실사 렌더링
Physically based Rendering

Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui

KAIST

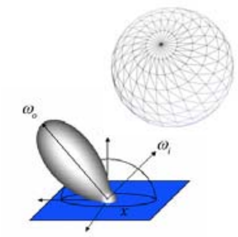
About the Instructor

- Joined KAIST at 2007
- Main research focus
  - Handling of massive geometric data for various computer graphics and geometric problems
- Research history for rendering
  - Did volume rendering at M.S.
  - Did large-scale, real-time, rasteriation based rendering at Ph.D.
  - Have been doing high-quality rendering at KAIST

KAIST

Overview

- We will discuss various parts of computer graphics



Modelling → Simulation & Rendering → Image

Computer vision inverts the process
Image processing deals with images

KAIST

Application of Rendering/Computer Graphics

- Games
- Movies and film special effects
- Product design and analysis
- Medical applications
- Scientific visualization

KAIST

Games

2D game

3D shooting game

Movies and Film Special Effects

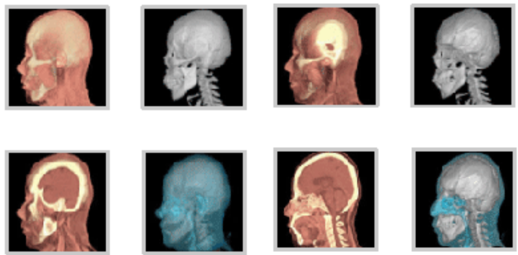
Avatar

Product Design and Analysis

- Computer-aided design (CAD)

Medical Applications

- Visualizing data of CT, MRI, etc



Rapidia homepage

About the Course

- We will focus on:
  - Photo-Realistic Rendering
  - Study basic concepts of physically-based rendering

KAIST

Photo-Realistic Rendering

- Achieved by simulating light and material interactions

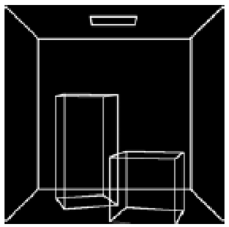
from Prof. Bala's slide

- Rendering equation
  - Mathematical formulation of light and material interactions

KAIST

Global Illumination (GI)

- GI algorithms solve the rendering equation
  - Generate 2D image from 3D scene



from Prof. Bala's slide

⇨ GI Algorithm ⇨

\+
Emission (light sources)
Geometry (objects)
BRDF (materials)

KAIST

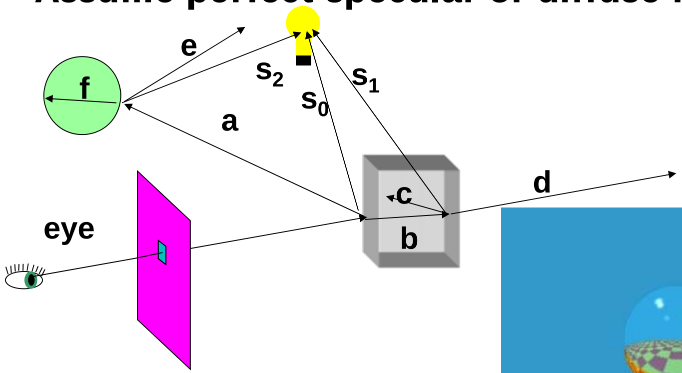
Classic Methods of GI

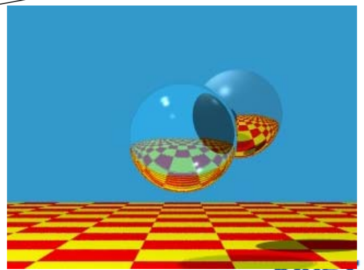
- Ray tracing
  - Introdued by Whitted in 1980
- Radiosity
  - Introduced in 1984
- Monte Carlo rendering

KAIST

Ray Tracing

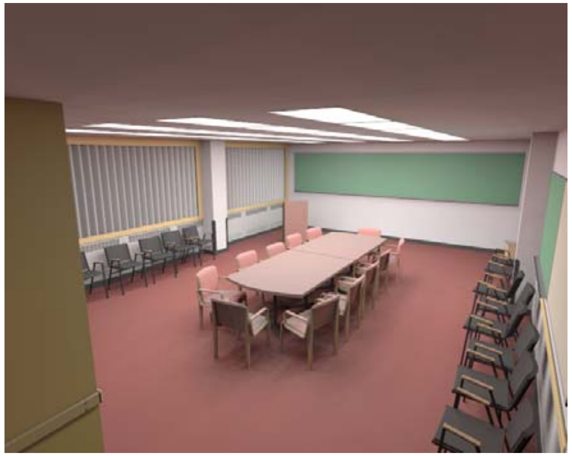
- Assume perfect specular or diffuse material





Radiosity

- Assume diffuse inter-reflections



Advanced Global Illumination

- Extend to handle more realistic materials than just perfect specular / diffuse
  - Classic ray tracing and classic radiosity are basic building blocks

from photon map paper

from Pixar movie

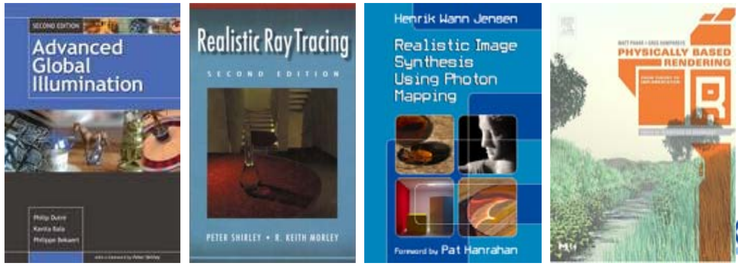
Scalable GI

- How can we handle complexity?
  - Many objects
  - Many triangles
  - Many lights
  - Complex BRDFs
  - Dynamic scenes, etc.

- Can we achieve interactive GI on commodity hardware?

Resource

- Reference
  - Physically based renderig, Matt Pharr et al.
  - Advanced Global Illumination, Philip Dutre et al. 2nd edition
  - Realistic Image Synthesis Using Photon Mapping, Henrik Jensen
  - Realistic Ray Tracing, 2nd edition, Peter Shirley et al.



Other Reference

- Technical papers
  - Graphics-related conference (SIGGRAPH, etc)
  - http://kesen.huang.googlepages.com/
- SIGGRAPH course notes and video encore
- Course homepages
- Google or Google scholar

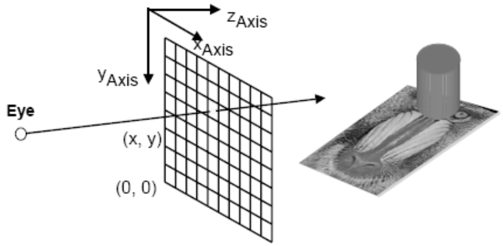
Classic Rendering Pipeline

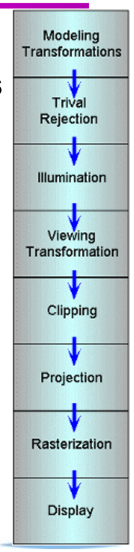
Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/GCG/

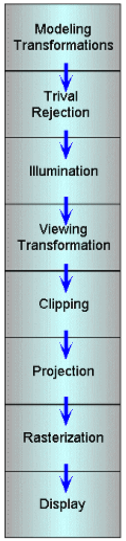
Course Objectives

- Understand classic rendering pipeline
  - Just high-level concepts, not all the details
  - Brief introduction of common under. CG
- Know its pros and cons





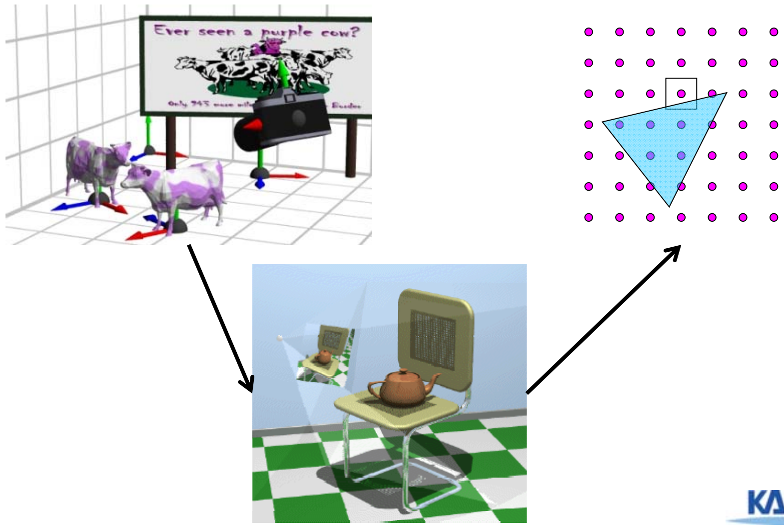
The Classic Rendering Pipeline



- Adopted in OpenGL and DirectX
  - Most of games are based on this pipeline
- Object primitives defined by vertices fed in at the top
- Pixels come out in the display at the bottom

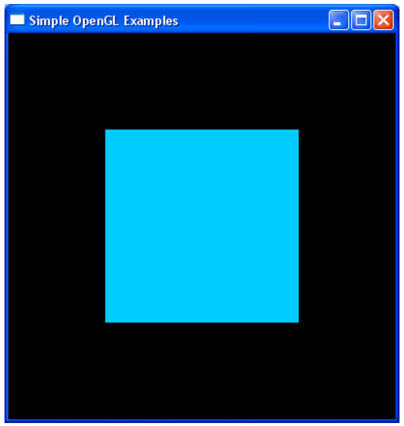
KAIST

The Classic Rendering Pipeline



KAIST

Code Example



OpenGL Code:

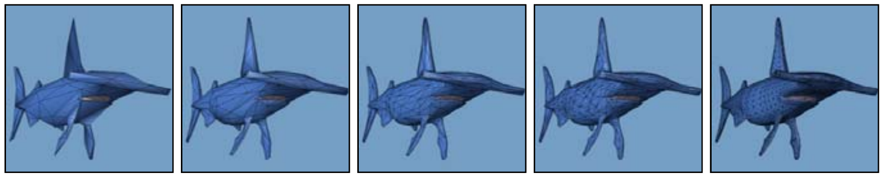
```
glColor3d(0.0, 0.8, 1.0);

glBegin(GL_POLYGON);
      glVertex2d(-0.5, -0.5);
      glVertex2d( 0.5, -0.5);
      glVertex2d( 0.5,  0.5);
      glVertex2d(-0.5,  0.5);
glEnd();
```


KAIST

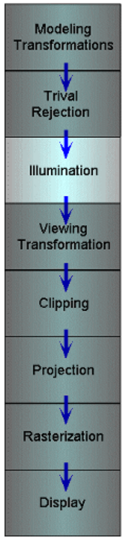
Triangle Representation, Mesh

- Triangles can approximate any 2-dimensional shape (or 3D surface)
  - Polygons are a locally linear (planar) approximation
- Improve the quality of fit by increasing the number edges or faces

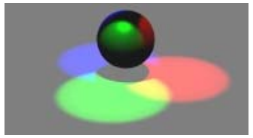


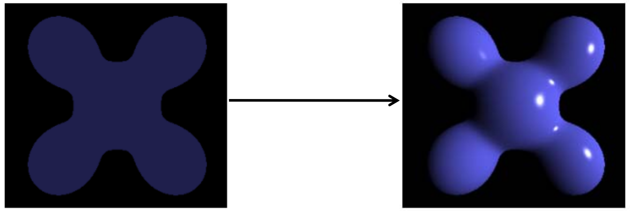
KAIST

Illumination



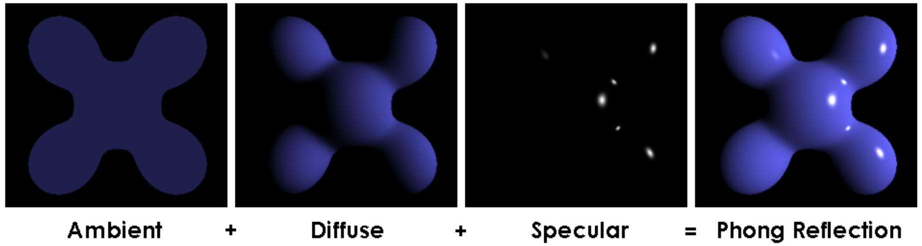
- Illuminate potentially visible objects
- Final rendered color is determined by object's orientation, its material properties, and the light sources in the scene





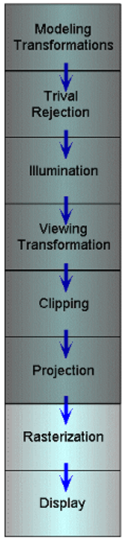
OpenGL's Illumination Model

$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j \max((\hat{N} \bullet \hat{L}_j),0) + k_s^j I_s^j \max((\hat{V} \bullet \hat{R})^{n_s},0))$$

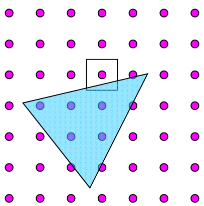


From Wikipedia

Rasterization and Display



- Transform to screen space
- Rasterization converts objects pixels

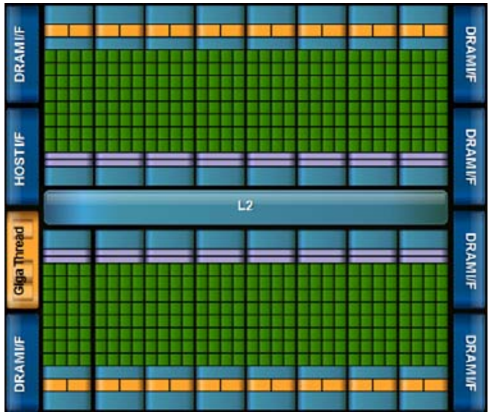


Why we are using rasterization?

- Efficiency
- Reasonably quality

Fermi GPU Architecture

16 SM (streaming processors)



512 CUDA cores

Memory interfaces

Where Rasterization Is

From Battlefield: Bad Company, EA Digital Illusions CE AB

From Eric Haines

But what about other visual cues?

- Lighting
  - Shadows
  - Shading: glossy, transparency
- Color bleeding, etc.
- Generality

from Henrik's homepage

Ray Tracing

Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui

KAIST

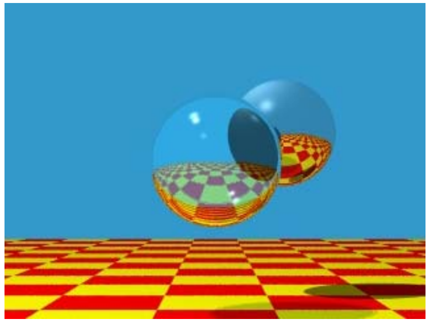
Class Objectives

- Understand a basic ray tracing
- Implement its acceleration data structure and know how to use it

KAIST

Recursive Ray Casting

- Gained popularity in when Turner Whitted (1980) recognized that *recursive* ray casting could be used for global illumination effects



KAIST

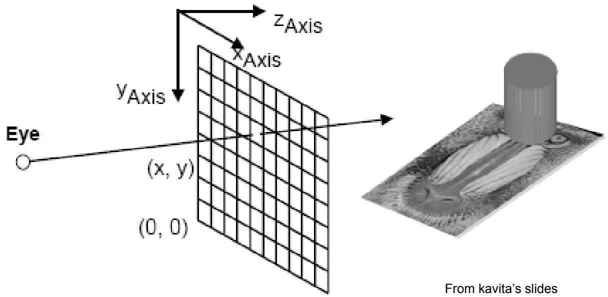
Ray Casting and Ray Tracing

- Trace rays from eye into scene
  - Backward ray tracing
- Ray casting used to compute visibility at the eye
- Perform ray tracing for arbitrary rays needed for shading
  - Reflections
  - Refraction and transparency
  - Shadows

KAIST

Basic Algorithms

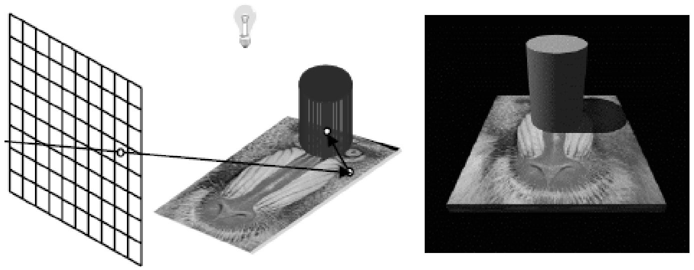
- **Rays are cast from the eye point through each pixel in the image**



From kavita's slides

Shadows

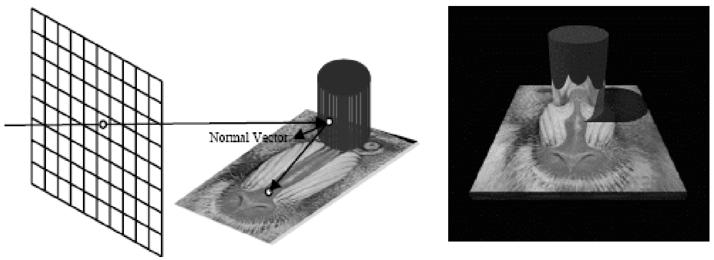
- **Cast ray from the intersection point to each light source**
  - **Shadow rays**



From kavita's slides

Reflections

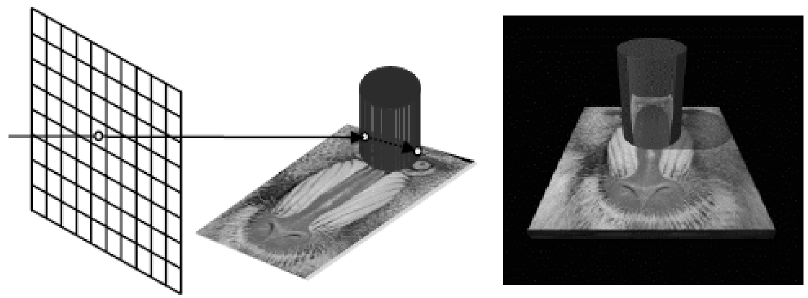
- **If object specular, cast secondary reflected rays**



From kavita's slides

Refractions

- **If object tranparent, cast secondary refracted rays**



From kavita's slides

An Improved Illumination Model [Whitted 80]

- **Phong illumination model**

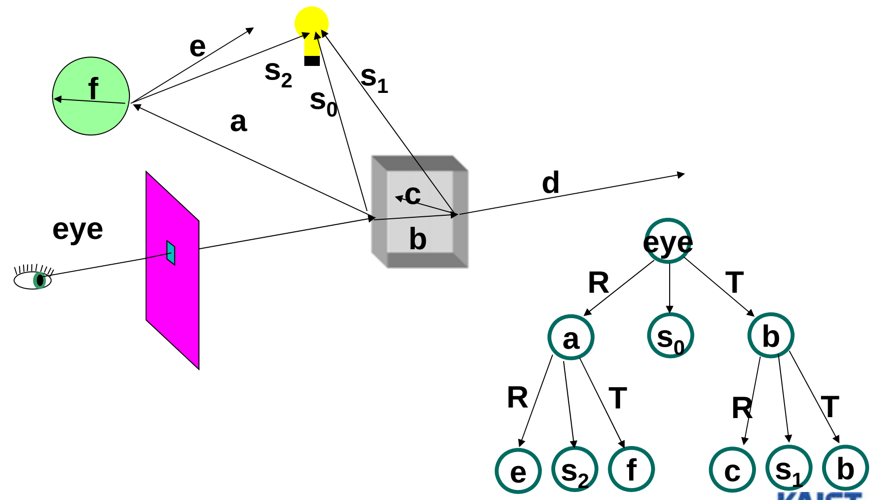
$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j (\hat{N}\bullet\hat{L}_j) + k_s^j I_s^j (\hat{V}\bullet\hat{R})^{n_s})$$

- **Whitted model**

$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j (\hat{N}\bullet\hat{L}_j)) + k_s S + k_t T$$

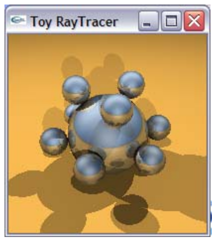
  - **S and T are intensity of light from reflection and transmission rays**
  - **Ks and Kt are specular and transmission coefficient**

Ray Tree



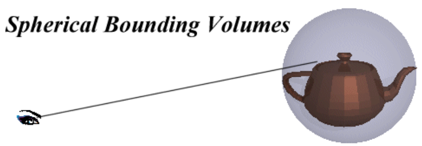
Acceleration Methods for Ray Tracing

- **Rendering time for a ray tracer depends on the number of ray intersection tests per pixel**
  - **The number of pixels X the number of primitives in the scene**
- **Early efforts focused on accelerating the ray-object intersection tests**
  - **Ray-triangle intersection tests**
- **More advanced methods required to make ray tracing practical**
  - **Bounding volume hierarchies**
  - **Spatial subdivision (e.g., kd-trees)**



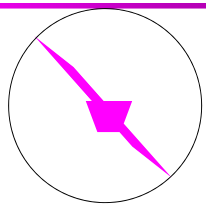
Bounding Volumes

- **Enclose complex objects within a simple-to-intersect objects**
  - **If the ray does not intersect the simple object then its contents can be ignored**
  - **The likelihood that it will strike the object depends on how tightly the volume surrounds the object.**
- **Spheres are simple, but not tight**
- **Axis-aligned bounding boxes often better**
  - **Can use nested or hierarchical bounding volumes**

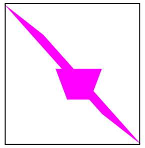


Bounding Volumes

- **Sphere [Whitted80]**
  - **Cheap to compute**
  - **Cheap test**
  - **Potentially very bad fit**

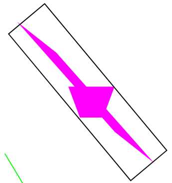


- **Axis-Aligned Bounding Box**
  - **Very cheap to compute**
  - **Cheap test**
  - **Tighter than sphere**

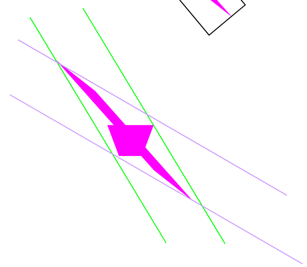


Bounding Volumes

- **Oriented Bounding Box**
  - **Fairly cheap to compute**
  - **Fairly Cheap test**
  - **Generally fairly tight**

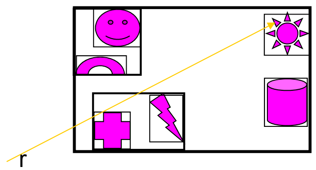


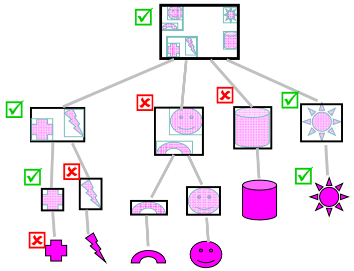
- **Slabs / K-dops**
  - **More expensive to compute**
  - **Fairly cheap test**
  - **Can be tighter than OBB**



Hierarchical Bounding Volumes

- **Organize bounding volumes as a tree**
  - **Choose a partitioning plane and distribute triangles into left and right nodes**
- **Each ray starts with the scene BV and traverses down through the hierarchy**

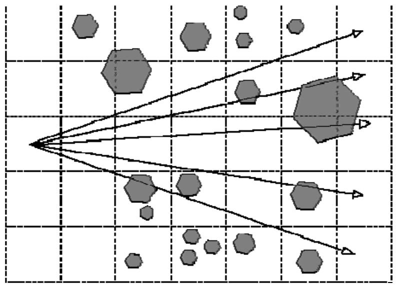




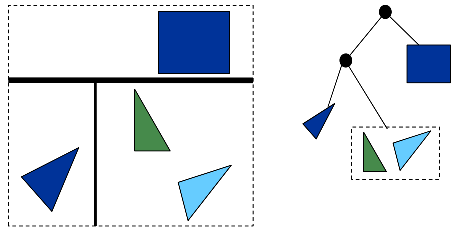
Spatial Subdivision

Idea: Divide space in to subregions

- **Grids and kd-trees are also commonly used**



Grids
Good for GPUs



kd-trees
Higher performance

Classic Ray Tracing

- Gathering approach
  - From lights, reflected, and refracted directions
- Pros of ray tracing
  - Simple and improved realism over the rendering pipeline

- Cons:
  - Simple light model, material, and light propagation
  - Not a complete solution
  - Hard to accelerate with special-purpose H/W

History

- Problems with classic ray tracing
  - Not realistic
  - View-dependent
- Radiosity (1984)
  - Global illumination in diffuse scenes
- Monte Carlo ray tracing (1986)
  - Global illumination for any environment

Class Objectives were:

- Understand a basic ray tracing
- Implement its acceleration data structure and know how to use it

PA1

- Get to know pbrt

Radiosity

Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui

KAIST

Class Objective

- Understand radiosity
  - Radiosity equation
  - Solving the equation

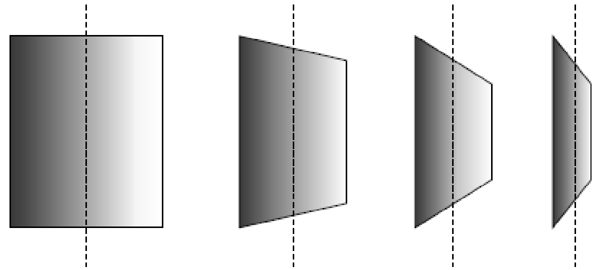
KAIST

Radiosity

- Physically based method for diffuse environments
  - Support diffuse interactions, color bleeding, indirect lighting and penumbra
  - Account for very high percentage of total energy transfer
  - Finite element method

KAIST

Key Idea #1: Diffuse Only



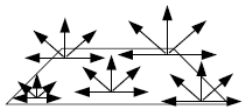
From kavita's slides

- Radiance independent of direction
  - Surface looks the same from any viewpoint
  - No specular reflection

KAIST

Diffuse Surfaces

- **Diffuse emitter**
  - $L(x \to \Theta)$ = **constant over** Θ

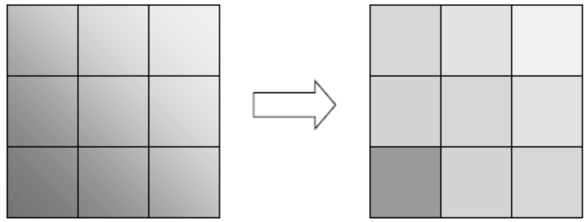


- **Diffuse reflector**
  - **Constant reflectivity**

From kavita's slides

Key Idea #2: Constant Polygons

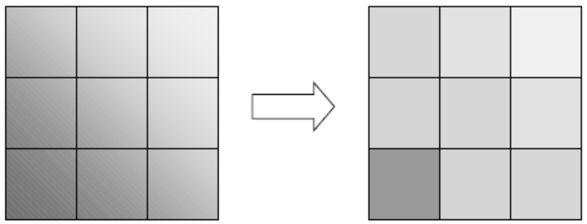
- **Radiosity is an approximation**
  - **Due to discretization of scene into patches**



From kavita's slides

  - **Subdivide scene into small polygons**

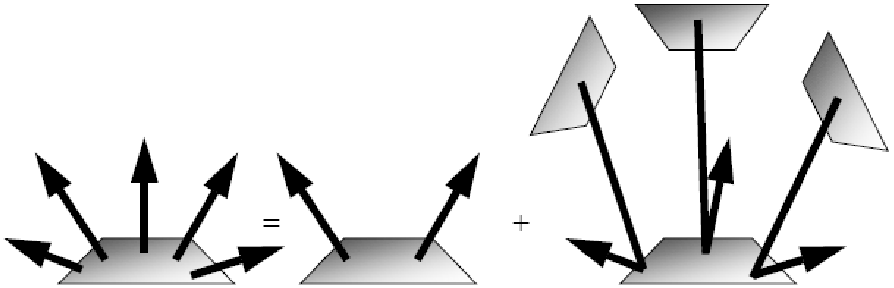
Constant Radiance Approximation



From kavita's slides

- **Radiance is constant over a surface element**
  - **L(x) = constant over x**

Radiosity Equation



Emitted radiosity = self-emitted radiosity + received & reflected radiosity

$$Radiosity_i = Radiosity_{self,i} + \sum_{j=1}^{N} a_{j\to i} Radiosity_j$$

Radiosity Equation

- Radiosity equation for each polygon i

$$Radiosity_1 = Radiosity_{self,1} + \sum_{j=1}^{N} a_{j\to 1} Radiosity_j$$

$$Radiosity_2 = Radiosity_{self,2} + \sum_{j=1}^{N} a_{j\to 2} Radiosity_j$$

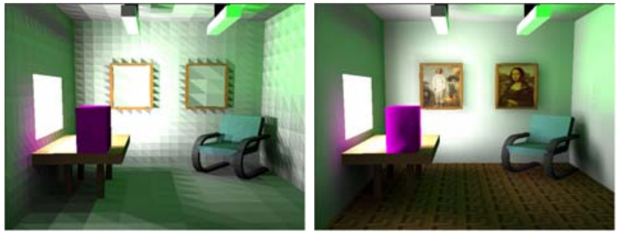
$$\dots$$

$$Radiosity_N = Radiosity_{self,N} + \sum_{j=1}^{N} a_{j\to N} Radiosity_j$$

- N equations; N unknown variables

Radiosity Algorithm

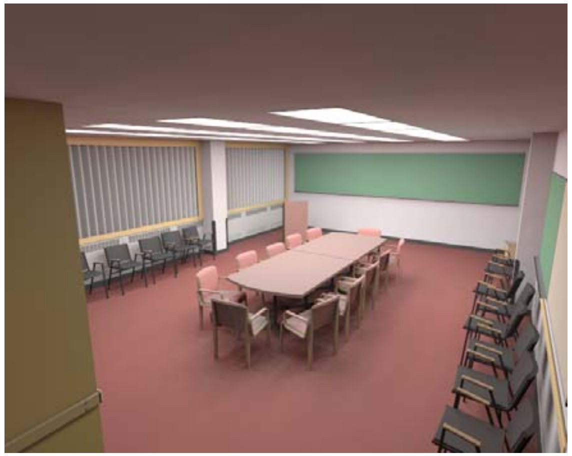
- **Maps well to rasterization pipeline**
  - **Subdivide the scene in small polygons**
  - **Compute a constant illumination value for each polygon**
  - **Choose a viewpoint and display the visible polygon**



From Donald Fong's slides

KAIST

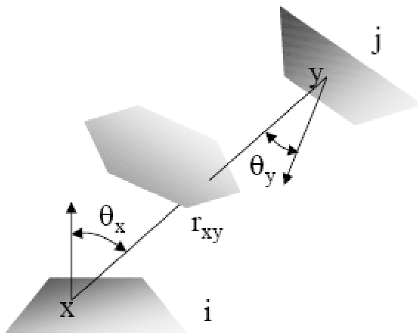
Radiosity Result



KAIST

Compute Form Factors

$$F(j \to i) = \frac{1}{A_j} \int_{A_i} \int_{A_j} \frac{\cos\theta_x \cdot \cos\theta_y}{\pi \cdot r_{xy}^2} \cdot V(x,y) \cdot dA_y \cdot dA_x$$



Radiosity Equation

- Radiosity for each polygon i

$$\forall i : B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} B_j F(i \to j)$$

- Linear system
  - B_i : radiosity of patch i (unknown)
  - $B_{e,i}$: emission of patch i (known)
  - ρ_i : reflectivity of patch i (known)
  - F(i→j): form-factor (coefficients of matrix)

Linear System of Radiosity Equations

$$\begin{bmatrix} 1-\rho_1 F_{1\to 1} & -\rho_1 F_{1\to 2} & \dots & -\rho_1 F_{1\to n} \\ -\rho_2 F_{2\to 1} & 1-\rho_2 F_{2\to 2} & \dots & -\rho_2 F_{2\to n} \\ \dots & \dots & \dots & \dots \\ -\rho_n F_{n\to 1} & -\rho_n F_{n\to 2} & \dots & 1-\rho_n F_{n\to n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{bmatrix} = \begin{bmatrix} B_{e,1} \\ B_{e,2} \\ \dots \\ B_{e,n} \end{bmatrix}$$

known — unknown — known

How to Solve Linear System

- **Matrix inversion**
  - **Takes $O(n^3)$**
- **Gather methods**
  - **Jacobi iteration**
  - **Gauss-Seidel**
- **Shooting**
  - **Southwell iteration**

Iterative Approaches

- **Jacobi iteration**
  - **Start with initial guess for energy distribution (light sources)**
  - **Update radiosity of all patches based on the previous guess**

$$B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} B_j F(i \to j)$$

new value (B_i) — old values (B_j)

  - **Repeat until converged**
- **Guass-Seidel iteration**
  - **New values used immediately**

Hybrid and Multipass Methods

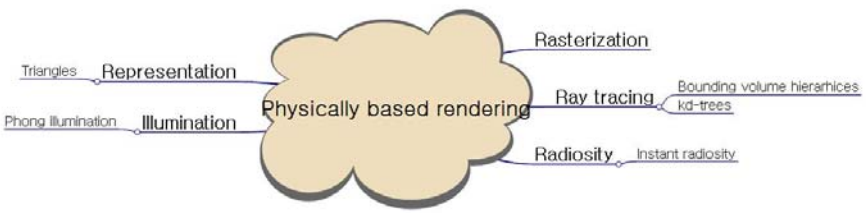
- Ray tracing
  - Good for specular and refractive indirect illumination
  - View-dependent
- Radiosity
  - Good for diffuse
  - Allows interactive rendering
  - Does not scale well for massive models
- Hybrid methods
  - Combine both of them in a way

Instant Radiosity

- Use the concept of Radiosity
- Map its functions to those of classic rendering pipeline
  - Utilize fast GPU
- Additional concepts
  - Virtual point lights
  - Shadow maps
- Micro-Rendering for Scalable, Parallel Final Gathering (Video)
  - *Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, Carsten Dachsbacher*
  - ACM Trans. Graph. 28(5) (Proc. SIGGRAPH Asia 2009), 2009.

Class Objectives were:

- Understand radiosity
  - Radiosity equation
  - Solving the equation



Radiometry and Rendering Equation

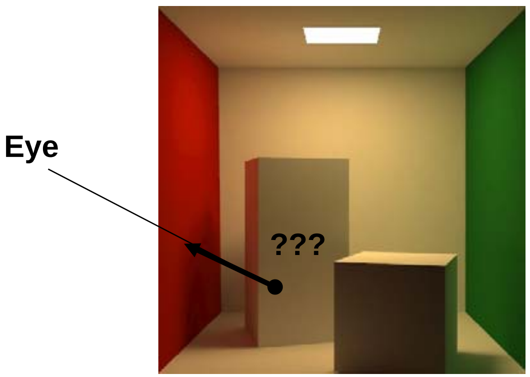
Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui

Class Objectives

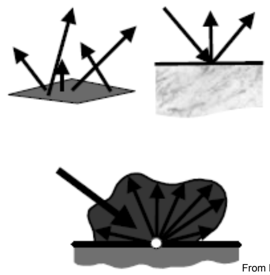
- Know terms of:
  - Hemispherical coordinates and integration
  - Various radiometric quantities (e.g., radiance)
  - Basic material function, BRDF
  - Understand the rendering equation

Motivation



Light and Material Interactions

- Physics of light
- Radiometry
- Material properties
- Rendering equation



From kavita's slides

Models of Light

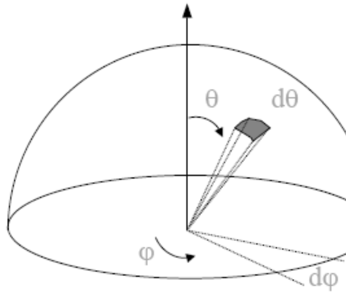
- Quantum optics
  - Fundamental model of the light
  - Explain the dual wave-particle nature of light
- Wave model
  - Simplified quantum optics
  - Explains diffraction, interference, and polarization

- Geometric optics
  - Most commonly used model in CG
  - Size of objects >> wavelength of light
  - Light is emitted, reflected, and transmitted

KAIST

Hemispheres

- Hemisphere
  - Two-dimensional surfaces
- Direction
  - Point on (unit) sphere



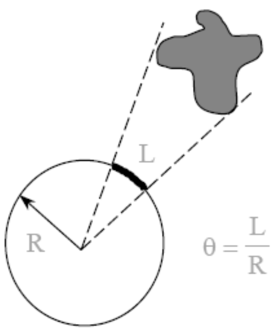
$$\theta \in [0, \frac{\pi}{2}]$$
$$\varphi \in [0, 2\pi]$$

From kavita's slides

KAIST

Solid Angles

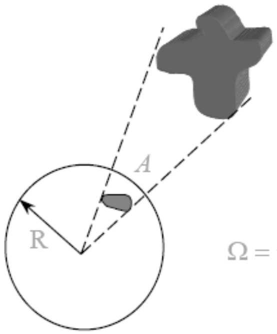
2D



$$\theta = \frac{L}{R}$$

Full circle
= 2pi radians

3D



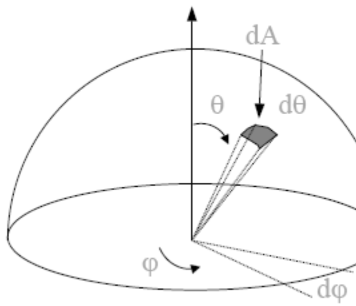
$$\Omega = \frac{A}{R^2}$$

Full sphere
= 4pi steradians

KAIST

Hemispherical Coordinates

- Direction, Θ
  - Point on (unit) sphere



$$dA = (r \sin\theta d\varphi)(r d\theta)$$

From kavita's slides

KAIST

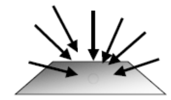
Hemispherical Coordinates

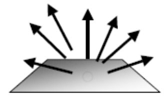
- Differential solid angle

$$d\omega = \frac{dA}{r^2} = \sin\theta d\theta d\varphi$$

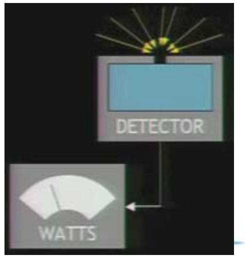
Irradiance

- Incident radiant power per unit area (dP/dA)
  - Area density of power



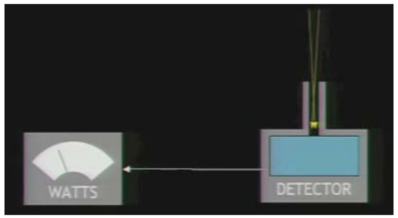


- Symbol: E, unit: W/ m^2
  - Area power density existing a surface is called radiance exitance (M) or radiosity (B)
- For example
  - A light source emitting 100 W of area 0.1 m^2
  - Its radiant exitance is 1000 W/ m^2



Radiance

- Radiant power at x in direction θ
  - $L(x \rightarrow \Theta)$: 5D function
    - Per unit area
    - Per unit solid angle

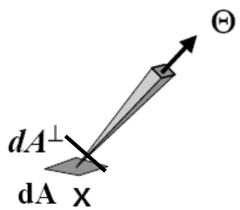


- Important quantity for rendering

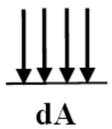
Radiance: Projected Area

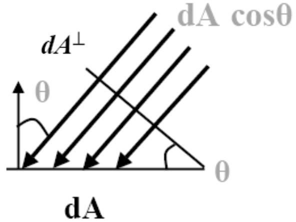
$$L(x \rightarrow \Theta) = \frac{d^2P}{dA^{\perp} d\omega_{\Theta}}$$

$$= \frac{d^2P}{d\omega_{\Theta} dA \cos\theta}$$



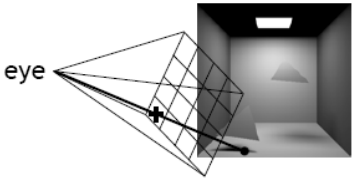
- Why per unit projected surface area





Sensitivity to Radiance

- **Responses of sensors (camera, human eye) is proportional to radiance**

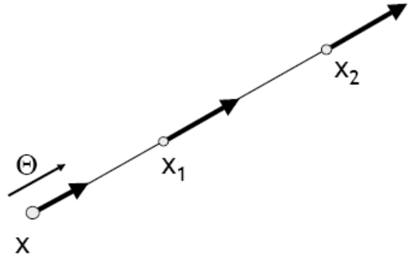


From kavita's slides

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

Properties of Radiance

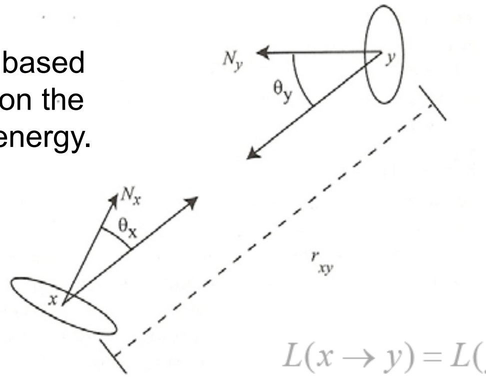
- **Invariant along a straight line (in vacuum)**



From kavita's slides

Invariance of Radiance

We can prove it based on the assumption the conservation of energy.

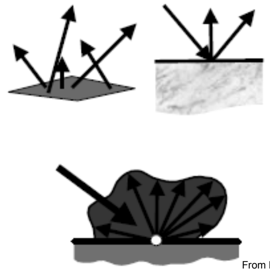


$$L(x \to y) = L(y \leftarrow x)$$

Figure 2.3. Invariance of radiance.

Light and Material Interactions

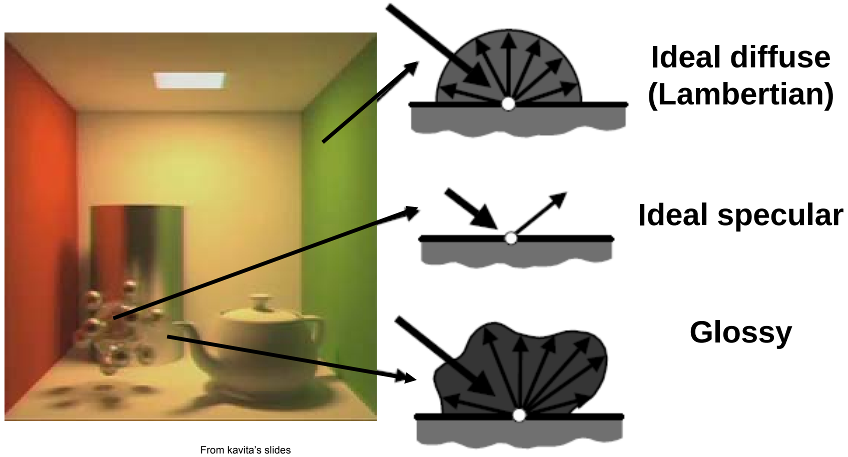
- **Physics of light**
- **Radiometry**
- **Material properties**



From kavita's slides

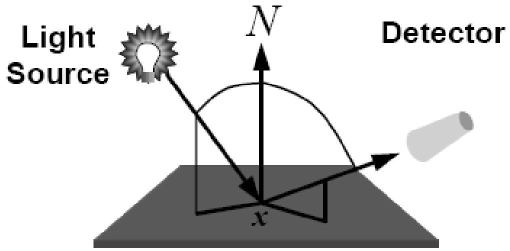
- **Rendering equation**

Materials



From kavita's slides

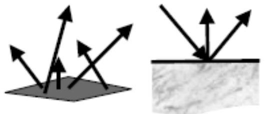
Bidirectional Reflectance Distribution Function (BRDF)



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

Light and Material Interactions

- Physics of light



- Radiometry
- Material properties

From kavita's slides

- **Rendering equation**

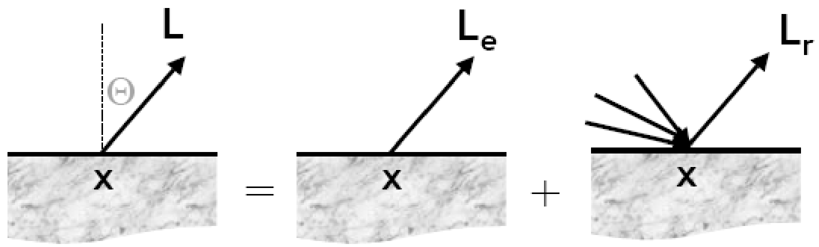
Light Transport

- Goal
  - Describe steady-state radiance distribution in the scene
- Assumptions
  - Geometric optics
  - Achieves steady state instantaneously

Rendering Equation

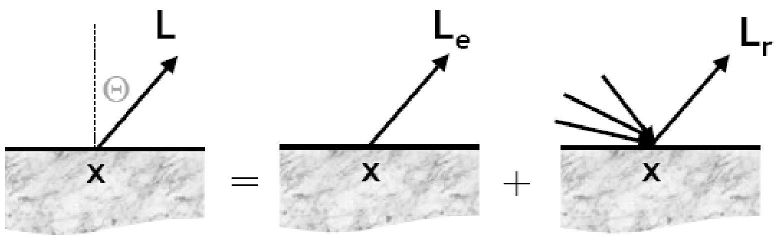
- **Describes energy transport in the scene**
- **Input**
  - **Light sources**
  - **Surface geometry**
  - **Reflectance characteristics of surfaces**
- **Output**
  - **Value of radiances at all surface points in all directions**

Rendering Equation



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

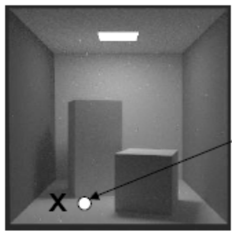
Rendering Equation



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

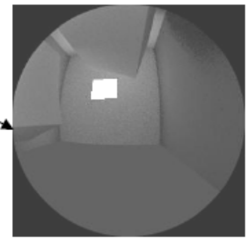
- Applicable for each wavelength

Rendering Equation



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

incoming radiance



Monte Carol Integration

Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui

KAIST

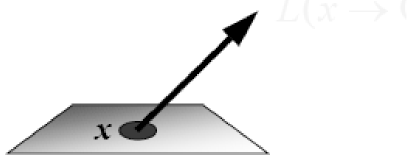
Class Objectives

- Sampling approach for solving the rendering equation
  - Monte Carlo integration
  - Estimator and its variance

KAIST

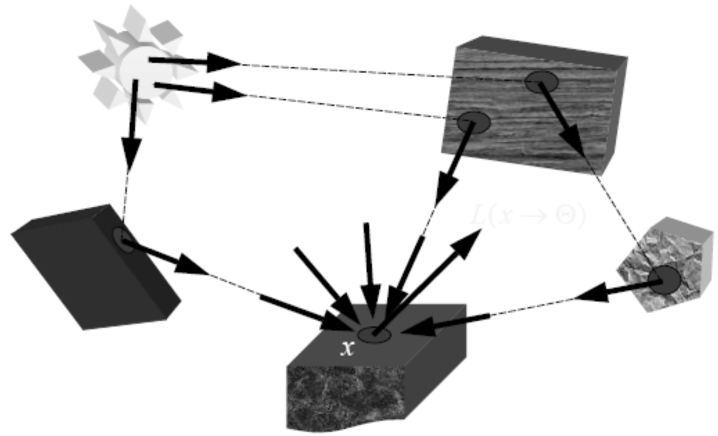
Radiance Evaluation

- Fundamental problem in GI algorithm
  - Evaluate radiance at a given surface point in a given direction
  - Invariance defines radiance everywhere else



KAIST

Radiance Evaluation



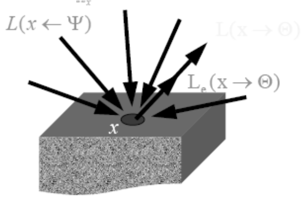
… find paths between sources and surfaces to be shaded

KAIST

Why Monte Carlo?

- **Radiace is hard to evaluate**

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



From kavita's slides

- **Sample many paths**
  - **Integrate over all incoming directions**
- **Analytical integration is difficult**
  - **Need numerical techniques**

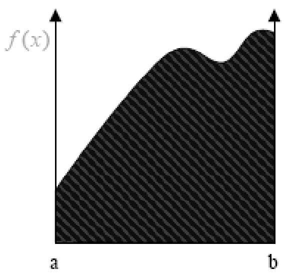
Monte Carlo Integration

- **Numerical tool to evaluate integrals**
  - **Use sampling**
- **Stochastic errors**
- **Unbiased**
  - **On average, we get the right answer**

Numerical Integration

- A one-dimensional integral:

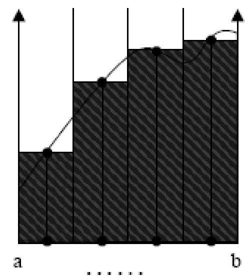
$$I = \int_a^b f(x)dx$$



Deterministic Integration

- Quadrature rules:

$$I = \int_a^b f(x)dx$$
$$\approx \sum_{i=1}^{N} w_i f(x_i)$$

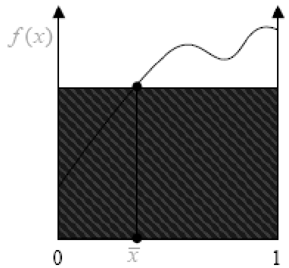


Monte Carlo Integration

Primary estimator:

$$I = \int_a^b f(x)dx$$

$$I_{prim} = f(\bar{x})$$

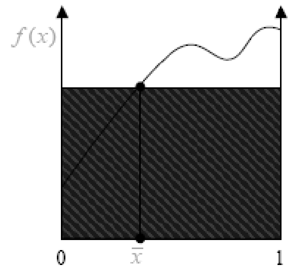


Monte Carlo Integration

Primary estimator:

$$I = \int_a^b f(x)dx$$

$$I_{prim} = f(\bar{x})$$



$$E(I_{prim}) = \int_0^1 f(x)\,p(x)dx = \int_0^1 f(x)1dx = I$$

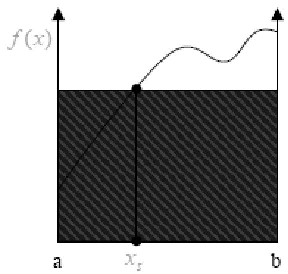
Unbiased estimator!

Monte Carlo Integration

Primary estimator:

$$I = \int_a^b f(x)dx$$

$$I_{prim} = f(x_s)(b-a)$$



$$E(I_{prim}) = \int_a^b f(x)(b-a)\,p(x)dx = \int_a^b f(x)(b-a)\frac{1}{(b-a)}dx = I$$

Unbiased estimator!

Monte Carlo Integration: Error

Variance of the estimator $\rightarrow$ a measure of the stochastic error

$$\sigma^2_{prim} = \int_a^b [\frac{f(x)}{p(x)} - I]^2\, p(x)dx$$

- **Consider p(x) for estimate**
- **We will study it as importance sampling later**

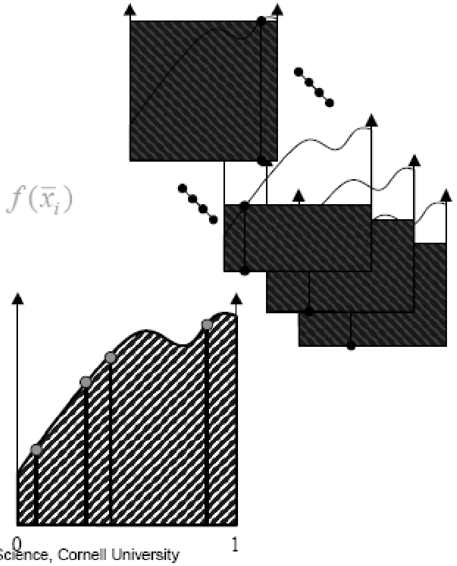
More samples

Secondary estimator

Generate N random samples x_i

Estimator: $\langle I \rangle = I_{\sec} = \frac{1}{N}\sum_{i=1}^{N} f(\bar{x}_i)$

Variance $\sigma_{\sec}^2 = \sigma_{prim}^2 / N$



Monte Carlo Integration

- Expected value of estimator

$$E[\langle I \rangle] = E[\frac{1}{N}\sum_i^N \frac{f(x_i)}{p(x_i)}] = \frac{1}{N}\int (\sum_i^N \frac{f(x_i)}{p(x_i)}) p(x)dx$$
$$= \frac{1}{N}\sum_i^N \int (\frac{f(x)}{p(x)}) p(x)dx$$
$$= \frac{N}{N}\int f(x)dx = I$$

  – on ‘average’ get right result: **unbiased**

- Standard deviation σ is a measure of the stochastic error

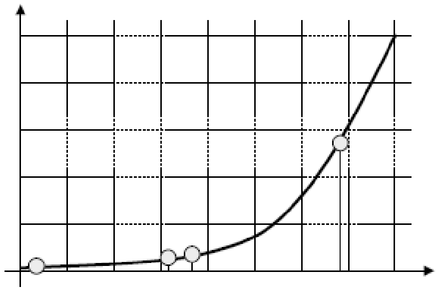
$$\sigma^2 = \frac{1}{N}\int_a^b [\frac{f(x)}{p(x)} - I]^2 p(x)dx$$

MC Integration - Example

– Integral $I = \int_0^1 5x^4 dx = 1$

– Uniform sampling

– Samples :



$x_1 = .86$ <I> = 2.74

$x_2 = .41$ <I> = 1.44

$x_3 = .02$ <I> = 0.96

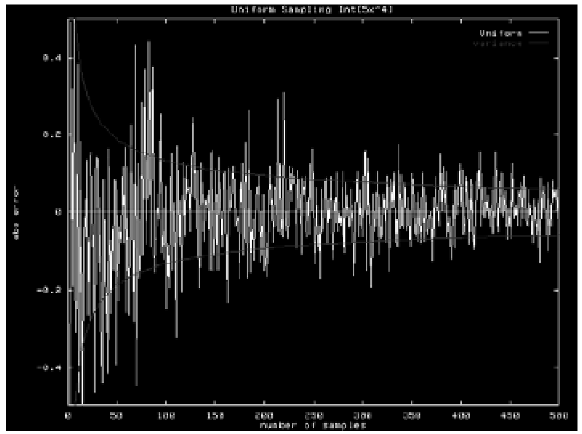
$x_4 = .38$ <I> = 0.75

MC Integration - Example

- **Integral**

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

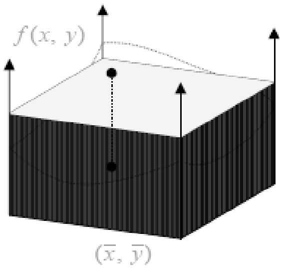
- **Variance**



MC Integration: 2D

- Primary estimator:

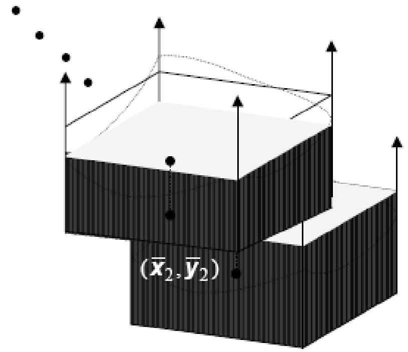
$$\bar{I}_{prim} = \frac{f(\bar{x},\bar{y})}{p(\bar{x},\bar{y})}$$



MC Integration: 2D

- Secondary estimator:

$$I_{\sec} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\bar{x}_i,\bar{y}_i)}{p(\bar{x}_i,\bar{y}_i)}$$

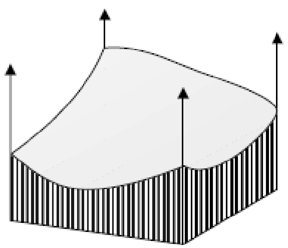


Monte Carlo Integration - 2D

- MC Integration works well for higher dimensions
- Unlike quadrature

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

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



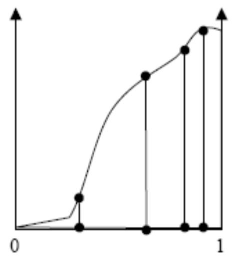
Advantages of MC

- **Convergence rate of** $O(\frac{1}{\sqrt{N}})$
- **Simple**
  - **Sampling**
  - **Point evaluation**
- **General**
  - **Works for high dimensions**
  - **Deals with discontinuities, crazy functions, etc.**

KAIST

Importance Sampling

- Take more samples in important regions, where the function is large



From kavita's slides

Class Objectives were:

- Sampling approach for solving the rendering equation
  - Monte Carlo integration
  - Estimator and its variance

Monte Carlo Ray Tracing: Part I

Sung-Eui Yoon
(윤성의)

http://sglab.kaist.ac.kr/~sungeui

KAIST

Class Objectives

- Understand a basic structure of Monte Carlo ray tracing
  - Russian roulette for its termination
  - Path tracing

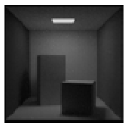
KAIST

Rendering Equation

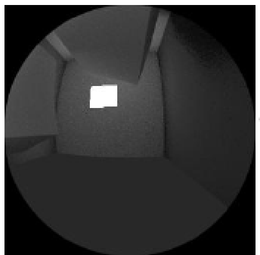
$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

function to integrate over all incoming directions over the hemisphere around x

Value we want



$= L_e + \int_{\Omega_x}$



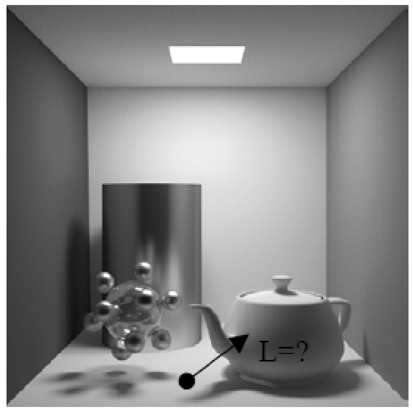
$\cdot f_r \cdot \cos$

How to compute?

L(x→Θ) = ?

Check for $L_e(x \rightarrow \Theta)$

Now add $L_r(x \rightarrow \Theta)$ =



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

How to compute?

- **Use Monte Carlo**
- **Generate random directions on hemisphere Ω_x using pdf $p(\Psi)$**

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

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\Psi_i \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\Psi_i, n_x)}{p(\Psi_i)}$$

How to compute?

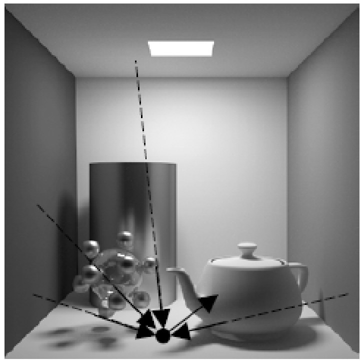
Generate random directions Ψ_i

$$\langle L \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\ldots) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\ldots)}{p(\Psi_i)}$$

- evaluate brdf
- evaluate cosine term
- evaluate $L(x \leftarrow \Psi_i)$

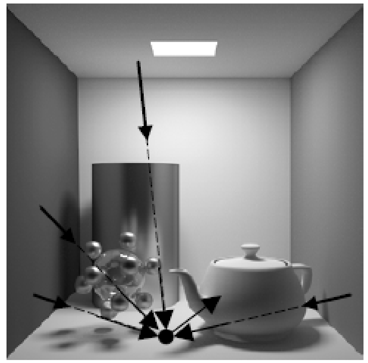
How to compute?

- evaluate $L(x \leftarrow \Psi_i)$?
- Radiance is invariant along straight paths
- $vp(x, \Psi_i)$ = first visible point
- $L(x \leftarrow \Psi_i) = L(vp(x, \Psi_i) \to \Psi_i)$

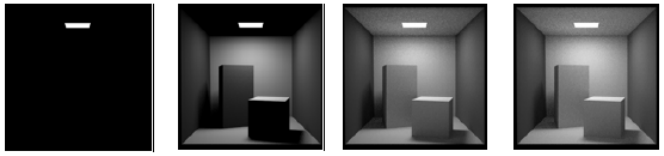


How to compute? Recursion ...

- Recursion ….
- Each additional bounce adds one more level of indirect light
- Handles ALL light transport
- “Stochastic Ray Tracing”



When to end recursion?



From kavita's slides

- Contributions of further light bounces become less significant
  - Max recursion
  - Some threshold for radiance value
- If we just ignore them, estimators will be biased

KAIST

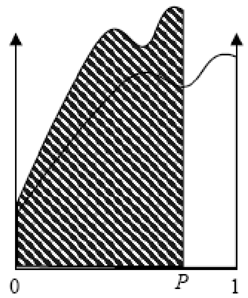
Russian Roulette

Integral

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

Estimator

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



Variance $\sigma_{roulette} > \sigma$

Russian Roulette

- Pick absorption probability, α = 1-P
  - Recursion is terminated
- 1- α is commonly to be equal to the reflectance of the material of the surface
  - Darker surface absorbs more paths

KAIST

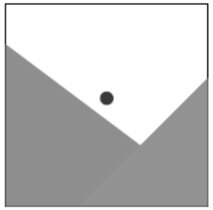
Algorithm so far

- Shoot primary rays through each pixel
- Shoot indirect rays, sampled over hemisphere
- Terminate recursion using Russian Roulette

KAIST

Pixel Anti-Aliasing

- **Compute radiance only at the center of pixel**
  - **Produce jaggies**
- **Simple box filter**
  - **The averaging method**

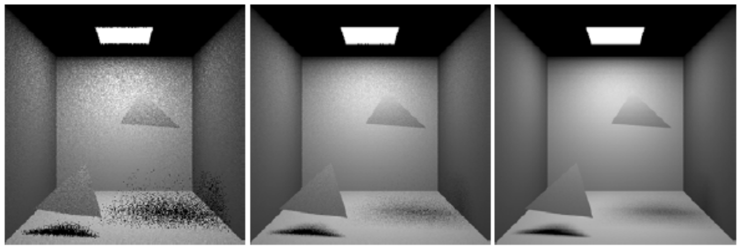


- **We want to evaluate using MC**

Stochastic Ray Tracing

- **Parameters**
  - **Num. of starting ray per pixel**
  - **Num. of random rays for each surface point (branching factor)**
- **Path tracing**
  - **Branching factor = 1**

Path Tracing



1 ray / pixel | 10 rays / pixel | 100 rays / pixel

From kavita's slides

- **Pixel sampling + light source sampling folded into one method**

Algorithm so far

- **Shoot primary rays through each pixel**
- **Shoot indirect rays, sampled over hemisphere**
  - **Path tracing shoots only 1 indirect ray**
- **Terminate recursion using Russian Roulette**

Performance

- Want better quality with smaller # of samples
  - Fewer samples/better performance
  - Quasi Monte Carlo: well-distributed samples
  - Adaptive sampling

PA2

Uniform sampling (64 samples per pixel) | Adaptive sampling | Reference

Class Objectives were:

- Understand a basic structure of Monte Carlo ray tracing
  - Russian roulette for its termination
  - Path tracing

MC Ray Tracing:
Part II, Acceleration and Biased Tech.

Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/GCG

Class Objectives:

- Extensions to the basic MC path tracer
  - Bidirectional path tracer
  - Metropolis sampling
- Biased techniques
  - Irradiance caching
  - Photon mapping

General GI Algorithm

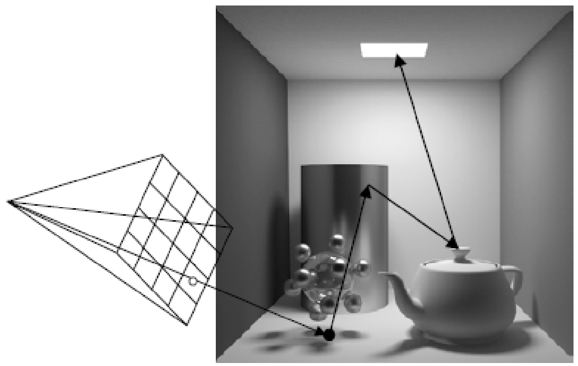
- Design path generators
- Path generators determine efficiency of GI algorithm
- Black boxes
  - Evaluate BRDF, ray intersection, visibility evaluations, etc

Other Rendering Techniques

- Bidirectional path tracing
- Metropolis
- Biased techniques
  - Irradiance caching
  - Photon mapping

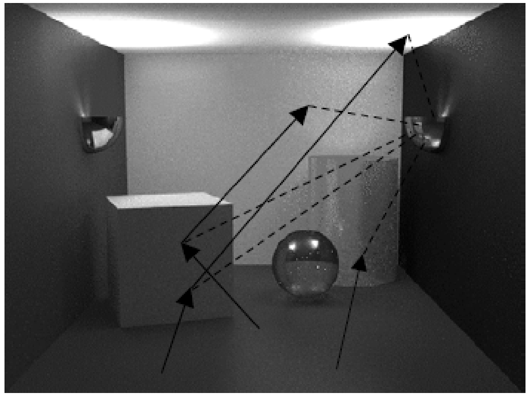
Stochastic ray tracing: limitations

- Generate a path from the eye to the light source



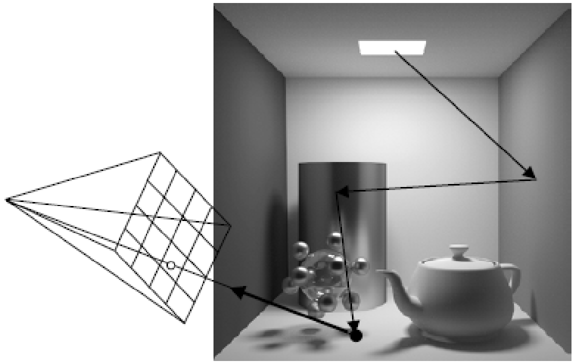
When does it not work?

- Scenes in which indirect lighting dominates



Bidirectional Path Tracing

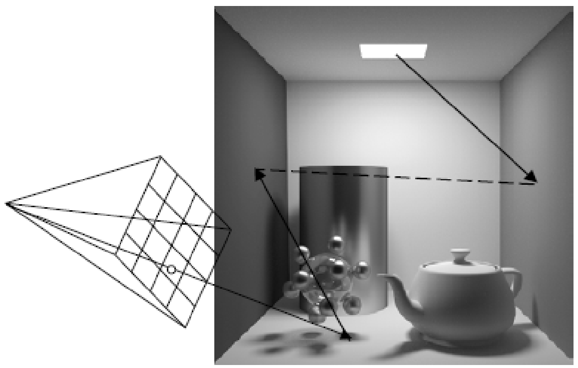
- So … we can generate paths starting from the light sources!



- Shoot ray to camera to see what pixels get contributions

Bidirectional Path Tracing

- Or paths generated from both camera and source at the same time …!



- Connect endpoints to compute final contribution

Unbiased vs. Consistent

- **Unbiased**
  - **No systematic error**
  - **$E[I_{estimator}] = I$**
  - **Better results with larger N**
- **Consistent**
  - **Converges to correct results with more samples**
  - **$E[I_{estimator}] = I + \varepsilon$, where $\lim_{n \to \infty} \varepsilon = 0$**

Biased Methods

- MC methods
  - Too noisy and slow
  - Nose is objectionable
- Biased methods: store information (caching)
  - Irradiance caching
  - Photon mapping

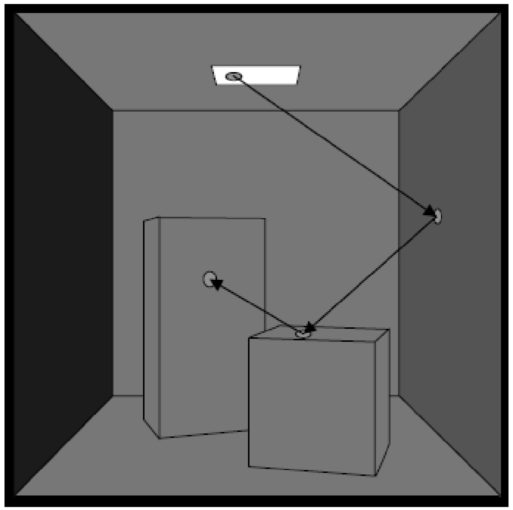
KAIST

Photon Mapping

- 2 passes:
  - Shoot "photons" (light-rays) and record any hit-points
  - Shoot viewing rays and collect information from stored photons

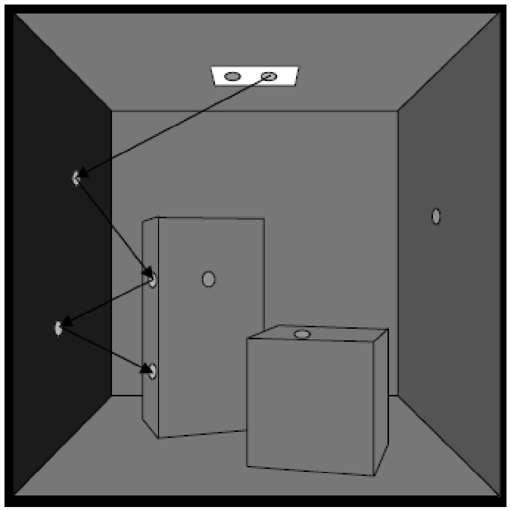
KAIST

Pass 1: shoot photons



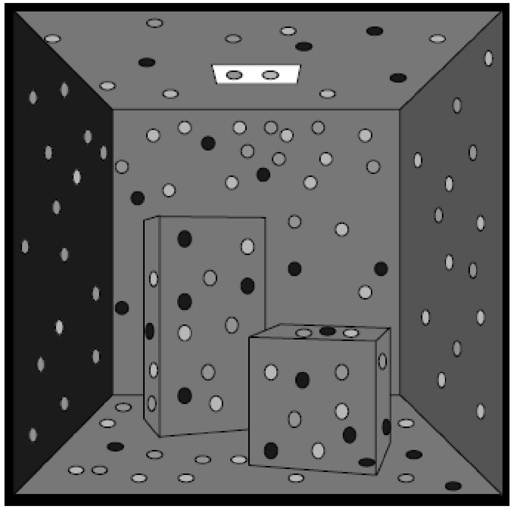
- Light path generated using MC techniques and Russian Roulette
- Store:
  - position
  - incoming direction
  - color
  - ...

Pass 1: shoot photons



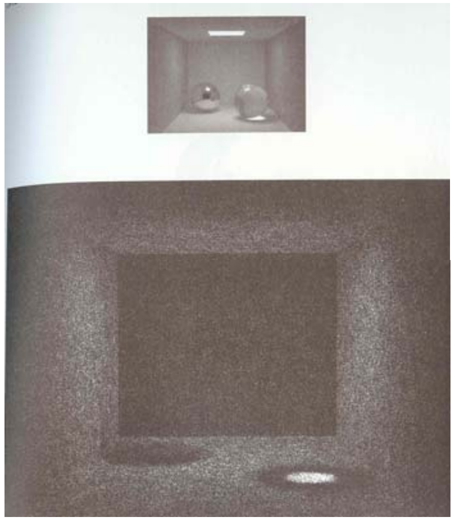
- Light path generated using MC techniques and Russian Roulette
- Store: **Flux for each photon**
  - position
  - incoming direction
  - color
  - ...

Pass 1: shoot photons



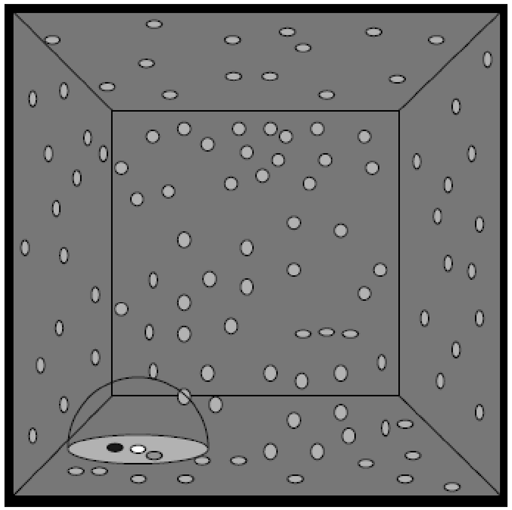
- Light path generated using MC techniques and Russian Roulette
- Store: **for diffuse materials**
  - position
  - incoming direction
  - color
  - ...

Stored Photons



Generate a few hundreds of thousands of photons

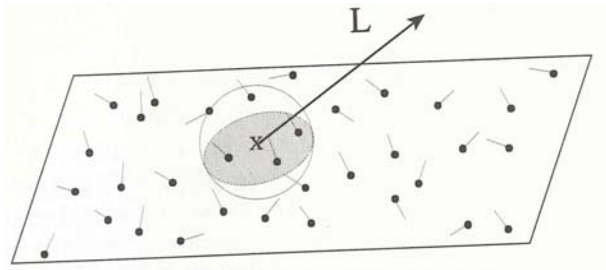
Pass 2: viewing ray



- Search for N closest photons (+check normal)
- Assume these photons hit the point we're interested in
- Compute average radiance

Radiance Estimation

- **Compute N nearest photons**
  - **Consider a few hundreds of photons**
  - **Compute the radiance for each photon to outgoing direction**
  - **Consider BRDF and**
  - **Divided by area**

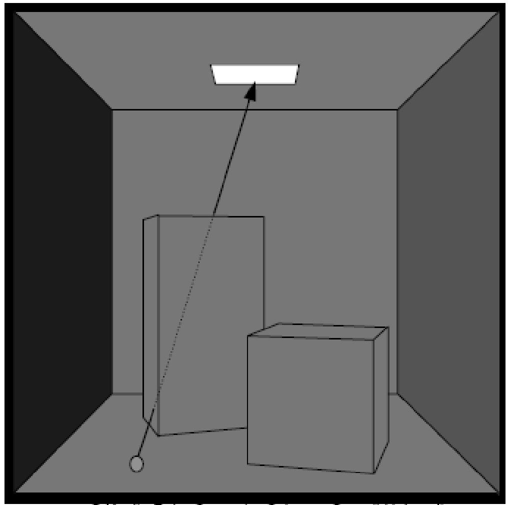


Efficiency

- Want k nearest photons
  - Use kd-tree
- Using photon maps as it create noisy images
  - Need extremely large amount of photons

KAIST

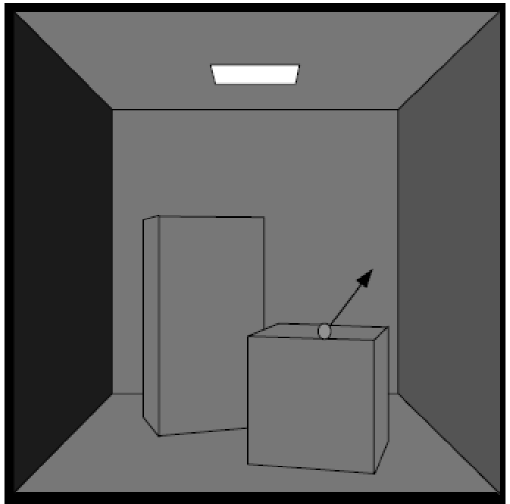
Pass 2: Direct Illumination



© Kavita Bala, Computer Science, Cornell University

Perform direct illumination for visible surface using regular MC sampling

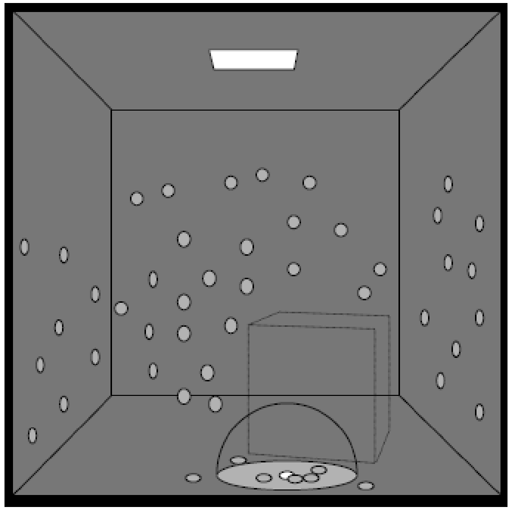
Pass 2: Specular reflections



© Kavita Bala, Computer Science, Cornell University

Specular reflection and transmission are ray traced

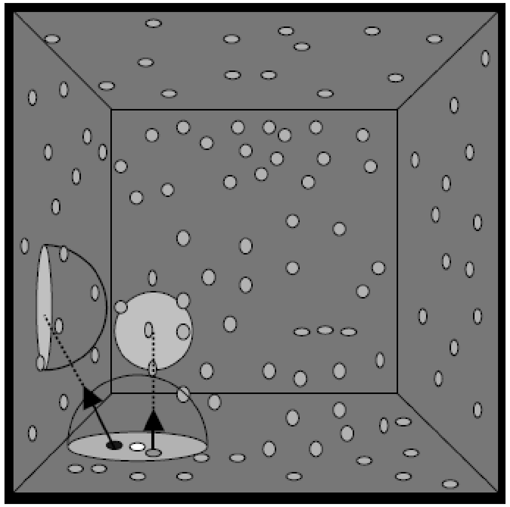
Pass 2: Caustics



© Kavita Bala, Computer Science, Cornell University

- Direct use of "caustic" maps
- The "caustic" map is similar to a photon map but treats LS*D path
- Density of photons in caustic map usually high enough to use as is

Pass 2:Indirect Diffuse



- Search for N closest photons
- Assume these photons hit the point
- Compute average radiance by importance sampling of hemisphere

Result

350K photons for the caustic map

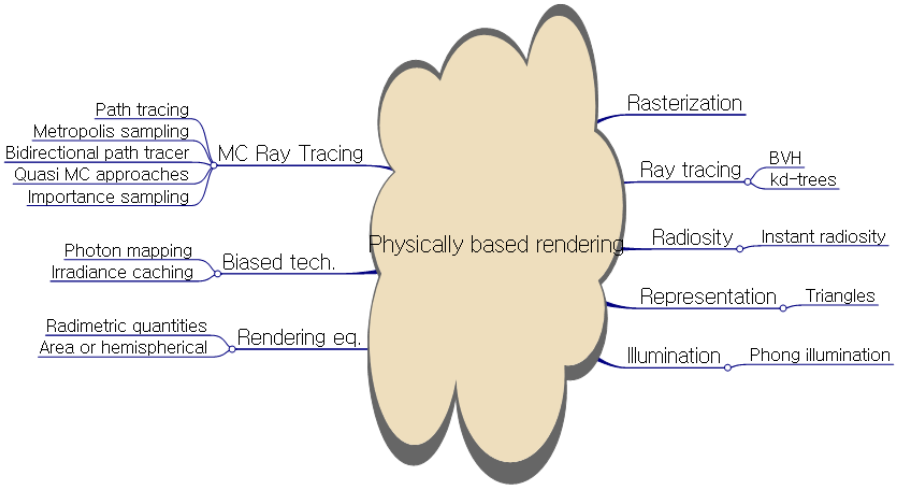
Class Objectives were:

- Extensions to the basic MC path tracer
  - Bidirectional path tracer
- Biased techniques
  - Photon mapping

Summary

- Two basic building blocks
- Radiometry
- Rendering equation
- MC integration
- MC ray tracing
  - Unbiased methods
  - Biased methods

Summary



Physically based rendering
MC Ray Tracing
Path tracing
Metropolis sampling
Bidirectional path tracer
Quasi MC approaches
Importance sampling
Biased tech.
Photon mapping
Irradiance caching
Rendering eq.
Radimetric quantities
Area or hemispherical
Rasterization
Ray tracing
BVH
kd-trees
Radiosity
Instant radiosity
Representation
Triangles
Illumination
Phong illumination

Scalable Graphics Algorithms

Sung-eui Yoon

Associate Professor

KAIST

http://sglab.kaist.ac.kr

KAIST

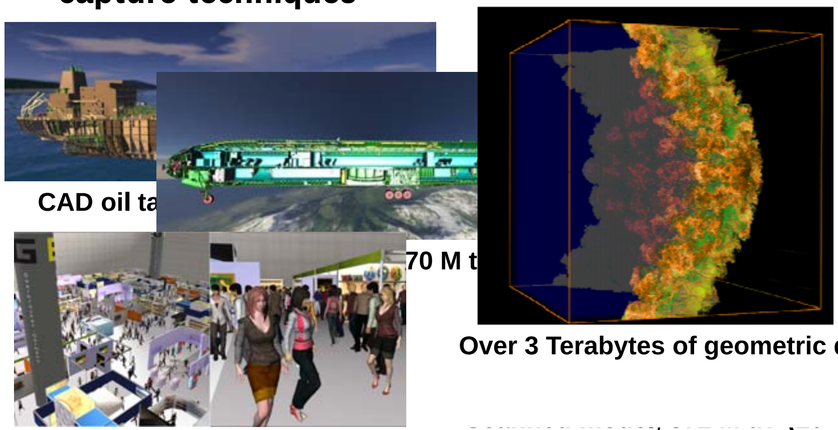
Acknowledgements

- Collaborators
  - My students, M. Gopi, Miguel Otaduy, George Drettakis, SeungYoung Lee, YuWing Tai, John Kim, Dinesh Manocha, Peter Lindstrom, Yong Joon Lee, Pierre-Yves Laffont, Jeong Mo Hong, Sun Xin, Nathan Carr, Zhe Lin
- Funding sources
  - Korea Research Foundation
  - Ministry of Knowledge Economy
  - Samsung, Microsoft Research Asia, Adobe, Boeing

KAIST

Massive Geometric Data

- Due to advances of modeling, simulation, and data capture techniques



CAD oil ta

70 M t

Over 3 Terabytes of geometric data

Large-scale virtual world, 83 M tri.

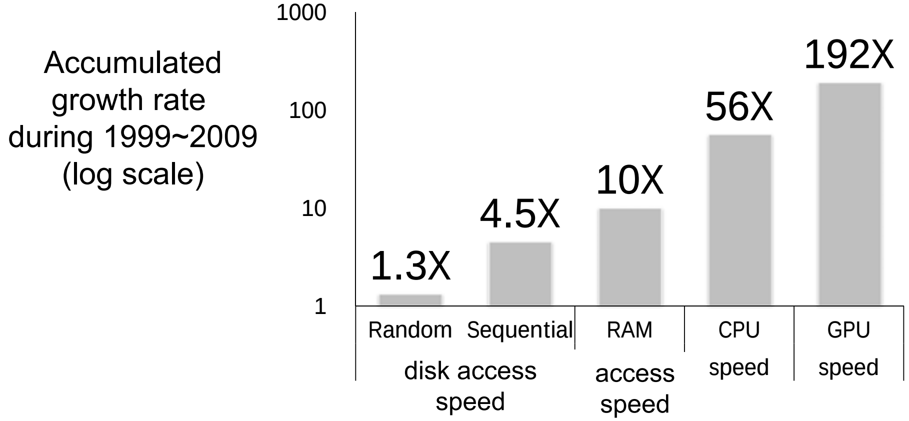
KAIST

Possible Solutions?

- Hardware improvement will address the data avalanche?
  - Moore's law: the number of transistor is roughly double every 18 months

KAIST

Current Architecture Trends



Data access time becomes the major computational bottleneck!

KAIST

Data Growth

- **An observation**
  - **If we got higher performance, we attempt to produce bigger data to derive more useful information and handle such bigger data**
- **Amount of data is doubling every 18 ~ 24 months**
  - ***"How Much Information," 2003, Lyman, Peter and Hal R.***

KAIST

Main Research Theme

- **Designing *scalable graphics and geometric algorithms* to efficiently handle massive models on commodity hardware**

•Multi-resolution methods
•Cache-coherent algorithms
•Culling techniques
•Data compression
•Parallel computation,
•Reducing data dimensions, etc

•Rendering
•Collision detection and path planning
•Image retrieval

Proximity queries

KAIST

Rendering in 2002

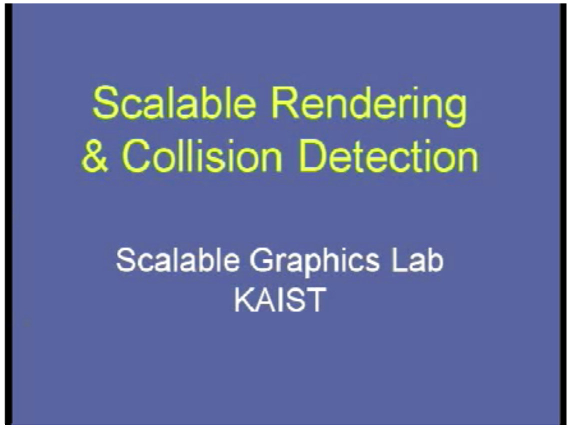
Running on SGI Reality Monster

SGI Reality Monster (consisting of 40 processing units)

KAIST

Rendering Today

- Applied various algorithms that we designed for rendering and collision detection



Cache-Oblivious Ray Reordering [Moon et al., ToG 2010]

Adaptive Rendering based on Weighted Local Regression [Moon et al. ToG 14]

- Denoise Monte Carlo ray tracing results, and generate more rays on regions with higher reconstruction error
  - Uses a novel image-space reconstruction method
  - Derives a robust error estimation process
  - Uses it to drive our sampling process

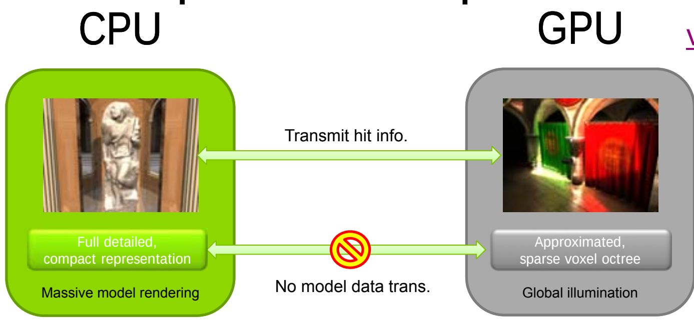
(a) Ours 106 spp (879 s) MSE 0.00446 | (b) MC 128 spp (886 s) MSE 0.06308 | (c) NLM 108 spp (892 s) MSE 0.01358 | (d) SURE 103 spp (888 s) MSE 0.01621 | (e) Ours 106 spp (879 s) MSE 0.00446 | (f) Reference 16K spp

Video

T-ReX: Interactive Global Illumination of Massive Models on Heterogeneous Computing Resources

- Use compact, decoupled representations for CPUs and GPUs [Kim et al., TVCG 14]
  - Use HCCMesh, random-accessible compressed mesh and BVHs, at CPU
  - Use a compact volumetric rep. at GPU

video



Codes are available

- http://sglab.kaist.ac.kr/software.htm
- T-Rex is available
- Adaptive sampling and reconstruction will be available
  - Will be discussed tomorrow at 신진연구자 세션