

---

# CS482: Monte Carlo Ray Tracing:

---

Sung-Eui Yoon  
(윤성익)

<http://sglab.kaist.ac.kr/~sungeui/IC>



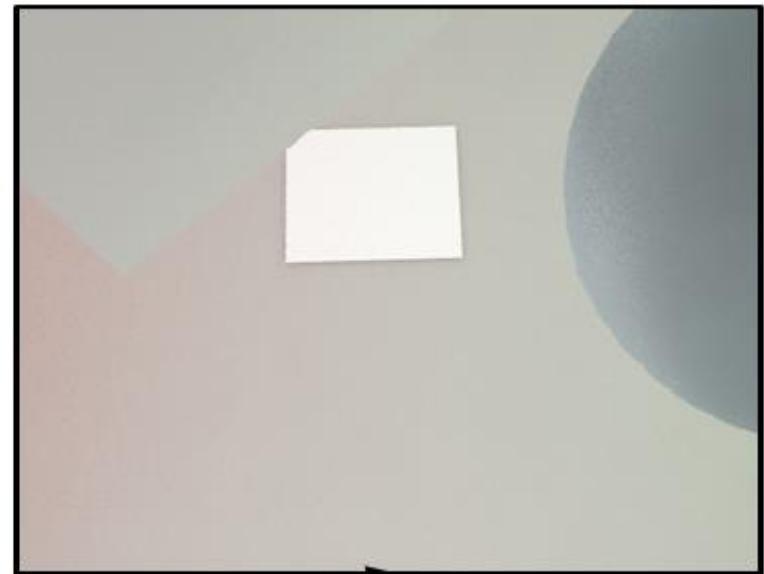
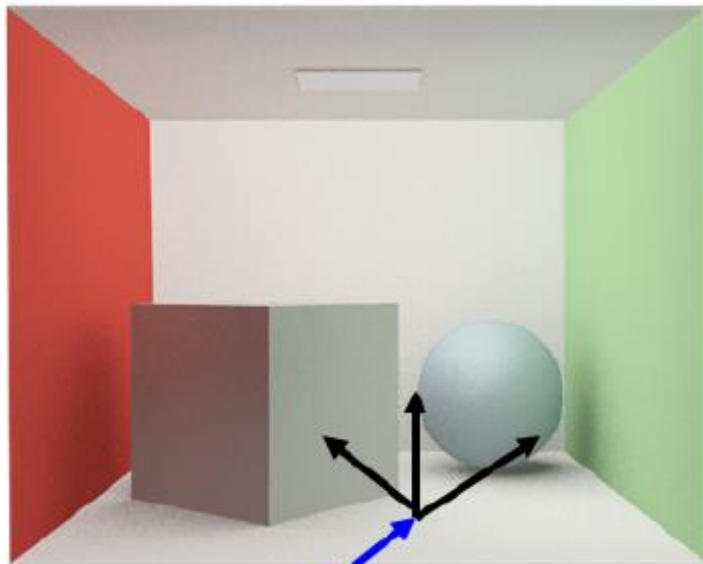
# Class Objectives (Ch. 15)

---

- **Understand a basic structure of Monte Carlo ray tracing**
  - **Russian roulette for its termination**
  - **Path tracing**
- **Last time:**
  - **Monte Carlo integration: sampling approach for solving the rendering equation**
  - **Estimator and its variance**



# Rendering Equation

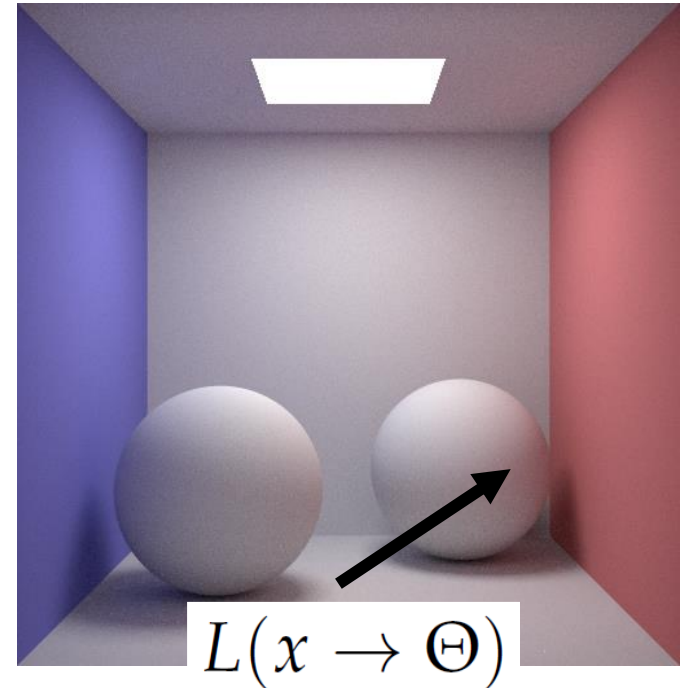


Incoming radiance on the hemisphere

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

# Evaluation

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



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

# Evaluation

---

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

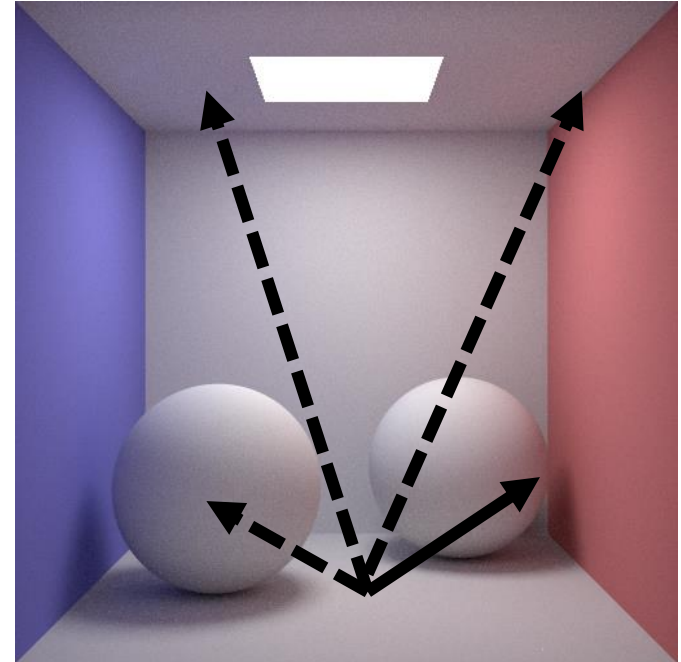
$$L_r(x \rightarrow \Theta) = \int_{\Psi} L(x \leftarrow \Psi) f_r(x, \Psi \rightarrow \Theta) \cos \theta_x d\omega_{\Psi}$$

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

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

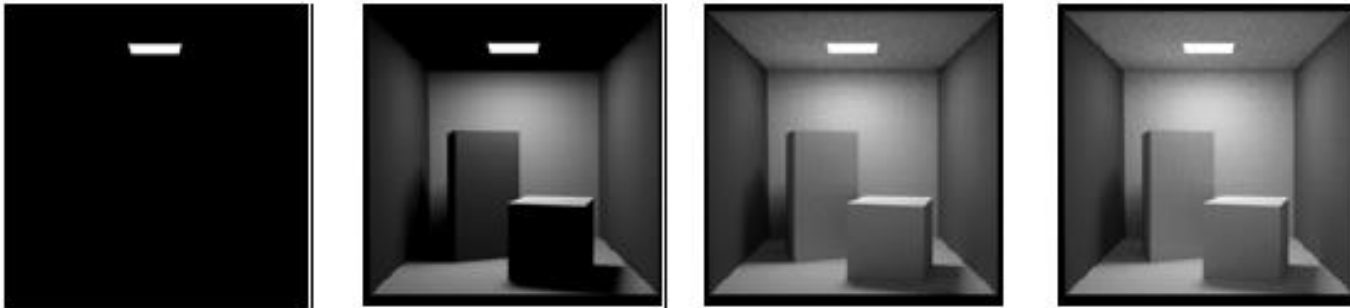
# Evaluation

- 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**

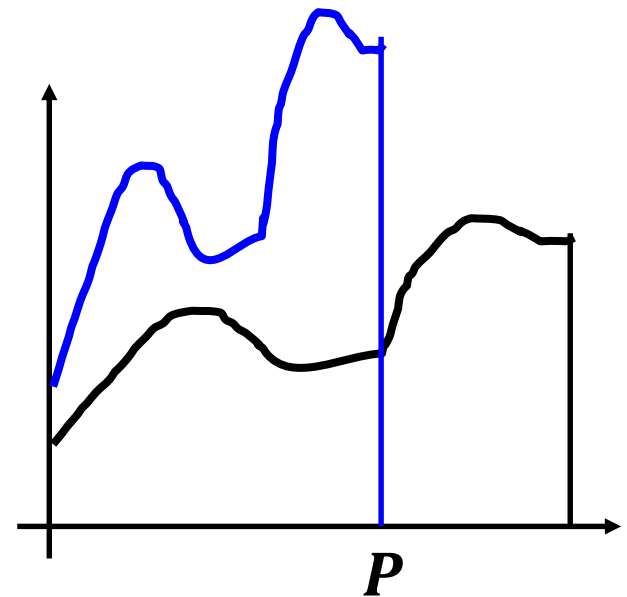
# Russian Roulette

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

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

- **Estimator**

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



- **Variance?**



# Russian Roulette

---

- **Pick absorption probability,  $\alpha = 1-P$** 
  - Recursion is terminated
- **$1-\alpha$ , 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

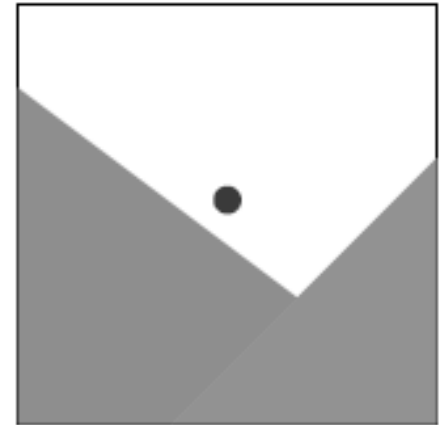
---

- **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

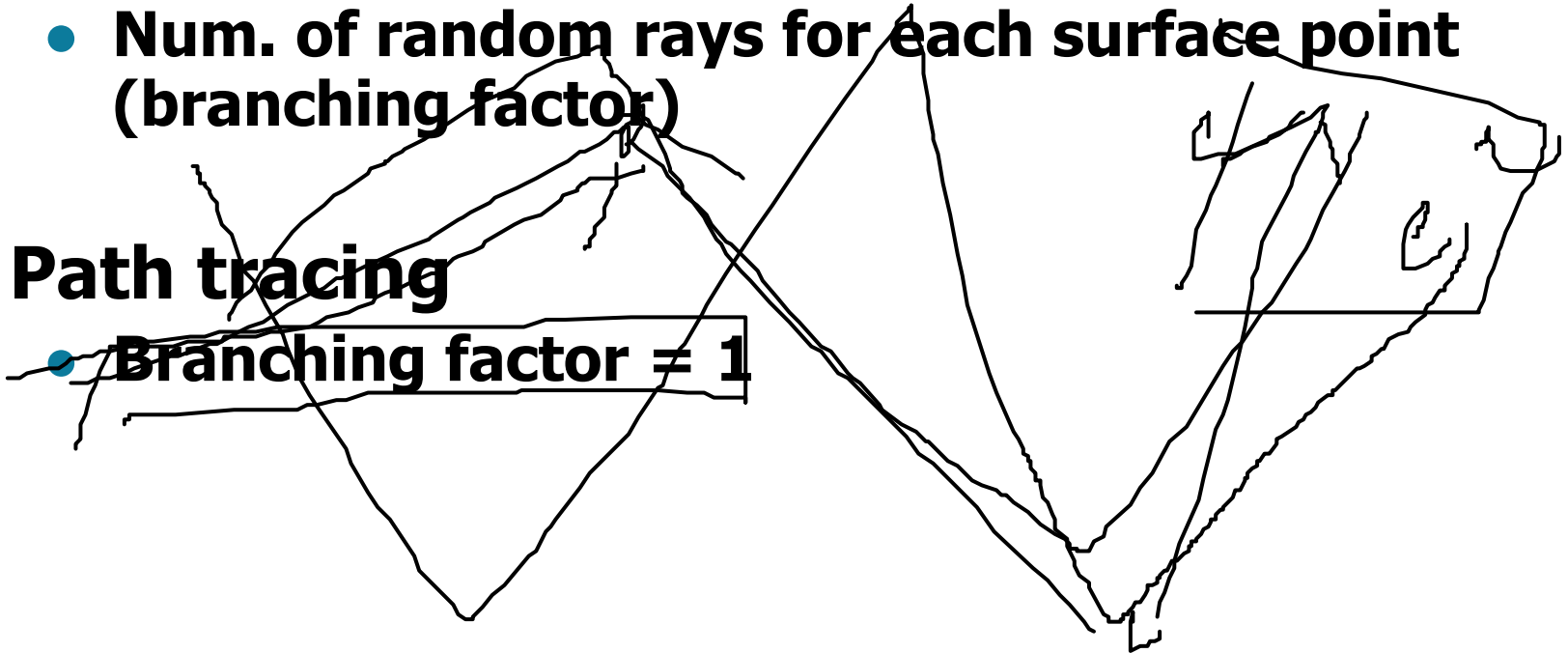
---

- **Parameters**

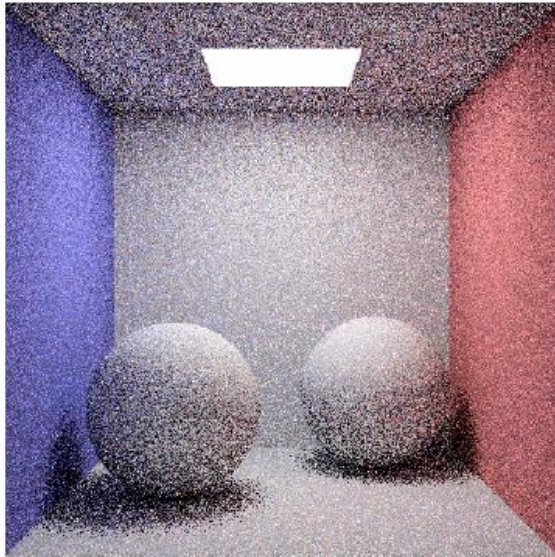
- Num. of starting ray per pixel
- Num. of random rays for each surface point (branching factor)

- **Path tracing**

- **Branching factor = 1**

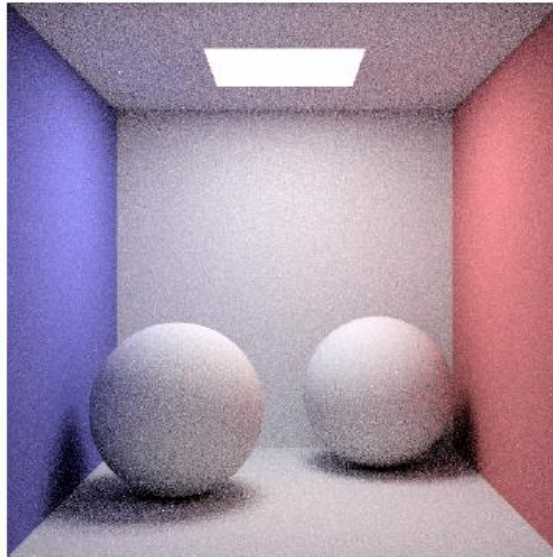


# Path Tracing

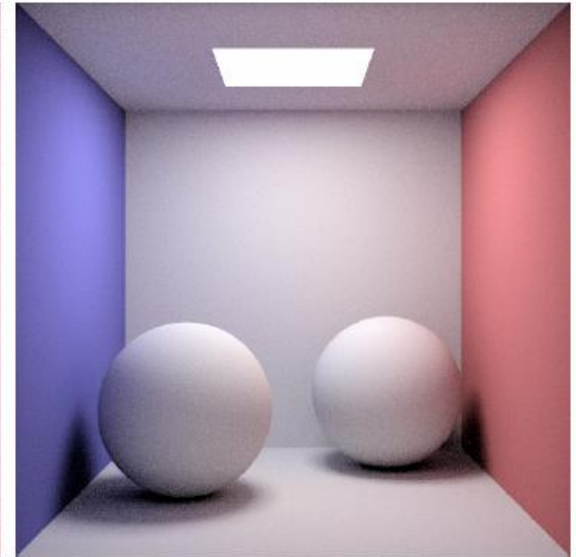


1 spp

(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**

# Performance

---

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

# Some Example

---



**Uniform sampling  
(64 samples per pixel)**



**Adaptive sampling**



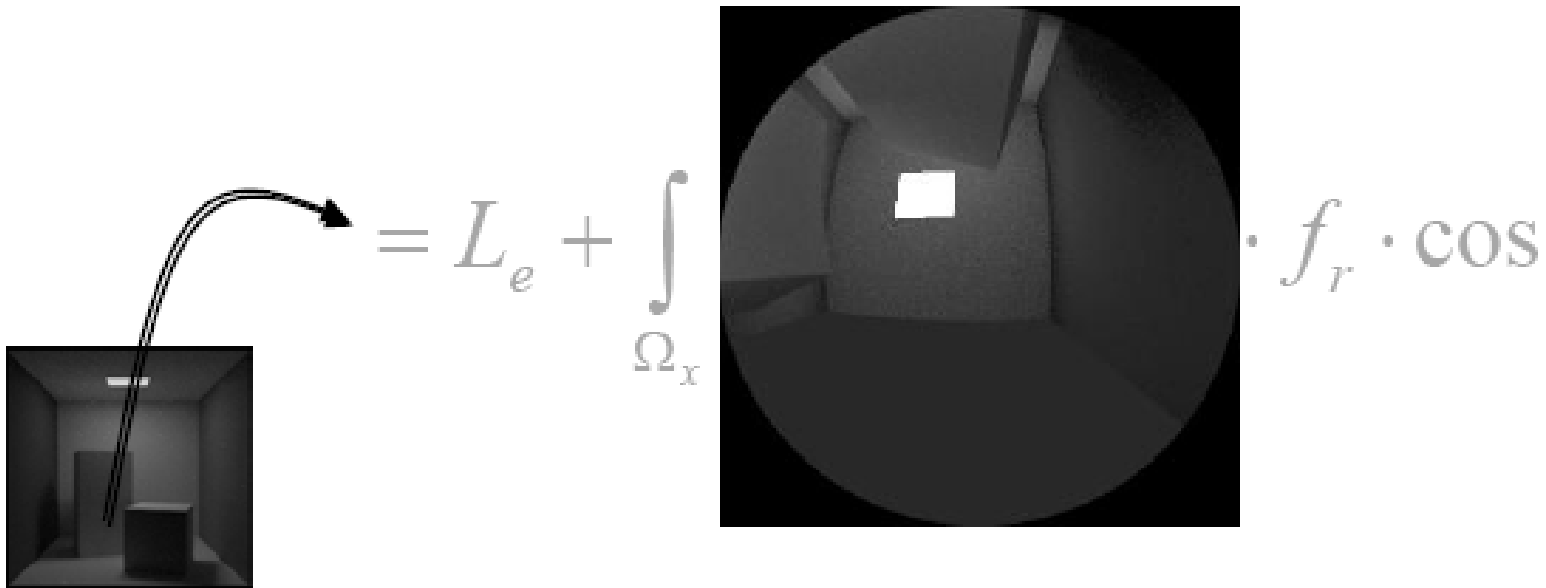
**Reference**



# Importance Sampling

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

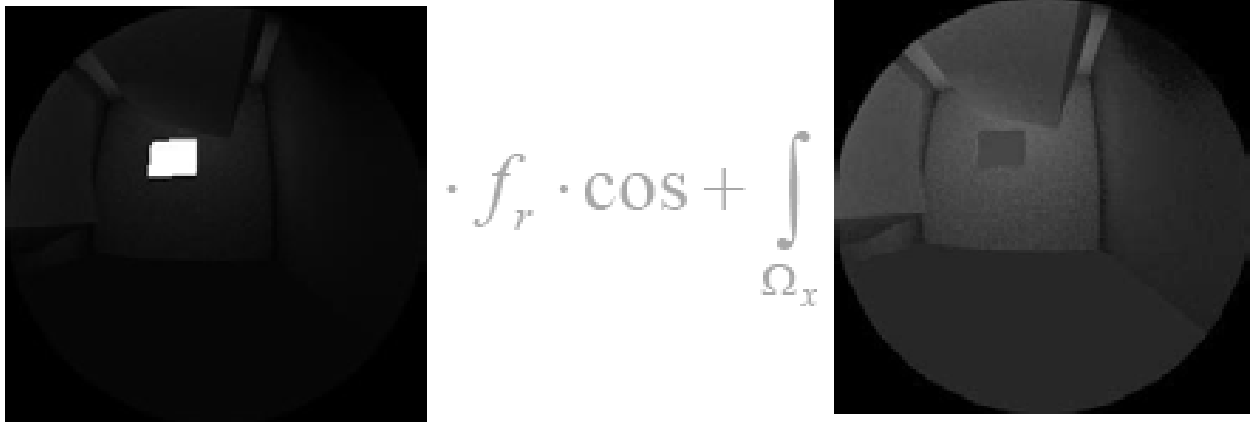
Radiance from light sources + radiance from other surfaces



# Importance Sampling

---

$$L(x \rightarrow \Theta) = L_e + L_{direct} + L_{indirect}$$

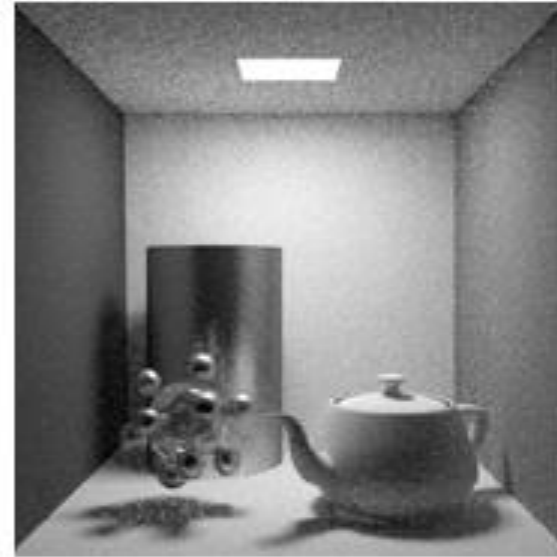
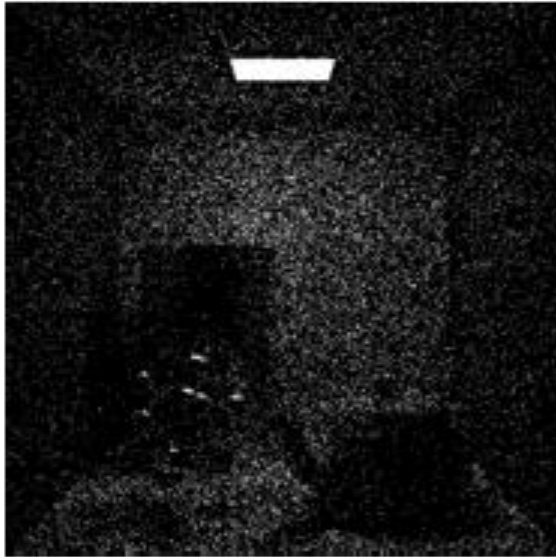
$$= L_e + \int_{\Omega_x} \text{img}_1 \cdot f_r \cdot \cos + \int_{\Omega_x} \text{img}_2 \cdot f_r \cdot \cos$$


- So ... sample direct and indirect with separate MC integration



# Comparison

---



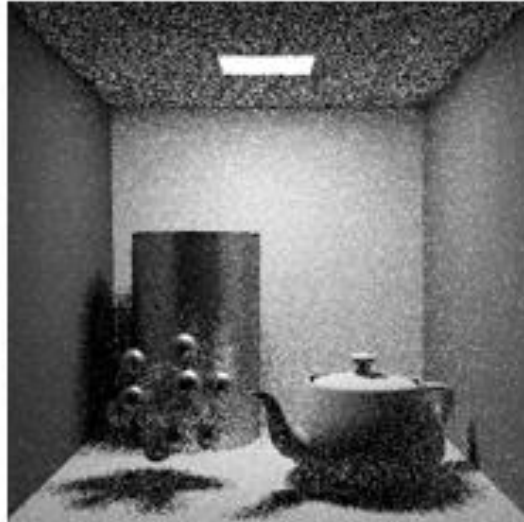
From kavita's slides

- **With and without considering direct illumination**
  - **16 samples / pixel**

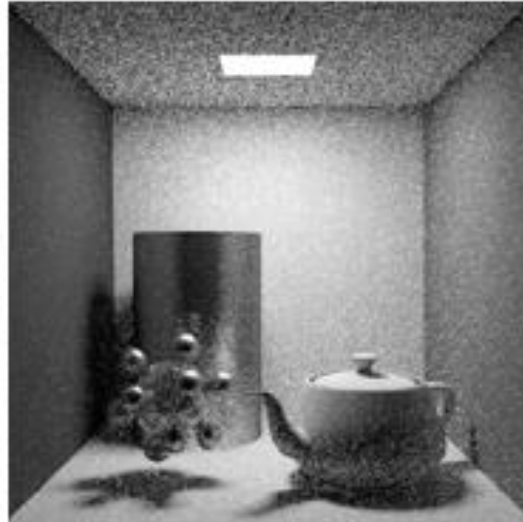
# Rays per pixel

---

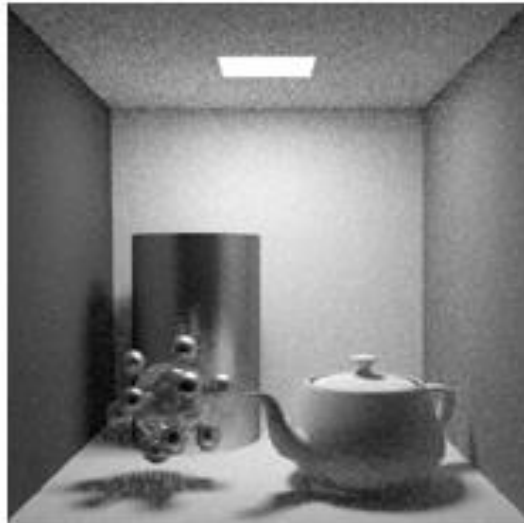
1 sample/  
pixel



4 samples/  
pixel



16 samples/  
pixel



256 samples/  
pixel



# Class Objectives were:

---

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

# Class Objectives were:

---

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

# Next Time...

---

- **Acceleration techniques for global illumination methods**

# Homework

---

- **Go over the next lecture slides before the class**
- **Watch 2 SIG/CVPR/ISMAR videos and submit your summaries every Mon. class**
  - **Just one paragraph for each summary**
  - **Any top-tier conf (e.g., ICRA) is okay**

## Example:

**Title: XXX XXXX XXXX**

**Abstract: this video is about accelerating the performance of ray tracing. To achieve its goal, they design a new technique for reordering rays, since by doing so, they can improve the ray coherence and thus improve the overall performance.**