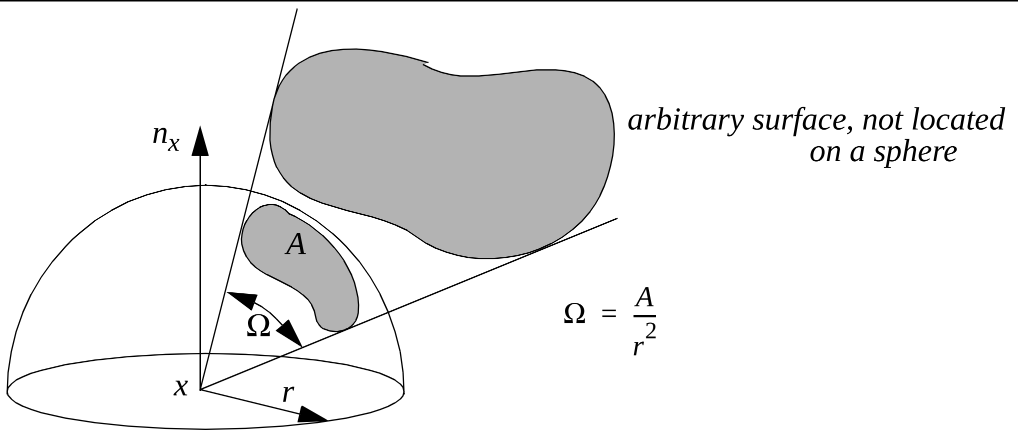
The concept of solid angle is completely analogous to angles in 2D. Since the area of a hemisphere equals $2\pi r^2$, the solid angle covered by the entire hemisphere is 2π; the solid angle covered by a complete sphere is 4π. Solid angles are dimensionless, but are expressed in steradians (sr). Note that the solid angle is not dependent on the shape of surface *A*. To compute the solid angle subtended by an arbitrary surface or object in space, seen from a specific point, we first project that surface on the hemisphere centered at that point, and compute the solid angle of the projection.

The solid angle is only dependent on the area of the projection, and does not depend on the shape. Therefore, two objects with different shapes can still subtend the same solid angle.

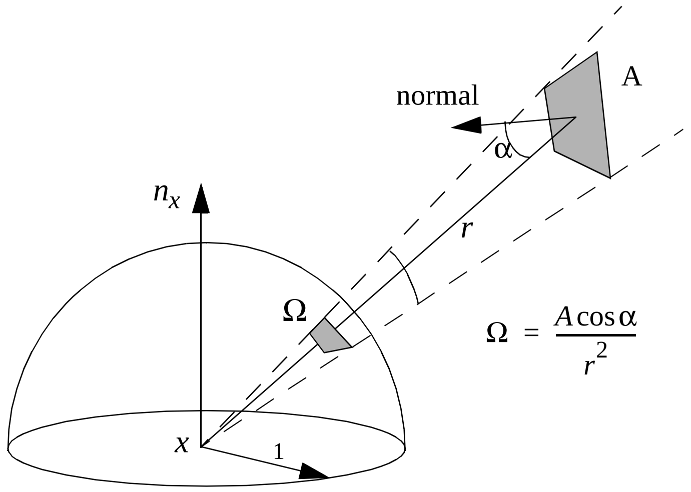
For small surfaces, we can use an approximation to compute the solid angle subtended by a surface or object: $A\cos\alpha$ is called the *projected surface area*. It is an approximation of the projected area of *A* on a hemisphere.

3. Integration over hemispheres

Just as differential surface areas or differential volumes can be defined in Cartesian XY or XYZ space to integrate functions, we can define differential solid angles to integrate functions in hemispherical space.



Finite solid angle subtended by an arbitrary surface.



Approximation of the solid angle subtended by small surfaces.

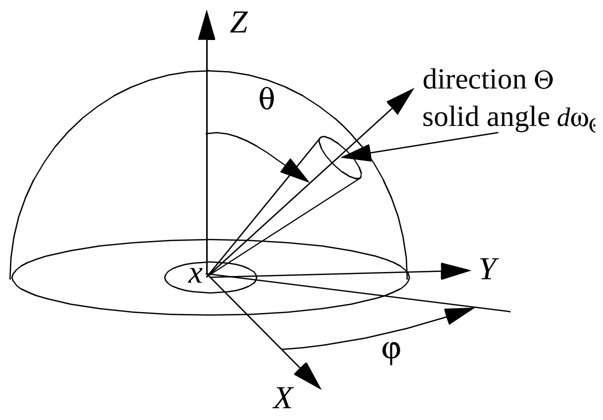
However, unlike differential areas/volumes in Cartesian space, there is a complication that arises in hemispherical space: for a constant θ near the horizon, the ‘area’ on the hemisphere ‘sweeped’ out by some differential $d\varphi$ is much larger than the area sweeped out for a constant θ near the pole. The differential solid angle takes this into account, by using a compensating factor $\sin\theta$.

A differential solid angle, centered around direction Θ, is written as: $d\omega_\Theta = \sin\theta d\theta d\varphi$

Integrating a function $f(\Theta) = f(\varphi, \theta)$ over the hemisphere can then be written as:

$$\int_\Omega f(\Theta)d\omega_\Theta = \int_0^{2\pi}\int_0^{\pi/2} f(\varphi, \theta)\sin\theta d\theta d\varphi$$

E.g. 1: Computing the area of the hemisphere:



Differential solid angle on the hemisphere.

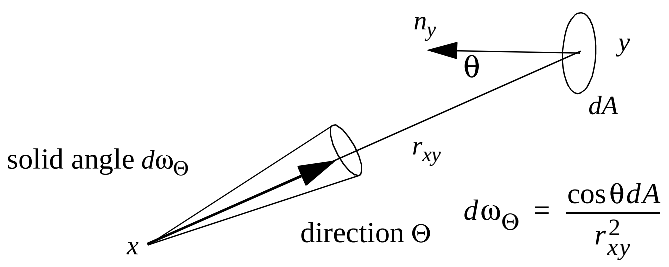
$$\begin{aligned}\int_{\Omega} d\omega_\Theta &= \int_0^{2\pi} d\varphi \int_0^{\pi/2} \sin\theta d\theta \\ &= \int_0^{2\pi} d\varphi[-\cos\theta]_0^{\pi/2} \\ &= \int_0^{2\pi} 1.d\varphi \\ &= 2\pi\end{aligned}$$

E.g. 2: Integrating a cosine distribution over the hemisphere

$$f(\Theta) = f(\varphi,\theta) = \cos\theta$$

$$\begin{aligned}\int_{\Omega} \cos(\Theta, n_x) d\omega_\Theta &= \int_0^{2\pi} d\varphi \int_0^{\pi/2} \cos\theta \sin\theta d\theta \\ &= \int_0^{2\pi} d\varphi\left[-\frac{\cos^2\theta}{2}\right]_0^{\pi/2} \\ &= \int_0^{2\pi} \frac{1}{2} d\varphi \\ &= \frac{2\pi}{2} \\ &= \pi\end{aligned}$$

In rendering, we will often make use of a transformation that switches between integrals over the hemisphere and integrals over surfaces. For example, if we want to compute all incident light at a point due to a distant light source, we can integrate over all directions within the total solid angle subtended by the light source, or we can integrate over the actual area of the light source. In order to carry out this transformation, we have to know what the relationship is between a differential solid angle and a differential surface area on a distant surface. We use the approximation for small surfaces, which is the exact formula in the case of differential surfaces:


n_y
θ
y
dA
solid angle $d\omega_\Theta$
r_{xy}
$d\omega_\Theta = \frac{\cos\theta dA}{r_{xy}^2}$
x
direction Θ

Solid angle - differential area transformation.

The differential solid angle $d\omega_\Theta$ around direction Θ is transformed to a differential surface dA around surface point *y*. Therefore, any integral over the hemisphere can also be written as an integral over surfaces visible in each direction as follows:

$$\int_\Omega f(\Theta) d\omega_\Theta = \int_A f(y) \frac{\cos\theta}{r_{xy}^2} dA$$

Monte Carlo Integration

1. Terms and definitions

A *random variable* describes the possible outcomes of an experiment. Associated with a random variable y is a *probability distribution function* $F(y)$. This function gives the probability with which an event occurs with an outcome lower than or equal to the value of y.

$$F(y) = \text{probability of random variable} \le y$$

$F(y)$ is a non-decreasing function, and is non-negative over the entire domain of the random variable. Associated with a probability distribution function $F(y)$ is a *probability density function* (PDF) $f(y)$. If $F(y)$ is continuous over the domain of the random variable, the PDF can be written as the derivative of the distribution. Given a PDF, we can always construct the corresponding probability distribution function by integrating the PDF:

$$F(y) = \int_{-\infty}^{y} f(x)dx$$

Intuitively, one can say that $f(x)dx$ is the probability that the value of the random variable will be equal to x. A PDF has the following properties:

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

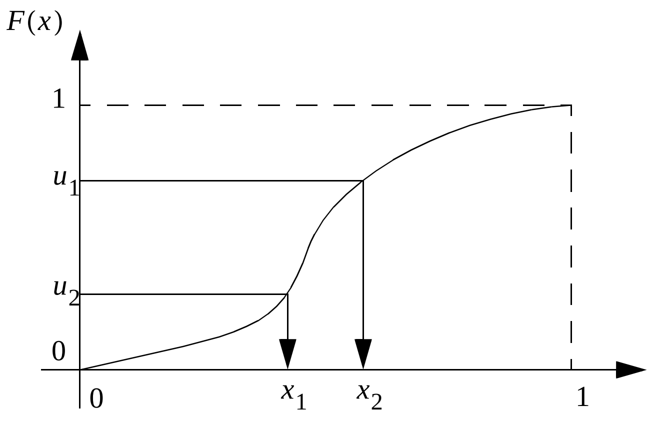
$$\forall x: \quad p(x) \ge 0$$

The probability of the event for which the value of the random variable is situated between two values a and b is given by:

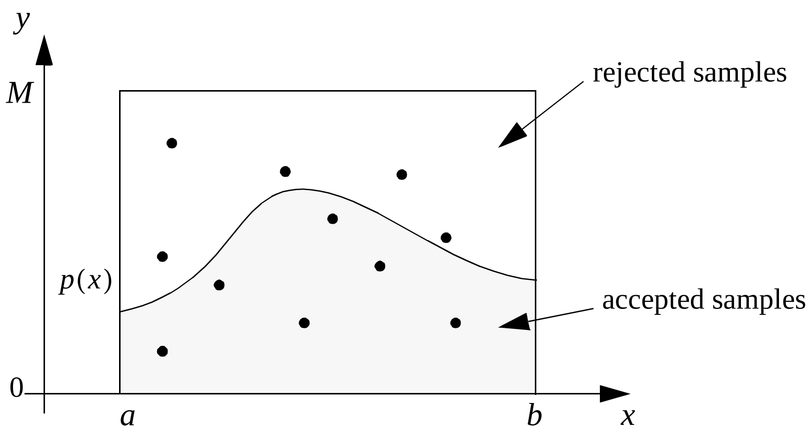
$$P[a < y \le b] = \int_{a}^{b} p(x)dx$$

A sample can be generated according to a given distribution by applying the inverse cumulative distribution function to a uniformly generated random variable u over the $[0, 1]$ interval. The resulting sample is then computed as $F^{-1}(u)$. This is a well-known method of generating random samples, and is graphically represented in the figure below. This method can only be used however, when the inverse of the probability distribution function is known.

In most cases, it is not possible to derive an analytical formula for the inverse of the cumulative distribution function. An alternative is to use rejection sampling. This technique raises the dimension of the sampled particles by one, and generates uniform sample points over a bounding box which totally encloses the PDF. For a one-dimensional PDF, whose maximum value over the domain $[a, b]$ to be sampled is M, a pair (x, y) is uniformly generated over the rectangle $[a, b] \times [0, M]$. In order to become a distribution for x according to the PDF $p(x)$, the sampled pair is rejected if $p(x) < y$, and is accepted otherwise. The distribution of the accepted samples x is equal to the PDF $p(x)$.



Generating sample points according to *F*(x).



Rejection sampling.

Still another alternative is to use a numerical table, which is constructed by approximating the distribution function by a piecewise linear function. This method, however, is not widely used, except perhaps when the PDF to be sampled is extracted from measured data.

The *expected value* of a random variable is the mean value obtained by performing an infinite number of experiments. Mathematically, the expected value of a random variable x ($E[x]$) with PDF $p(x)$ can be expressed as:

$$E[x] = \int_{-\infty}^{\infty} xp(x)dx$$

More generally, if we want to compute the expected value of a function of a random variable:

$$E[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)dx$$

The *variance* of a random variable is a measure of the mean square deviation from the expected value. Variance is defined as the expected value of the squared distance between the outcome of an experiment and its expected value:

$$\sigma^2[f(x)] = E[(f(x) - E[f(x)])^2]$$

A well-known expression for the variance can easily be derived:

$$\sigma^2[f(x)] = E[f(x)^2] - E[f(x)]^2$$

The definitions and terms used for probability functions can easily be expanded to higher dimensions.

2. Basic Monte Carlo Integration

Suppose we want to numerically integrate a function $f(x)$ over an integration domain *D*, i.e., we want to compute the value of the integral *I*:

$$I = \int_D f(x)dx$$

$$D = [\alpha_1 \ldots \beta_1] \times [\alpha_2 \ldots \beta_2] \times \ldots \times [\alpha_d \ldots \beta_d] \qquad (\alpha_i, \beta_i \in \Re)$$

where *d* is the dimension of the integration domain. Deterministic quadrature formulas would construct a number of sample points, and use the function values at those points to compute an estimate of *I*. Monte Carlo integration basically uses the same approach, but uses a stochastic process to generate the sample points. It is easy to imagine that we are able to generate *N* sample points x_i $(i = 1, \ldots, N)$ distributed uniformly over the domain *D*. The mean of the evaluated function values $f(x_i)$ multiplied by the area of the integration domain, provides us with an unbiased estimator for *I*:

$$\langle I \rangle = \left(\frac{1}{N} \sum_{i=1}^{N} f(x_i) \right) \cdot \left(\prod_{i=1}^{d} (\beta_i - \alpha_i) \right)$$

An estimator is said to be unbiased if its expected value equals the exact result of the expression we want to estimate by means of the stochastic process. The above estimator $\langle I \rangle$ is indeed unbiased, because its expected value equals the value of *I*. This can be proven easily: the PDF is uniform over the entire domain, and must thus be equal to:

$$p(x) = \left(\prod_{i=1}^{d} (\beta_i - \alpha_i) \right)^{-1}$$

It is now straightforward to compute the expected value of $\langle I \rangle$:

$$\begin{aligned}
E(\langle I\rangle) &= E\left(\left(\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right)\cdot\left(\prod_{i=1}^{d}(\beta_i-\alpha_i)\right)\right)\\
&= \sum_{i=1}^{N}\frac{1}{N}\cdot\left(\prod_{i=1}^{d}(\beta_i-\alpha_i)\right)\cdot E(f(x_i))\\
&= \left(\prod_{i=1}^{d}(\beta_i-\alpha_i)\right)\cdot\int_D \frac{f(x)}{\prod_{i=1}^{d}(\beta_i-\alpha_i)}dx\\
&= \int_D f(x)dx\\
&= I
\end{aligned}$$

This estimator also has a variance σ^2, which is expressed as:

$$\begin{aligned}
\sigma^2 &= \frac{1}{N\cdot\prod_{i=1}^{d}(\beta_i-\alpha_i)}\int_D (f(x)-I)^2 dx\\
&= \frac{1}{N\cdot\prod_{i=1}^{d}(\beta_i-\alpha_i)}\left(\int_D f(x)^2 dx - I^2\right)
\end{aligned}$$

and which in turn can be estimated by:

$$\langle\sigma^2\rangle = \frac{\frac{1}{N}\sum_{i=1}^{N} f(x_i)^2 - \left(\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right)^2}{N\cdot\prod_{i=1}^{d}(\beta_i-\alpha_i)}$$

Basic Monte Carlo integration will often produce much larger errors compared to deterministic quadrature formulas with a comparable number of sample points. However, Monte Carlo integration can be useful when we want to compute integrals of high dimensions. When using classic quadrature rules, the number of sampling points usually grows exponentially with the dimension of the integral, whereas the number of sampling points in a Monte Carlo integration can be of any nature. Another situation where we might opt to use a Monte Carlo integration technique is when the functions to be integrated are of a rather complex nature, and do not allow us to make any predictions about the expected error margins. The main advantage of Monte Carlo integration is that it provides us with an unbiased estimator, which can be important in the cases mentioned.

3. Importance Sampling

Monte Carlo integration methods can roughly be subdivided in two categories: those that have no information about the function to be integrated, and those that do have some kind of information available about the function. Some authors call the first class of methods 'blind Monte Carlo', and the second class 'informed Monte Carlo' techniques. Intuitively, one expects that informed Monte Carlo methods are able to produce more accurate results as opposed to blind Monte Carlo methods. The basic Monte Carlo integration algorithm outlined above is a blind Monte Carlo method, because it generates its sample points uniformly, without looking at the function itself.

This section describes a sampling technique known as importance sampling. Importance sampling uses a non-uniform probability function for generating samples. By choosing the probability function wisely on the basis of some knowledge of the function to be integrated, we can often reduce the variance. If we have a PDF $p(x)$ defined over the integration domain D, and if we are able to generate sample points x_i according to $p(x)$, we can then estimate the value of I by generating N sample points and computing the weighted mean:

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

The expected value of $\langle I \rangle$ equals I, and this ensures that the estimator is unbiased:

$$\begin{aligned} E(\langle I \rangle) &= E\left(\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N}E\left(\frac{f(x_i)}{p(x_i)}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_D \frac{f(x)}{p(x)}p(x)dx \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_D f(x)dx \\ &= I \end{aligned}$$

To determine whether this estimator behaves better than uniform sampling, we have to compute its variance σ:

$$\sigma^2 = \int_D\left(\frac{f(x)}{p(x)} - I\right)^2 p(x)dx = \int_D\left(\frac{f(x)}{p(x)}\right)^2 p(x)dx - I^2$$

which can be estimated by its own estimator $\langle \sigma^2 \rangle$:

$$\langle\sigma^2\rangle = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{f(x_i)}{p(x_i)}\right)^2 - \left(\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right)^2}{N}$$

It is obvious that the choice of $p(x)$ has an influence on the value of the variance. The difficulty of importance sampling is to choose a $p(x)$ in such a way that the variance is as low as possible, in order to make the estimator as reliable as possible. The optimal $p(x)$ can be found by minimizing the expression for variance, using variational techniques and Lagrange multipliers.

We have to find the scalar λ for which the following expression L, a function of $p(x)$, reaches a minimum:

$$L(p) = \int_D \left(\frac{f(x)}{p(x)}\right)^2 p(x)dx + \lambda\int_D p(x)dx$$

We have one boundary condition, which states that the integral of $p(x)$ equals 1 over the integration domain:

$$\int_D p(x)dx = 1$$

This kind of problem can be solved by using the Euler-Lagrange differential equation.

$$L(p) = \int_D \left(\frac{f(x)^2}{p(x)} + \lambda p(x)\right)dx$$

To minimize, we need to differentiate:

$$\Rightarrow 0 = \frac{\partial}{\partial p}\left(\frac{f(x)^2}{p(x)} + \lambda p(x)\right)$$

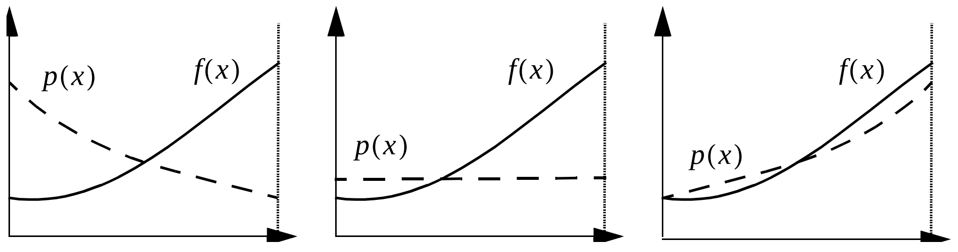
$$\Rightarrow 0 = -\frac{f^2(x)}{p^2(x)} + \lambda$$

$$\text{or} \quad p(x) = \frac{1}{\sqrt{\lambda}}|f(x)|$$

The constant $1/\sqrt{\lambda}$ is a scaling factor, such that $p(x)$ can fulfill the boundary condition. The optimal $p(x)$ is then given by:

$$p(x) = \frac{|f(x)|}{\int_D f(x)dx}$$

If we use this $p(x)$, the variance will be exactly 0. It is not possible to construct such a $p(x)$ because this implies we have to know the value of I, which is exactly what we seek. This does not imply, however, that importance sampling cannot be used as a sampling tool. It is one of the major tools in order to enhance Monte Carlo integration techniques. Intuitively, a good importance sampling function matches the shape of the original function as closely as possible. The figure below shows three different probability functions. Each of them will produce an estimator whose expected value will be equal to the value of the integral, but the variance of the one on the left-hand side will be larger than the variance of the sampling function shown on the right hand side.



Importance sampling with different PDFs.

There are various methods for constructing a ‘good’ probability function. The most obvious one is to build a numerical probability table by sampling the function to be integrated and use that table to generate samples. Another (adaptive) strategy could consist of constructing the PDF at various steps during the sampling process, based on the information gathered with all samples so far. This kind of strategy will be explained further in this text.

Another way of looking at importance sampling is to consider importance sampling as a transformation of variables of an integral. Suppose we want to evaluate the following one-dimensional integral:

$$I = \int_a^b f(x)dx = \int_a^b \frac{f(x)}{p(x)} \cdot p(x)dx = \int_a^b \frac{f(x)}{p(x)} dP(x)$$

We can rewrite this integral by carrying out a variable transformation:

$$y = P(x) \qquad x = P^{-1}(y)$$
$$dy = p(x)dx$$

The integral then is expressed as:

$$I = \int_{P(a)}^{P(b)} \frac{f(P^{-1}(y))}{p(P^{-1}(y))} d(y)$$

If $p(x)$ is a PDF, then $P(a) = 0$ and $P(b) = 1$. Evaluating this integral using a simple Monte Carlo integration requires the generation of a number of samples y_i over the interval $[0, 1]$, and evaluating $P^{-1}(y_i)$. This is exactly the same procedure as was used for generating non-uniform samples, by taking the inverse of the cumulative probability distribution function.

4. Stratified Sampling

When generating samples over a domain, we have no control over where the samples will be positioned relative to each other. It is therefore possible that we have clusters of samples in one region, and almost no samples in another region. In other words, there may occur severe deviations from the number of samples we can expect in a certain part of the domain. This can happen irrespective of the PDF used, because the PDF only tells us something about the expected number of samples in parts of the domain, but not about the number of samples actually generated in that part.

Stratified sampling is a sampling technique that counters the effect of clumping. The basic idea is to split up the integration domain in *m* disjunct subdomains (also called *strata*), and evaluate the integral in each of the subdomains separately with one or more samples. More precisely:

$$\int_0^1 f(x)dx = \int_0^{\alpha_1} f(x)dx + \int_{\alpha_1}^{\alpha_2} f(x)dx + \ldots + \int_{\alpha_{m-2}}^{\alpha_{m-1}} f(x)dx + \int_{\alpha_{m-1}}^{1} f(x)dx$$

Stratified sampling often leads to a smaller variance as opposed to a crude Monte Carlo integration method. The variance of a stratified sampling method, where each stratum receives a number of samples n_j, which are in turn distributed uniformly over their respective intervals, is equal to [Hamm64]:

$$\sigma^2 = \sum_{j=1}^{m} \frac{(\alpha_j - \alpha_{j-1})}{n_j} \int_{\alpha_{j-1}}^{\alpha_j} f(x)^2 dx - \sum_{j=1}^{m} \frac{1}{n_j} \left\{ \int_{\alpha_{j-1}}^{\alpha_j} f(x)dx \right\}^2$$

If all the strata are of equal size ($\alpha_j - \alpha_{j-1} = 1/m$), and each stratum contains one uniformly generated sample ($n_j = 1; N = m$), the above equation can be simplified to:

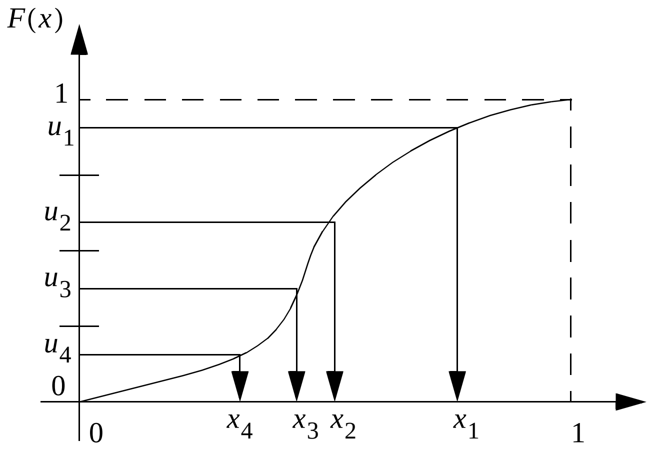
$$\sigma^2 = \sum_{j=1}^{m} \frac{1}{N} \int_{\alpha_{j-1}}^{\alpha_j} f(x)^2 dx - \sum_{j=1}^{m} \left\{ \int_{\alpha_{j-1}}^{\alpha_j} f(x)dx \right\}^2$$

$$= \frac{1}{N}\int_0^1 f(x)^2 dx - \sum_{j=1}^{N} \left\{ \int_{\alpha_{j-1}}^{\alpha_j} f(x)dx \right\}^2$$

This expression indicates that this variance is always smaller than the one obtained by a pure Monte Carlo sampling scheme. As a consequence, there is no advantage in generating more than one sample within a single stratum, since a simple equal subdivision of the stratum such that each sample is attributed to a single substratum, always yields a better result.

This does not mean however, that the above sampling scheme always gives us the smallest possible variance, because we did not take into account the size of the strata relative to each other and the number of samples per stratum. It is not an easy problem to determine how these degrees of freedom can be chosen optimally, such that the final variance is the smallest possible. It can be proven that the optimal number of samples in one subdomain is proportional to the variance of the function values relative to the average function value in that subdomain. Applied to the principle of one sample per stratum, this implies that the size of the strata should be chosen such that the function variance is equal in all strata. Such a sampling strategy assumes prior knowledge of the function in question, which is often not available. However, such sampling strategy might be used in an adaptive sampling algorithm [Pres90].

Stratified sampling can also be combined with importance sampling. This is quite logical, since importance sampling basically is a transformation from one integration domain to another. This strategy avoids the clumping of samples, and at the same time distributes the samples according to the preset probability distribution.



Stratified Monte Carlo sampling combined with importance sampling.

5. When to use Monte Carlo integration?

Monte Carlo integration often produces results that are worse than deterministic quadrature formulas that require an equal amount of work. However, Monte Carlo integration can yield some advantages:

- Monte Carlo integration always provides us with an unbiased estimator of the integral. This means that the expected value of the integration procedure equals the exact value of the integral.
- Monte Carlo integration can prove to be useful for integrating functions with complex behavior (e.g. discontinuities). Deterministic integration algorithms often assume that the integrand does not differ much from the class of functions for which the integration method is originally designed (e.g. low order polynomials).
- Monte Carlo integration can be used for high-dimensional integrals, and does not require complex subdivisions of the integration domain. Moreover, irregular or complex integration domains can be handled by means of a Monte Carlo integration scheme quite easily.

The main drawback is the fact that we still end up with a significant amount of error, and that it is hard to provide an upper bound for the error, due to the variance of the stochastic process. One way to reduce the variance is to use stratified sampling or importance sampling as discussed above. Another way of reducing the error is to compute as much of the integral as possible using deterministic techniques. This can be summarized by the following principle: ‘*If, at any point of a Monte Carlo calculation, we can replace an estimate by an exact value, the sampling error in the final result will be reduced.*’ The second drawback is that the variance only decreases proportional to the square root of the number of samples. Deterministic quadrature formulas usually give faster convergence rates.

As a summary, one can state that Monte Carlo can be used for unknown, high-dimensional functions; if possible, stratified sampling and importance sampling should be used to generate a good distribution of samples.

6. Fredholm Equations and Monte Carlo Integration

This section describes various strategies for solving Fredholm equations using Monte Carlo integration. These equations are important for the global illumination problem, because the transport of radiance and potential is expressed exactly by this kind of equations. Because the transport equations are essentially recursive, which means the integrand is unknown, Monte Carlo integration is a viable alternative to compute function values described by these equations.

6.1 Fredholm equations of the second kind

A Fredholm equation of the second kind (a recursive, linear integral equation, with a fixed integration domain) in its most general form can be written as:

$$a(x) = b(x) + \int_D K(x, y)a(y)dy$$

The unknown function is $a(x)$. $b(x)$ and the kernel $K(x, y)$ are given functions. Generally, we want to evaluate $a(x)$ in a number of points x in order to learn more about the behavior of $a(x)$.

Since the equation is recursive and no termination condition to stop the recursive nature of the computation is explicitly stated, one evaluation of a requires in its turn the evaluation of an integral. The evaluation can therefore be thought of as an integral of infinite dimension.

$$\begin{aligned} a(x) &= b(x) + \int_D K(x, y)a(y)dy \\ &= b(x) + \int_D K(x, y)b(y)dy + \int_D K(x, y)\left(\int_D K(y, y')a(y')dy'\right)dy \\ &= b(x) + \int_D K(x, y)b(y)dy + \int_D\int_D K(x, y)K(y, y')b(y')dy'dy + \dots \end{aligned}$$

A Fredholm equation can generally not be solved using analytical techniques. Because of the high dimension of the integral, a Monte Carlo method could be a good alternative.

6.2 Relationship with global illumination

The Fredholm equation of the second kind is a very important equation for the global illumination problem. Indeed, all transport equations of radiance are equations of this kind.

$$L(x \to \Theta_{out}) = L_e(x \to \Theta_{out}) + \int_{\Omega_x} d\omega_\Theta L(y \to -\Theta)f_r(x, \Theta \leftrightarrow \Theta_{out})\cos(\Theta, n_x)$$

$$\text{where} \quad y = r(x, \Theta)$$

y is the first visible point as seen from x in direction Θ. The BRDF, together with the cosine term, takes the role of the kernel. The emittance of light sources and the initial importance act as the initially known function *b*. These equations, however, are not one-dimensional equations, but two-dimensional Fredholm equations in which the integration domain is the hemisphere around a single surface point.

6.3 Recursive Monte Carlo solutions

When solving a Fredholm equation, we basically want to evaluate the unknown function $a(x)$ at a number of points x_i. These points are determined by the requirements of another computation, possibly an integration method of the function $a(x)$ over a certain integration domain. Suppose the function a needs to be evaluated for a certain argument value x_0. $b(x)$ is a known function, so $b(x_0)$ can be evaluated directly. To estimate the integral part of $a(x_0)$, we generate a number of sample points y_i over the domain *D* using a PDF $p_1(y)$. An estimator for $a(x_0)$ is then given by:

$$\langle a(x_0)\rangle = b(x_0) + \frac{1}{N}\sum_{i=1}^{N}\frac{K(x_0, y_i)a(y_i)}{p_1(y_i)}$$

This expression does not really give us a usable value, since the values for $a(y_i)$ are still unknown. Thus, $a(x_0)$ can be approximated by adding more approximations recursively:

$$\begin{aligned}\langle a(x_0)\rangle &= b(x_0) + \frac{1}{N_1}\sum_{i=1}^{N_1}\frac{K(x_0, y_i)}{p_1(y_i)}\left(b(y_i) + \frac{1}{N_2}\sum_{j=1}^{N_2}\frac{K(y_i, z_j)}{p_2(z_j|y_i)}a(z_j)\right)\\ &= b(x_0) + \frac{1}{N_1}\sum_{i=1}^{N_1}\frac{K(x_0, y_i)}{p_1(y_i)}\left(b(y_i) + \frac{1}{N_2}\sum_{j=1}^{N_2}\frac{K(y_i, z_j)}{p_2(z_j|y_i)}(b(z_j) + \ldots)\right)\end{aligned}$$

There are a few problems with this method of estimating $a(x_0)$:

- The sum is in principle infinite and will never stop. However, if the kernel function fulfils certain conditions, the subsequent terms become smaller and smaller, and will contribute less to the final result. Nevertheless, one has to provide some sort of termination criterion to stop the recursive algorithm.

- The number of samples required for estimating each term in the sum grows very quickly; i.e., estimating the second term requires $N_1 \cdot N_2$ new samples. Since the nature of the Monte Carlo algorithm requires that the number of samples is large, this means a significant amount of computational work.

- One might wonder whether the infinite series of sums will ever converge. This depends on the nature of the kernel function $K(x, y)$. Without going into a detailed description, one can say that the Fredholm equation has a solution provided that the series

  $$K(x, y) + \int_D K(x, x_1)K(x_1, y)dx_1 + \int_D dx_1\int_D dx_2 K(x, x_1)K(x_1, x_2)K(x_2, y) + \ldots$$

  converges for a given *x* and *y*. A sufficient condition for convergence of this series is that:

  $$\|K\| = \int_D |K(x, y)|\,dy < 1$$

This is indeed the case for the global illumination transport equations, where the kernel function $K(x, y)$ equals the BRDF times a cosine factor. The convergence can also be deduced from the physical interpretation of the equation. Due to the restriction of energy conservation, the equations describing light transport must always yield a viable result, without diverging or infinite solutions.

Absorption

Because the Fredholm equation is inherently recursive, an estimator produced by a Monte Carlo algorithm is given by an infinite sum. If we want the evaluating algorithm to stop, we have to introduce a termination condition. However, we still want to maintain the advantage of a non-biased estimator. Simply ignoring the terms of the sum which fall beyond a certain threshold will introduce a bias. The concept of absorption provides one possibility of handling this problem.

Suppose we want to integrate the function $f(x)$ over its integration domain $[0, 1]$. Before evaluating the function for each sample point x_i, we first decide whether or not to use x_i (or *absorb* x_i). The probability of absorption, is given by the absorption coefficient α ($0 < \alpha < 1$). The absorption is decided by generating a uniform random variable u over $[0, 1]$.

For each sample point x_i, we now decide what estimator to use. Because a number of samples will be absorbed, we have to give the non-absorbed samples a higher weight. An estimator y is constructed such that:

$$u \le \alpha: \qquad y = 0$$

$$u > \alpha: \qquad y = \frac{f(x_i)}{1-\alpha}$$

The expected value of y equals:

$$E[y] = 0 \cdot \alpha + \frac{f(x_i)}{1-\alpha}(1-\alpha) = f(x_i)$$

Thus, absorption does not change the expected value of the experiment.

The variance can be expressed as:

$$\sigma^2[y] = \left(\frac{f(x_i)}{1-\alpha} - f(x_i)\right)^2 (1-\alpha) + (0 - f(x_i))^2\alpha = \frac{f(x_i)^2\alpha}{(1-\alpha)^2}$$

It is easy to imagine that the $f(x)$ can be of any nature. $f(x)$ can be a recursive function evaluation, or even the result of another stochastic process. If the absorption test produces a value for u which is larger than α, we evaluate $f(x)$, otherwise, the result is 0. If we apply such a scheme to a recursive evaluation (e.g. the Fredholm equation), we decide at each step whether we evaluate the next recursive term. This provides us with a nice termination algorithm, that still keeps the advantage that the final result will be unbiased, due to the extra weighting factor.

The value of α should be chosen carefully. If α is large, the recursive evaluation will terminate rapidly, but we can expect a high variance on the final result. If α is small, the recursive evaluation has a small chance of terminating, but the result will be more reliable. Usually, the context of the specific problem provides some insight in order to choose a good value for α.

The relation with global illumination lies in the fact that during light-surface interaction, part of the irradiance is absorbed by the surface, and is being transformed into heat or some other form of energy. This physical process also has to be modelled somehow in our simulation. Thus use of an absorption coefficient is an elegant way to do this.

Next event estimation

A Monte Carlo evaluation of the Fredholm equation can be expressed as:

$$\langle a(x_0)\rangle = b(x_0) + \frac{1}{N}\sum_{i=1}^{N}\frac{K(x_0, y_i)a(y_i)}{p_1(y_i)}$$

which is an estimator for:

$$a(x_0) = b(x_0) + \int_D K(x_0, y)a(y)dy$$

The choice of PDF $p_1(y)$ strongly influences the variance on the final result. According to the principle of importance sampling, we get better results if $p_1(y)$ closely resembles $K(x_0, y)a(y)$. But, $a(y)$ is still unknown at his point of the global evaluation, so we can only base our choice on the knowledge of the kernel $K(x_0, y)$.

However, the function *a* is not completely unknown, because the first term of the sum is the known function *b*. We can rewrite the Fredholm equation as:

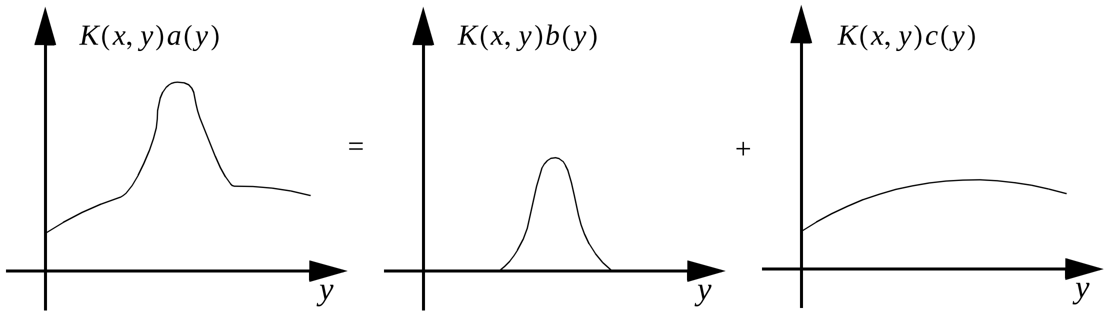
$$a(x) = b(x) + c(x) \quad \text{where} \quad c(x) = \int_D K(x, y)a(y)dy$$

$$\begin{aligned} c(x) &= \int_D K(x, y)\left(b(y) + \int_D K(y, y')a(y')dy'\right)dy \\ &= \int_D K(x, y)b(y)dy + \int_D K(x, y)\int_D K(y, y')a(y')dy'dy \\ &= \int_D K(x, y)b(y)dy + \int_D K(x, y)c(y)dy \end{aligned}$$

In order to evaluate $c(x)$, we now have to evaluate two integrals. However, there is one distinct advantage: because we have split the integral in two parts, we can apply much better estimation techniques to each part individually. The first term of this sum contains known functions $K(x, y)$ and $b(y)$. We can therefore apply an informed Monte Carlo algorithm to this first term. Possibly, we might even be able to evaluate this first term analytically. As a result, the variance for an estimator of $c(x)$ will decrease. The second term still is a recursive evaluation. Here, we still can rely only on our knowledge of $K(x, y)$ in order to come up with a suitable sampling scheme.

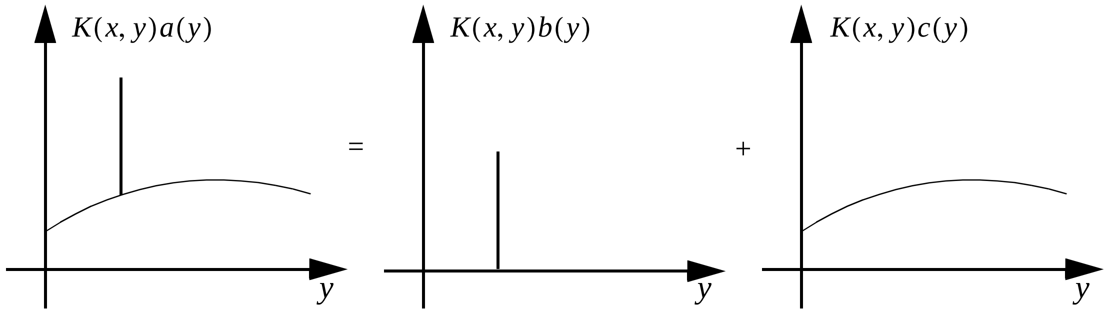
The next event estimation technique will prove to be useful if $K(x, y)b(y)$ has some distinct properties which makes it easy to sample or to compute analytically, so that the decrease in variance will be obvious. Some typical examples are listed below.

- If $b(y)$ is a function which is zero over a large part of the domain, and contributes significantly to the overall result of the integral, we might choose to generate more samples in the part of the integration domain where $b(y)$ differs from zero. By using next event estimation, we can limit the generation of samples over the subdomain only.



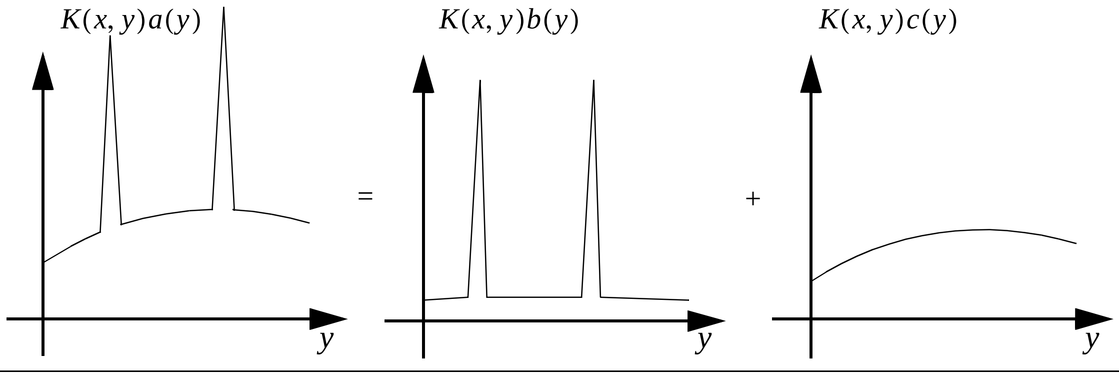
Next event estimation applied to a function which is zero over a significant part of the integration domain.

- A more extreme situation occurs if $b(y)$ equals a Dirac-impulse. We can integrate the first term analytically. In this case, sampling never yields the correct result, since we are never able to generate the exact point where $b(y)$ differs from zero. An analytical evaluation provides us with an exact result and thus a better approximation to the overall value $c(y)$.



Next event estimation applied to a Dirac-impulse.

- A combination of the above cases arises when $b(y)$ has some very sharp spikes. Due to their high function values, these spikes will cause a high variance when selected with a general sampling scheme. If we can separate these spikes from the main integral, we can adjust the sampling function such that the spikes themselves are sampled with a much higher frequency. This might reduce the overall variance.

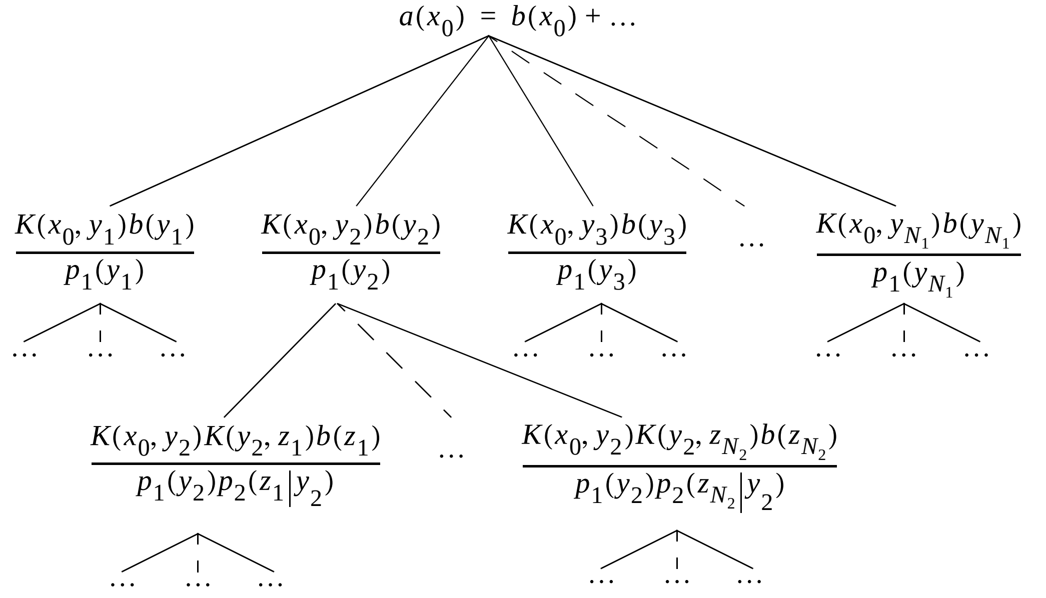


Next event estimation applied to a function containing sharp spikes.

The above examples, and their respective solutions, are actually based on the same principle as importance sampling: if the function value is high, we want to generate more samples at that point. If we can compute the integral analytically (e.g. a Dirac-impulse), we might gain a much more significant decrease in the variance of the final estimator.

Random walks

As stated above, one of the problems associated with solving Fredholm equations using a Monte Carlo approach, is that the number of samples might increase very rapidly, due to the recursive nature of the equation. At each step of the recursion, a number of samples is generated, thereby effectively generating a tree of samples. At each node of the tree, we have to compute a possible contribution to the overall result.


$a(x_0) = b(x_0) + \ldots$
$\frac{K(x_0, y_1)b(y_1)}{p_1(y_1)}$
$\frac{K(x_0, y_2)b(y_2)}{p_1(y_2)}$
$\frac{K(x_0, y_3)b(y_3)}{p_1(y_3)}$
$\ldots$
$\frac{K(x_0, y_{N_1})b(y_{N_1})}{p_1(y_{N_1})}$
$\frac{K(x_0, y_2)K(y_2, z_1)b(z_1)}{p_1(y_2)p_2(z_1|y_2)}$
$\ldots$
$\frac{K(x_0, y_2)K(y_2, z_{N_2})b(z_{N_2})}{p_1(y_2)p_2(z_{N_2}|y_2)}$

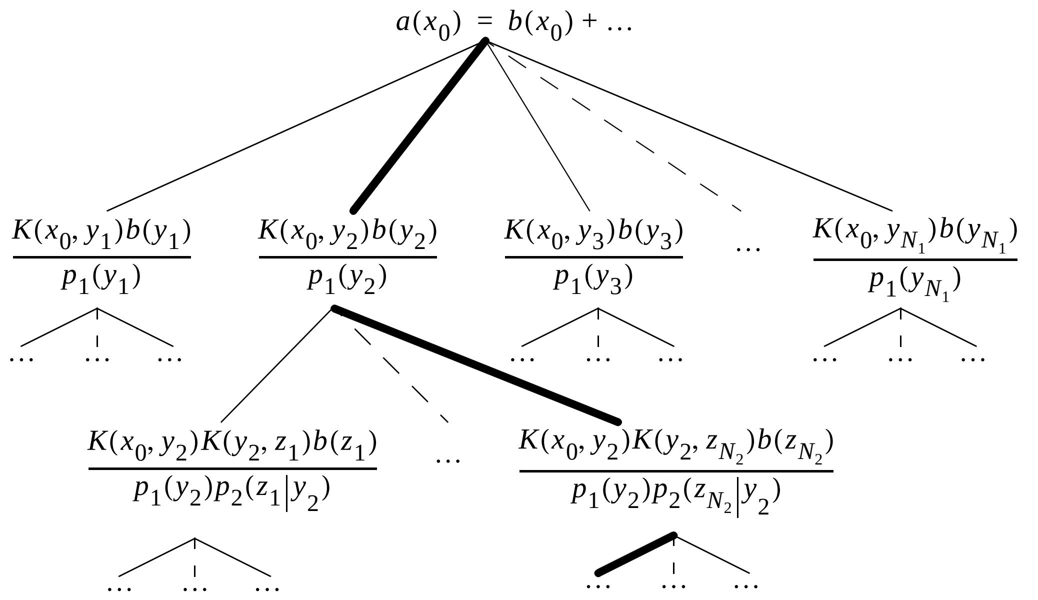
Tree resulting from evaluating $a(x_0)$.

As can be seen, much work is spent computing the contributions of the nodes lower in the tree, simply because there are a lot more nodes at those depths. These lower nodes typically yield less significant contributions to the overall result. This is a consequence of the increasing multiplications of the kernel function. This approach seems disadvantageous, because much work is spent evaluating and generating samples that have no visible impact on the final result of $a(x_0)$. On the other hand, we do not want to ignore these deeper nodes, because they ensure us of an unbiased estimator for $a(x_0)$. A compromise is needed to concentrate more work in the higher branches of the tree, without ignoring possible contributions of the lower nodes.

One might be tempted to think that increasing the probability of absorption will solve this problem. A higher absorption ratio will indeed limit the depth of the tree, but there are still more lower branches when the overall number of branches is kept constant.

A more elegant solution is to distribute the work evenly over all the levels of the tree. We can accomplish this by generating only one sample at each recursive level. This gives rise to a so-called random walk. The concept of absorption is maintained to end the recursion..

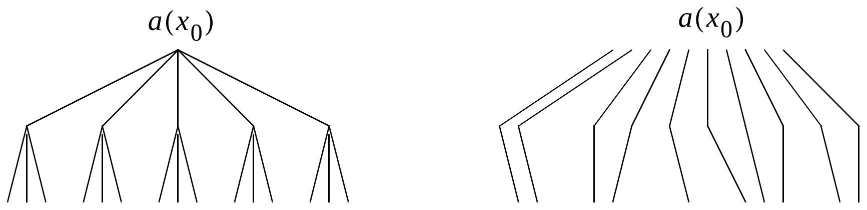
A single value $a(x_0)$ can be approximated by adding all contributions from the random walk:


$a(x_0) = b(x_0) + \dots$
$\frac{K(x_0, y_1)b(y_1)}{p_1(y_1)}$
$\frac{K(x_0, y_2)b(y_2)}{p_1(y_2)}$
$\frac{K(x_0, y_3)b(y_3)}{p_1(y_3)}$
$\dots$
$\frac{K(x_0, y_{N_1})b(y_{N_1})}{p_1(y_{N_1})}$
$\frac{K(x_0, y_2)K(y_2, z_1)b(z_1)}{p_1(y_2)p_2(z_1|y_2)}$
$\dots$
$\frac{K(x_0, y_2)K(y_2, z_{N_2})b(z_{N_2})}{p_1(y_2)p_2(z_{N_2}|y_2)}$

Random walk for evaluating *a*(x_0).

$$\langle a(x_0)\rangle = b(x_0) + \frac{K(x_0, x')}{p'(x')}b(x') + \frac{K(x_0, x')K(x', x'')}{p'(x')p''(x''|x')}b(x'') + \dots$$

The variance associated with this estimate is very bad, because we approximate each level of the recursion with just one sample. On the other hand, we have not used as many samples as we would have done when generating an entire tree. This gives us an opportunity to generate several random walks. By generating several random walks originating from $a(x_0)$, and subsequently computing the average of these walks, we have more estimators for $a(x_0)$, and the result might improve. The difference between these two approaches is drawn schematically below.



Single tree versus several random walks.

The net result of using random walks is that more effort will be spent in computing the first terms of the recursive sum. Under the conditions given above for convergence of the Fredholm equation, this seems reasonable. Indeed, the contributions of the terms later in the series are less and less significant, and will have a decreasing effect on the final result. Therefore, more relative effort should be put in the first terms.

The total estimator obtained by performing several random walks can be expressed as:

$$\langle a(x_0)\rangle = \frac{1}{N_{walks}} \sum_{i=1}^{N_{walks}} \left(b(x_0) + \frac{K(x_0, x')}{p'(x')}b(x') + \frac{K(x_0, x')K(x', x'')}{p'(x')p''(x''|x')}b(x'') + \dots \right)$$

Bibliography

[Arvo86] James Arvo. “Backward ray tracing.” In *SIGGRAPH ’86 Developments in Ray Tracing seminar notes*, volume 12, August 1986.

[Beka95] Ph. Bekaert and Y.D. Willems. “Importance-driven progressive refinement radiosity.” In *Proceedings of the Sixth Eurographics Workshop on Rendering*, pages 349–358, Dublin, Ireland, June 1995.

[Boli97] Mark Bolin and Gary Meyer. “An Error Metric for Monte Carlo Ray Tracing.” In *Proceedings of the Eighth Eurographics Workshop on Rendering*, pages 57–68, St-Etienne, France, June 1997.

[Chat87] Sudeb Chattopadhyay and Akira Fujimoto. “Bi-directional ray tracing.” In Tosiyasu L. Kunii, editor, *Computer Graphics 1987 (Proceedings of CG International ’87)*, pages 335–43, Tokyo, 1987. Springer-Verlag.

[Chen91] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. “A progressive multi-pass method for global illumination.” In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH ’91 Proceedings)*, volume 25, pages 165–174, July 1991.

[Chri93] P.H. Christensen, D.H. Salesin, and T.D. DeRose. “A continuous adjoint formulation for radiance transport.” In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 95–104, Paris, France, June 1993.

[Cohe88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. “A progressive refinement approach to fast radiosity image generation.” In John Dill, editor, *Computer Graphics (SIGGRAPH ’88 Proceedings)*, volume 22, pages 75–84, August 1988.

[Cohe93] Michael F. Cohen and John R. Wallace. “Radiosity and realistic image synthesis.” Academic Press Professional, San Diego, CA, 1993.

[Cook81] Robert L. Cook and Kenneth E. Torrance. “A reflectance model for Computer Graphics”. In *Computer Graphics (SIGGRAPH '81 Proceedings)*, 15(3):307--316, August 1981

[Cook84] Robert L. Cook, Thomas Porter, and Loren Carpenter. “Distributed ray tracing.” In *Computer Graphics (SIGGRAPH ’84 Proceedings)*, volume 18, pages 137–45, July 1984.

[Davi84] Ph.J. Davis and Ph. Rabinowitz, editors. *Methods of Numerical Integration*. Academic press, Orlando, 1984.

[Dutr93] Ph. Dutré, E.P. Lafortune, and Y.D. Willems. “Monte Carlo light tracing with direct computation of pixel intensities.” In *Proceedings of CompuGraphics*, pages 128–137, Alvor, Portugal, December 1993.

[Dutr94a] Ph. Dutré. “A mathematical framework for global illumination algorithms.” In *Winter School of Computer Graphics and CAD Systems*, pages 75–84, Pilzen, Czech Republic, January 1994.

[Dutr94b] Ph. Dutré and Y.D. Willems. “Importance-driven light tracing.” In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 185–194, Darmstadt, Germany, June 1994.

[Dutr95] Ph. Dutré and Y.D. Willems. “Potential-driven Monte Carlo particle tracing for diffuse environments with adaptive probability density functions.” In *Proceedings of the Sixth Eurographics Workshop on Rendering*, pages 339–348, Dublin, Ireland, June 1995.

[Fole90] J. D. Foley, A. van Dam, Steven K. Feiner, and John F. Hughes. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley Publishing Company, second edition, 1990.

[Glas89] Andrew Glassner. *An Introduction to Ray Tracing*. Academic Press, 1989.

[Glas95] Andrew Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.

[Gora84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. “Modelling the interaction of light between diffuse surfaces.” In *Computer Graphics (SIGGRAPH ’84 Proceedings)*, volume 18, pages 212–22, July 1984.

[Hamm64] J.M. Hammersly and D.C. Handscomb. *Monte Carlo Methods*. Chapman and Hall, London, 1964.

[He92] Xiao D. He, Patrick O. Heynen, Richard L. Phillips, Kenneth E. Torrance, David H. Salesin, and Donald P. Greenberg. “A fast and accurate light reflection model”. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):253--254, July 1992.

[Heck92] Paul Heckbert. “Discontinuity meshing for radiosity.” *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.

[Jens95] Henrik Wann Jensen. “Importance driven path tracing using the photon map.” In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.

[Jens96] Henrik Wann Jensen. “Global illumination using photon maps.” In *Proceedings of the Seventh Eurographics Workshop on Rendering*, pages 22–31, June 1996.

[Kaji86] James T. Kajiya. “The rendering equation.” In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH ’86 Proceedings)*, volume 20, pages 143–150, August 1986.

[Kalo86] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods*. Wiley & Sons, New York, 1986.

[Kell95] A. Keller. “Quasi-monte carlo methods in computer graphics: The global illumination problem.” In *SIAM Conference in Park City*, 1995.

[Kell96] A. Keller. “Quasi-monte carlo radiosity.” In *Proceedings of the Seventh Eurographics Workshop on Rendering*, pages 102–111, June 1996.

[Lafo93] E.P. Lafortune and Y.D. Willems. “Bi-directional path tracing.” In *Proceedings of CompuGraphics*, pages 145–153, Alvor, Portugal, December 1993.

[Lafo94] Eric P. Lafortune and Yves D. Willems. “A theoretical framework for physically based rendering.” *Computer Graphics Forum*, 13(2):97–107, June 1994.

[Lafo96] E.P. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Katholieke Universiteit Leuven, Belgium, February 1996.

[Lafo97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. “Non-linear approximation of reflectance functions”. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 117--126. ACM SIGGRAPH, Addison Wesley, August 1997.

[Lepa78] G.P. Lepage. “A new algorithm for adaptive multidimensional integration.” *Journal of Computational Physics*, 27:192–203, 1978.

[Lisc92] Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. “Discontinuity meshing for accurate radiosity.” *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.

[Lisc93] Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. “Combining hierarchical radiosity and discontinuity meshing.” In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 199–208, 1993.

[Neum95a] L. Neumann, W. Purgathofer, R. Tobler, A. Neumann, P. Elias, M. Feda, and X. Pueyo. “The stochastic ray method for radiosity.” In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques ’95*. Eurographics, Springer-Verlag, July 1995.

[Patt92a] S. N. Pattanaik and S. P. Mudur. “Computation of global illumination by monte carlo simulation of the particle model of light.” *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.

[Patt92b] S.N. Pattanaik and S.P. Mudur. “Computation of global illumination by Monte Carlo simulation of the particle model of light.” In *Proceedings of the Third Eurographics Workshop on Rendering*, pages 71–83, Bristol, UK, May 1992.

[Patt93b] S. N. Pattanaik and S. P. Mudur. “The potential equation and importance in illumination computations.” *Computer Graphics Forum*, 12(2):131–136, 1993.

[Phon75] Bui-T. Phong. “Illumination for computer generated pictures.” *Communications of the ACM*, 18(6):311–317, June 1975.

[Pres90] W.H. Press and G.R. Farrar. *Computers in Physics*, 4:190–195, 1990.

[Pres92] W.H. Press, W.T. Vetterling, S.A. Teukolsky, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[Shir90] P. Shirley. *Physically Based Lighting Calculations for Computer Graphics*. Ph.D. thesis, Dept. of Computer Science, U. of Illinois, Urbana-Champaign, November 1990.

[Shir91] P. Shirley and C. Wang. “Direct lighting calculation by monte carlo integration.” In *Eurographics Workshop on Rendering*, 1991.

[Shre66] Y.A. Shreider, editor. *The Monte Carlo Method*. Pergamon Press, Oxford, 1966.

[Sill91] Francois X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. “A global illumination solution for general reflectance distributions.” In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH ’91 Proceedings)*, volume 25, pages 187–196, July 1991.

[Sill94] Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994.

[Smit92] Brian E. Smits, James R. Arvo, and David H. Salesin. “An importance-driven radiosity algorithm.” In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH ’92 Proceedings)*, volume 26, pages 273–282, July 1992.

[Stro71] A.H. Stroud. *Approximative Calculation of Multiple Integrals*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[Tams97] Rasmus Tamstorf and Henrik Wann Jensen. “Adaptive Sampling and Bias Estimation in Path Tracing.” In *Proceedings of the Eighth Eurographics Workshop on Rendering*, pages 57–68, St-Etienne, France, June 1997.

[Uren97] Carlos Urena and Juan C. Torres. “Improved Iradiance Computation by Importance Sampling.” In *Proceedings of the Eighth Eurographics Workshop on Rendering*, pages 57–68, St-Etienne, France, June 1997.

[Veac94] Eric Veach and Leonidas Guibas. “Bidirectional estimators for light transport.” In *Fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany, June 1994.

[Veac95] Eric Veach and Leonidas J. Guibas. “Optimally combining sampling techniques for monte carlo rendering.” In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 419–428. ACM SIGGRAPH, 1995.

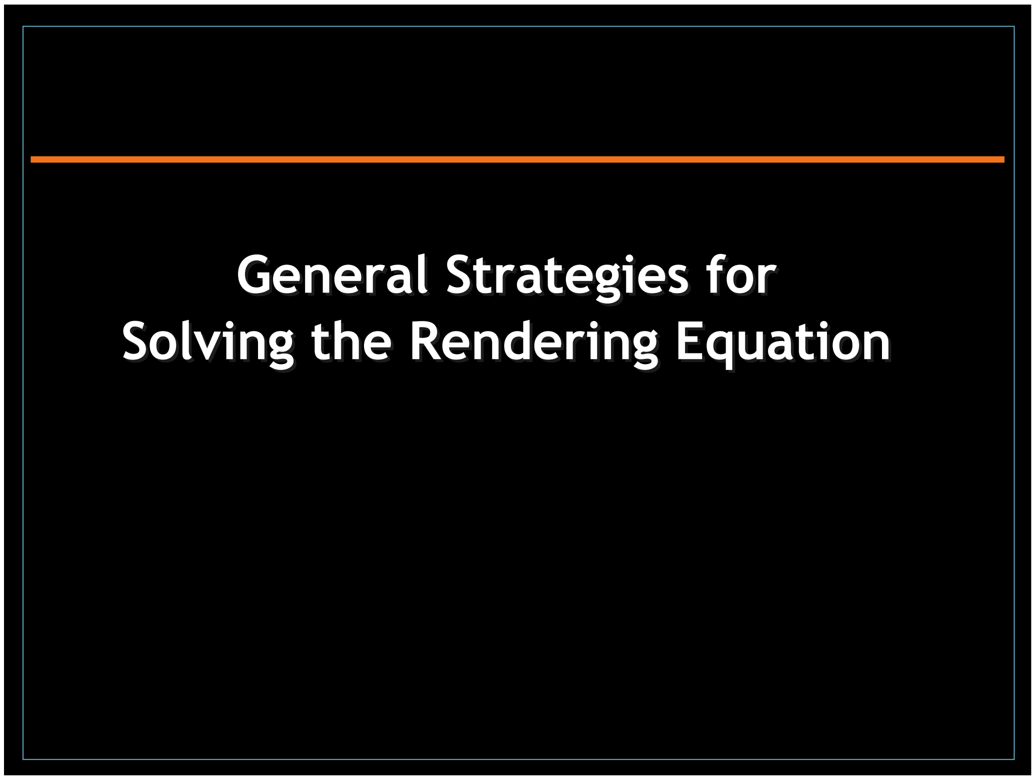
[Veac96] Eric Veach. “Non-symmetric scattering in light transport algorithms.” In *Seventh Eurographics Workshop on Rendering*, pages 82–91, June 1996.

[Veac97] Eric Veach and Leonidas J. Guibas. “Metropolis Light Transport.” In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 65–76. ACM SIGGRAPH, 1997.

[Ward92] Gregory J. Ward. “Measuring and Modeling Anisotrpoic Behavior.” In *Computer Graphics (SIGGRAPH ’92 Proceedings)*, 26(4), pages 265-272, July 1992.

[Whit80] Turner Whitted. “An improved illumination model for shaded display.” *Communications of the ACM*, 23(6):343–349, June 1980.

[Zimm95] Kurt Zimmerman and Peter Shirley. “A two-pass realistic image synthesis method for complex scenes.” In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.

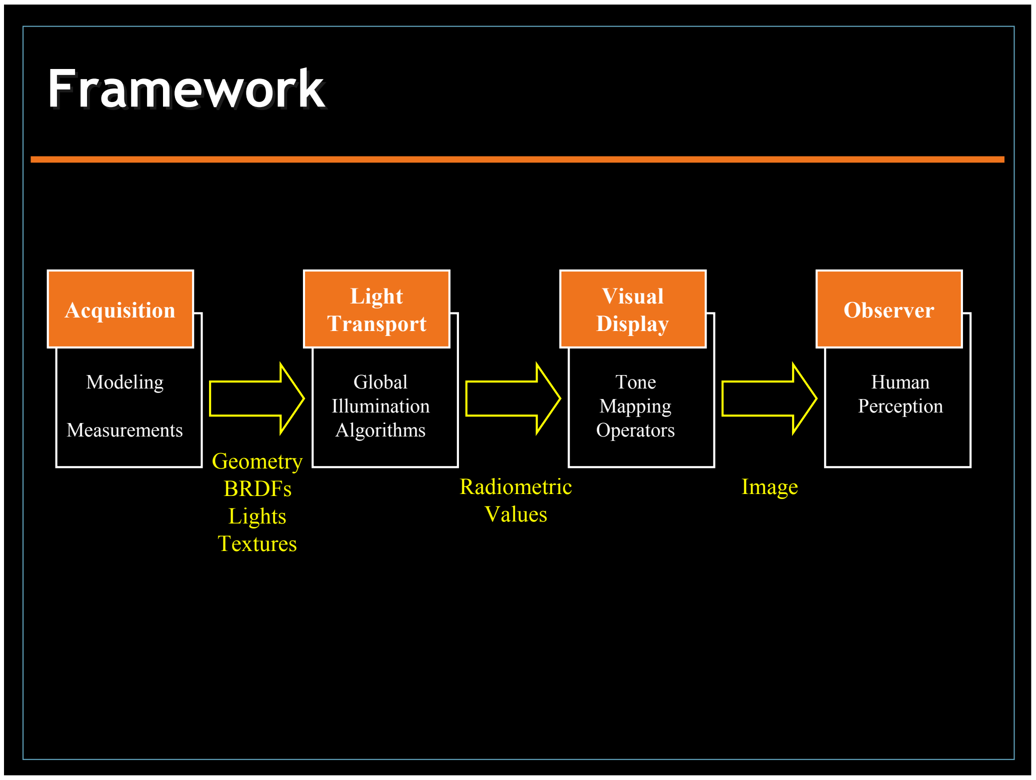


General Strategies for Solving the Rendering Equation

In this section of the course, we will look at some general strategies for computing the rendering equation.

Rather than give specific examples of algorithms, this section will outline some useful general design principles that can be used to design your own algorithms.

Two aspects are important: the notion that all light transport between a source and a receiver is really taking into account all possible light paths between that source and that receiver; and secondly, Monte Carlo integration is a useful tool for integrating the transported energy over all these possible paths.

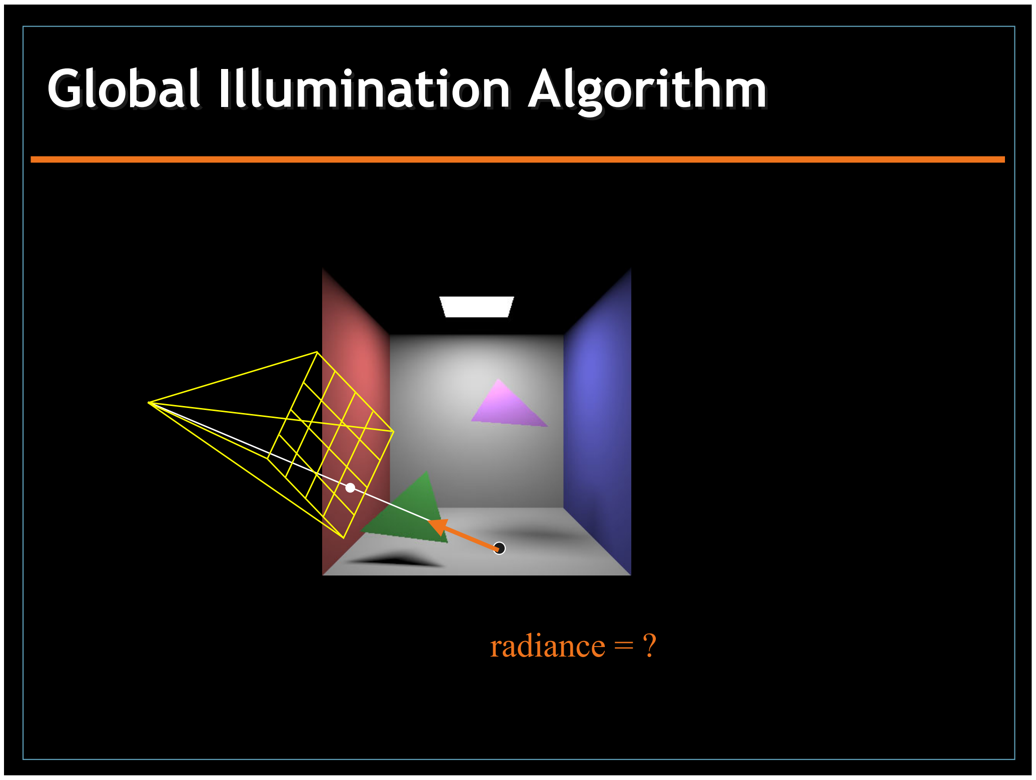


Framework

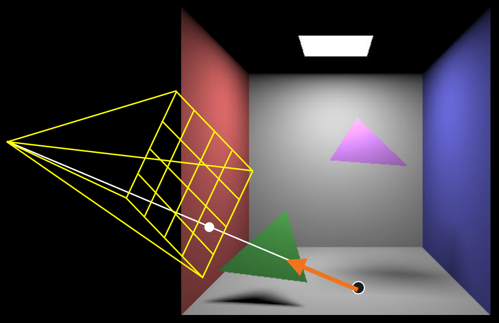
This is again the overall framework for photo-realistic rendering.

After the acquisition of BRDFs and geometry, we can now solve the rendering equation in order to compute the light transport between the light sources in the scene and the pixels for which we want to compute a radiance value.

The result of the light transport phase will be that for each pixel, we will obtain a radiance value that passes through that pixel to the direction of the eye. As such, we have a matrix of radiance values. Afterwards, these radiance values will have to be transformed to RGB values that can be used to display the picture on the screen.



Global Illumination Algorithm

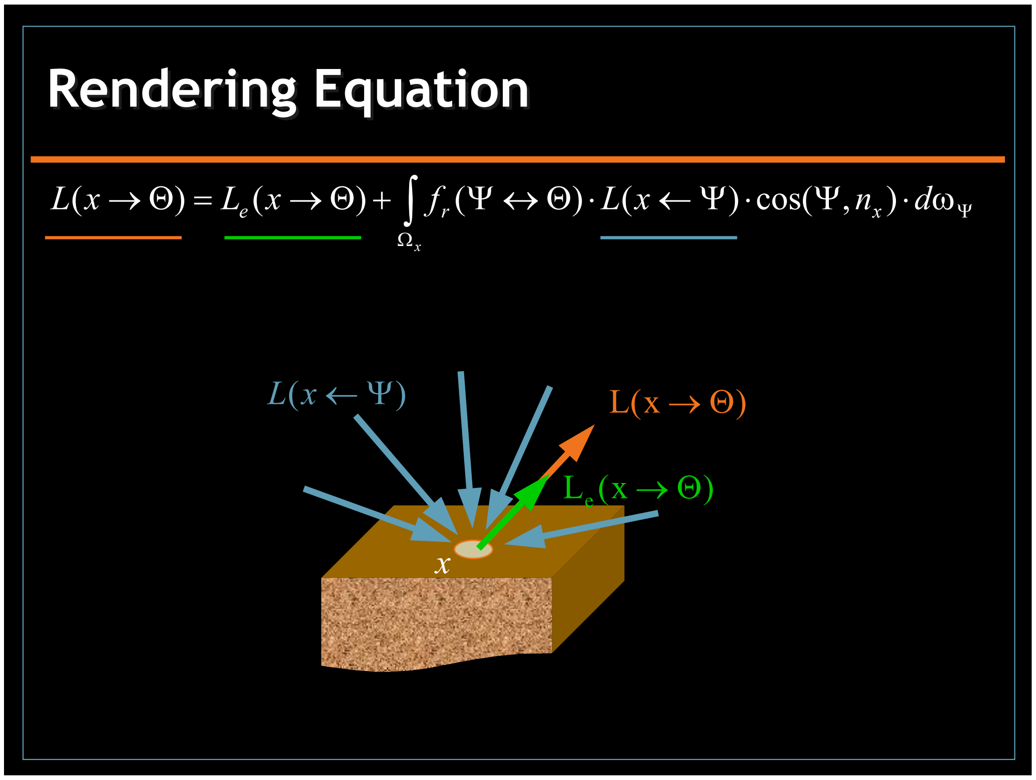


radiance = ?

This is the general setup:

Given a scene (in this case a simple box with two floating triangles), the virtual camera (specified by its location, viewing direction, viewing angle, resolution, etc.), we can trace a viewing ray from the eye through each pixel, and find the closest intersection point in the scene. This is really a simple ray-casting operation, and we assume we have an appropriate implementation (which might use acceleration structures such as hierarchical bounding volumes or spatial voxel subdivision which are well described in classic ray tracing literature).

We now want to compute the radiance value leaving this surface point in the direction of the eye. So, the light transport algorithm really starts once we have found that first intersection point.



Rendering Equation

$$\underline{L(x \to \Theta)} = \underline{L_e(x \to \Theta)} + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot \underline{L(x \leftarrow \Psi)} \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

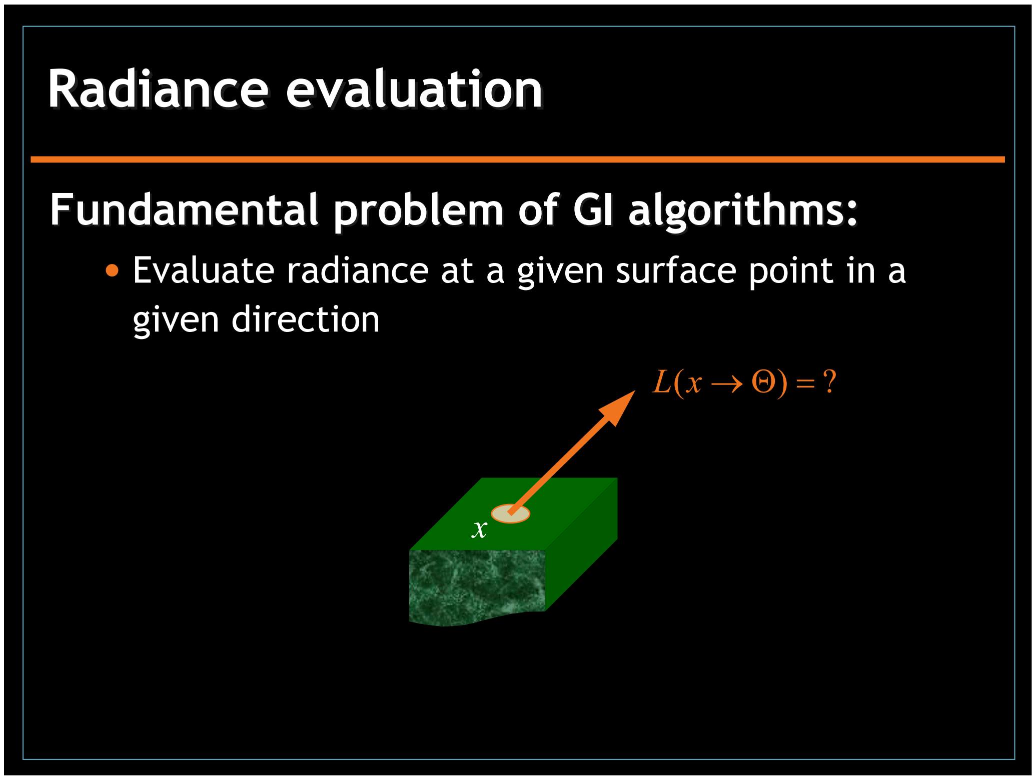
This is the rendering equation we want to solve:

The radiance emitted by a point x in a direction Θ equals:

• the self-emitted radiance (which is different from 0 only if x is on the surface of a light source)

• plus the reflected radiance, which is all incoming radiance, multiplied by the BRDF, multiplied by the cosine factor, and integrated over the entire hemisphere.

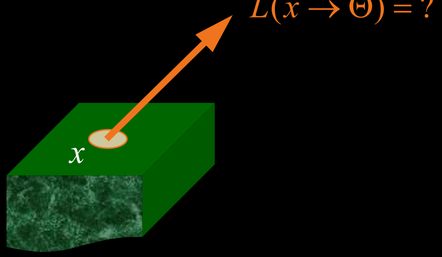
Remember that the incoming radiance can be traced back to another surface point in space where it is the outgoing radiance. Thus, the rendering equation is inherently recursive.



Radiance evaluation

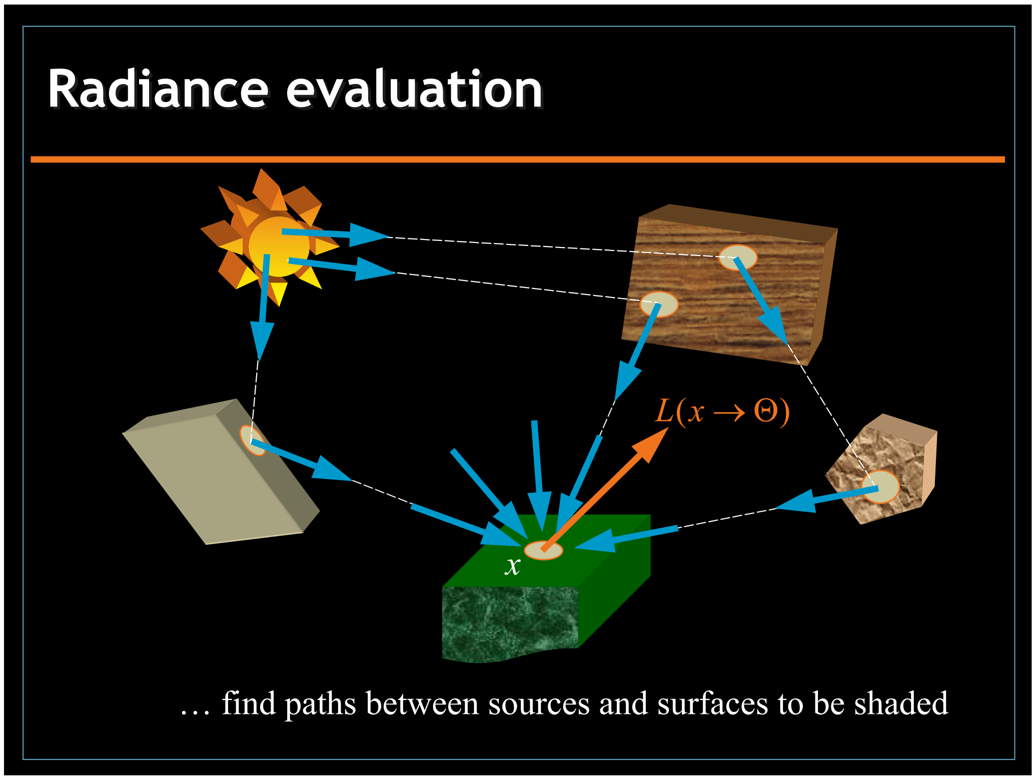
Fundamental problem of GI algorithms:

- Evaluate radiance at a given surface point in a given direction

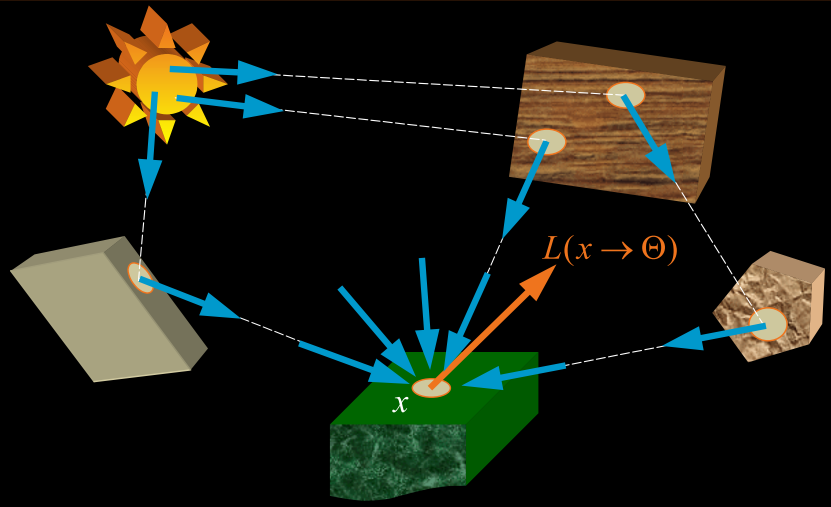


Evaluating the radiance value at a surface point x in a direction Θ really is the fundamental problem in all global illumination algorithms.

Not only do we want these radiance values at all points visible through the pixels, but also we need to evaluate these radiance values at others points in the scene, when we will trace back the incoming radiance at x and transform it in some outgoing radiance at another point.



Radiance evaluation



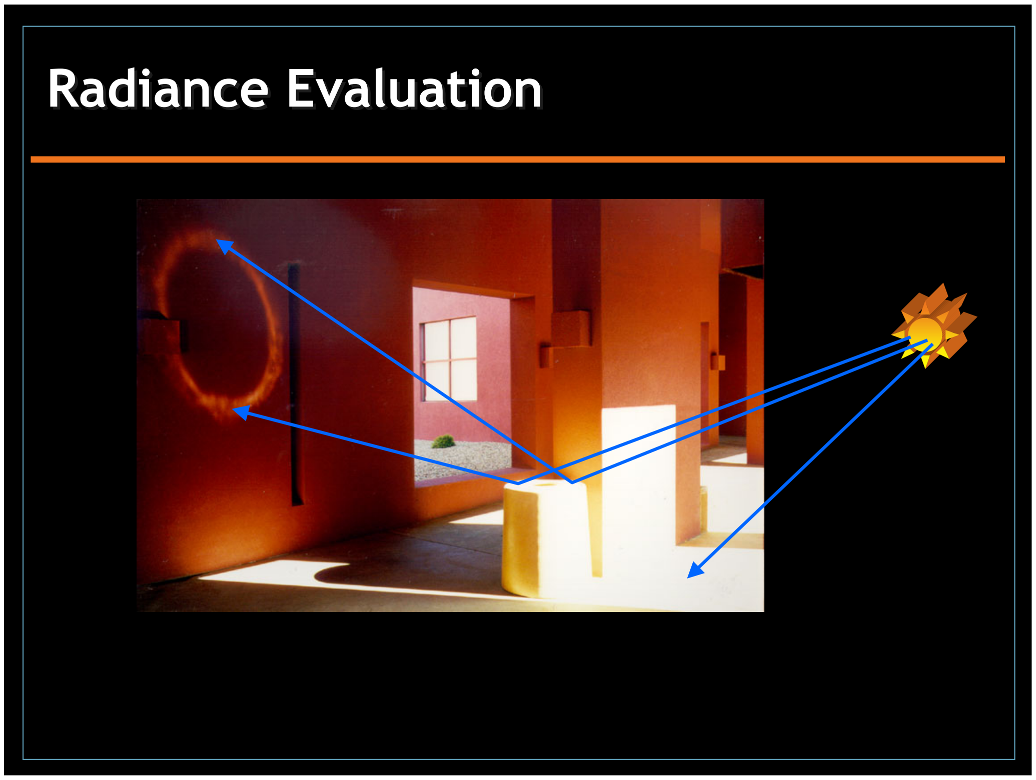
… find paths between sources and surfaces to be shaded

This is an illustration of the recursive nature of the rendering equation.

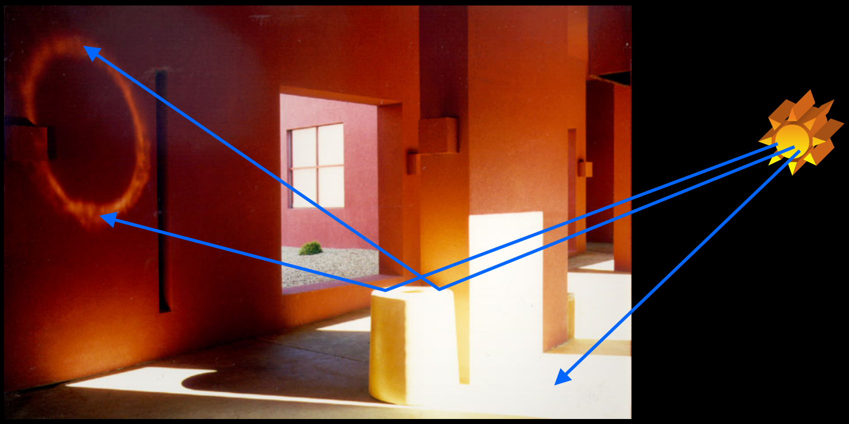
All radiance values incident at surface point x are themselves outgoing radiance values. We have to trace back the path they arrive from. This means tracing a ray from x in the direction of the incoming radiance. This results in some surface point, and the incoming radiance along the original direction now simply equals the outgoing radiance at this new point.

The problem is then stated recursively, since this new radiance value is also described exactly by the rendering equation.

This process will continue until the paths are traced back to the light source. Then we can pick up the self-emitted radiance, take into account all possible cosine factors and possible BRDF values along the path, perform the necessary integration at each surface point, to finally arrive at the original radiance value we are interested in.



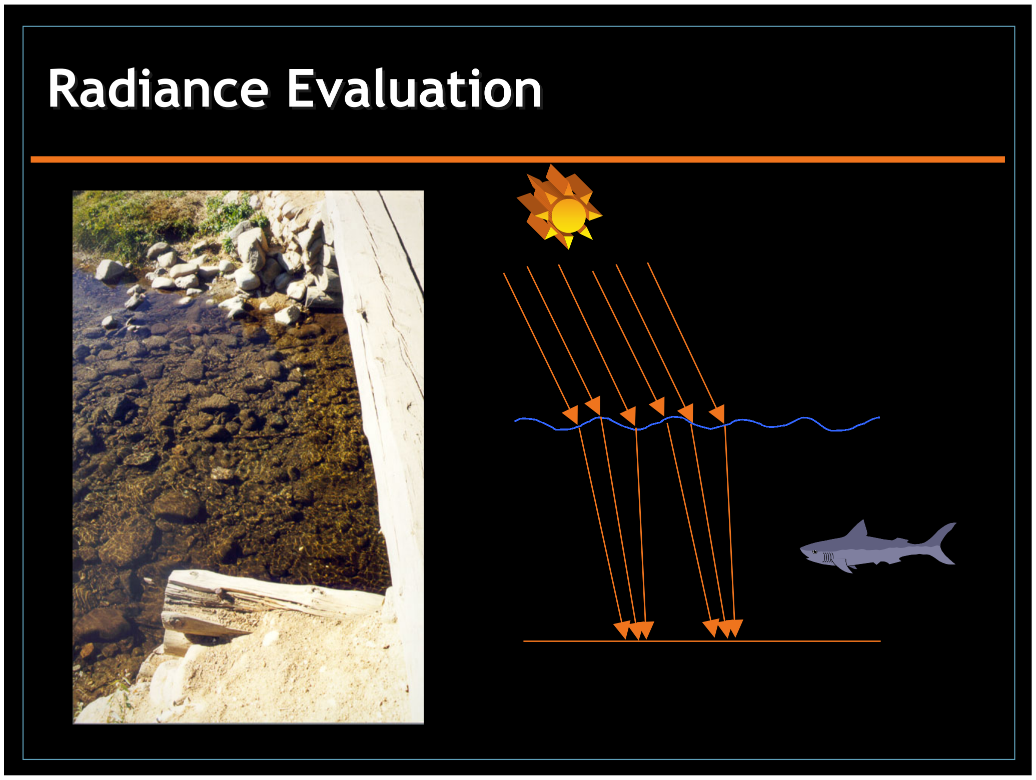
Radiance Evaluation



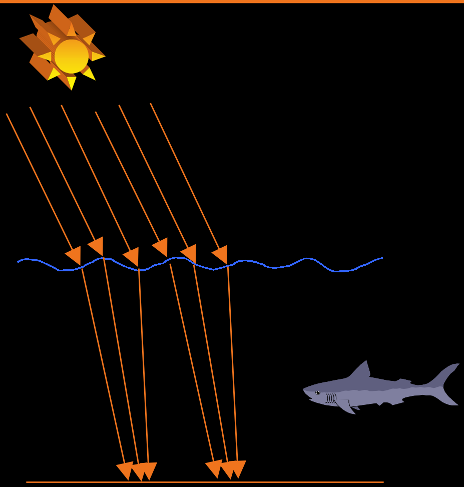
Reconstructing all possible paths between the light sources and the point for which one wants to compute a radiance value is the core business of all global illumination algorithms.

This photograph was taken on a sunny day in New Mexico. It is shown here just to illustrate some of the unexpected light paths one might have to reconstruct when computing global illumination solutions.

The circular figure on the left wall is the reflection of the lid on the trash-can. The corresponding light paths (traced from the sun), hit the lid, then hit the wall, and finally end up in our eye. For a virtual scene, these same light paths need to be followed to reconstruct the reflection.



Radiance Evaluation

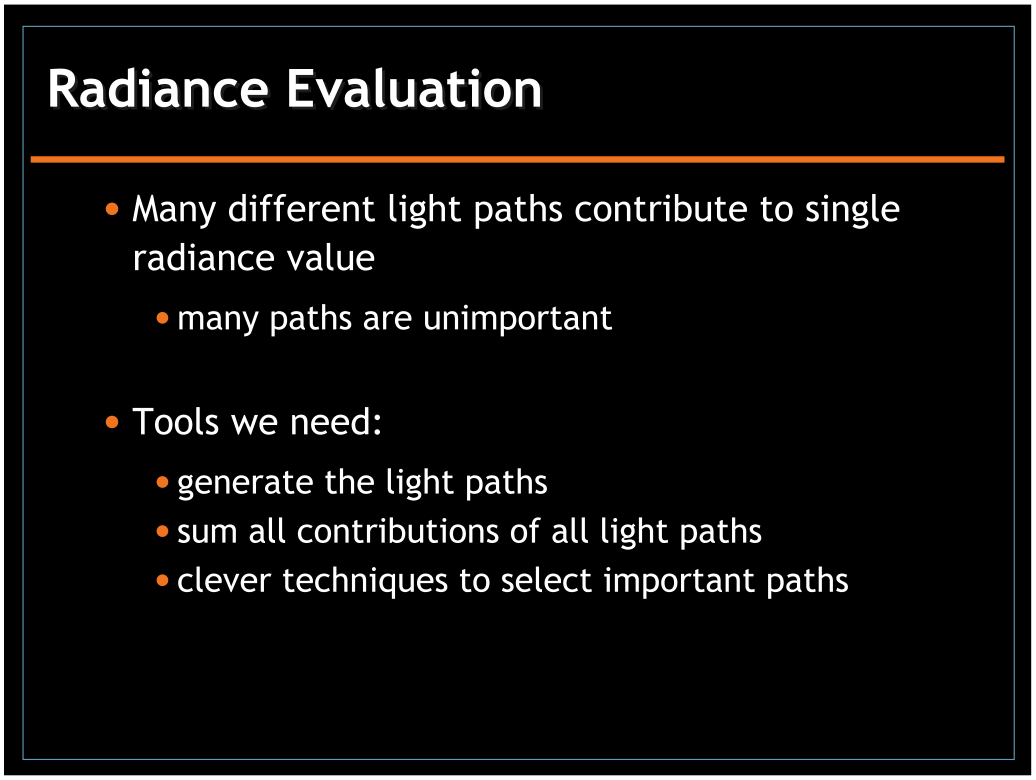


This photograph shows a similar effect.

We see shimmering waves on the bottom of the river (a similar effect is noticable in swimming pools). Light rays from the sun hit the transparent and wavy surface of the water, then are reflected on the bottom of the river, are refracted again by the water, the they hit our eye.

The complex pattern of light rays hitting the bottom, together with the changing nature of the surface of the water, causes these shimmering waves.

This effect is known as a caustic: light rays are reflected or refracted in different patterns and form images of the light source: the circular figure in the previous photograph, or the shimmering waves in this one.



Radiance Evaluation

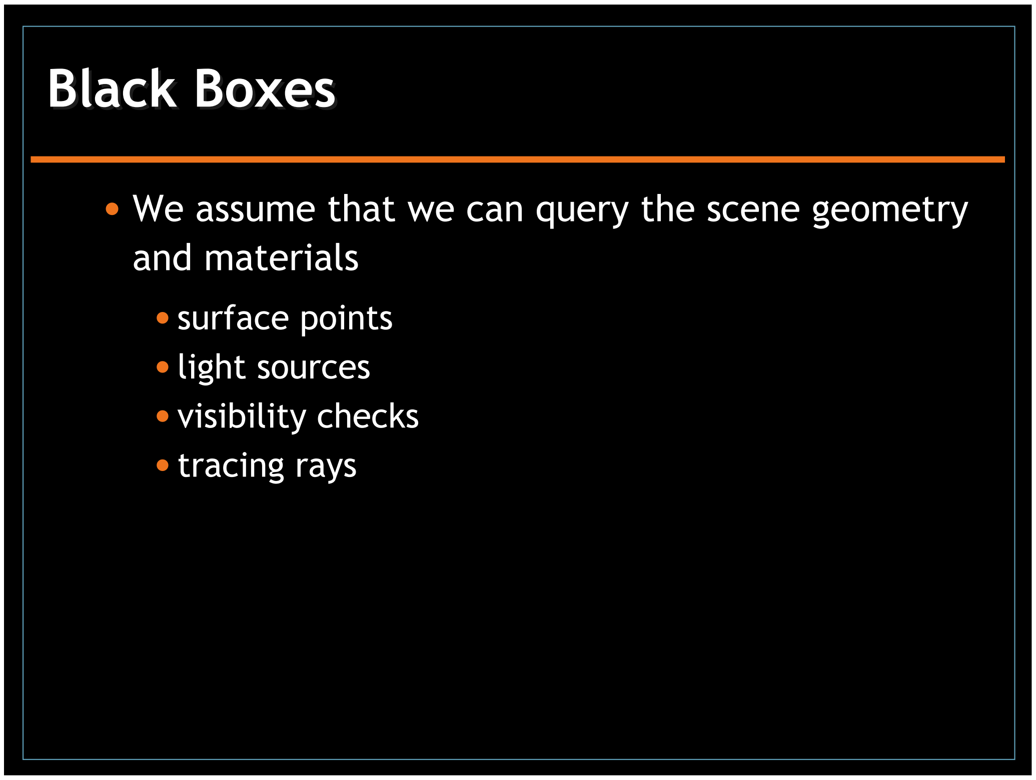
- Many different light paths contribute to single radiance value
  - many paths are unimportant
- Tools we need:
  - generate the light paths
  - sum all contributions of all light paths
  - clever techniques to select important paths

So, many different light paths, all originating at the light sources, will contribute to the value of the radiance at a single surface point.

Many of these light paths will be unimportant. Imagine a light switched on on the 1st floor of a building. You can imagine that some photons will travel all the way up to the 4th floor, but it is very unlikely that this will contribute significantly to the illumination on the 4th floor. However, we cannot exclude these light paths from consideration, since it might happen that the contribution is significant after all.

So, one of the mechanisms that a good global illumination algorithm needs is how to select the important paths from amongst many different possibilities, or at least how to try to put more computational effort into the ones that are likely to give the best contributions.

This is of course a chicken and egg problem. If we would know what the importance of each path was, we would have solved the global illumination problem. So the best we can do is to make clever guesses.



Black Boxes

- We assume that we can query the scene geometry and materials
  - surface points
  - light sources
  - visibility checks
  - tracing rays

To design the light transport component of a global illumination algorithm, a few building blocks need to be in place.

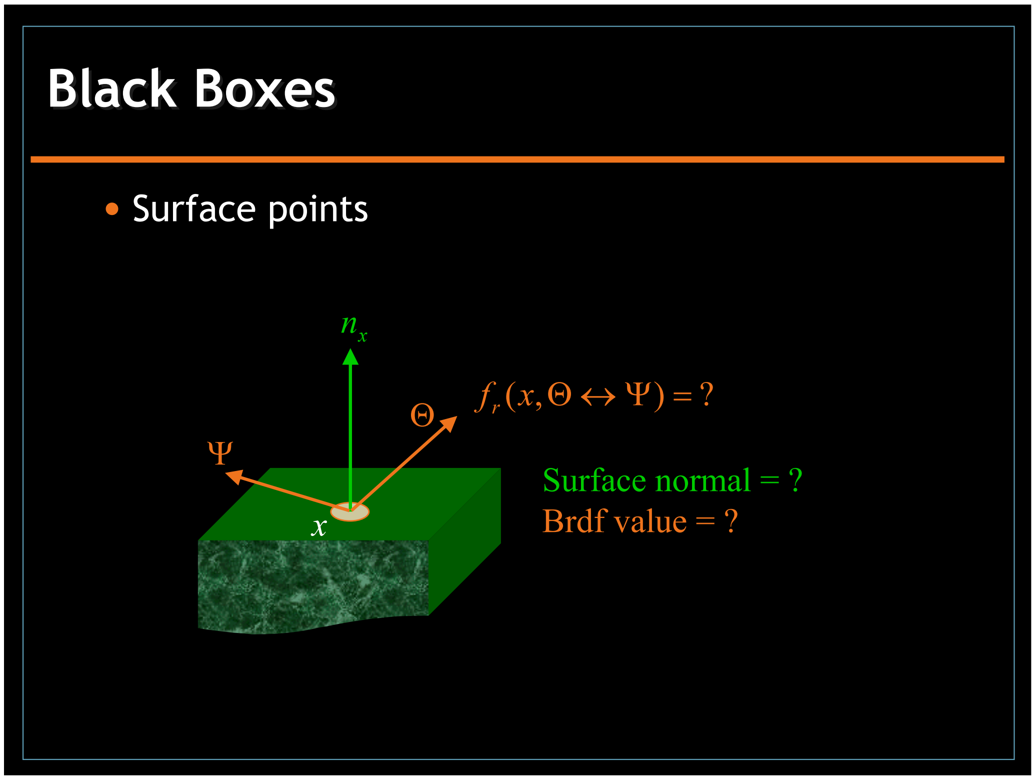
One should assume that one can query the virtual scene: the properties of surface points, light sources etc. should be easily accessible.

Also, we need a way to check visibility. There are two different sorts of visibility checks:

• Given 2 points in 3D space, are the mutually visible? In other words, is any other solid object intersecting the line connecting those two points?

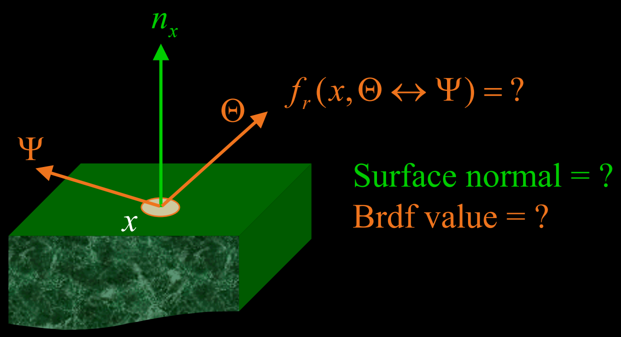
• Given a surface point and a direction, what is the nearest surface points one can see looking from the surface point in the given direction?

Both of these visibility queries need to be implemented, usually this is done using a hierarchy of bounding volumes or a spatial acceleration structure (e.g. a voxel-grid). Classic ray tracing literature gives many details on how this can be done efficiently.



Black Boxes

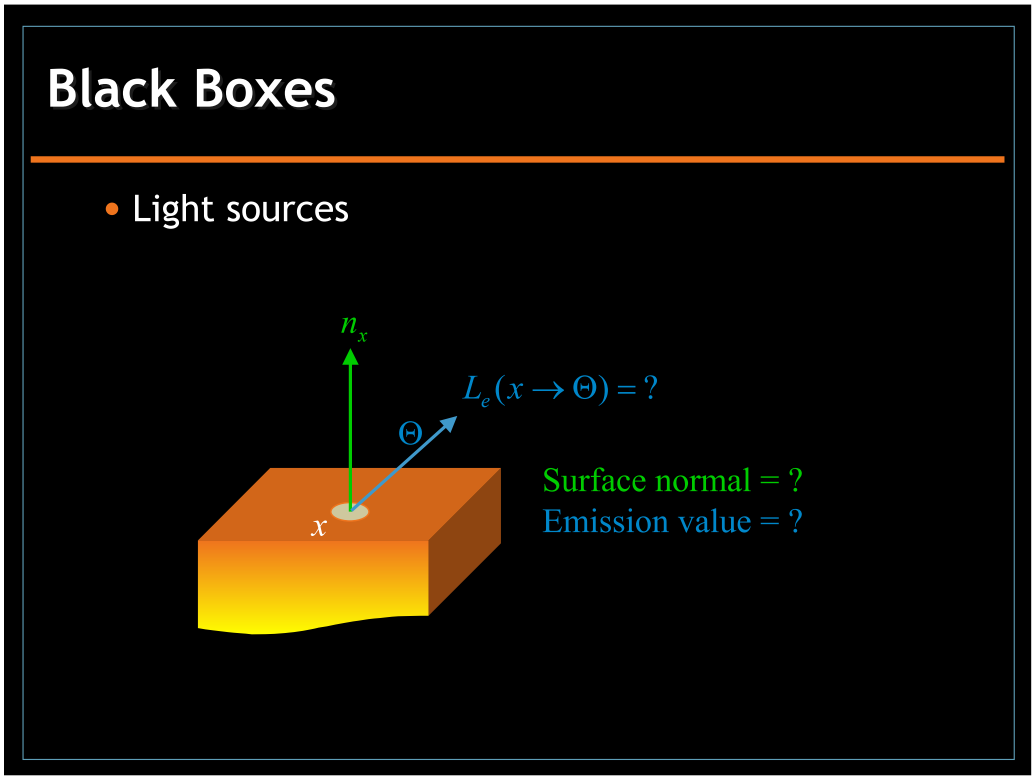
- Surface points



The queries we want to make about a surface point:

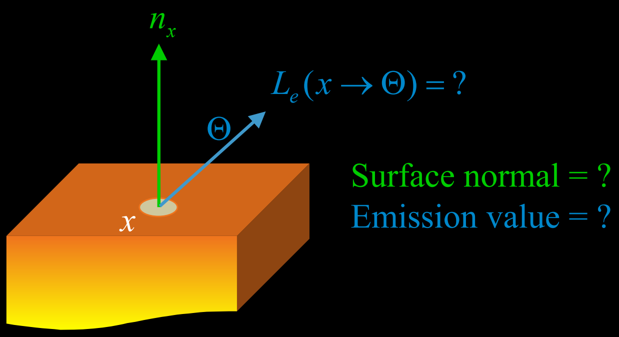
• What is the surface normal at this point?

• Given two directions, what is the corresponding BRDF value? Note that this might involve looking up texture values, since the BRDF might be given in the form of a texture map. In general, this is not the same as a 'shader'. Care has to be taken that the returned BRDF values are physically plausible (reciprocity constraint and energy conservation constraint).



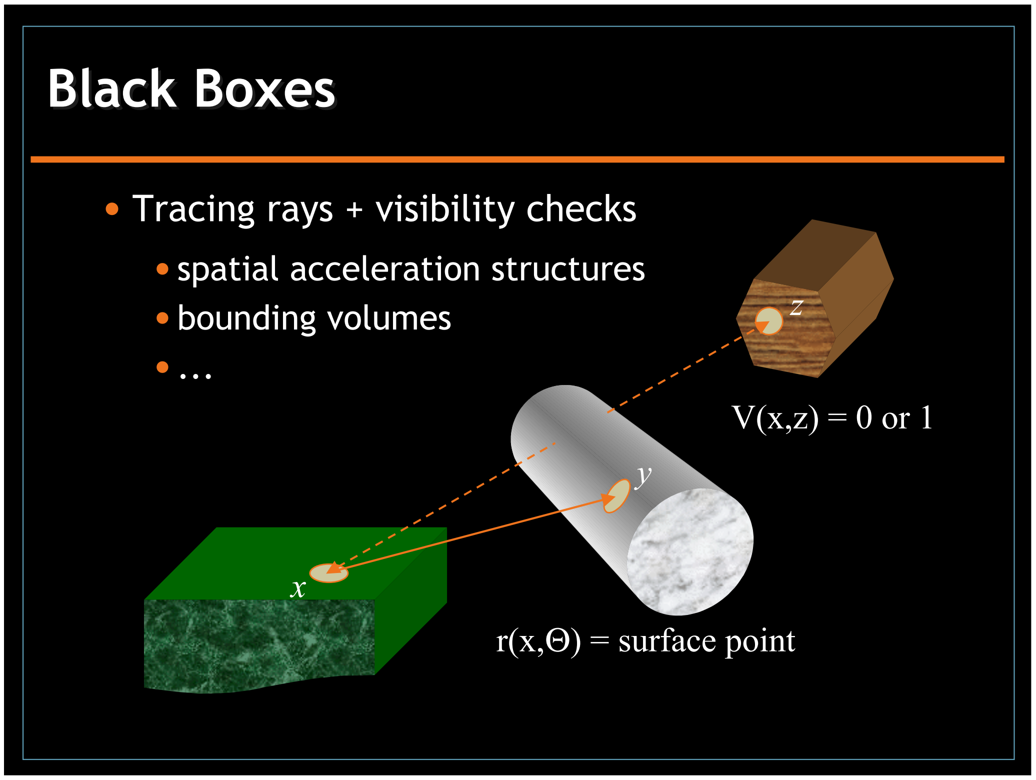
Black Boxes

- Light sources



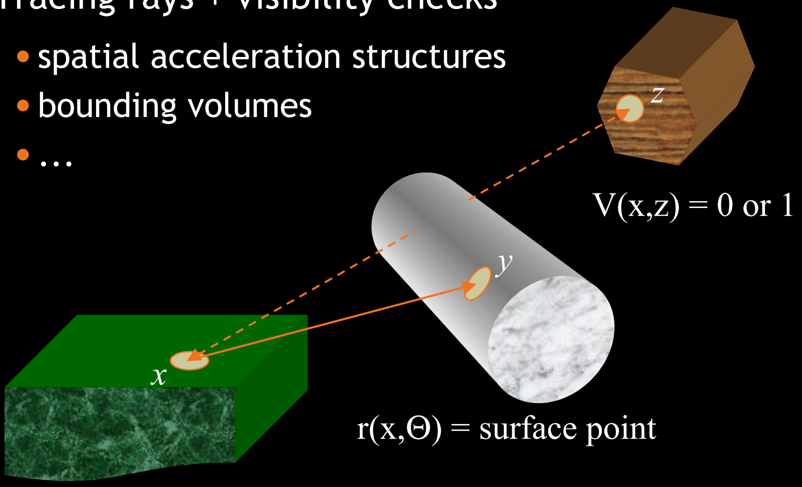
The queries we want to make about a point on a light source:

- What is the surface normal at this point?
- Given a direction, what is the self-emitted radiance value at this point?



Black Boxes

- Tracing rays + visibility checks
  - spatial acceleration structures
  - bounding volumes
  - …

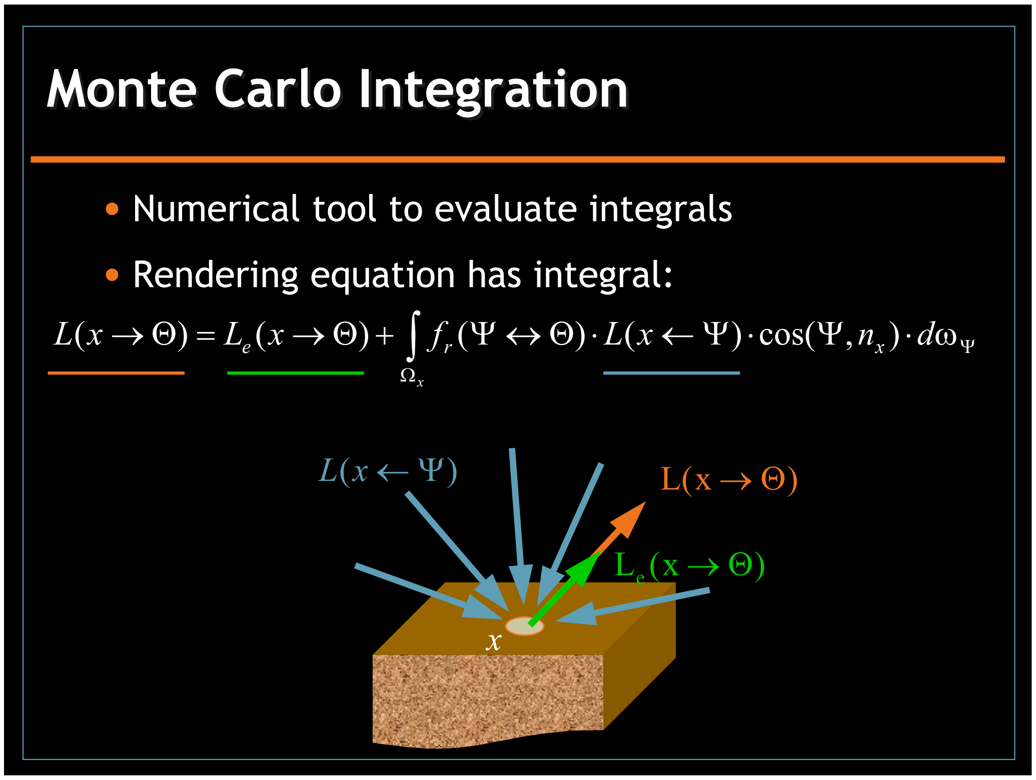


As we mentioned before, 2 types of visibility tests are needed:

• Given 2 points in 3D space, are the mutually visible? In other words, is any other solid object intersecting the line connecting those two points?

• Given a surface point and a direction, what is the nearest surface points one can see looking from the surface point in the given direction?

Both of these visibility queries need to be implemented, usually this is done using a hierarchy of bounding volumes or a spatial acceleration structure (e.g. a voxel-grid). Classic ray tracing literature gives many details on how this can be done efficiently.



Monte Carlo Integration

- Numerical tool to evaluate integrals
- Rendering equation has integral:

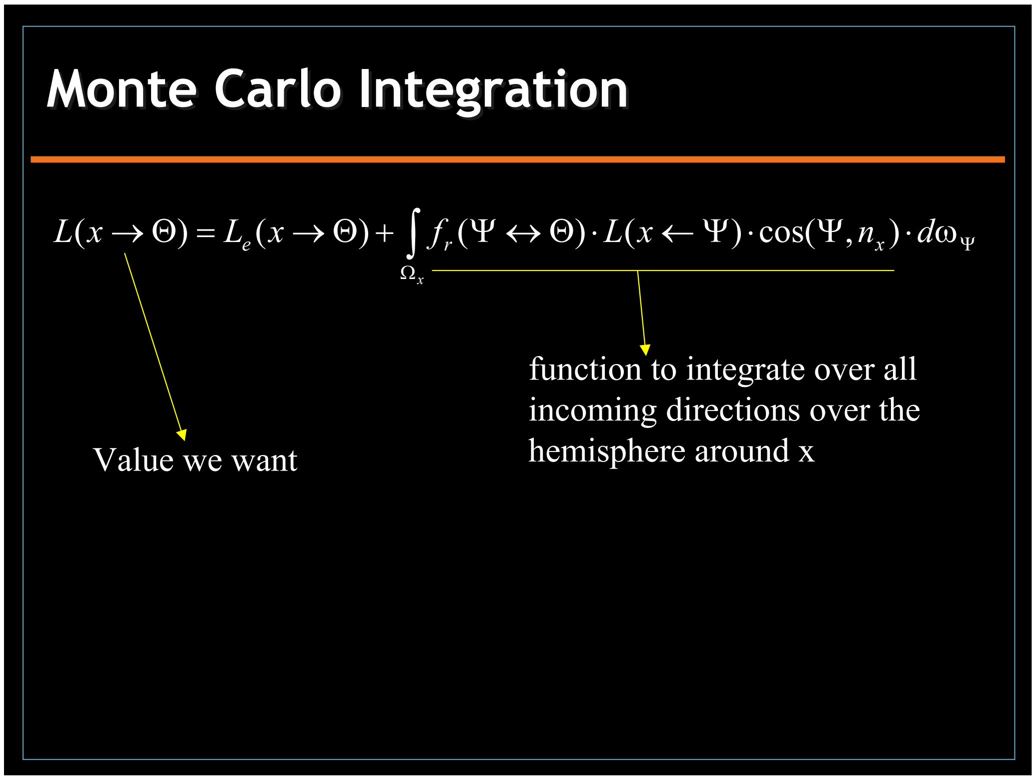
$$\underline{L(x \to \Theta)} = \underline{L_e(x \to \Theta)} + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot \underline{L(x \leftarrow \Psi)} \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

$L(x \leftarrow \Psi)$
$L(x \to \Theta)$
$L_e(x \to \Theta)$
x

Once the black boxes described previously are in place, we can start to look at solving the rendering equation itself.

The RE is a recursive integral equation. In practice, it cannot be solved analytically, so we need to find numerical ways to compute specific value of this equation in certain points and directions.

One of the most commonly used tool in global illumination to compute integrals is Monte Carlo integration.

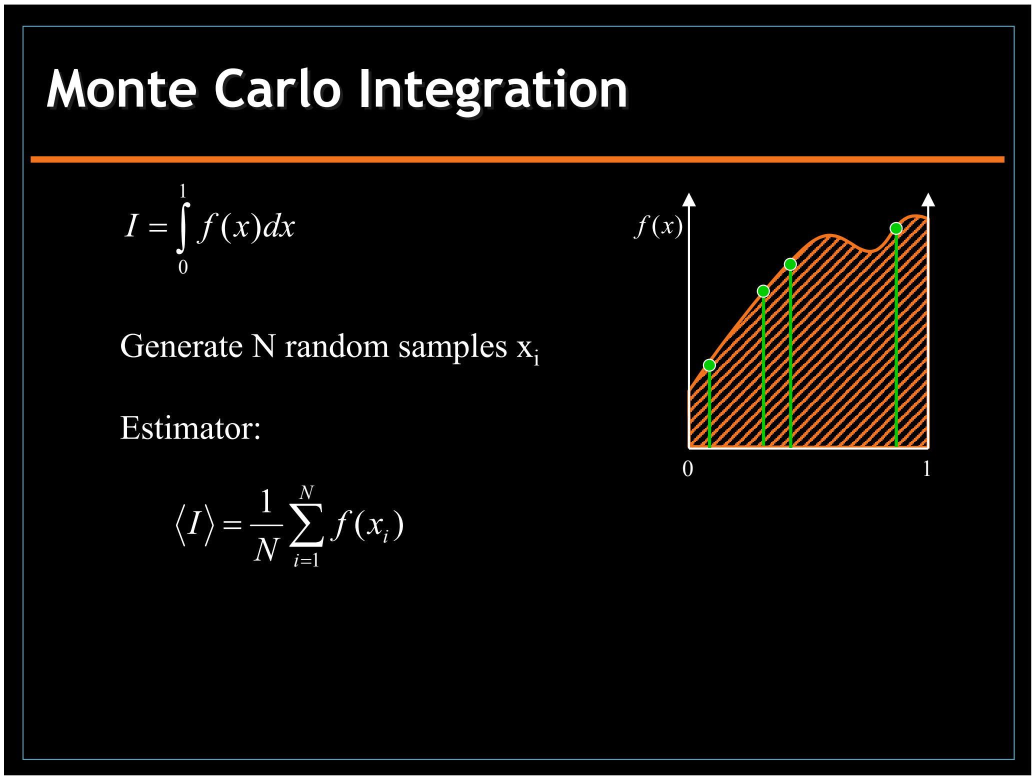


Monte Carlo Integration

$$\underbrace{L(x \to \Theta)}_{\text{Value we want}} = L_e(x \to \Theta) + \int_{\Omega_x} \underbrace{f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi}_{\text{function to integrate over all incoming directions over the hemisphere around x}}$$

In order to apply Monte Carlo integration to the RE, it is useful just to assume we have a function that needs to be integrated over some integration domain.

In our case, the integration domain is the hemisphere around point x. Using a parametrization of the hemisphere (e.g. hemispherical coordinates), we can rewrite this integral as a two-dimensional integral. The function to be evaluated is a recursive function, and that will give rise to a recursive Monte Carlo procedure.



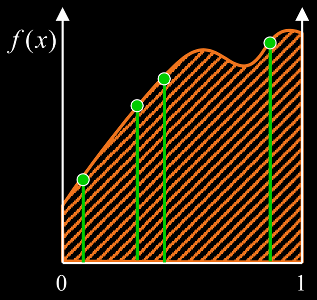
Monte Carlo Integration

$$I = \int_0^1 f(x)dx$$

Generate N random samples x_i

Estimator:

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

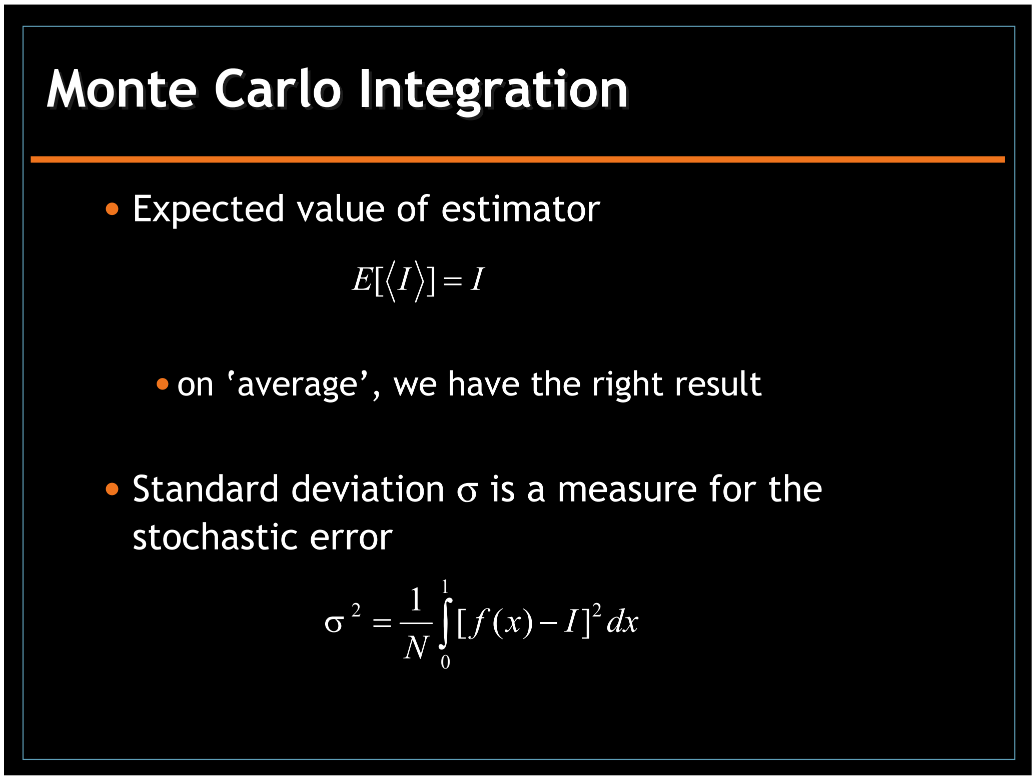


Let us look at a simple Monte Carlo scheme first.

Suppose we want to integrate a function f(x) over the integration domain [0,1]. Monte Carlo integration generates a number of random points in the integration domain, and takes the average values of their function values. This average value is an estimator for the value of the integral.

Note that the random points are sampled uniformly over the entire integration domain.

Since each time we will perform this computation, we will generate different random points, the estimator <I> will also be different. <I> is therefore a stochastic variable, with its own distribution.



Monte Carlo Integration

- Expected value of estimator

$$E[\langle I \rangle] = I$$

  - on 'average', we have the right result

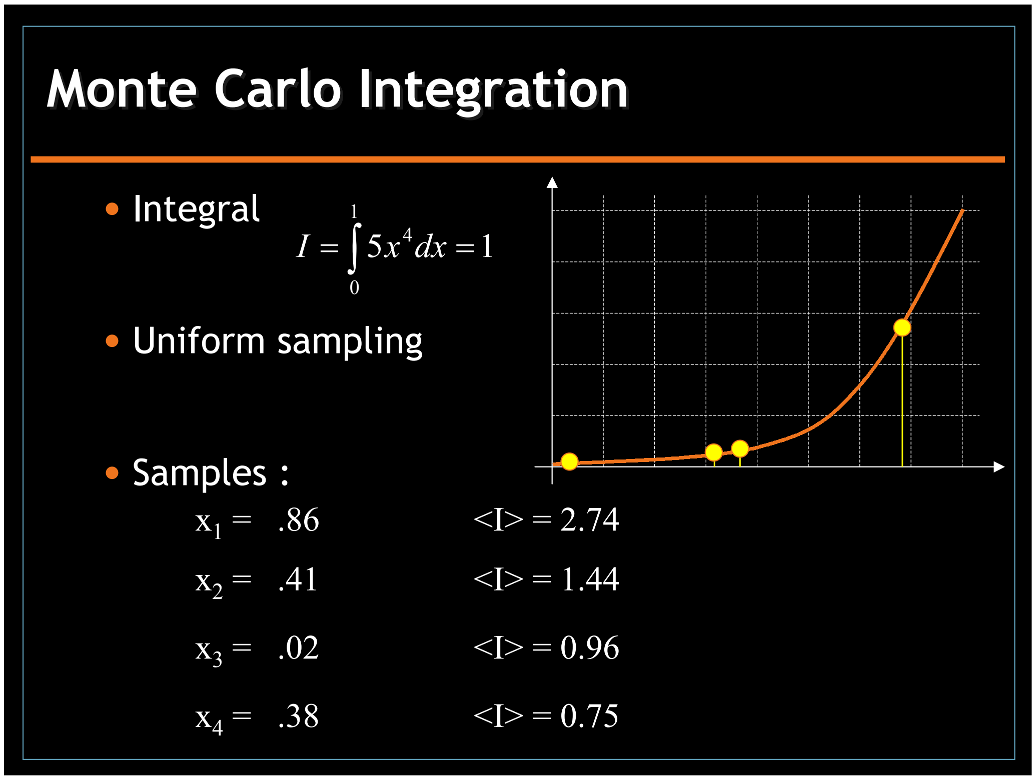
- Standard deviation σ is a measure for the stochastic error

$$\sigma^2 = \frac{1}{N}\int_0^1 [f(x) - I]^2 dx$$

The most useful property of <I> is, that the expected outcome of the Monte Carlo experiment is the value of the integral itself. Thus, if we repeat the Monte Carlo experiment an infinite number of times, and compute the average of all these experiments, we will end up exactly with I.

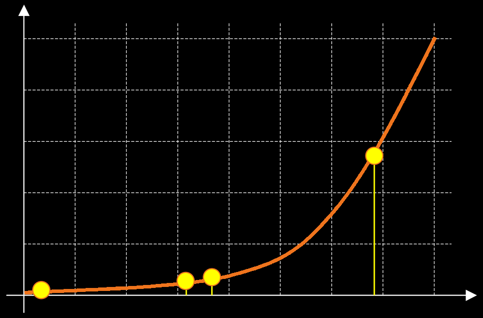
Intuitively, this means that 'on average', we will obtain a right answer. However, each individual value of <I> does not equal I. But the expected value of <I> equals I.

The standard deviation is a measure for the stochastic error. The real error can be larger than this, since we are working with a stochastic process. The more samples we generate, the lower the standard deviation becomes.



Monte Carlo Integration

- Integral $I = \int_0^1 5x^4 dx = 1$
- Uniform sampling



- Samples :

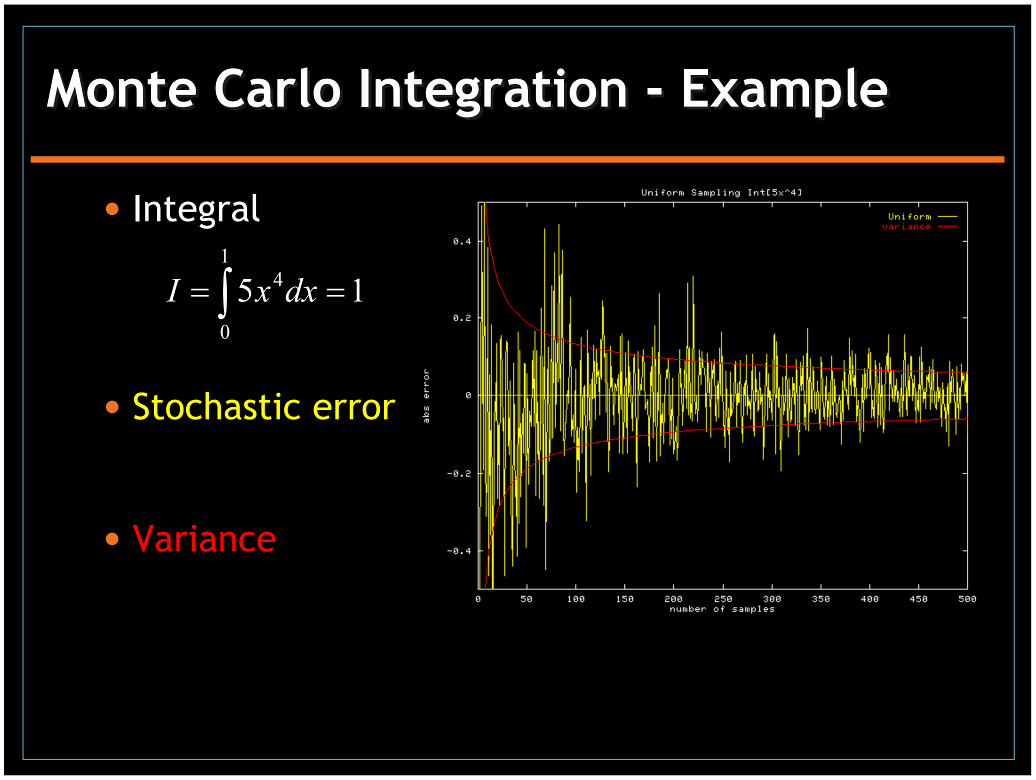
$x_1 =$ .86	$\langle I \rangle = 2.74$
$x_2 =$ .41	$\langle I \rangle = 1.44$
$x_3 =$ .02	$\langle I \rangle = 0.96$
$x_4 =$ .38	$\langle I \rangle = 0.75$

Let's take a look at a simple example.

Suppose we want to integrate a polynomial function 5x^4 over the integration domain [0,1]. The real value of this integral is 1, but what happens when we apply MC integration?

Suppose the first sample generates a point at x = 0.86. The our estimate so far is 2.74, or an error of 1.74.

We can go on drawing more samples, and the error we make will decrease on average.

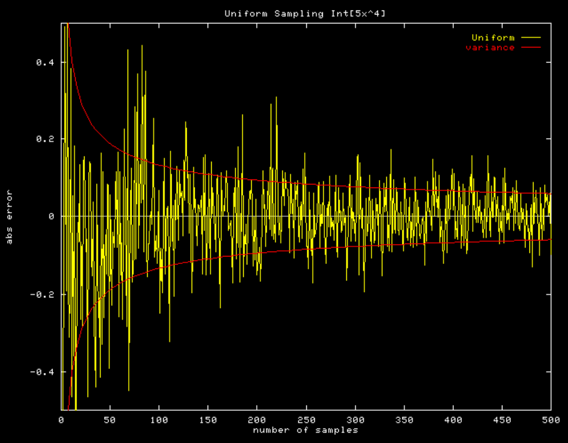


Monte Carlo Integration - Example

- Integral

$$I = \int_0^1 5x^4 dx = 1$$

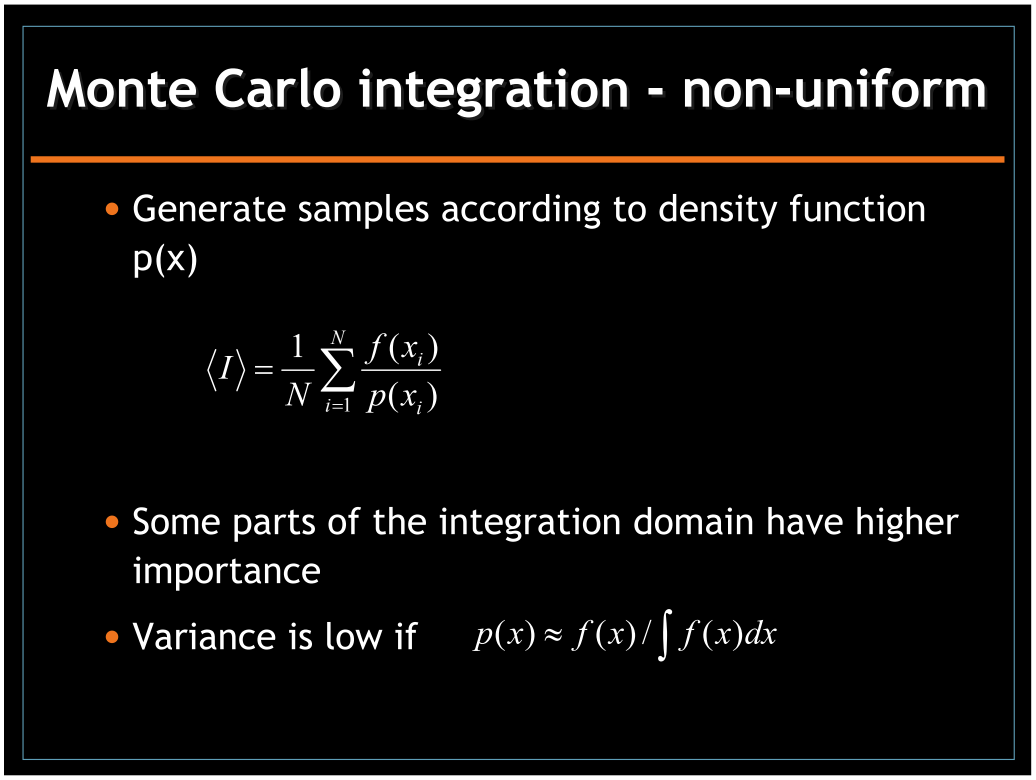
- Stochastic error
- Variance



This plot shows the estimated value of the integral in function of the number of samples.

The yellow line plots the error, the red line plots the theoretical standard deviation. So one can see that the error can become larger than the standard deviation, but the deviation is a good measure for the error.

It is also obvious that the more samples are drawn, the smaller the error becomes, although the error decreases rather slowly.



Monte Carlo integration - non-uniform

- Generate samples according to density function p(x)

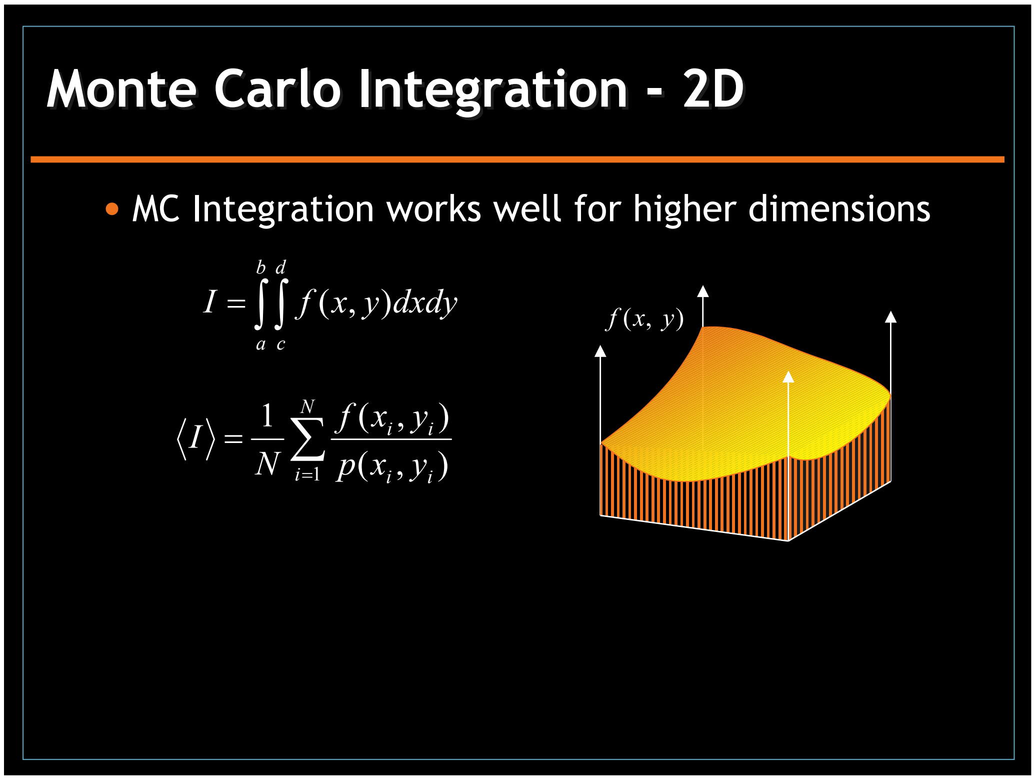
$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

- Some parts of the integration domain have higher importance
- Variance is low if $p(x) \approx f(x) / \int f(x)dx$

The previous example assumed that the samples were drawn uniformly over the integration domain.

One can also draw the samples non-uniformly, using a pdf p(x). Some parts of the domain will now be sampled more often than others, and this needs to be taken into account in the estimator.

By choosing a good pdf, one can decrease the variance. As a rule of thumb, the variance is low if the shape of the pdf matches the shape of the function to be integrated.

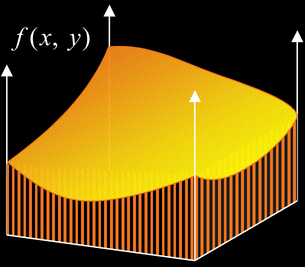


Monte Carlo Integration - 2D

- MC Integration works well for higher dimensions

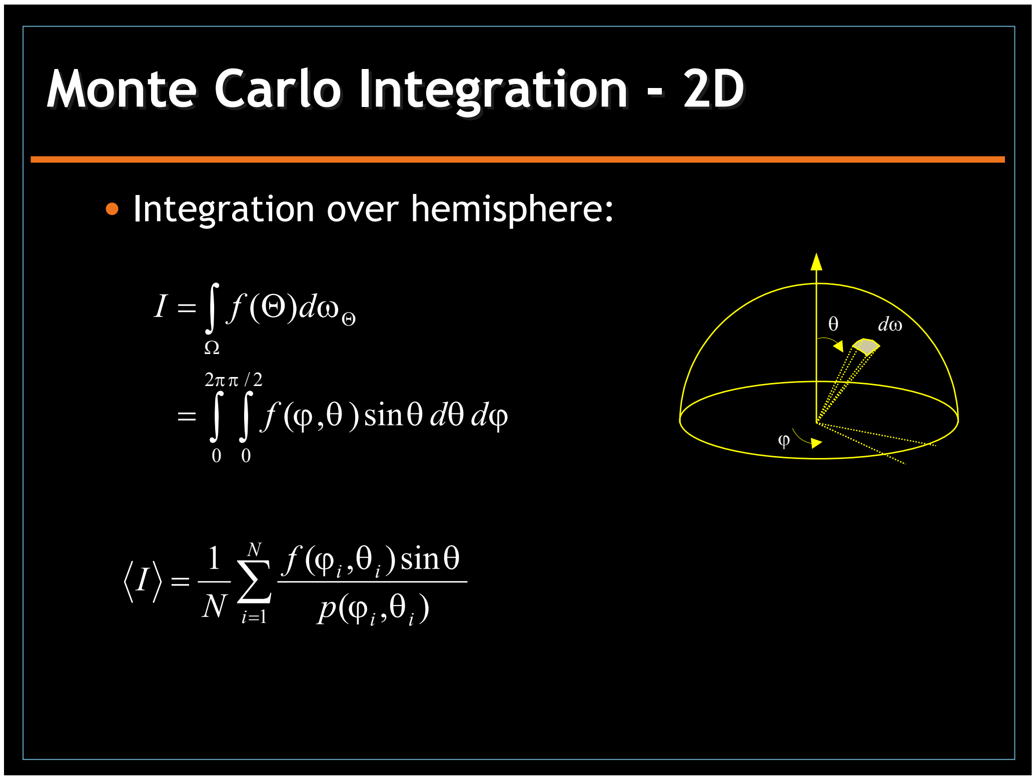
$$I = \int_a^b \int_c^d f(x,y)\,dx\,dy$$

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i, y_i)}{p(x_i, y_i)}$$



Monte Carlo integration really starts to pay of for multi-dimensional functions. The principle remains exactly the same: random points are generated in the integration domain, and an average function value is computed, weighted by the value of the PDF.

The stochastic error still decreases by the square root of 1/N. This is usually better than many deterministic integration methods for higher-dimensional functions.

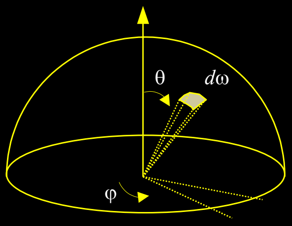


Monte Carlo Integration - 2D

- Integration over hemisphere:

$$I = \int_{\Omega} f(\Theta)\, d\omega_{\Theta}$$

$$= \int_0^{2\pi} \int_0^{\pi/2} f(\varphi,\theta) \sin\theta \, d\theta \, d\varphi$$

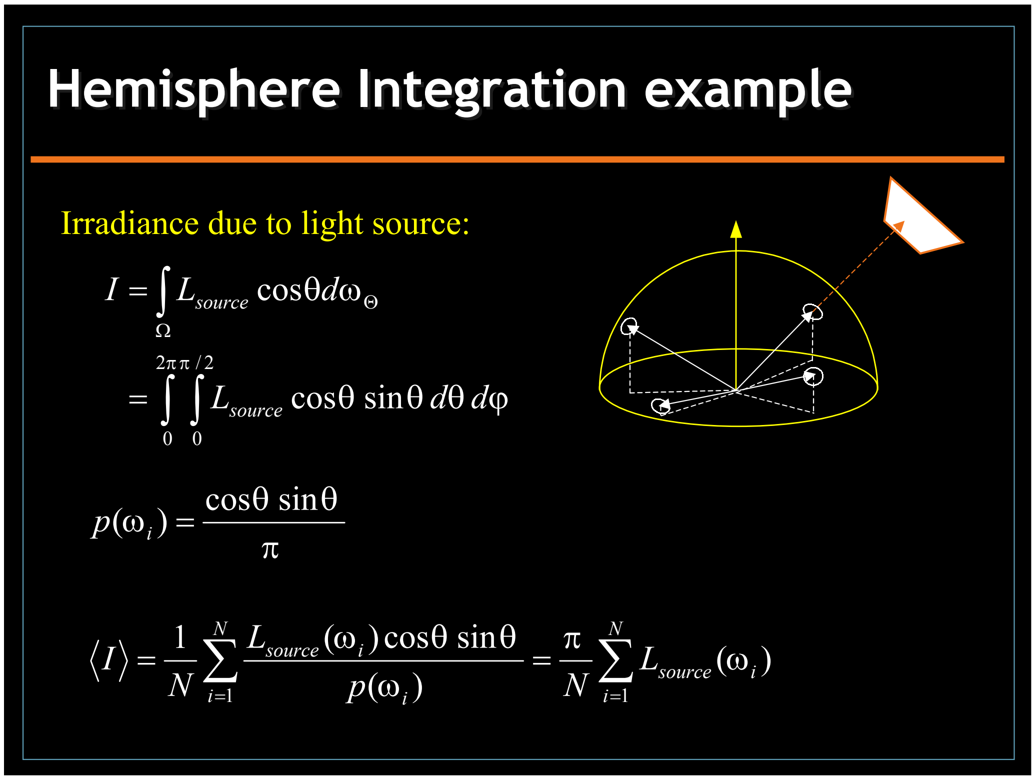


$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\varphi_i,\theta_i)\sin\theta}{p(\varphi_i,\theta_i)}$$

This slide shows an example of a Monte Carlo integration over the hemisphere.

Suppose we want to integrate a function f, defined for all directions over the hemisphere. First, one has to choose a parametrisation (e.g. hemispherical coordinates), and rewrite the integral. Now we can simply define any pdf in the phi – theta domain, generate samples, evaluate f, and compute the weighted average.

The expected value of this stochastic process is the exact value of the integral to be computed.



Hemisphere Integration example

Irradiance due to light source:

$$I = \int_{\Omega} L_{source} \cos\theta \, d\omega_{\Theta}$$

$$= \int_0^{2\pi} \int_0^{\pi/2} L_{source} \cos\theta \sin\theta \, d\theta \, d\varphi$$

$$p(\omega_i) = \frac{\cos\theta \sin\theta}{\pi}$$

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{L_{source}(\omega_i) \cos\theta \sin\theta}{p(\omega_i)} = \frac{\pi}{N} \sum_{i=1}^{N} L_{source}(\omega_i)$$

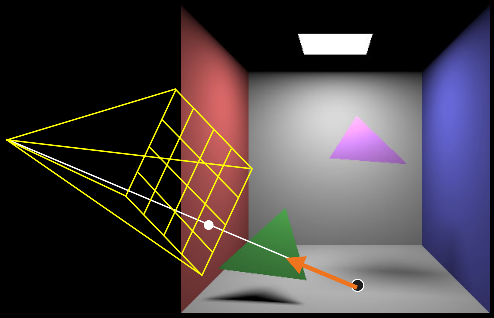
Here we have a more complicated example.

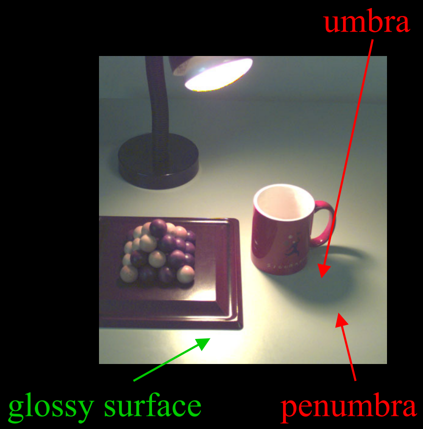
Suppose we want to compute the irradiance at a point due to a single light source.
Suppose the pdf samples a direction proportional to the cosine of the angle theta.

Then, by writing out the estimator, we obtain a very simple formula. Thus, we are generating direction according to a cosine distribution, and therefore only have to evaluate the radiance of the source for each sample. Note that the radiance of the source might be 0 if the sampled direction does not point to the light source.

Direct Illumination

- Paths of length 1 only, between receiver and light source

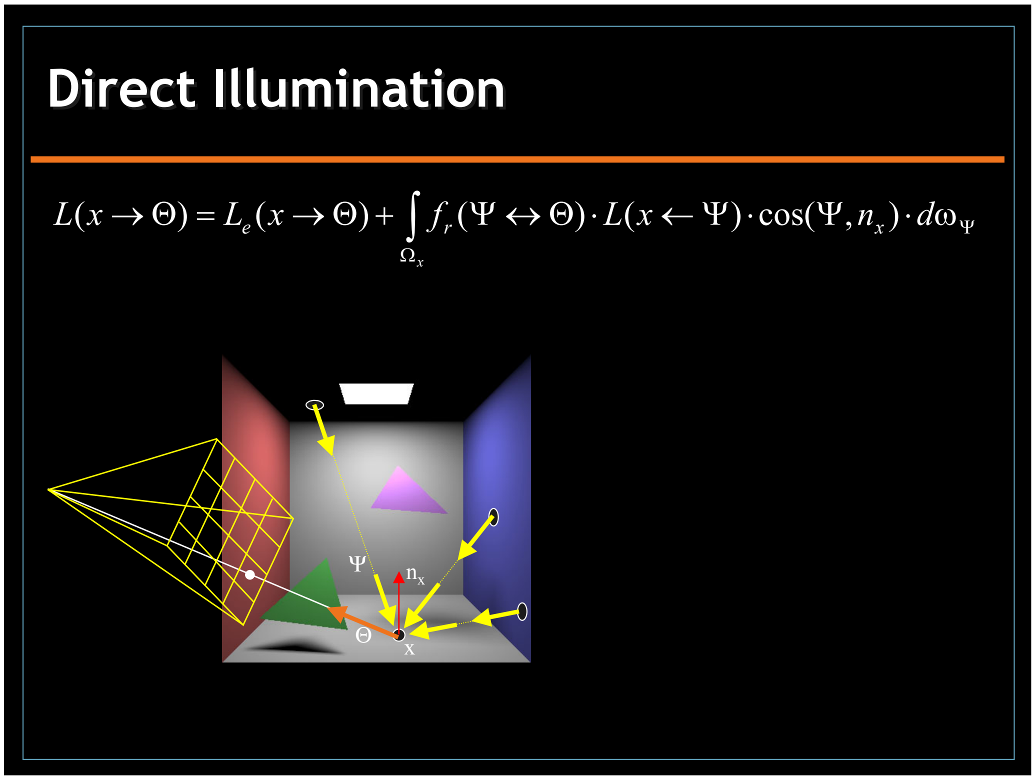




Let us apply the principles of Monte Carlo evaluation to the computation of the direct illumination.

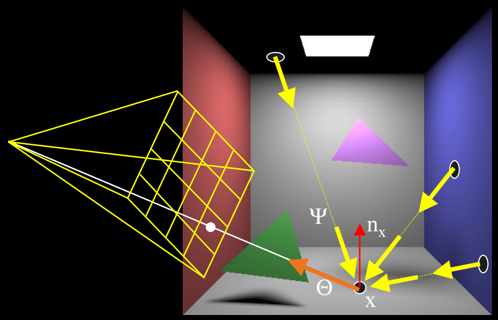
Direct illumination is only one component of the rendering equation. It describes the reflected radiance coming directly from the light sources.

Direct illumination is usually an important contribution for many images, so it is worthwhile using a specific sampling procedure.



Direct Illumination

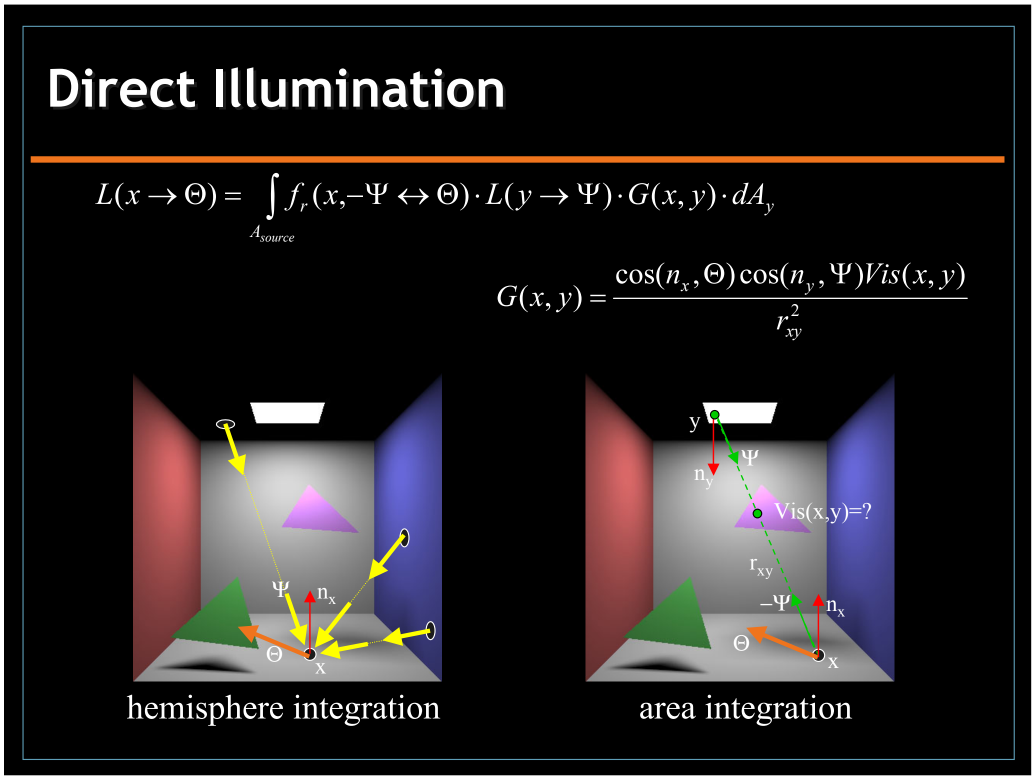
$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$



This is the general Rendering Equation. To compute the radiance at a single surface point in a given direction, we need to integrate the incoming radiance over the hemisphere, multiply by the correct BRDF and cosine factors.

If we would just apply MC integration to this integral, we end up sampling directions over the hemisphere. For each of these directions, we need to recursively evaluate the radiance along this new ray. But, since we are only interested in direct illumination, the majority of samples is probably wasted, since these rays will not connect to the light source.

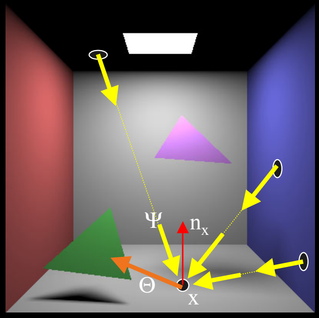
Although such an estimator would produce a valid and unbiased estimator, the variance will be high.



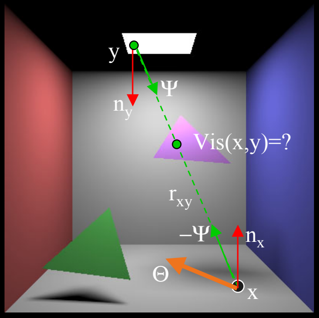
Direct Illumination

$$L(x \to \Theta) = \int_{A_{source}} f_r(x, -\Psi \leftrightarrow \Theta) \cdot L(y \to \Psi) \cdot G(x,y) \cdot dA_y$$

$$G(x,y) = \frac{\cos(n_x, \Theta)\cos(n_y, \Psi) Vis(x,y)}{r_{xy}^2}$$



hemisphere integration

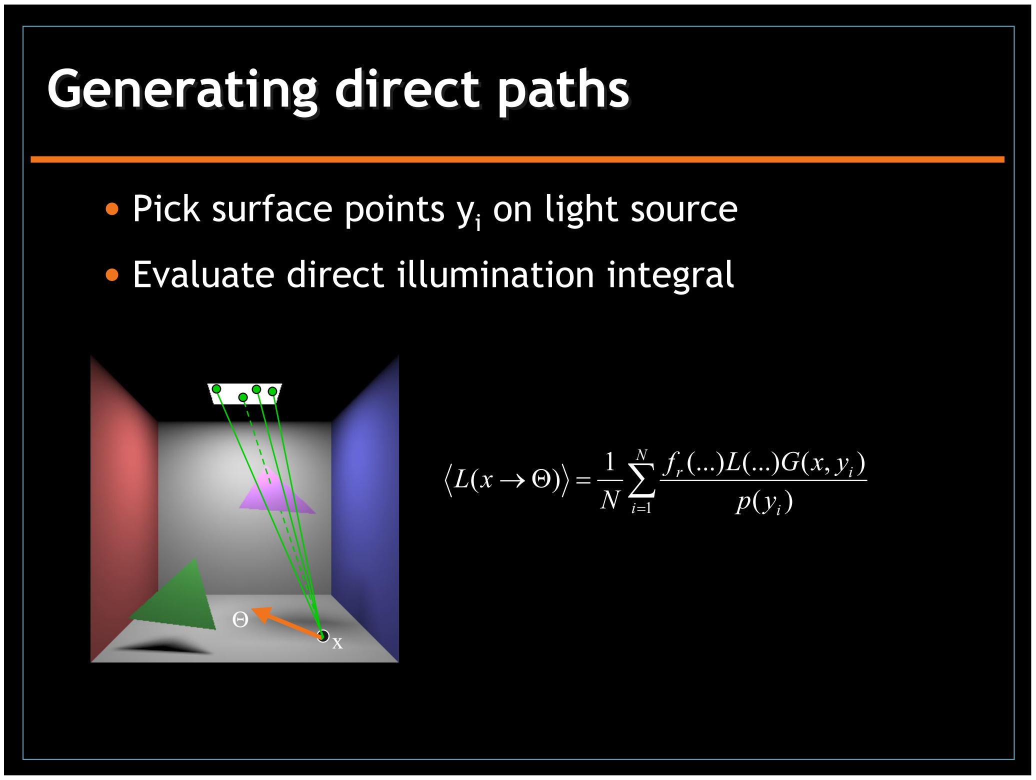


area integration

One can do better by reformulating the rendering equation for direct illumination. Instead of integrating over a hemisphere, we will integrate over the surface area of the light source. This is valid, since we are only interested in the contribution due to the light source.

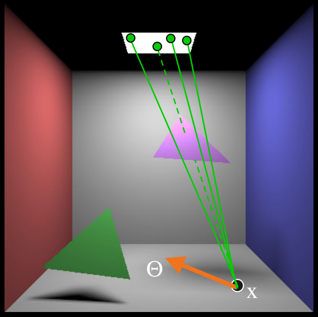
To transform the hemispherical coordinates to area coordinates over the hemisphere, we need to transform a differential solid angle to a differential surface. This introduces an extra cosine term and an inverse distance squared factor.

Additionally, the visibility factor, which was hidden in the hemispherical formulation since we 'traced' the ray to the closest intersection point, now needs to be mentioned explicitly.



Generating direct paths

- Pick surface points y_i on light source
- Evaluate direct illumination integral

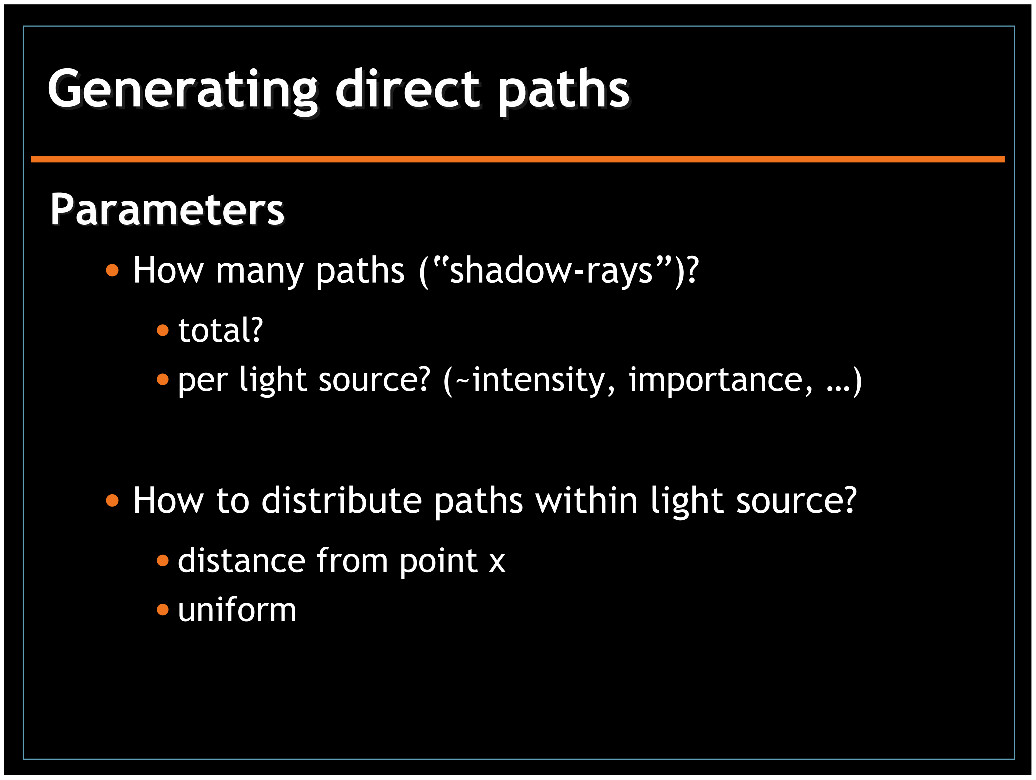


$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\ldots) L(\ldots) G(x, y_i)}{p(y_i)}$$

The complete Monte Carlo procedure for direct illumination now works as follows:

Pick random points on the light source, using an appropriate pdf. Then evaluate the function of the integral, which includes the geomtry term G, the BRDF, and the radiance of the light source.

One can see that this is in fact equivalent to traditional ‘shadow rays’ in classic ray tracing, except that we now have an unbiased, physically correct estimator of the radiometric quantity radiance. Classic ray tracing does not always take all these factors into account.



Generating direct paths

Parameters

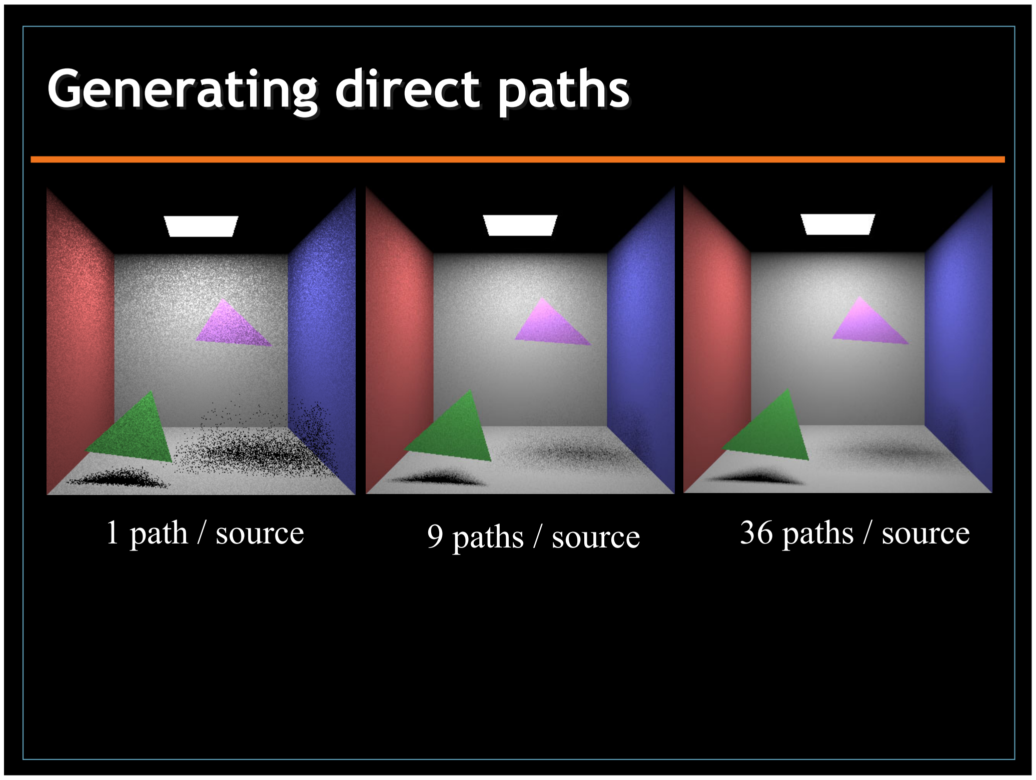
- How many paths ("shadow-rays")?
  - total?
  - per light source? (~intensity, importance, …)
- How to distribute paths within light source?
  - distance from point x
  - uniform

To compute the direct illumination using Monte Carlo integration, the following parameters can now be chosen:

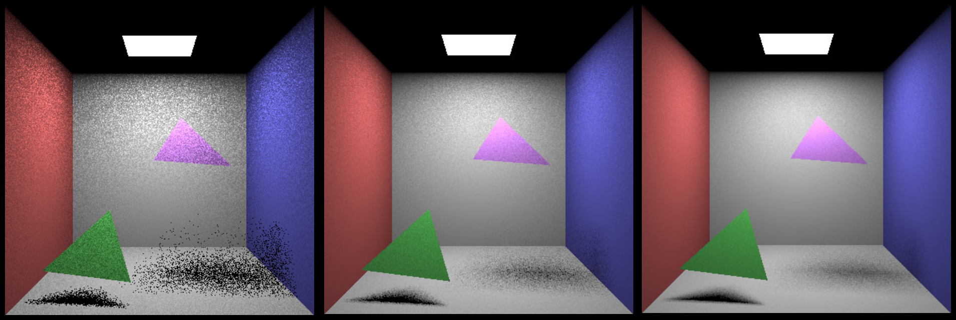
- How many paths will be generated total for each radiance value to be computed? More paths result in a more accurate estimator, but the computational cost increases.

- How many of these paths will be send to each light source? It is intuitively obvious that one wants to send more paths to bright light sources, closer light sources, visible light sources.

- How to distribute the paths within each light source? When dealing with large light sources, points closer to the point to be shaded are more important than farther-away points.



Generating direct paths

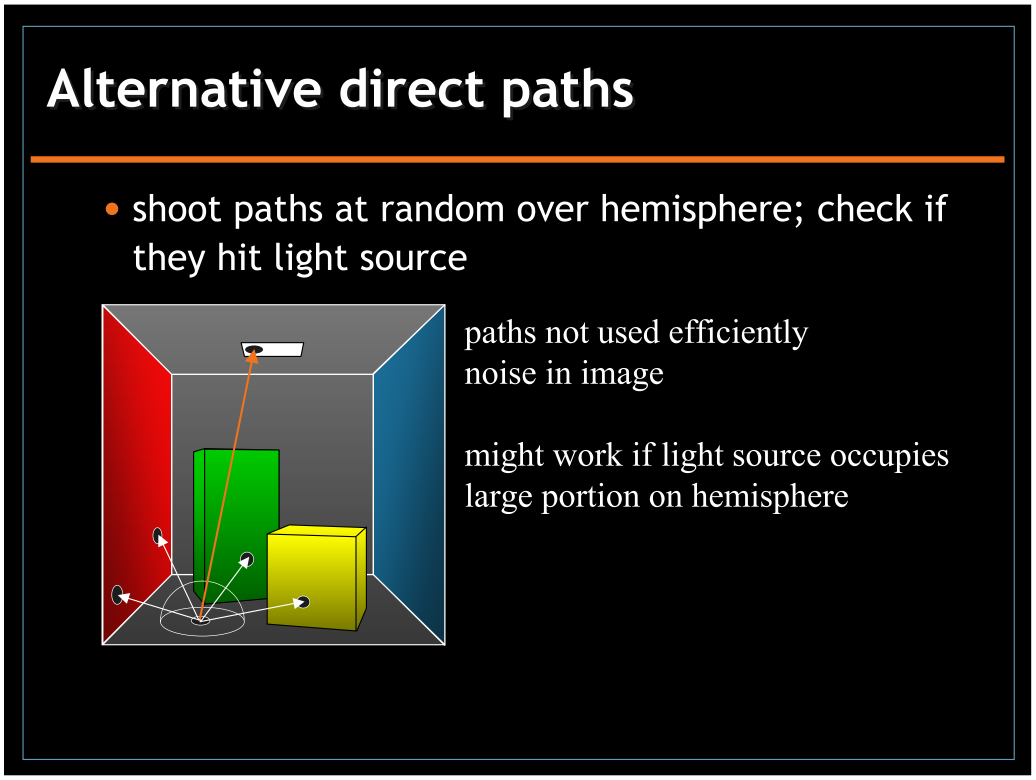


1 path / source   9 paths / source   36 paths / source

Here are a few examples of the results when generating a different number of paths per light source.

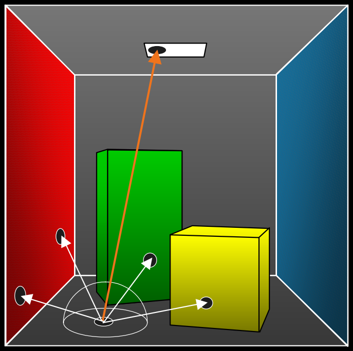
This simple scene has only one light source, and respectively 1, 9 and 36 paths are generated. The radiance values are computed more accurately in the latter case, and thus visible noise is less objectionable.

Although the first image is unbiased, its stochastic error is much higher compared to the last picture.



Alternative direct paths

- shoot paths at random over hemisphere; check if they hit light source



paths not used efficiently
noise in image

might work if light source occupies large portion on hemisphere

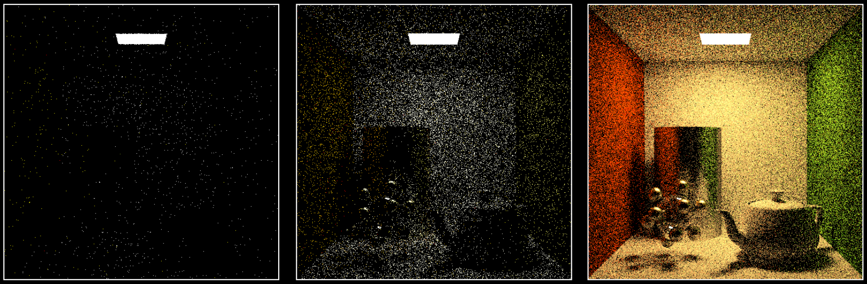
The algorithm in which the area of the light source is sampled is the most widely used way of computing direct illumination.

However, many more ways are possible, all based on a Monte Carlo evaluation of the rendering equation.

This slide shows an algorithm we have shown before: directions are sampled over the hemisphere, and they are traced to see whether they hit the light source and contribute to the radiance value we are interested in.

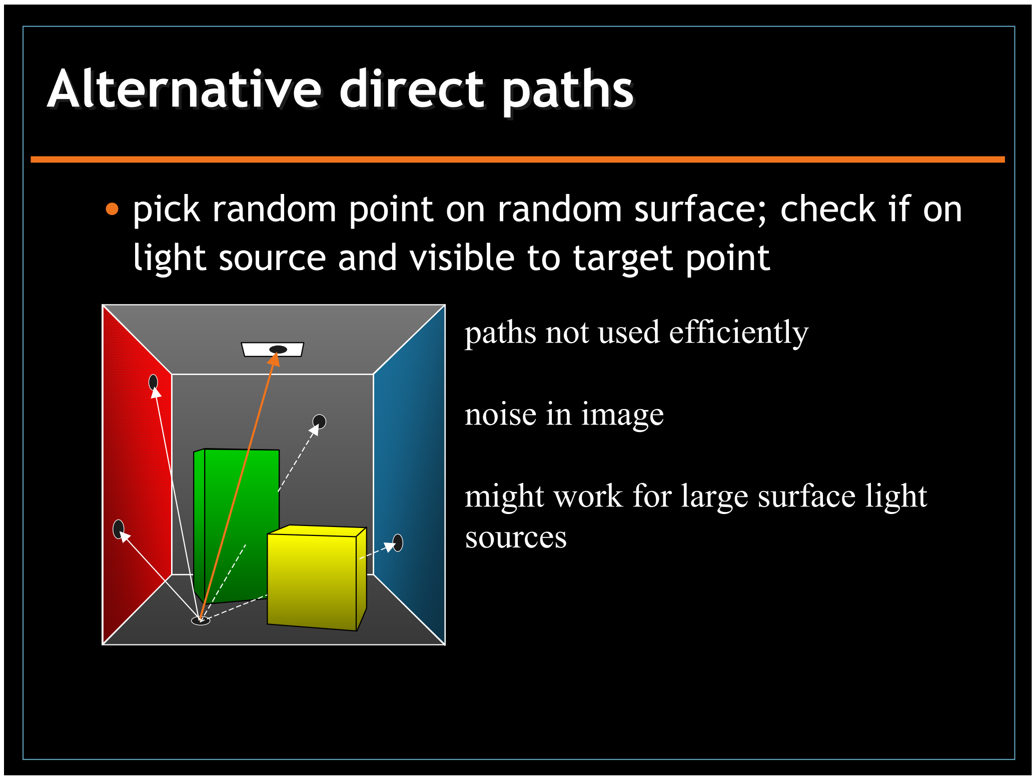
In this approach, many samples are wasted since their contribution is 0.

Alternative direct paths



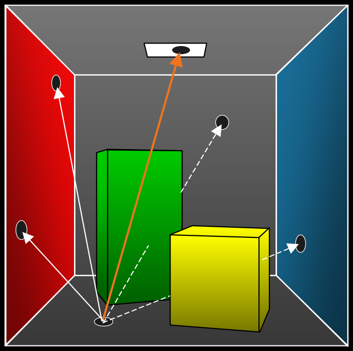
1 paths / point   16 paths / point   256 paths / point

These images show the result of hemispherical sampling. As can be expected, many pixels are black when using only 1 sample, since we will only have a non-black pixel if the generated direction points to the light source.



Alternative direct paths

- pick random point on random surface; check if on light source and visible to target point



paths not used efficiently

noise in image

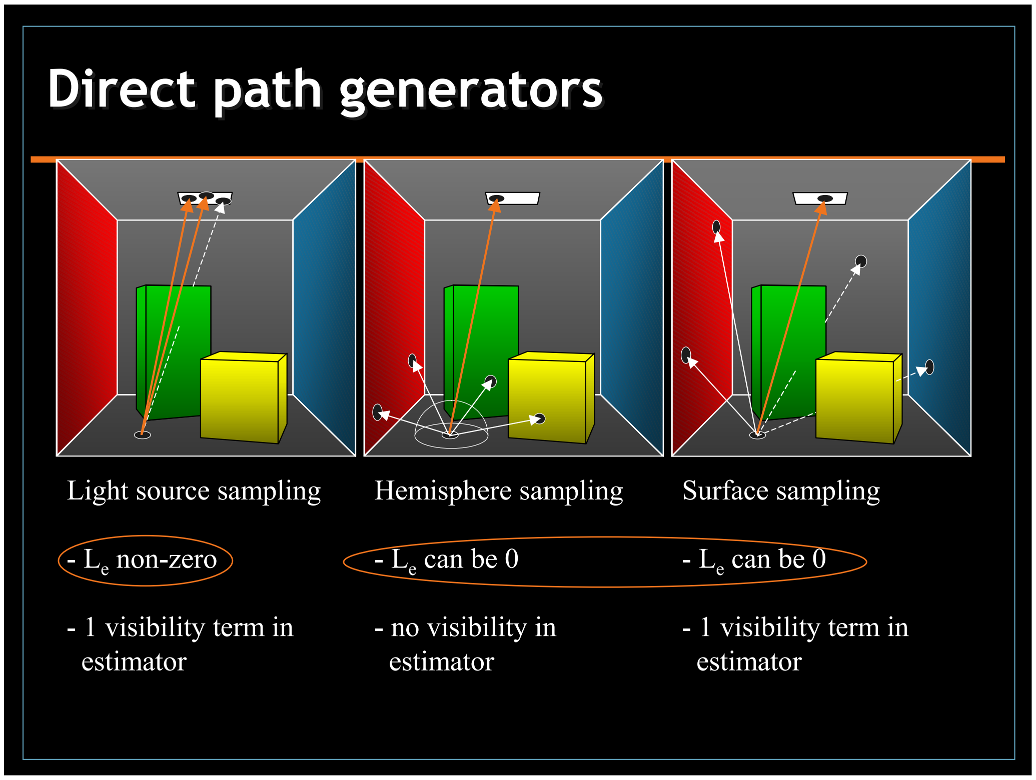
might work for large surface light sources

This is another algorithm for direct illumination:

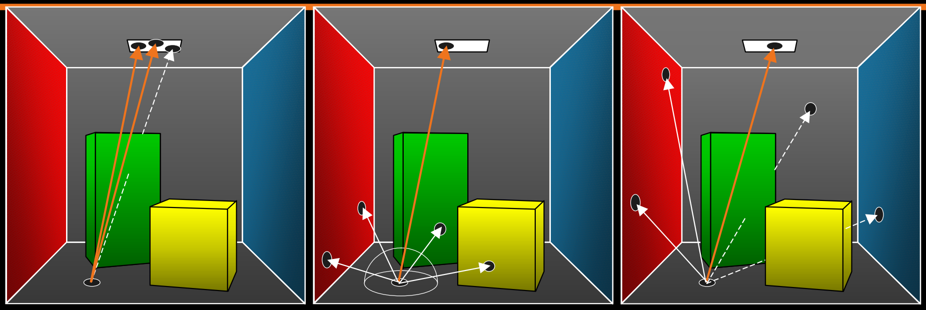
We can write the rendering equation for as an integral over ALL surfaces in the scene, not just the light sources. Of course, the direct illumination contribution of most of these surfaces will be 0.

A Monte Carlo procedure will then sample a random surface point. For each of these surface points, we need to evaluate the self-emitted radiance (only different from 0 when a light source), the visibility between the sampled point and the target point, the geometry factor, and the BRDF.

Since both the self-emitted radiance and the visibility term might produce a 0 value in many cases, many of the samples will be wasted.



Direct path generators



Light source sampling	Hemisphere sampling	Surface sampling
- L_e non-zero	- L_e can be 0	- L_e can be 0
- 1 visibility term in estimator	- no visibility in estimator	- 1 visibility term in estimator

Here we see the 3 different approaches next to each other.

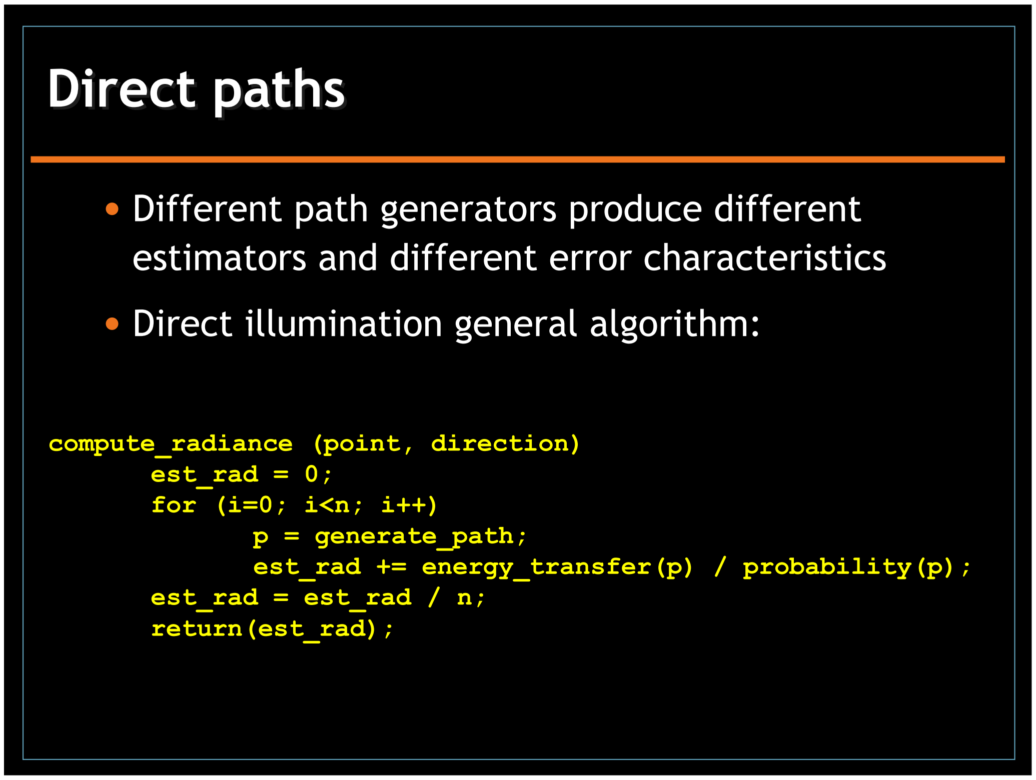
The noise resulting from each of these algorithms has different causes.

When sampling the area of the light source, most of the noise will come from failed visibility tests, and a little noise from a varying geometry factor.

When sampling the hemisphere, most noise comes from the self-emitted radiance being 0 on the visible point, but the visibility itself does not cause noise. However, each sample is more costly to evaluate, since the visibility is now folded into the ray tracing procedure.

When sampling all surfaces in the scene, noise comes failed visibility checks AND self-emitted radiance being 0. So this is obviously the worst case for computing direct illumination.

Although all these algorithms produce unbiased images when using enough samples, the efficiency of the algorithms is obviously different.



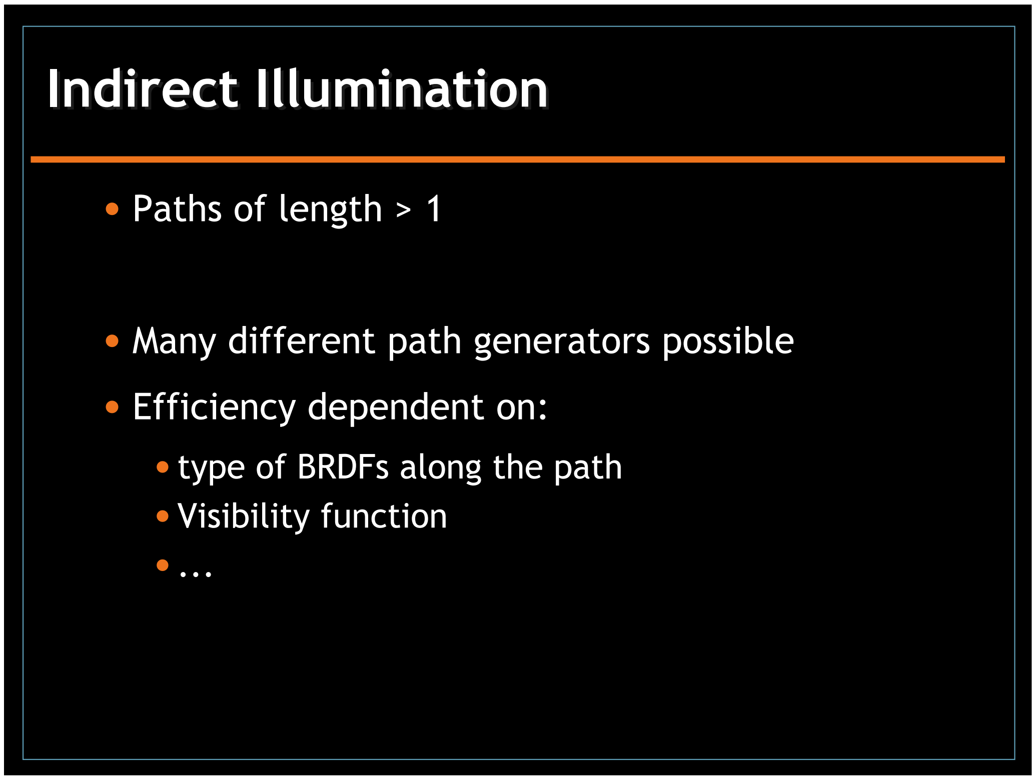
Direct paths

- Different path generators produce different estimators and different error characteristics
- Direct illumination general algorithm:

```
compute_radiance (point, direction)
       est_rad = 0;
       for (i=0; i<n; i++)
              p = generate_path;
              est_rad += energy_transfer(p) / probability(p);
       est_rad = est_rad / n;
       return(est_rad);
```

A general MC algorithm for computing direct illumination then generates a number of paths, evaluates for each path the necessary energy transfer along the path (radiance * BRDF * geometry), and computes the weighted average.

The differences in different algorithms lie in how efficient the paths are w.r.t. energy transfer.



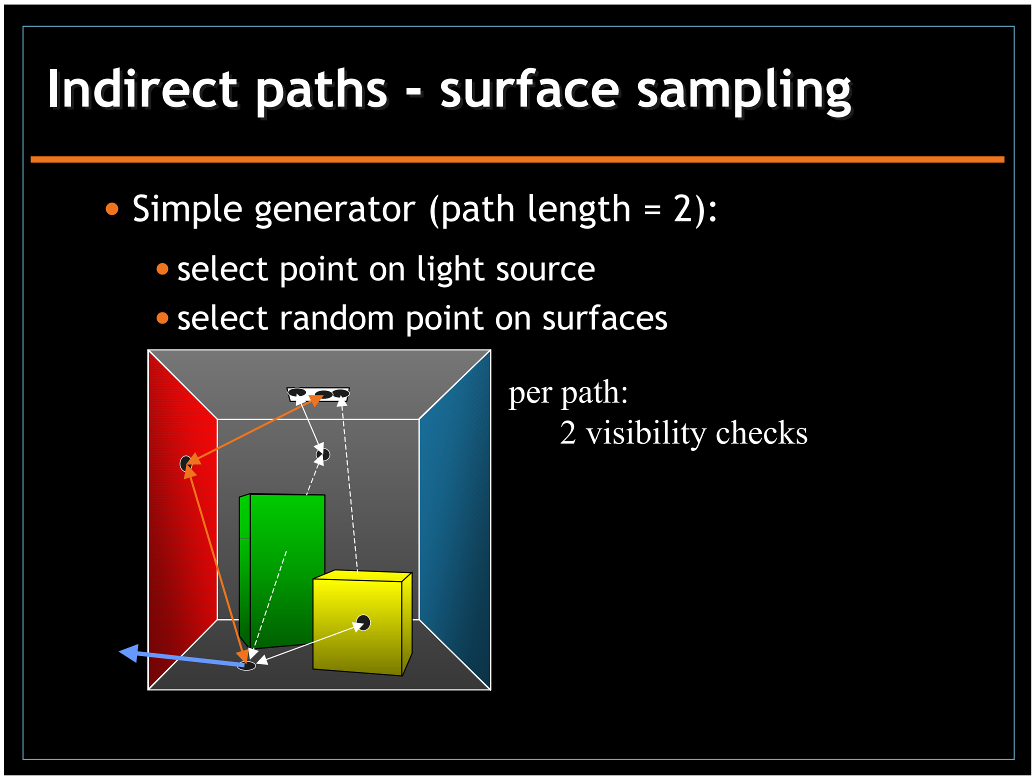
Indirect Illumination

- Paths of length > 1
- Many different path generators possible
- Efficiency dependent on:
  - type of BRDFs along the path
  - Visibility function
  - ...

What about indirect illumination?

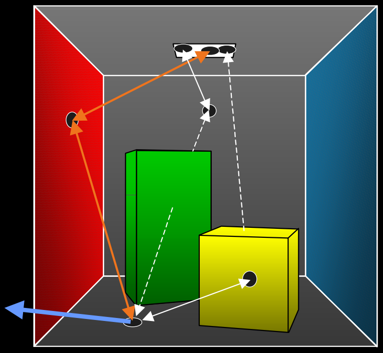
The principle remains exactly the same: we want to generate paths between a light source and a target point. The only difference is that the path will be of length greater than 1.

Again, the efficiency of the algorithm will depend on how clever the most useful paths can be generated.



Indirect paths - surface sampling

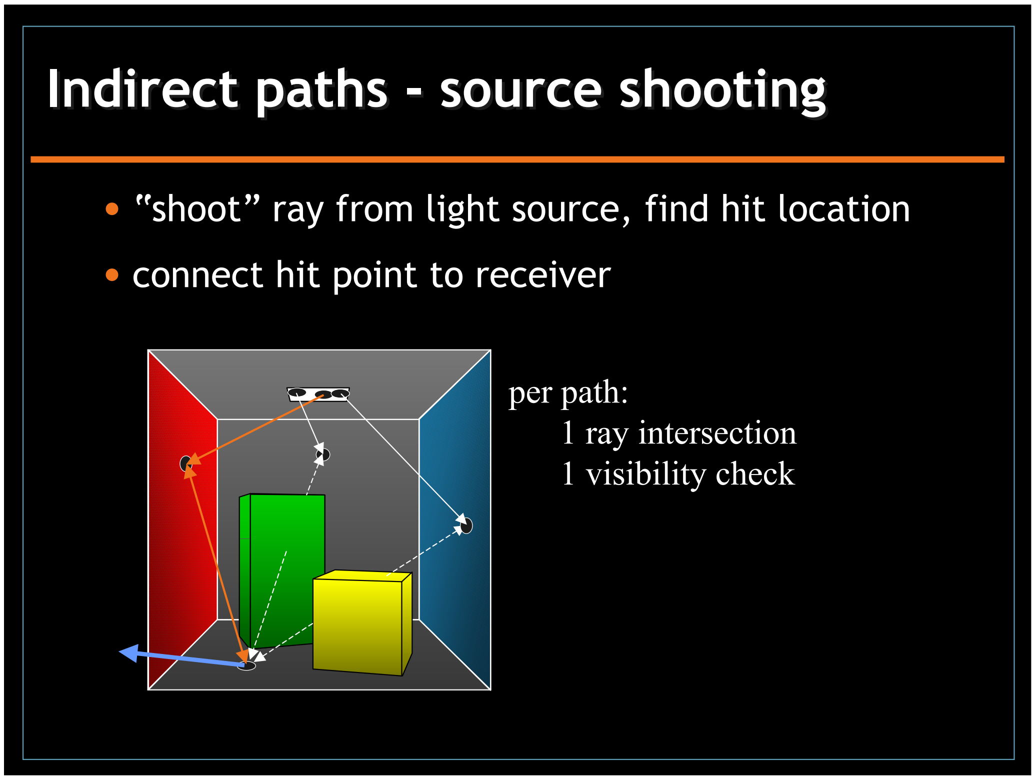
- Simple generator (path length = 2):
  - select point on light source
  - select random point on surfaces



per path:
2 visibility checks

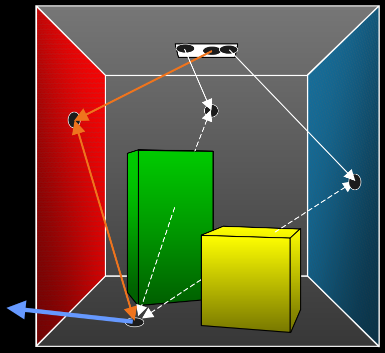
An added complexity is that we now have to deal with recursive evaluations. Although we show in these slides only the final paths between the light source and the target point, in an actual algorithm these paths will be generated recursively.

A simple algorithm involves samples all surface points in the scene. To generate paths of length 2, one can generate a random point on the surfaces, and a random point on a light source (direct illumination for the intermediate point). The necessary energy transfer is computed along the paths, and a weighted average using the correct pdf's is computed.



Indirect paths - source shooting

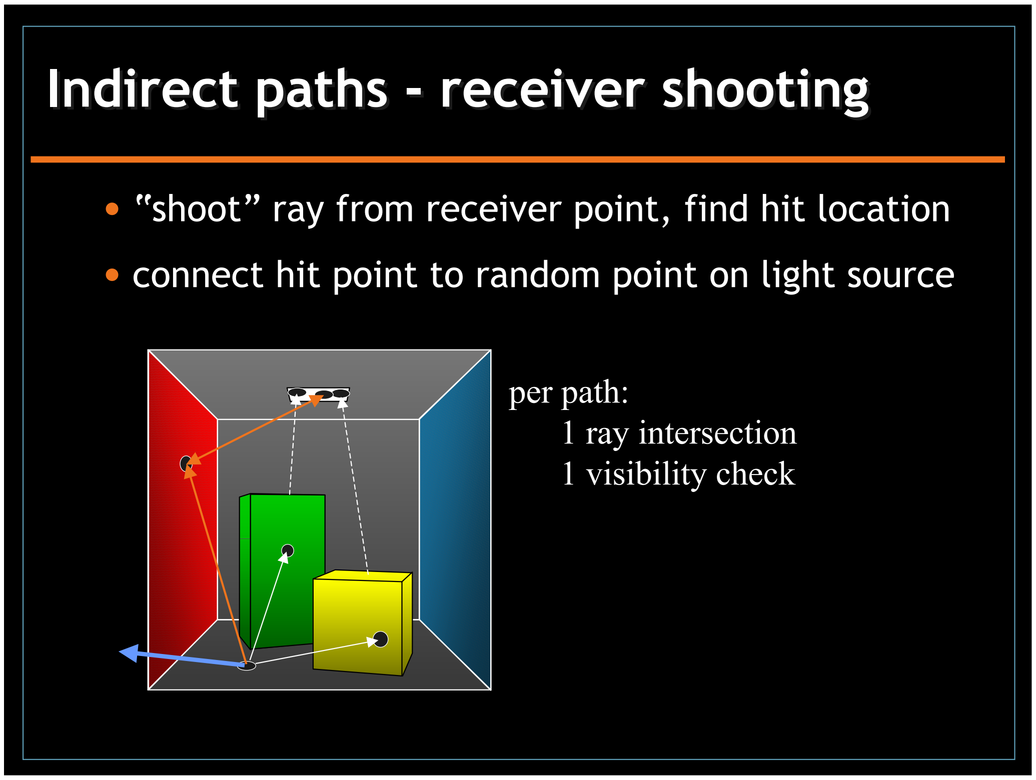
- "shoot" ray from light source, find hit location
- connect hit point to receiver



per path:
 1 ray intersection
 1 visibility check

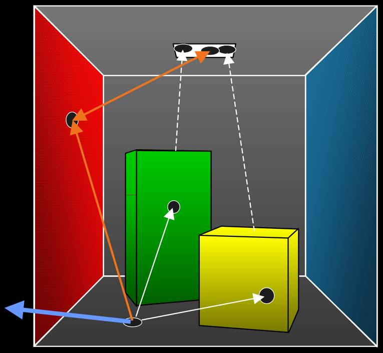
This algorithm might generate the intermediate point in a slightly different way: a random direction is sampled over the hemisphere around a random point on the light source, this ray is traced in the environment, and the closest intersection point found.

Then this visible point is connected to the target point.



Indirect paths - receiver shooting

- "shoot" ray from receiver point, find hit location
- connect hit point to random point on light source

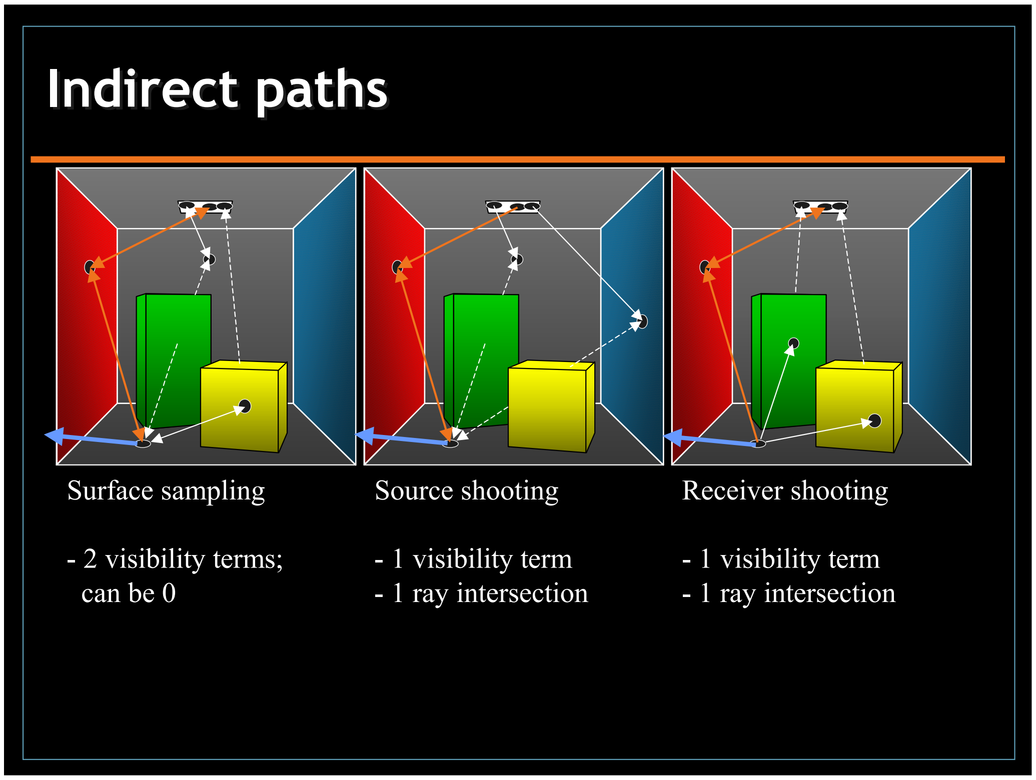


per path:
    1 ray intersection
    1 visibility check

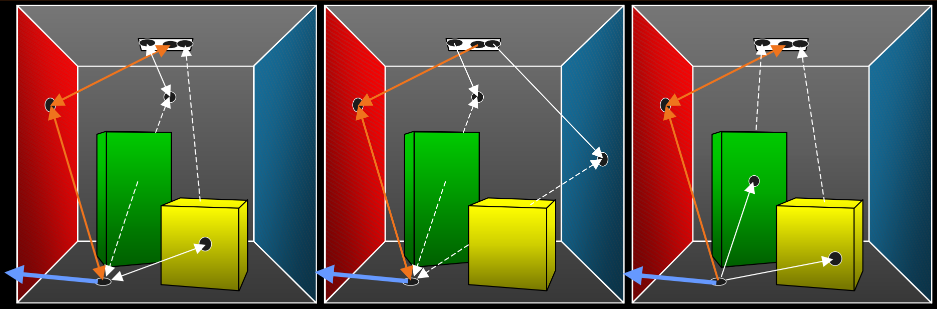
Another algorithm might generate the intermediate point in a slightly different way: a random direction is sampled over the hemisphere around the target point, this ray is traced in the environment, and the closest intersection point found.

Then this visible point is connected to a random surface point generated on the light source.

This is the usual way of generating indirect paths in stochastic ray tracing.



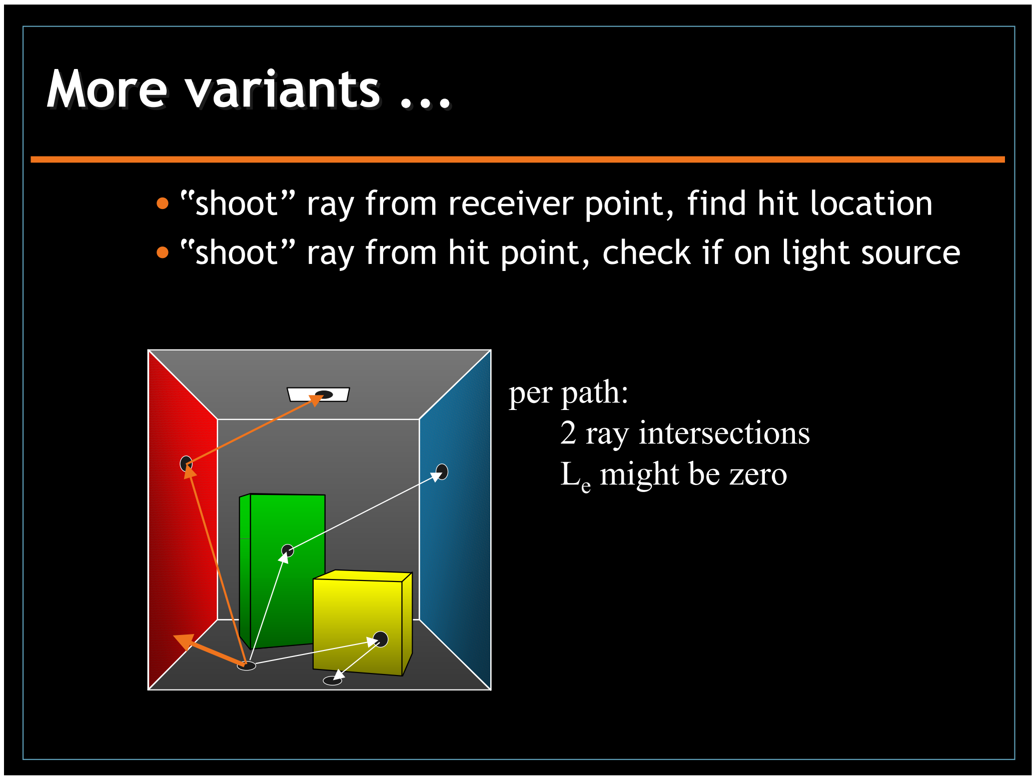
Indirect paths



Surface sampling	Source shooting	Receiver shooting
- 2 visibility terms; can be 0	- 1 visibility term - 1 ray intersection	- 1 visibility term - 1 ray intersection

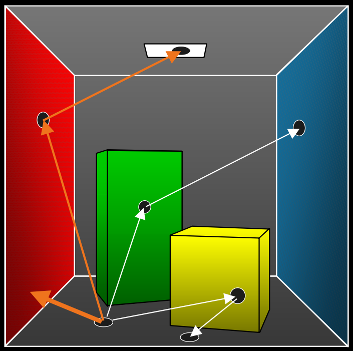
Here are all the different approaches compared.

All three of these algorithms will produce an unbiased image when generating enough samples, but the efficiency will be very different.



More variants ...

- "shoot" ray from receiver point, find hit location
- "shoot" ray from hit point, check if on light source



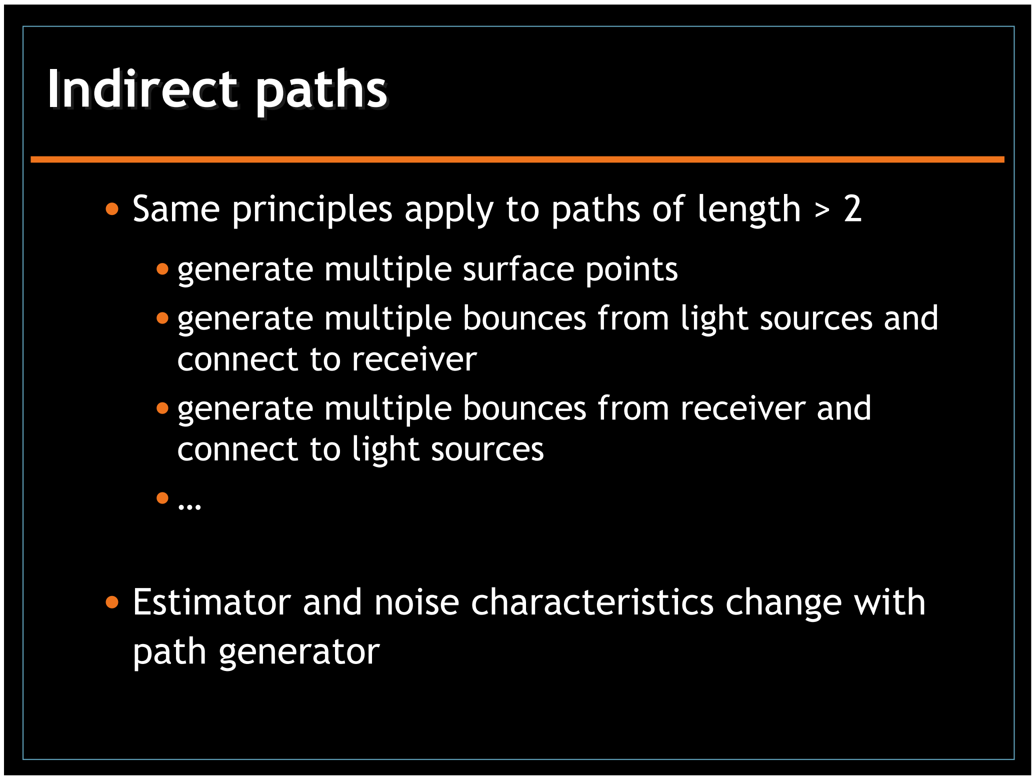
per path:

2 ray intersections

L_e might be zero

Even more variants can be thought of, as shown on this slide.

This is just to illustrate the general principle, that any path generator will do, as long as the correct energy transfer and correct probabilities for all the paths are computed.

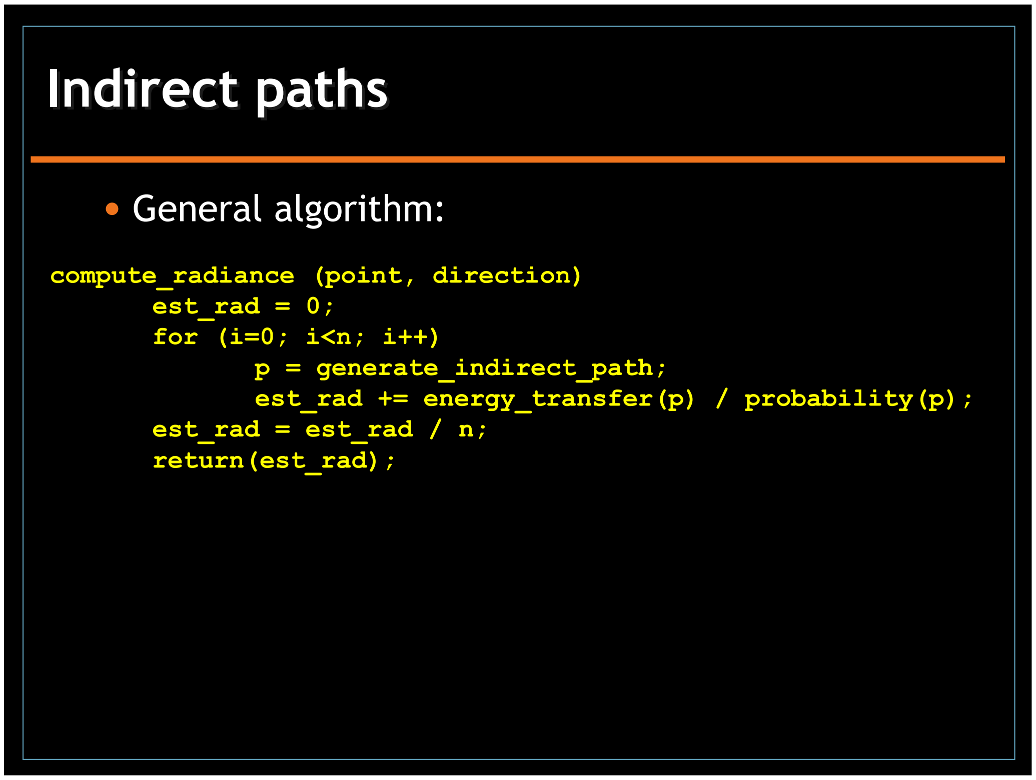


Indirect paths

- Same principles apply to paths of length > 2
  - generate multiple surface points
  - generate multiple bounces from light sources and connect to receiver
  - generate multiple bounces from receiver and connect to light sources
  - …
- Estimator and noise characteristics change with path generator

For paths of length greater than 2, one can also come up with a lot of different path generators.

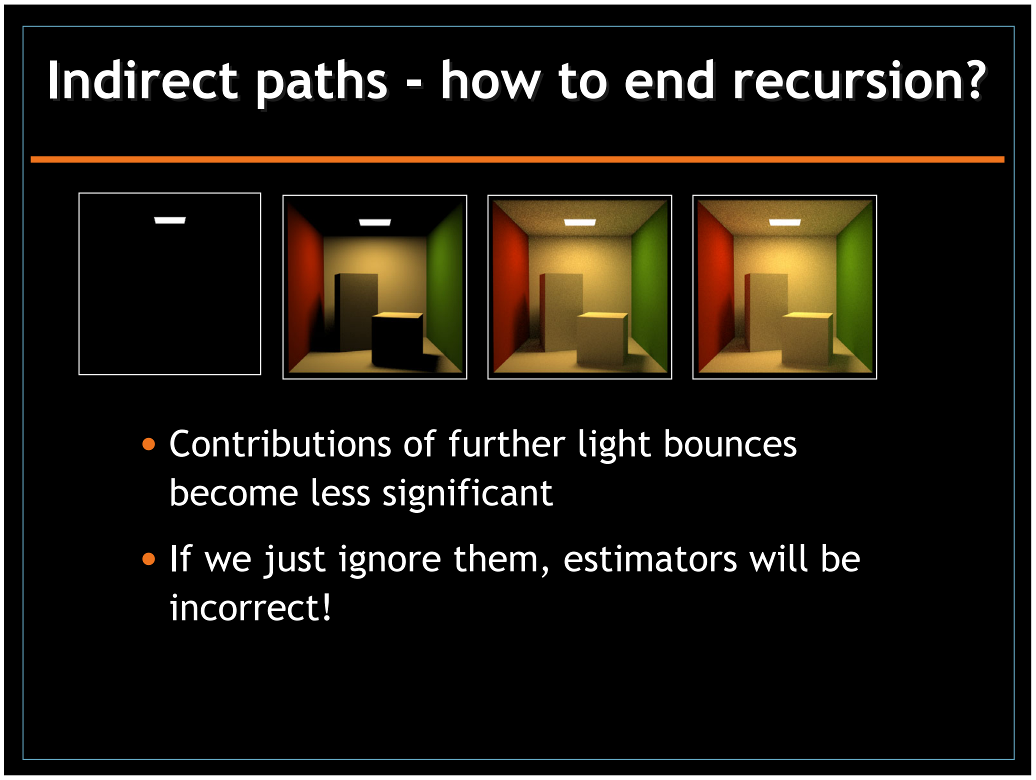
Usually these are implemented recursively.



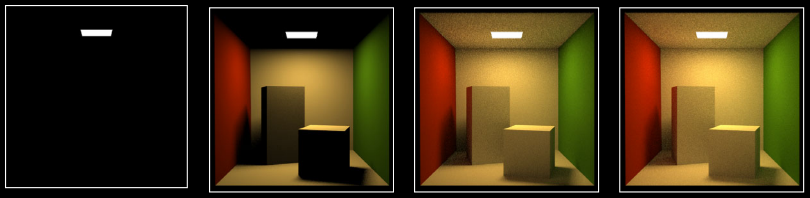
Indirect paths

- General algorithm:

```
compute_radiance (point, direction)
       est_rad = 0;
       for (i=0; i<n; i++)
              p = generate_indirect_path;
              est_rad += energy_transfer(p) / probability(p);
       est_rad = est_rad / n;
       return(est_rad);
```



Indirect paths - how to end recursion?

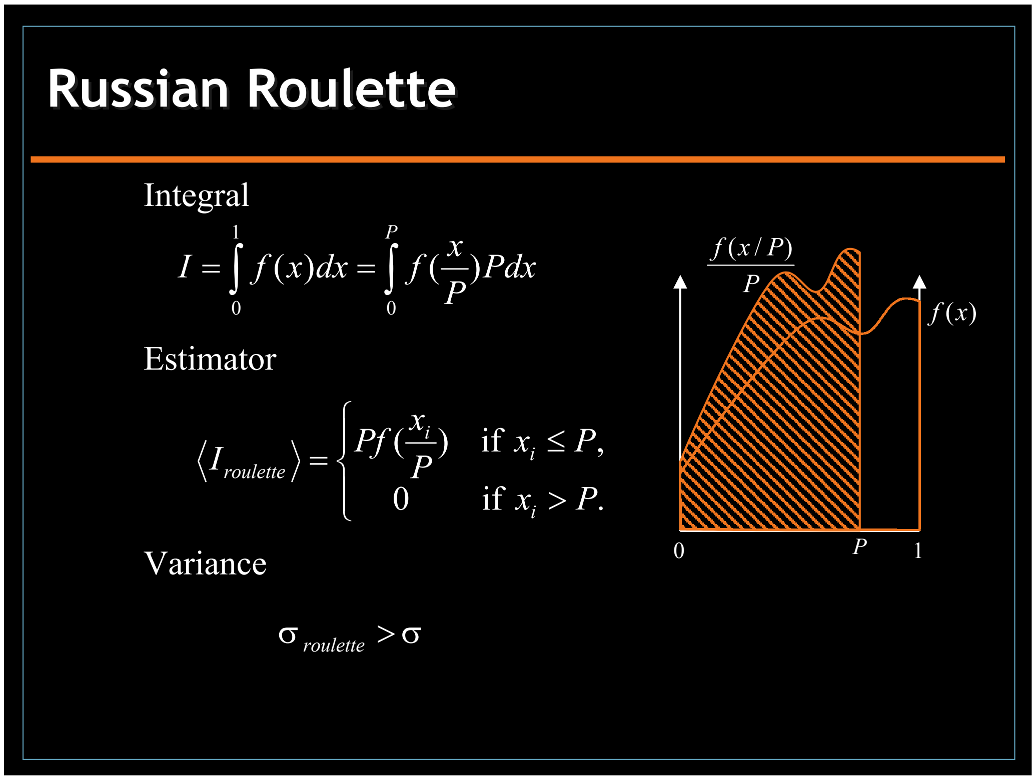


- Contributions of further light bounces become less significant
- If we just ignore them, estimators will be incorrect!

An important issue when writing a recursive path generator is how to stop the recursion.

Our goal is still to produce unbiased images, that is, images which will be correct if enough samples are being generated.

As such, we cannot ignore deeper recursions, although we would like to spend less time on them, since the light transport along these longer paths is will probably be less significant.



Russian Roulette

Integral

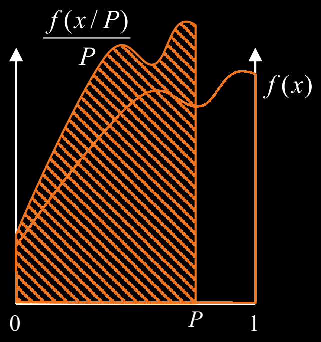
$$I = \int_0^1 f(x)dx = \int_0^P f(\frac{x}{P})Pdx$$

Estimator

$$\langle I_{roulette} \rangle = \begin{cases} Pf(\frac{x_i}{P}) & \text{if } x_i \le P, \\ 0 & \text{if } x_i > P. \end{cases}$$

Variance

$$\sigma_{roulette} > \sigma$$

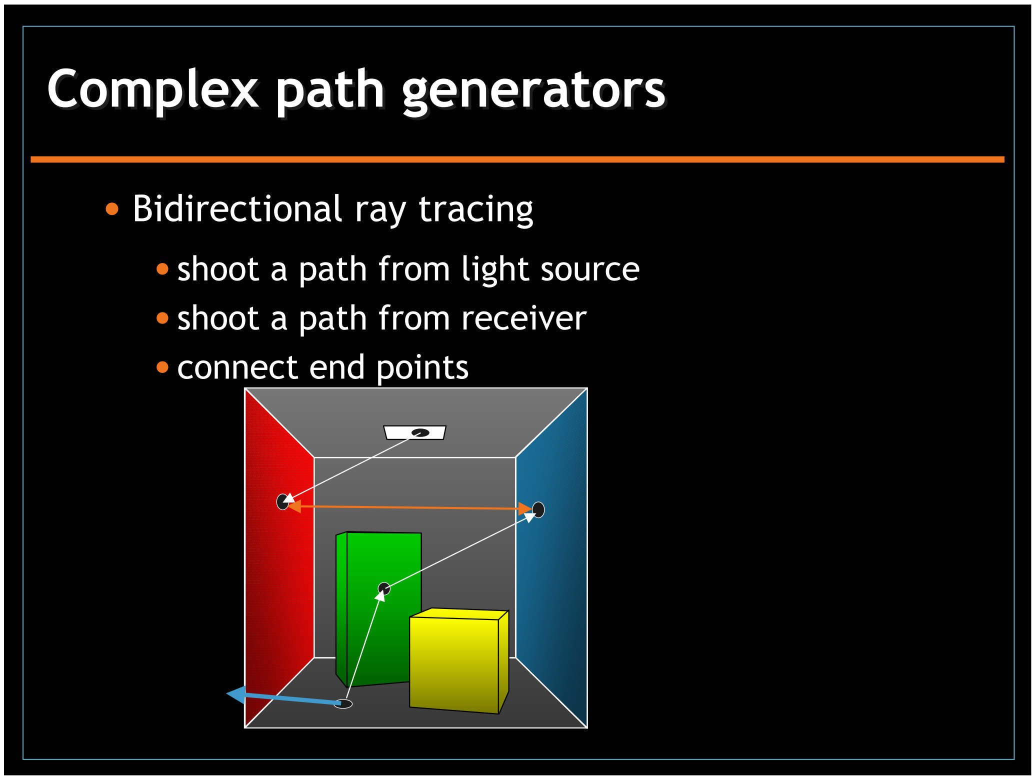


Russian Roulette is a technique that can be used to stop the recursion.

Mathematically, it means that we will cosnider part of integration domain to have a function value of 0. If a sample is generated in this part of the domain, it is 'absorbed'. Of course, this means that the samples which are not absorbed will need to get a greater weight, since they have to compensate for the fact that we still want an unbiased estimator for the original integral.

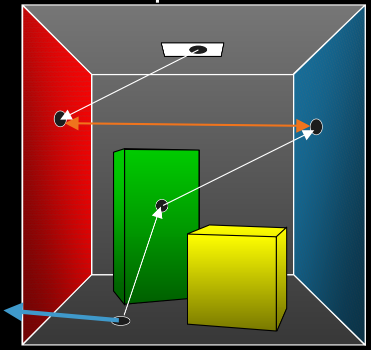
Russian Roulette

- In practice: pick some 'absorption probability' α
  - probability $1-\alpha$ that ray will bounce
  - estimated radiance becomes $L/(1-\alpha)$
- E.g. $\alpha = 0.9$
  - only 1 chance in 10 that ray is reflected
  - estimated radiance of that ray is multiplied by 10
- Intuition
  - instead of shooting 10 rays, we shoot only 1, but count the contribution of this one 10 times



Complex path generators

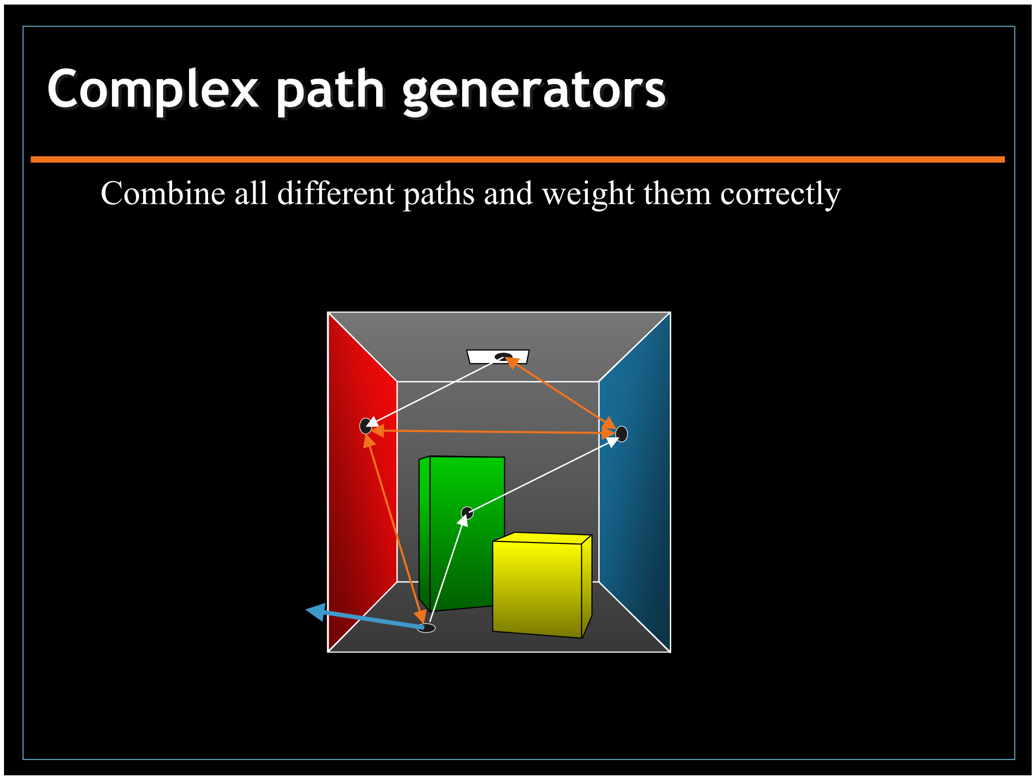
- Bidirectional ray tracing
  - shoot a path from light source
  - shoot a path from receiver
  - connect end points



More complex path generators are also possible.

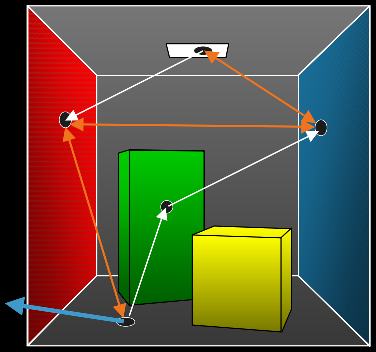
Bidirectional ray tracing is an algorithm that generates paths with variable length, both from the light source and the eye, and connects the end points.

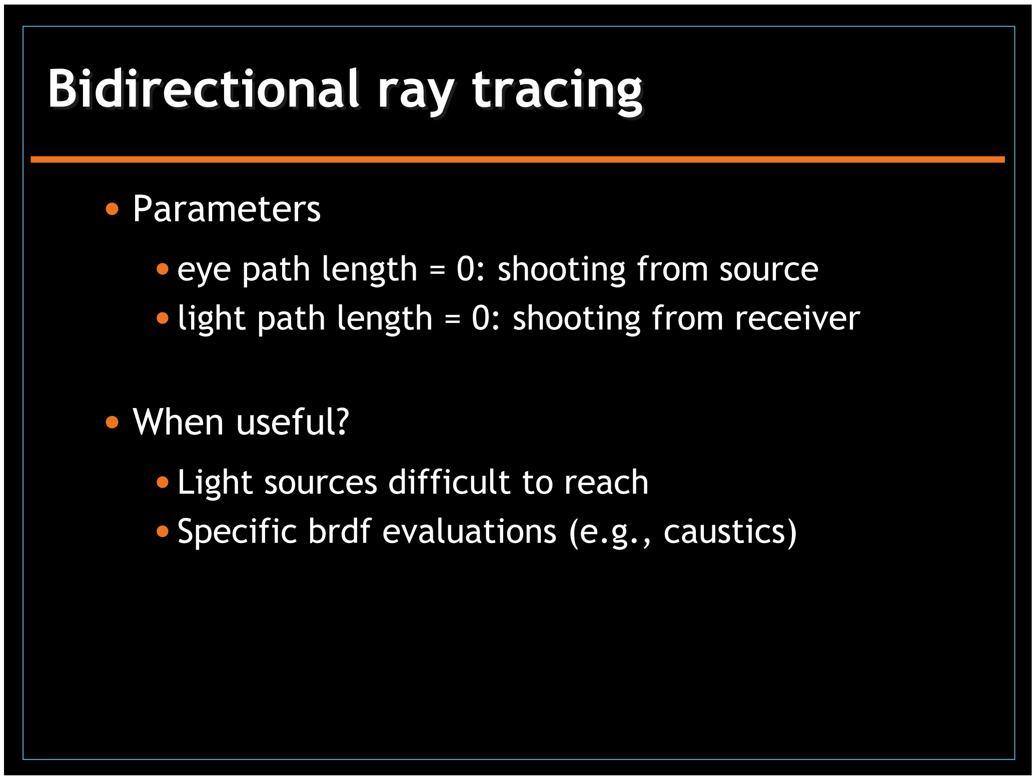
Again, this is path generator, and results in an unbiased images if all relevant pdf's are taken into account.



Complex path generators

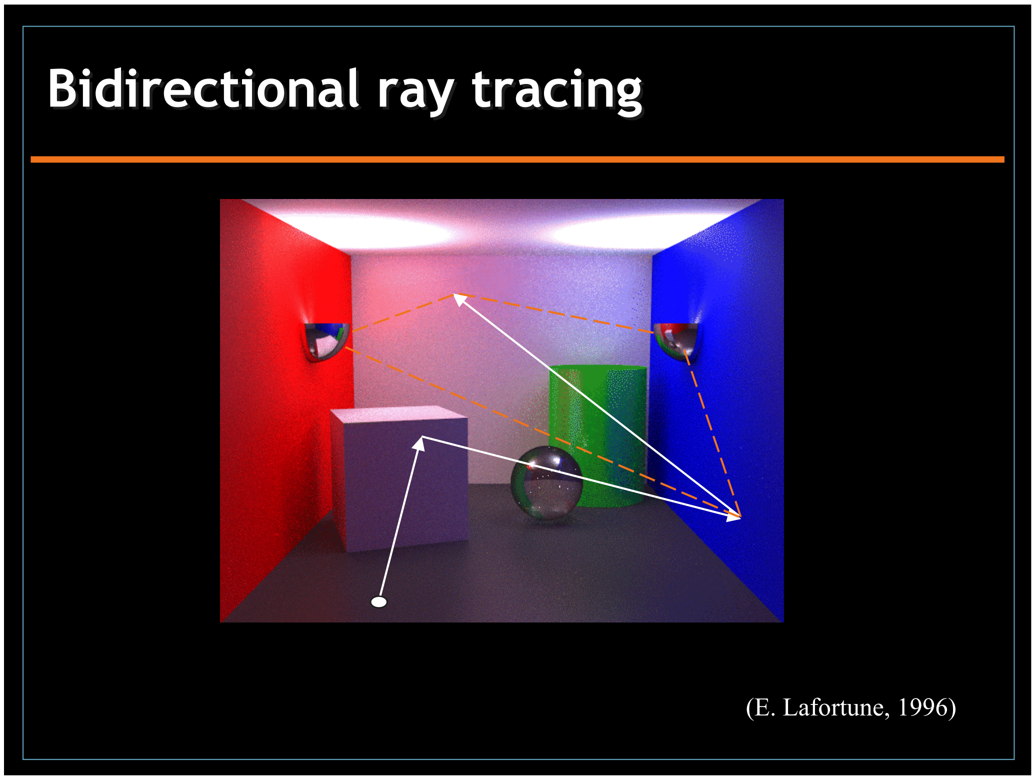
Combine all different paths and weight them correctly



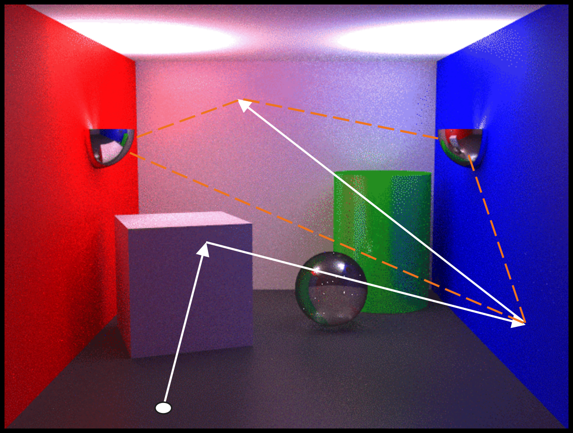


Bidirectional ray tracing

- Parameters
  - eye path length = 0: shooting from source
  - light path length = 0: shooting from receiver
- When useful?
  - Light sources difficult to reach
  - Specific brdf evaluations (e.g., caustics)

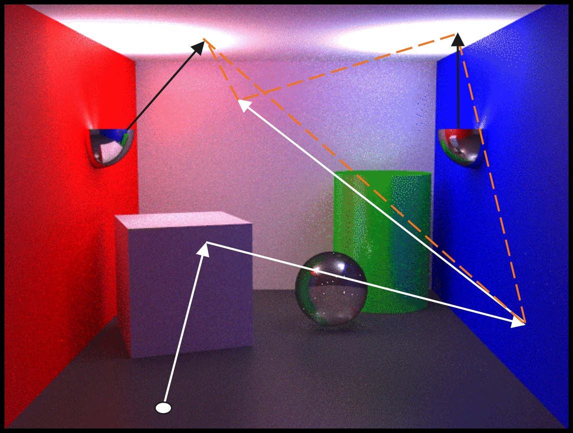


Bidirectional ray tracing

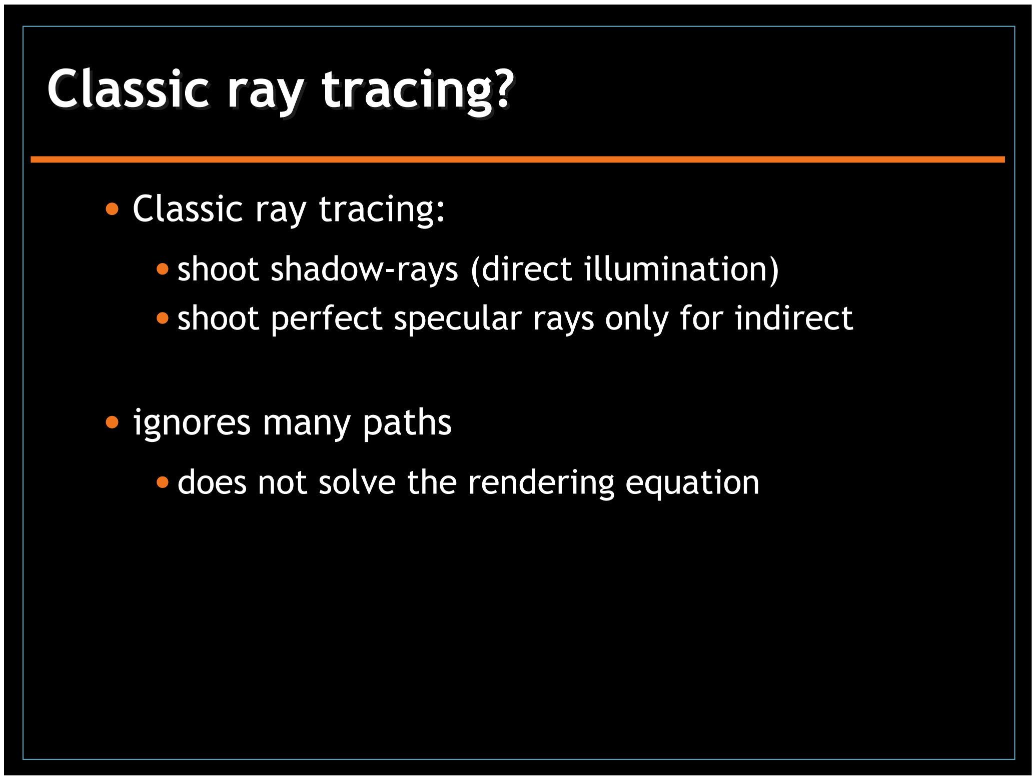


(E. Lafortune, 1996)

Bidirectional ray tracing



(E. Lafortune, 1996)

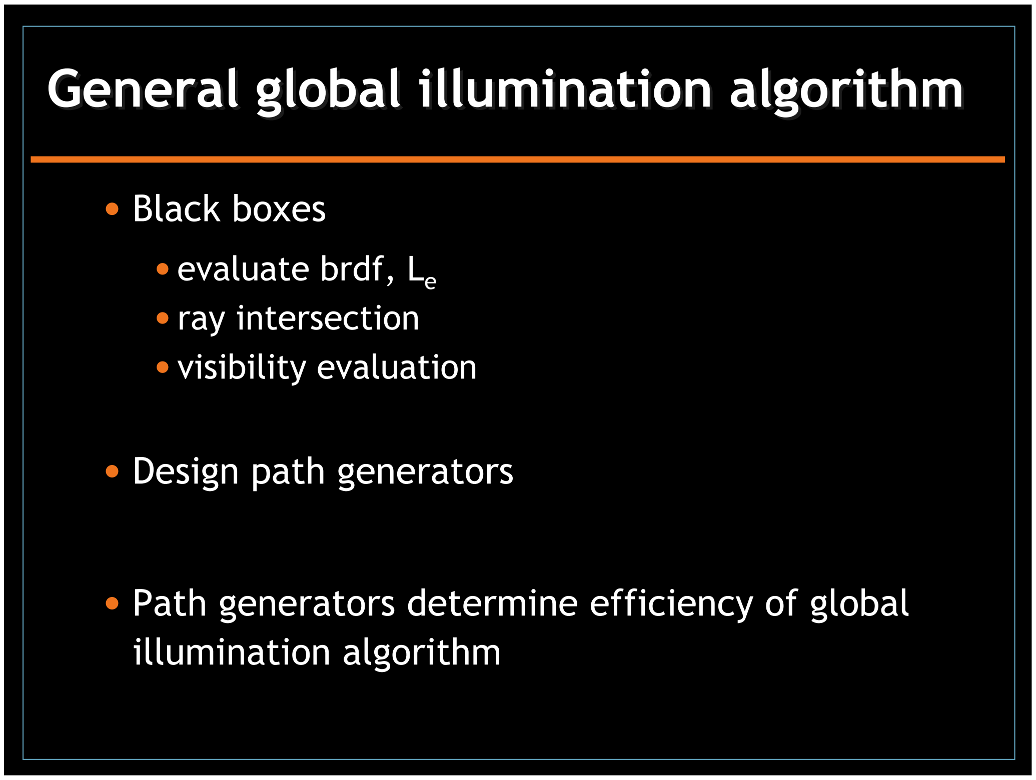


Classic ray tracing?

- Classic ray tracing:
  - shoot shadow-rays (direct illumination)
  - shoot perfect specular rays only for indirect
- ignores many paths
  - does not solve the rendering equation

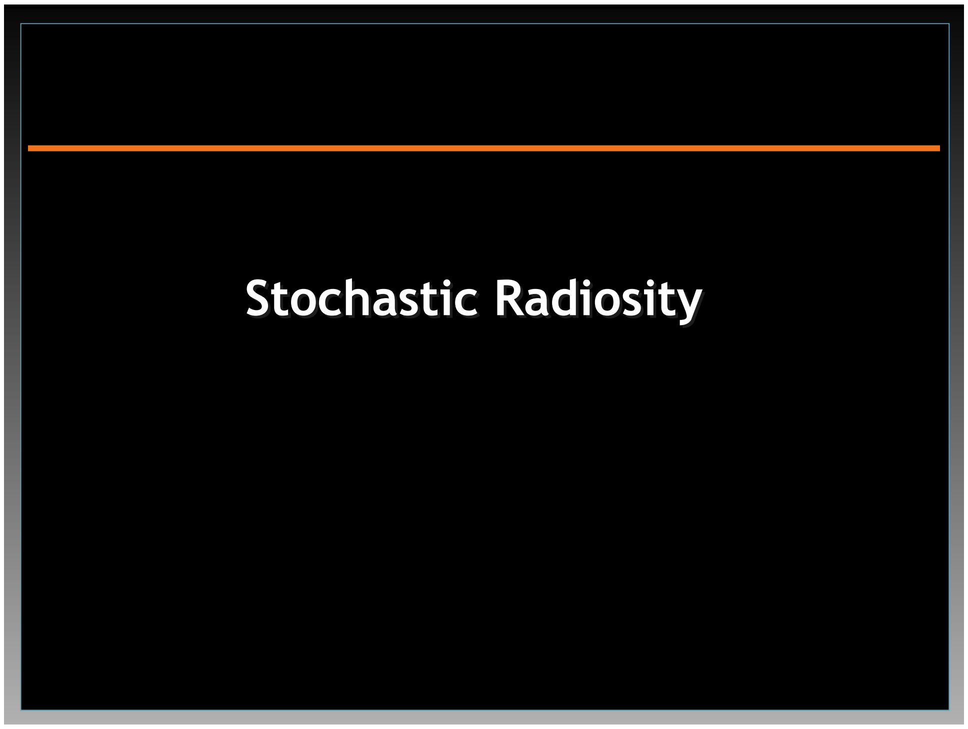
How does classic ray tracing compare to the physically correct path genertors described so far?

Classic ray tracing only generates a subset of all possible paths: shadow rays, and the perfect specular and refractive paths. As such, classic ray tracing ignores many of the other paths along which energy is transported from the light sources to the receiving surfaces.

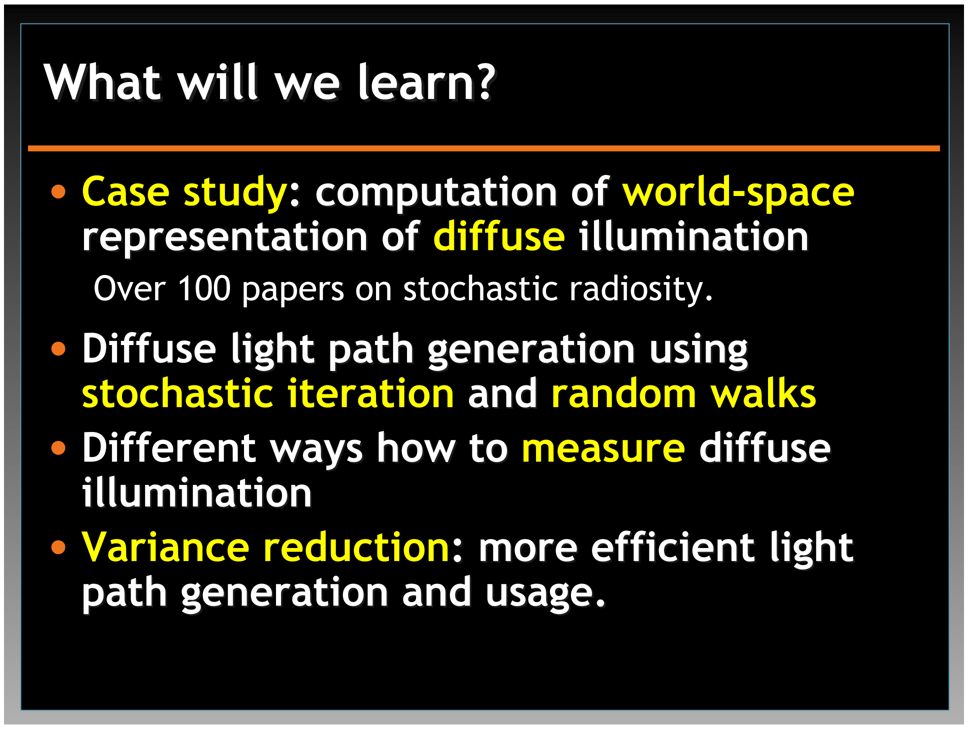


General global illumination algorithm

- Black boxes
  - evaluate brdf, L_e
  - ray intersection
  - visibility evaluation
- Design path generators
- Path generators determine efficiency of global illumination algorithm



Stochastic Radiosity

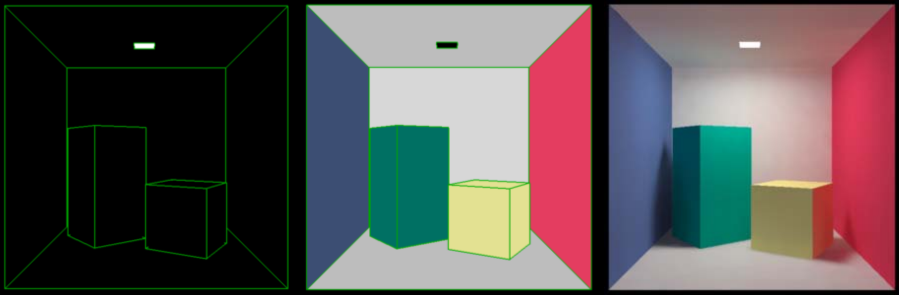


What will we learn?

- Case study: computation of world-space representation of diffuse illumination

  Over 100 papers on stochastic radiosity.
- Diffuse light path generation using stochastic iteration and random walks
- Different ways how to measure diffuse illumination
- Variance reduction: more efficient light path generation and usage.

Mathematical problem description (1): Rendering Equation (general)



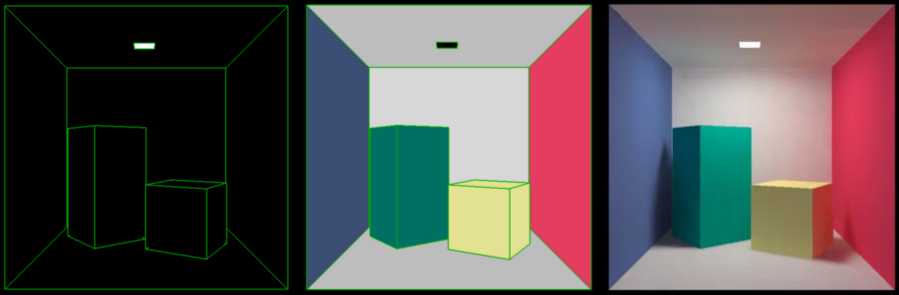
Self-emitted radiance

brdf

total radiance

$$L(x\to\Psi) = L^e(x\to\Psi) + \int_{\Omega_x} f_r(x;\Psi\leftrightarrow\Theta_x)L(x\leftarrow\Theta_x)\cos\theta_x d\omega_{\Theta_x}$$

Mathematical problem description (2): Radiosity Integral Equation (diffuse)



Self-emitted radiosity

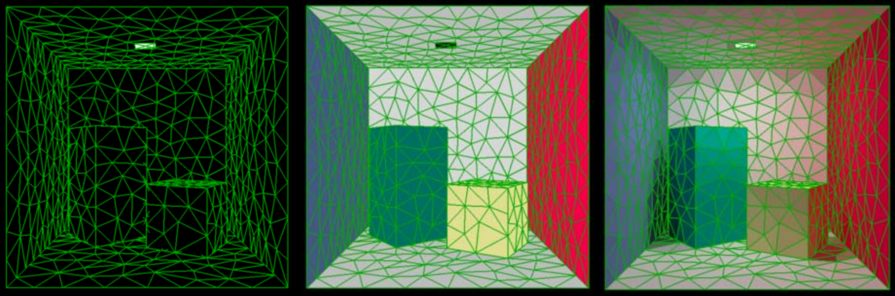
reflectivity

total radiosity

$$B(x) = E(x) + \int_S \rho(x)G(x,y)B(y)dA_y$$

$$G(x,y) = \frac{\cos\theta_x\cos\theta_y}{\pi r_{xy}^2}\mathrm{vis}(x,y)$$

Mathematical problem description (3): Radiosity Linear System



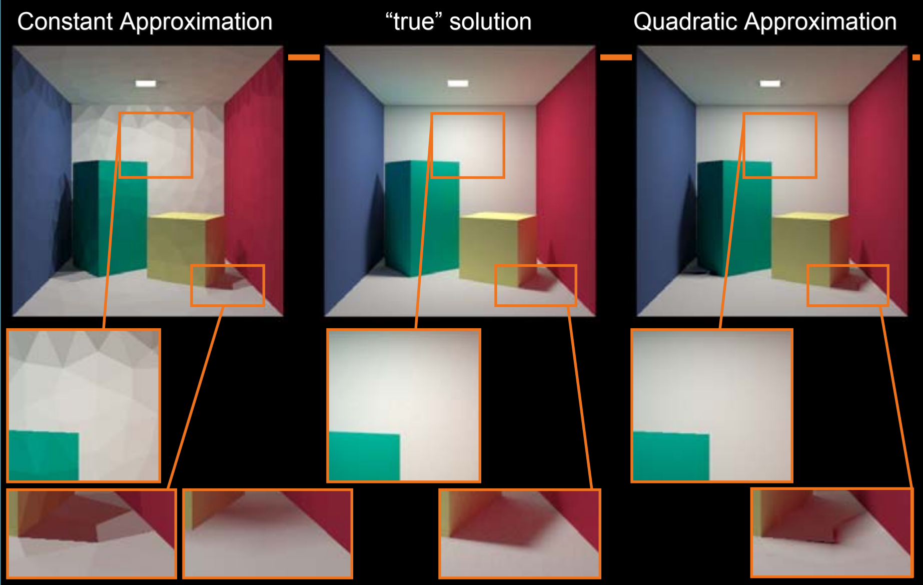
$$B_i = E_i + \rho_i \sum_j F_{ij} B_j$$

Classical Radiosity

1. Discretise the input scene
   Problem: discretisation artifacts
2. Compute form factors
   Problem: huge number of non-trivial
   integrals: 95% of the computation time,
   very large storage requirements,
   computational error.
3. Solve radiosity system
4. Tone mapping and display

In practice intertwined!

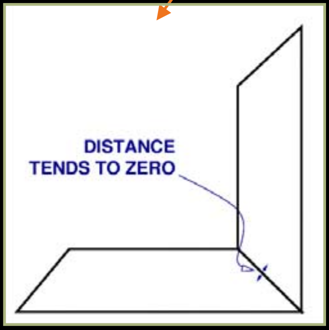
Discretisation Artifacts

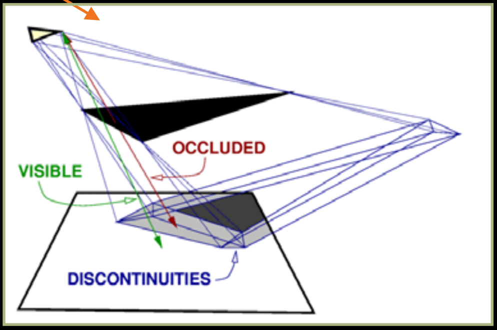


Form Factor Singularities and Discontinuities

$$F_{ij} = \frac{1}{A_i} \int_{S_i} \int_{S_j} G(x,y)\, dA_y dA_x$$

$$G(x,y) = \frac{\cos\theta_x \cos\theta_y}{\pi r_{xy}^2} \mathrm{vis}(x,y).$$





Monte Carlo Methods

Principle:

- Formulate solution of a problem as the expectation of a random variable
- Sample random variable
- Mean of samples yields estimate for solution

Example: simple estimation of a sum

Summation by Monte Carlo

Sum

$$S = \sum_{i=1}^{n} a_i$$

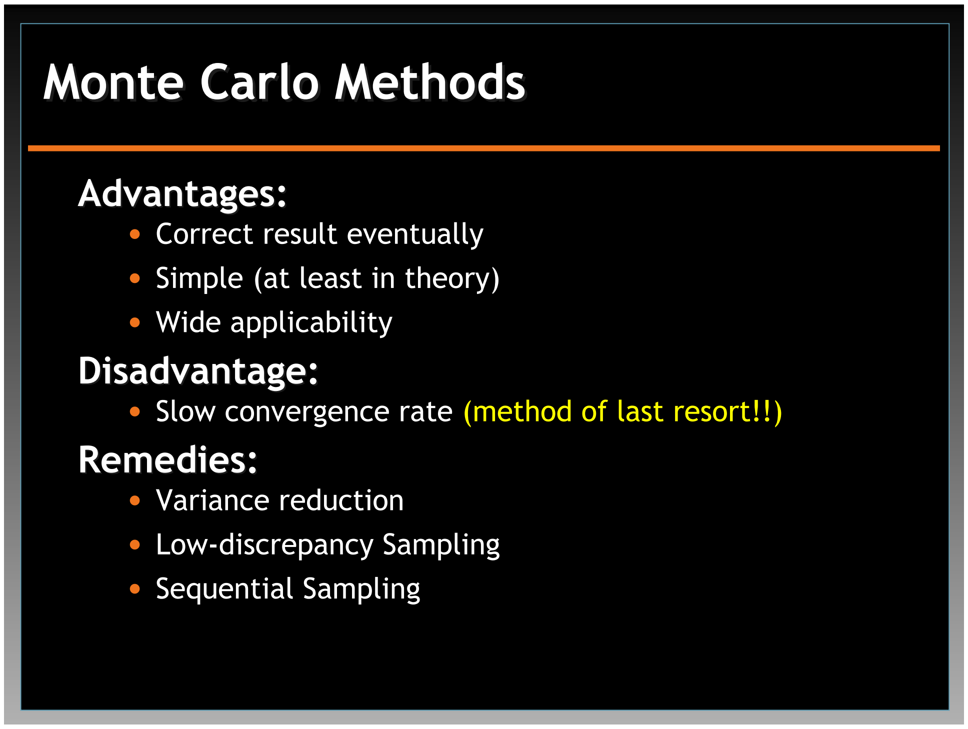
Estimate

$$\tilde{S} = \frac{1}{N} \sum_{k=1}^{N} \frac{a_{i_k}}{p_{i_k}} \approx S$$

Variance

$$V[\hat{S}_N] = \frac{1}{N} \left(\sum_{i=1}^{n} \frac{a_i^2}{p_i} - S^2 \right)$$

Only useful for sums with a large number of complicated terms.



Monte Carlo Methods

Advantages:

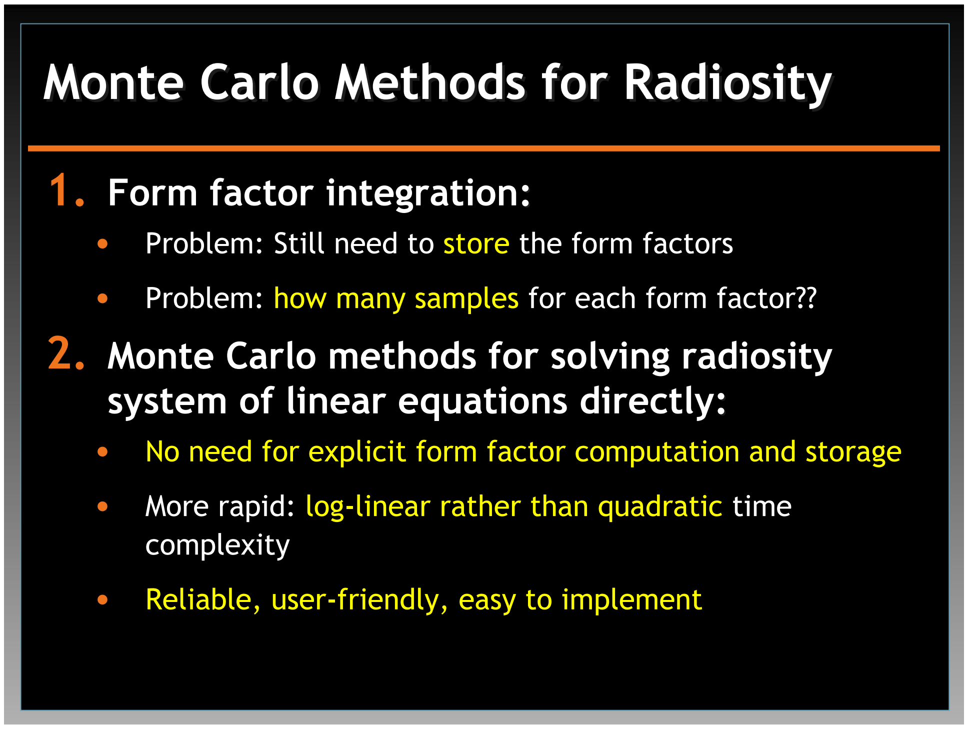
- Correct result eventually
- Simple (at least in theory)
- Wide applicability

Disadvantage:

- Slow convergence rate (method of last resort!!)

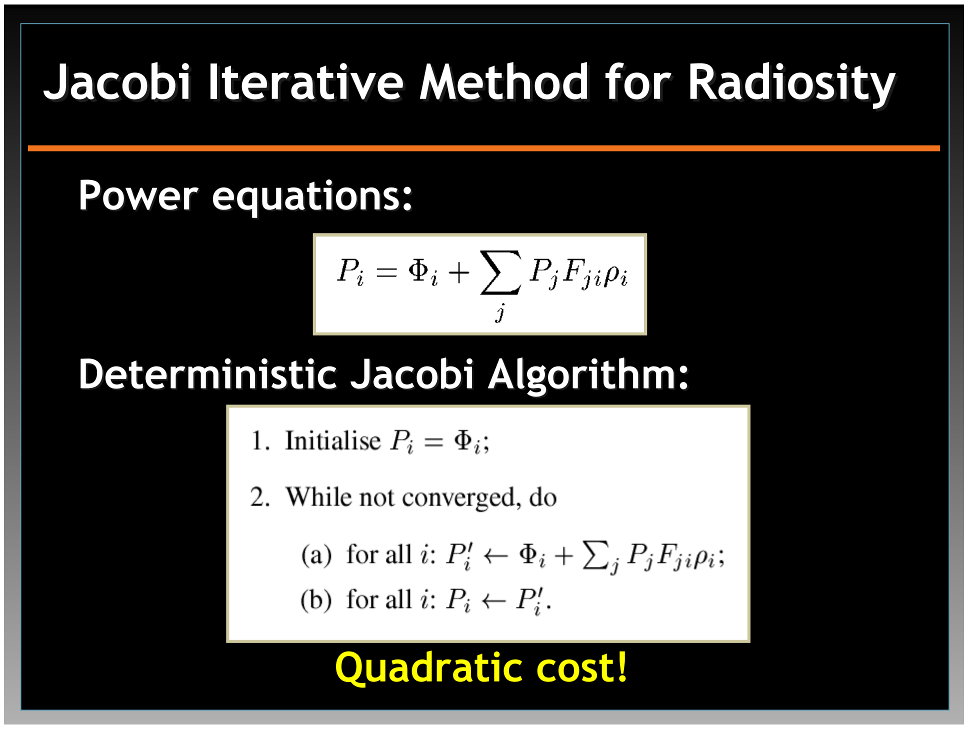
Remedies:

- Variance reduction
- Low-discrepancy Sampling
- Sequential Sampling



Monte Carlo Methods for Radiosity

1. **Form factor integration:**
   - Problem: Still need to store the form factors
   - Problem: how many samples for each form factor??
2. **Monte Carlo methods for solving radiosity system of linear equations directly:**
   - No need for explicit form factor computation and storage
   - More rapid: log-linear rather than quadratic time complexity
   - Reliable, user-friendly, easy to implement



Jacobi Iterative Method for Radiosity

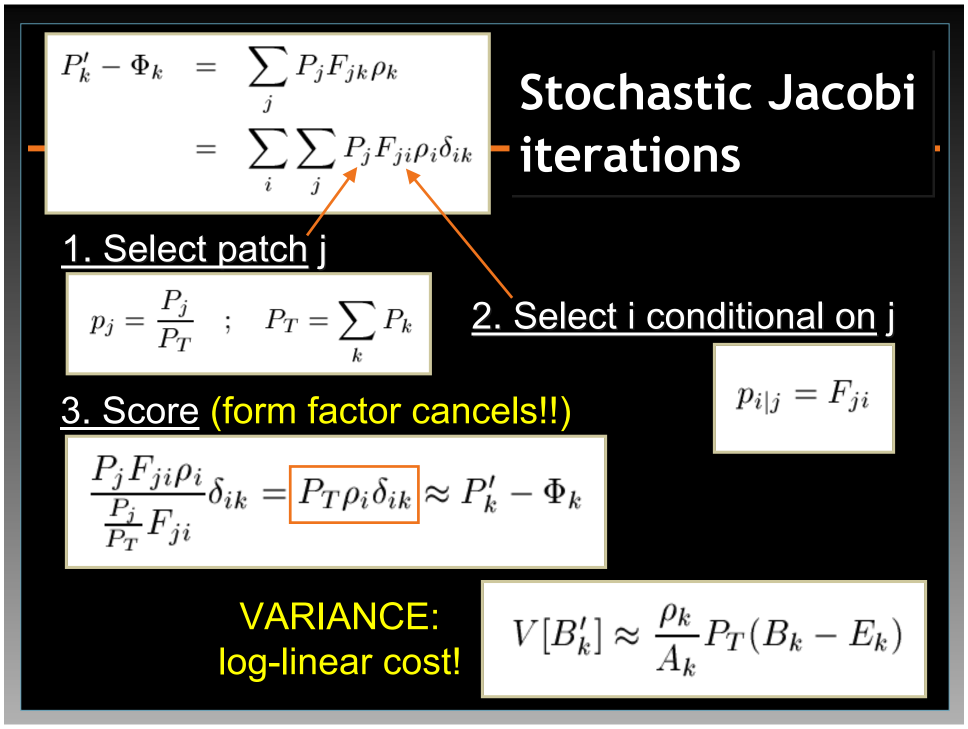
Power equations:

$$P_i = \Phi_i + \sum_j P_j F_{ji} \rho_i$$

Deterministic Jacobi Algorithm:

```
1. Initialise $P_i = \Phi_i$;
2. While not converged, do
    (a) for all $i$: $P'_i \leftarrow \Phi_i + \sum_j P_j F_{ji} \rho_i$;
    (b) for all $i$: $P_i \leftarrow P'_i$.
```

Quadratic cost!



Stochastic Jacobi iterations

$$\begin{aligned} P'_k - \Phi_k &= \sum_j P_j F_{jk} \rho_k \\ &= \sum_i \sum_j P_j F_{ji} \rho_i \delta_{ik} \end{aligned}$$

1. Select patch j

$$p_j = \frac{P_j}{P_T} \quad ; \quad P_T = \sum_k P_k$$

2. Select i conditional on j

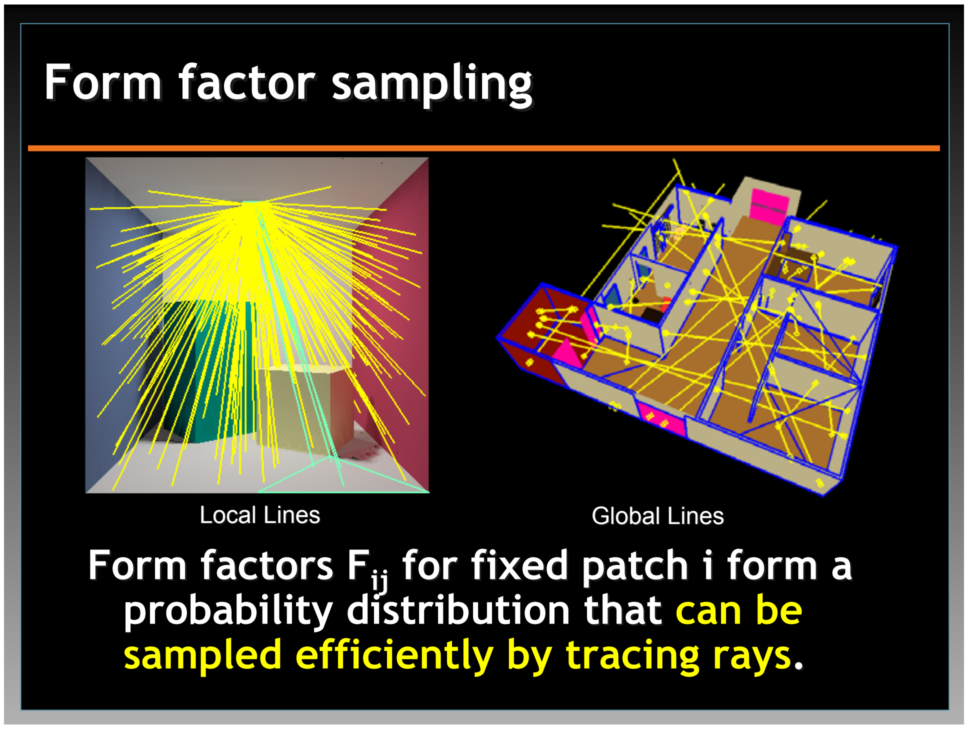
$$p_{i|j} = F_{ji}$$

3. Score (form factor cancels!!)

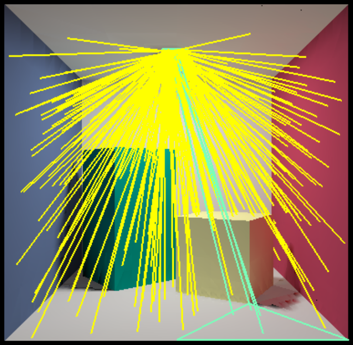
$$\frac{P_j F_{ji} \rho_i}{\frac{P_j}{P_T} F_{ji}} \delta_{ik} = \boxed{P_T \rho_i \delta_{ik}} \approx P'_k - \Phi_k$$

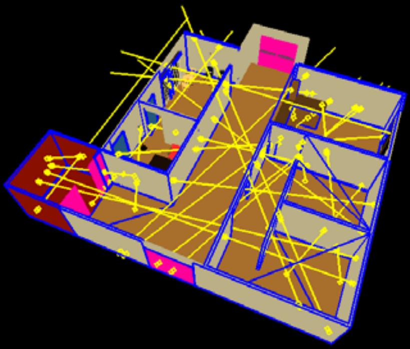
VARIANCE:
log-linear cost!

$$V[B'_k] \approx \frac{\rho_k}{A_k} P_T (B_k - E_k)$$

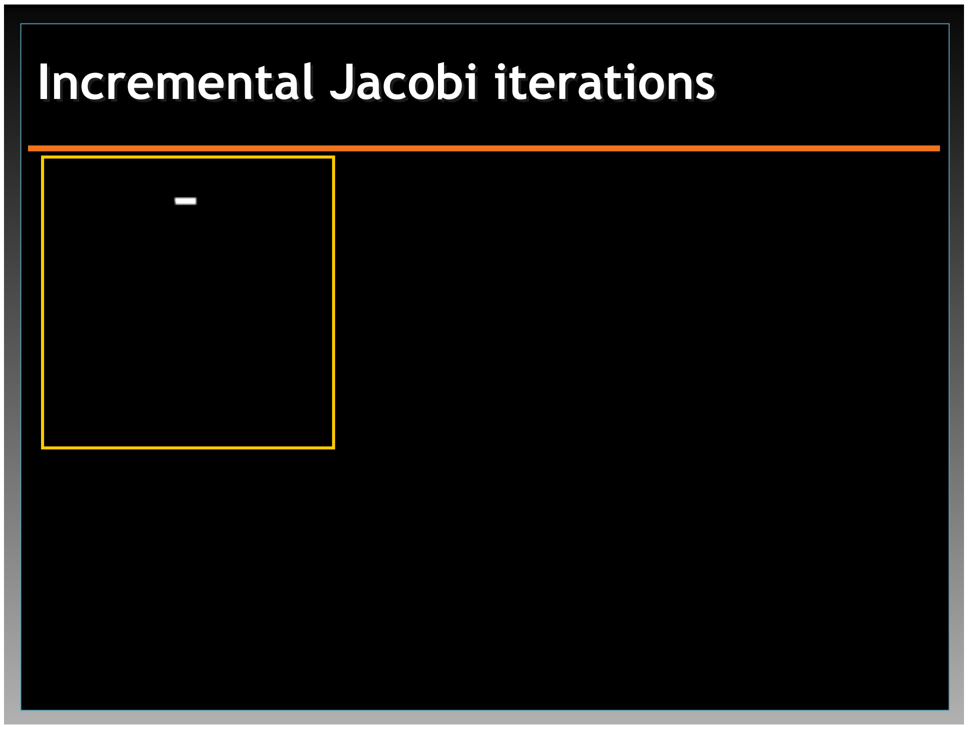


Form factor sampling


Local Lines

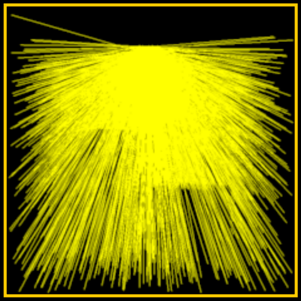

Global Lines

Form factors F_{ij} for fixed patch i form a probability distribution that can be sampled efficiently by tracing rays.

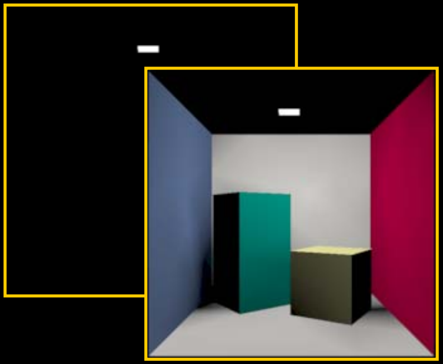


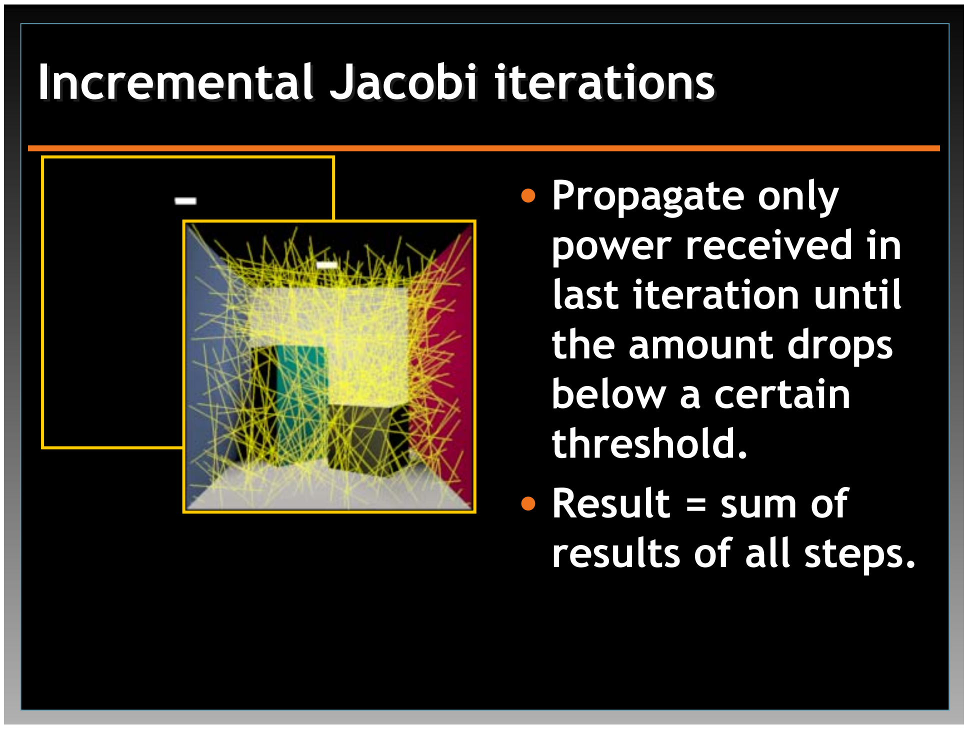
Incremental Jacobi iterations

Incremental Jacobi iterations

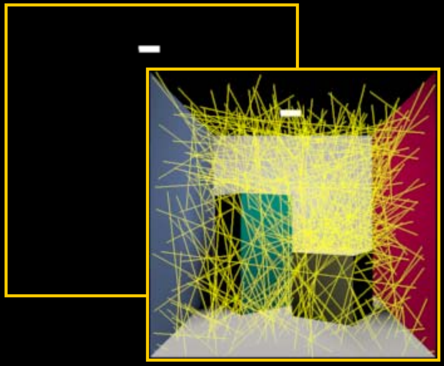


Incremental Jacobi iterations

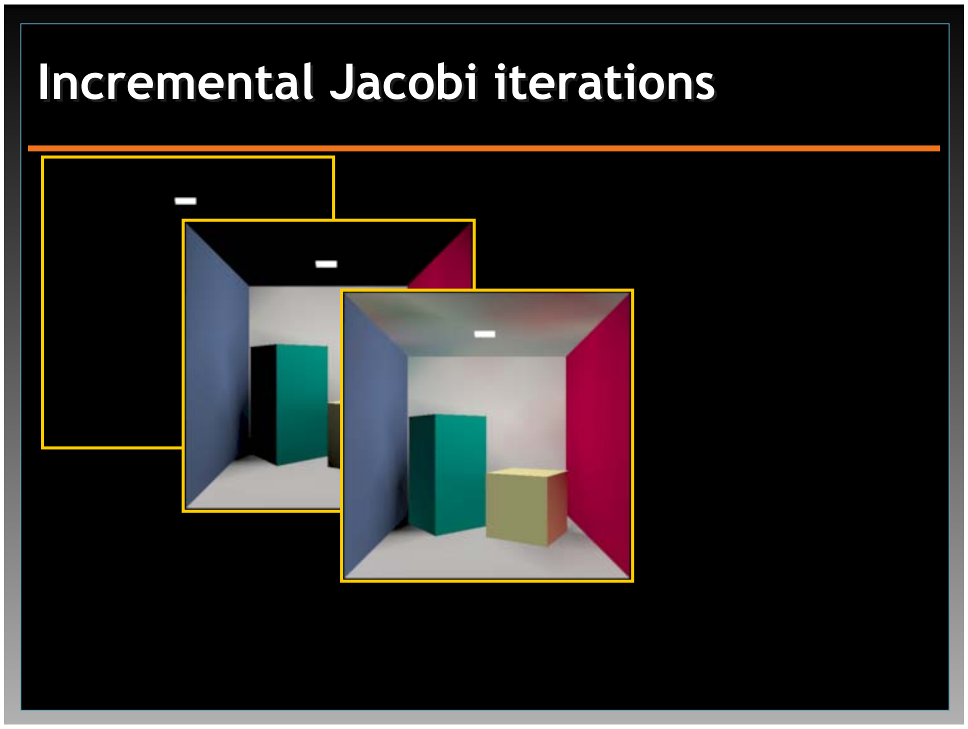




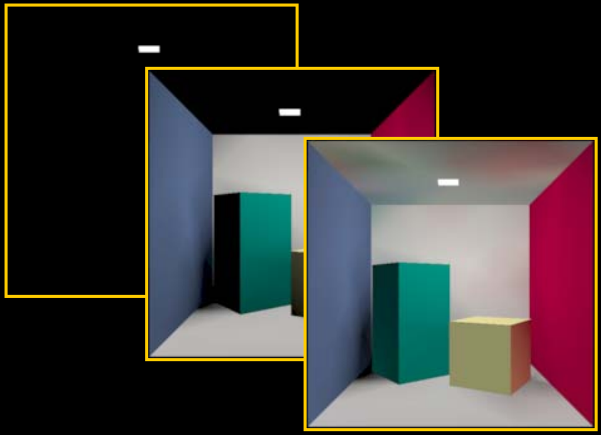
Incremental Jacobi iterations

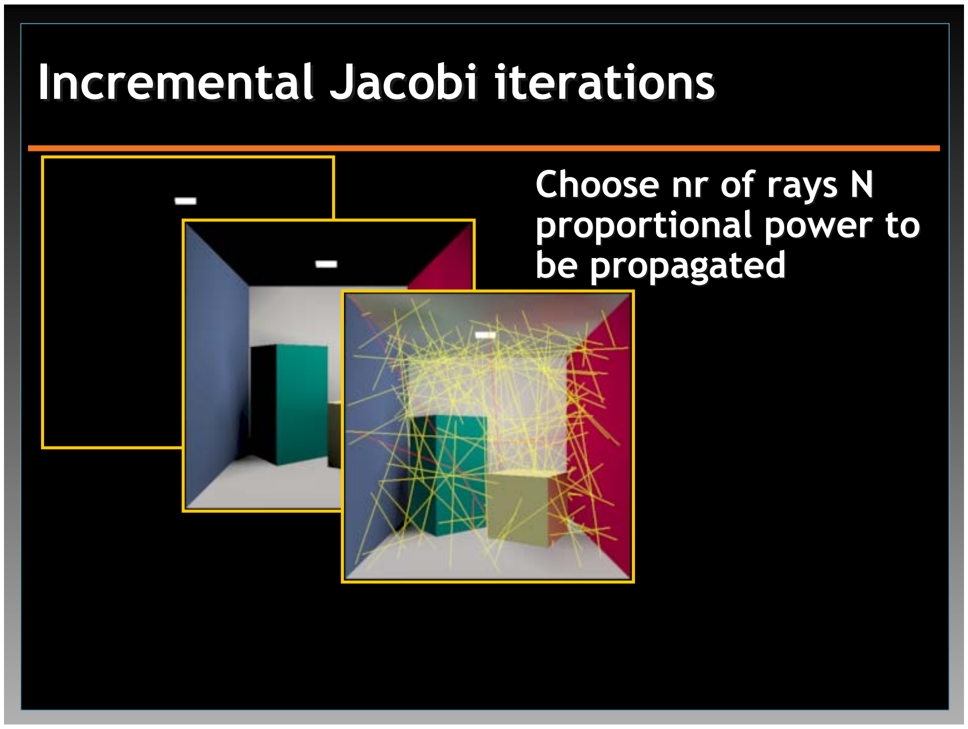


- Propagate only power received in last iteration until the amount drops below a certain threshold.
- Result = sum of results of all steps.



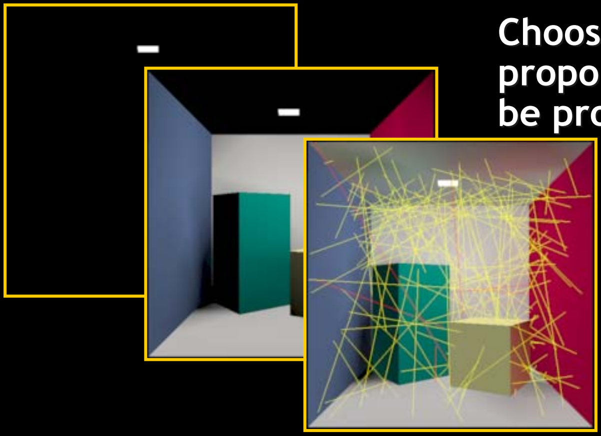
Incremental Jacobi iterations

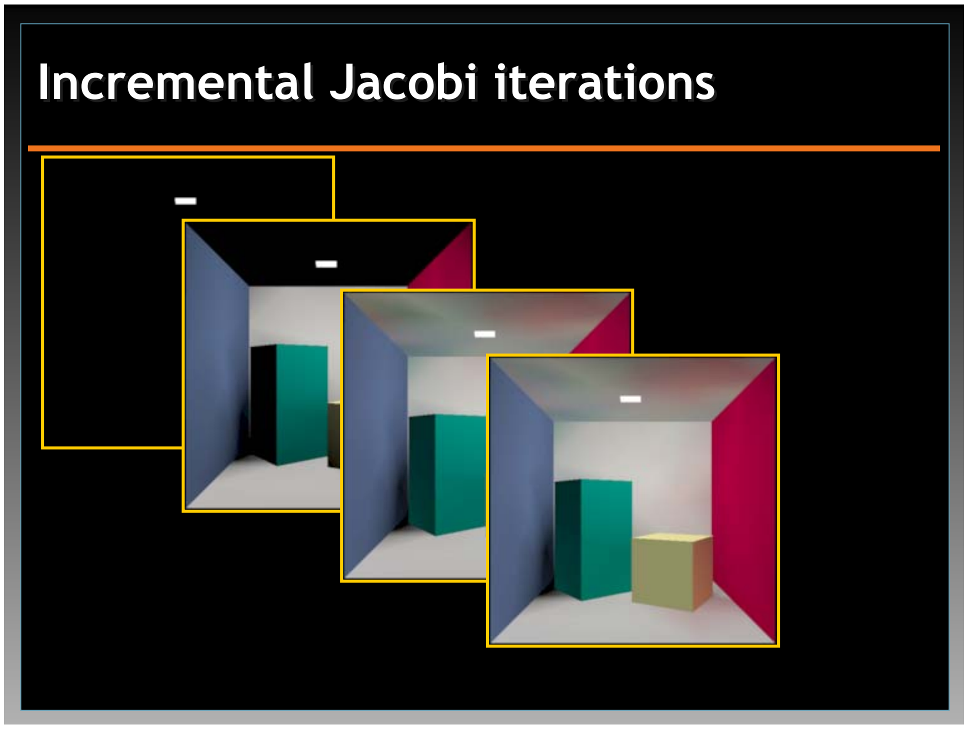




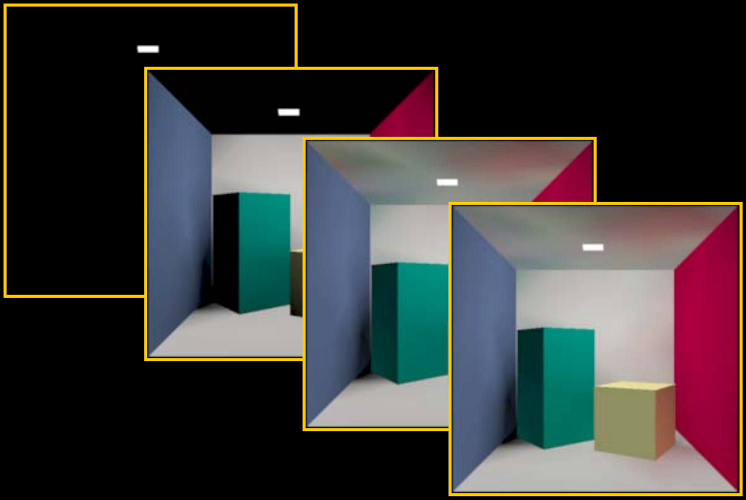
Incremental Jacobi iterations

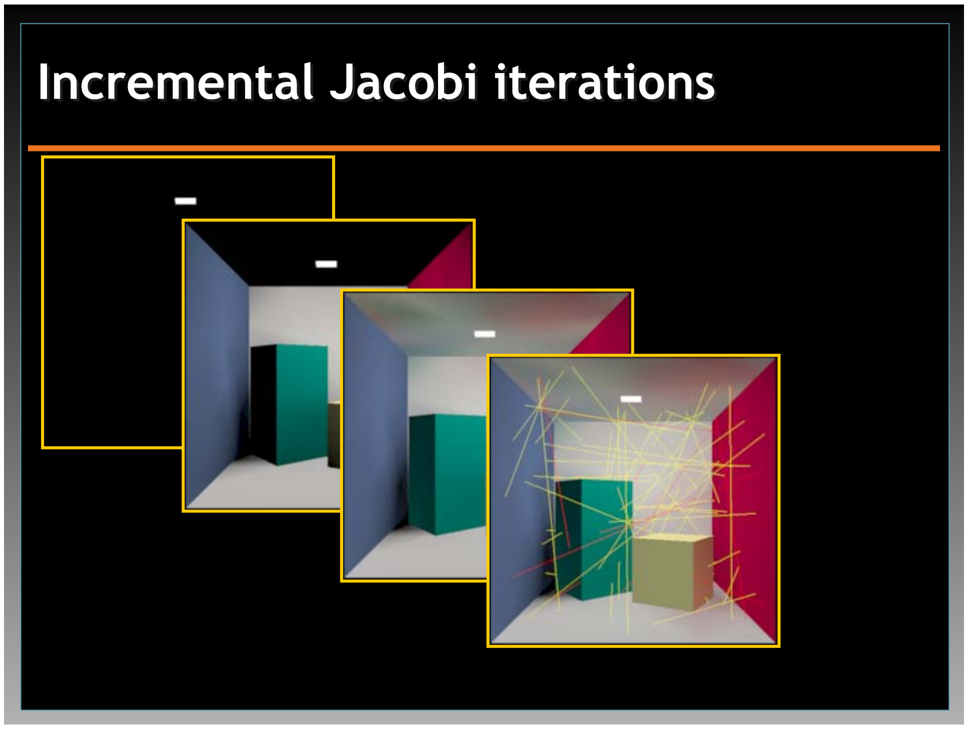
Choose nr of rays N proportional power to be propagated



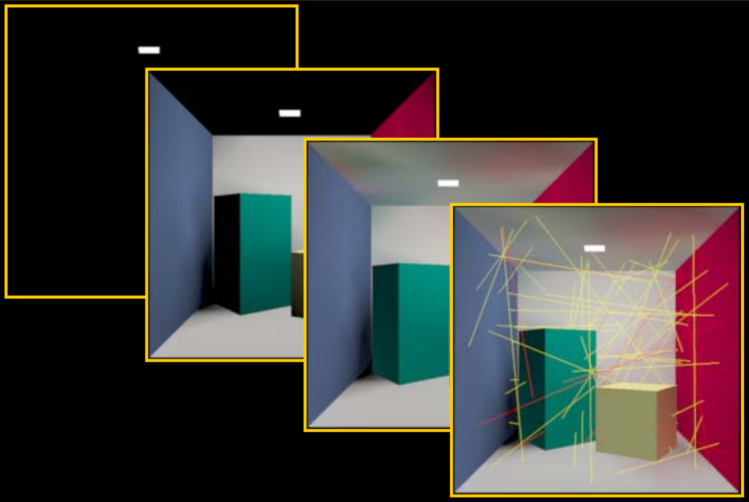


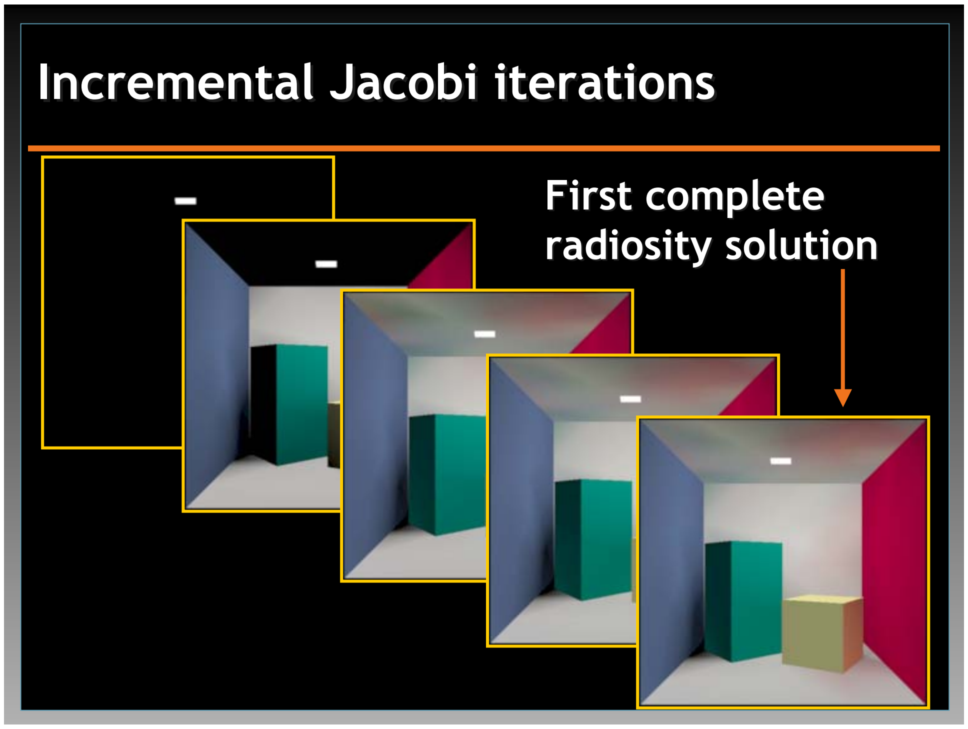
Incremental Jacobi iterations



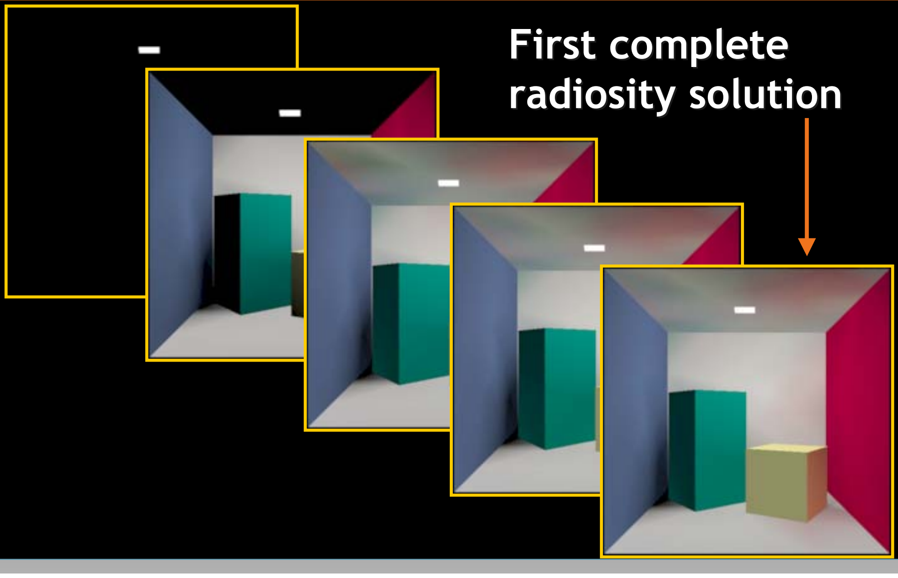


Incremental Jacobi iterations





Incremental Jacobi iterations



First complete radiosity solution

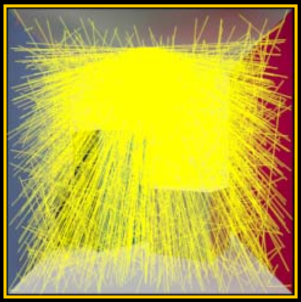
Regular Jacobi iterations

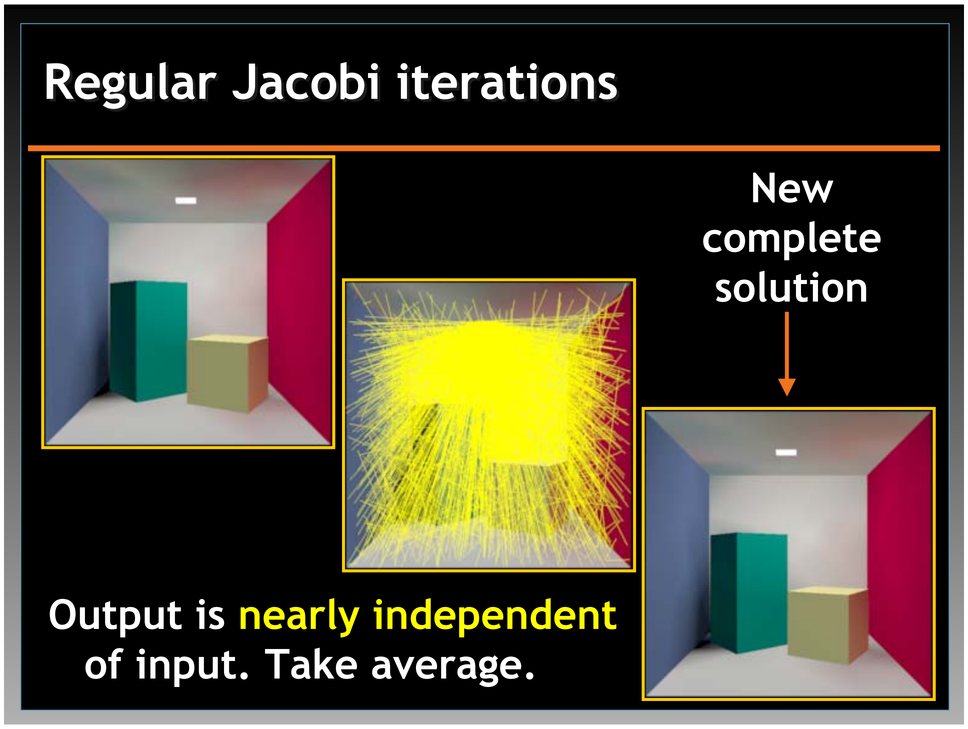
Complete solution

Regular Jacobi iterations

Complete solution

Propagate all power from all patches

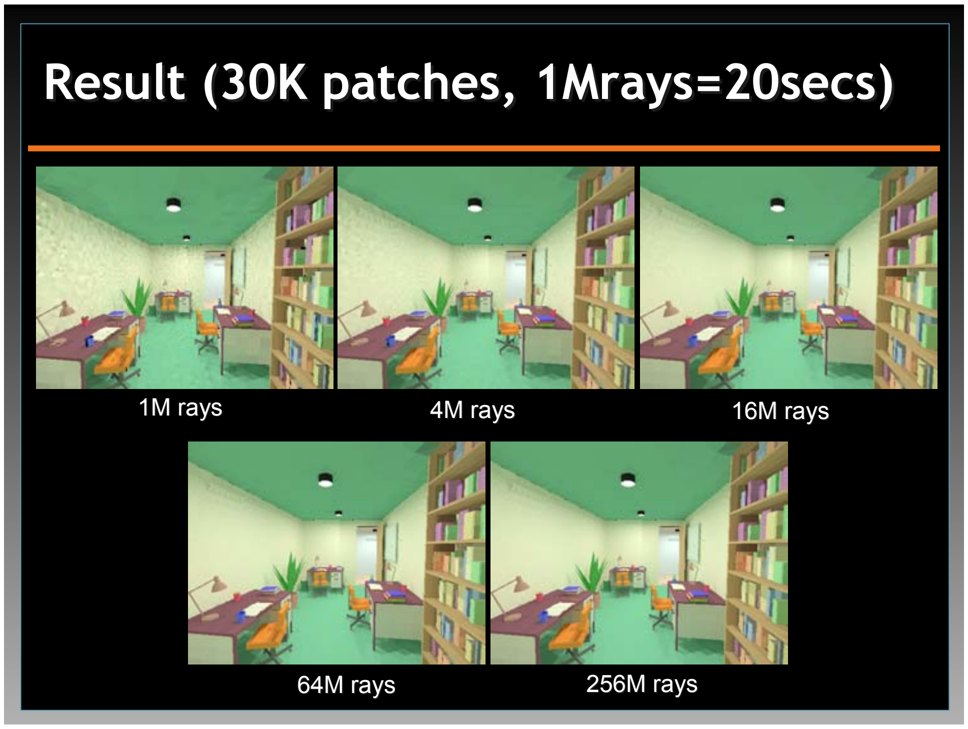




Regular Jacobi iterations

New complete solution

Output is nearly independent of input. Take average.



Result (30K patches, 1Mrays=20secs)

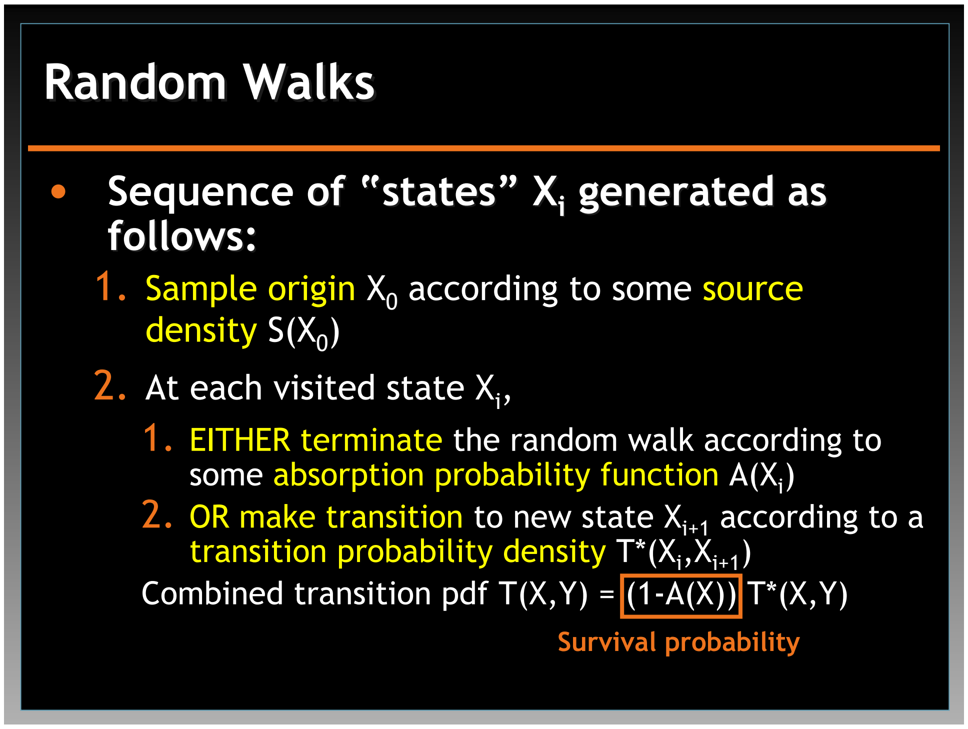
1M rays

4M rays

16M rays

64M rays

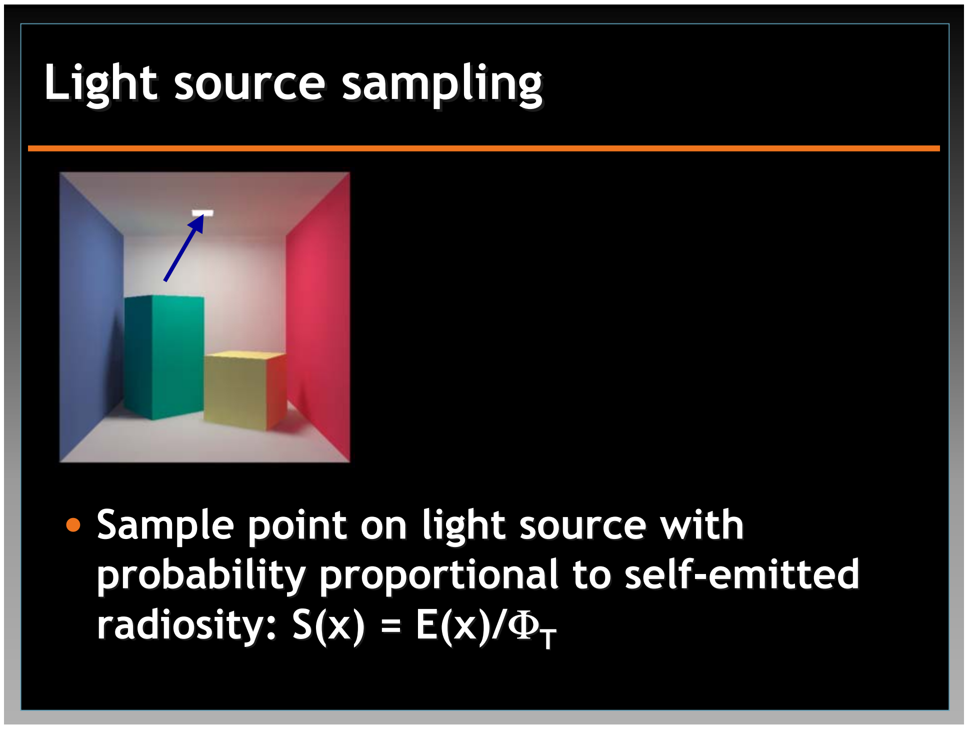
256M rays



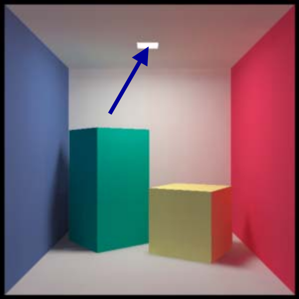
Random Walks

- Sequence of “states” X_i generated as follows:
  1. Sample origin X_0 according to some source density $S(X_0)$
  2. At each visited state X_i,
     1. EITHER terminate the random walk according to some absorption probability function $A(X_i)$
     2. OR make transition to new state X_{i+1} according to a transition probability density $T^*(X_i,X_{i+1})$

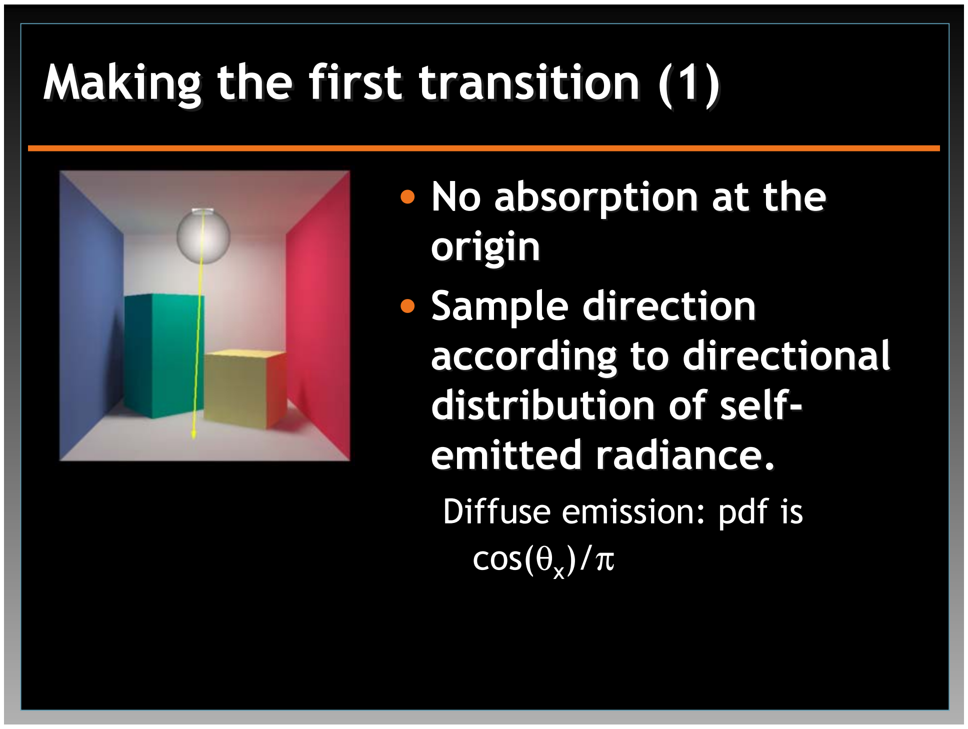
     Combined transition pdf $T(X,Y) = \underbrace{\boxed{(1-A(X))}}_{\text{Survival probability}} T^*(X,Y)$



Light source sampling



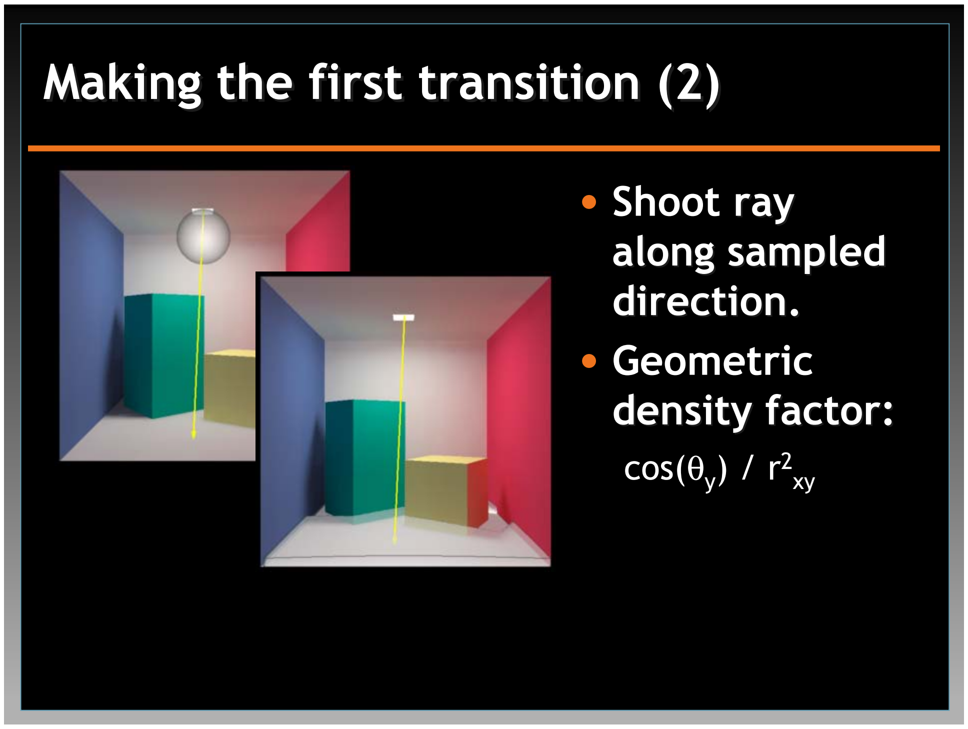
- Sample point on light source with probability proportional to self-emitted radiosity: $S(x) = E(x)/\Phi_T$



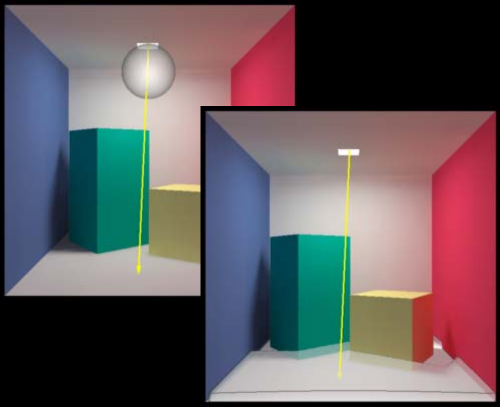
Making the first transition (1)

- No absorption at the origin
- Sample direction according to directional distribution of self-emitted radiance.

  Diffuse emission: pdf is $\cos(\theta_x)/\pi$



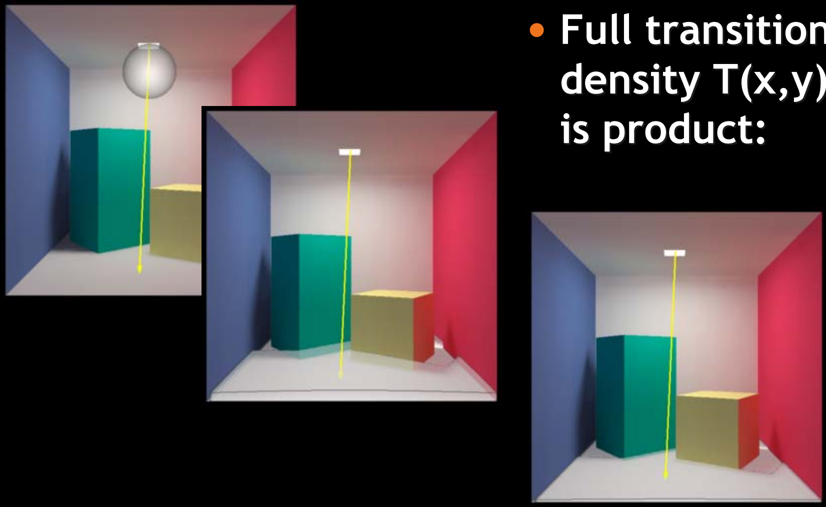
Making the first transition (2)



- Shoot ray along sampled direction.
- Geometric density factor:

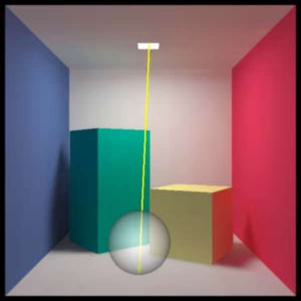
  $\cos(\theta_y) / r^2_{xy}$

Making the first transition (3)



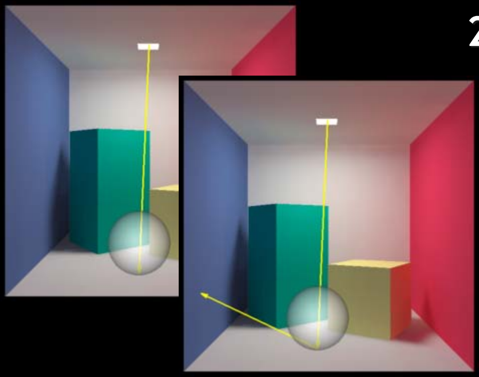
- Full transition density T(x,y) is product:

Further transitions



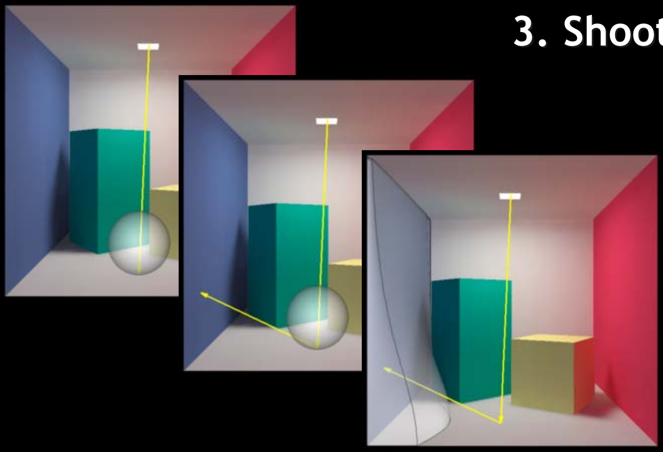
1. Absorption / survival test according to albedo

Further transitions (2)



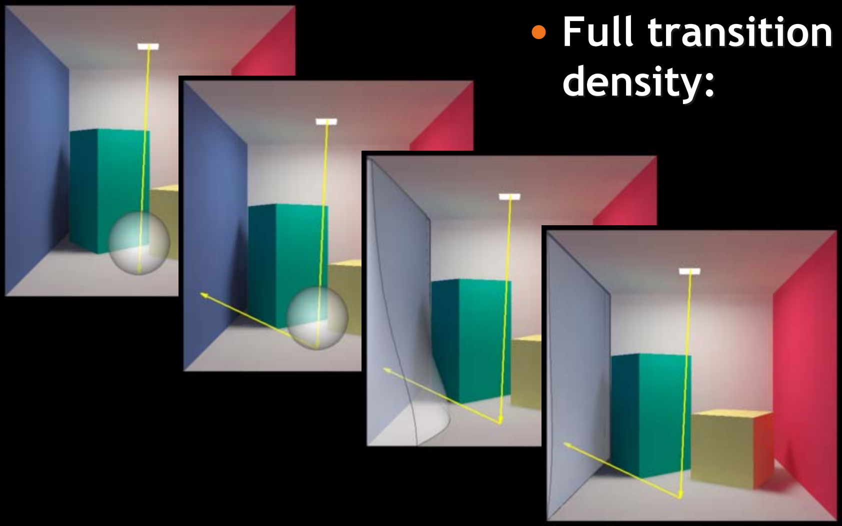
2. Sample direction according to brdf

Further transitions (3)



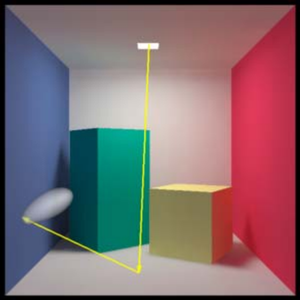
3. Shoot ray

Further transitions (4)

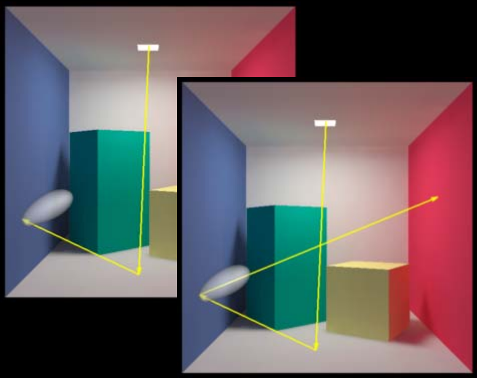


- Full transition density:

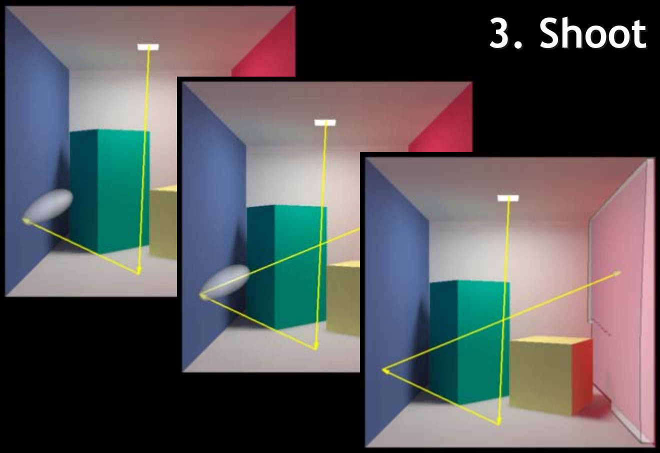
Once more …



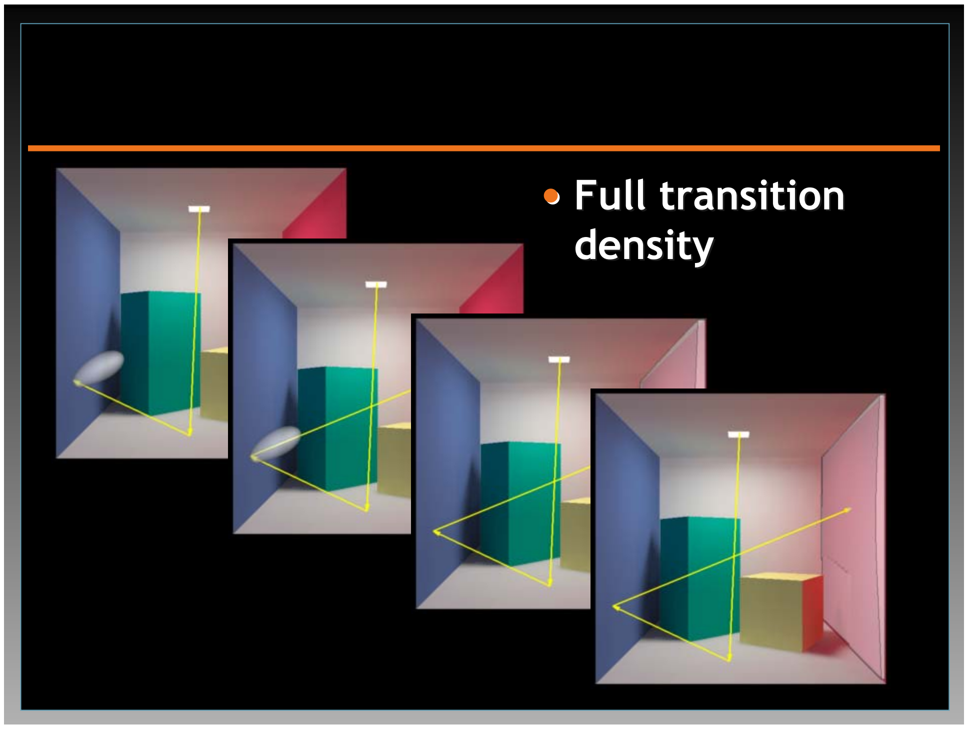
1. Absorption / survival test

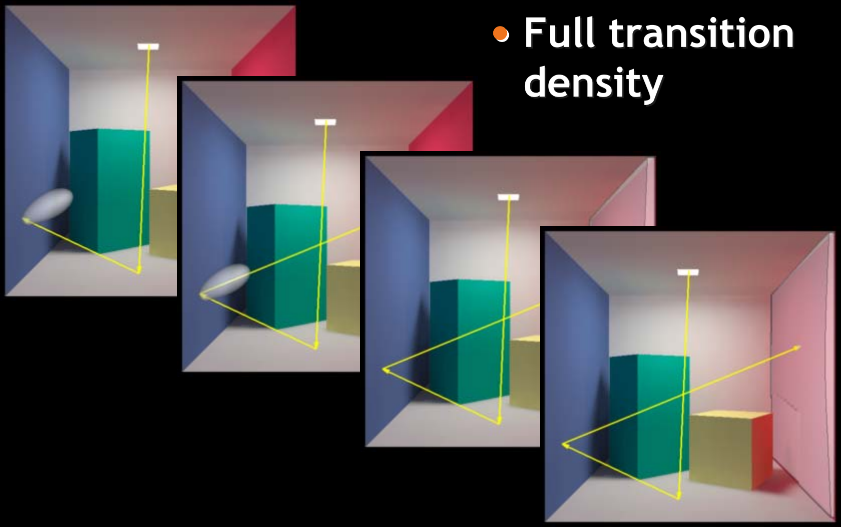


2. Sample direction according to brdf.

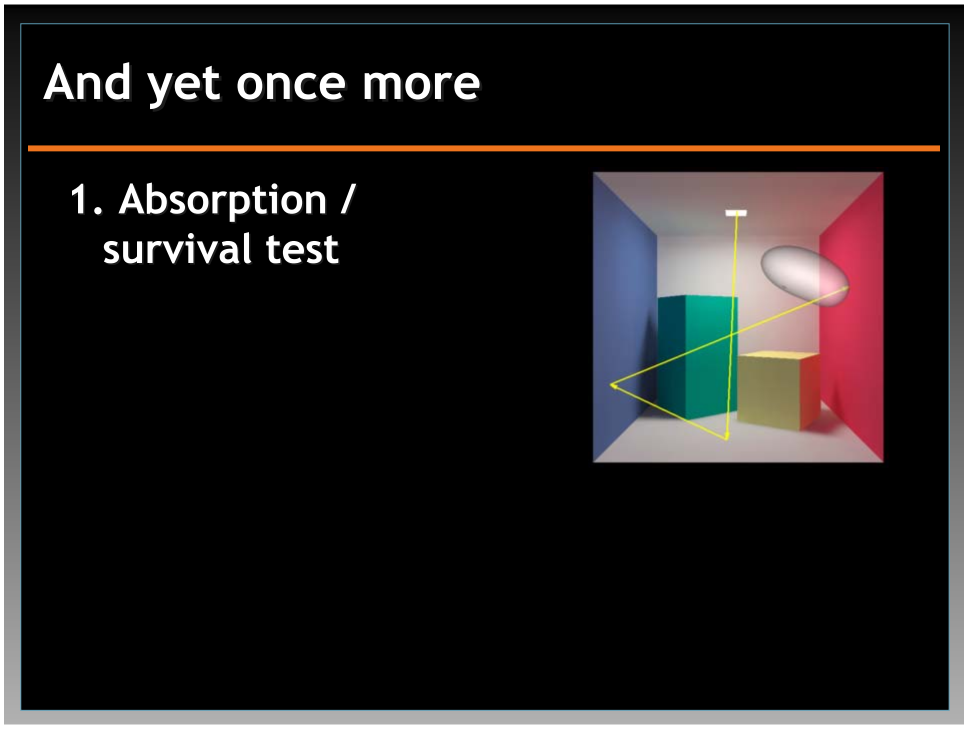


3. Shoot ray.



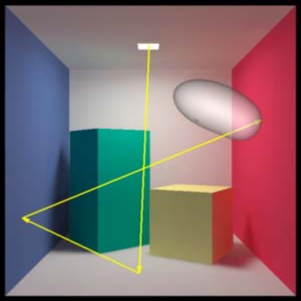


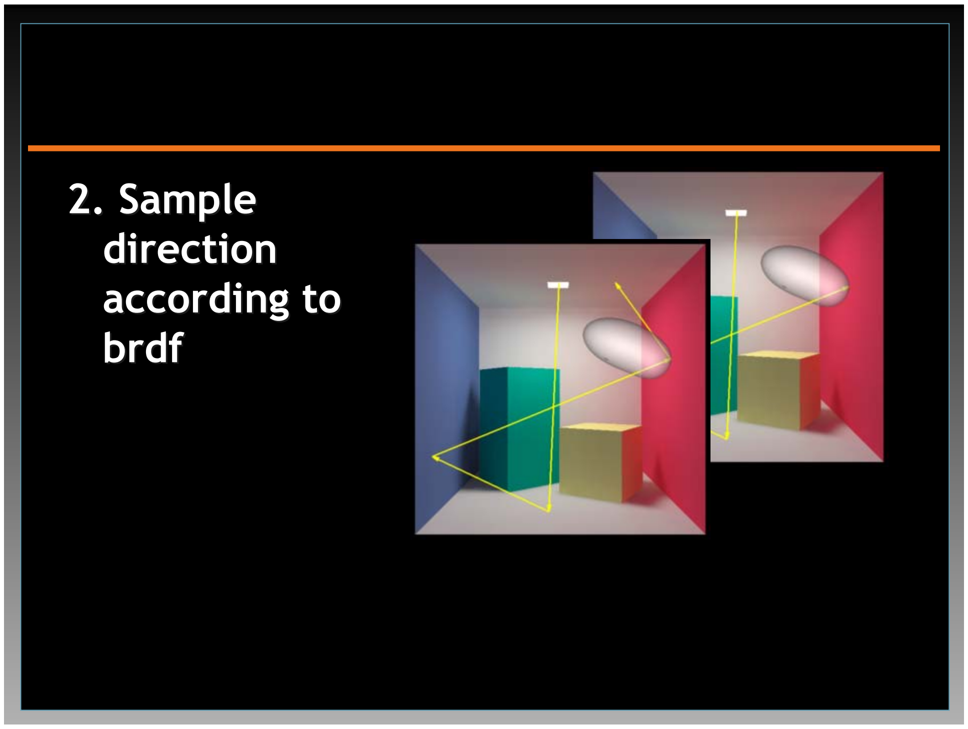
- Full transition density



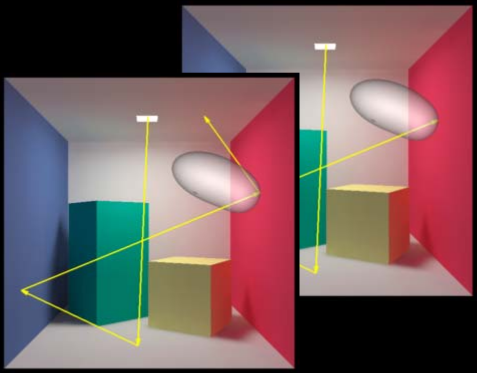
And yet once more

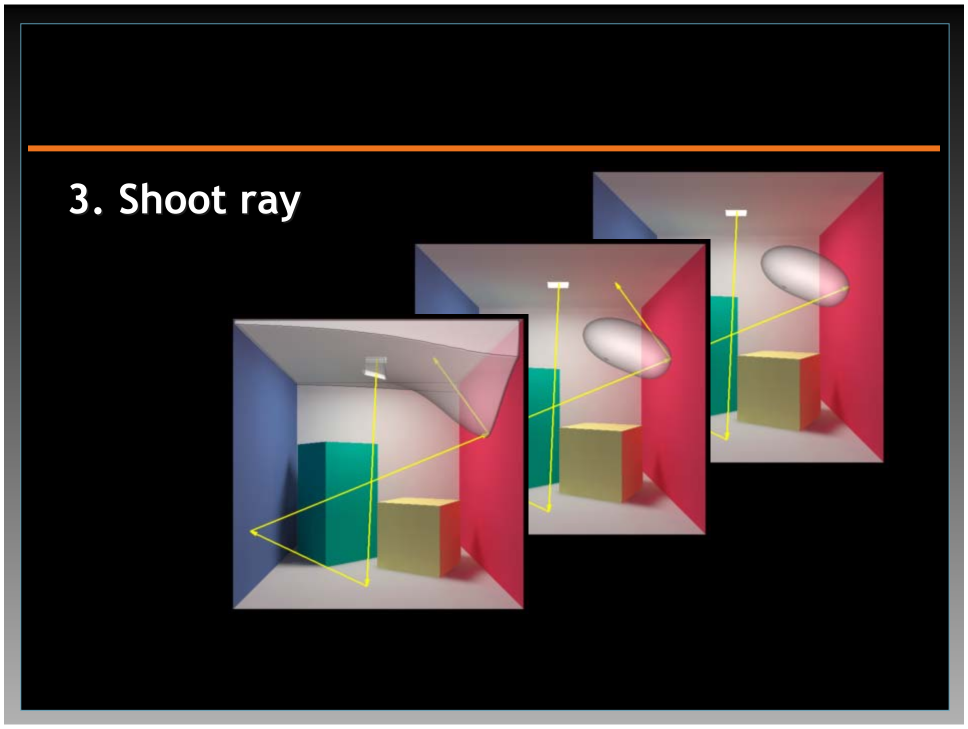
1. Absorption / survival test



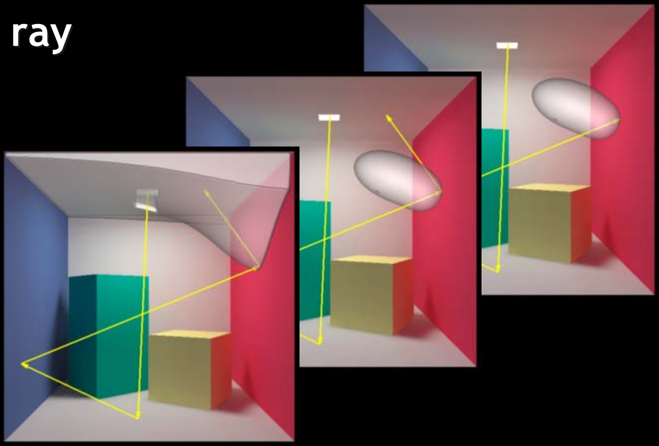


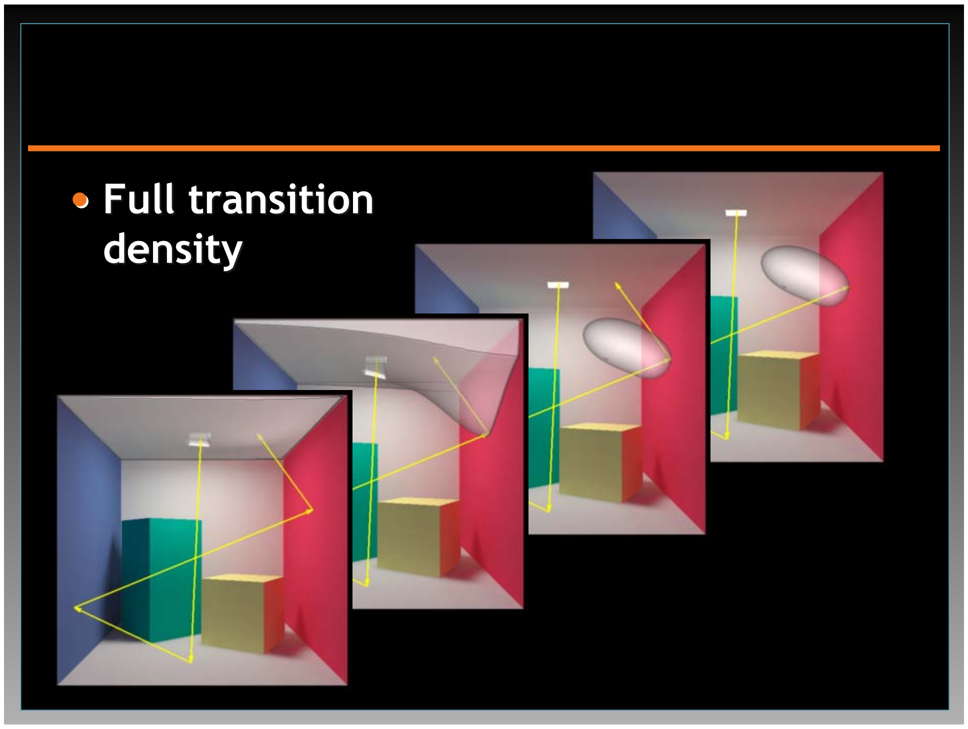
2. Sample direction according to brdf



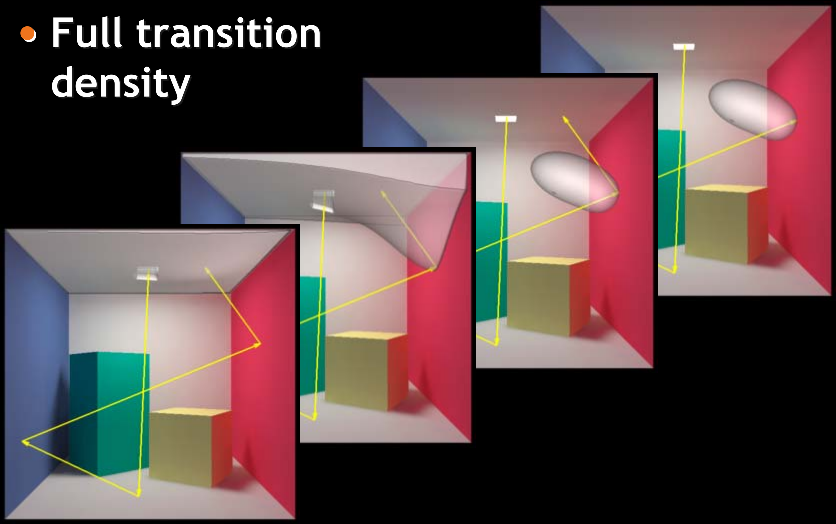


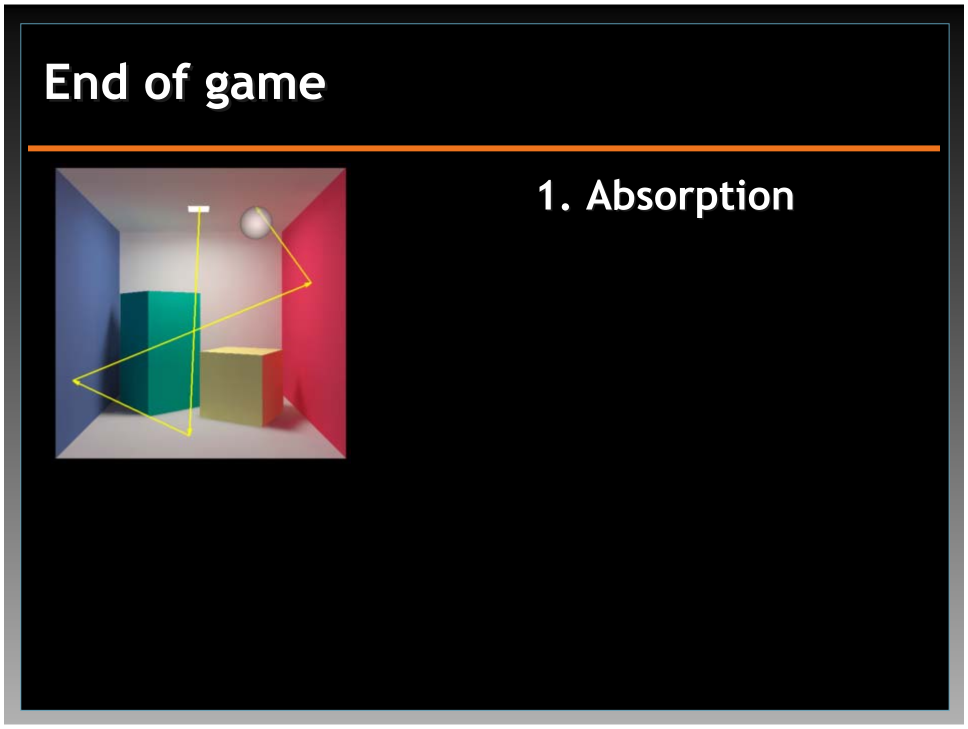
3. Shoot ray



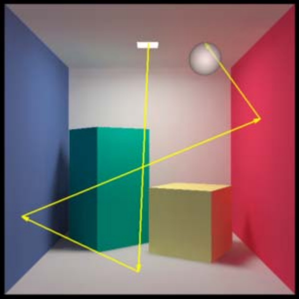


- Full transition density



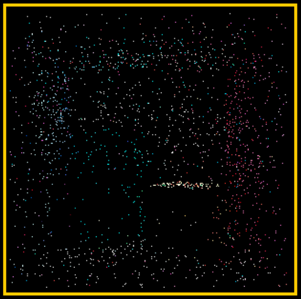


End of game



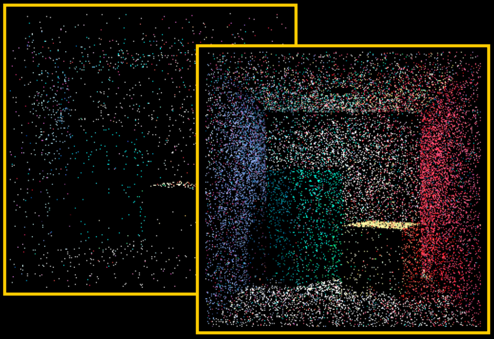
1. Absorption

Sampled points

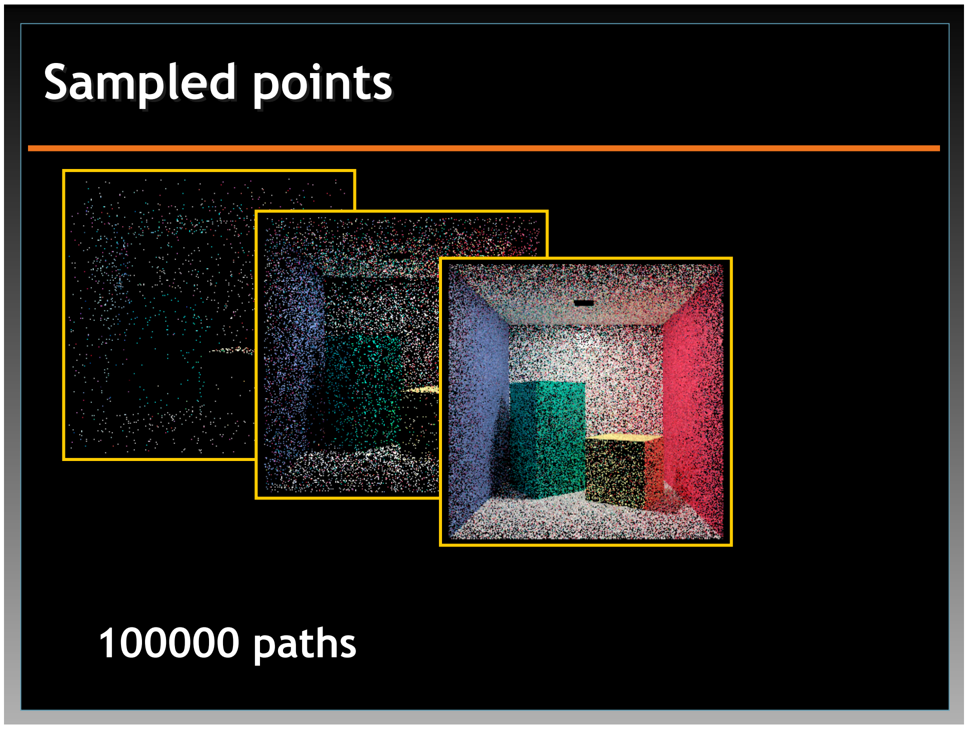


1000 paths

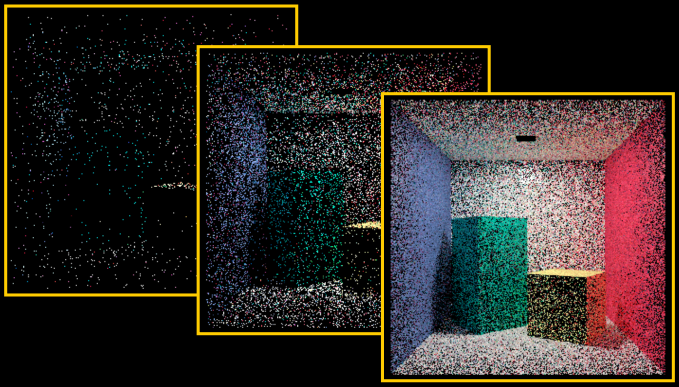
Sampled points



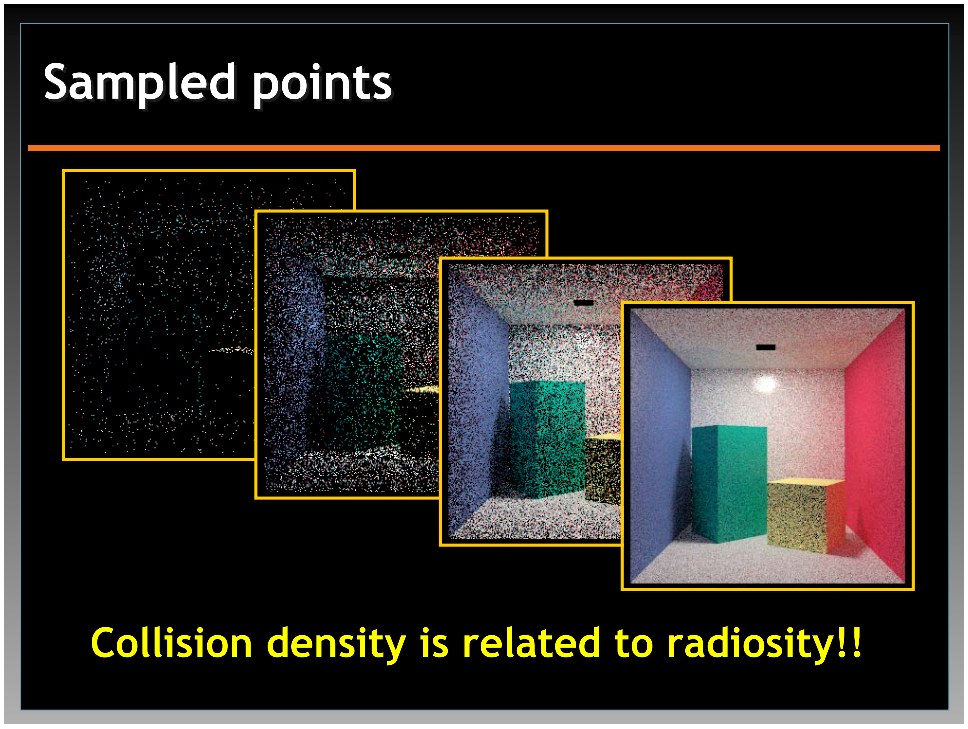
10000 paths



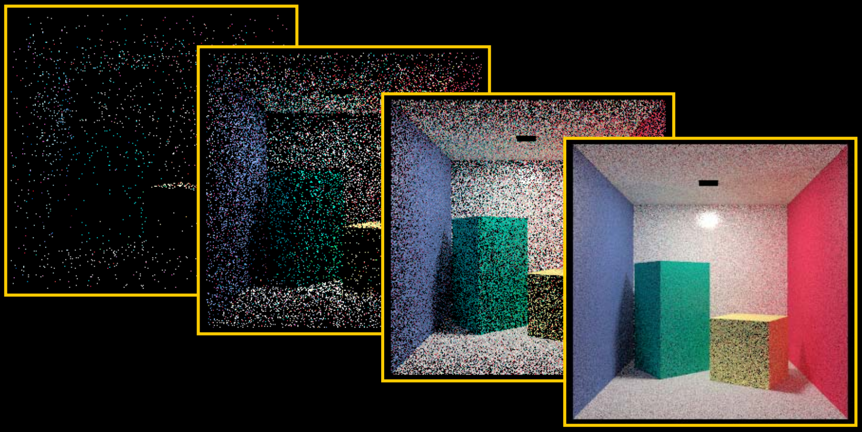
Sampled points



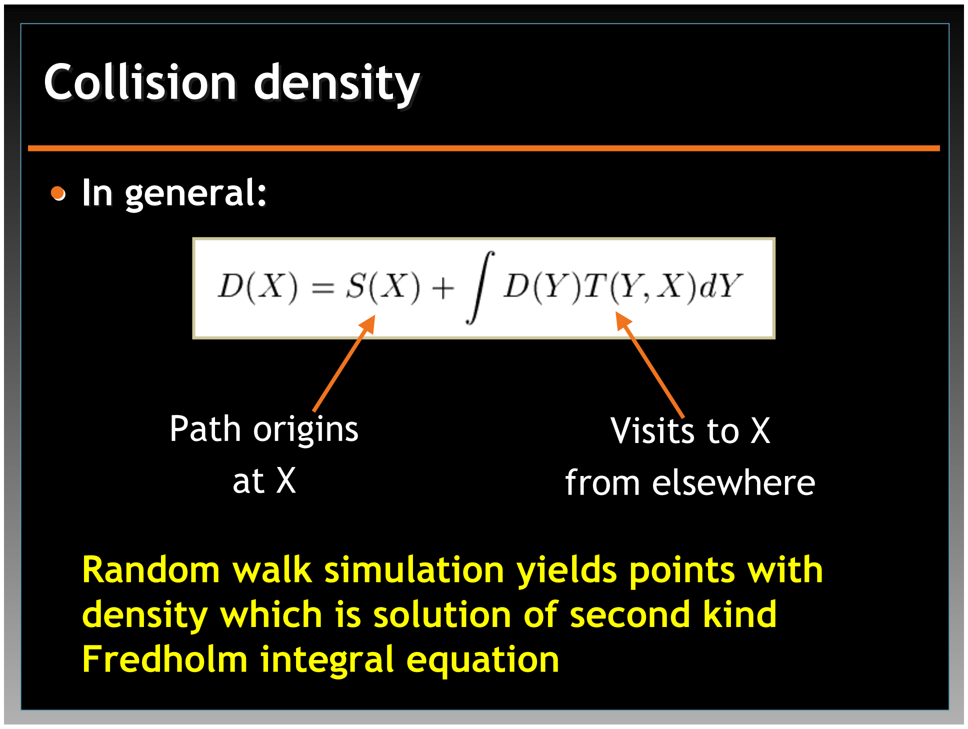
100000 paths



Sampled points



Collision density is related to radiosity!!



Collision density

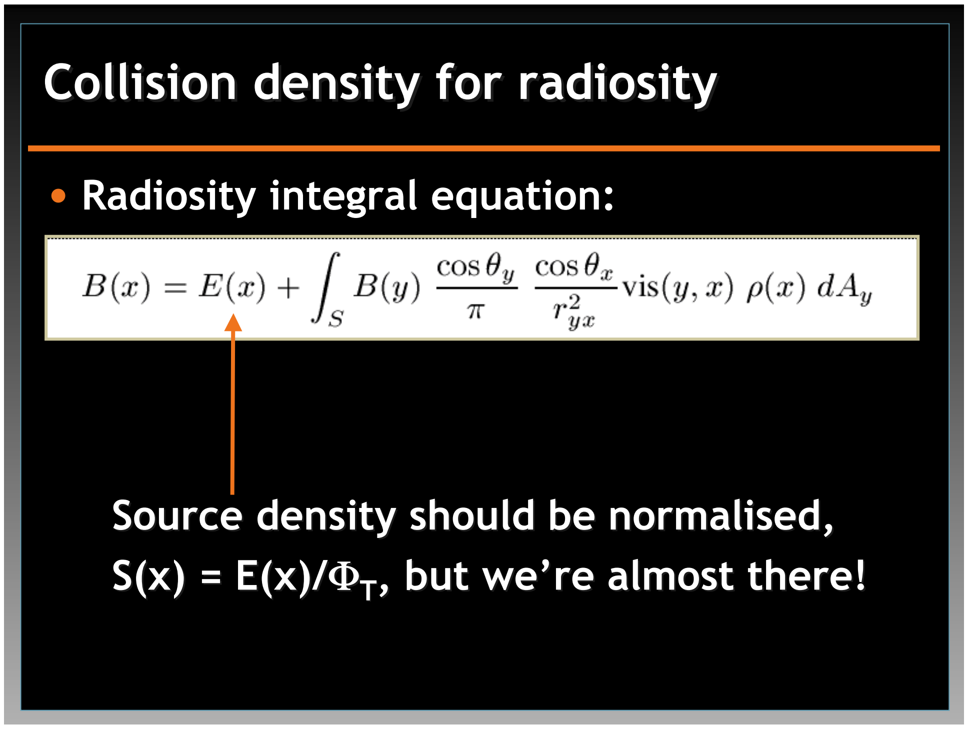
- In general:

$$D(X) = S(X) + \int D(Y)T(Y,X)dY$$

Path origins at X

Visits to X from elsewhere

Random walk simulation yields points with density which is solution of second kind Fredholm integral equation

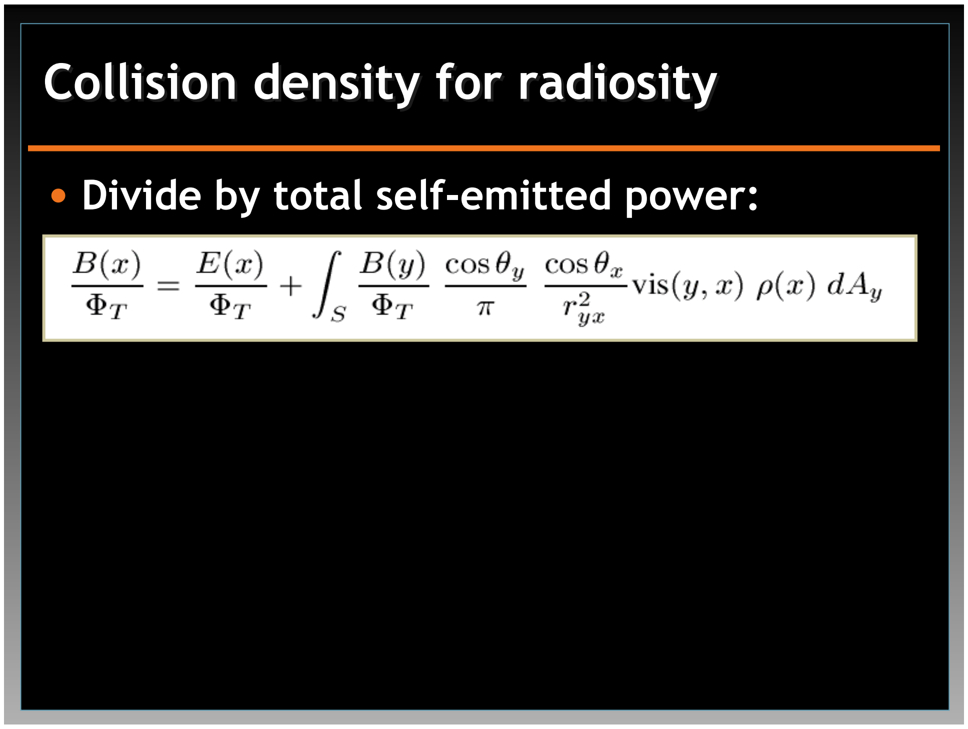


Collision density for radiosity

- Radiosity integral equation:

$$B(x) = E(x) + \int_S B(y)\,\frac{\cos\theta_y}{\pi}\,\frac{\cos\theta_x}{r_{yx}^2}\mathrm{vis}(y,x)\;\rho(x)\;dA_y$$

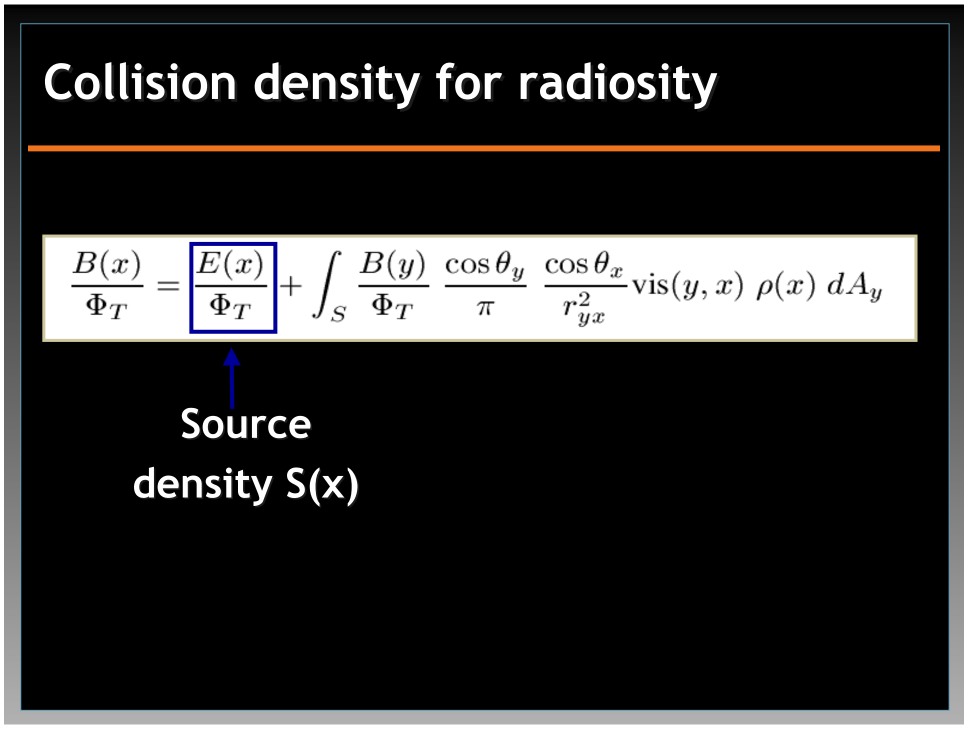
Source density should be normalised, $S(x) = E(x)/\Phi_T$, but we're almost there!



Collision density for radiosity

- Divide by total self-emitted power:

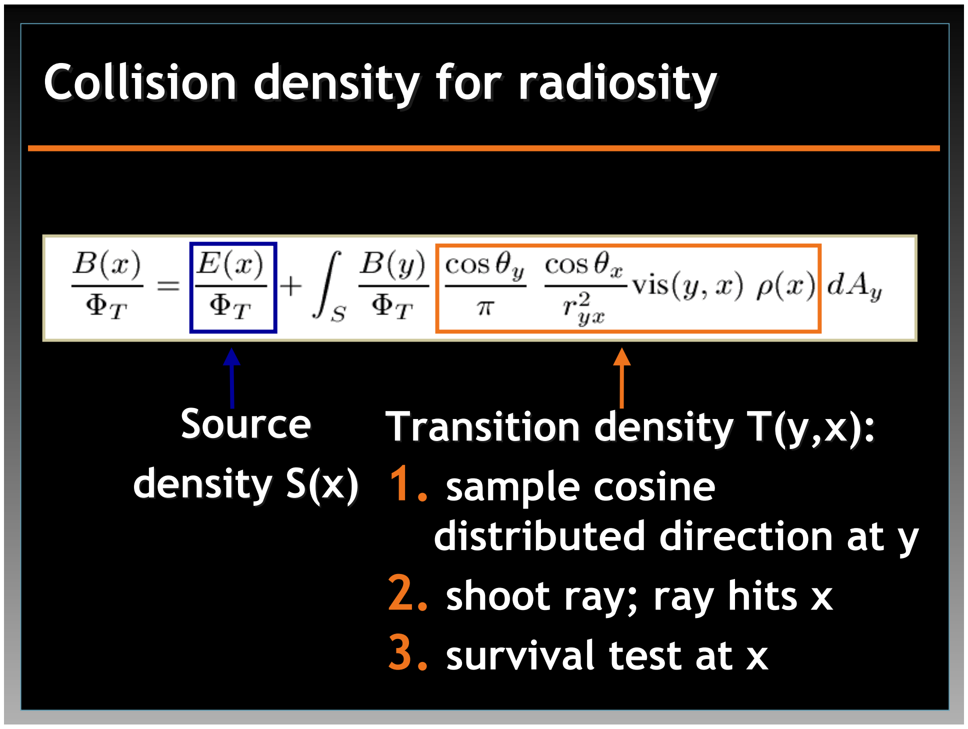
$$\frac{B(x)}{\Phi_T} = \frac{E(x)}{\Phi_T} + \int_S \frac{B(y)}{\Phi_T} \frac{\cos\theta_y}{\pi} \frac{\cos\theta_x}{r_{yx}^2} \mathrm{vis}(y,x)\ \rho(x)\ dA_y$$



Collision density for radiosity

$$\frac{B(x)}{\Phi_T} = \boxed{\frac{E(x)}{\Phi_T}} + \int_S \frac{B(y)}{\Phi_T} \frac{\cos\theta_y}{\pi} \frac{\cos\theta_x}{r_{yx}^2} \mathrm{vis}(y,x)\ \rho(x)\ dA_y$$

Source density S(x)



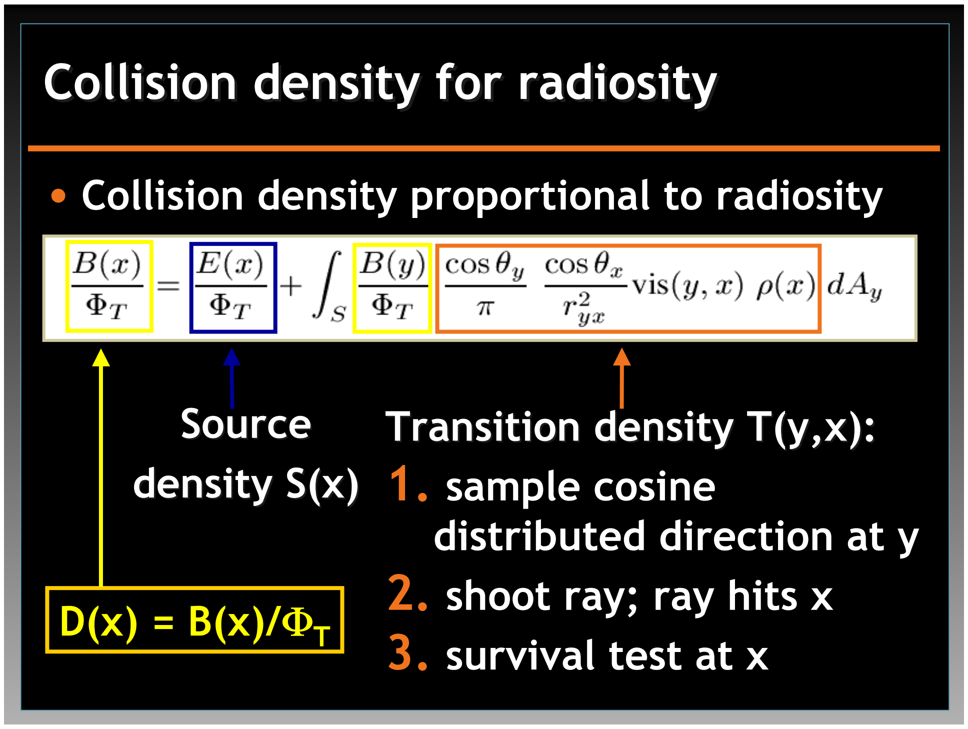
Collision density for radiosity

$$\frac{B(x)}{\Phi_T} = \boxed{\frac{E(x)}{\Phi_T}} + \int_S \frac{B(y)}{\Phi_T} \boxed{\frac{\cos\theta_y}{\pi} \frac{\cos\theta_x}{r_{yx}^2} \mathrm{vis}(y,x)\ \rho(x)}\, dA_y$$

Source density S(x)

Transition density T(y,x):

1. sample cosine distributed direction at y
2. shoot ray; ray hits x
3. survival test at x



Collision density for radiosity

- Collision density proportional to radiosity

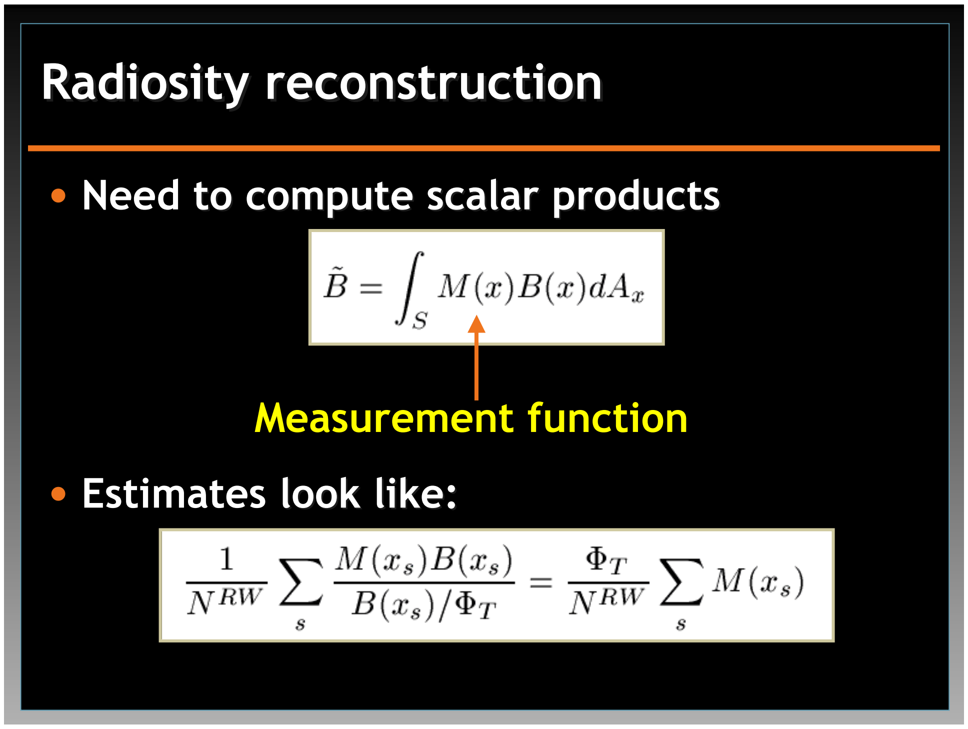
$$\boxed{\frac{B(x)}{\Phi_T}} = \boxed{\frac{E(x)}{\Phi_T}} + \int_S \boxed{\frac{B(y)}{\Phi_T}} \boxed{\frac{\cos\theta_y}{\pi} \frac{\cos\theta_x}{r_{yx}^2} \mathrm{vis}(y,x)\ \rho(x)}\, dA_y$$

Source density S(x)

Transition density T(y,x):

1. sample cosine distributed direction at y
2. shoot ray; ray hits x
3. survival test at x

$\boxed{D(x) = B(x)/\Phi_T}$



Radiosity reconstruction

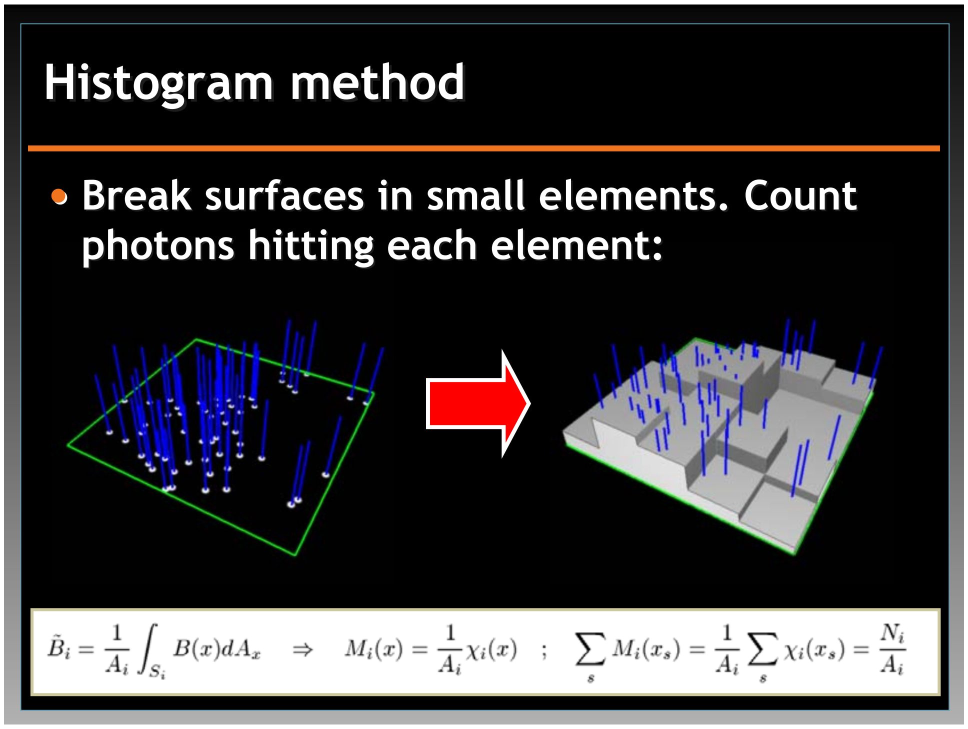
- Need to compute scalar products

$$\tilde{B} = \int_S M(x)B(x)dA_x$$

Measurement function

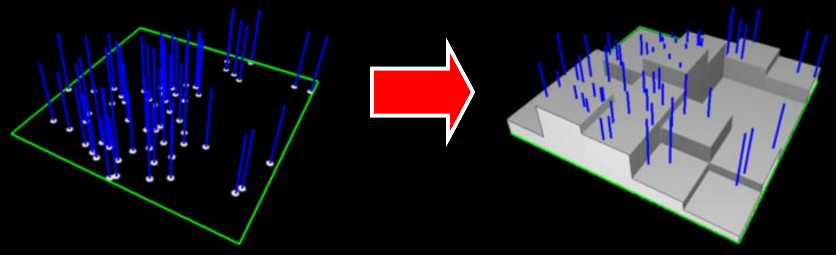
- Estimates look like:

$$\frac{1}{N^{RW}}\sum_s \frac{M(x_s)B(x_s)}{B(x_s)/\Phi_T} = \frac{\Phi_T}{N^{RW}}\sum_s M(x_s)$$



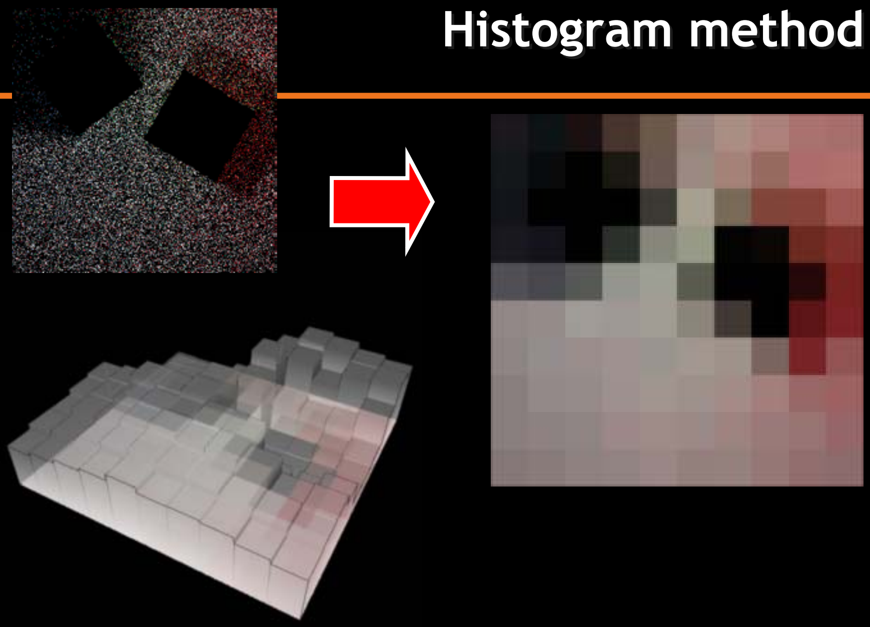
Histogram method

- Break surfaces in small elements. Count photons hitting each element:



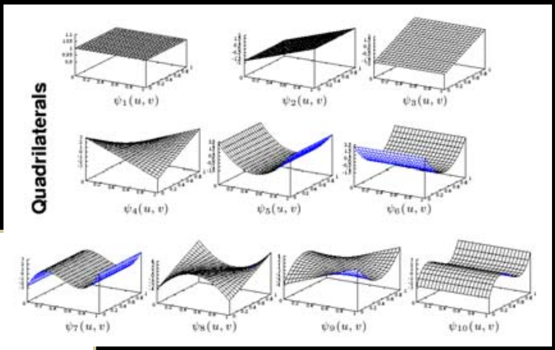
$$\tilde{B}_i = \frac{1}{A_i}\int_{S_i} B(x)dA_x \quad \Rightarrow \quad M_i(x) = \frac{1}{A_i}\chi_i(x) \quad ; \quad \sum_s M_i(x_s) = \frac{1}{A_i}\sum_s \chi_i(x_s) = \frac{N_i}{A_i}$$

Histogram method



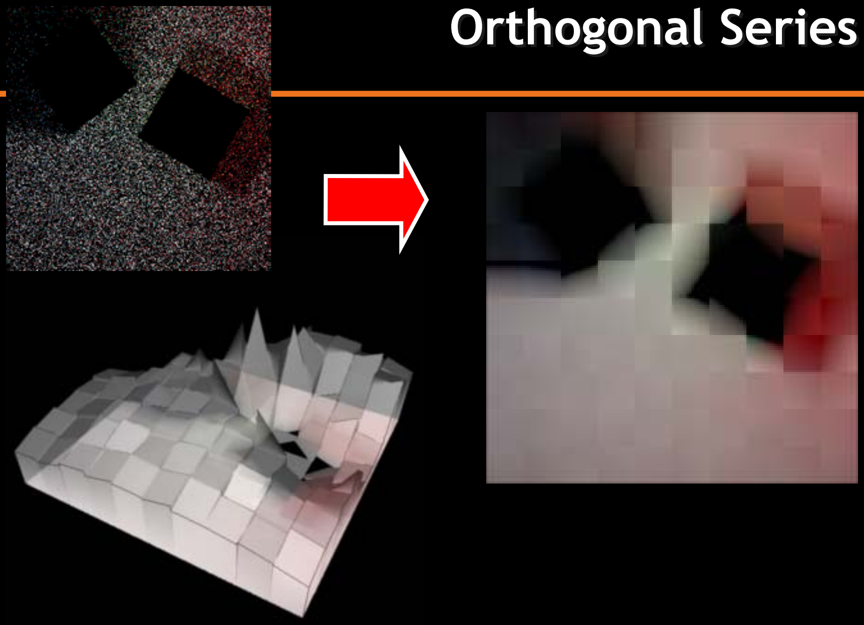
Orthogonal series density estimation

- Linear, bi-linear, quadratic, cubic, … approximations



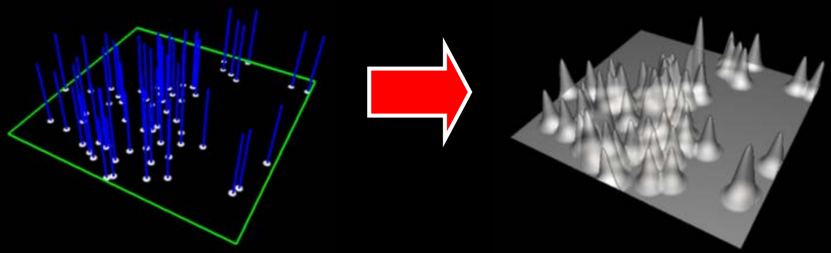
$$\begin{aligned}\tilde{B}(z) &= \sum_\alpha B_\alpha \psi_\alpha(z)\\ B_\alpha &= \int_S B(x)\tilde{\psi}_\alpha(x)\,dA_x\\ &\Rightarrow M_z(x) = \sum_\alpha \tilde{\psi}_\alpha(x)\psi_\alpha(z)\end{aligned}$$

Orthogonal Series

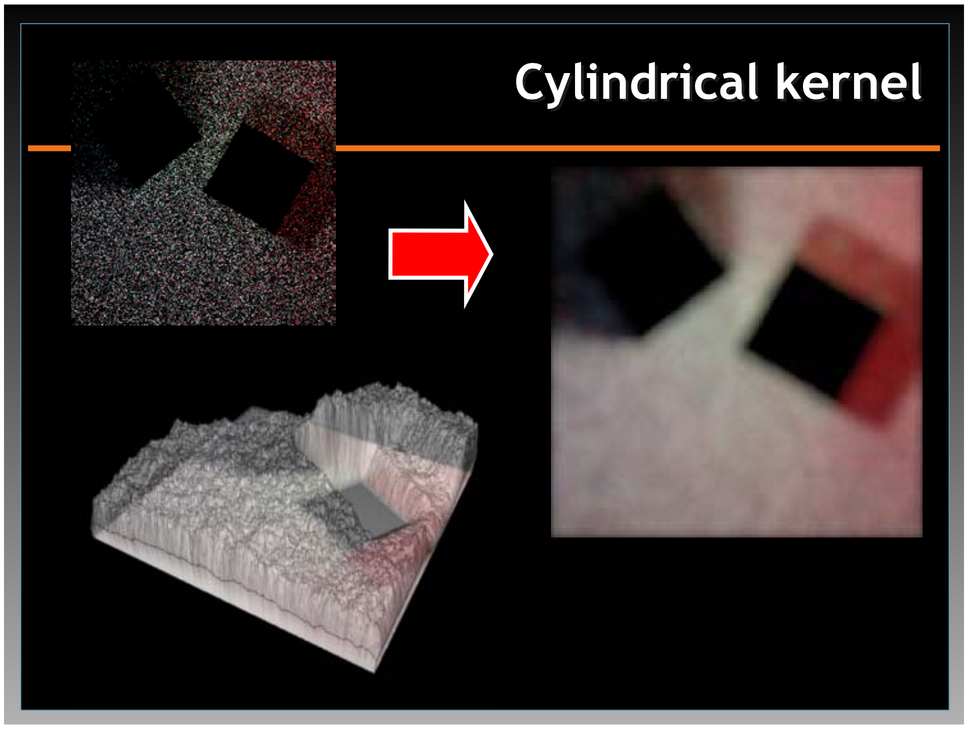


Kernel density estimation

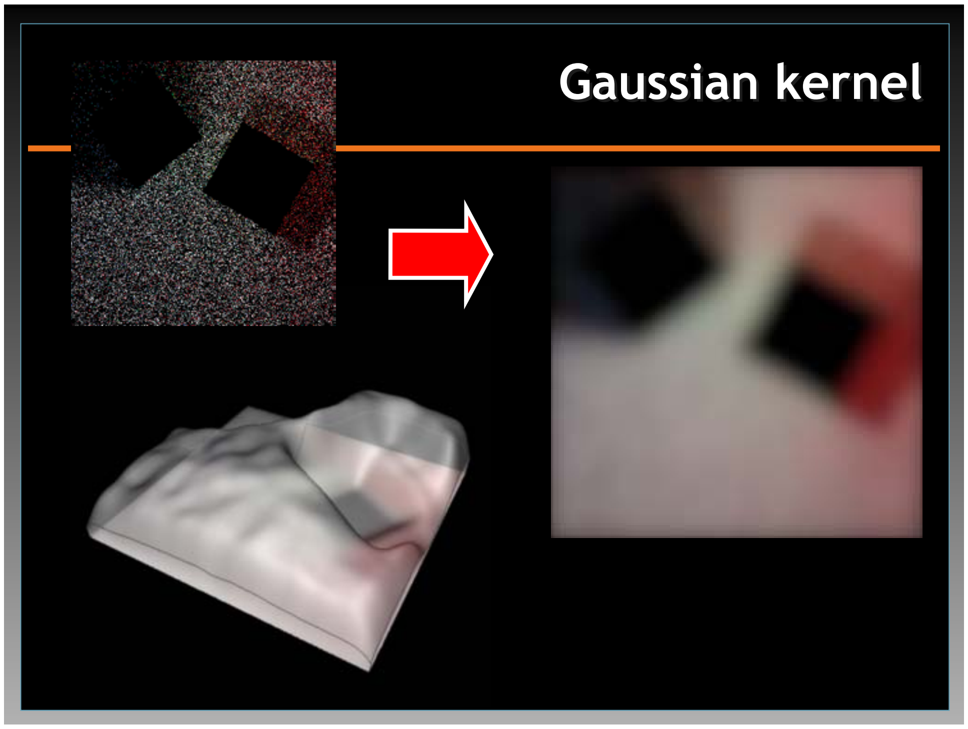
- Place finite-width density kernel at each sample point.



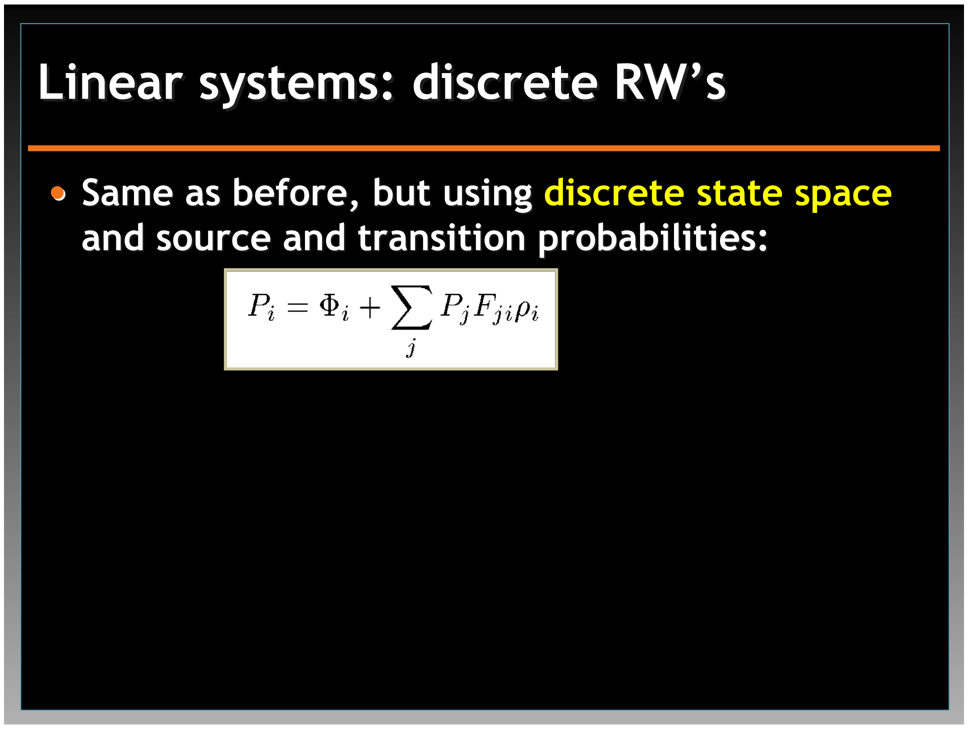
$$\tilde{B}(z) = \int_S K(z-x)B(x)dA_x \quad \Rightarrow \quad M_z(x) = K(z-x) \quad ; \quad \tilde{B}(z) \approx \frac{\Phi_T}{N^{RW}} \sum_s K(z-x_s)$$



Cylindrical kernel



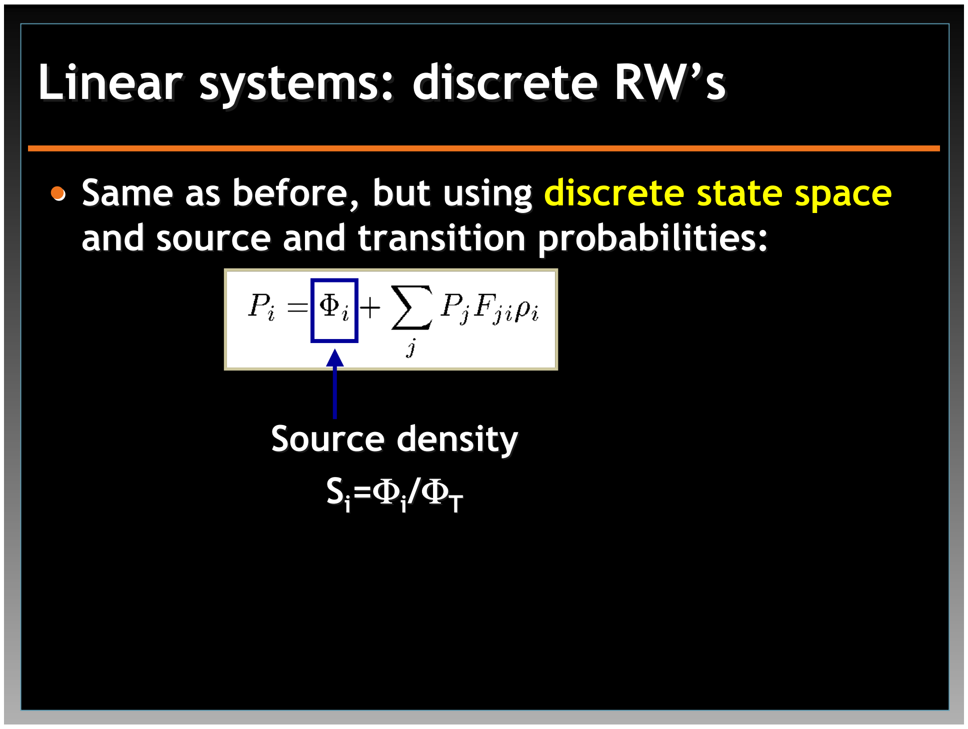
Gaussian kernel



Linear systems: discrete RW's

- Same as before, but using discrete state space and source and transition probabilities:

$$P_i = \Phi_i + \sum_j P_j F_{ji} \rho_i$$



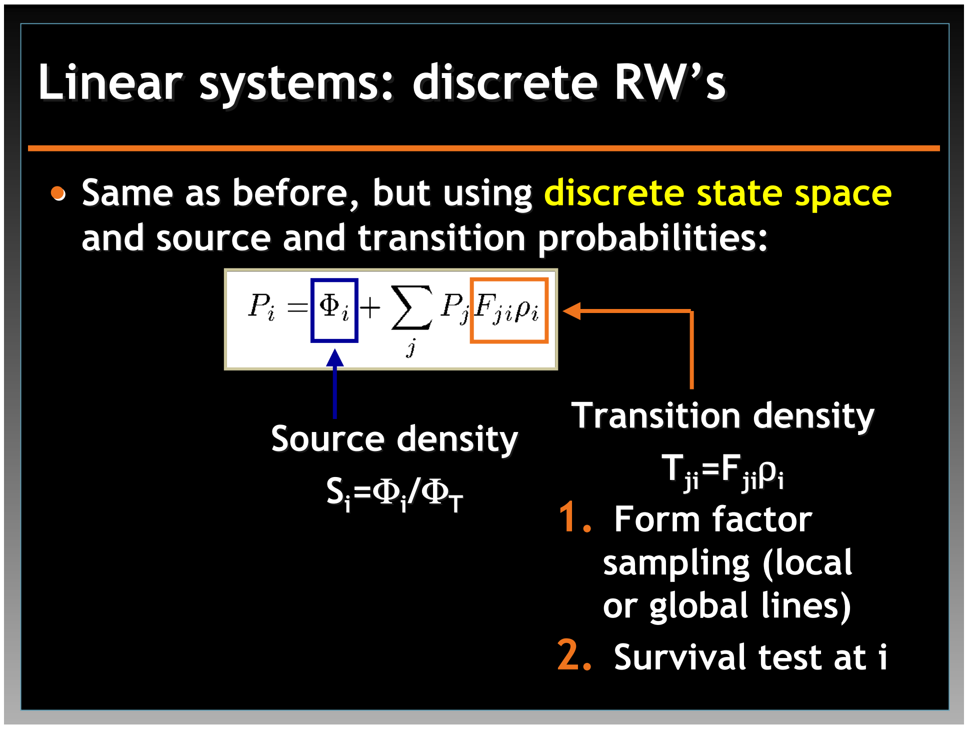
Linear systems: discrete RW's

- Same as before, but using discrete state space and source and transition probabilities:

$$P_i = \boxed{\Phi_i} + \sum_j P_j F_{ji} \rho_i$$

Source density

$S_i=\Phi_i/\Phi_T$



Linear systems: discrete RW’s

- Same as before, but using discrete state space and source and transition probabilities:

$$P_i = \boxed{\Phi_i} + \sum_j P_j \boxed{F_{ji}\rho_i}$$

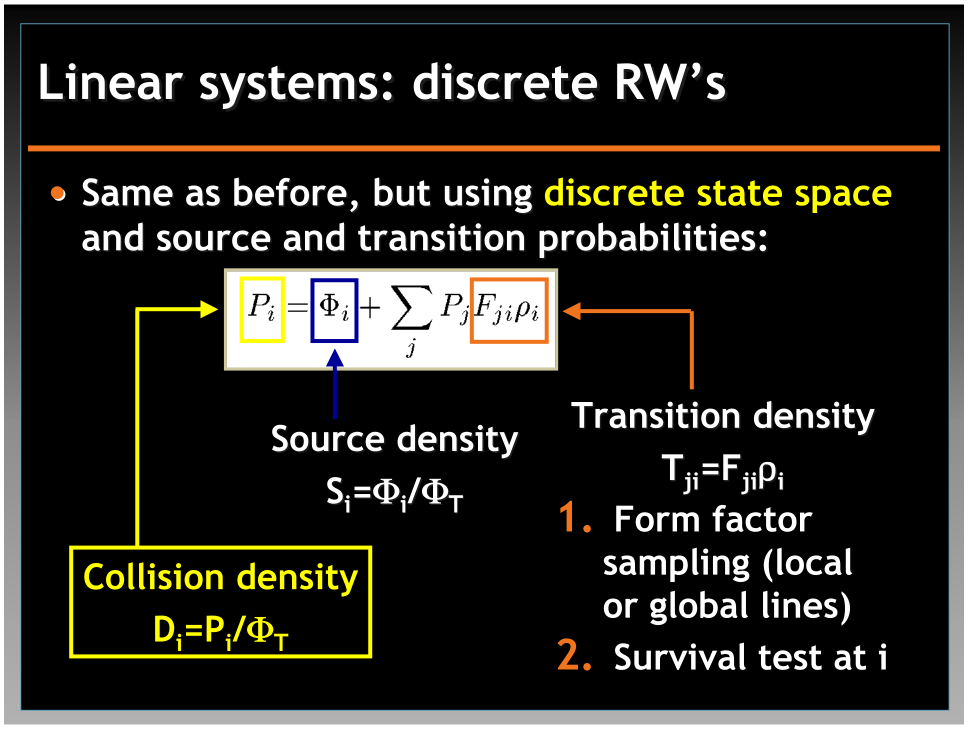
Source density

$S_i=\Phi_i/\Phi_T$

Transition density

$T_{ji}=F_{ji}\rho_i$

1. Form factor sampling (local or global lines)
2. Survival test at i



Linear systems: discrete RW’s

- Same as before, but using discrete state space and source and transition probabilities:

$$\boxed{P_i} = \boxed{\Phi_i} + \sum_j P_j \boxed{F_{ji}\rho_i}$$

Source density

$S_i=\Phi_i/\Phi_T$

Collision density

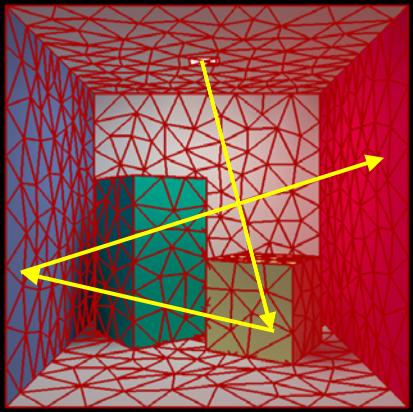
$D_i=P_i/\Phi_T$

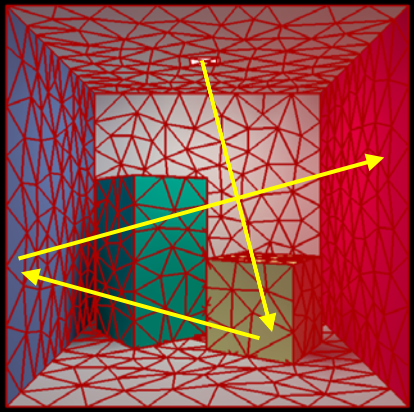
Transition density

$T_{ji}=F_{ji}\rho_i$

1. Form factor sampling (local or global lines)
2. Survival test at i

Discrete Random Walks (local lines)


Continuous
(solves integral equation)


Discrete
(solves linear system)

Ad-joint systems: gathering RW's

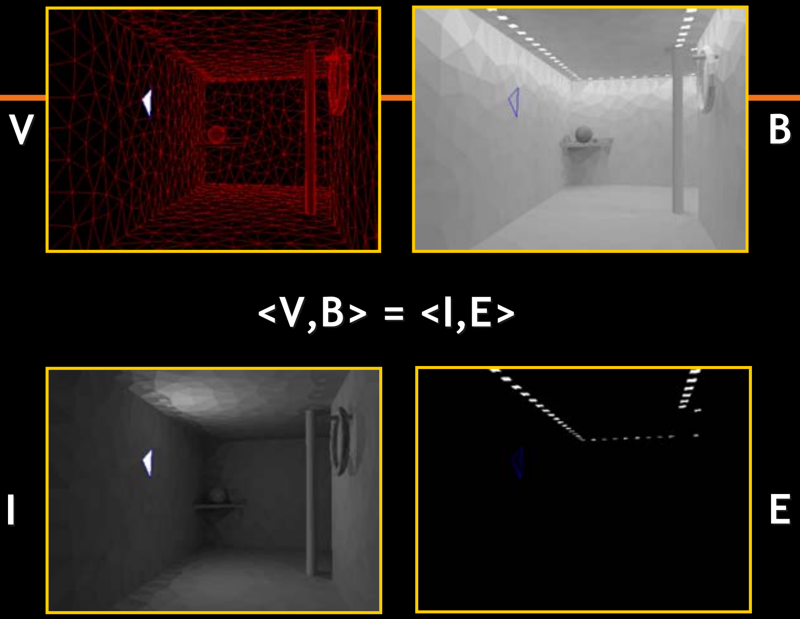
- With each scalar product <V,B> with TB=E corresponds a scalar product <I,E> with I the solution of an ad-joint equation T*I=V:

  Proof: <V,B>=<T*I,B>=<I,TB>=<I,E>

- Ad-joint radiosity systems:

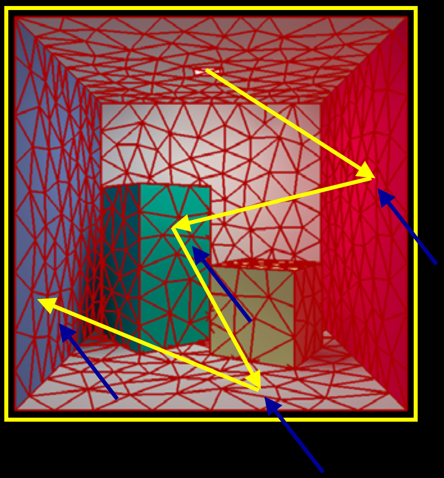
$$\boxed{I_i = V_i + \sum_j F_{ij} \rho_j I_j.}$$

- Random walks start at "region of interest" and yield scores when hitting light sources



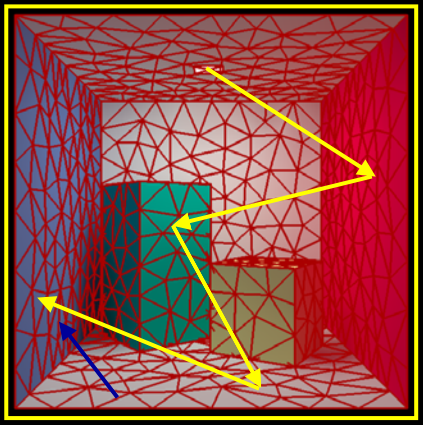
$\langle V,B \rangle = \langle I,E \rangle$

Scoring (1)



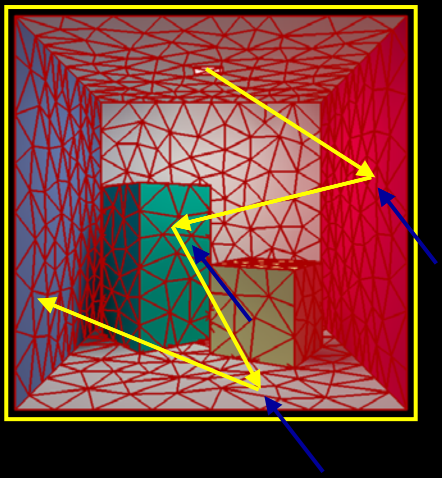
Collision Estimator:
scores everywhere

Scoring (2)



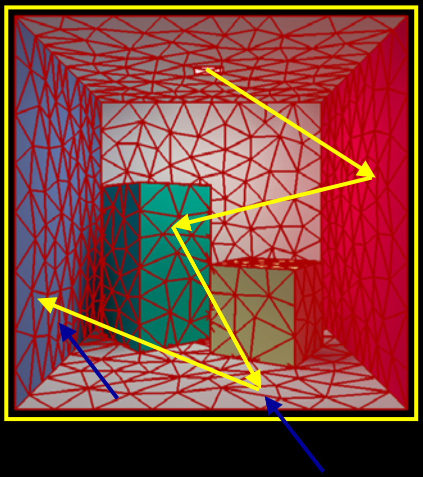
Absorption estimator:
scores only where
path terminates

Scoring (3)



Survival estimator:
scores everywhere
except when
absorbed

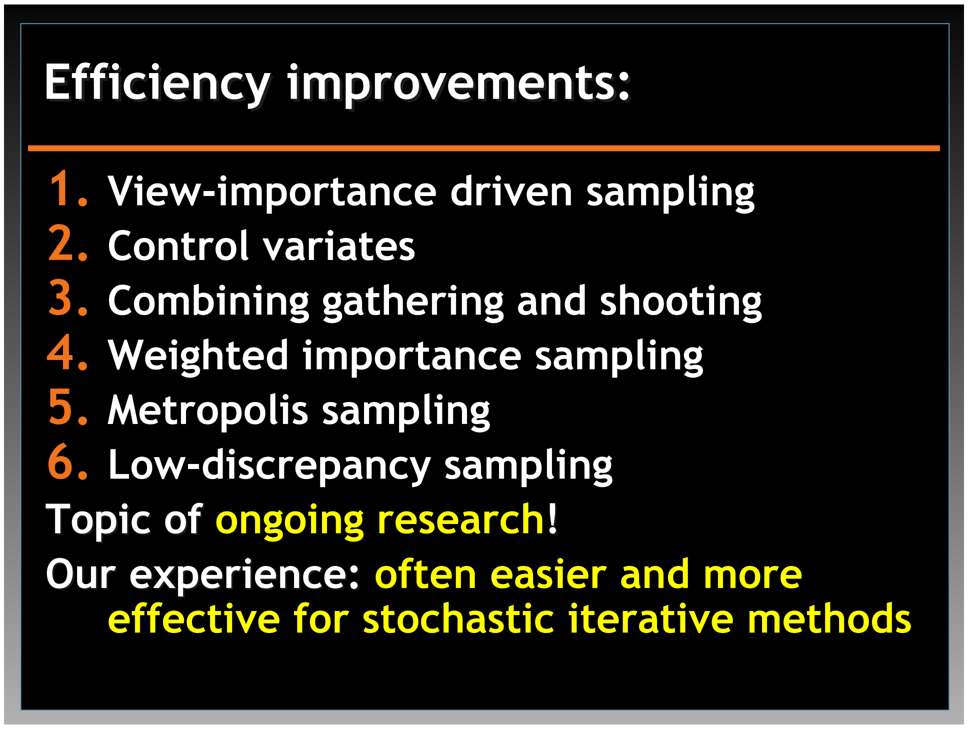
Scoring (4)



Many more Possibilities!

Comparison

1. **Discrete versus continuous RW (histogram):**
   Slightly higher discretisation error, but QMC sampling is much more effective for discrete RW.
2. **Scoring:**
   Collision RW most often preferable
3. **Gathering versus shooting:**
   Shooting more effective, except on small patches
4. **Discrete collision shooting RW versus stochastic iterations:**
   Basic estimators equally efficient, but variance reduction more easy and effective for stochastic iterations.

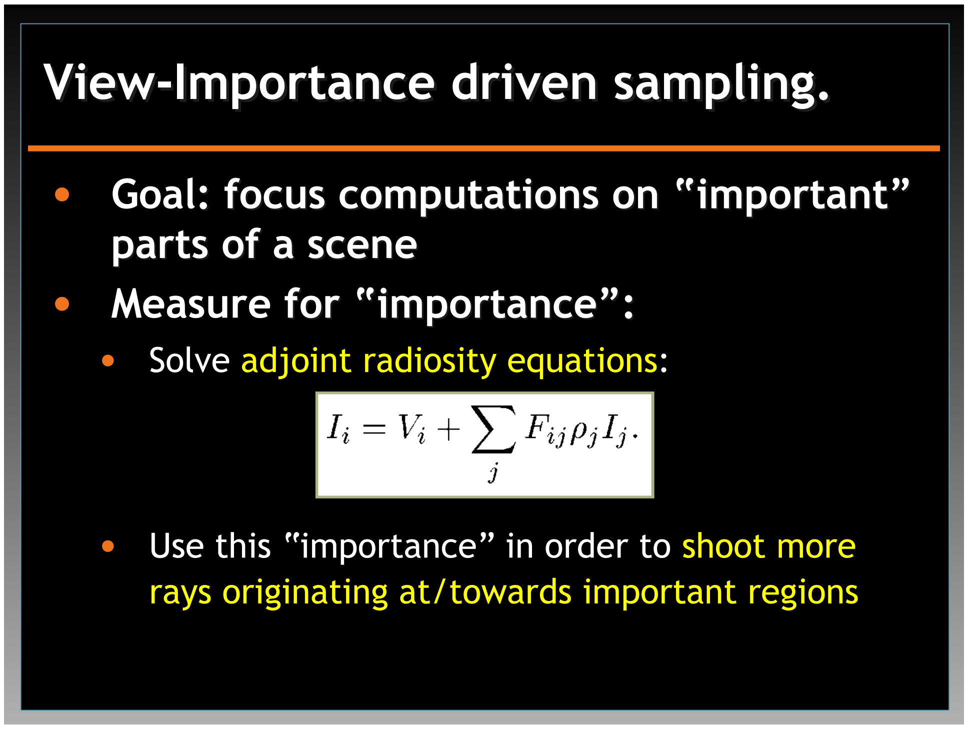


Efficiency improvements:

1. View-importance driven sampling
2. Control variates
3. Combining gathering and shooting
4. Weighted importance sampling
5. Metropolis sampling
6. Low-discrepancy sampling

Topic of ongoing research!

Our experience: often easier and more effective for stochastic iterative methods



View-Importance driven sampling.

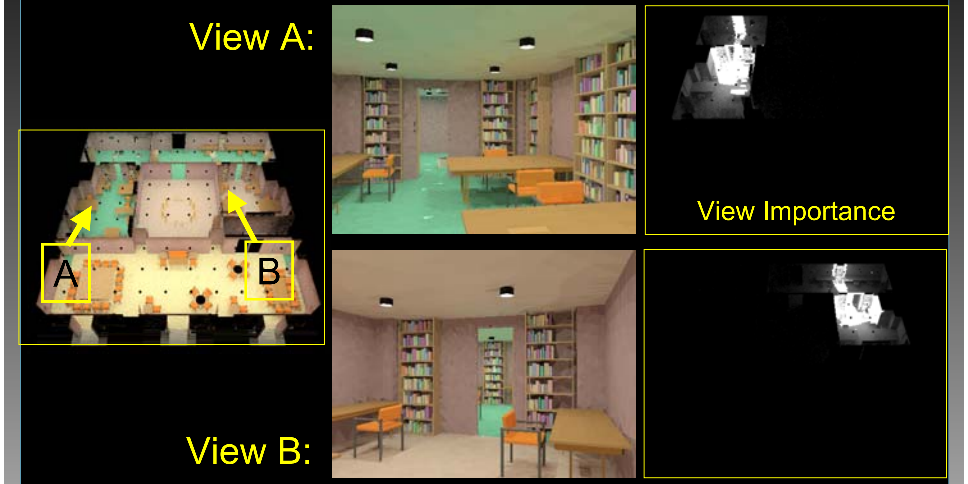
- Goal: focus computations on "important" parts of a scene
- Measure for "importance":
  - Solve adjoint radiosity equations:

$$I_i = V_i + \sum_j F_{ij} \rho_j I_j .$$

  - Use this "importance" in order to shoot more rays originating at/towards important regions

View Importance:

View A:



View B:

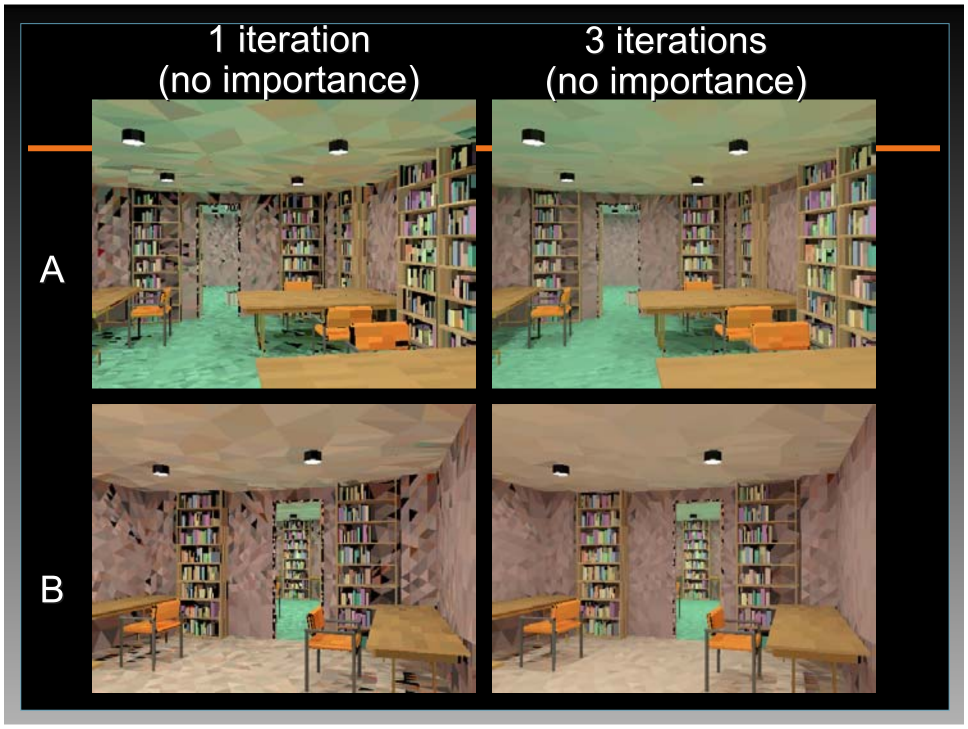
View-importance driven stochastic iterations

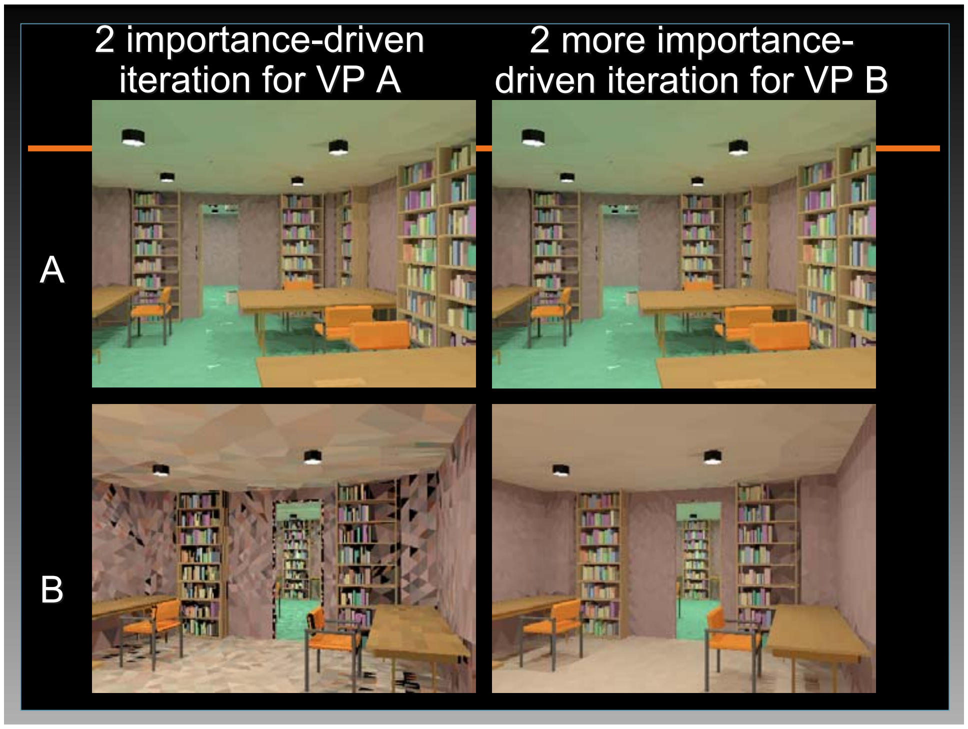
$$P_i I_i = \Phi_i I_i + \sum_j P_j I_j F_{ji} \rho_i \frac{I_i}{I_j}$$

A: Sample more rays originating at patches with high importance

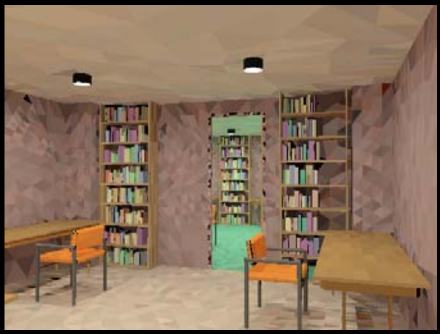
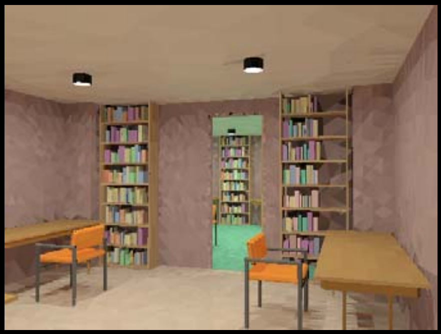
B: Sample more rays originating at patches with high indirect importance

C: B + also aim rays towards regions with high indirect importance


1 iteration
(no importance)
3 iterations
(no importance)
A
B


2 importance-driven
iteration for VP A
2 more importance-
driven iteration for VP B
A
B

View-importance driven iteration is more expensive than non-importance driven iteration. For same computation cost:

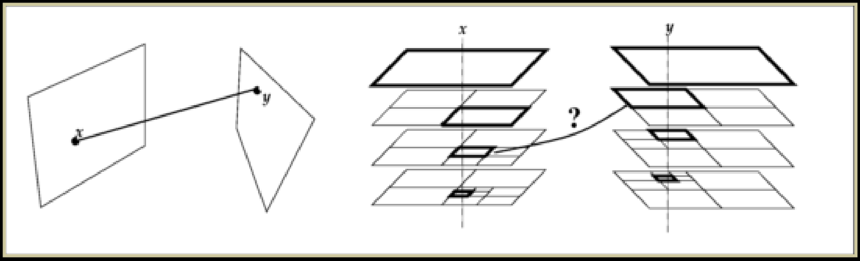
Hierarchical Refinement

Problem:

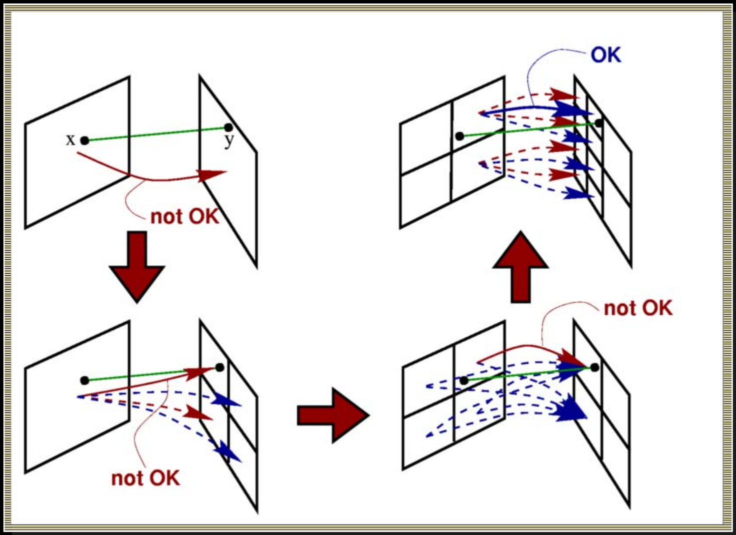
- Noisy artifacts on small patches: group them!
- Large patches are too “flat”: subdivide them!

Per-ray refinement

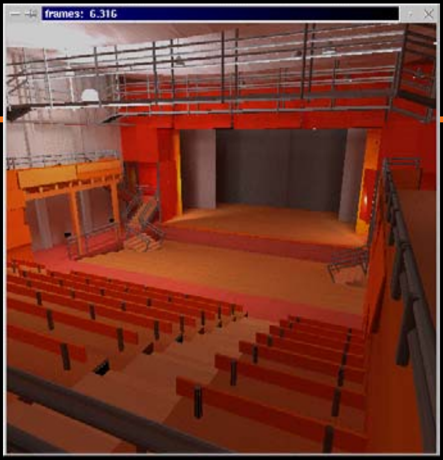
Principle:



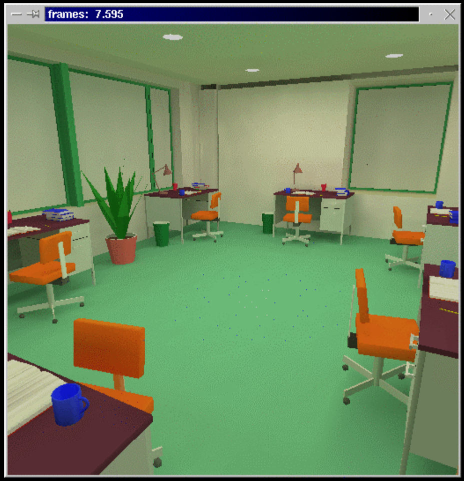
Related with Monte Carlo solution of wavelet-preconditioned linear systems.



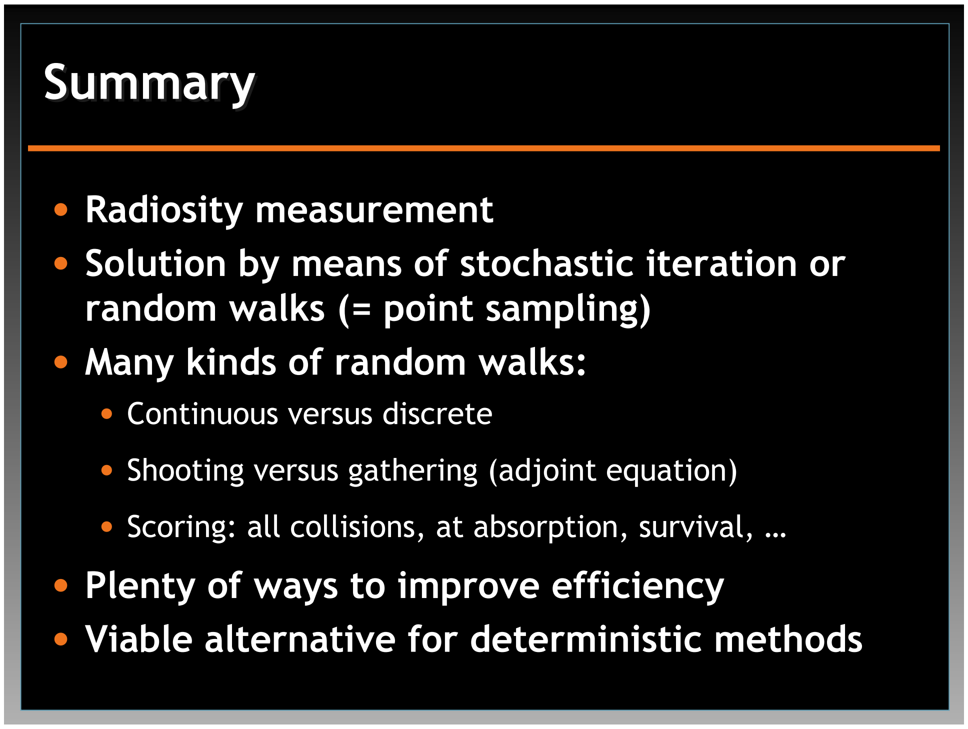
87



8Mpoly's in 1 night

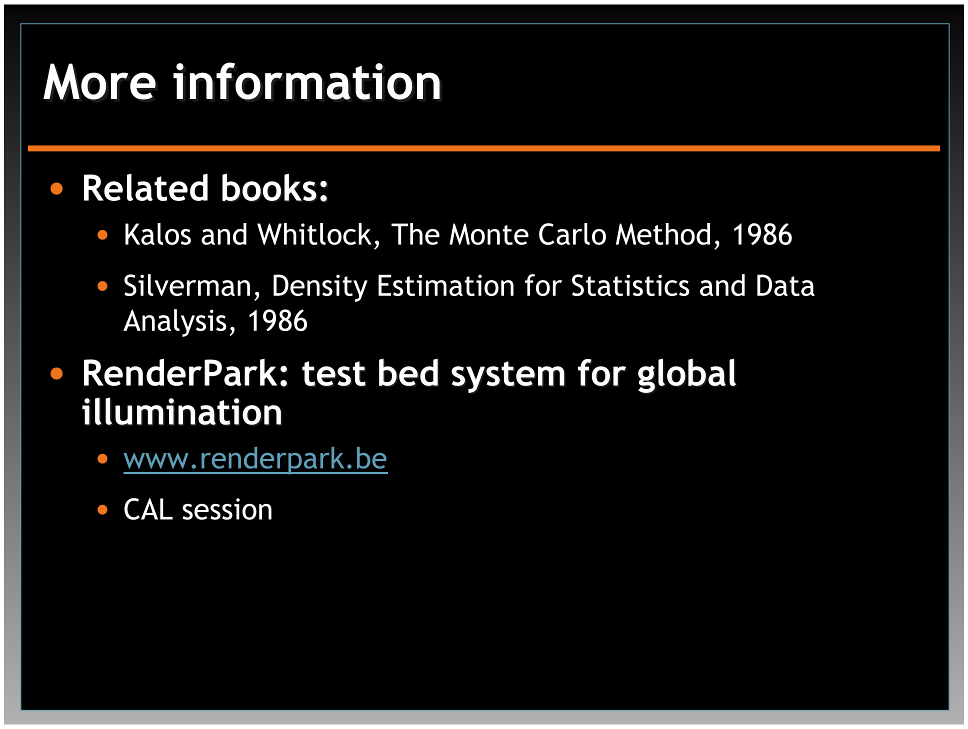


(interactive visualisation with ray casting)



Summary

- Radiosity measurement
- Solution by means of stochastic iteration or random walks (= point sampling)
- Many kinds of random walks:
  - Continuous versus discrete
  - Shooting versus gathering (adjoint equation)
  - Scoring: all collisions, at absorption, survival, …
- Plenty of ways to improve efficiency
- Viable alternative for deterministic methods



More information

- Related books:
  - Kalos and Whitlock, The Monte Carlo Method, 1986
  - Silverman, Density Estimation for Statistics and Data Analysis, 1986
- RenderPark: test bed system for global illumination
  - www.renderpark.be
  - CAL session

Stochastic Radiosity
Doing Radiosity without Form Factors

Ph. Bekaert[*]

Introduction

This course section focuses on algorithms to compute light transport in purely diffuse 3D environments. The input consists of a model of a 3-dimensional scene, with surfaces broken up in patches, most often triangles or convex quadrilaterals. With each patch the self-emitted radiosity E_i (dimensions: $[W/m^2]$) and reflectivity ρ_i (dimensionless) are given. The self-emitted radiosity is the radiosity that a patch emits "on its own", even if there were no other patches in the model, or all other patches were perfectly black. The reflectivity is a number (for each considered wavelength) between 0 and 1. It indicates what fraction of the power incident on the patch gets reflected (the rest gets absorbed). These data suffice in order to compute the total emitted radiosity B_i (dimension: $[W/m^2]$) by each patch, containing besides the self-emitted radiosity, also the radiosity received via any number of bounces from other patches in the scene. The problem of computing B_i is commonly called the *radiosity problem* [18, 9, 43].

The restriction to diffuse emission and reflection may seem draconic at first sight. In addition, subdividing a scene into patches limits what kind of geometry descriptions can be used. There are other light transport algorithms that do not suffer these limitations. We believe that radiosity is nevertheless an interesting topic for study, because:

- diffuse reflection is a reasonable approximation in many indoor and outdoor environments, where in particular indirect non-diffuse illumination is often not so important;
- tessellation of input models is also required in other rendering algorithms, for instance when using graphics hardware. Although mesh quality requirements are higher, many meshing problems are not unique for radiosity. Meshing is a topic that has been studied extensively;
- the computed radiosities can be converted into patch and vertex colors, or into a texture, which can be rendered in *real time* using commodity graphics hardware. Reasonable approximations for non-diffuse direct illumination, such as glossy highlights, and even mirror reflections, can also be added using graphics hardware. At this time, this is a unique advantage of radiosity methods;

[*]Current adress: Max Planck Institut für Informatik, Im Stadtwald 46.1, 66123 Saarbrücken, Germany. E-mail: `Philippe.Bekaert@mpi-sb.mpg.de`

- the radiosity problem is a simplified instance of the general light transport problem. Both problems can be solved using the same techniques, at least in theory. We will show in this course that the set of techniques for solving radiosity is richer than for the general light transport problem. We hope the overview in this document may therefore inspire further research for the general light transport problem too.

This course section is structured as follows: first, a concise overview is given of the radiosity method (§1). Our focus is on Monte Carlo algorithms for the radiosity problem, which have been proposed relatively recently. The main underlying principles of these algorithms are presented in §2. Monte Carlo algorithms for radiosity are more reliable and easier to implement and to use than their deterministic counterparts. In addition, they require less storage and yield fair quality images more rapidly in far most cases. These advantages are related with the fact that explicit computation and storage of so called form factors is avoided. There are basically two ways to do this: by stochastic adaptations of classical iterative methods for solving linear systems (§3) and by random walks (§4). Next (§5), several variance reduction techniques and the use of low discrepancy sampling are presented. These techniques can sometimes yield very significant speed-ups. We conclude with a description how higher order approximations and hierarchical refinement can be incorporated in Monte Carlo radiosity algorithms (§6).

All results reported in these notes have been obtained using RENDERPARK on a 195MHz R10000 SGI Octane system with 256MB RAM. RENDERPARK is a free software test bed system for global illumination algorithms, developed at the computer graphics research group of the department of computer science at the K. U. Leuven, Leuven, Belgium. The source code of RENDERPARK can be downloaded from `www.renderpark.be`.

1 The Radiosity Method

In this section, we will present a very concise overview of the radiosity method. More extensive treatments can be found in the books by Cohen and Wallace [11] and Sillion and Puech [63] as well as various SIGGRAPH courses organized in the past. First, a mathematical description of the problem is given. Next, the classical radiosity method is outlined and discussed.

1.1 Mathematical problem description

The radiosity problem can be described mathematically in three different ways: by the general rendering equation, an instance of it for purely diffuse environments, and by a discretized version of the latter.

1.1.1 The general rendering equation

The rendering equation, introduced in the field by Kajiya [25], describes light transport in environments exhibiting general light emission and scattering. The average radiosity B_i emitted by a patch i, in such an environment is given by:

$$B_i = \frac{1}{A_i} \int_{S_i} \int_{\Omega_x} L^{\rightarrow}(x, \Theta) \cos\theta d\omega_\Theta dA_x \tag{1}$$

with

$$L^{\rightarrow}(x, \Theta) = L_e^{\rightarrow}(x, \Theta) + \int_{\Omega_x} f_r(x; \Theta' \leftrightarrow \Theta) L^{\leftarrow}(x, \Theta') \cos\theta' d\omega_{\Theta'} \tag{2}$$

The meaning of the symbols is summarized in a table at the end of these notes (page 44).

1.1.2 The radiosity integral equation

In a purely diffuse environment, self-emitted radiance $L_e^{\rightarrow}(x)$ and the brdf $f_r(x)$ do not depend on directions Θ and Θ'. The rendering equation then becomes:

$$L^{\rightarrow}(x) = L_e^{\rightarrow}(x) + \int_{\Omega_x} f_r(x) L^{\leftarrow}(x, \Theta') \cos\theta' d\omega'_\Theta.$$

Of course, the incident radiance $L^{\leftarrow}(x, \Theta')$ still depends on incident direction. It corresponds to the exitant radiance $L^{\rightarrow}(y)$ emitted by the point y visible from x along the direction Θ'. The integral above, over the hemisphere Ω_x, can be transformed into an integral over all surfaces S in the scene, yielding an integral equation in which no directions appear:

$$L^{\rightarrow}(x) = L_e^{\rightarrow}(x) + \rho(x) \int_S G(x, y) L^{\rightarrow}(y) dA_y$$

or (multiplication with π on both sides):

$$B(x) = E(x) + \rho(x) \int_S G(x, y) B(y) dA_y. \tag{3}$$

The kernel of this integral equation is:

$$G(x, y) = \frac{\cos\theta_x \cos\theta_y}{\pi r_{xy}^2} \text{vis}(x, y). \tag{4}$$

Equation (1) now becomes:

$$B_i = \frac{1}{A_i} \int_{S_i} B(x) dA_x. \tag{5}$$

1.1.3 The radiosity system of linear equations

One method to solve integral equations like (3) is the Galerkin method [14, 32, 11, 63]. Basically, the left and right hand side of (3) are *projected* onto a set of *basis functions* and the resulting *coefficients* left and right are equated. With a constant basis function for each patch i ($\psi_i(x) = 1$

if $x \in S_i$ and 0 if $x \notin S_i$), approximating $B(x)$ by $B(x) \approx \tilde{B}(x) = \sum_i B'_i \psi_i(x)$, the Galerkin method yields the classical radiosity system of linear equations:

$$B'_i = E_i + \rho_i \sum_j F_{ij} B'_j. \tag{6}$$

The factors F_{ij} are called *patch-to-patch form factors*:

$$F_{ij} = \frac{1}{A_i} \int_{S_i} \int_{S_j} G(x, y) dA_y dA_x. \tag{7}$$

The coefficients B'_i that result after solving the system of linear equations (6) are only an *approximation* for the average radiosities (5) in the previous section. The difference is that discretisation errors get propagated, resulting in diffuse *reflections of* for instance light leaks. It is possible to construct scenes in which the difference is visible, but such cases are very rare in practice. We will denote both the average radiosity (5) and the radiosity coefficients in (6) by B_i in the remainder of this text.

1.2 The classical radiosity method

Solving the radiosity problem takes the following four steps:

1. Discretisation of the input geometry in patches i. For each resulting patch i, a radiosity value (per considered wavelength) B_i will be computed;

2. Computation of form factors F_{ij} (7), for every pair of patches i and j;

3. Numerical solution of the radiosity system of linear equations (6);

4. Display of the solution, including the transformation of the resulting radiosity values B_i (one for each patch and considered wavelength) to display colors. This involves tone mapping and gamma correction.

In practice, these steps are intertwined, for instance: form factors are only computed when they are needed, intermediate results are displayed already during system solution, in adaptive and hierarchical radiosity [10, 21], discretisation is performed during system solution, etc . . . .

1.3 Problems

At first sight, one would expect that step 3, radiosity system solution, would be the main problem of the radiosity method due to the size of the linear systems that need to be solved (one equation per patch, 100,000 patches is quite common). The radiosity system of linear equations is in practice however very well behaved, so that simple iterative methods such as Jacobi or Gauss-Seidel iterations converge after relatively few iterations.

The main problems of the radiosity method are related to the first two steps:

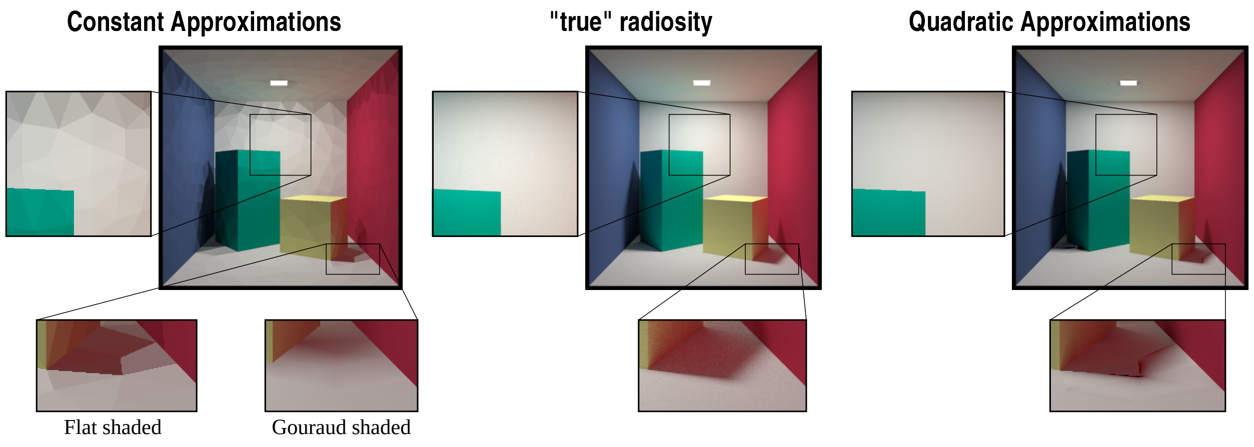


Figure 1: Meshing artifacts in radiosity with constant approximations (left) include undesired shading discontinuities along patch edges. Gouraud shading can be used to blur these discontinuities. Where ever the radiosity varies smoothly, a higher order approximation of radiosity on each patch results in a more accurate image on the same mesh (a quadratic approximation was used in the right column), but artifacts remain near discontinuities such as shadow boundaries. The middle column shows the “true” radiosity solution (computed with bidirectional path tracing).

1. Scene discretisation: the patches shall be small enough to capture illumination variations such as near shadow boundaries: the radiosity $B(x)$ across each patch needs to be approximately constant. Figure 1 shows what image artifacts may result from an improper discretisation. On the other hand, the number of patches shouldn’t be too high, because this would result in an exaggerated storage requirements and computation times;

2. Form factor computation: the number of form factors is not only huge (10,000,000,000 form factors for 100,000 patches), but each form factor in addition requires the solution of a non-trivial 4-dimensional integral (7). The integral will be singular for abutting patches, where the distance r_{xy} in the denominator of (4) vanishes. The integrand can also exhibit discontinuities of various degrees due to changing visibility (see figure 2).

Extensive research has been carried out in order to address these problems. Proposed solutions include custom algorithms form factor integration (hemi-cube algorithm, shaft culling ray tracing acceleration, . . .), discontinuity meshing, adaptive and hierarchical subdivision, clustering, form factor caching strategies the use of view importance and higher order radiosity approximations.

The techniques presented in these notes address the latter problem by *avoiding* form factor computation and storage completely. This results in more reliable algorithms (no problems with form factor computational error), that require less storage (no form factors need to be stored). In addition, the presented algorithms are more easy to implement and to use and they result in images of reasonable quality, showing multiple inter-reflection effects, sometimes much more rapidly than other radiosity algorithms.

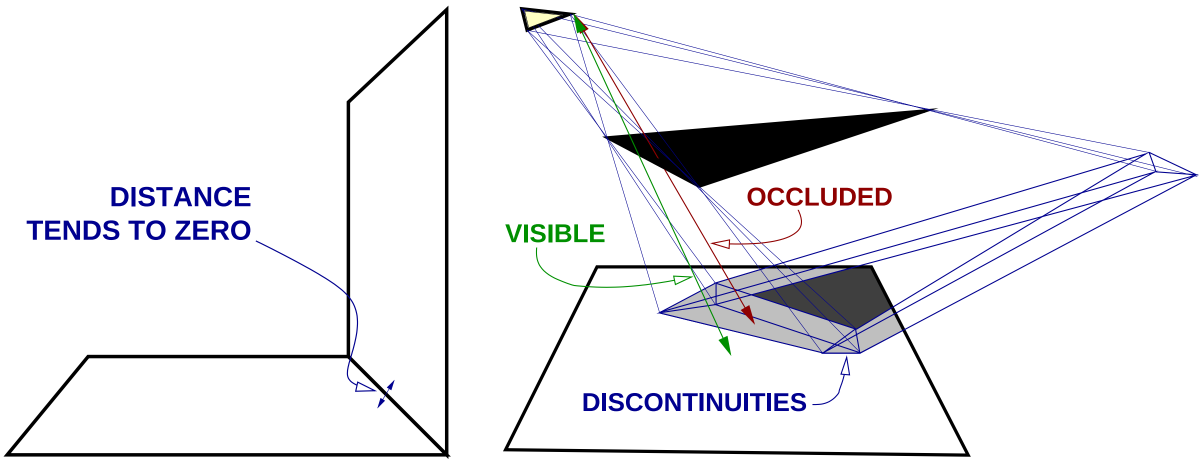


Figure 2: Form factor difficulties: the form factor integral, equations (7) and (4), contains the square distance between points in the denominator. This causes a singularity for abutting patches (left). Changing visibility introduces discontinuities of various degrees in the form factor integrand (right). Due to this problems, reliable form factor integration is a difficult task.

2 Monte Carlo Radiosity Basics

It is possible to avoid form factor computation and storage in radiosity because the form factors F_{ij} (7) for fixed i and variable j form a *probability distribution* that can be sampled efficiently. Using form factor sampling, Monte Carlo estimation of sums that occur in radiosity yields expressions in which the form factor appears in numerator and denominator, so it cancels and the numerical value of a form factor is never needed. In this section, an outline is given how sums can be estimated using the Monte Carlo method, and how form factor based sampling can be carried out.

2.1 Monte Carlo estimation of sums

2.1.1 Monte Carlo methods

Only a very brief outline is given here. An excellent introduction to Monte Carlo methods, can be found in [26].

The basic idea of Monte Carlo methods is to formulate a quantity to be computed as the *expected value* of a *random variable*. The mean of independent samples of the random variable yields an estimate for its expected value and thus for the quantity to be computed.

A random variable is a set of possible *outcomes*, say a_i, with associated probabilities p_i that indicate the chance that the outcome will show up in a random trial. The outcomes can be discrete events ("heads" or "tails", one of the six faces of a dice, a integer number in a given range), or continuous (for instance a point in a square, or a direction in the hemisphere above a point on a surface in 3D space). The probabilities need to be positive and they sum up to 1. In these notes, we will deal mainly with discrete random variables.

The expected value of a discrete random variable $\hat{A} = (a_i, p_i), i = 1 \ldots n$ (n is the number of potential outcomes), is defined as

$$E[\hat{A}] = \sum_{i=1}^{n} a_i p_i. \tag{8}$$

A second quantity related to random variables, the *variance*, plays in important role in the analysis of the performance of a Monte Carlo method. The variance is the mean square deviation of the outcomes from the expected value:

$$V[\hat{A}] = \sum_{i=1}^{n} \left(a_i - E[\hat{A}]\right)^2 p_i = \sum_{i=1}^{n} a_i^2 p_i - E[\hat{A}]^2. \tag{9}$$

2.1.2 Monte Carlo summation

We will now apply the above definitions in order to estimate a finite sum[1]

$$S = \sum_{i=1}^{n} a_i.$$

[1]Monte Carlo summation is the same as Monte Carlo integration, but with a discrete random variable rather than a continuous one.

Suppose that N times a term $a_{i_s}, s = 1, \ldots N$ is randomly and independently picked from the sum, with the term a_i having a probability p_i of being picked. The average ratio of the value a_{i_s} of the picked terms, over their probability p_{i_s}, then yields an estimate for the sum S:

$$\frac{1}{N} \sum_{s=1}^{N} \frac{a_{i_s}}{p_{i_s}} \approx S. \tag{10}$$

Indeed, the random variable that corresponds with this procedure is $\hat{S} = (a_i/p_i, p_i)$, with outcomes a_i/p_i and associated probabilities p_i, $i = 1 \ldots n$. The expected value of $\hat{S}$ is:

$$E[\hat{S}] = \sum_{i=1}^{n} \frac{a_i}{p_i} p_i = S.$$

This implies that estimating S by a single sample works, but it is easy to show that in that case, also the mean of N independent trials will yield a correct estimate [26]. The variance

$$V[\hat{S}] = \sum_{i=1}^{n} \frac{a_i^2}{p_i} - S^2 \tag{11}$$

indicates how effective such Monte Carlo summation will be. It can be shown that an estimate with N trials will be off by less than one *standard error* $\sqrt{V[\hat{S}]/N}$ 68.3% of the time. It will be off by less than twice the standard error 95.4% of the time. The probability that the estimate is off by less than three times the standard error is 99.7%. If the variance $V[\hat{S}]$ is large, more samples (larger N) will be required in order to obtain estimates which are within a fixed threshold from the true sum with given confidence.

Ideally, the variance should be zero, in which case a single random trial already yields the correct result. The *estimator* $\hat{S}$ for S is then called *perfect*. Unfortunately, perfect estimation is not possible if the quantity to be estimated is not already known. In the case of summation, perfect estimation would result if p_i is chosen proportional to a_i and all a_i are of the same sign. The probabilities p_i however need to be normalized: $\sum_{i=1}^{n} p_i = 1$. Normalization implies that they would have to equal $p_i = a_i/S$, but S is not known in advance! Note that with these perfect probabilities, $a_i/p_i = S$ always. Any random trial would yield the correct result indeed.

The variance formula above indicates that good estimation will result already if p_i is chosen approximately proportional to a_i. On the other hand, care must be taken that none of the terms gets a too low probability, yielding large ratios a_i^2/p_i. In particular p_i shall never be zero if a_i isn't: the estimates would no longer converge to the correct solution (they would be *biased*) and the variance would be infinitely large.

2.1.3 Discussion

Monte Carlo methods are widely applicable and simple. It is sometimes said that they are a way of solving complicated mathematical problems without complicated math. Indeed, all one needs to do is 1) design a random variable with appropriate mean and low variance, 2) take random samples from it, and 3) average the results[2]. Disregarding rounding errors and other sources of inaccurate arithmetic, the more samples one takes, the closer the estimate will be to the correct result. In the case of image synthesis, the error manifests itself in the form of noisy artifacts. If an image computed with a correctly implemented unbiased Monte Carlo algorithm exhibits no noise, it will be correct.

The main disadvantage of Monte Carlo methods is their slow, $\mathcal{O}(1/\sqrt{N})$ convergence: in order to reduce the standard error by a factor of 10, 100 times more samples are needed. For this reason, a lot of research has been carried out (and still is being done) in order to transform a given estimator for some problem into one with lower variance (*variance reduction* techniques), or to take samples according to non-random but more uniform patterns (*low-discrepancy sampling*), see §5.

Considering that computers are very good at adding numbers, Monte Carlo summation is in general not recommended. The situation however changes for sums with a large number of terms, which are not simple numbers, but which are the result of some complicated calculation. With an appropriate probability distribution p_i, it may happen that only a small set of all terms in the sum needs to be evaluated in order to obtain an estimate for the sum that is of sufficient accuracy. The impact of the terms which are not sampled is taken into account in the probability distribution, "by the fact that they *could* have been sampled". We shall see that this is the case in radiosity.

2.2 Form factor sampling

In radiosity, we will need to evaluate sums with as many terms as there are patches in the scene. Each term contains a form factor F_{ij}, which is given by a non-trivial 4-dimensional integral (7).

The form factors F_{ij} for fixed i and variable j form a probability distribution because they are all positive or zero, and they sum up to 1 in a closed environment: $\sum_{j=1}^{n} F_{ij} = 1$. This probability distribution can be sampled by means of *local* or *global uniformly distributed lines*. In both cases, random lines (rays) are traced through the scene in such a way that the probability of obtaining a ray from a fixed patch i to any other patch j in the scene corresponds to the form factor F_{ij} (see figure 3). In other words, given that a uniformly distributed local or global line pierces the patch i, it will have its next intersection with a surface in the scene on any other patch j with probability equal to F_{ij}. Such lines will be used to sample *transitions* from a patch i to a random other patch j with probability of selecting j being equal to F_{ij}.

2.2.1 Local line sampling

The first way to sample according to the form factors is to select

1. a uniformly chosen random ray origin x on the surface S_i of the first patch i;

2. a cosine-distributed ray direction Θ w.r.t. the surface normal at x.

[2]Quite often however, there *is* complicated math involved in the design of an appropriate estimator or in the sampling algorithm!

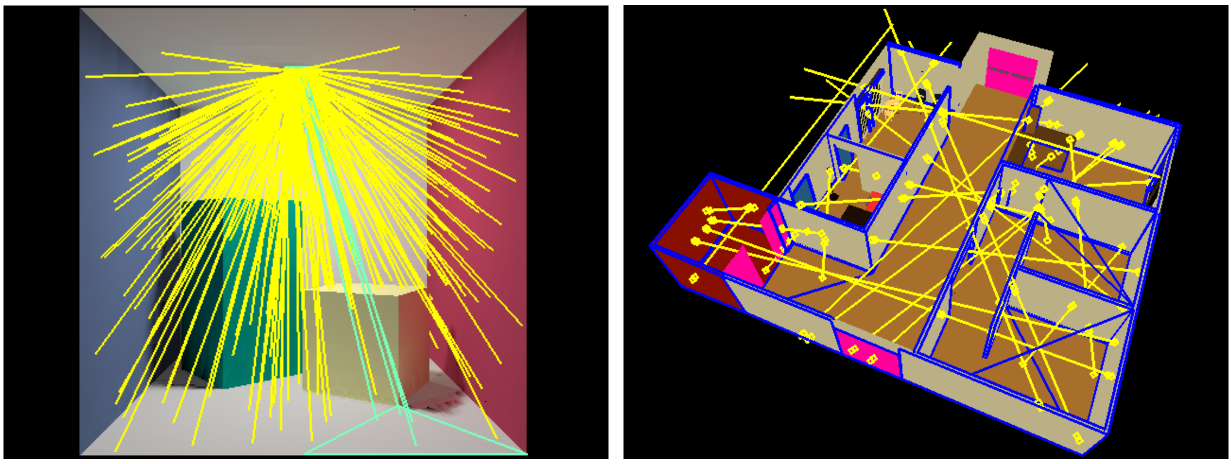


Figure 3: Local uniformly distributed lines (left) are constructed by explicitly sampling the origin on a patch in the scene. Global lines (right) are constructed without reference to any of the patches in the scene. Their intersection points with the surfaces in the scene are however also uniformly distributed. The angle between these lines and the normal on each intersected surface is cosine distributed, just like with local lines. The intersection points define spans on each line. Each global line span can be used bidirectionally for form factor computation between the connected patches.

The probability that the first hit point $y = h(x, \Theta)$ of this ray with a surface of the scene lays on a patch j is given by F_{ij}[3]. This sampling scheme has been proposed at the end of the '80-ies as a ray-tracing alternative for the hemi-cube algorithm for form factor computation [62, 57].

2.2.2 Global line sampling

There also exist techniques to construct uniformly distributed lines without explicitly sampling the origin on a patch in the scene. Uniformly distributed lines constructed without explicit sampling the origin on a patch, are called *global uniformly distributed lines*. The construction and properties of such lines have been studied extensively in *integral geometry* [48, 49, 50].

In the context of radiosity, the following algorithms have been used:

- **Two-points-on-a-sphere** method: two uniformly distributed points p and q are sampled on the surface of a sphere bounding the scene. The line connecting p and q can be shown to be a uniformly distributed line within the scene [49]. A field of N uniformly distributed lines is obtained by sampling N pairs of points x_k and y_k, $k = 1, \ldots, N$, on the surface of the bounding sphere;

- **Plane-intercept** method [46, 37, 50, 70]: a uniformly distributed point Θ is sampled on the unit sphere. As such, Θ is a uniform global direction. Consider the plane P through the origin and perpendicular to Θ: the plane equation is $\Theta \cdot x = 0$. Now consider the orthogonal

[3]Exercise: proof this. Hint: calculate the probability p_{ij} that $h(x, \Theta) \in S_j$

projection of the scene onto this plane. Each uniformly sampled point x in the projection defines, together with Θ, a uniformly distributed line through the scene.

The resulting lines cross several surfaces in the scene. The intersection points with the intersected surfaces define *spans* of mutually visible patches along the line (see figure 3). Each such a line span corresponds to *two* local cosine-distributed lines — one in both directions along the line — because the global uniformly distributed lines are uniformly distributed w.r.t. every patch in the scene. This is unlike local lines, which are uniformly distributed only w.r.t. the patch on which the origin was sampled.

It can be shown that the probability that a global uniform line, generated with the aforementioned algorithms, intersects a given patch i, is proportional to the surface area A_i [50]. If N global lines are generated, the number N_i of lines crossing a patch i will be

$$N_i \approx N \frac{A_i}{A_T}. \tag{12}$$

It can also be shown that, if N_{ij} is the number of lines that have subsequent intersections with the surfaces in the scene on patch i and on patch j, then

$$\frac{N_{ij}}{N_i} \approx F_{ij}.$$

2.2.3 Local versus global line sampling

The main advantage of global lines over local lines is that geometric scene coherence can be exploited in order to generate global lines more efficiently:

- In a naive ray-tracing implementation for instance, the two-points-on-a-sphere method would yield all k intersections of a line with the surfaces in the scene at the same cost of determining only the nearest intersection of a local line. In a properly constructed scene, the number of line spans on a global line is half the number k of intersection points. Since each span is used bidirectionally, this means that the global line yields the equivalent of k local lines at the same cost. Even when using ray-tracing acceleration techniques that allow to stop tracing a local line before all its potential intersections with the scene are determined, there still is a speed-up.

- The plane-intercept method allows bundles of parallel global lines to be generated using a Z-buffer like algorithm: first, a uniform random direction Θ is chosen. Next, a rectangular window is chosen in the plane, through the origin and perpendicular to Θ, that contains the orthogonal projection of the whole scene on the plane. A certain resolution for rendering is chosen in the window. Each pixel will correspond to a parallel global line. Finally, a suitable orthogonal projection matrix is set up and the scene projected onto the plane using a Z-buffer-like algorithm. Instead of keeping only the nearest Z-value in each pixel however, a full sorted list of all patches visible through each pixel is kept [37]. Alternatively, it is possible to use sweep-plane algorithms to solve the visibility problem analytically [46]. This corresponds to a bundle of parallel lines with infinite density [69].

The main limitation of global lines w.r.t. local lines is that their construction cannot easily be adapted in order to increase or decrease the line density on a given patch. In particular, when used for form factor calculation, it can be shown that the form factor variance is approximately inverse proportional to the area A_i of the source patch i. The variance will be high on small patches.

3 Stochastic Relaxation Radiosity

The radiosity system of linear equations (6) is usually solved using an iterative solution method such as Jacobi, Gauss-Seidel or Southwell iterations. Each iteration of such a relaxation method consists of sums: dot products of a row of the form factor matrix with the radiosity or power vector. When these sums are estimated using a Monte Carlo method, as explained in the previous section, a stochastic relaxation method results. In this section, we explore stochastic relaxation methods for radiosity, based on form factor sampling. Not only is form factor computation and storage avoided in stochastic relaxation methods, but also their time complexity is much lower: roughly log-linear in the number of patches rather than quadratic.

3.1 The Jacobi iterative method for radiosity

3.1.1 Regular gathering of radiosity

The Jacobi iterative method for radiosity constructs a sequence of approximations $B_i^{(k)}$ for the solution of the radiosity system of equations (6). As the first approximation $B_i^{(0)} = E_i$, self-emitted radiosity can be taken. A next approximation $B_i^{(k+1)}$ is then obtained by filling in the previous approximation $B^{(k)}$ in the right hand side of (6):

$$\begin{aligned} B_i^{(0)} &= E_i \\ B_i^{(k+1)} &= E_i + \rho_i \sum_j F_{ij} B_j^{(k)} \end{aligned} \tag{13}$$

A hemi-cube algorithm for instance, allows to compute all form factors F_{ij} for fixed patch i simultaneously. Doing so, iteration steps according to the above scheme can be interpreted as *gathering* steps: in each step, the previous radiosity approximations $B_j^{(k)}$ for all patches j are "gathered" in order to obtain a new approximation for the radiosity $B^{(k+1)}$ at i.

3.1.2 Regular shooting of power

A shooting variant of the above iteration algorithm can be obtained by multiplying the left and right hand side of (6) by the area A_i, yielding the so called *power* system of linear equations:

$$P_i = \Phi_i + \sum_j P_j F_{ji} \rho_i. \tag{14}$$

The Jacobi iterative method can also be applied to this system of equations, yielding the following iteration scheme:

$$\begin{aligned} P_i^{(0)} &= \Phi_i \\ P_i^{(k+1)} &= \Phi_i + \sum_j P_j^{(k)} F_{ji} \rho_i. \end{aligned} \tag{15}$$

3.1.3 Incremental shooting of power

Each regular power-shooting iteration above *replaces* the previous approximation of power $P^{(k)}$ by a new approximation $P^{(k+1)}$. Similar to in progressive refinement radiosity [8], it is possible to construct iterations in which *unshot* power is propagated rather than total power. An approximation for the total power is then obtained as the sum of *increments* $\Delta P^{(k)}$ computed in each iteration step:

$$\begin{aligned}\Delta P_i^{(0)} &= \Phi_i \\ \Delta P_i^{(k+1)} &= \sum_j \Delta P_j^{(k)} F_{ji} \rho_i \\ P_i^{(k)} &= \sum_{l=0}^{k} \Delta P_i^{(l)}\end{aligned}$$

3.1.4 Discussion

With deterministic summation, there is no difference between the results after complete iterations with the above three iteration schemes. We will see below however, that they lead to quite different algorithms when the sums are estimated stochastically.

The computation cost of each deterministic iteration is quadratic in the number of patches.

3.2 Stochastic Jacobi radiosity

3.2.1 Stochastic incremental shooting of power

Consider the incremental power shooting iterations above. The sum $\sum_j \Delta P_j^{(k)} F_{ji} \rho_i$ can also be written as a double sum, by introducing Kronecker's delta function $\delta_{li} = 1$ if $l = i$ and 0 if $l \neq i$:

$$\Delta P_i^{(k+1)} = \sum_{j,l} \Delta P_j^{(k)} F_{jl} \rho_l \delta_{li}.$$

The double sum can be estimated using a Monte Carlo method as explained in §2.1:

1. Pick terms (pairs of patches) (j, l) in either of the following ways:

   (a) By local line sampling:

   - Select a "source" patch j with probability p_j proportional to its unshot power:

     $$p_j = \Delta P_j^{(k)} / \Delta P_T^{(k)} \quad \text{with:} \quad \Delta P_T^{(k)} = \sum_j \Delta P_j^{(k)}$$

   - Select a "destination" patch l with conditional probability $p_{l|j} = F_{jl}$ by tracing a local line as explained in §2.2.1.

   The combined probability of picking a pair of patches (j, l) is

   $$p_{jl} = p_j p_{l|j} = \Delta P_j^{(k)} F_{jl} / \Delta P_T^{(k)}.$$

(b) By global line sampling (transillumination method [37, 68]): the intersections of each global line (§2.2.2) with the surfaces in the scene define spans of mutually visible pairs of points along the line. Each such pair corresponds to a term (j, l) in the sum. The associated probability is:

$$p_{jl} = A_j F_{jl} / A_T.$$

2. Each picked term yields a score equal to the value of that term divided by its probability p_{jl}. The average score is an unbiased estimate for $\Delta P_i^{(k+1)}$. Estimation with N local lines for instance, yields:

$$\frac{1}{N} \sum_{s=1}^{N} \frac{\Delta P_{j_s}^{(k)} F_{j_s,l_s} \rho_{l_s} \delta_{l_s,i}}{\Delta P_{j_s}^{(k)} F_{j_s,l_s} / \Delta P_T^{(k)}} = \rho_i \Delta P_T^{(k)} \frac{N_i}{N} \approx \Delta P_i^{(k+1)}$$

$N_i = \sum_{s=1}^{N} \delta_{l_s,i}$ is the number of local lines that land on i.

The procedure above can be used to estimate $\Delta P_i^{(k+1)}$ for all patches i simultaneously. The same samples (rays) can be used. The difference is only in the scores, which basically require to count the number of rays hitting each patch. With stratified local line sampling, algorithm 1 results.

3.2.2 Discussion

The most expensive operation in the algorithm above is ray shooting. The number of rays that needs to be shot in order to compute the radiosities in the scene to given accuracy with given confidence is determined by the variance of the involved estimators. We discuss here the case of local line sampling.

Variance of a single iteration The variance of the above sketched Monte Carlo method for estimating $\Delta P_i^{(k+1)}$ is straightforward to compute according to (11) [1]:

$$V[\Delta \hat{P}_i^{(k+1)}] = \rho_i \Delta P_T^{(k)} \Delta P_i^{(k+1)} - \left(\Delta P_i^{(k+1)} \right)^2. \tag{16}$$

The latter term is usually negligible compared to the former ($\Delta P_i^{(k+1)} \ll \Delta P_T^{(k)}$).

Variance of a sequence of iterations until convergence The solution P_i is eventually obtained as a sum of increments $\Delta P_i^{(k)}$ computed in each iteration step. The variance on each increment $\Delta P_i^{(k)}$ is given above. Assuming that subsequent iterations are independent (which is to good approximation true in practice), and that N_k independent samples are used in the k-th iteration, the variance on the result of K iterations will be

$$V[\hat{P}_i] = \sum_{k=1}^{K} \frac{1}{N_k} V[\Delta \hat{P}_i^{(k)}].$$

Algorithm 1 Incremental stochastic Jacobi iterative method.

```
1. Initialize total power $P_i \leftarrow \Phi_i$, unshot power $\Delta P_i \leftarrow \Phi_i$, received power $\delta P_i \leftarrow 0$ for all
   patches $i$ and compute total unshot power $\Delta P_T = \sum_i \Delta P_i$;
2. Until $||\Delta P_i|| \leq \varepsilon$ or number of steps exceeds maximum, do
   (a) Choose number of samples $N$;
   (b) Generate a random number $\xi \in (0,1)$;
   (c) Initialize $N_{prev} \leftarrow 0$; $q \leftarrow 0$;
   (d) Iterate over all patches $i$, for each $i$, do
       i. $q_i \leftarrow \Delta P_i / \Delta P_T$;
       ii. $q \leftarrow q + q_i$;
       iii. $N_i \leftarrow \lfloor Nq + \xi \rfloor - N_{prev}$;
       iv. Do $N_i$ times,
           A. Sample random point $x$ on $S_i$;
           B. Sample cosine-distributed direction $\Theta$ at $x$;
           C. Determine patch $j$ containing the nearest intersection point of the ray originat-
              ing at $x$ and with direction $\Theta$, with the surfaces of the scene;
           D. Increment $\delta P_j \leftarrow \delta P_j + \frac{1}{N}\rho_j \Delta P_T$.
       v. $N_{prev} \leftarrow N_{prev} + N_i$.
   (e) Iterate over all patches $i$, increment total power $P_i \leftarrow P_i + \delta P_i$, replace unshot power
       $\Delta P_i \leftarrow \delta P_i$ and clear received power $\delta P_i \leftarrow 0$. Compute new total unshot power $\Delta P_T$
       on the fly.
   (f) Display image using $P_i$.
```

Optimal allocation of $N = \sum_{k=1}^{K} N_k$ samples over the individual iterations is obtained if $1/N_k$ is inverse proportional to $V[\Delta \hat{P}_i^{(k)}]$. For all patches i, $V[\Delta \hat{P}_i^{(k)}]$ (16) is approximately proportional to $P_T^{(k-1)}$, suggesting to choose the number of samples in the k-th iteration proportional to the total unshot power $\Delta P_T^{(k-1)}$ to be propagated in that iteration:

$$N_k \approx N \frac{\Delta P_T^{(k-1)}}{P_T}.$$

When N_k drops below a small threshold, convergence has been reached. Combining all above results, it can be shown that the variance on the radiosity B_i after convergence is *to good approximation* given by [1]:

$$V[\hat{B}_i] \approx \frac{P_T}{N} \frac{\rho_i (B_i - E_i)}{A_i} \qquad (17)$$

Time complexity In order to compute all radiosities B_i to prescribed accuracy ε with 99.7% confidence, the number of samples N shall be chosen so that

$$3\sqrt{\frac{V[\hat{B}_i]}{N}} \leq \varepsilon$$

for all i. Filling in (17) then yields:

$$N \geq \frac{9P_T}{\varepsilon^2} \cdot \max_i \frac{\rho_i(B_i - E_i)}{A_i}. \tag{18}$$

This formula allows us to examine how the number of rays to be shot must be increased as a scene to be rendered is “made larger”. There are however many possible scenarios how a scene can be “made larger”. For instance new objects can be added, or one can switch to a finer tessellation of the surfaces in the scene without adding new objects. If all patches in a scene are split in two, the required number of rays in order to obtain a given accuracy will need to be doubled as dividing the patches (asymptotically) has no effect on reflectivities and radiosities. The cost of shooting a ray is often assumed to be logarithmic in the number of polygons. Although the truth thus is much more complicated, it is often stated that Monte Carlo radiosity algorithms have log-linear complexity. In any case, their complexity is much lower than quadratic. This result is not only valid for incremental stochastic shooting of power, but also for other Monte Carlo radiosity algorithms based on shooting [59, 51, 1].

3.2.3 Stochastic regular shooting of power

The sums in regular power shooting iterations (15) can be estimated using a very similar Monte Carlo method as described above for incremental power shooting. The first stochastic Jacobi radiosity algorithms, proposed by L. and A. Neumann et al. [38], consisted entirely of such iterations. Unlike in its deterministic counterpart, the resulting radiosity solutions of each iteration are averaged, rather than having the result of a new iteration replace the previous solution. The main disadvantage of using only regular iterations is that higher order inter-reflections appeared in the result only at a slow pace, especially in bright environments. This problem has been called the *warming up* or *burn in* problem [38, 37, 40, 39]

The warming up problem can be avoided by first performing a sequence of incremental power shooting iterations until convergence, as explained above. This results in a first *complete* radiosity solution, including higher order inter-reflections. Especially when the number of samples N is rather low, this first complete solution will exhibit noisy artifacts. Stochastic regular power shooting iterations can then be used in order to reduce these artifacts. A regular power shooting iteration can be viewed as a transformation, transforming a first complete radiosity solution into a new complete one. It can be shown that the output is largely independent of the input. The average of the two radiosity distributions obtained subsequently is to good approximation the same as the result of one iteration with twice the number of samples.

3.2.4 Stochastic regular gathering of radiosity

Also regular radiosity gathering iterations (13) can be converted into a stochastic variant using the procedure outlined above. The main difference with power shooting iterations is that now, a new radiosity estimate is obtained as the average score associated with rays that are *shot from* each patch i, rather than from rays that land on i.

The variance of regular gathering is in practice most often higher than that of shooting, but it does not depend on the patch area. Gathering can therefore be useful in order to “clean” noisy artifacts from small patches, which have a small chance of being hit by shooting rays from elsewhere and therefore can suffer from a large variance with shooting.

3.3 Other stochastic relaxation methods for radiosity

It is possible to design stochastic adaptations of other relaxation methods in the same spirit. Shirley has investigated algorithms that can be viewed as stochastic incremental Gauss-Seidel and Southwell algorithms [57, 59, 58]. Bekaert has studied stochastic adaptations of over-relaxation, Chebyshevs iterative method, and the conjugate gradient method (suggested by L. Neumann). These relaxation methods have been developed in hope of reducing the number of iterations to convergence. Since the deterministic iterations have a fixed computation cost, strongly related with the size of a linear system, reducing the number of iterations clearly reduces the total computation cost to convergence. This is however not so with the stochastic variants. The computation cost of stochastic relaxation methods is dominated by the number of samples to be taken. The number of samples is only loosely related with the size of the system. In the radiosity case, it turns out that the simple stochastic Jacobi iterations described above is at least as good as other stochastic relaxation methods. Figure 4 illustrates our claim that stochastic relaxation can yield useful images much faster than corresponding deterministic relaxation algorithms.

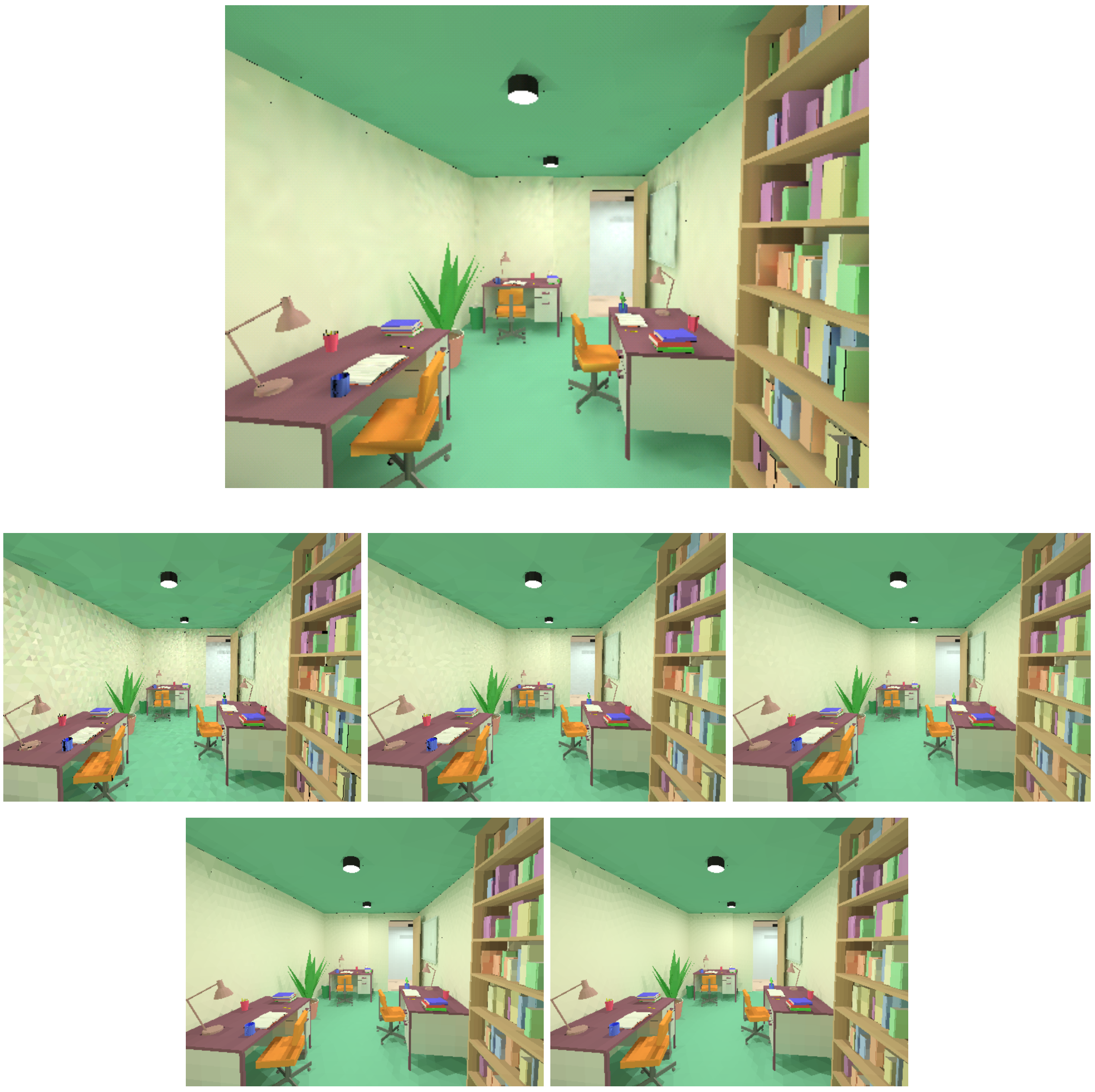


Figure 4: Stochastic relaxation methods can yield useful images much faster than their deterministic counterparts. The shown environment consists of slightly more than 30,000 patches. The top image was obtained with incremental stochastic power shooting iterations in less than 2 minutes on a 195MHz R10000 SGI Octane system, using about 10^6 rays. Even if only 1 ray were used for each form factor, $9 \cdot 10^8$ rays would be required with a deterministic method. Noisy artifacts are still visible, but are progressively reduced using regular stochastic power shooting iterations. After about 30 minutes, they are not visible anymore.
This progressive variance reduction is illustrated in the bottom images, shown without Gouraud shading to make noisy artifacts better visible. The shown images have been obtained after 1, 4, 16, 64 and 252 (right-to-left, top-to-bottom) iterations of less than 2 minutes each.
The model shown is an edited part of the Soda Hall VRML model made available at the University of California at Berkeley.

4 Random Walk Radiosity

Unlike stochastic relaxation methods, random walk methods for linear systems are well covered in Monte Carlo literature [20, 66, 19, 16, 47]. Their application to the radiosity system of equations (6) and equivalent power system (14) has been proposed by Sbert [50, 51]. They are called *discrete* random walk methods, because they operate on a discrete *state space*. In this section, we will show how they differ from more familiar *continuous* random walk methods for second kind Fredholm integral equations. There are many different kinds of random walks besides the familiar *collision* random walk which contributes a score whenever it hits a surface. We will compare them with each other and with the stochastic Jacobi radiosity methods of the previous section.

4.1 Random walks in a continuous state space

4.1.1 Particle transport simulations and integral equations

Light transport is an instance of a wider class of linear particle transport problems, that can be solved as follows [26]:

1. Fix a description of the *state* X of a particle. In many applications, including illumination computation, particles are sufficiently characterized by their position x, direction Θ, energy E[4] and the time t;

2. Fix a description of the particle sources by means of a normalized *source (or birth) density distribution* $S(X)$ and a constant S_T expressing the total emission intensity. With $X = (x, \Theta, E, t)$, $S(X)$ expresses the relative intensity of emission of particles with energy E at time t from position x and into direction Θ;

3. Fix a description of how particles interact with the medium and surfaces in which they travel. Particles can be scattered or absorbed. If the scattering and absorption of the particles only depends on their present state, and not on their past history, particle scattering and absorption is fully determined by a *transition density function* $T(X \to X')$ from each state X to each other state X'. The transition density function need not to be normalized:

$$\rho(X) = \int_{\Omega} T(X \to X')dX'$$

   describes the average number of particles resulting when a particle scatters at X. Here, Ω denotes the full state space of the particles. $\rho(X)$ can be larger than 1, e.g. in nuclear reactions, in which case the medium is called a multiplying or super-critical medium. If $\rho(X) \leq 1$, $\alpha(X) = 1 - \rho(X)$ expresses the intensity of absorption at X. In case $\alpha(X) > 0$, the medium is called absorbing or sub-critical;

4. Particle paths are simulated by sampling emission events according to the source density function $S(X)$ and subsequently sampling scattering events according to $T(X \to X')$ until

[4]for photons: $E = \hbar c/\lambda$ with $\hbar$ Planck's constant, c the velocity of light and λ the wavelength of the photon.

the particle is either absorbed or disappears from the region of interest (assuming that the particle will not re-enter the region of interest). While simulating particle paths, events of interest are counted.

Such a simulation is suited for computing weighted integrals of the particle density function $\chi(X)$:

$$G = \int_\Omega g(X)\chi(X)dX \tag{19}$$

The *response* or *detector* function $g(X)$ expresses our interest in particles with given location, direction, energy and time X. For instance by taking $g(X) = 1$ for particles located on a given surface and 0 on other surfaces, the particle flux on the surface will be estimated.

The particle density $\chi(X)$ is the differential particle flux at given location, direction and energy at a fixed time. It is the sum of the source density $S(X)$ and the density of particles that are scattered into X from elsewhere:

$$\chi(X) = S(X) + \int_\Omega \chi(X')T(X' \to X)dX'. \tag{20}$$

In short: *simulation of particle paths with given source density $S(X)$ and transition density $T(X' \to X)$ is a technique to sample points X in state space Ω with (non-normalized) density $\chi(X)$ that is the solution of the integral equation (20).*

4.1.2 Continuous random walks for radiosity

The general rendering equation (2) and the radiosity integral equation (3) are of the same form as (20). In the case of a purely diffuse environment, equation (3), we proceed as follows:

- Only the location of the particle is of interest: $X = x$;
- The source density $S(X)$ corresponds to the normalized self-emitted radiosity $E(x)/\Phi_T$, with Φ_T the total self-emitted power in the scene;
- The transition density $T(X' \to X)$ corresponds to $G(y,x)\rho(x)$. It can be sampled by shooting a ray in a cosine distributed direction w.r.t. the surface normal at y. Next a survival/absorption test is carried out at the new location x, taking the probability of survival equal to the reflectivity $\rho(x)$;
- The particle density $\chi(X)$ corresponds with the radiosity: $\chi(x) = B(x)/\Phi_T$.

Such particle simulation can be used in order to estimate integrals containing the radiosity function $B(x)$. With the general rendering equation (2), particle transport is simulated with non-diffuse emission and scattering. The resulting particle density is again proportional to the radiosity if only location is taken into account. It will be proportional to the exitant radiance $L^{\rightarrow}(x,\Theta)$ when we take into account both location and direction. The basic idea is however the always the same. We discuss a number of applications for the purely diffuse case below. Particle transport simulation faithfull to the laws of physics is called *analog* simulation.

4.1.3 The histogram method

The average radiosity on a patch i is given by an integral equation of $B(x)$:

$$B_i = \frac{1}{A_i} \int_{S_i} B(x) dA_x.$$

Random walk constructed as outlined above are a technique to sample points x with density $\chi(x) = B(x)/\Phi_T$. With N random walks, B_i can be estimated as

$$\frac{\Phi_T N_i}{A_i N} \approx B_i$$

where N_i is the number of visits to the patch i. The histogram method for radiosity computations has been demonstrated by Heckbert [22], Pattanaik [44] and others later on.

4.1.4 Basis function methods

Bouatouch et al. [5] and Feda [17] have used such random walks in order to estimate higher order (linear, quadratic, ...) approximations of radiosity on patches:

$$\tilde{B}_i(x) = \sum_\alpha B_{i,\alpha} \psi_{i,\alpha}(x).$$

The functions $\psi_{i,\alpha}(x)$ are called basis functions. The sum above is over all basis functions defined on patch i. A constant approximation is obtained when using just one basis functions $\psi_i(x)$ per patch, which is 1 on the patch and 0 outside (see §1.1.3). The coefficients $B_{i,\alpha}$ can be obtained as scalar products with so called *dual* basis functions $\tilde{\psi}_{i,\alpha}$:

$$B_{i,\alpha} = \int_S B(x) \tilde{\psi}_{i,\alpha}(x) dA_x.$$

With N random walks, these integrals can be estimated as

$$\frac{\Phi_T}{N} \sum_s \tilde{\psi}_{i,\alpha}(x_s) \approx B_{i,\alpha}.$$

The sum is over all points x_s visited by the random walks. The dual basis function $\tilde{\psi}_{i,\alpha}$ is the unique linear combination of the original basis functions $\psi_{i,\beta}$ that fulfills the relations

$$\int_{S_i} \tilde{\psi}_{i,\alpha}(x) \psi_{i,\beta}(x) dA_x = \delta_{\alpha,\beta}.$$

In the case of a constant approximation, the dual basis function is $\tilde{\psi}_i(x) = 1/A_i$ if $x \in S_i$ and 0 elsewhere. This results in the same expression as with the histogram method.

4.1.5 Kernel methods

The radiosity $B(y)$ at a point y could also be written as an integral involving a so called Dirac pulse function:

$$B(y) = \int_S B(x)\delta(x-y)dA_x.$$

Estimating the latter integral with random walks wouldn't work, because the Dirac pulse function is zero everywhere, except when its argument is zero. The chance of finding a particle hitting exactly the point y is zero in theory[5]. Even if we would find a particle hitting exactly at y, the value of the Dirac pulse is not determinate. It can't be finite, because the Dirac function is zero everywhere except at one point and its integral is equal to 1. An approximation for the radiosity at y can however be obtained by using a different, normalized *density kernel* function $K(x-y)$ centered around y:

$$\tilde{B}(y) = \int_S B(x)K(x-y)dA_x \approx B(y)$$

This integral can be estimated using N random walks as:

$$\frac{\Phi_T}{N}\sum_s K(x_s - y) \approx \tilde{B}(y)$$

The sum is again over all points x_s visited by the random walks, and can be viewed alternatively as a sum of the value at the query point y of mirrored kernels $\tilde{K}(y-x_s) = K(x_s-y)$ centered around the hit points x_s. This form of *density estimation* has been used by Chen [6], Collins [12] and Shirley et al. [60, 74]. Also the photon map algorithm by Jensen et al. [24] is based on a similar principle. Density estimation and the photon map are discussed in detail in other courses at this conference.

4.1.6 Final gathering using dependent tests

Final gathering is a well known view-dependent technique to compute very high-quality images based on a rough radiosity solution. Basically, it re-evaluates the radiosity emitted by points x_p visible through every pixel in an image, by filling in the approximate pre-computed view-independent radiosity solution $\tilde{B}(y)$ in the right hand side of the radiosity integral equation:

$$B^{\text{disp}}(x_p) = E(x_p) + \int_S \tilde{B}(y)G(y,x_p)\rho(x_p)dA_y.$$

Consider the equation:

$$B^{\text{disp}}(x_p) = E(x_p) + \int_S B(y)G(y,x_p)\rho(x_p)dA_y.$$

[5]In practice, the chance is not zero because of finite precision arithmetic.

This is also an integral containing the radiosity function $B(y)$, and can thus be estimated using random walks. Estimates for $B^{\text{disp}}(x_p)$ can be obtained using the same set of random walk hit points y_s for all x_p:

$$E(x_p) + \rho(x_p)\frac{\Phi_T}{N}\sum_s G(y_s, x_p) \approx B^{\text{disp}}(x_p).$$

Evaluating the kernel values $G(y_s, x_p)$ for a fixed random walk hit point y_s, requires visibility testing similar to in algorithms for computing shadows in an image due to a point light source [29].

4.1.7 Collision estimation

In global illumination algorithms, the random walks are used in a slightly more efficient way than explained above:

- Particles at the light source are not counted, because they estimate the self-emitted light distribution which is known. We call this *source term estimation suppression*;
- When sampling transitions, first an absorption/survival test is carried out. Only if the particle survives, it is propagated to another surface, by tracing a ray;
- Particles are counted not only when they survive a collision with a surface (survival estimation), but also when they are absorbed (collision estimation). This is compensated for by multiplying the estimates above with the probability of survival, i.o.w. with the reflectivity ρ_i.

4.2 Random walks in a discrete state space

4.2.1 Discrete random walks and linear systems

The states in which a particle can be found do not need to form a continuous set. They can also form a discrete set. For instance, the states can be "the particle is on patch i", with one such state per patch.

Just like the particle density resulting from a random walk with continuous source and transition density is the solution of a second-kind Fredholm integral equation, the (discrete) particle density χ_i resulting from a discrete random walk with source density π_i and transition density p_{ij} (for particles going from i to j), is the solution of a system of linear equations[6]:

$$\chi_i = \pi_i + \sum_j \chi_j p_{ji}. \tag{21}$$

This density can be used in order to estimate scalar products

$$G =< g, \chi >= \sum_i g_i \chi_i. \tag{22}$$

For instance, by choosing $g_i = \delta_{ik}$, the k-th component χ_k of the solution of (21) is computed.

[6]Note the switch of indices p_{ji} instead of p_{ij}!

4.2.2 Discrete shooting random walks for radiosity

Using local or global uniformly distributed lines (§2.2), we are able to simulate particle transitions from a patch i to patch j according to the form factor F_{ij}.

Due to the order in which the indices of the form factors appear in the equations, discrete random walk simulation as outlined above is therefore suited to solve the power system (14)

$$P_i = \Phi_i + \sum_j P_j F_{ji} \rho_i.$$

The origin of the random walks is chosen according to birth probabilities $\pi_i = \Phi_i / \Phi_T$, proportional to the self-emitted flux (Φ_T is the total self-emitted flux, division by it normalizes the birth probabilities). The transition probabilities are $p_{ji} = F_{ji}\rho_i$. In order to simulate transitions, first a uniformly distributed random line through j is traced in order to determine i, the next patch to be visited. Next, a survival test is done with ρ_i being the probability of survival. If the particle survives, a contribution Φ_T/N is recorded on the patch i on which the particle survived. N is the total number of random walks being traced. Both local or global lines can be used for sampling transitions. Global lines yield so called *global multi-path* algorithms [55, 50].

The random walk estimator sketched above is called a *survival shooting* estimator:

- *survival* estimation: particles contribute a score only if they survive a collision with a surface. Some alternatives will be discussed below (§4.3);
- *shooting*: the physical interpretation is that of particles being shot from the light sources.

4.2.3 Discrete gathering random walks for radiosity

A well known result from algebra states that each scalar product $<\mathbf{x}, \mathbf{w}>$ like (22) with the solution $\mathbf{x}$ of a linear system $\mathbf{C}\mathbf{x} = \mathbf{e}$ can also be obtained as a scalar product $<\mathbf{e}, \mathbf{y}>$ of the source term $\mathbf{e}$ with the solution of the *adjoint* system of linear equations $\mathbf{C}^\top \mathbf{y} = \mathbf{w}$ with source term $\mathbf{w}$:

$$<\mathbf{w}, \mathbf{x}> = <\mathbf{C}^\top \mathbf{y}, \mathbf{x}> = <\mathbf{y}, \mathbf{C}\mathbf{x}> = <\mathbf{y}, \mathbf{e}> .$$

$\mathbf{C}^\top$ denotes the transposed matrix $\mathbf{C}$: if $\mathbf{C} = \{c_{ij}\}$, then $\mathbf{C}^\top = \{c_{ji}\}$.

Adjoint systems corresponding to the radiosity system of equation (6) look like:

$$Y_i = W_i + \sum_j Y_j \rho_j F_{ji}. \tag{23}$$

These adjoints systems and the statement above can be interpreted as follows: consider the power P_k emitted by a patch k. P_k can be written as a scalar product $P_k = A_k B_k = <B, W>$ with $W_i = A_i \delta_{ik}$: all components of the *direct importance* vector W are 0, except the k-th component, which is equal to $W_k = A_k$. The statement above implies that P_k can also be obtained as $P_k = < Y, E > = \sum_i Y_i E_i$, which is a weighted sum of the self-emitted radiosities at the light sources in the scene.

The solution Y of the adjoint system (23) indicates to what extent each light source contributes to the radiosity at k. Y is called *importance* or *potential* in literature [65, 45, 7].

The adjoints (23) of the radiosity system also have the indices of the form factors in the right order, so they can be solved using a random walk simulation with transitions sampled with local or global lines. The particles are now however shot from the patch of interest ($\pi_i = \delta_{ki}$), instead of from the light sources. The transitions probabilities are $p_{ji} = \rho_j F_{ji}$: first an absorption/survival test is performed. If the particle survives, it is propagated to a new patch, with probabilities corresponding to the form factors. A non-zero contribution to the radiosity of patch k results whenever the imaginary particle hits a light source. Its physical interpretation is that of *gathering*.

Continuous gathering random walk methods can be obtained in a very similar way, by introducing adjoints of an integral equation. Adjoints of the radiosity integral equation for instance, look like:

$$I(x) = V(x) + \int_S I(y)\rho(y)G(y,x)dA_y.$$

Continuous gathering random walks are the basis of path tracing algorithms [13, 25].

4.3 Scoring

The previous paragraphs already mentioned that random walks can be used in different ways. The straightforward analog shooting simulations correspond to so called *survival* estimation, in which a particle yields a score whenever it survives a collision with a surface. In graphics, we are more familiar with random walks in which particles yield a score whenever they hit a surface, also at absorbed. This kind of random walk estimators are called *collision* estimators. A third kind which is occasionally mentioned in graphics literature are so called *absorption* estimators, in which a particle yields a score only when it is absorbed. In the same spirit, many more exotic random walk estimators can be developed, for instance estimators that yield a score only on the one but last, second but last, $\ldots$ collision, or estimators that yield a score on the two, three, ... last collision (see figure 5).

In all these cases, the random walks are constructed in an identical manner, as outlined before. The difference is only in their scores: 1) when a score is contributed and, related, 2) what value is contributed. The derivation of the scores to be associated with a random walk in order to obtain unbiased estimators for solving linear systems, and the calculation of the variance of random walk estimators, involves a fair amount of algebra. In most texts, a particular expression of the scores is proposed and the unbiasedness and variance derived according to the definition of expected value and variance. A more compact, constructive approach, based on just two theorems, can be found in [1]. The first theorem leads to a set of equations, allowing to derive scores that guarantee unbiased estimation by construction. The second theorem yields a general expression for the variance.

Table 1 and 2 summarize the results for the discrete absorption, collision and survival shooting and gathering random walks for radiosity [50, 51].

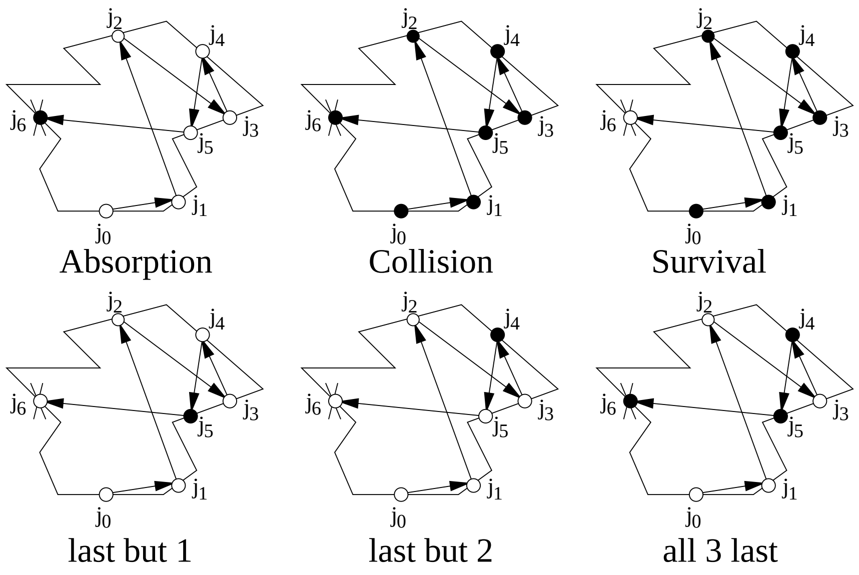


Figure 5: Kinds of random walk estimators: the black dots indicate at what nodes the path is allowed to yield a contribution. Gathering random walks originate at a patch of interest. They yield a non-zero score only when a light source is hit at the allowed locations. Shooting random walks originate at a light source and contribute a score to all patches at the allowed locations.

estimator	score $\tilde{s}(j_0, \ldots, j_\tau)$	variance $V[\tilde{s}]$
absorption	$\frac{\rho_k}{A_k} \frac{\Phi_T}{1-\rho_k} \delta_{j_\tau k}$	$\frac{\rho_k}{A_k} \frac{\Phi_T}{1-\rho_k} b_k - b_k^2$
collision	$\frac{\rho_k}{A_k} \Phi_T \sum_{t=1}^{\tau} \delta_{j_t k}$	$\frac{\rho_k}{A_k} \Phi_T (1 + 2\zeta_k) b_k - b_k^2$
survival	$\frac{1}{A_k} \Phi_T \sum_{t=1}^{\tau-1} \delta_{j_t k}$	$\frac{1}{A_k} \Phi_T (1 + 2\zeta_k) b_k - b_k^2$

Table 1: Score and variance of discrete shooting random walk estimators for radiosity. j_0 is the patch at which a random walk originates. It is a patch on a light source in the scene, chosen with probability proportional to its self-emitted power. $j_1, \ldots, j_\tau$ are the patches subsequently visited by the random walk. Transitions are sampled by first doing a survival/absorption test, with survival probability equal to the reflectivity. After survival, the next visited patch is selected with probability equal to the form factor, by tracing local or global lines. τ is the length of the random walk: the random walk is absorbed after hitting the patch j_τ. The expectation of all these estimators is the non-selfemitted radiosity $b_k = B_k - E_k$ at a patch k (source term estimation is suppressed). The left column with mathematical expressions indicates what score shall be contributed to the result for unbiased estimation. The right column contains the variance. ζ_k is the recurrent radiosity at k: if k would be the only source of radiosity, with unit strength, the total radiosity on k would be larger than 1, say I_k, because other patches in the scene reflect part of the light emitted by k back to k. The recurrent radiosity then would be $\zeta_k = I_k - 1$.

4.4 Discussion

Random walk estimators for radiosity thus can be classified according to the following criteria:

estimator	score $\tilde{s}(j_0 = k, \ldots, j_\tau) = \tilde{s}_k(J)$	variance $V[\tilde{s}_k]$
absorption	$\rho_k \frac{E_{j_\tau}}{1-\rho_{j_\tau}}$	$\rho_k \sum_s \frac{E_s}{1-\rho_s} b_{ks} - b_k^2$
collision	$\rho_k \sum_{t=1}^{\tau} E_{j_t}$	$\rho_k \sum_s (E_s + 2b_s) b_{ks} - b_k^2$
survival	$\rho_k \sum_{t=1}^{\tau-1} \frac{E_{j_t}}{\rho_{j_t}}$	$\rho_k \sum_s \frac{E_s+2b_s}{\rho_s} b_{ks} - b_k^2$

Table 2: Score and variance of discrete gathering random walk estimators for radiosity. The expectation is $b_k = B_k - E_k$, the non-selfemitted radiosity on patch k (also here, source term estimation is suppressed). j_0 is the patch at which a particular random walk originates. It is always the patch k of interest. $j_1, \ldots, j_\tau$ are subsequently visited patches. Transitions are sampled in exactly the same way as for shooting random walks: first an absorption/survival test is carried out, with probability of survival equal to the reflectivity. After survival, a next patch is selected with probability equal to the form factor, by tracing local or global lines. τ is the length of the random walk. b_{ks} is the radiosity at k due to the light source s, received directly or via interreflections from other patches ($b_s = \sum_s b_{ks}$).

- whether they are *continuous* or *discrete* random walks;
- whether they are *shooting* or *gathering*;
- according to where they generate a contribution: at absorption, survival, at every collision, etc . . . .

4.4.1 Continuous versus discrete random walks

Continuous random walks and discrete random walks estimate different quantities (see §1.1.3). The difference is in practice however only rarely noticeable. Also the algorithmic difference is quite small: with a continuous random walk, a particle is always reflected from its point of incidence on a patch. In a discrete random walk, a particle is reflected from a uniformly chosen different location on the patch on which it landed.

Experiments in which a continuous and discrete collision shooting random walk have been compared, indicate that there is no significant difference in variance. Low discrepancy sampling however, appears to be significantly more effective with the discrete random walk than with the continuous random walk [1].

4.4.2 Shooting versus gathering

The variance expressions in the tables above allow to make a detailed theoretical comparison of discrete shooting and gathering random walks. The shooting estimators have lower variance, except on small patches which have low probability of being hit by rays shot from light sources. Unlike shooting estimators, the variance of gathering estimators does not depend on the patch area A_k. For sufficiently small patches, gathering will be more efficient. Gathering could, like in the case of stochastic relaxation methods, be used in order to “clean” noisy artifacts on small patches after shooting.

4.4.3 Absorption, survival or collision?

The variance results in the tables above also indicate that the survival estimators are always worse than the corresponding collision estimators, because the reflectivity ρ_k is always smaller than 1. As a rule, the collision estimators also have lower variance than the absorption estimators, because the recurrent radiosity ζ_k is in general negligible (shooting) and self-emitted radiosity E_s of a light source is in general much larger than the non-selfemitted radiosity in practice (gathering).

These results hold when transitions are sampled according to the form factors. When the transition probabilities are modulated, for instance to shoot more rays into important directions (§5.1), an absorption estimation can sometimes be better than a collision estimator. In particular, it can be shown that a collision estimator can never be perfect, because random walks can contribute a variable number of scores. An absorption estimator always yields a single score. The more exotic estimators mentioned above, which yield scores at the two last, three last, ... collisions also yield a fixed number of scores. For that reason, they may also yield perfect estimators. Their analysis is an interesting topic for further research.

4.4.4 Discrete collision shooting random walks versus stochastic Jacobi relaxation

According to table 1, the variance of N^{RW} discrete collision shooting random walks is approximately:

$$\frac{V^{RW}}{N^{RW}} \approx \frac{1}{N^{RW}} \frac{\rho_k}{A_k} \Phi_T (B_k - E_k)$$

The variance of incremental power shooting (17) with N^{SR} rays is approximately:

$$\frac{V^{SR}}{N^{SR}} \approx \frac{1}{N^{SR}} \frac{\rho_k}{A_k} P_T (B_k - E_k).$$

It can be shown that N^{RW} random walks result on the average in $N^{RW} P_T / \Phi_T$ rays to be shot. Filling in $N^{SR} = N^{RW} P_T / \Phi_T$ in the expression above thus indicates for *for same number of rays, discrete collision shooting random walks and incremental power shooting Jacobi iterations are approximately equally efficient*. This observation has been confirmed in experiments [1].

Both algorithms have an intuitive interpretation in the sense of particles being shot from patches. The particles have uniform starting position on the patches and they have cosine-distributed directions w.r.t. the normal on the patches. The number of particles shot from each patch is proportional to the power propagated from the patch. Since the two methods compute the same result, the same number of particles will be shot from each of the patches. If also the same random numbers are used to shoot particles from each patch, the particles themselves can also be expected to be the same. The main difference is the order in which the particles are shot: they are shot in “breath-first” order in stochastic relaxation and in “depth-first” order with random walks (see figure 6).

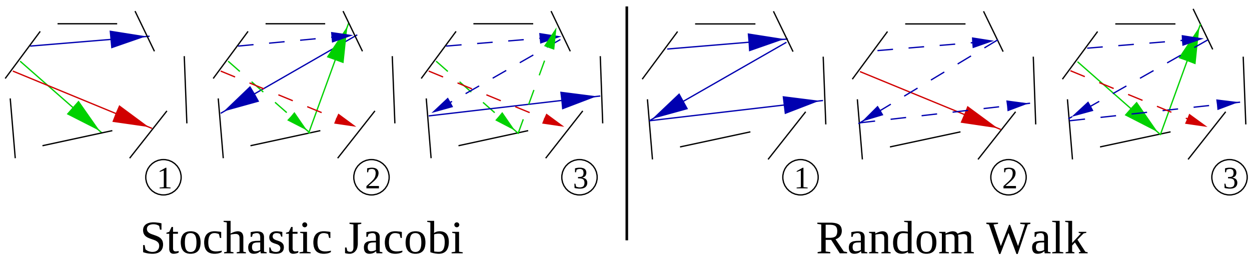


Figure 6: This figure illustrates the difference in order in which particles are shot in stochastic Jacobi iterations ("breadth-first" order) and in collision shooting random walk radiosity ("depth-first" order). Eventually, the shot particles are very similar.

There are however also other, more subtle, differences between the algorithms, in particular in the survival sampling. Experiments with very simple scenes, such as an empty cube, where recurrent radiosity ζ_k is important, do reveal a different performance. The conclusion that stochastic Jacobi iterations and random walks are equally efficient is also no longer true when higher order approximation are used, or with low discrepancy sampling or in combination with variance reduction techniques. Many variance reduction techniques and low discrepancy sampling are easier to implement and appear more effective for stochastic relaxation than with random walks (both continuous or discrete, see §5). Stochastic relaxation with higher order approximation appears at least as good as with continuous random walks, and is significantly superior to a discrete random walk (§6.1).

5 Variance Reduction and Low Discrepancy Sampling

The basic algorithms of the previous sections can be made more effective by using variance reduction techniques and low discrepancy sampling. In this section, we will discuss variance reduction by view-importance sampling, by control variates, by combining gathering and shooting estimators using the same random walks or rays and by weighted importance sampling.

5.1 View-importance driven shooting

5.1.1 View-importance

In the basic algorithms of the previous sections, transitions are sampled using probabilities that reflect the laws of physics. The quality of the computed result mainly depends on the area and reflectivity of the patches, but is furthermore uniform in the whole scene. Sometimes however, one would like to save computation time by having a high quality only in a part of the scene, for instance the part of the scene that is visible in a view, while compromising on the quality in unimportant parts of the scene. For instance, when computing an image inside a single room in a large building with several floors each containing many rooms, the basic estimators would spend a lot of work in computing the illumination in all rooms on all floors to similar quality. One might prefer to concentrate the computation work on the room one is in, at the expense of a lower quality of the radiosity solution in other rooms and other floors of the building. With view-importance sampling, the sampling probabilities in our Monte Carlo radiosity algorithms are modulated in such a way that more samples are taken in important regions of a scene, and fewer in less important regions.

This requires in the first place a measure for the importance of the illumination across the surfaces in the scene. As explained in §4.2.3, the adjoints of the radiosity system of equations yield such a measure. Here, it will be more convenient to use adjoints of the power system of equations (14)[7]:

$$I_i = V_i + \sum_j F_{ij} \rho_j I_j. \tag{24}$$

The importances I_i are always defined w.r.t. some direct importance distribution V_i. When choosing $V_i = 1$ for the patches i that are visible in a view, and $V_i = 0$ for patches that are not visible in a view, I_i is called *view-importance* and indicates what fraction of the radiosity B_i will be contributed to the patches visible in a view, directly or via inter-reflections.

A continuous view importance function $I(x)$ on the surfaces of the scene can be defined in a very similar way by means of adjoints of the radiosity integral equation (3):

$$I(x) = V(x) + \int_S I(y)\rho(y)G(y,x)dA_y. \tag{25}$$

[7] Adjoint radiosity systems (23) are obtained by multiplying the left and right hand side of if (24) with the patch area A_i. Y_i and W_i in (23) are related to I_i and V_i here as $Y_i = A_i I_i$ and $W_i = A_i V_i$.

The equations from which importance is to be solved are of the same form as the equations that describe light transport, and therefore the same algorithms as for light transport can be used for computing importance in a scene. This can happen either in separate phases, or both at the same time. Moreover, the computation of importance can possibly be sped up by taking advantage of the adjoint of importance: the radiosity. In practice, one should take care however that importance is only used for computing radiosity (and vice versa) if the importance solution is sufficiently stable.

5.1.2 View-importance driven shooting random walks

View-importance I_i can be used in various ways during random walk sampling:

- for modulating the transition probabilities, so that random walks are scattered preferentially towards regions of high importance. Unfortunately, this can no longer be done using uniformly distributed local or global lines and requires that incoming importance at every patch is stored or can be queried efficiently in some way [33, 67];
- for modulating the survival probabilities only, so particles near important regions get a higher chance of survival. In regions of low importance, particles will be killed off with a higher probability than according to the reflectivity (*Russian roulette*). In interesting regions, it even is possible to split a particle in two new particles of which the scores are appropriately combined (*splitting*);
- for modulating the birth probabilities, so that more random walks are started from important light sources and fewer from unimportant sources. This can be combined with importance-modulated transition sampling, or can be done with analog transition sampling. In the latter case, the best results are obtained by modulating the analog birth probabilities at light sources (proportional to self-emitted power) by the *square root* of view-importance [52].

In order to keep the estimation unbiased, scores shall be decreased when probabilities are increased and vice versa. If the survival chance of a particle is reduced in Russian roulette for instance, the contribution of a particle that survives the test shall be increased in order to compensate. View-importance based sampling has been studied for continuous as well as discrete random walks [45, 15, 52, 54, 1].

5.1.3 View-importance driven stochastic relaxation radiosity

In the context of incremental and regular power shooting (§3.2), view importance can be used in order to

- aim particles preferentially towards interesting regions: the problem is the same as with random walks: local or global line sampling is no longer helpful and incoming importance needs to be stored with each patch;
- increase or decrease the probability of shooting a ray from a given patch: this yields the same effect as Russian roulette, splitting and modulating birth probabilities together in random walks. It is very easy to implement with local line sampling.

In general, view-importance driven stochastic relaxation methods can be derived in exactly the same way as analog stochastic relaxation methods by considering the power system of equations (14) modified as follows:

$$P_i I_i = \Phi_i I_i + \sum_j P_j (I_j - V_j) F_{ji} \frac{\rho_i I_i}{I_j - V_j}.$$

Non view-importance driven stochastic relaxation radiosity corresponds with the choices $I_i = 1/\rho_i$ and $V_i = 1/\rho_i - 1$. (These choices are always a valid solution of (24) in closed environments). Figure 7 shows some results, obtained with algorithms developed by Neumann and Bekaert [36, 1].

5.2 Control variates

Another well known variance reduction technique is by means of so called control variates. Suppose a function $f(x)$ is to be numerically integrated and that we know the integral G of a similar function $g(x)$. If the difference $f(x) - g(x)$ is to good approximation constant, it will be more efficient to use a Monte Carlo method for integrating the difference $f(x) - g(x)$ and add G afterwards. The function $g(x)$ is called a control variate. Control variates have been proposed for variance reduction in stochastic raytracing by Lafortune [34]. We discuss here the application to discrete random walks and stochastic relaxation.

5.2.1 Control variates for linear systems

This idea can be applied to the solution of linear systems (and integral equations) in the following way: suppose we know an approximation $\tilde{\mathbf{x}}$ for the solution $\mathbf{x}$ of $\mathbf{x} = \mathbf{e} + \mathbf{A}\mathbf{x}$. The correction $\Delta\mathbf{x} = \mathbf{x} - \tilde{\mathbf{x}}$ then fulfills

$$\Delta\mathbf{x} = (\mathbf{e} + \mathbf{A}\tilde{\mathbf{x}} - \tilde{\mathbf{x}}) + \mathbf{A} \cdot \Delta\mathbf{x} \tag{26}$$

Proof:

$$\Delta\mathbf{x} = (\mathbf{I} - \mathbf{A}) \cdot \Delta\mathbf{x} + \mathbf{A} \cdot \Delta\mathbf{x} \; ; \; (\mathbf{I} - \mathbf{A}) \cdot \Delta\mathbf{x} = \mathbf{x} - \mathbf{A}\mathbf{x} + \mathbf{A}\tilde{\mathbf{x}} - \tilde{\mathbf{x}} = \mathbf{e} + \mathbf{A}\tilde{\mathbf{x}} - \tilde{\mathbf{x}}$$

□

This is true regardless of the error in the approximation $\tilde{\mathbf{x}}$. Now suppose $\Delta\mathbf{x}$ is computed using for instance a random walk method. The resulting estimate $\Delta\tilde{\mathbf{x}}$ for the correction $\Delta\mathbf{x}$ will not be exact, so that $\tilde{\tilde{\mathbf{x}}} = \tilde{\mathbf{x}} + \Delta\tilde{\mathbf{x}}$ will not be exactly equal to the solution $\mathbf{x}$ of the system to be solved either. However, regardless of the error on the first approximation $\tilde{\mathbf{x}}$, the error on the new approximation $\tilde{\tilde{\mathbf{x}}}$ is only determined by the error on the computed correction $\Delta\tilde{\mathbf{x}}$! Sometimes, the correction $\Delta\tilde{\mathbf{x}}$ can be estimated more efficiently than $\mathbf{x}$ itself.

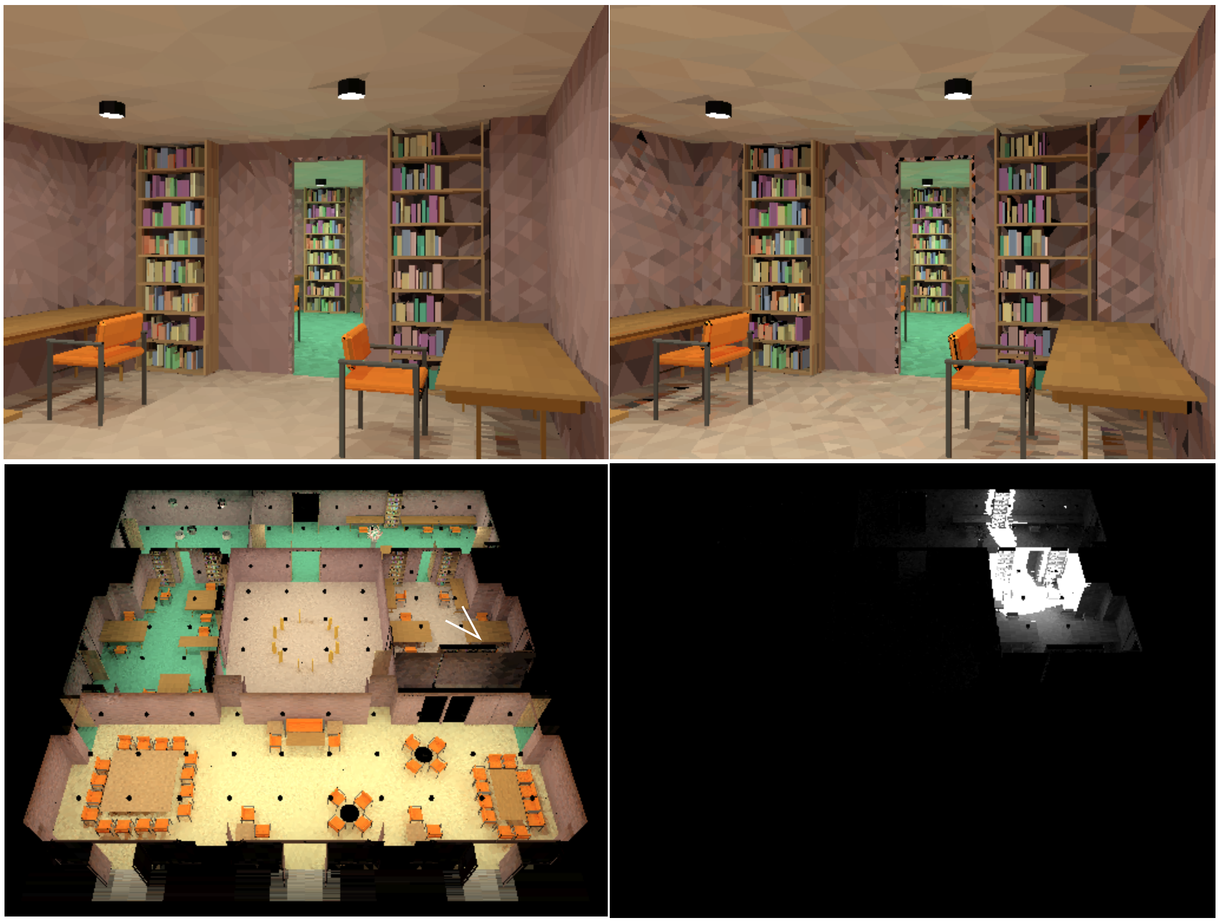


Figure 7: View-importance driven stochastic Jacobi radiosity: the top images have been obtained using approximately the same total amount of work (about 9 minutes, 3.3 10^6 rays). The top-left image, computed with view-importance, is significantly less noisy than the top-right image, which has been obtained without computing and taking advantage of view-importance. The bottom-left image shows an overview of the scene in which the view was taken. The scene was subdivided in 162,000 patches. The bottom-right image shows the importance distribution for the view. High intensity indicates high view-importance.
The shown model is an edited part of the Soda Hall VRML model made available at the University of California at Berkeley.

5.2.2 Constant control variates in random walk radiosity

The only choice for $\tilde{\mathbf{x}}$ that allows $\mathbf{A}\tilde{\mathbf{x}}$ to be calculated analytically in the case of radiosity, is the constant choice $\tilde{B}_i = \beta$. With this choice, we get

$$\begin{aligned}\Delta B_i &= \left(E_i + \sum_j \rho_i F_{ij}\beta - \beta\right) + \sum_j \rho_i F_{ij} \Delta B_j \\ &= (E_i - (1-\rho_i)\beta) + \sum_j \rho_i F_{ij} \Delta B_j\end{aligned}$$

The question now is how to determine an optimal value for β. Heuristics for choosing β can be derived by minimizing the expected mean square error of random walk estimators. Several crude approximations need to be made however, and the benefits are in practice not very significant.

5.2.3 Constant control variates in stochastic relaxation radiosity

In stochastic Jacobi relaxation however, constant control variate variance reduction is easier to obtain and more effective. Monte Carlo summation shall be applied to the following modified power equations:

$$P'_i = \Phi_i + A_i \rho_i \beta + \sum_i \sum_j A_j (B_j - \beta) F_{ji} \rho_i \delta_{ik}$$

A good value for the control radiosity β can be obtained by numerical optimization of $F(\beta) = \sum_s A_s \left| B_s - \beta \right|$ [1, 41].

One disadvantage of constant control variates in radiosity is that the scene being rendered needs to fulfill certain requirements:

- it needs to be closed, because otherwise $\sum_j F_{ij}\beta \neq \beta$ for some patches i in the scene;
- there cannot be closed "holes" in a scene that do not receive any light, e.g. the interior of a box.

The speed up that can be obtained with a constant control variate typically is in the range of 5-50%.

5.3 Gathering for free

Yet another variance reduction technique described in Monte Carlo literature is the combination of several estimators for the same quantity. In Monte Carlo radiosity, one will always find a gathering estimator corresponding with each shooting estimator. Gathering is in general less efficient than shooting, but by combining gathering and shooting over random walks and rays sampled for shooting, a moderate variance reduction is possible at negligible additional cost.

5.3.1 Combining estimators and multiple importance sampling

Suppose two estimators $\hat{S}_1$ and $\hat{S}_2$ for a given quantity S are available. Any linear combination $w_1\hat{S}_1 + w_2\hat{S}_2$ with constant weights $w_1 + w_2 = 1$ will then also be an estimator for S. The variance of the linear combination however depends on the weights:

$$V[w_1\hat{S}_1 + w_2\hat{S}_2] = w_1^2 V[\hat{S}_1] + w_2^2 V[\hat{S}_2] + 2w_1 w_2 \text{Cov}[\hat{S}_1, \hat{S}_2].$$

Cov$[\hat{S}_1, \hat{S}_2]$ denotes the *co-variance* of the two estimators:

$$\text{Cov}[\hat{S}_1, \hat{S}_2] = E[\hat{S}_1 \cdot \hat{S}_2] - E[\hat{S}_1] \cdot E[\hat{S}_2].$$

If $\hat{S}_1$ and $\hat{S}_2$ are independent, the covariance is zero[8]. Minimization of the variance expression above allows to fix the optimal combination weights:

$$\frac{w_1}{w_2} = \frac{V[\hat{S}_2] - \text{Cov}[\hat{S}_1, \hat{S}_2]}{V[\hat{S}_1] - \text{Cov}[\hat{S}_1, \hat{S}_2]}.$$

For independent estimators, the weights shall be inverse proportional to the variance. In practice, the weights can be calculated in two different ways:

- using analytical expressions for the variance of the involved estimators (such as presented in this text);
- using a-posteriori estimates for the variances based on the samples in an experiment themselves [26]. By doing so, a slight bias is introduced. As the number of samples is increased, the bias vanishes: the combination is asymptotically unbiased or *consistent*.

The combination of estimators can be generalized for more than two, say M estimators. If N_m out of a total of N samples are taken from each estimator $\hat{S}_m$, yielding primary estimates $\tilde{S}_m^k, k = 1, \ldots, N_m$ for S, the combined estimates look like:

$$\sum_{m=1}^{M} w_m \frac{1}{N_m} \sum_{k=1}^{N_m} \tilde{S}_m^k \approx S.$$

Veach [73] noted that very robust combination is often possible by assigning potentially different weights w_m^k to each individual sample, even for samples from the same estimator:

$$\sum_{m=1}^{M} \frac{1}{N_m} \sum_{k=1}^{N_m} w_m^k \tilde{S}_m^k \approx S.$$

The corresponding combined estimator is unbiased as long as $\sum_{m=1}^{M} w_m^k = 1$ for every sample. He proposed several heuristics for determining the weights. The basic idea behind these heuristics is to

[8]A zero covariance is a necessary, but not a sufficient condition for independence: there exist dependent estimators that also have zero covariance (see e.g. [26, p.13]).

give a sample a weight that takes into account the probability that the sample would have resulted with the other estimators: if the sample would be generated only with low probability with the other estimators, it is given a large weight. Vice versa, when any other estimator would yield the same sample with high probability, the weight of the sample is reduced. One of those heuristics is the so called *balance heuristic*, in which the weights w_m^k are simply chosen proportional to the probability of drawing the sample according to the m-th estimator, multiplied with the number of samples N_m.

5.3.2 Combining gathering and shooting in discrete random walk radiosity

Combining gathering and shooting over a single set of random walks can be done in several ways:

- using multiple importance sampling: the basic observation is that gathering radiosity over a path segment $j_t, j_{t+1}, \ldots, j_s$ is identical to shooting power over the reverse segment $j_s, j_{s-1}, \ldots, j_t$. Multiple importance sampling can be applied if the probability of having a sub-path originating at the end-points j_t and j_s are both known. In practice, combined gathering and shooting based on multiple importance sampling is useful only with global lines, in global multi-path algorithms [55, 50]. With local lines, the required probabilities are unfortunately not known in advance;

- using a-posteriori variance estimates: such estimates can be obtained by approximating analytical expressions [53]. Alternatively, sample based variance estimation is also possible [1]. Sample based variance estimation yields very good weights eventually, but the weights are unreliable in the beginning of the computations, when only few random walks have been visiting a patch. A-posteriori variance estimation allows to combine shooting and gathering also with local line sampling. Figure 8 shows the shooting and gathering contributions associated with a single path.

Combining gathering and shooting in random walk radiosity yields moderate variance reduction, again 5-50%, but the additional computation cost is negligible.

5.3.3 Combining gathering and shooting in stochastic Jacobi radiosity

Combining gathering and shooting in stochastic Jacobi iterations is again very simple [1]. Each line shot in power shooting iterations (§3.2.1) yields a contribution to the patch that it hits, while in gathering iterations (§3.2.4), the line yields a contribution to the patch from where it was shot. Also here, gathering corresponds with shooting over the reverse line. Unlike with random walks, the probability of shooting a line from every patch is known, so multiple importance sampling can be used. The result is that a score can be recorded at both ends of each shot line. For a line connecting the patches i and j, the scores at both end points are:

$$\begin{aligned} w_{ij} S_{ij}^{\leftarrow} &= \frac{\rho_i P_j}{p_i A_j + p_j A_i} \quad \text{on } i \\ w_{ij} S_{ij}^{\rightarrow} &= \frac{\rho_j P_i}{p_i A_j + p_j A_i} \quad \text{on } j \end{aligned}$$

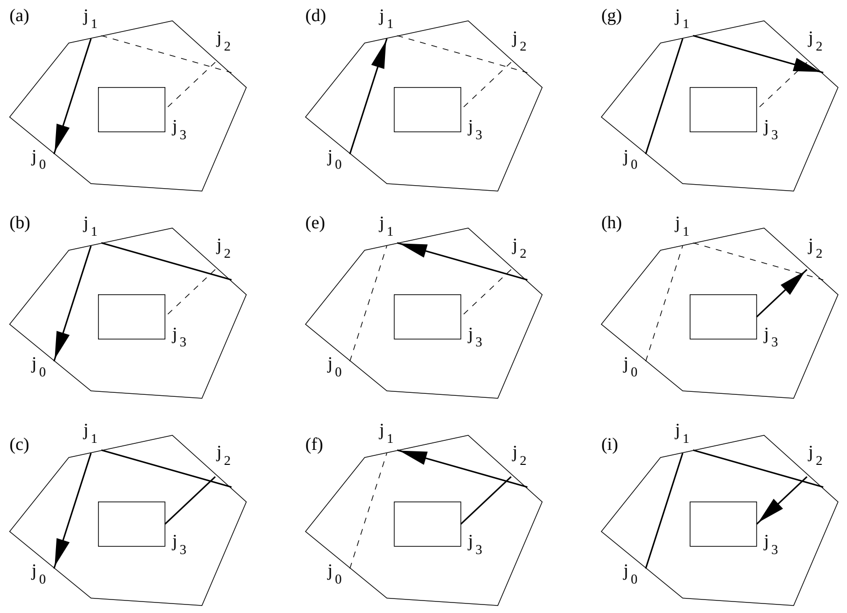


Figure 8: Contributions of a random walk j_0, j_1, j_2, j_3: (a,b,c) gathering at j_0; (d) shooting at j_1; (e,f) gathering at j_1; (g) shooting at j_2; (h) gathering at j_2; (i) shooting at j_3.

As before, p_i and p_j indicate the probability of shooting a line from i and j. With local lines, we can choose p_i proportional to the power to be shot from i. With global lines, p_i is proportional to the patch area A_i.

The technique is extremely simple to implement, it is always safe to use, it comes at no additional cost and can yield fair speed-ups: up to a factor of 2 if illumination is nearly uniform.

5.4 Weighted importance sampling

Weighted importance sampling is yet another variance reduction technique. It can be explained intuitively as follows: suppose one needs to compute an integral $F = \int f(x)dx$ and that one knows a second, similar, integral $G = \int g(x)dx$ with same domain. Both integrals can then be estimated using the same samples. The resulting Monte Carlo estimate $\tilde{G}$ for G can then be compared with the true, known, value of G. Due to its random nature, the estimate $\tilde{G}$ will sometimes be larger than G and sometimes be smaller. Suppose that one knows that the corresponding estimate $\tilde{F}$ for F will also be larger than F in case $\tilde{G}$ is larger than G, a more accurate estimate for F then may be $\tilde{F}G/\tilde{G}$: $\tilde{F}$ is decreased if $\tilde{G} > G$ and it is increased if $\tilde{G} < G$.

Unlike the variance reductions techniques described before, weighted importance sampling is biased, but it is consistent if f and g fulfill certain requirements. The bias vanishes as $1/N$ (N is the number of samples). This is much faster than the statistical error, which vanishes as $1/\sqrt{N}$. A more elaborate exposition of this idea, with application to form factor integration and stochastic relaxation radiosity, can be found in [4].

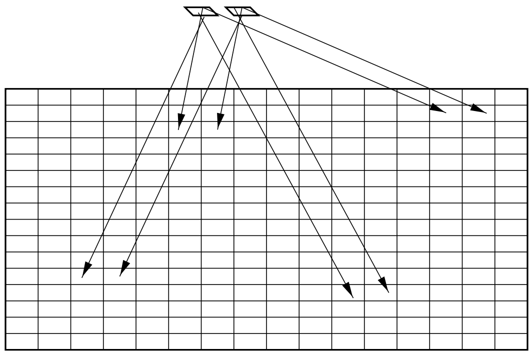


Figure 9: If sample vectors with same index are used on different patches, for choosing the origin and direction of the rays, distracting patterns may result in a computed image. This figure illustrates how such patterns may result due to clustered contributions from nearby sources.

5.5 Low-discrepancy sampling

Low discrepancy sampling [42] allows to achieve a higher convergence rate than $\mathcal{O}(1/\sqrt{N})$ by placing samples more uniformly than random. The theory behind numerical integration with low-discrepancy sampling, called *quasi-Monte Carlo* integration, is totally different than that of Monte Carlo methods based on random sampling. Integration with random samples is based on statistics. Quasi-Monte Carlo methods are based on number theory. The convergence rates than can be obtained depend on the dimension d of the integral and are non-trivial to analyze. They can be $\mathcal{O}(\log^d N/N)$.

In practice however, improved convergence rates are often obtained by little more than replacing the random number generator by a so called quasi-random number generator. Local line sampling (§2.2.1) for instance, requires 4-dimensional random vectors: two random numbers are needed for choosing a ray origin and two more for sampling a cosine distributed direction. Keller [27] showed that using a 4-dimensional quasi-random vector [42] yields speed-ups of about an order of magnitude when computing form factors with local lines. Neumann et al. observed a similar speed-up when using quasi-random numbers instead of random numbers in stochastic relaxation radiosity [39]. The speed-up obtained with quasi-random sampling in continuous shooting random walk radiosity [28] is however much smaller. In discrete shooting random walk radiosity, it is of the same magnitude as in stochastic relaxation radiosity, and often much higher than in continuous random walks [1]. A theoretical study of the convergence rate of quasi-random sampling in radiosity has been carried out by Szirmay-Kalos [71].

There are however a number of important differences between random and quasi-random sampling. The main difference in practice is that quasi-random samples are not statistically independent. They can even be very strongly correlated. Naive application can result in disturbing aliasing artifacts in computed images (see figure 9). With local line sampling in radiosity, the correlations can be broken by keeping a separate sample index for each patch. The sample index is initialized to a different value for each patch, and is incremented independently from other patches each time a new ray needs to be shot from the patch.

6 Extensions

To conclude, a number of useful extensions to the Monte Carlo algorithms described so far will be presented in this section. Especially the latter, the incorporation of hierarchical refinement, makes Monte Carlo radiosity algorithms highly competitive with deterministic radiosity algorithms.

6.1 Higher order radiosity

So far, the focus has been on the computation of a constant radiosity approximation on each patch. On patches where the radiosity function $B(x)$ varies smoothly (everywhere except near shadow boundaries), higher order approximations (linear, quadratic, . . .) will be more accurate at a little higher storage cost (one additional value per basis function, see figure 1 on page 5).

We already mentioned how continuous random walks can be used in order to compute a higher order approximation in §4.1.4. Similar algorithms can also be developed for stochastic relaxation radiosity and for discrete random walks [3, 1]. The necessary adaptations to stochastic relaxation radiosity are minimal: local or global lines can be cast as for constant approximations. Only the contributed scores are slightly different, and contain the dual basis function $\tilde{\psi}_{i,\alpha}(x)$ at the point x hit by the lines.

Feda [17] analysed the cost of using continuous random walks in order to estimate a product Legendre basis approximation of radiosity. He observed that the required number of samples for a K-th order approximation is K^2 times larger than for a constant approximations. Bekaert has shown that this is also the case for higher-order stochastic relaxation radiosity. The increase in computation time for higher order approximations is larger than in deterministic methods [23, 75], but the resulting algorithms are significantly easier to implement and are much less sensitive to computational errors (see figure 10).

Higher-order approximations can also be estimated with discrete random walks. The result that discrete random walks are as good as stochastic relaxation radiosity does however not hold for higher order approximations: discrete random walks have a much higher variance, which gets worse for bright environments and higher approximation order. They are not recommended for practical use.

6.2 Hierarchical refinement and clustering

Accurate a-priori meshing for radiosity is a difficult task. On the one hand side, the patches shall be small enough in order to accurately capture high-frequency illumination variations, such as near shadow boundaries. On the other hand, the number of patches needs to be kept as small as possible in order to avoid redundant computation work. With hierarchical refinement [10, 21, 56], large input patches will be broken up into smaller elements during the computations when needed, as predicted by a *refinement oracle* function.

Small input patches can also be grouped into *clusters* in order to reduce the number of form factors to be computed during *initial linking* in deterministic radiosity algorithms [64, 61]. A user is not only liberated from the difficult task of creating a suitable mesh himself, but hierarchical

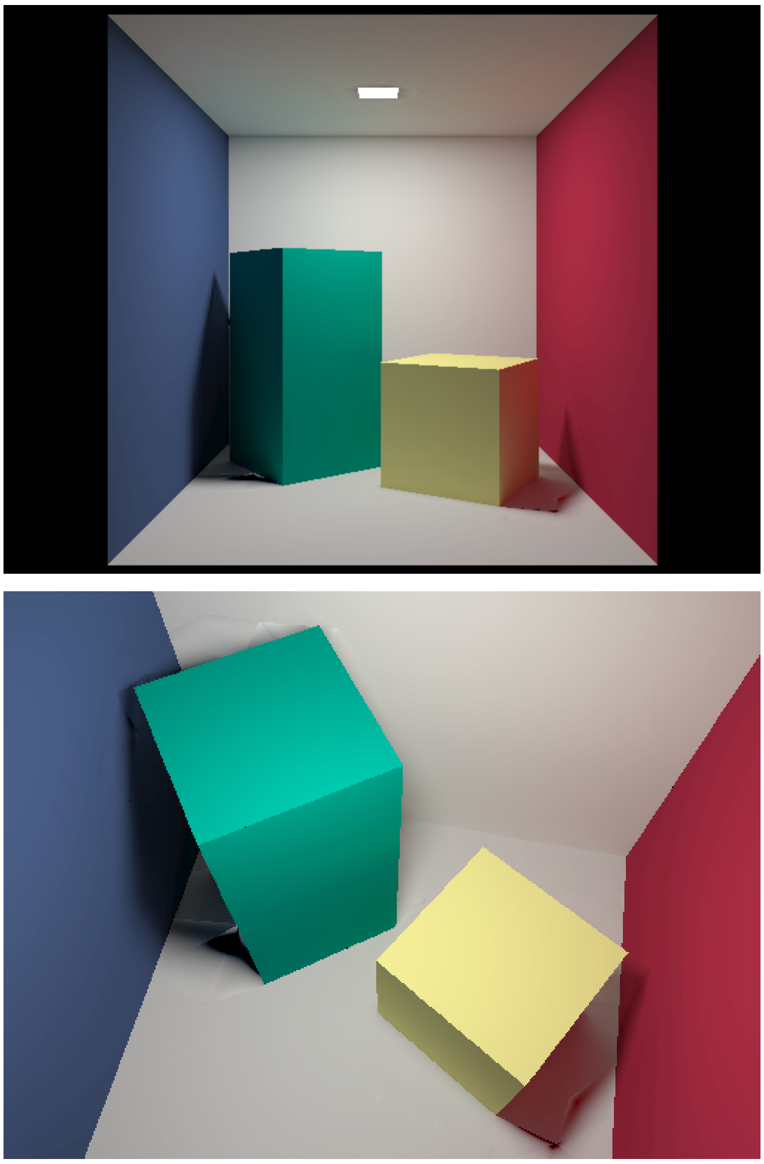


Figure 10: Two images generated from the same converged cubic approximation solution. Once the solution has been obtained, a new image for a new viewpoint can be generated in fractions of a second. These images illustrate also that the (discrete) higher order stochastic relaxation radiosity algorithm described in these notes can yield very high image quality in regions where the illumination varies smoothly: computational error is dealt with effectively in Monte Carlo radiosity. In the neighborhood of discontinuities however, disturbing image artifacts remain due to the discretisation error. The resulting image artifacts would be avoided if discontinuity meshing were used.

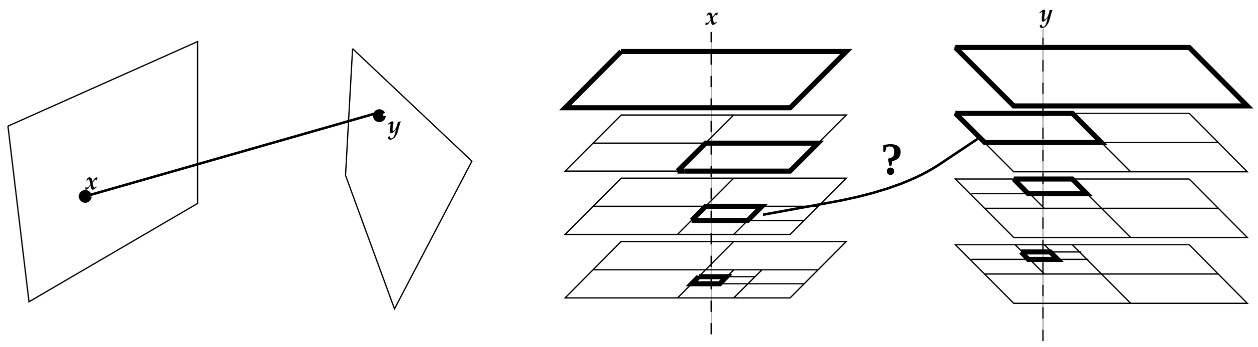


Figure 11: Per-ray hierarchical refinement in stochastic Jacobi radiosity: for each ray shot, connecting two points x and y, the algorithm will determine which level of the element hierarchies at x and y is appropriate for computing light transport from x to y. The element hierarchies are lazily constructed. In non-hierarchical Monte Carlo radiosity, light transport would be computed between the input patches containing the end-points x and y of the ray.

refinement and clustering also allow to compute light transport always at a proper level of detail. The number of form factors to be computed is reduced from quadratic to log-linear in this way.

Hierarchical refinement can be incorporated also in Monte Carlo radiosity algorithms, combining the benefits of hierarchical refinement with lower sensitivity for computational error in Monte Carlo radiosity.

The incorporation of hierarchical refinement during form factor computation with local lines has been proposed by several authors [35, 31, 30], who used it as a replacement for the hemi-cube algorithm in progressive refinement radiosity. The basic idea in these three proposals is identical: a large amount of rays is shot from the source patch. The surrounding scene is subdivided in receiver elements so that each receiver element (a surface or cluster) receives the same amount of rays. The disadvantage is that these techniques will only work if a large number of rays is shot simultaneously from the shooting patch. This is not the case in more recent stochastic relaxation algorithms.

Tobler et al. [72] have presented an adaptive meshing scheme for continuous shooting random walk radiosity (§4.1.3). By simultaneously keeping track of incident particles on successive hierarchical element levels, smoothness assumption violations can be detected.

Bekaert et al. [2, 1] proposed to incorporate hierarchical refinement in stochastic Jacobi radiosity by means of *per-ray* refinement. The basic observation is that each line cast in non-hierarchical stochastic Jacobi radiosity carries some flux from the patch containing its origin to the patch containing its destination point. With hierarchical refinement, a whole stack of elements corresponds with both end-points. A refinement oracle will predict for each line, at what level of the element stack containing the destination point of a line, the flux carried by the line shall be deposited (see figure 11). Elements are refined lazily, on the fly during the computations.

Per ray refinement works extremely well with a cheap oracle, such as based on transported power [21]. Some results are presented in figure 12. Hierarchical stochastic Jacobi radiosity has been used in order to rapidly compute radiosity solutions in scenes containing millions of polygons, such as entire building and car models, on high end PCs.

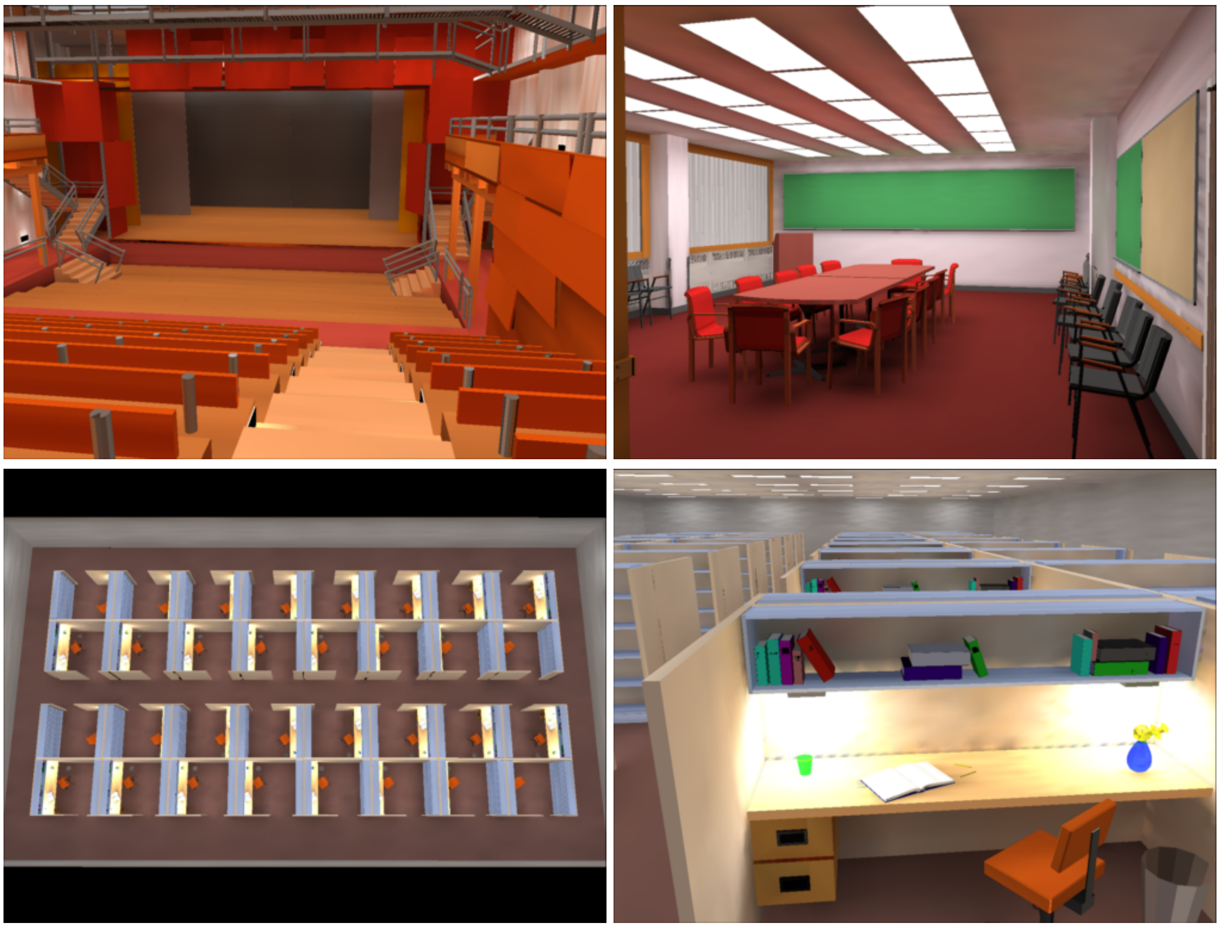


Figure 12: Complex scenes rendered with hierarchical Monte Carlo radiosity: theater (39,000 initial polygons, refined into 88,000 elements, 5 minutes), conference room (125,000 polygons, refinement yields 178,000 elements, 9 minutes) and cubicle office space (128,000 polygons, refined into 506,000 elements, 10 minutes).
Model credits: Candlestick Theater: Design: Mark Mack Architects, 3D Model: Charles Ehrlich and Greg Ward (work conducted as a research project during the Architecture 239X course taught by Kevin Matthews formerly at UC Berkeley, College of Environmental Design). Conference room and cubicle space models by Anat Grynberg and Greg Ward (Lawrence Berkeley Laboratory, Berkeley, California).

Notations

$< X, Y >$	scalar product of X and Y (functions or vectors)
$\tilde{X}$	estimate or approximation for X, or modified quantity X
$\hat{X}$	Monte Carlo estimator for a quantity X (random variable for estimating X)
$\lfloor x \rfloor$	largest integer smaller than or equal to x
$\mathbf{A}, \mathbf{C}$	matrices with elements a_{ij}, c_{ij}
$\mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{r}, \mathbf{x}, \mathbf{w}$	vectors with components $a_i, b_i, \ldots$
A_i	surface area of patch i
dA_x	differential area at a point x
$\alpha_i, \alpha(x)$	absorption probability of a random walk
B_i	radiosity emitted by patch i (constant approximations)
$B_{i,\alpha}$	radiosity emitted "under" basis function $\psi_{i,\alpha}$ in $\tilde{B}(x) = \sum_{i,\alpha} B_{i,\alpha}\psi_{i,\alpha}(x)$
$B(x)$	"true" radiosity function at point x
b_i	$B_i - E_i$: non-selfemitted radiosity at patch i
b_{il}	radiosity at i due to light source patch l: $b_i = \sum_l b_{il}$
Cov$[\hat{X}, \hat{Y}]$	co-variance of Monte Carlo estimators $\hat{X}$ and $\hat{Y}$
$\mathbf{C}^\top$	transpose of the matrix $\mathbf{C}$: $c^\top_{ij} = c_{ji}$
$\chi_i, \chi(x)$	random walk hit point density
δ_{ij}	Kronecker's delta: 1 if $i = j$, 0 if $i \neq j$
E_i	self-emitted radiosity by patch i (constant approximations)
$E(x)$	"true" self-emitted radiosity function at x
$E[\hat{X}]$	expectation of the Monte Carlo estimator $\hat{X}$ for X
F_{ij}	patch-i-to-patch-j form factor
$f_r(x; \Theta \leftrightarrow \Theta')$	BRDF at x for scattering from a direction Θ into direction Θ' or vice versa
$G(x, y)$	geometric radiosity kernel
$h(x, \Theta)$	first point on a surface visible from x in the direction Θ
I_i	importance at patch i (radiosity-like)
$L^{\rightarrow}(x, \Theta), L^{\leftarrow}(x, \Theta)$	exitant and incident radiance at x into/from direction Θ
N	number of samples in a Monte Carlo computation
n	size of a problem, for instance number of equations in a linear system
Ω_x	hemisphere of directions above x
$d\omega_\Theta$	infinitesimal solid angle containing direction Θ
P_i	A_iB_i: power emitted by patch i
$p(x)$	(continuous) probability density at a point x
p_i	(discrete) probability at a patch or state i
p_{ij}	transition probability from i to j
π_i	random walk birth probability at i
Φ_i	A_iE_i: self-emitted power at patch i

$\psi_{i,\alpha}$	α-th basis function on patch i
$\tilde{\psi}_{i,\alpha}$	α-th dual basis function on patch i: $\int_{S_i} \tilde{\psi}_{i,\alpha}(x)\psi_{i,\beta}(x)dA_x = \delta_{\alpha\beta}$
r_{xy}	distance between points x and y
ρ_i	reflectivity of patch i
$\rho(x)$	reflectivity at point x
S_i	surface of patch i (set of points)
Θ_x	out-going direction at point x
θ_x	angle between Θ_s and the surface normal at x
Y_i	$A_i I_i$: importance at patch i (power-like)
V_i	source-importance at patch i (radiosity-like)
$V[\hat{X}]$	variance of the Monte Carlo estimator $\hat{X}$ for X
vis(x, y)	visibility predicate: 1 if x and y are mutually visible, 0 if not
W_i	$A_i V_i$: source-importance at patch i (power-like)
ζ_i	recurrent radiosity: fraction of radiosity on i due to itself

References

[1] Ph. Bekaert. *Hierarchical and Stochastic Algorithms for Radiosity.* PhD thesis, K. U. Leuven, department of computer science, December 1999.

[2] Ph. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. D. Willems. Hierarchical Monte Carlo radiosity. In *Proceedings of the 9th. Eurographics Workshop on Rendering, Vienna, Austria*, June 1998.

[3] Ph. Bekaert, M. Sbert, and Y. Willems. The computation of higher-order radiosity approximations with a stochastic Jacobi iterative method. In *16th Spring Conference on Computer Graphics, Comenius University, Bratislava, Slovakia*. May 2000. 212–221.

[4] Ph. Bekaert, M. Sbert, and Y. Willems. Weighted importance sampling techniques for Monte Carlo radiosity. In *Rendering Techniques '2000 (Proceedings of the 11th Eurographics Workshop on Rendering, Brno, Czech Rep.)*, page 13 pages. Springer Computer Science, June 2000.

[5] K. Bouatouch, S. N. Pattanaik, and E. Zeghers. Computation of higher order illumination with a non-deterministic approach. *Computer Graphics Forum*, 15(3):327–338, August 1996.

[6] S. E. Chen, H. E. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 165–174, July 1991.

[7] P. H. Christensen, D. H. Salesin, and T. D. DeRose. A Continuous Adjoint Formulation for Radiance Transport. In *Fourth Eurographics Workshop on Rendering*, pages 95–104, Paris, France, June 1993.

[8] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.

[9] M. F. Cohen and D. P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):31–40, July 1985.

[10] M. F. Cohen, D. P. Greenberg, D. S. Immel, and P. J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(3):26–35, March 1986.

[11] M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press Professional, Boston, MA, 1993.

[12] S. Collins. Adaptive Splatting for Specular to Diffuse Light Transport. In *Fifth Eurographics Workshop on Rendering*, pages 119–135, June 1994.

[13] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics*, 18(3):137–145, July 1984.

[14] L. M. Delves and J. L. Mohamed. *Computational methods for integral equations.* Cambridge University Press, 1985.

[15] Ph. Dutré and Y. D. Willems. Potential-driven Monte Carlo particle tracing for diffuse environments with adaptive probability density functions. In *Eurographics Rendering Workshop 1995*, June 1995.

[16] S. M. Ermakow. *Die Monte-Carlo-Methode und verwandte Fragen*. V.E.B. Deutscher Verlag der Wissenschaften, Berlin, 1975.

[17] M. Feda. A Monte Carlo approach for Galerkin radiosity. *The Visual Computer*, 12(8):390–405, 1996.

[18] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84 Conference Proceedings (Minneapolis, MN, July 23-27, 1984)*, pages 213–222, July 1984.

[19] J. H. Halton. A restrospective and prospective survey of the Monte Carlo method. *SIAM Review*, 12(1):1 – 63, January 1970.

[20] J. M. Hammersley and D. C. Handscomb. *Monte Carlo methods.* Methuen London/Chapman and Hall, 1964.

[21] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.

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

[23] P. S. Heckbert and J. Winget. Finite element methods for global illumination. Technical Report UCB/CSD 91/643, Computer Science Division (EECS), University of California, Berkeley, California, USA, July 1991.

[24] H. W. Jensen. Global illumination using photon maps. In *Eurographics Rendering Workshop 1996*, pages 21–30. Eurographics, June 1996.

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

[26] M. H. Kalos and P. Whitlock. *The Monte Carlo method.* J. Wiley and sons, 1986.

[27] A. Keller. The Fast Calculation of Form Factors Using Low Discrepancy Sequences. In *Proceedings of the Spring Conference on Computer Graphics (SCCG '96)*, pages 195–204, Bratislava, Slovakia, June 1996. Comenius University Press.

[28] A. Keller. Quasi-Monte Carlo radiosity. In *Eurographics Rendering Workshop 1996*, pages 101–110, June 1996.

[29] A. Keller. Instant radiosity. In *SIGGRAPH 97 Conference Proceedings*, pages 49–56, August 1997.

[30] A. Keller. *Quasi-Monte Carlo methods for photorealistic image synthesis.* PhD thesis, Universität Kaiserslautern, Germany, June 1997.

[31] A. Kok. *Ray Tracing and Radiosity Algorithms for Photorealistic Images Synthesis.* PhD thesis, Technische Universiteit Delft, The Netherlands, 1994.

[32] R. Kress. *Linear Integral Equations.* Springer Verlag, 1989.

[33] E. Lafortune and Y. D. Willems. A 5d tree to reduce the variance of Monte Carlo ray tracing. In *Rendering Techniques '95 (Proceedings of the Eurographics Workshop on Rendering, Dublin, Ireland*, pages 11–20, June 1995.

[34] E. P. Lafortune and Y. D. Willems. The ambient term as a variance reducing technique for Monte Carlo ray tracing. In *Fifth Eurographics Workshop on Rendering*, pages 163–171, Darmstadt, Germany, June 1994.

[35] E. Languenou, K. Bouatouch, and P. Tellier. An adaptive discretization method for radiosity. *Computer Graphics Forum*, 11(3):C205–C216, 1992.

[36] A. Neumann, L. Neumann, Ph. Bekaert, Y. D. Willems, and W. Purgathofer. Importance-driven stochastic ray radiosity. In *Eurographics Rendering Workshop 1996*, pages 111–122, June 1996.

[37] L. Neumann. Monte Carlo radiosity. *Computing*, 55(1):23–42, 1995.

[38] L. Neumann, M. Feda, M. Kopp, and W. Purgathofer. A New Stochastic Radiosity Method for Highly Complex Scenes. In *Fifth Eurographics Workshop on Rendering*, pages 195–206, Darmstadt, Germany, June 1994.

[39] L. Neumann, A. Neumann, and Ph. Bekaert. Radiosity with well distributed ray sets. *Computer Graphics Forum*, 16(3), 1997.

[40] L. Neumann, W. Purgathofer, R. Tobler, A. Neumann, P. Elias, M. Feda, and X. Pueyo. The stochastic ray method for radiosity. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, July 1995.

[41] L. Neumann, R. F. Tobler, and P. Elias. The Constant Radiosity Step. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 336–344. Springer-Verlag, 1995.

[42] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *CBMS-NSF regional conference series in Appl. Math.* SIAM, Philadelphia, 1992.

[43] T. Nishita and E. Nakamae. Continuous tone representation of 3-D objects taking account of shadows and interreflection. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):23–30, July 1985.

[44] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by Monte Carlo simulation of the particle model of light. *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.

[45] S. N. Pattanaik and S. P. Mudur. Adjoint equations and random walks for illumination computation. *ACM Transactions on Graphics*, 14(1):77–102, January 1995.

[46] M. Pellegrini. Monte Carlo approximation of form factors with error bounded a priori. In *Proc. of the 11th. annual symposium on Computational Geometry*, pages 287 – 296. ACM Press, 1995.

[47] R. Y. Rubinstein. *Simulation and the Monte Carlo method.* J. Wiley and sons, 1981.

[48] L. Santaló. *Integral Geometry and Geometric Probability.* Addison-Welsey, Reading, Mass, 1976.

[49] M. Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum*, 12(3):C409–C420, 1993.

[50] M. Sbert. *The use of global random directions to compute radiosity — Global Monte Carlo techniques.* PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, November 1996.

[51] M. Sbert. Error and complexity of random walk Monte Carlo radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):23–38, March 1997.

[52] M. Sbert. Optimal source selection in shooting random walk Monte Carlo radiosity. *Computer Graphics Forum*, 16(3):301–308, August 1997.

[53] M. Sbert, A. Brusi, and Ph. Bekaert. Gathering for free in random walk radiosity. In *Rendering Techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering, Granada, Spain)*, pages 97–102. Springer Computer Science, June 1999.

[54] M. Sbert, A. Brusi, R. Tobler, and W. Purgathofer. Random walk radiosity with generalized transition probabilities. Technical Report IIiA-98-07-RR, Institut d'Informàtica i Aplicacions, Universitat de Girona, January 1998.

[55] M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer. Global multipath Monte Carlo algorithms for radiosity. *The Visual Computer*, 12(2):47–61, 1996.

[56] P. Schröder. Numerical integration for radiosity in the presence of singularities. In *4 th Eurographics Workshop on Rendering, Paris, France*, pages 177–184, June 1993.

[57] P. Shirley. A ray tracing method for illumination calculation in diffuse–specular scenes. In *Graphics Interface '90*, pages 205–212, May 1990.

[58] P. Shirley. Radiosity via ray tracing. In J. Arvo, editor, *Graphics Gems II*, pages 306–310. Academic Press, San Diego, 1991.

[59] P. Shirley. Time complexity of Monte Carlo radiosity. In *Eurographics '91*, pages 459–465. North-Holland, September 1991.

[60] P. Shirley, B. Wade, Ph. M. Hubbard, D. Zareski, B. Walter, and Donald P. Greenberg. Global Illumination via Density Estimation. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 219–230, 1995.

[61] F. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):240–254, September 1995.

[62] F. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 335–344, July 1989.

[63] F. Sillion and C. Puech. *Radiosity and Global Illumination.* Morgan Kaufmann, San Francisco, 1994.

[64] B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. In *SIGGRAPH '94 Proceedings*, pages 435–442, July 1994.

[65] B. Smits, J. Arvo, and D. Salesin. An importance-driven radiosity algorithm. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 273–282, July 1992.

[66] J. Spanier and E. M. Gelbard. *Monte Carlo Principles and Neutron Transport Problems.* Addison-Wesley, 1969.

[67] L. Szirmay-Kalos, C. Balasz, and W. Purgathofer. Importance-driven quasi-random walk solution of the rendering equation. *Computers and Graphics*, 23(2):203–211, 1999.

[68] L. Szirmay-Kalos, T. Foris, L. Neumann, and C. Balasz. An analysis of quasi-Monte Carlo integration applied to the transillumination radiosity method. *Computer Graphics Forum (Eurographics '97 Proceedings)*, 16(3), 1997. C271–C281.

[69] L. Szirmay-Kalos, T. Foris, and W. Purgathofer. Quasi-Monte Carlo global light tracing with infinite number of rays. In *WSCG '98 (Sixth European Conference in Central Europe on Computer Graphics and Visualization)*, pages 386–393, Plzen, Czech Republic, 1998. University of West Bohemia.

[70] L. Szirmay-Kalos and W. Purgathofer. Global ray-bundle tracing with hardware acceleration. In *Ninth Eurographics Workshop on Rendering*, Vienna, Austria, June 1998.

[71] L. Szirmay-Kalos and W. Purgathofer. Analysis of the quasi-Monte Carlo integration of the rendering equation. In *WSCG '99 (Seventh International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media)*, pages 281–288, Plzen-Bory, Czech Republic, February 1999. University of West Bohemia.

[72] R. Tobler, A. Wilkie, M. Feda, and W. Purgathofer. A hierarchical subdivision algorithm for stochastic radiosity methods. In *Eurographics Rendering Workshop 1997*, pages 193–204, June 1997.

[73] E. Veach and L. J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *SIGGRAPH 95 Conference Proceedings*, pages 419–428, August 1995.

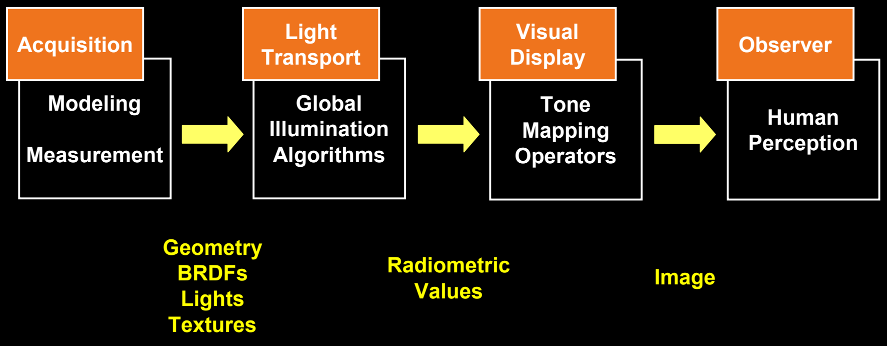
[74] B. Walter, Ph. M. Hubbard, P. Shirley, and D. F. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, July 1997.

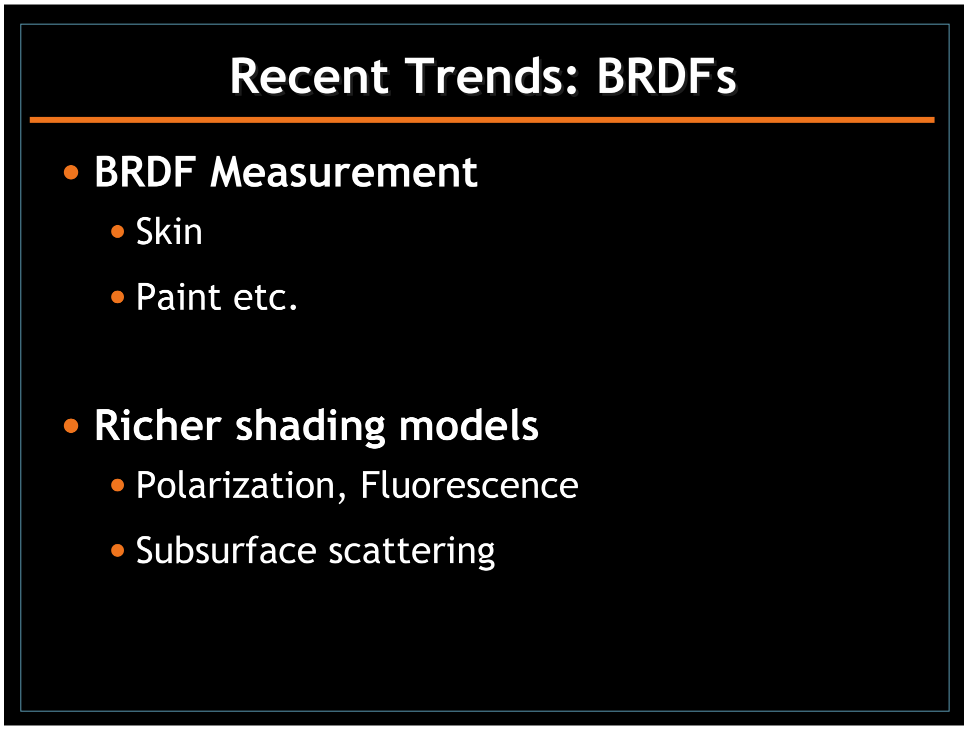
[75] H. R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 213–220, 1993.

Contents

Recent Trends and Future Directions

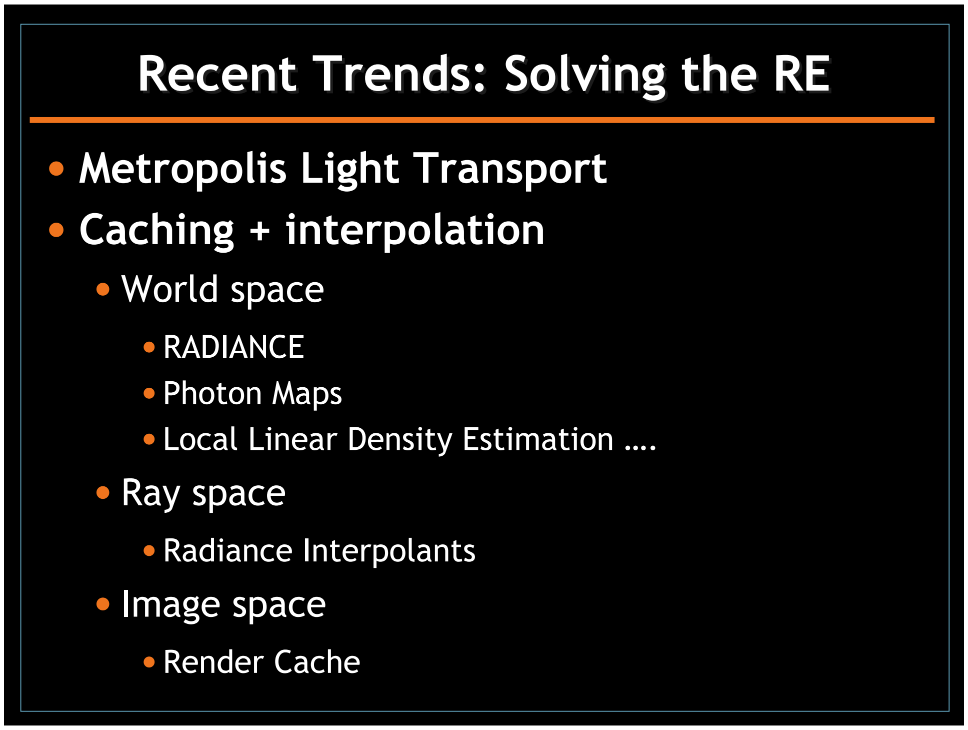
Framework





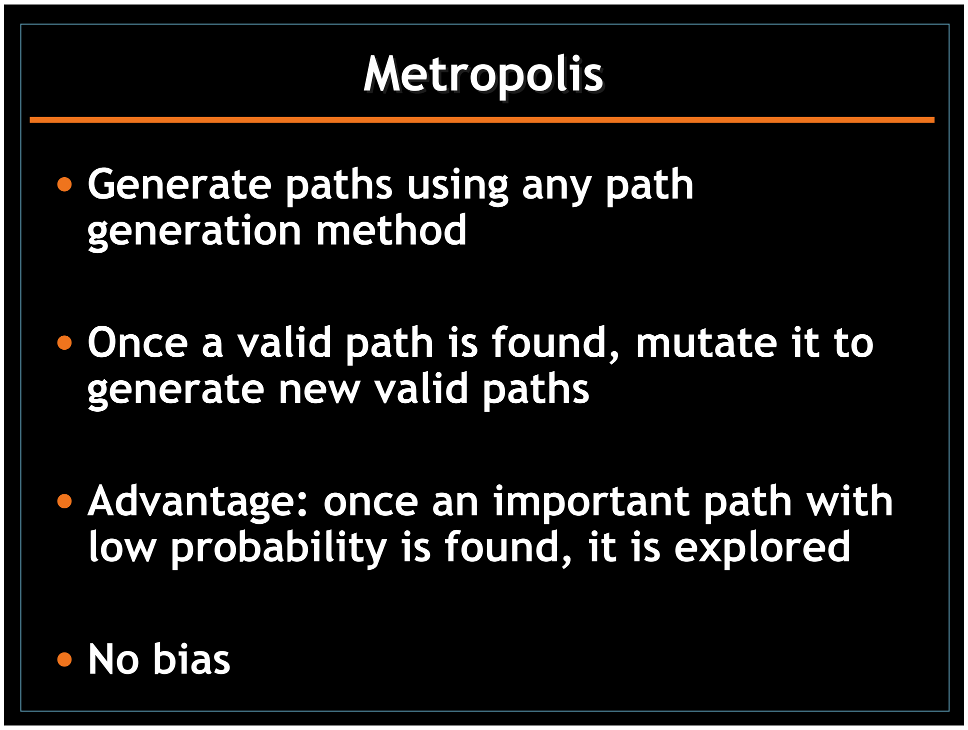
Recent Trends: BRDFs

- BRDF Measurement
  - Skin
  - Paint etc.
- Richer shading models
  - Polarization, Fluorescence
  - Subsurface scattering



Recent Trends: Solving the RE

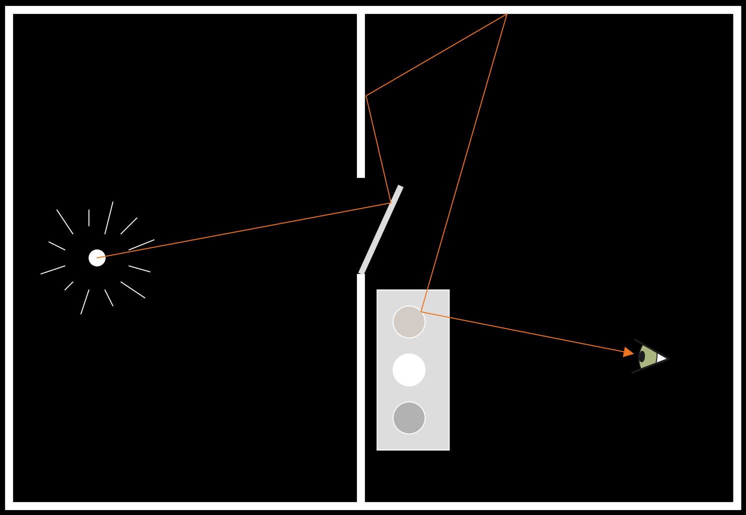
- Metropolis Light Transport
- Caching + interpolation
  - World space
    - RADIANCE
    - Photon Maps
    - Local Linear Density Estimation ….
  - Ray space
    - Radiance Interpolants
  - Image space
    - Render Cache

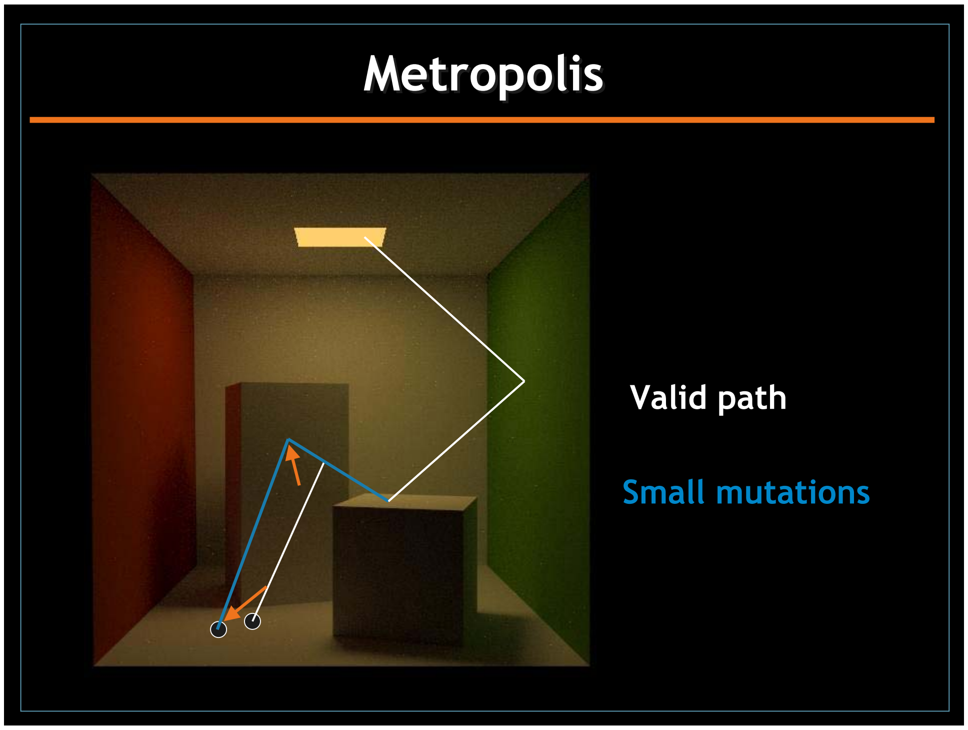


Metropolis

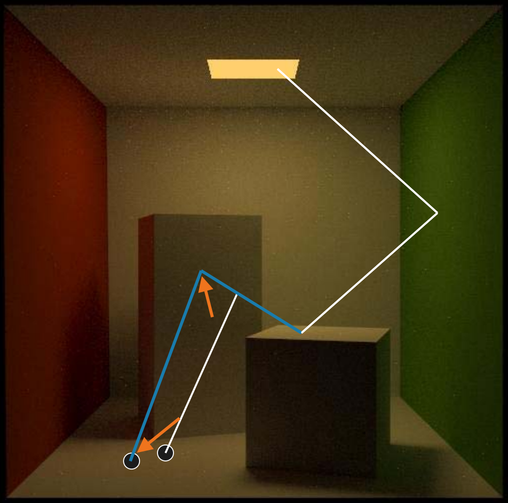
- Generate paths using any path generation method
- Once a valid path is found, mutate it to generate new valid paths
- Advantage: once an important path with low probability is found, it is explored
- No bias

Metropolis



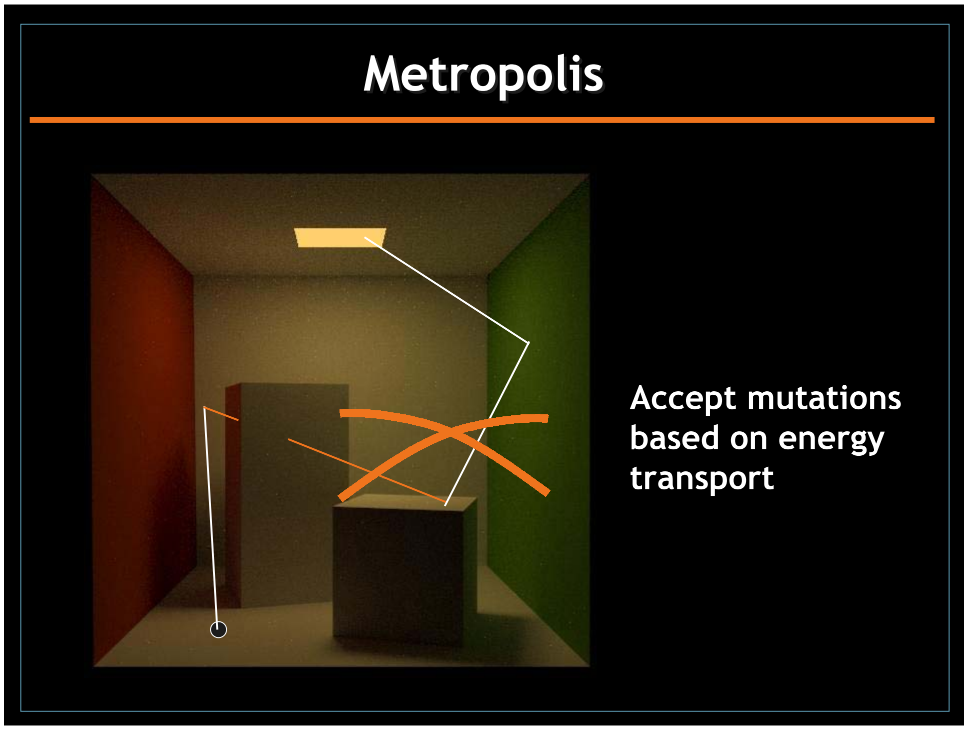


Metropolis

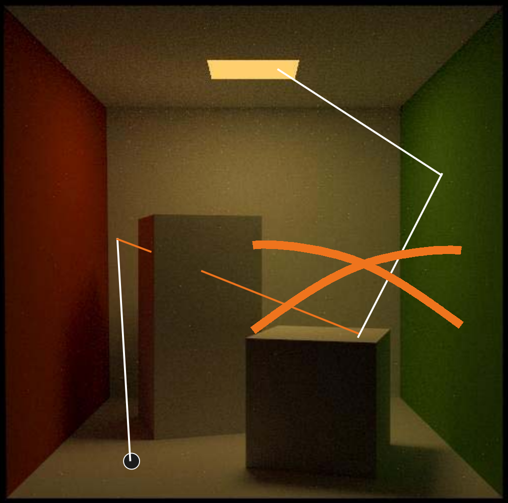


Valid path

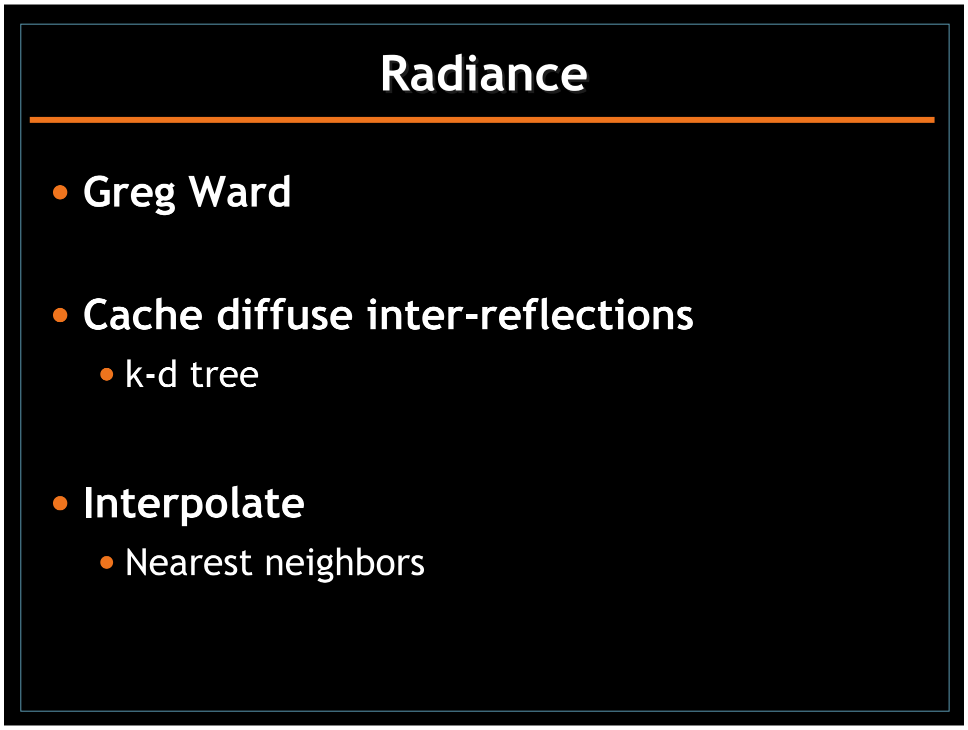
Small mutations



Metropolis

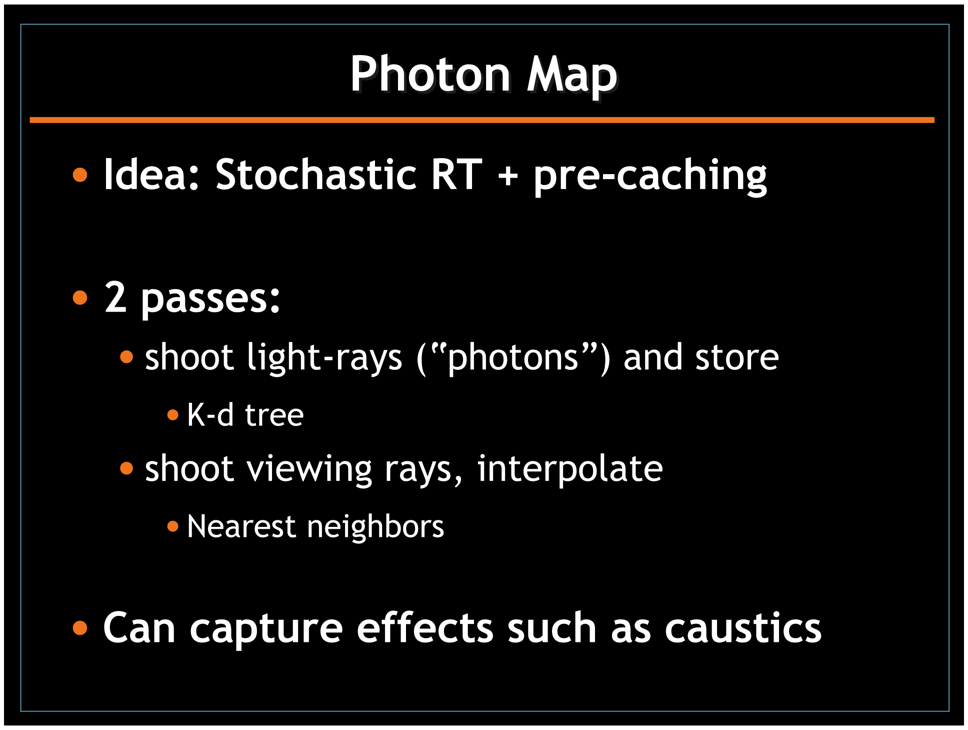


Accept mutations based on energy transport



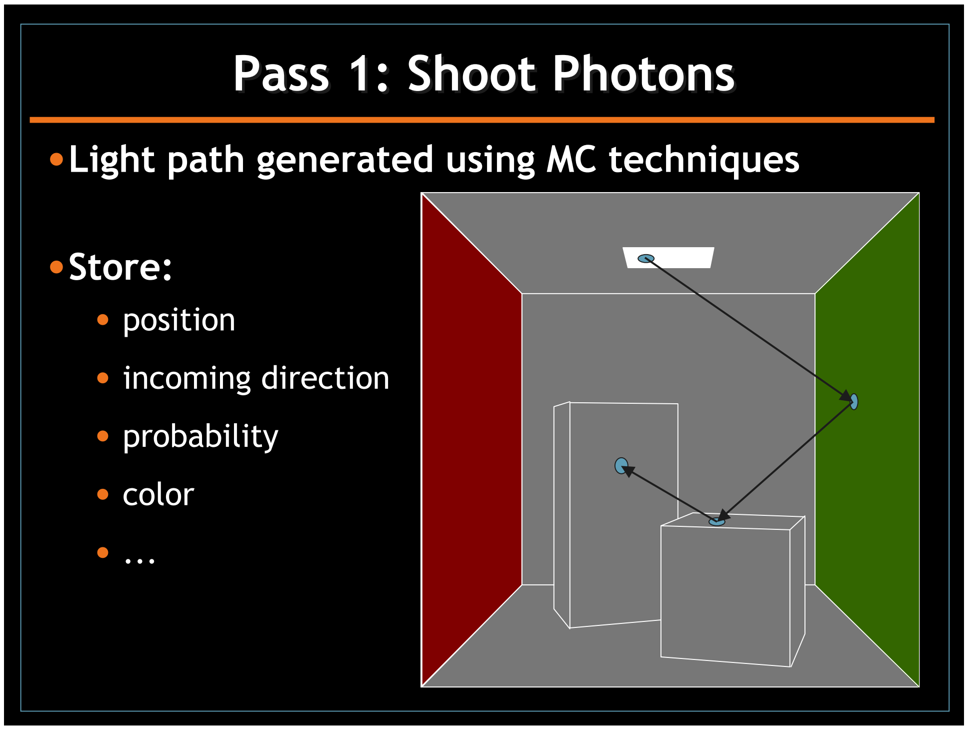
Radiance

- Greg Ward
- Cache diffuse inter-reflections
  - k-d tree
- Interpolate
  - Nearest neighbors



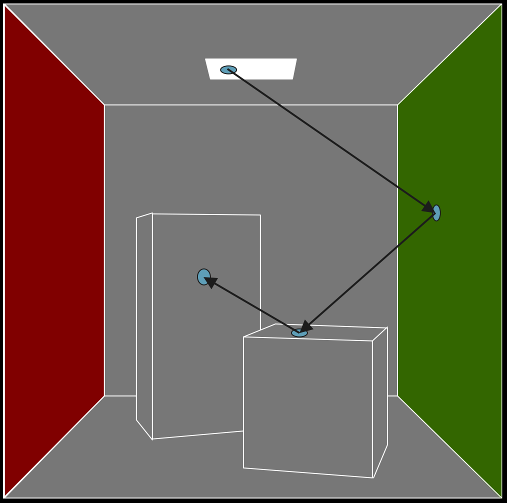
Photon Map

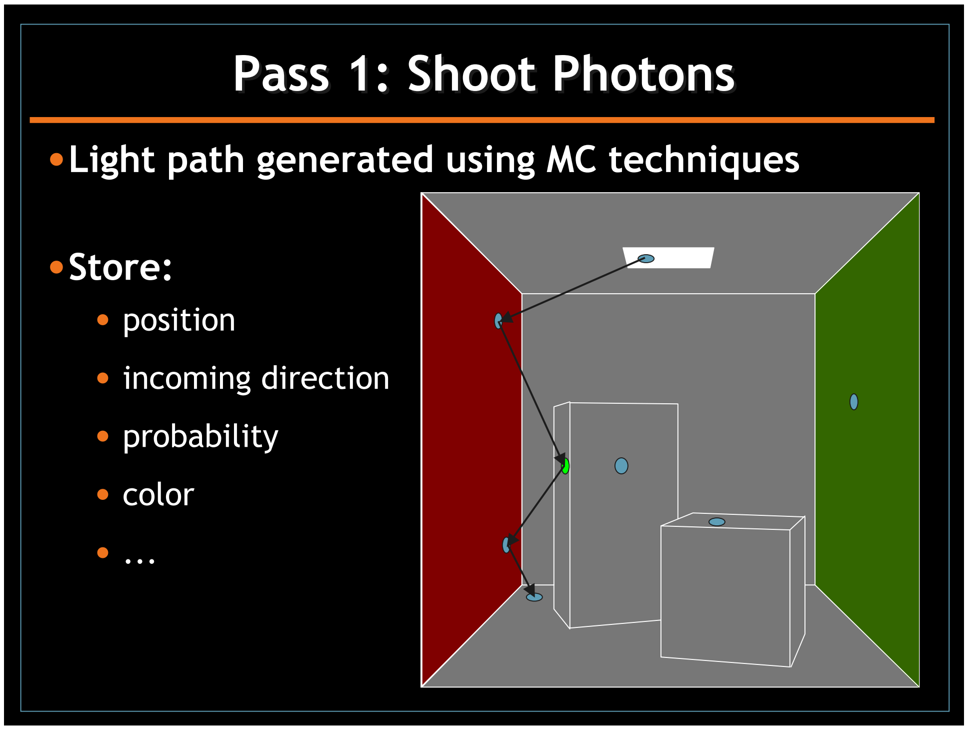
- Idea: Stochastic RT + pre-caching
- 2 passes:
  - shoot light-rays (“photons”) and store
    - K-d tree
  - shoot viewing rays, interpolate
    - Nearest neighbors
- Can capture effects such as caustics



Pass 1: Shoot Photons

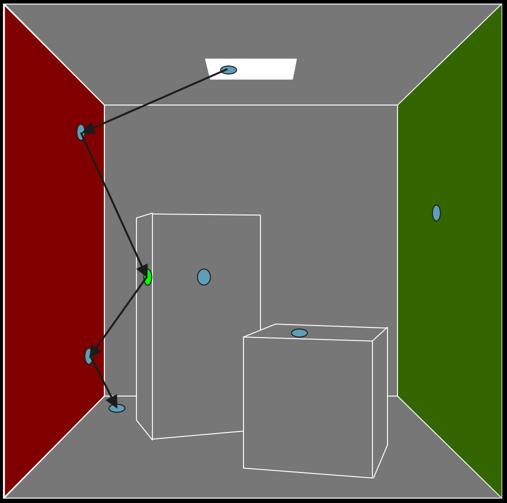
- Light path generated using MC techniques
- Store:
  - position
  - incoming direction
  - probability
  - color
  - …





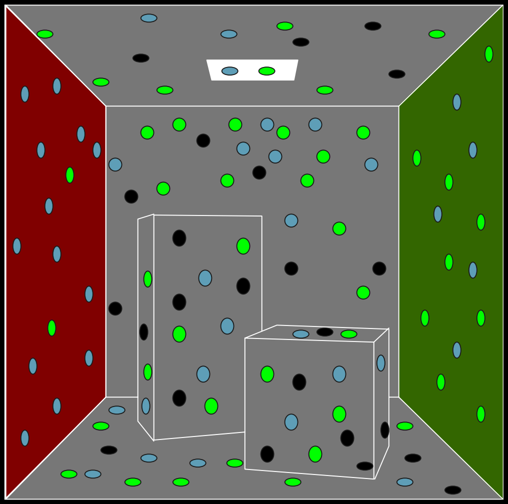
Pass 1: Shoot Photons

- Light path generated using MC techniques
- Store:
  - position
  - incoming direction
  - probability
  - color
  - …



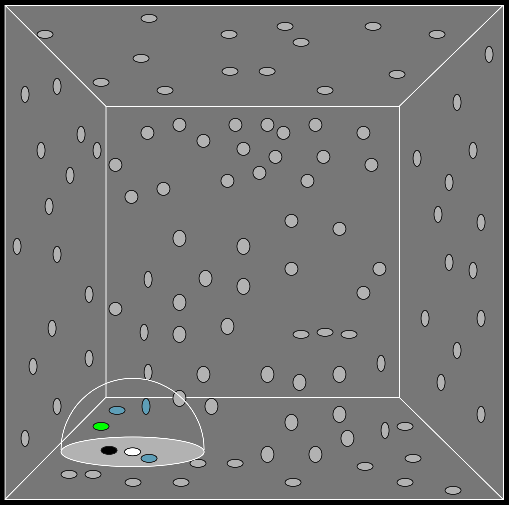
Pass 1: Shoot Photons

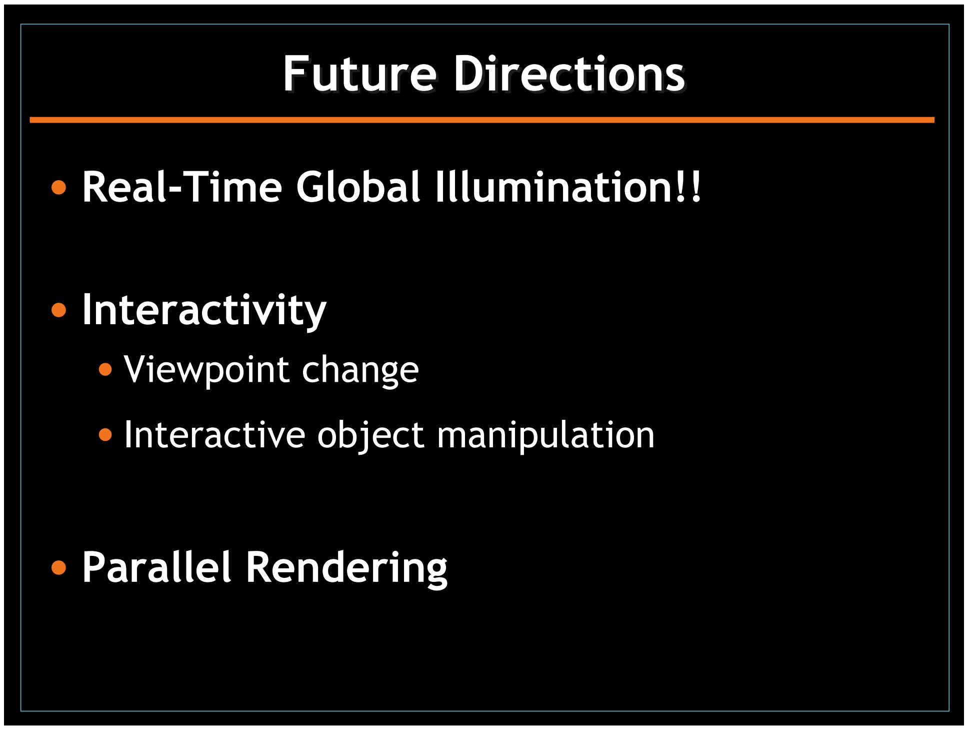
- Light path generated using MC techniques
- Store:
  - position
  - incoming direction
  - probability
  - color
  - …



Pass 2: Viewing Ray

- Find N closest photons
- Assume photons hit point of interest
- Compute average radiance





Future Directions

- Real-Time Global Illumination!!
- Interactivity
  - Viewpoint change
  - Interactive object manipulation
- Parallel Rendering