CS380: Monte Carlo Ray Tracing:

Sung-Eui Yoon
(윤성의)

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

KAIST

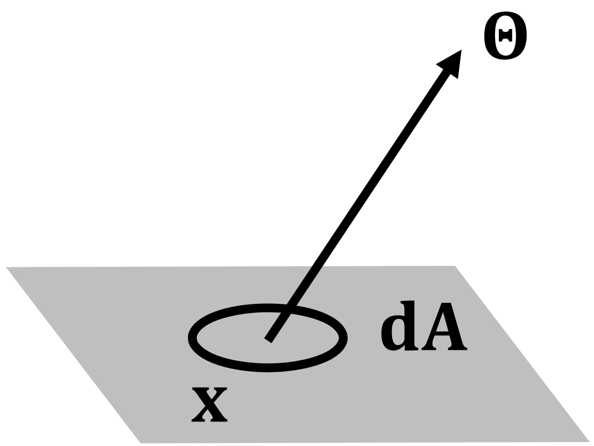
Class Objectives (Ch. 14 and 15)

- **Last time:**
  - **Rendering equation**

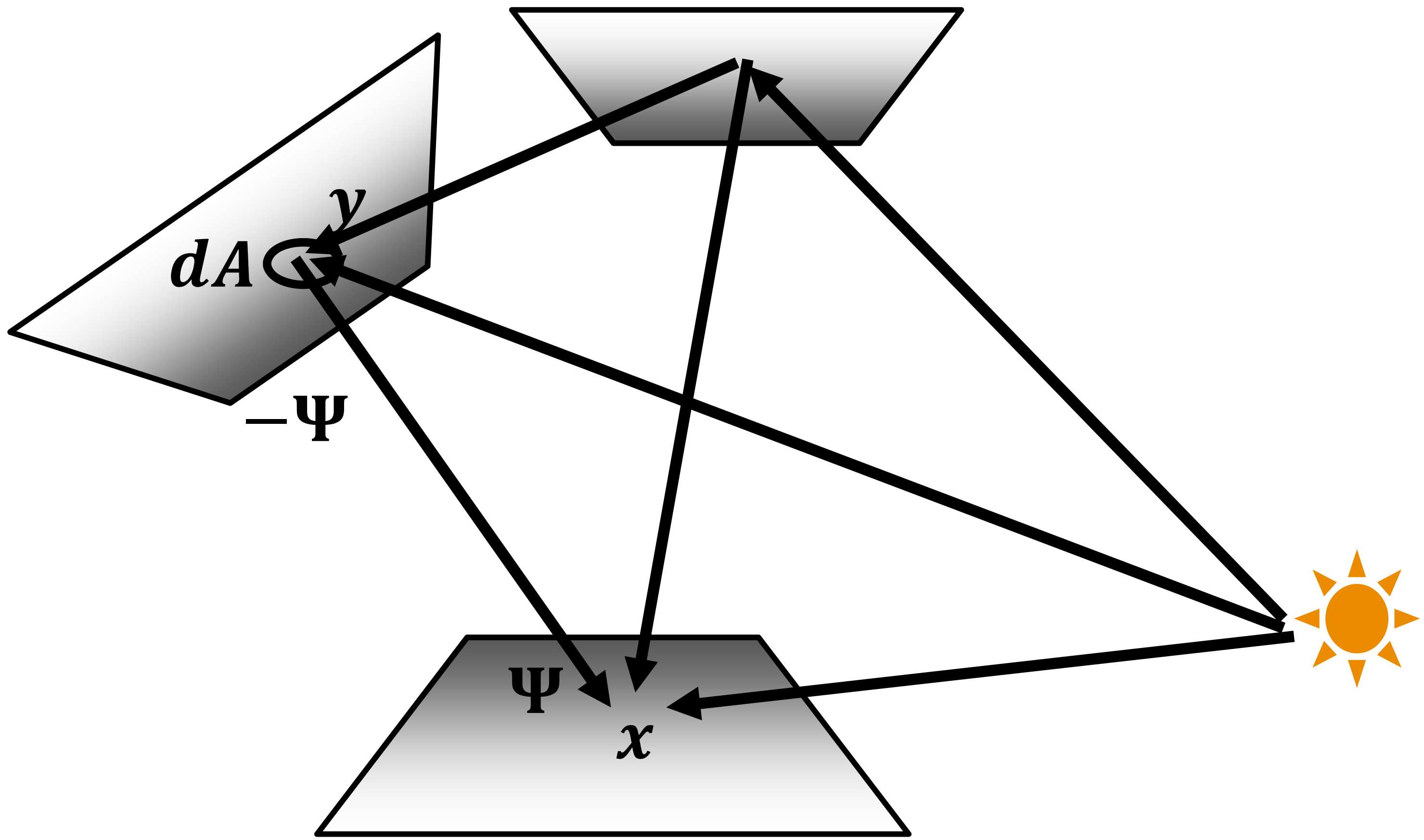
- **Sampling approach for solving the rendering equation**
  - **Monte Carlo integration**
- **Understand a basic structure of Monte Carlo ray tracing**
  - **Russian roulette for its termination**
  - **Path tracing**
- **Book:**
  - **https://sgvr.kaist.ac.kr/~sungeui/render/**

Radiance Evaluation

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



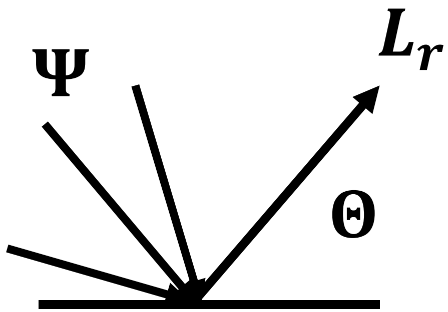
We need to find many paths…



Why Monte Carlo?

- **Radiance is hard to evaluate**

$$L_r(x \to \Theta) = \int_{\Psi} L(x \leftarrow \Psi) f_r(x, \Psi \to \Theta) \cos\theta_x dw_{\Psi},$$



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

Monte Carlo Integration

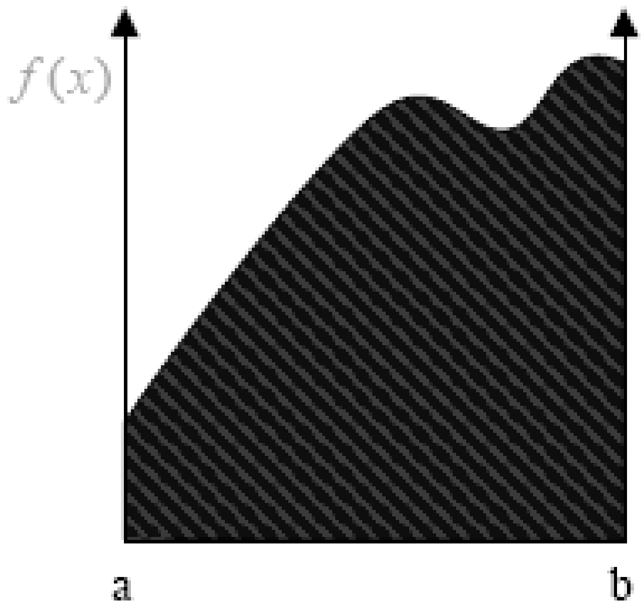
- **Numerical tool to evaluate integrals**
  - **Use sampling**
- **Stochastic errors → variance**
- **Unbiased**
  - **On average, we get the right answer**

- **We will skip theoretical analysis in this class**

Numerical Integration

- A one-dimensional integral:

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

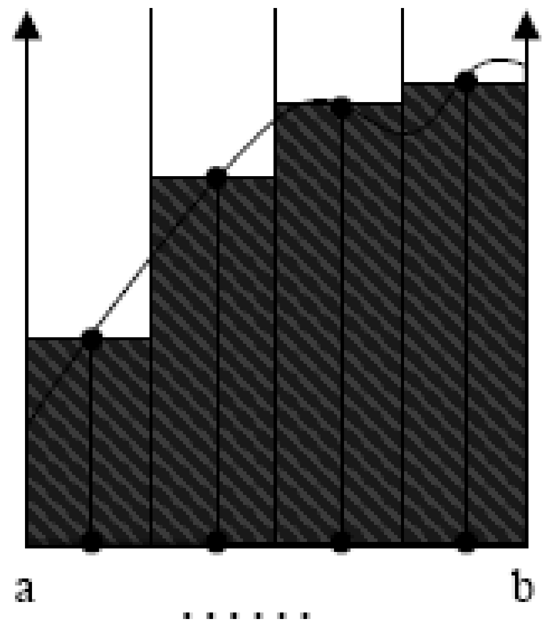


Deterministic Integration

- Quadrature rules:

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

$$\approx \sum_{i=1}^{N} w_i f(x_i)$$

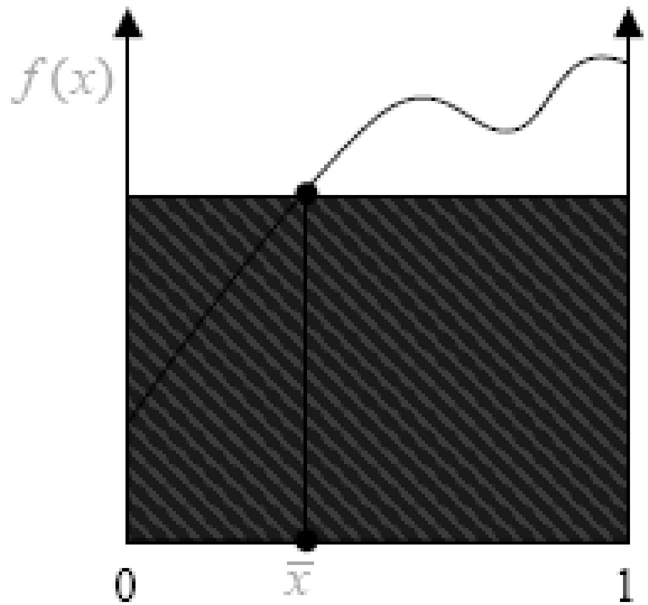


Monte Carlo Integration

Primary estimator:

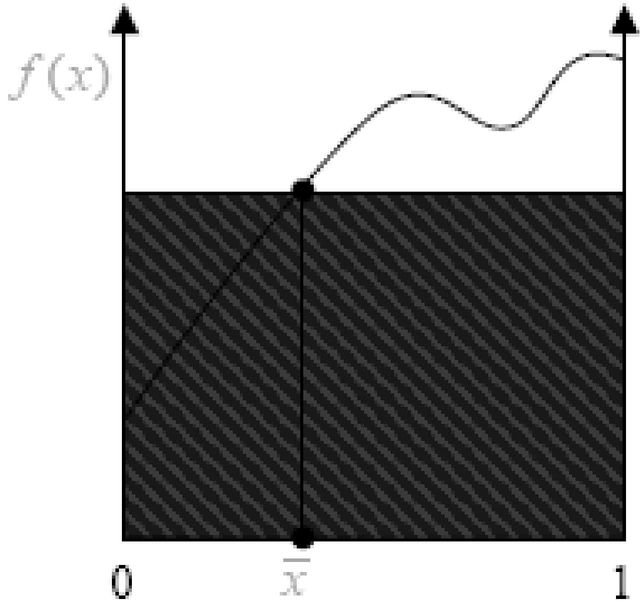
$$I = \int_a^b f(x)dx$$

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



Monte Carlo Integration

Primary estimator:



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

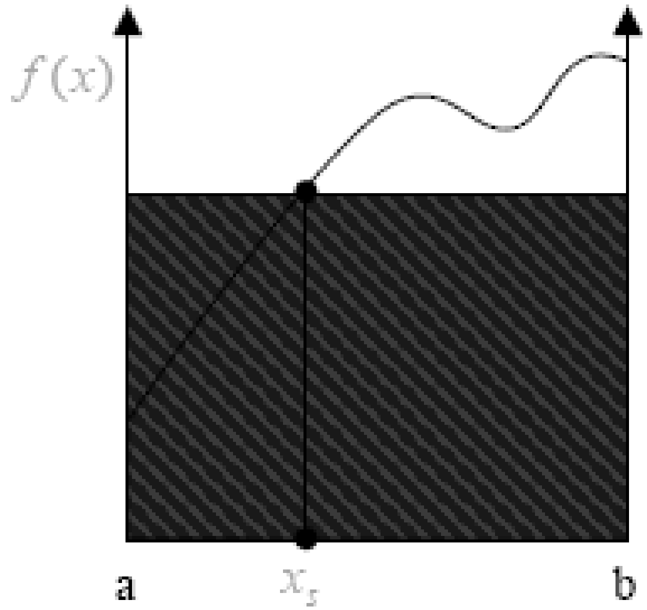
$$I_{prim} = f(\bar{x})$$

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

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 → a measure of the stochastic error

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

- Consider p(x) for estimate
- Importance sampling can be used, but skipped in this course

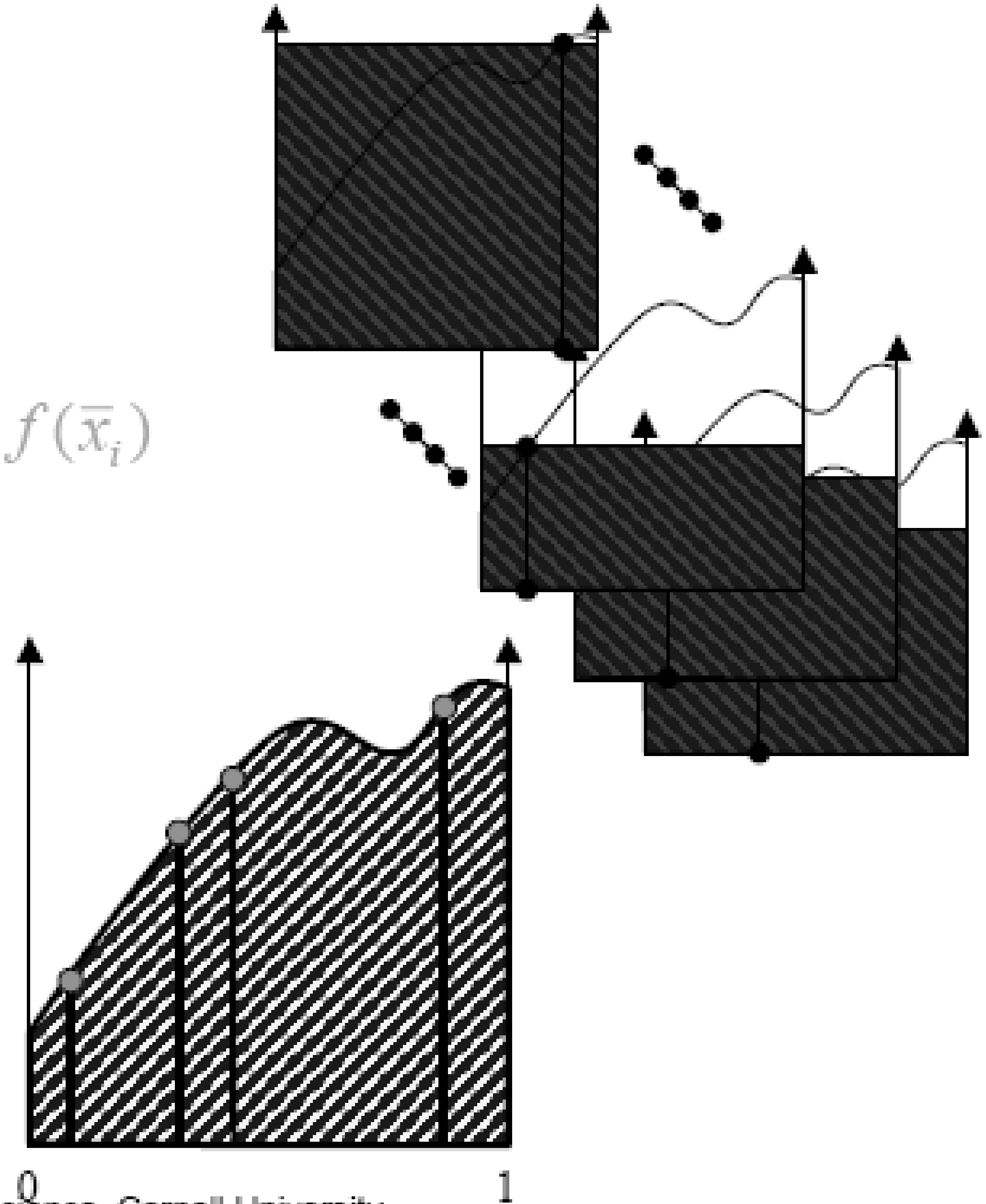
More samples

Secondary estimator

Generate N random samples x_i

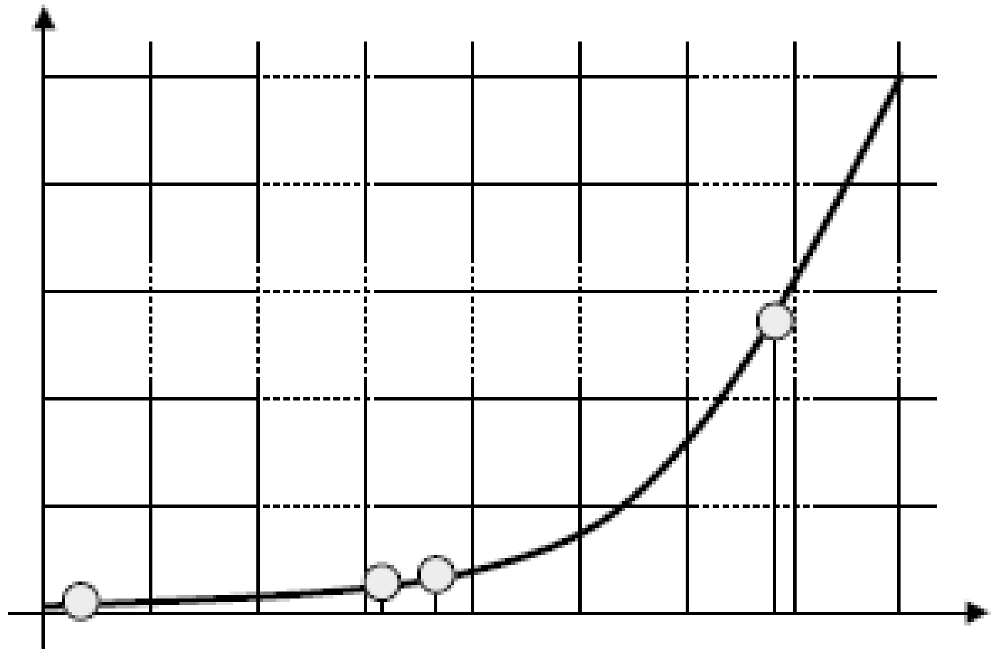
Estimator: $\langle I \rangle = I_{sec} = \frac{1}{N}\sum_{i=1}^{N} f(\bar{x}_i)$

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



MC Integration - Example

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



$x_1 = .86$ $\langle I \rangle = 2.74$

$x_2 = .41$ $\langle I \rangle = 1.44$

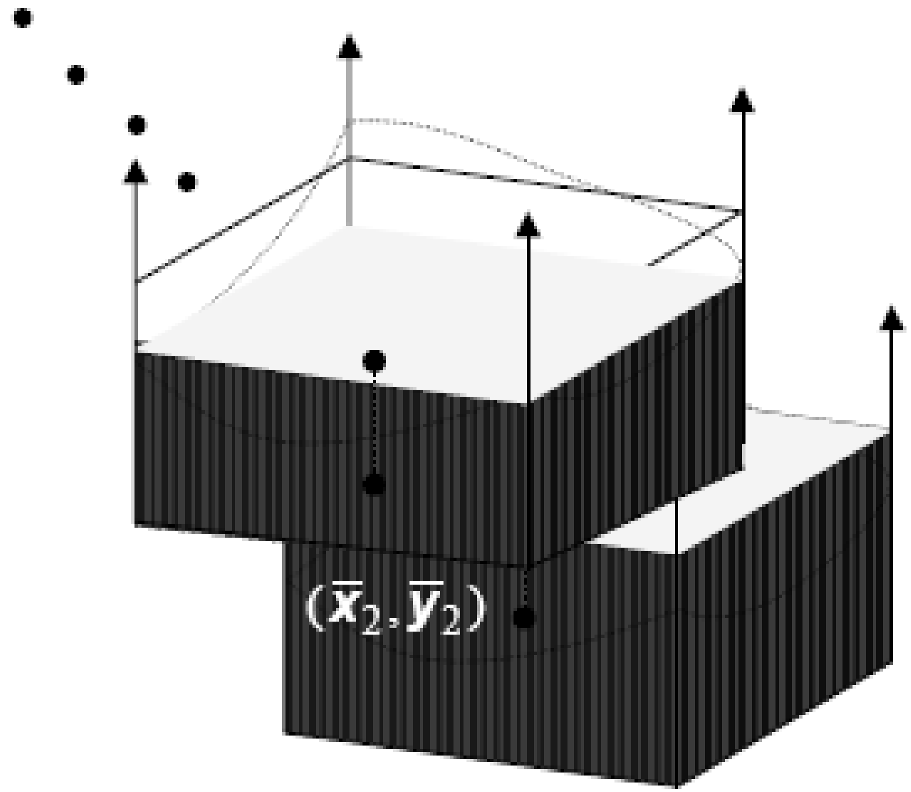
$x_3 = .02$ $\langle I \rangle = 0.96$

$x_4 = .38$ $\langle I \rangle = 0.75$

MC Integration: 2D

- Secondary estimator:

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

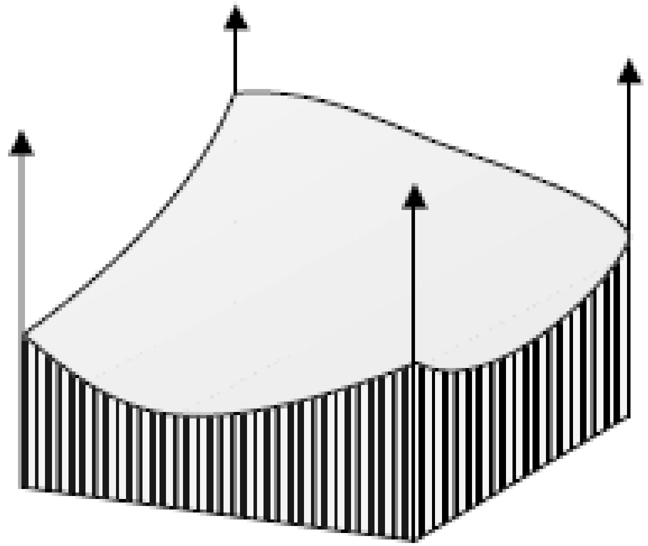


Monte Carlo Integration - 2D

- MC Integration works well for higher dimensions
- Unlike quadrature

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

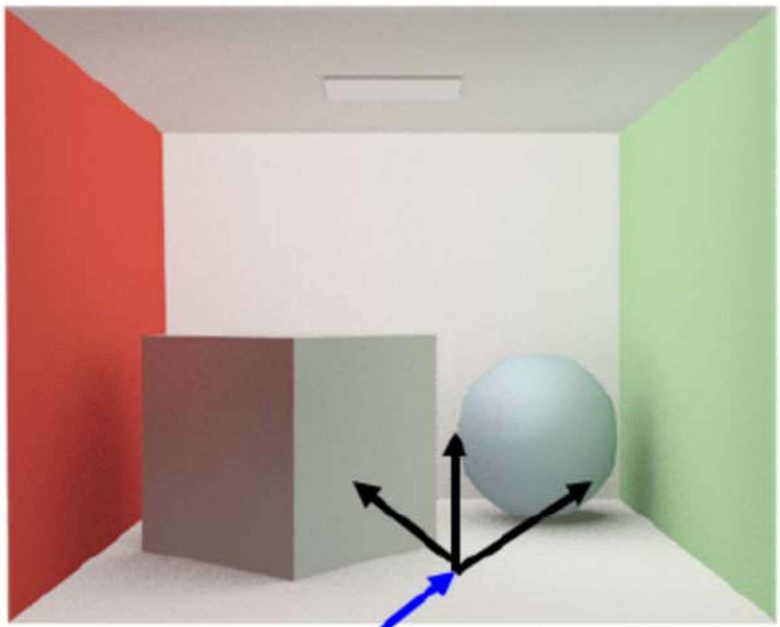
$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i, y_i)}{p(x_i, y_i)}$$

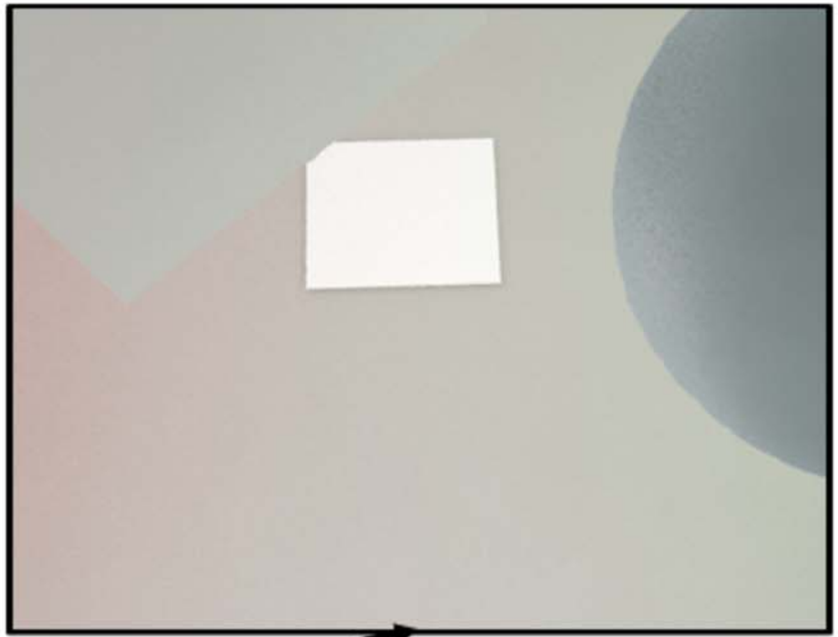


Advantages of MC

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

Review: Rendering Equation



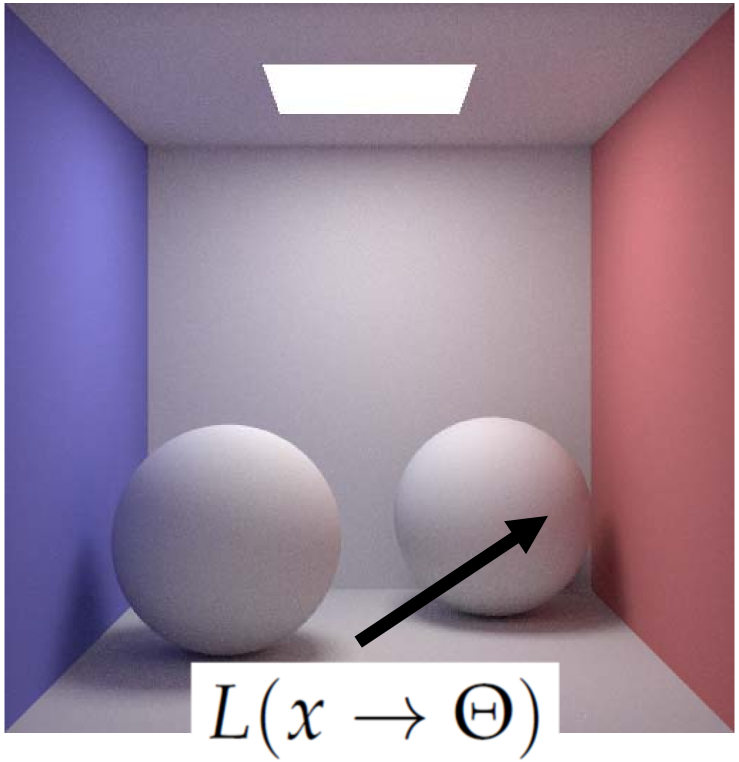


Incoming radiance on the hemisphere

$$L_r(x \rightarrow \Theta) = \int_{\Psi} L(x \leftarrow \Psi) f_r(x, \Psi \rightarrow \Theta) \cos\theta_x dw_{\Psi}$$

Evaluation

- **To compute** $L(x \rightarrow \Theta)$ **:**
  - **Check** $L_e(x \rightarrow \Theta)$
  - **Evaluate** $L_r(x \rightarrow \Theta)$



$$L_r(x \rightarrow \Theta) = \int_{\Psi} L(x \leftarrow \Psi) f_r(x, \Psi \rightarrow \Theta) \cos\theta_x dw_{\Psi}$$

Evaluation

- **Use Monte Carlo**

- **Generate random directions on hemisphere Ψ using pdf p(Ψ)**

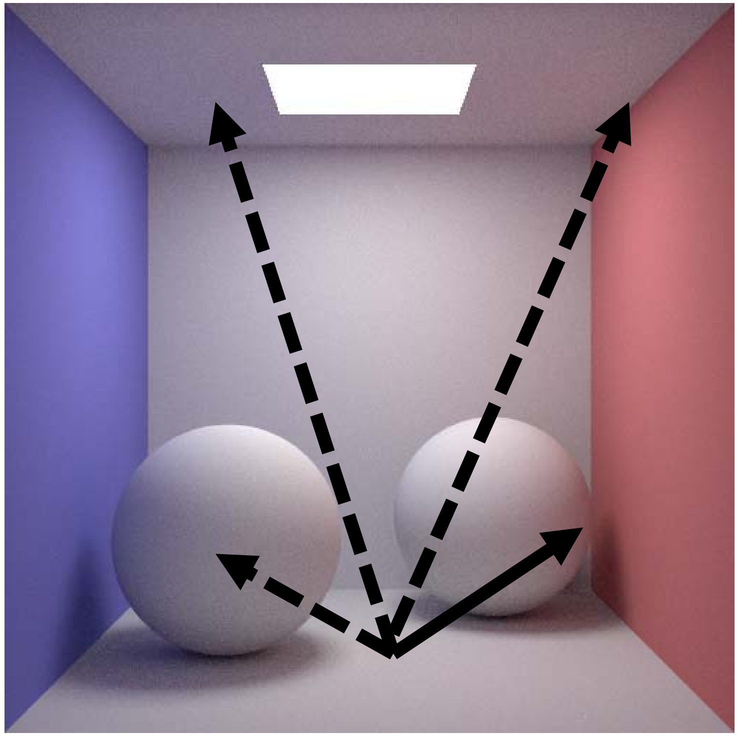
$$L_r(x \to \Theta) = \int_{\Psi} L(x \leftarrow \Psi) f_r(x, \Psi \to \Theta) \cos\theta_x dw_{\Psi}$$

$$\hat{L}_r(x \to \Theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{L(x \leftarrow \Psi_i) f_r(x, \Psi_i \to \Theta) \cos\theta_x}{p(\Psi_i)}$$

- **How about** $L(x \leftarrow \Psi_i)$ **?**

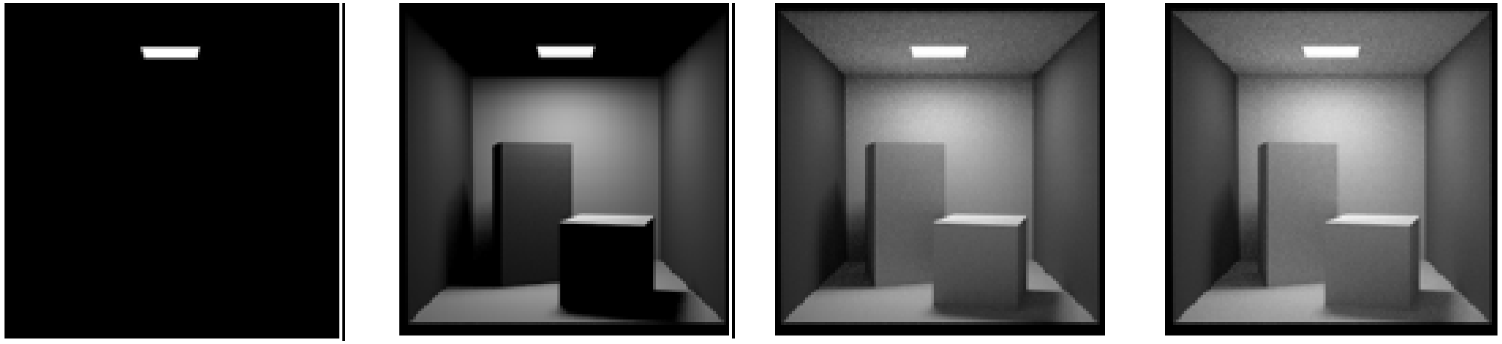
Evaluation

- **How about $L(x \leftarrow \Psi_i)$?**



- **Perform ray casting backward**
- **Compute radiance from those visible points to x**
  - **Assume reciprocity**
- **Recursively perform the process**
  - **Each additional bounce supports one more indirect illumination**

When to end recursion?



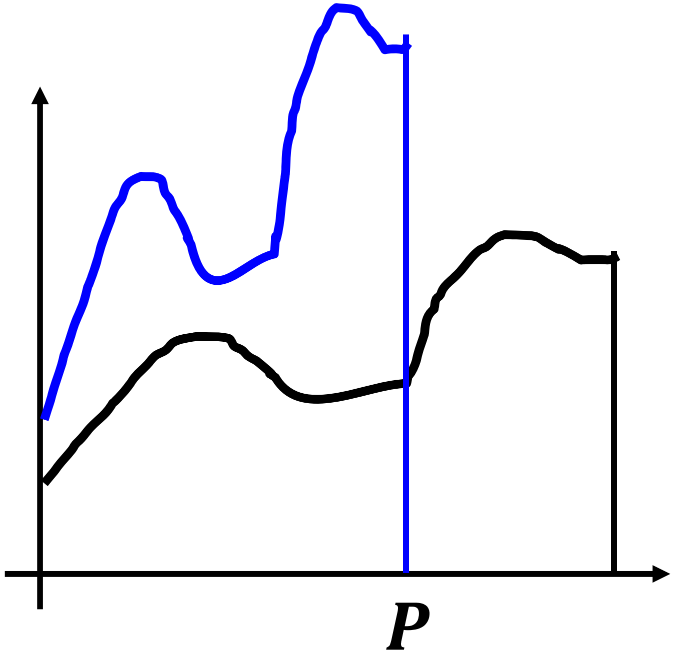
From kavita's slides

- **Contributions of further light bounces become less significant**
  - **Max recursion**
  - **Some threshold for radiance value**
- **If we just ignore them, estimators will be biased, i.e., cause a systematic error**

Russian Roulette

- **Integral: Substitute $y = Px$**

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



- **Estimator**

$$\hat{I}_{roulette} = \begin{cases} \frac{f(x_i)}{P} & \text{if } x_i \le P, \\ 0 & \text{if } x_i > P. \end{cases}$$

Russian Roulette

- **Pick absorption probability, α = 1-P**
  - **Recursion is terminated**

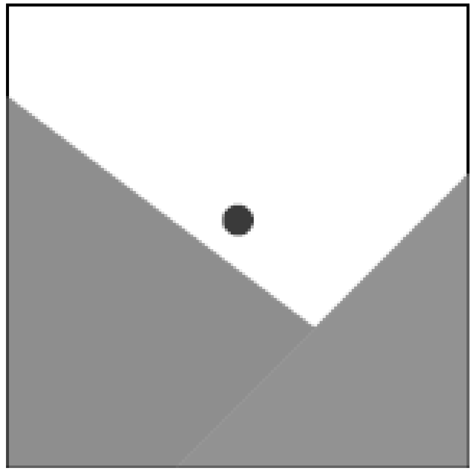
- **1- α, i.e., P, is commonly to be equal to the reflectance of the material of the surface**
  - **Darker surface absorbs more paths**

Algorithm so far

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

Pixel Anti-Aliasing

- **Compute radiance only at the center of pixel**
  - **Produce jaggies**

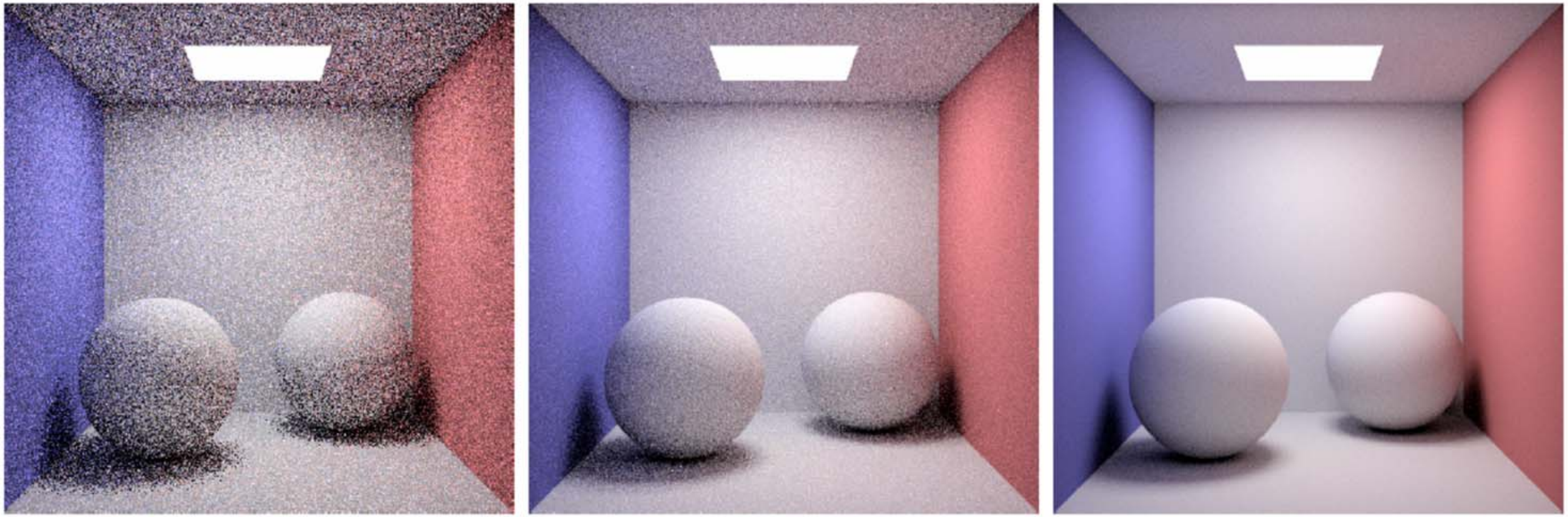


- **We want to evaluate using MC**
- **Simple box filter**
  - **The averaging method**

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 spp (samples per pixel) | 4 spp | 16 spp

- **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**

Class Objectives were:

- **Understand a basic structure of Monte Carlo ray tracing**
  - **Monte Carlo integration**
  - **Russian roulette for its termination**
  - **Path tracing**
  - **Rigorous analysis is important for this physically based approach, but skipped in this class**

Summary

- Rasterization based rendering
  - Rendering pipeline and various transformations
  - Culling and clipping
  - Illumination and rasterization
  - Texture mapping

- Physically based rendering
  - Basic ray tracing structures
  - Radiometric quantities (e.g., radiance)
  - Basic material function, BRDF
  - Rendering equation

Related Courses

- **CS580: Advanced Computer Graphics**
  - **Focus on rendering techniques that generate photo-realistic images**
- **CS482: Interactive Computer Graphics**
  - **Advanced techniques on rendering**
  - **Recent deep learning methods**
  - **I'll teach it at Fall of this year**

Coming Lectures

- Denoising
- Nerf
- Diffusion