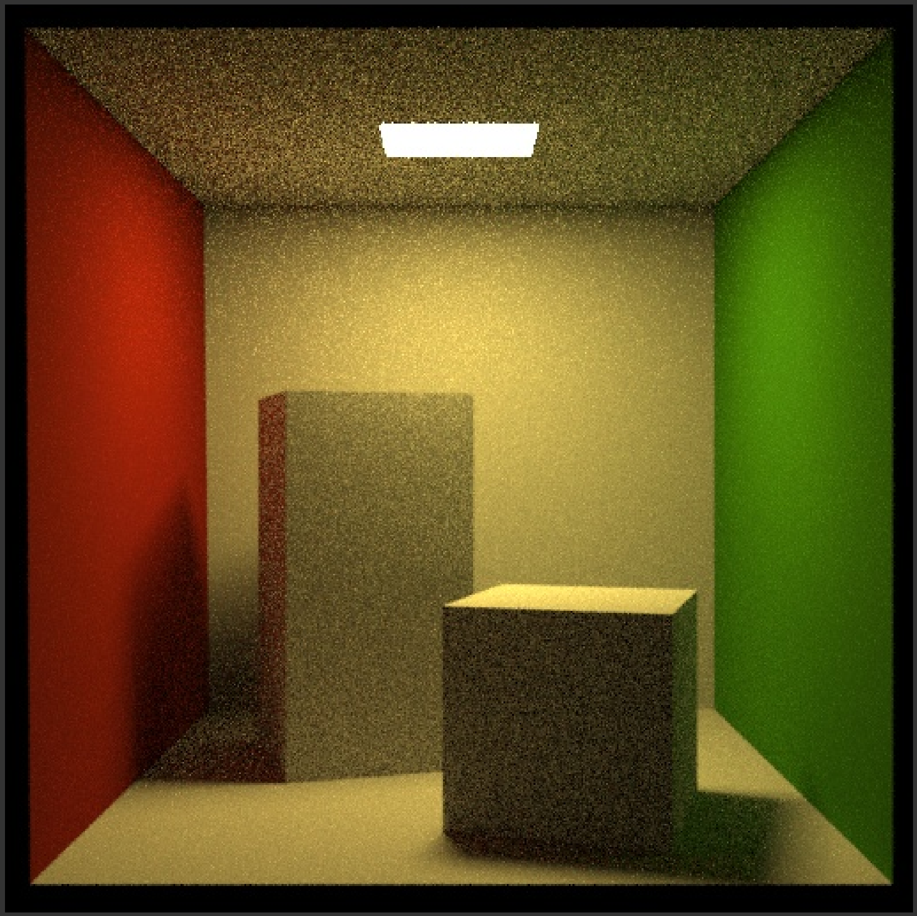
# CSE168: Rendering Algorithms
# Monte Carlo Ray Tracing II



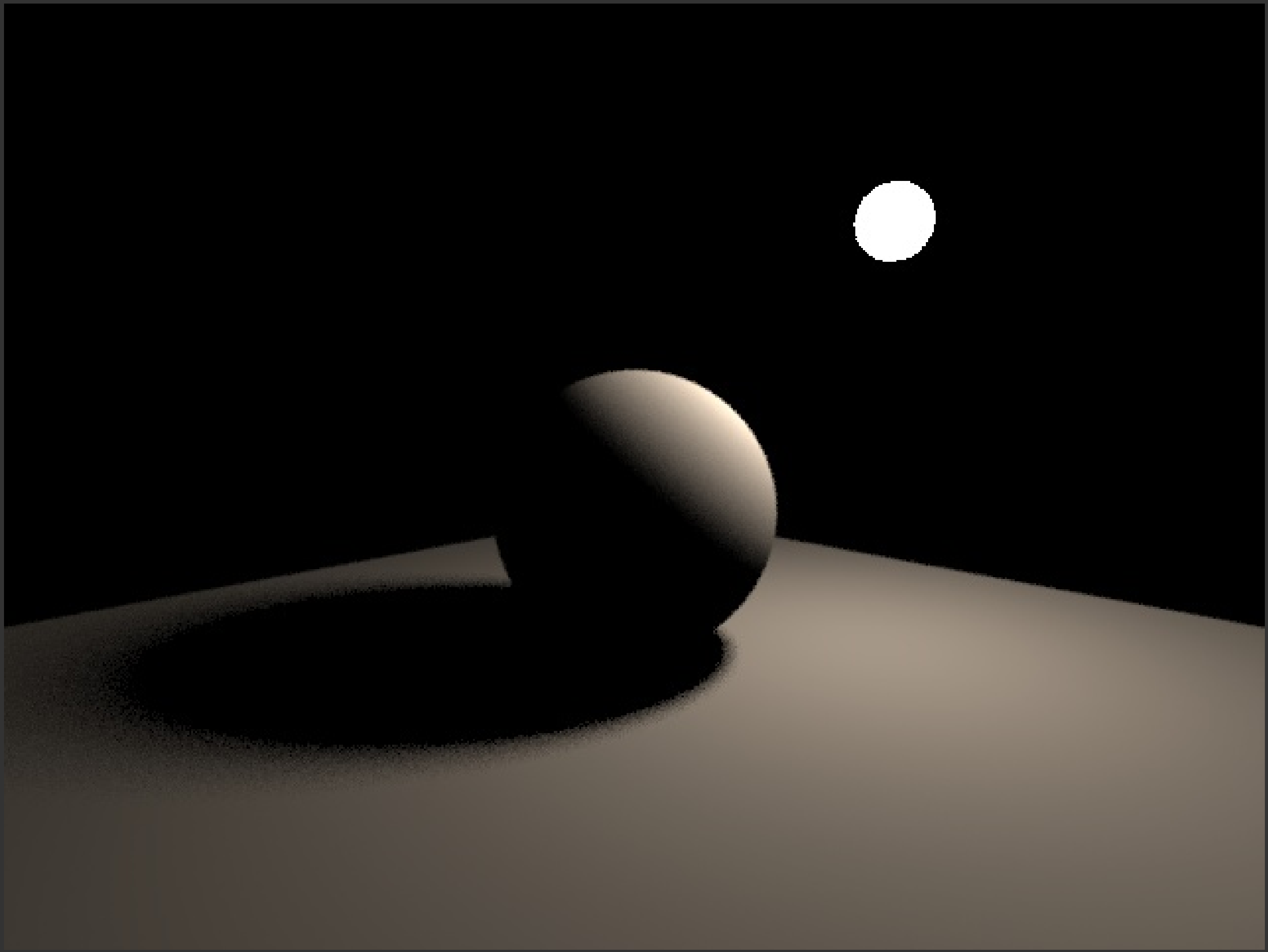
Henrik Wann Jensen

henrik@cs.ucsd.edu

# Todays Menu

- Local illumination
- Indirect diffuse lighting
- Glossy reflections
- Global illumination
- The rendering equation
- Light transport
- Path tracing

# Local Illumination



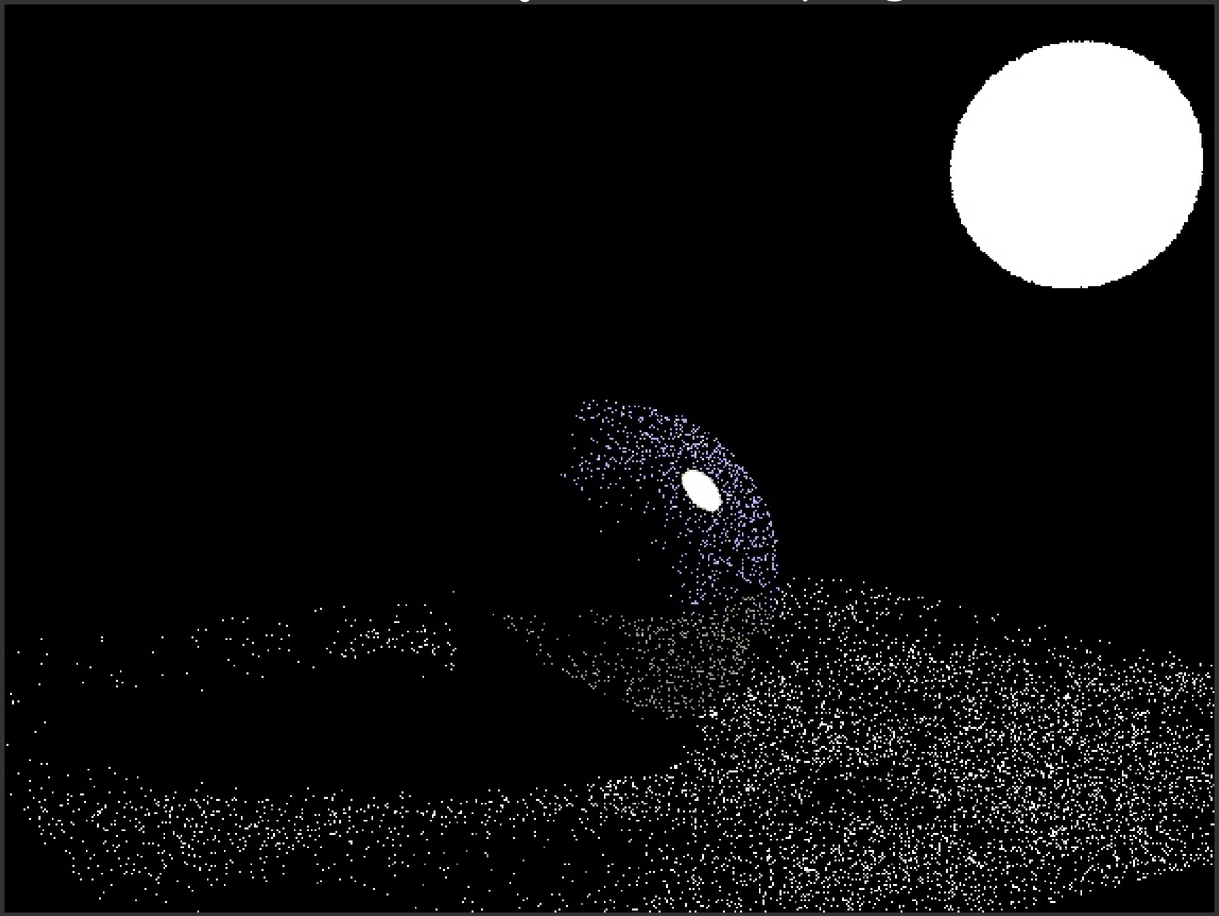
# Diffuse Lighting

$$L(x,\vec{\omega}) = \frac{R_d}{\pi}\int_{2\pi} L_i(x,\vec{\omega}')(\vec{n}\cdot\vec{\omega}')\,d\vec{\omega}'$$

- Accounts for all direct light on a diffuse surface
- Use Monte Carlo ray tracing...

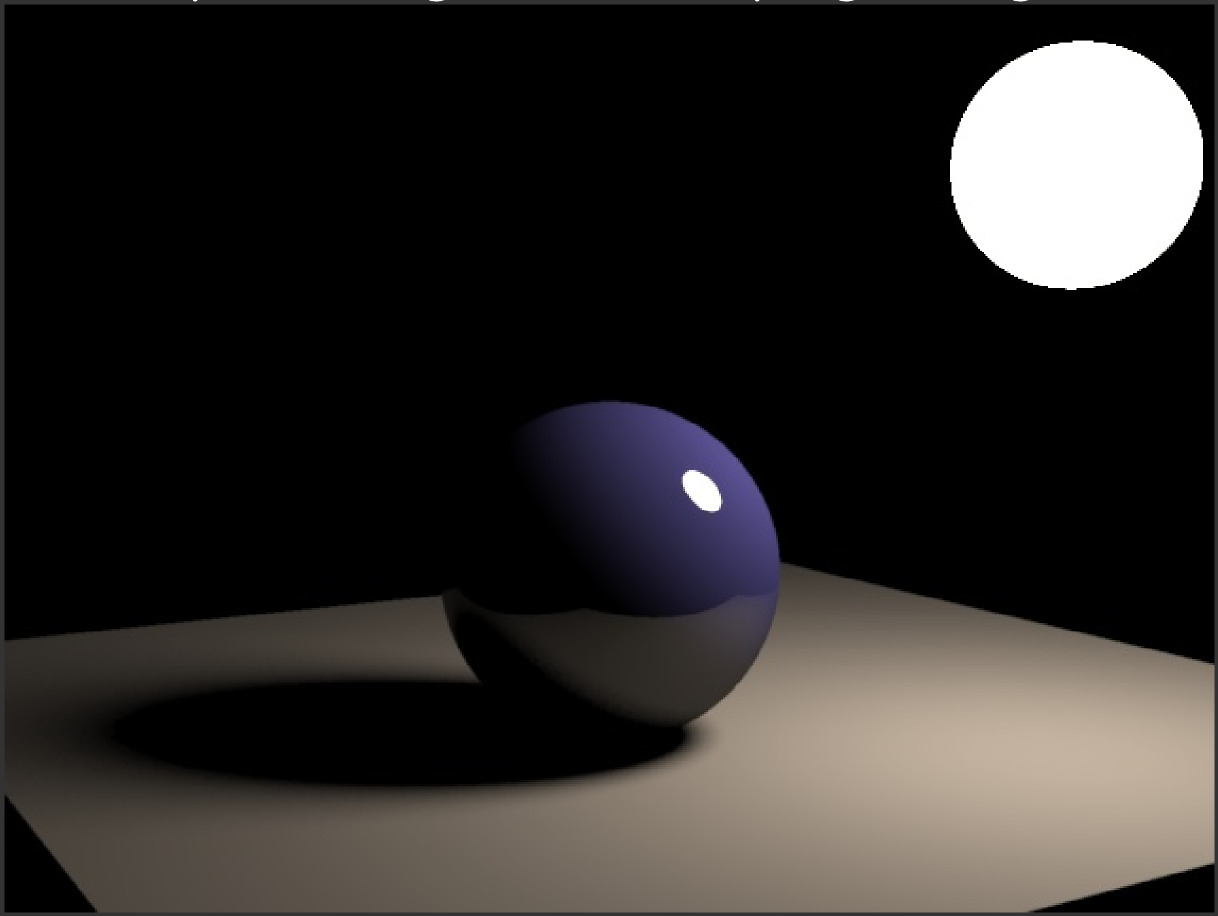
# Example: Diffuse Lighting

Naive rejection sampling



# Diffuse Lighting

Explicit integration - sampling the light



# Sphere Light

Use Monte Carlo sampling to estimate visibility

- Find disc on light source oriented towards $x$
- Create $N$ samples in disc
- Trace a ray fromn $x$ to each sample
- Visibility equals number of times the ray hits the light divided by $N$
- $E \approx \frac{Phi}{4\pi r^2} \cdot$ visibility

# Indirect Diffuse Lighting

$$L(x,\vec{\omega}) = \frac{R_d}{\pi}\int_{2\pi} L_i(x,\vec{\omega}')(\vec{n}\cdot\vec{\omega}')\,d\vec{\omega}'$$

- Sample all incident directions to estimate indirect diffuse lighting

# Diffuse Sampling Method 1

Trace $N$ rays. Each ray has a random direction $\vec{\omega}$ in the upper hemisphere.

$$L \approx \frac{R_d}{\pi N} \sum_{i=1}^{N} L(\vec{\omega}) \cos\theta$$

# Diffuse Sampling Method 2

Trace $N$ rays. Use importance sampling where the probability of generating a ray in the upper hemisphere is proportional to $\cos\theta$:

- $L = \frac{R_d}{\pi} \int L_i \cos\theta$
- pdf: $p(\theta, \phi) \propto \cos\theta$
- Nusselt analog
  - $\star$ $\theta = \sin^{-1} \sqrt{\xi_1}$
  - $\star$ $\phi = 2\pi\xi_2$

# Diffuse Sampling Method 2

$$L \approx \frac{R_d}{\pi N} \sum_{i=1}^{N} L(\theta, \phi)$$

where $\theta = \sin^{-1} \sqrt{\xi_1}$ and $\phi = 2\pi\xi_2$.

# The Radiance Equation

$$L(x, \vec{\omega}) = \int_{2\pi} f_r(x, \vec{\omega}, \vec{\omega}') L_i(x, \vec{\omega}') (\vec{n} \cdot \vec{\omega}')\, d\vec{\omega}'$$

- Accounts for all direct light on a surface

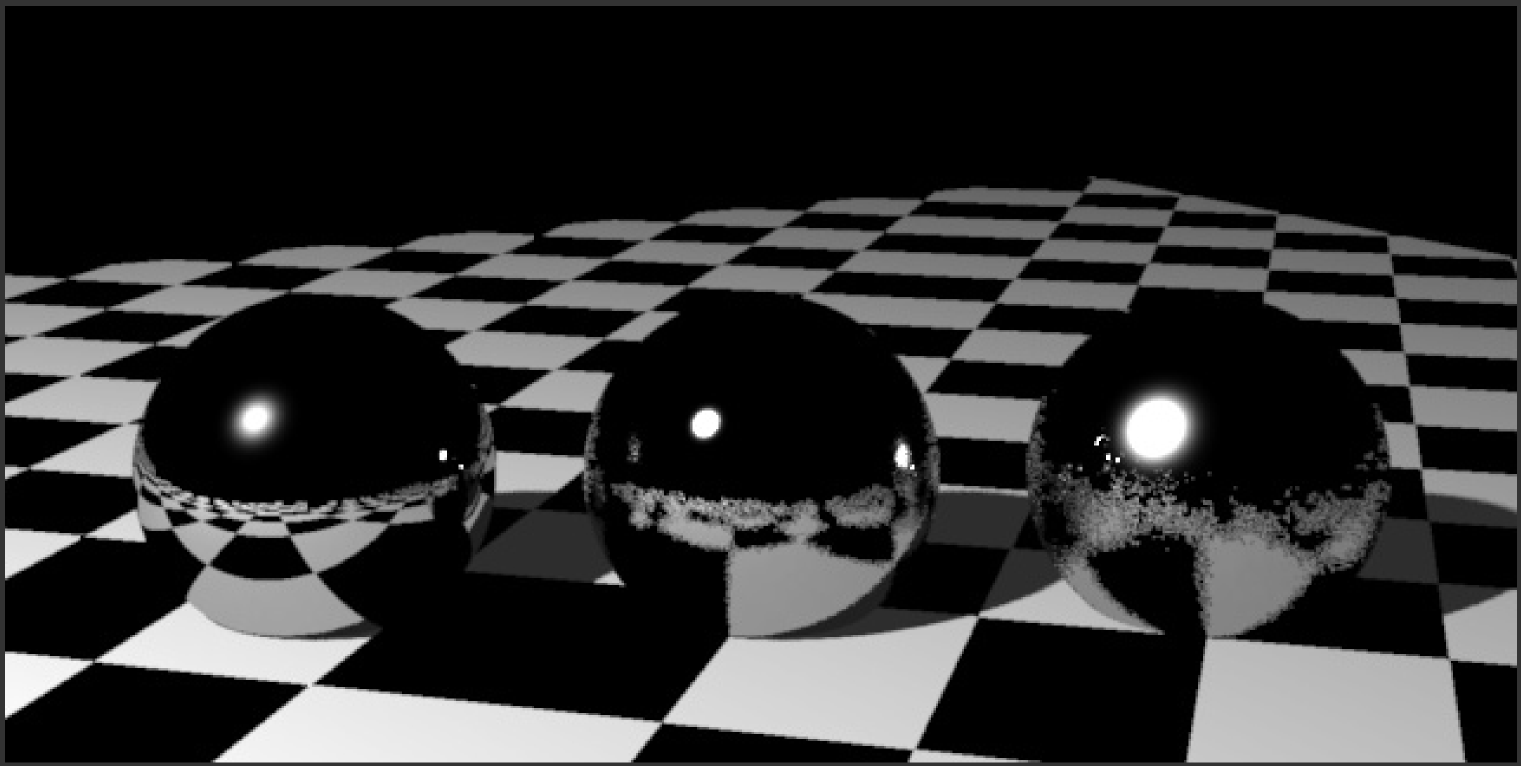
# Glossy Phong

$$p(\theta, \phi) = \frac{n+1}{2\pi} \cos^n \theta$$

$$P(\theta, \phi) = \int_0^{\phi} \int_0^{\theta} p(\theta', \phi') \sin \theta' \, d\theta' \, d\phi'$$

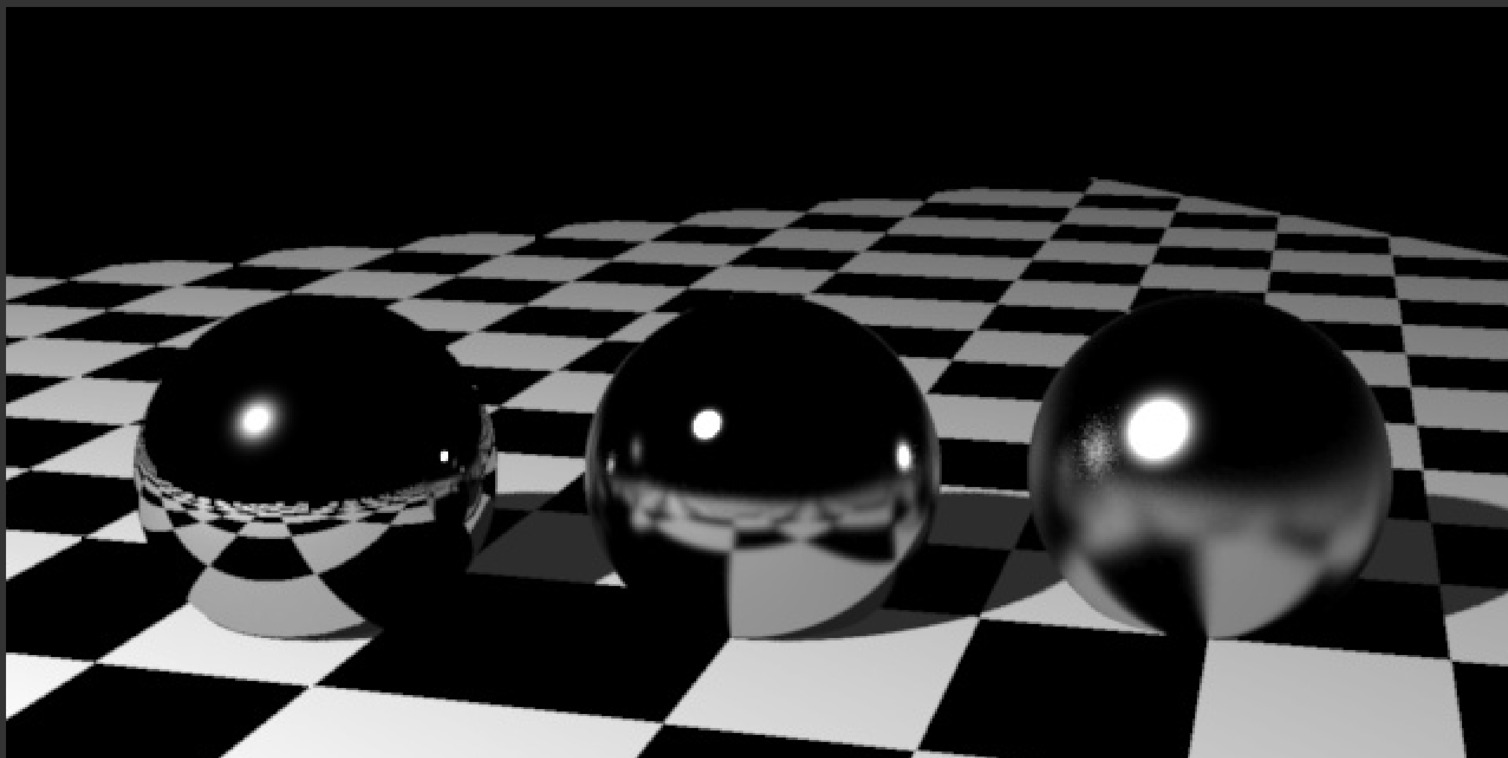
$$(\theta, \phi) = (\ acos((1-u)^{1/(n+1)}),\ 2\pi v)$$

# Glossy Spheres



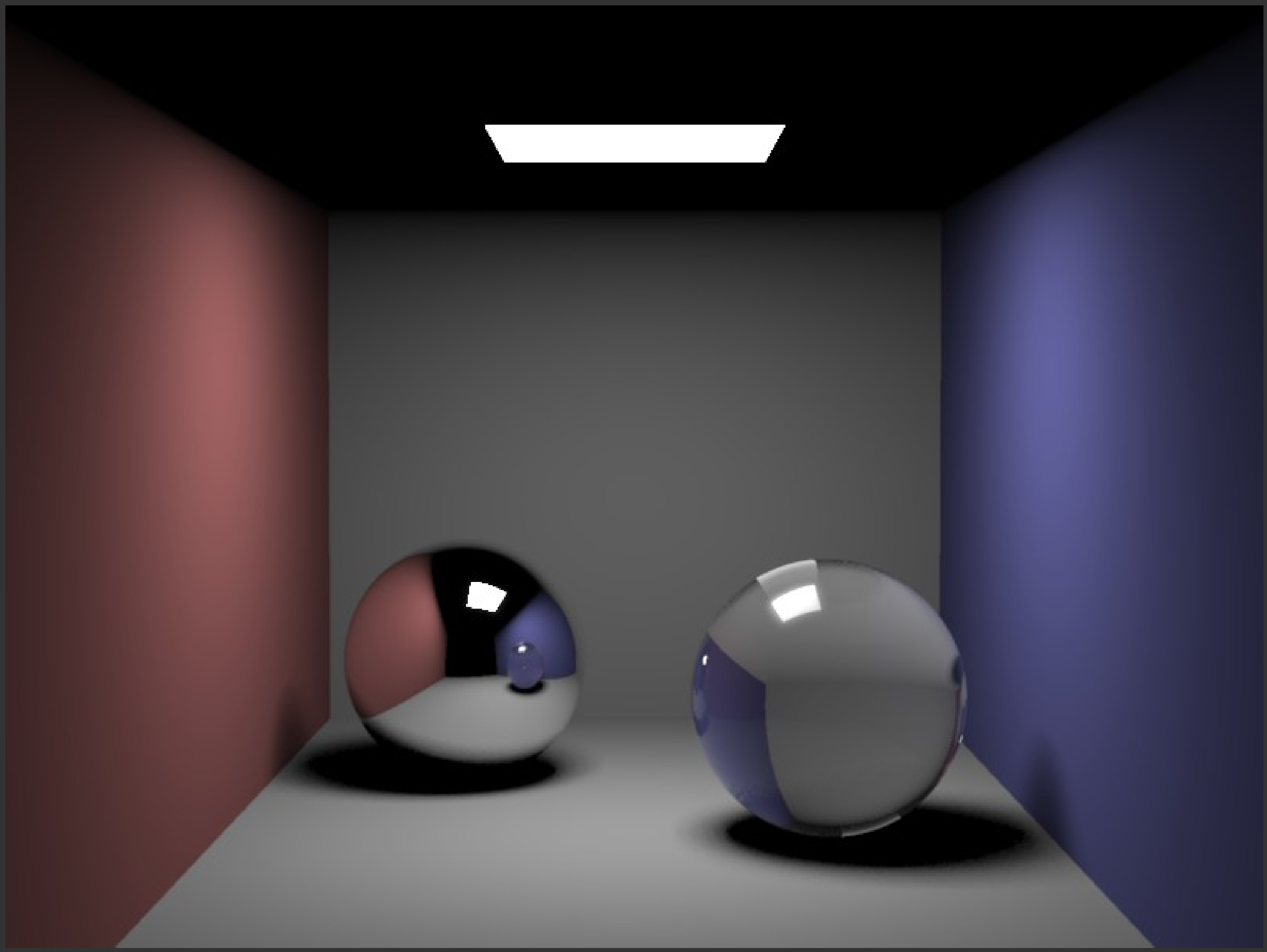
1 sample

# Glossy Spheres

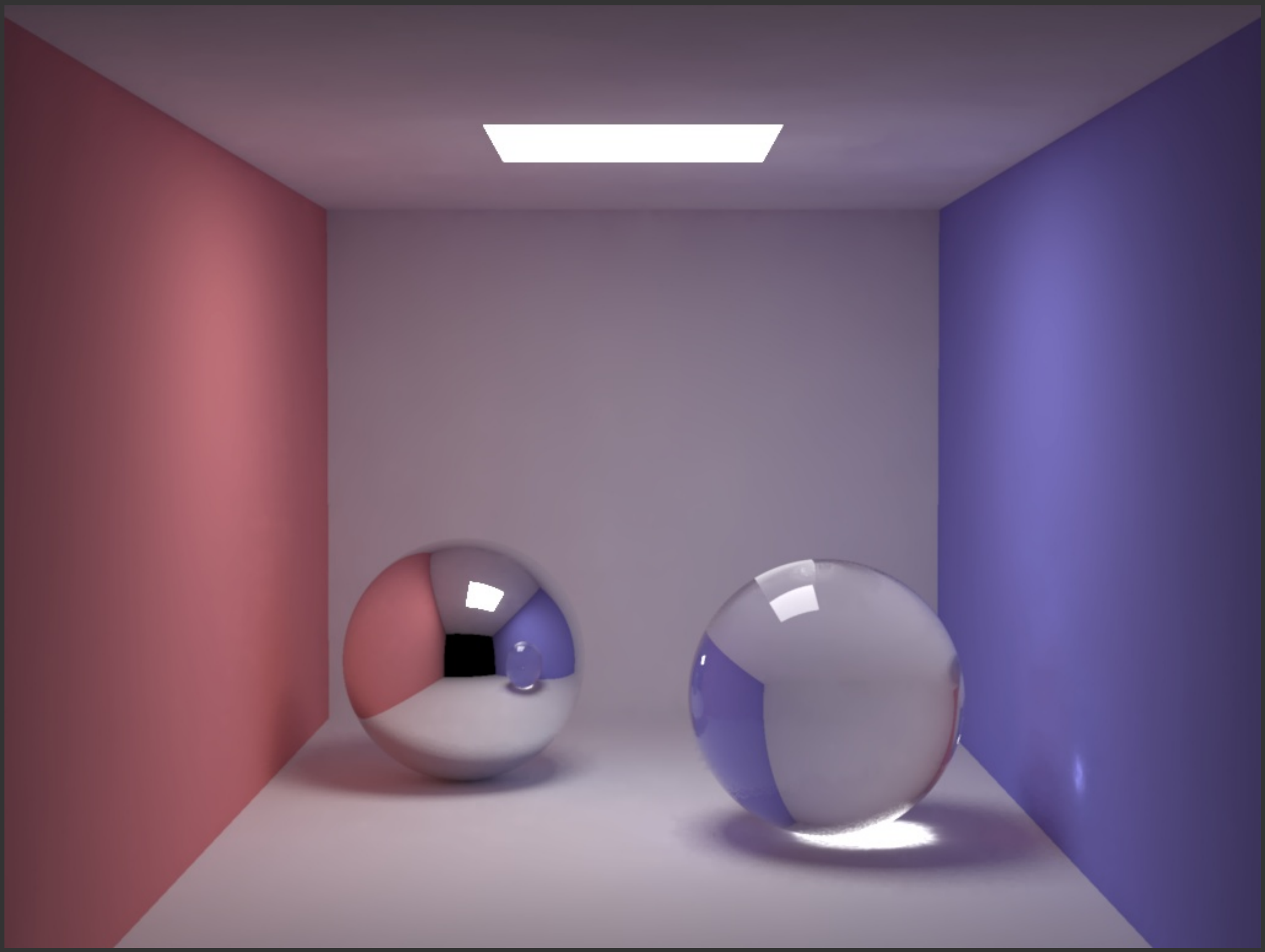


256 samples

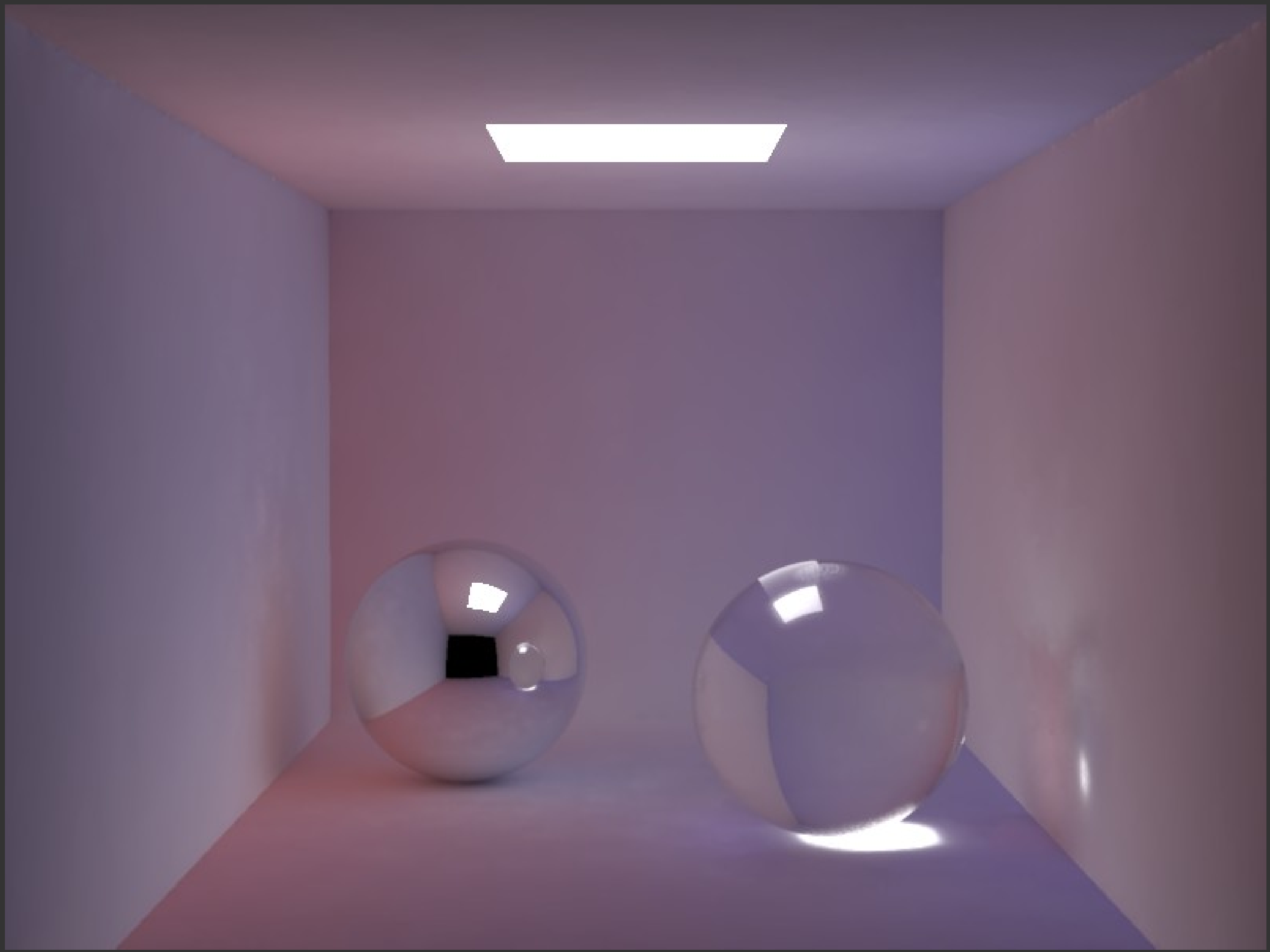
# Box: Direct Illlumination



# Box: Global Illlumination



# Box: Indirect Illlumination



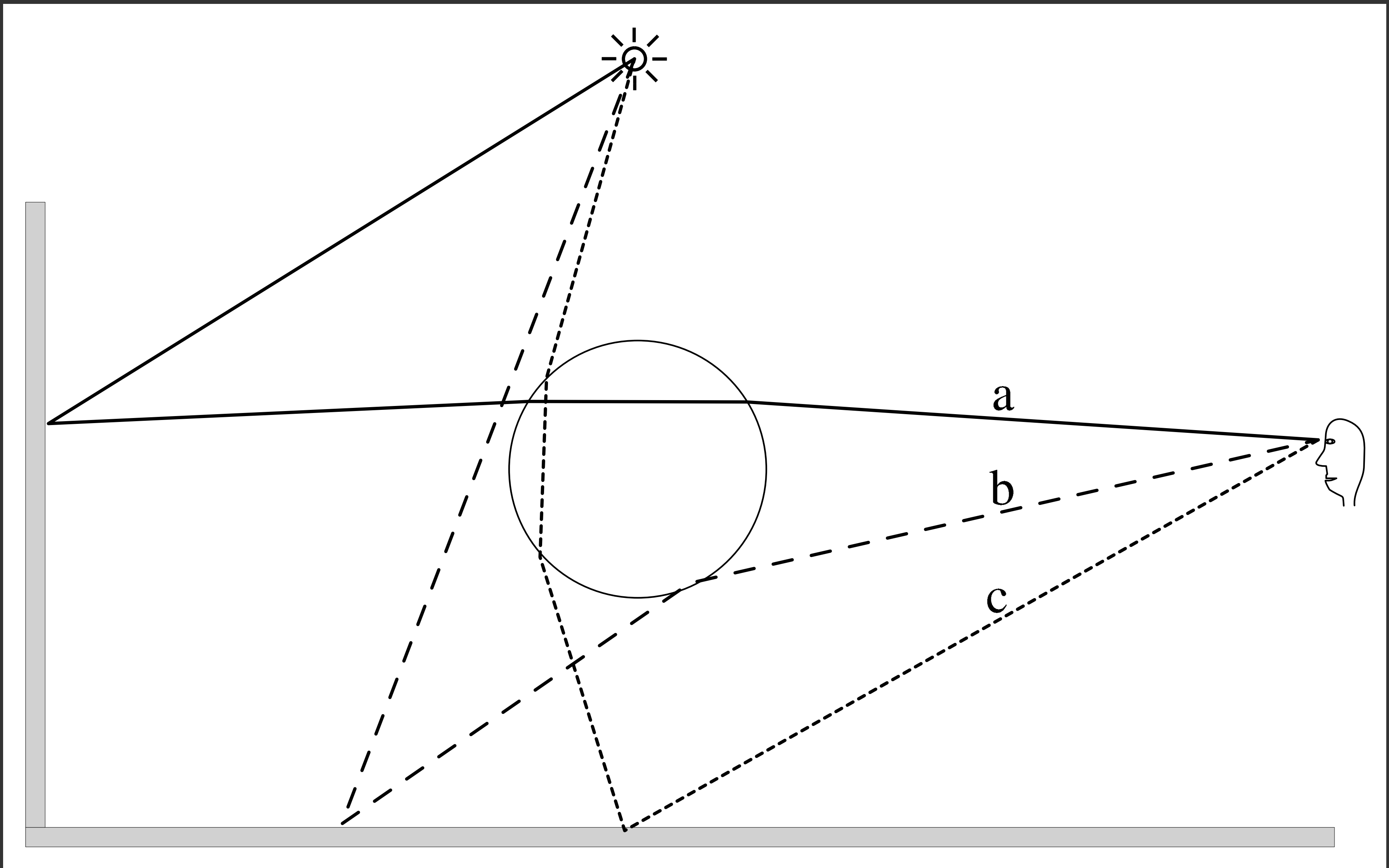
# Global Illumination

- Significant computer graphics research topic for the last 20 years
  - ⋆ Finite element radiosity
  - ⋆ Monte Carlo ray tracing

# Light Transport

- L = a light
- D = diffuse reflection
- S = specular reflection
- E = the eye

# Light Transport



# Global Illumination

- Described by the rendering equation
- Need to solve for global equilibrium of light
- Similar equations in heat transfer and neutron transport

# The Rendering Equation

$$L_{out} \;=\; L_{emitted} + L_{reflected}$$

$$\begin{aligned} L(w, \vec{\omega}) &= L_e(x, \vec{\omega}) + L_r(x, \vec{\omega}) \\ &= L_e(x, \vec{\omega}) + \int_{2\pi} f_r(x, \vec{\omega}, \vec{\omega}') L(x, \vec{\omega}')(\vec{\omega} \cdot \vec{n})\, d\vec{\omega}' \end{aligned}$$

[Jim Kajiya, "The Rendering Equation", SIGGRAPH 1986]

# Monte Carlo Ray Tracing

- Recursive integration of the light

# Naive Monte Carlo

- Sample incoming light using N sample rays

$$L \approx \frac{1}{N} \sum_{i=1}^{N} f_r L(\xi_{i,1}\ ,\ \xi_{i,2}\ ,\ \xi_{i,3})(\vec{\omega} \cdot \vec{n})$$

Diffuse reflection:

$$L \approx \frac{R_d}{\pi N} \sum_{i=1}^{N} L(\theta, \phi)$$

where $\theta = \sin^{-1} \sqrt{\xi_1}$ and $\phi = 2\pi\xi_2$.

# Monte Carlo Ray Tracing

```
trace( ray )
  intersect object
  shade object
  return color

shade( object )
  for (s=0; s<N; s++)
    color += trace random ray
```

# Monte Carlo Ray Tracing

- Can simulate everything
- Flexible
- Incredibly slow...
- Exponential ray growth...

# Path Tracing

- Reformulate rendering equation
- Neumann expansion
- Solve path integral

# Neumann Expansion

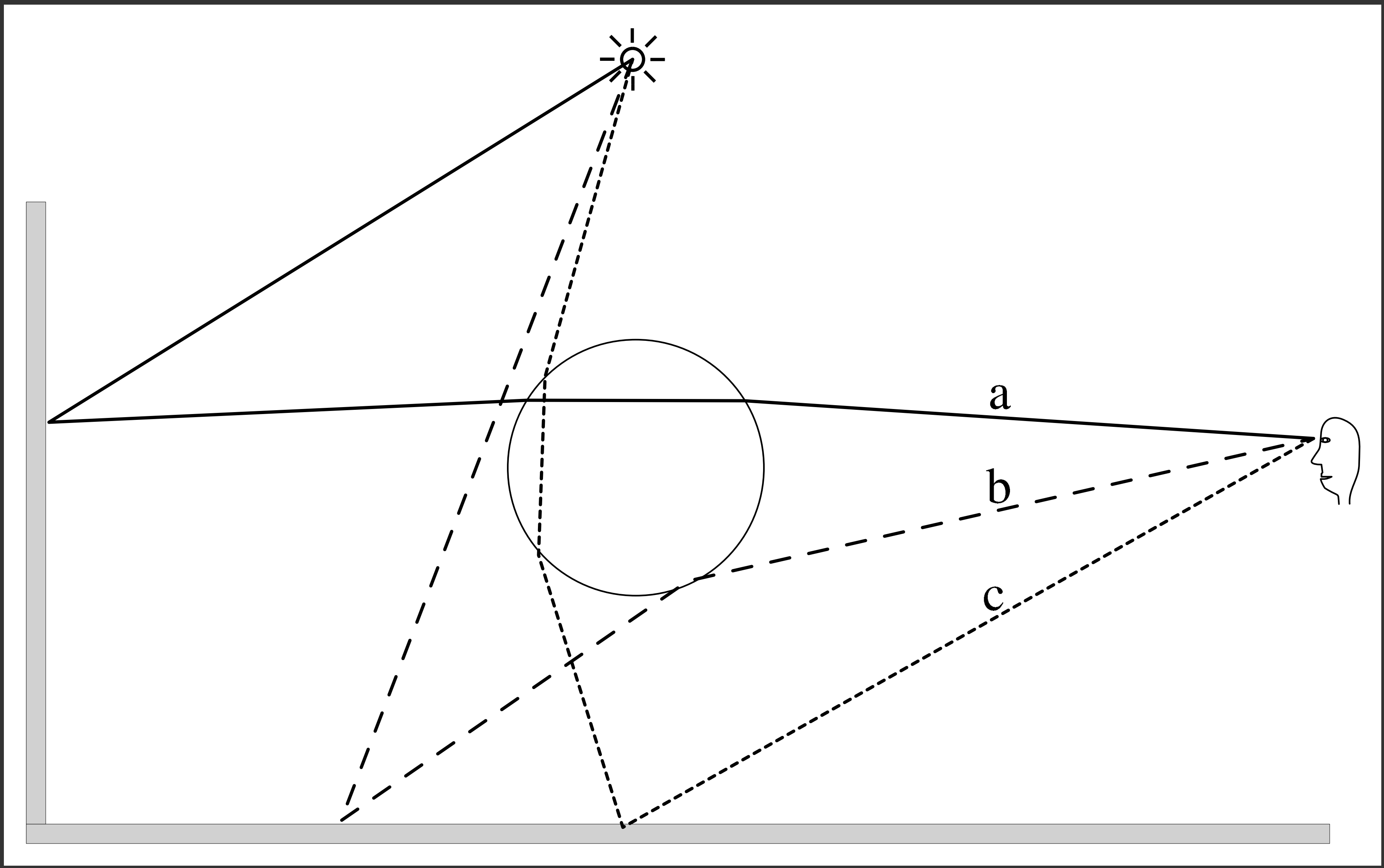
$$
\begin{aligned}
L &= L_e + \int f_r L \\
&= L_e + \int f_r L_e + \int f_r \int f_r L \\
&= L_e + \int f_r L_e + \int f_r \int f_r L_e + \int f_r \int f_r \int f_r L \\
&= \ldots
\end{aligned}
$$

$$
L = L_e + T L_e + T^2 L_e + T^3 L_e \ldots = \sum_{m=0}^{\infty} T^m L_e
$$

# Path Tracing

- Use one sample ray to avoid combinatorial explosion
- [Jim Kajiya, "The Rendering Equation", SIGGRAPH 1986]

# Path Tracing



# Path Tracing Algorithm

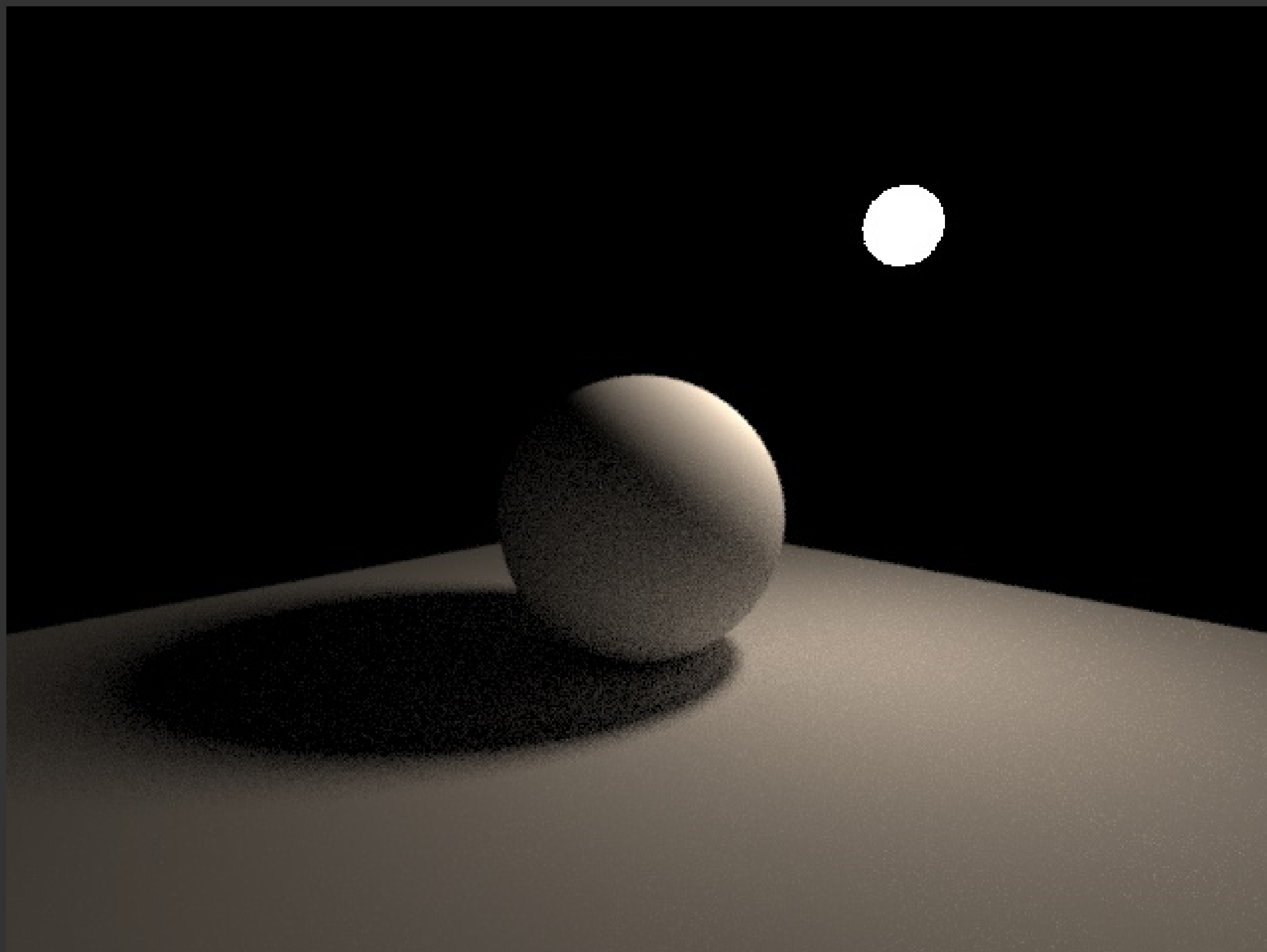
```
trace( ray )
  intersect object
  shade object
  return color

shade( object )
  trace random ray to sample incoming light
```

# Path Tracing Algorithm

- Use many samples per pixel
- Reflect only one ray when shading

# A Diffuse Shader



# Next Time

- Midterm