

# **Monte Carlo Ray Tracing**

---

**Computer Graphics  
CMU 15-462/15-662**

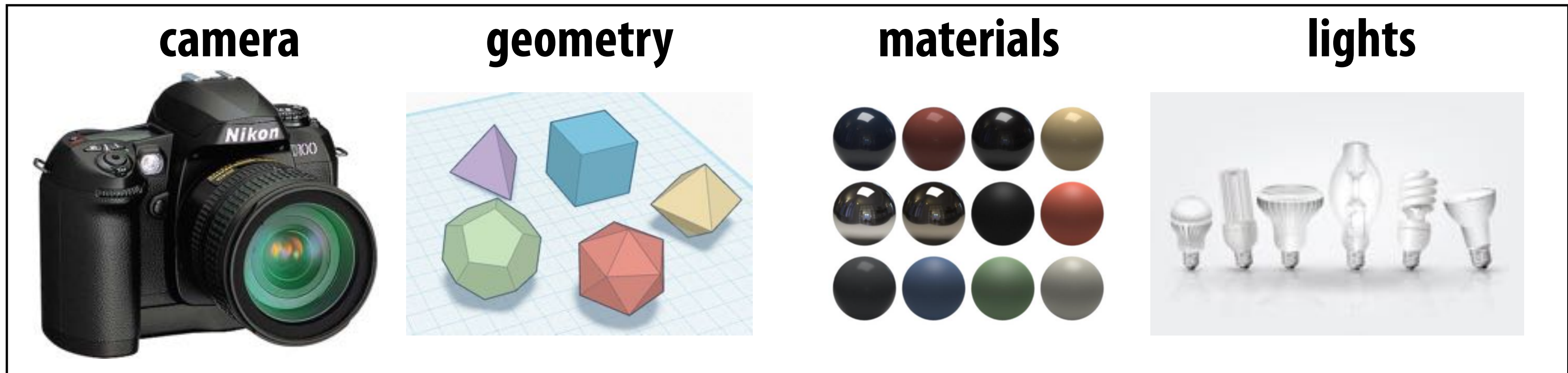
# TODAY: Monte Carlo Ray Tracing

- How do we render a photorealistic image?
- Put together many of the ideas we've studied:
  - color
  - materials
  - radiometry
  - numerical integration
  - geometric queries
  - spatial data structures
  - rendering equation
- Combine into final Monte Carlo ray tracing algorithm
- Alternative to rasterization, lets us generate much more realistic images (usually at much greater cost...)



# Photorealistic Ray Tracing—Basic Goal

What are the **INPUTS** and **OUTPUTS**?



(“scene”)

Ray Tracer

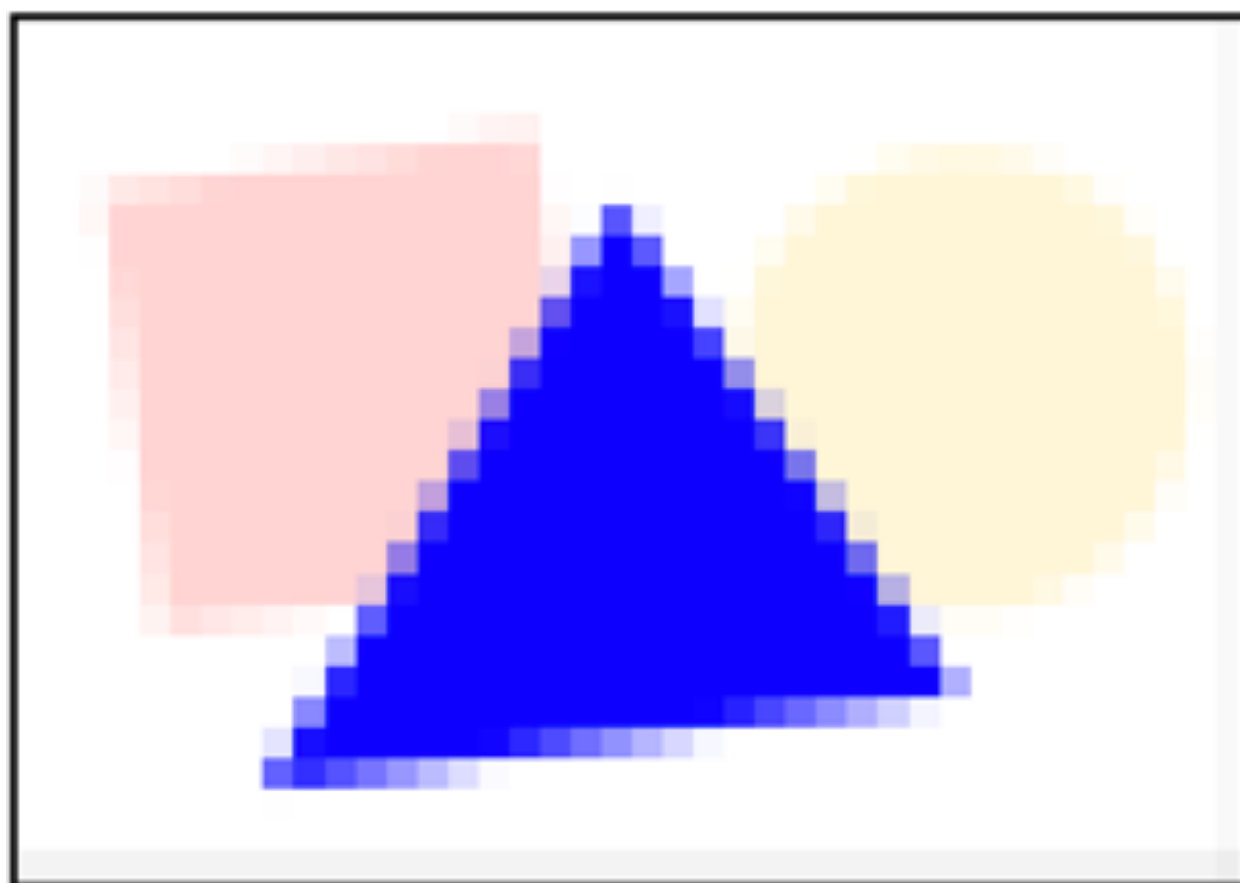


image

# Ray Tracing vs. Rasterization—Order

- Both rasterization & ray tracing will generate an image
- What's the difference?
- One basic difference: order in which we process samples

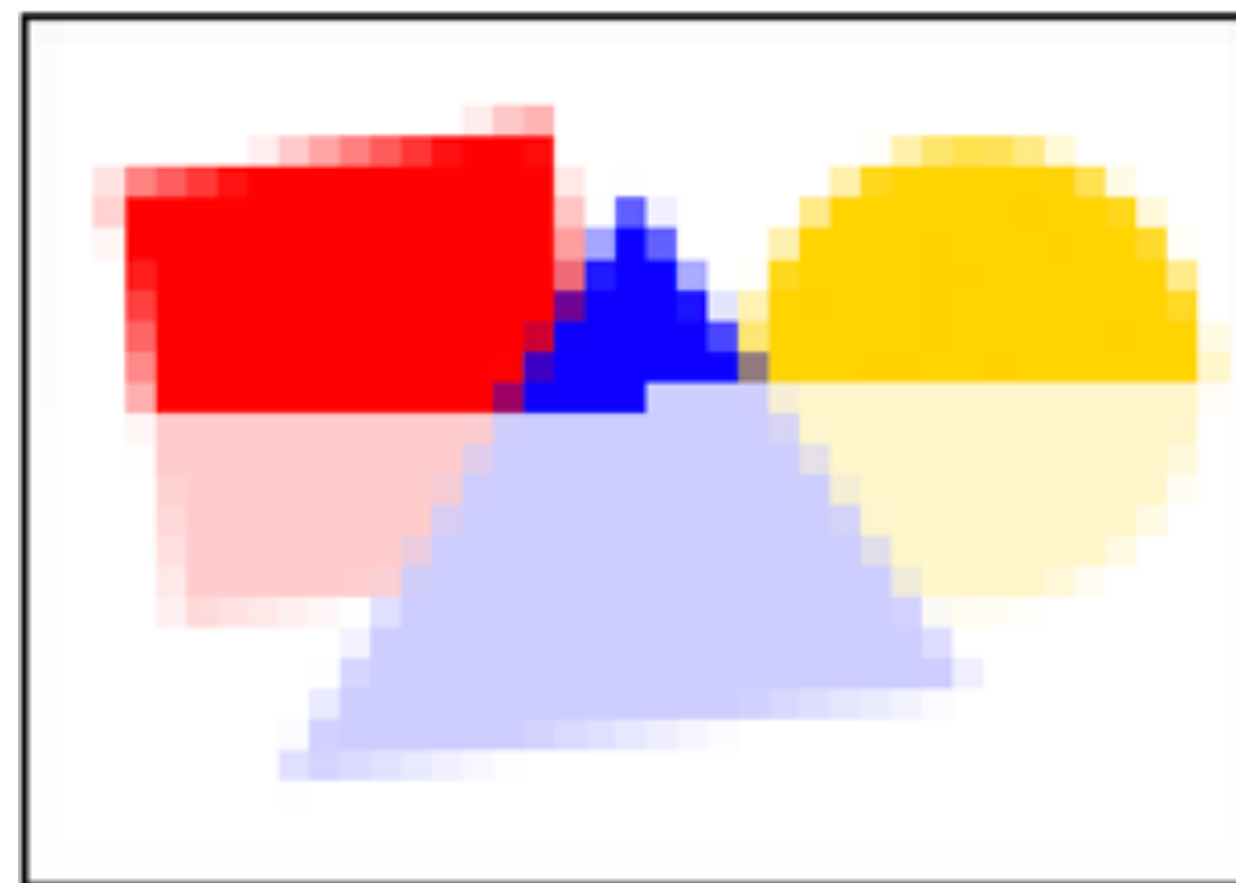
RASTERIZATION



for each **primitive**:  
  for each **sample**:  
    determine coverage  
    evaluate color

(Use Z-buffer to determine  
which primitive is visible)

RAY TRACING



for each **sample**:  
  for each **primitive**:  
    determine coverage  
    evaluate color

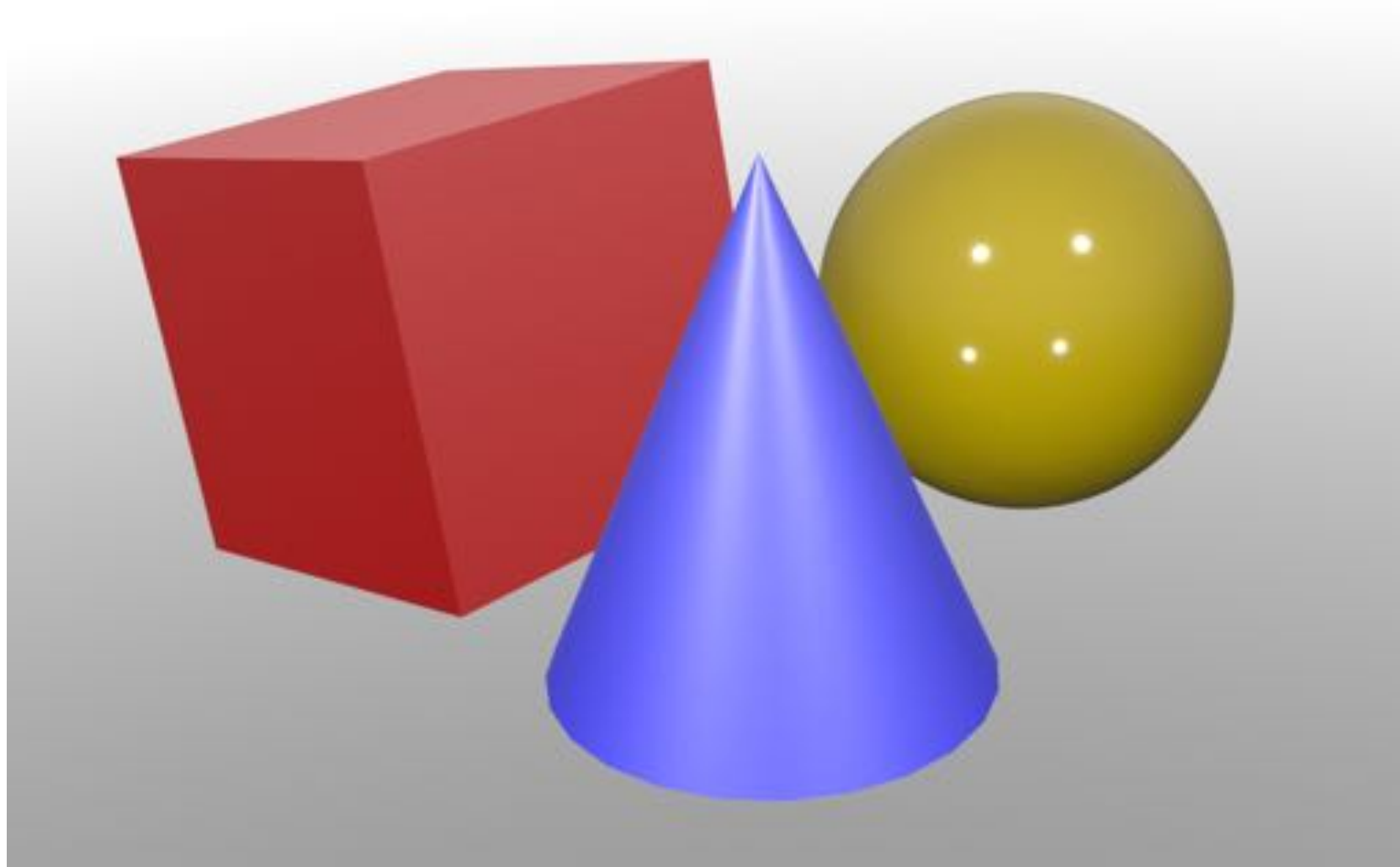
(Use spatial data structure like BVH to  
determine which primitive is visible)



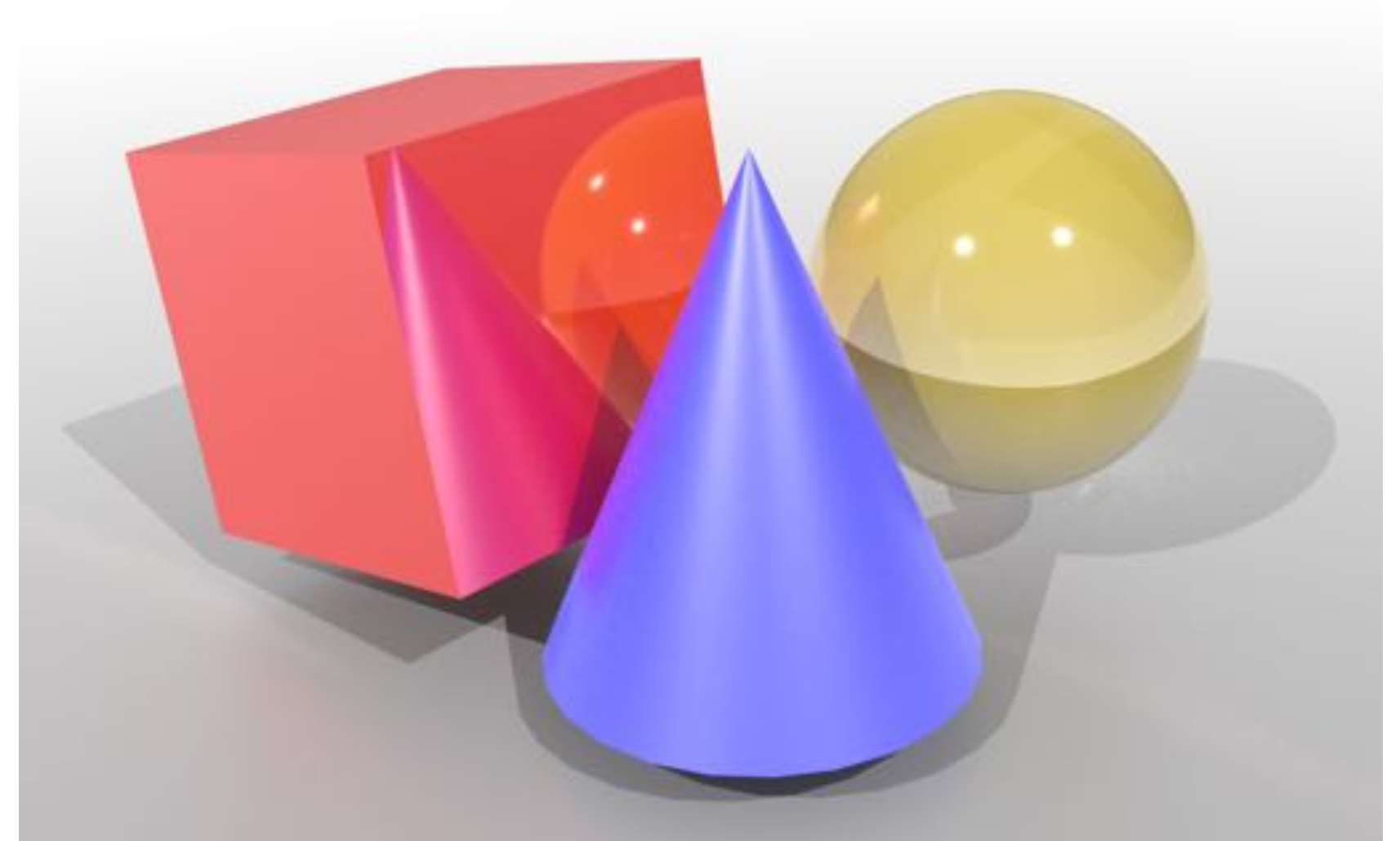
# Ray Tracing vs. Rasterization—Illumination

- More major difference: sophistication of illumination model
  - [LOCAL] rasterizer processes one primitive at a time; hard\* to determine things like “A is in the shadow of B”
  - [GLOBAL] ray tracer processes on ray at a time; ray knows about everything it intersects, easy to talk about shadows & other “global” illumination effects

RASTERIZATION



RAY TRACING

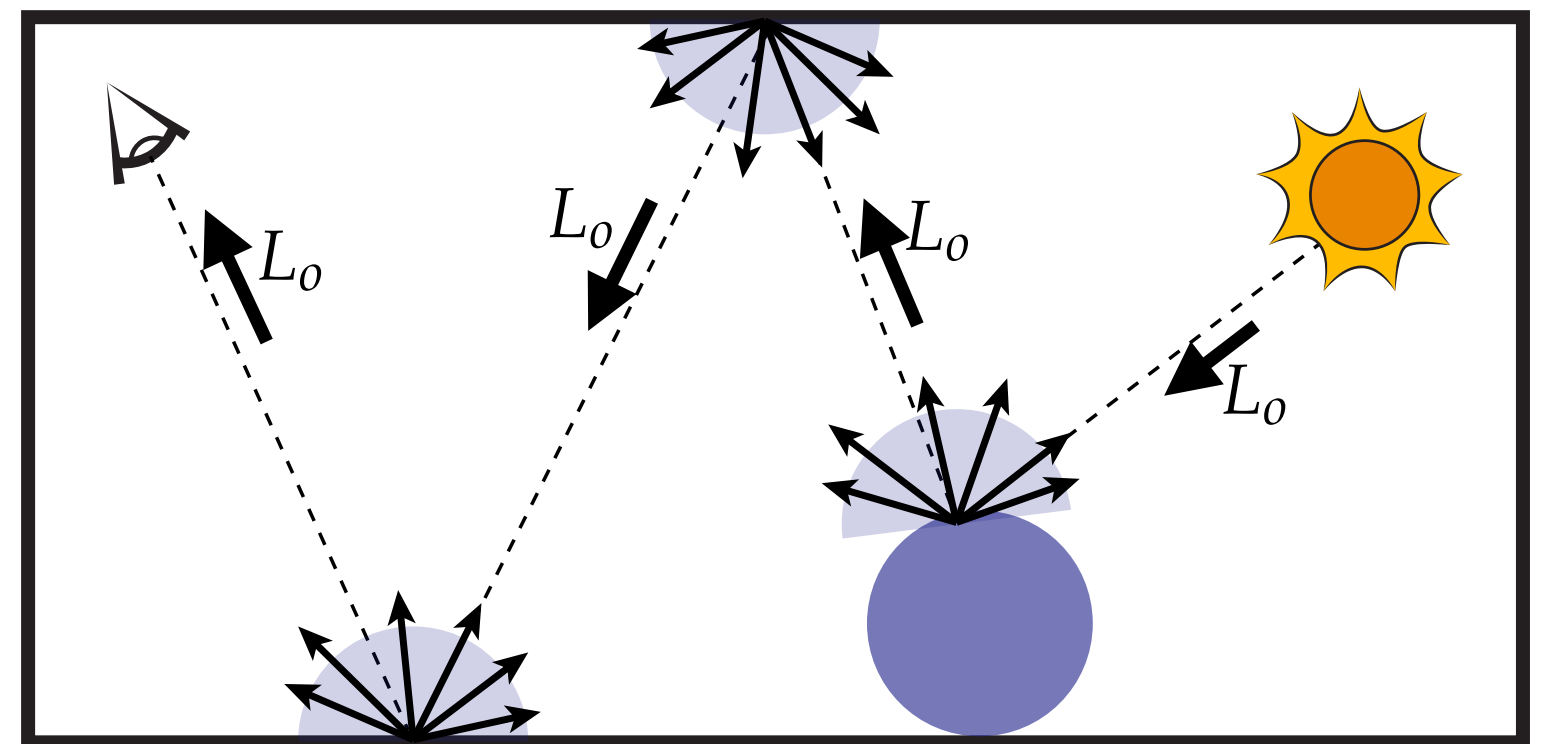


**Q: What illumination effects are missing from the image on the left?**

\*But not impossible to do some things with rasterization (e.g., shadow maps)... just results in more complexity

# Monte Carlo Ray Tracing

- To develop a full-blown photorealistic ray tracer, will need to apply Monte Carlo integration to the rendering equation
- To determine color of each pixel, integrate incoming light
- What function are we integrating?
  - illumination along different paths of light
- What does a “sample” mean in this context?
  - each path we trace is a sample



$$L_o(\mathbf{p}, \omega_o) = L_e(\mathbf{p}, \omega_o) + \int_{\mathcal{H}^2} f_r(\mathbf{p}, \omega_i \rightarrow \omega_o) L_i(\mathbf{p}, \omega_i) \cos \theta d\omega_i$$

# Monte Carlo Integration

- Started looking at Monte Carlo integration in our lecture on numerical integration
- Basic idea: take average of random samples
- Will need to flesh this idea out with some key concepts:
  - **EXPECTED VALUE** — what value do we get on average?
  - **VARIANCE** — what's the expected deviation from the average?
  - **IMPORTANCE SAMPLING** — how do we (correctly) take more samples in more important regions?

$$\lim_{N \rightarrow \infty} \frac{|\Omega|}{N} \sum_{i=1}^N f(X_i) = \int_{\Omega} f(x) dx$$

# Expected Value

**Intuition: what value does a random variable take, on average?**

- E.g., consider a fair coin where heads = 1, tails = 0
- Equal probability of heads & tails (1/2 for both)
- Expected value is then  $(1/2) \cdot 1 + (1/2) \cdot 0 = 1/2$

expected value of random variable  $Y$

number of possible outcomes

$$E(Y) := \sum_{i=1}^k p_i y_i$$

probability of  $i$ th outcome

value of  $i$ th outcome

**Properties of expectation:**

$$E \left[ \sum_i Y_i \right] = \sum_i E[Y_i]$$

$$E[aY] = aE[Y]$$

**(Can you show these are true?)**



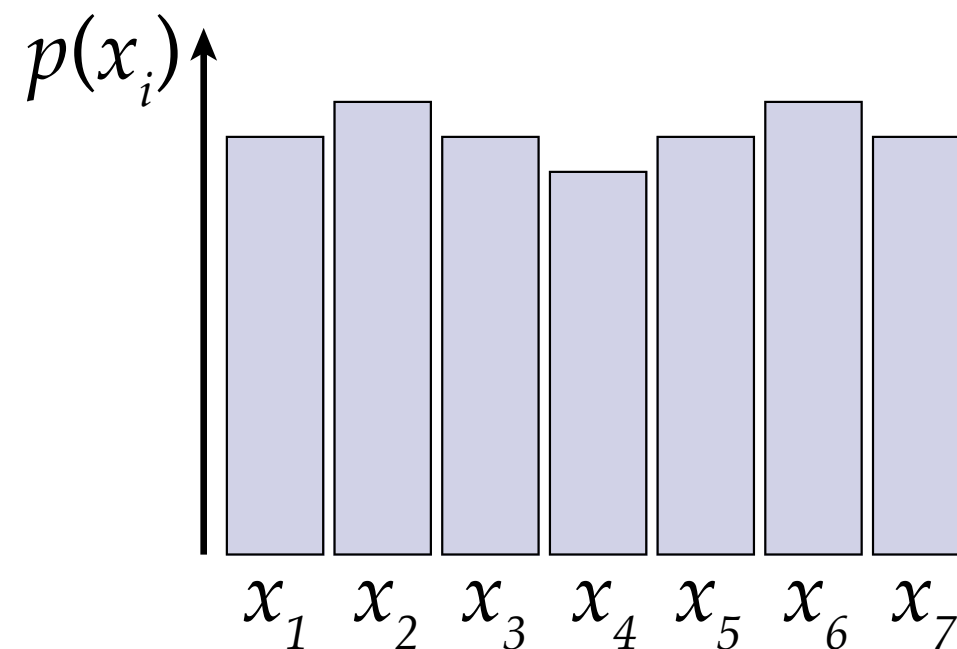
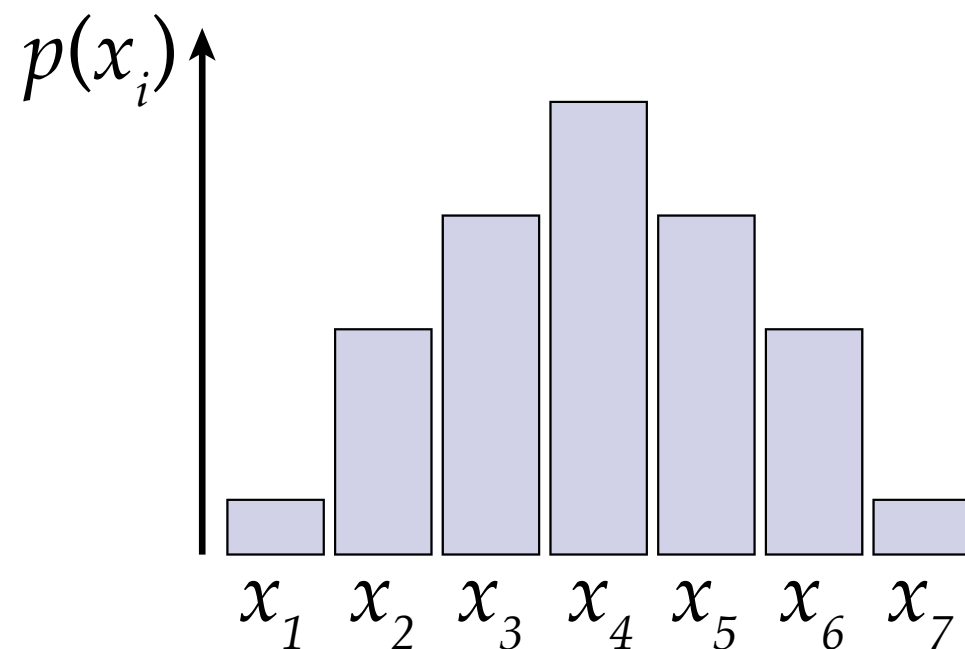
# Variance

**Intuition: how far are our samples from the average, on average?**

## Definition

$$V[Y] = E[(Y - E[Y])^2]$$

**Q: Which of these has higher variance?**



## Properties of variance:

$$V[Y] = E[Y^2] - E[Y]^2$$

$$V\left[\sum_{i=1}^N Y_i\right] = \sum_{i=1}^N V[Y_i]$$

$$V[aY] = a^2 V[Y]$$

**(Can you show these are true?)**

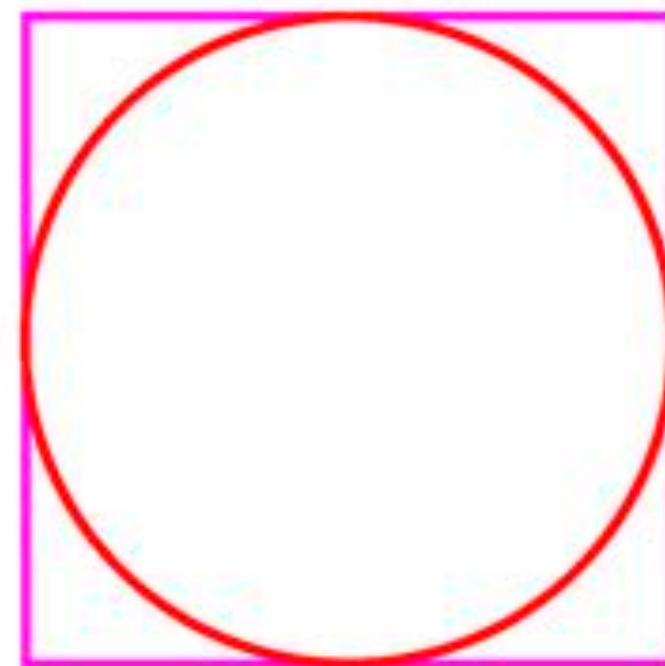
# Law of Large Numbers

- **Important fact: for any random variable, the average value of  $N$  trials approaches the expected value as we increase  $N$**
- **Decrease in variance is always linear in  $N$ :**

$$V \left[ \frac{1}{N} \sum_{i=1}^N Y_i \right] = \frac{1}{N^2} \sum_{i=1}^N V[Y_i] = \frac{1}{N^2} N V[Y] = \frac{1}{N} V[Y]$$

**Consider a coconut...**

| nCoconuts | estimate of $\pi$ |
|-----------|-------------------|
| 1         | 4.000000          |
| 10        | 3.200000          |
| 100       | 3.240000          |
| 1000      | 3.112000          |
| 10000     | 3.163600          |
| 100000    | 3.139520          |
| 1000000   | 3.141764          |



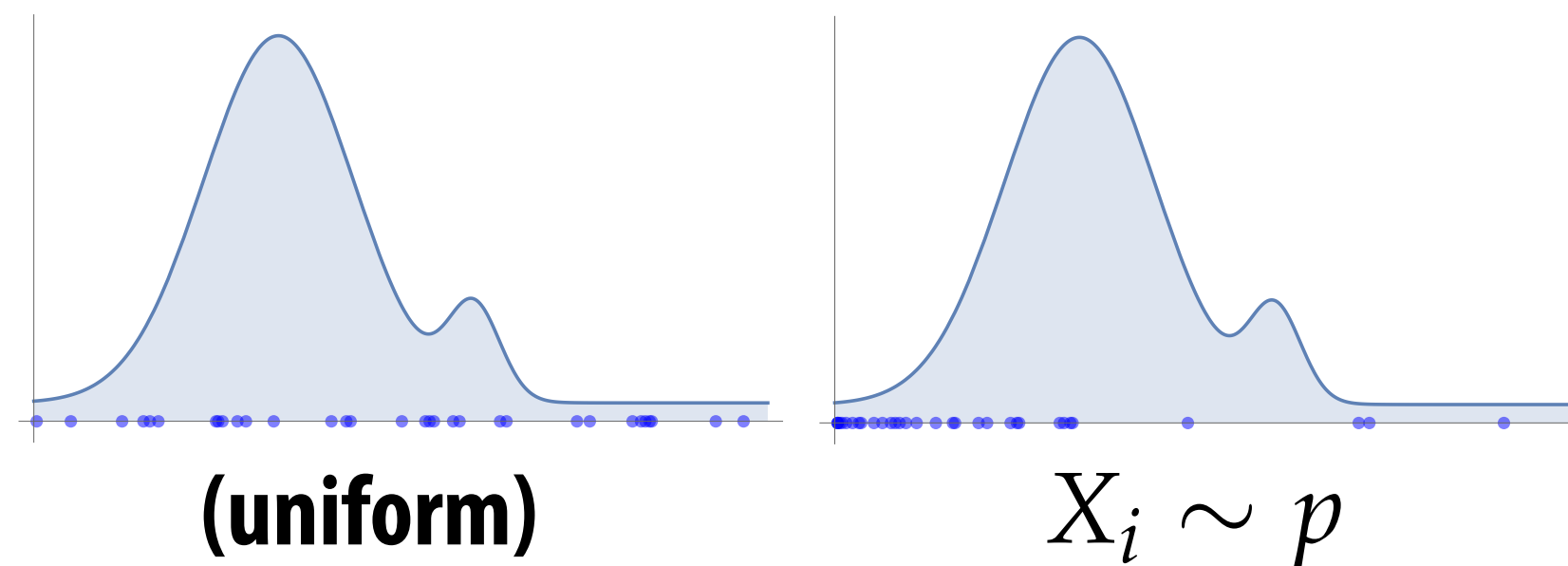
**Q: Why is the law of large numbers important for Monte Carlo ray tracing?**

**A: No matter how hard the integrals are (crazy lighting, geometry, materials, etc.), can always\* get the right image by taking more samples.**

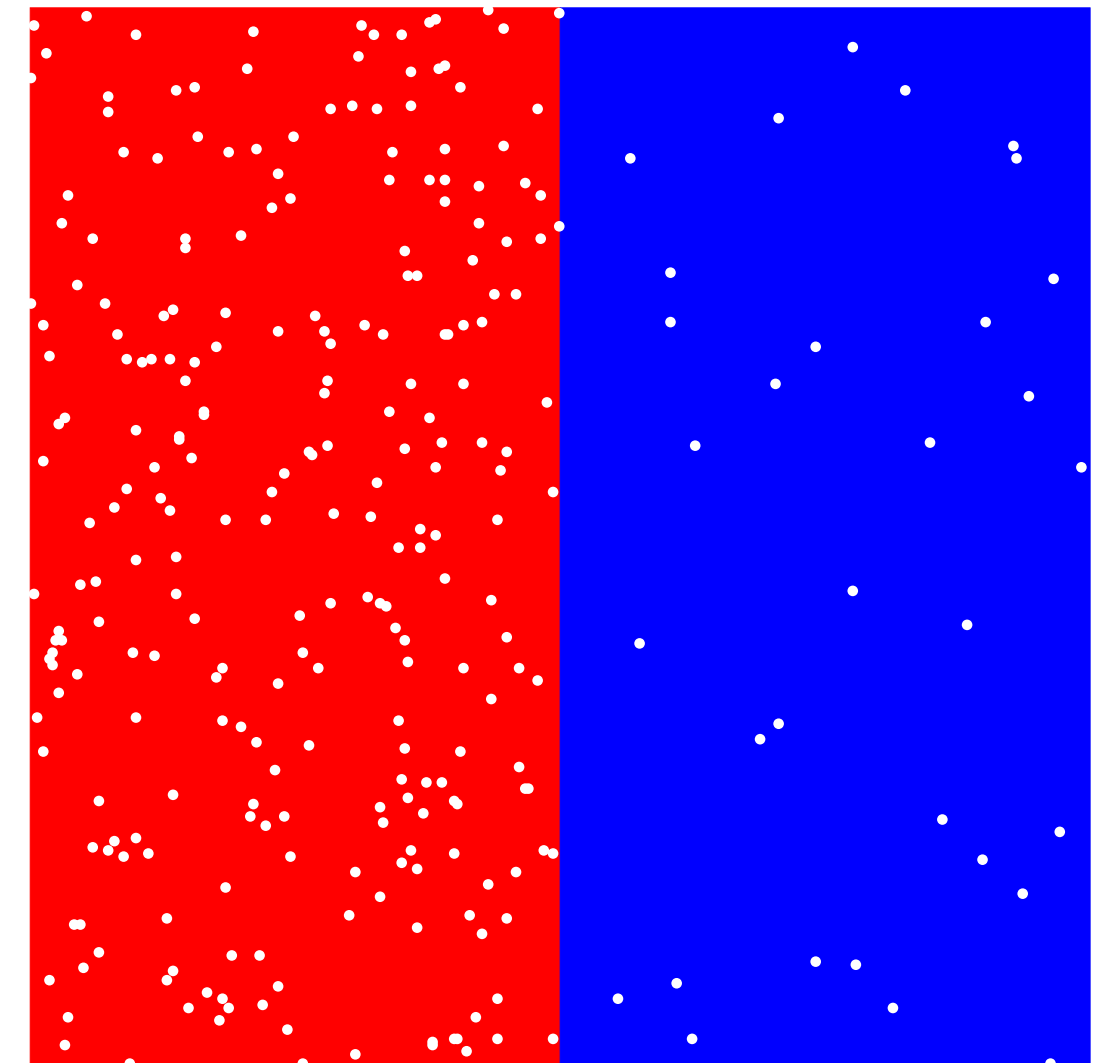
**\*As long as we make sure to sample all possible kinds of light paths...**

# Biasing

- So far, we've picked samples uniformly from the domain (every point is equally likely)
- Suppose we pick samples from some other distribution (more samples in one place than another)
- Q: Can we still use samples  $f(X_i)$  to get a (correct) estimate of our integral?
- A: Sure! Just weight contribution of each sample by how likely we were to pick it
- Q: Are we correct to divide by  $p$ ? Or... should we multiply instead?
- A: Think about a simple example where we sample RED region 8x as often as BLUE region
  - average color over square should be purple
  - if we multiply, average will be TOO RED
  - if we divide, average will be JUST RIGHT



$$\int_{\Omega} f(x) dx \approx \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)}$$



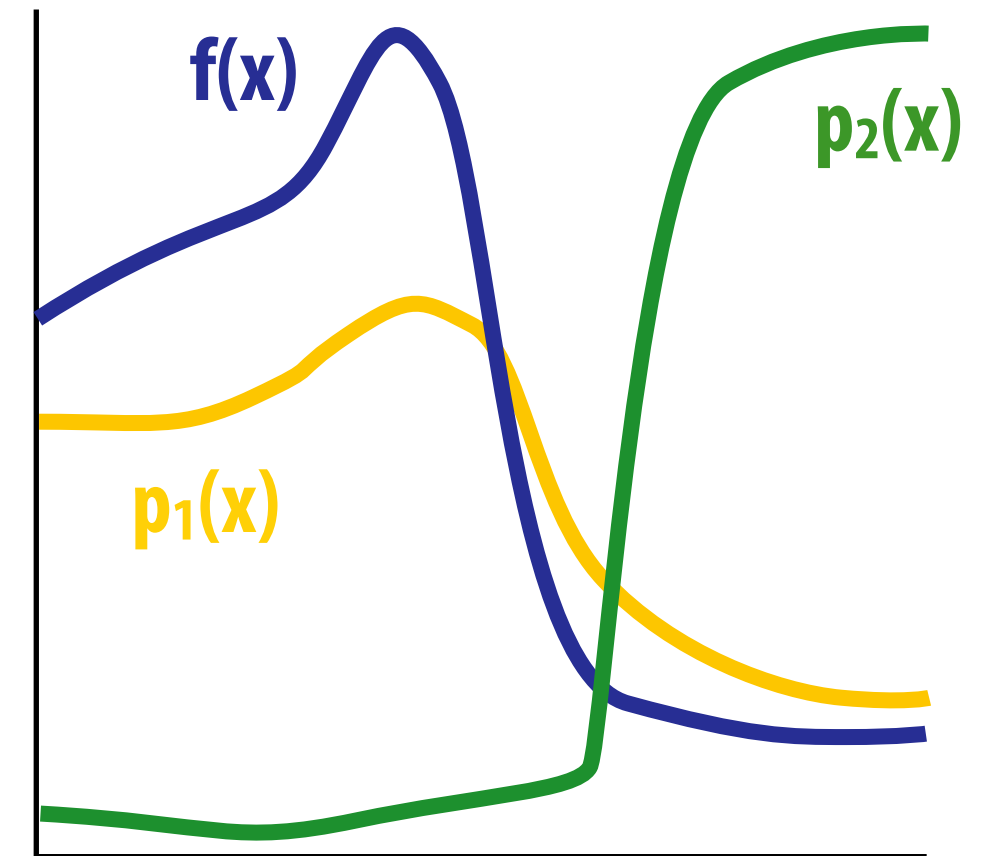


# Importance sampling

Q: Ok, so then WHERE is the best place to take samples?

Think:

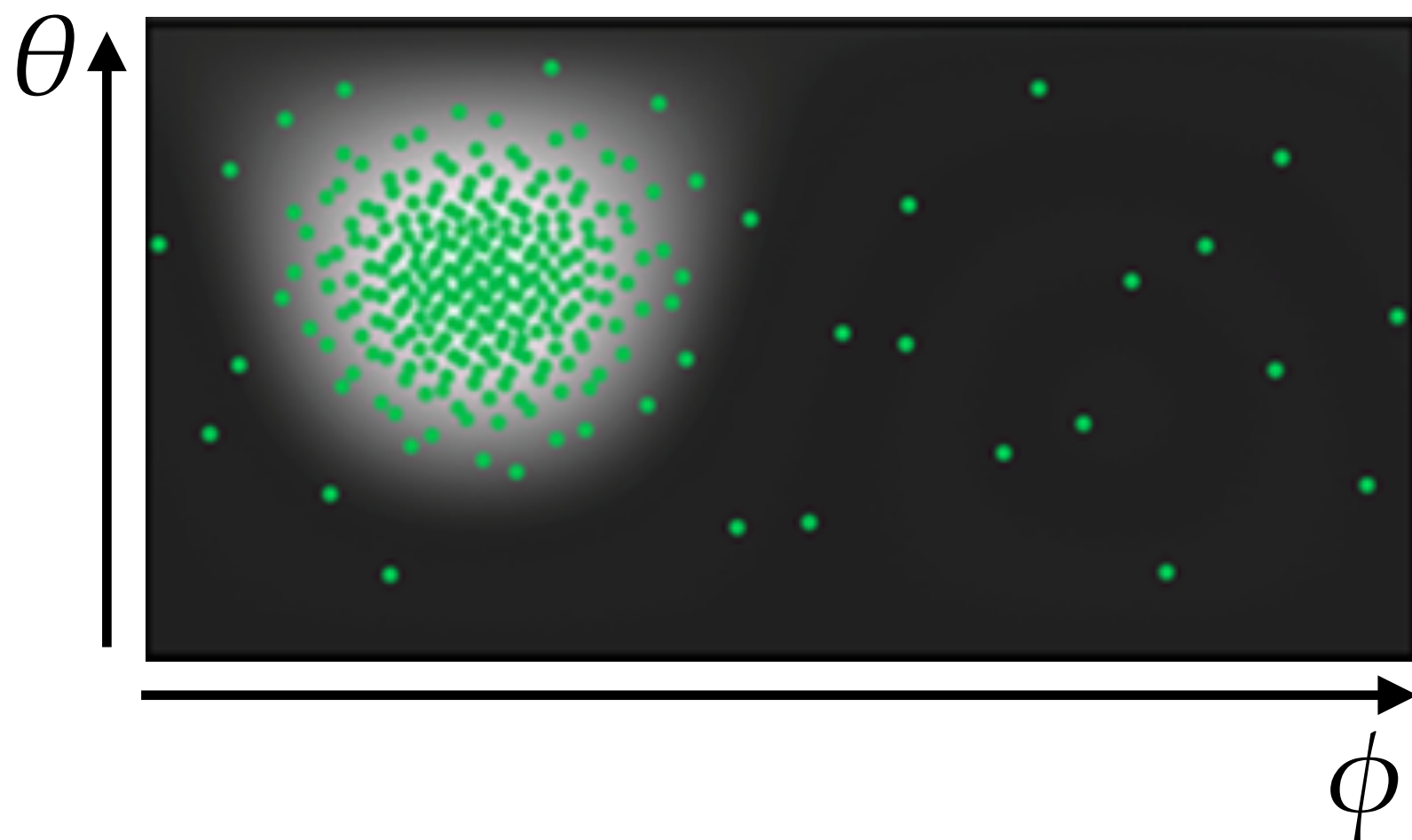
- What is the behavior of  $f(x)/p_1(x)$ ?  $f(x)/p_2(x)$ ?
- How does this impact the variance of the estimator?



Idea: put more where integrand is large (“most useful samples”). E.g.:

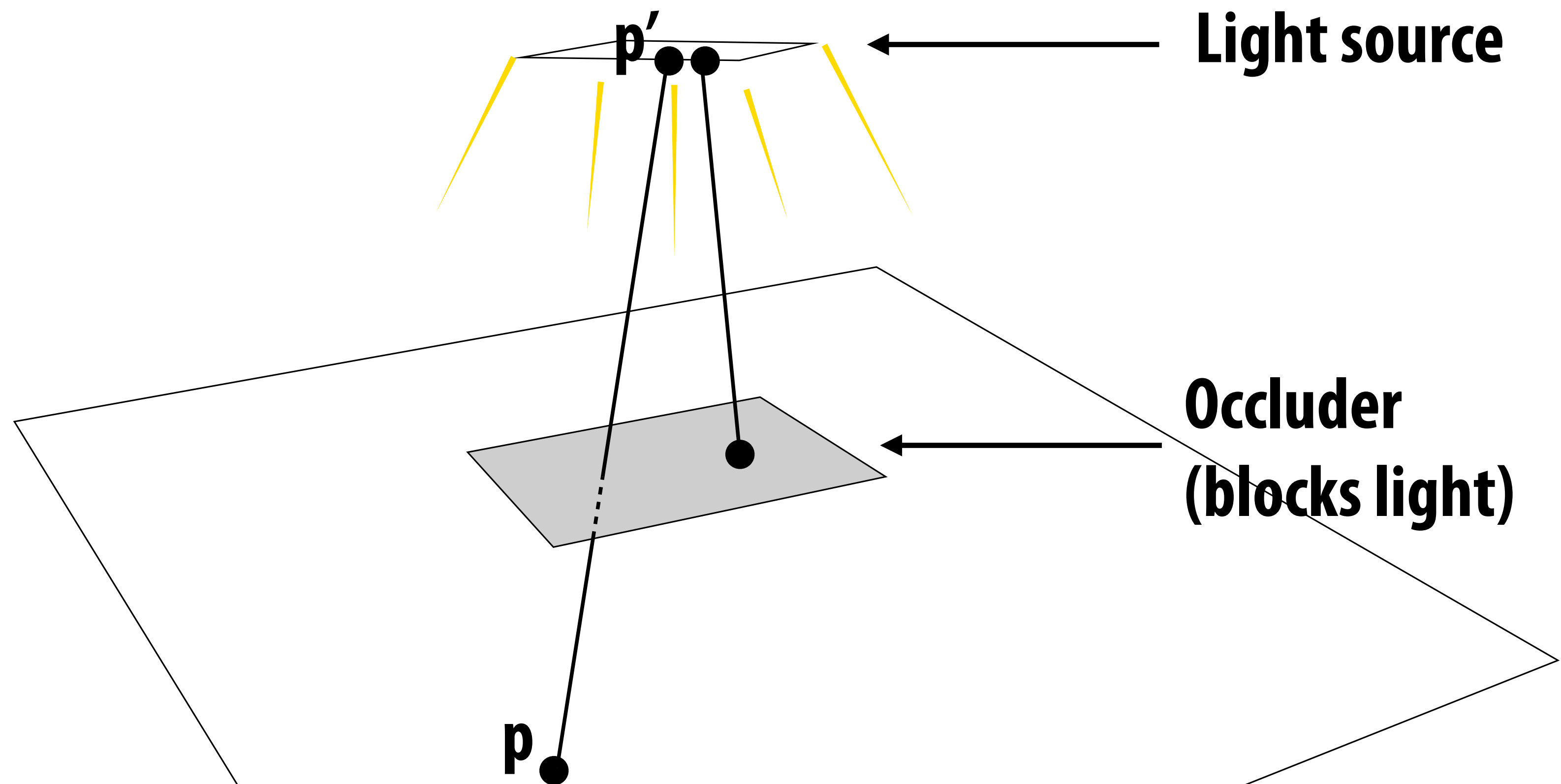
(image-based lighting)

(BRDF)





# Example: Direct Lighting



Visibility function:

$$V(p, p') = \begin{cases} 1, & p \text{ "sees" } p' \\ 0, & \text{otherwise} \end{cases}$$

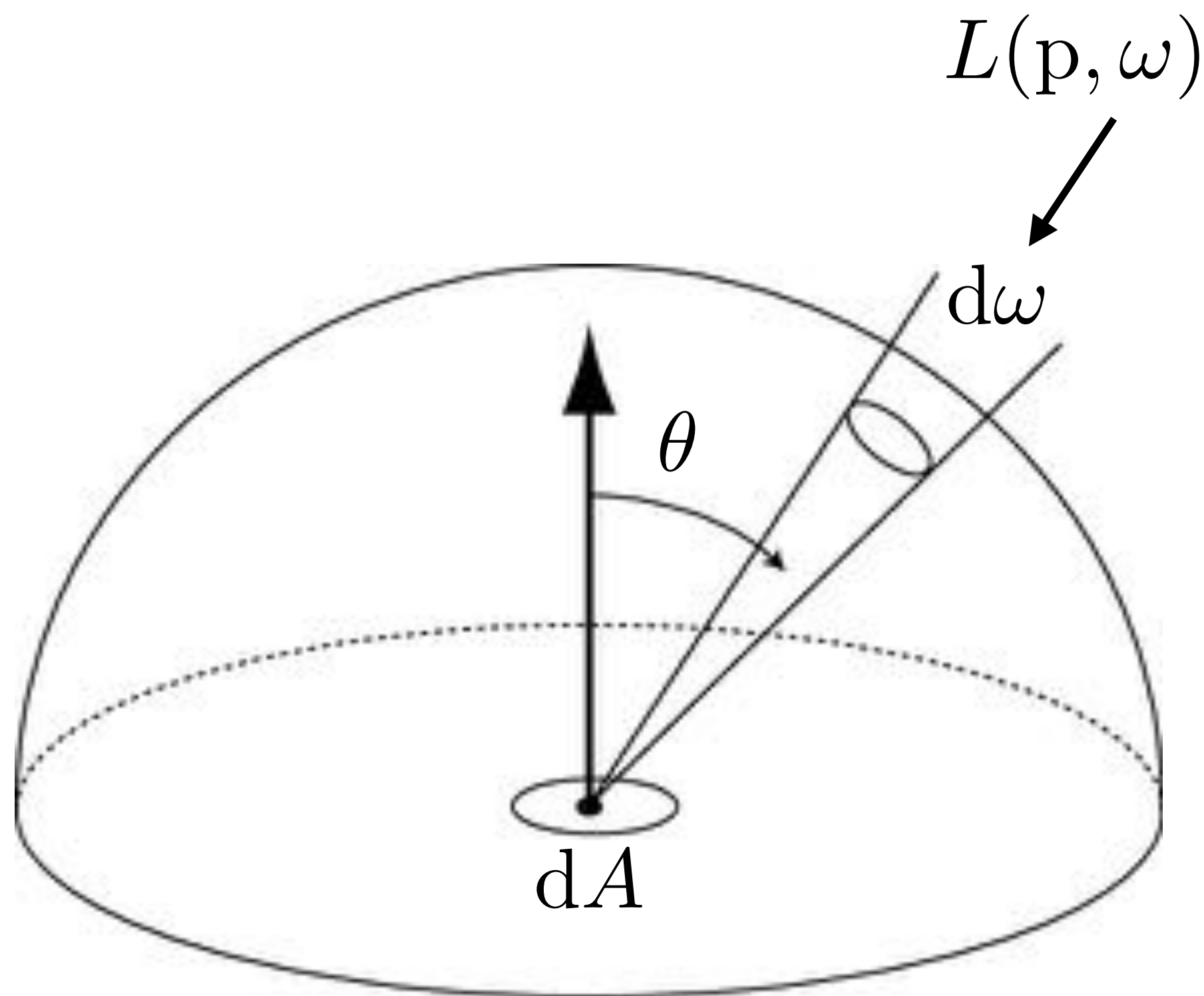
**How bright is each point on the ground?**

# Direct lighting—uniform sampling

Uniformly-sample hemisphere of directions with respect to solid angle

$$p(\omega) = \frac{1}{2\pi}$$

$$E(p) = \int L(p, \omega) \cos \theta \, d\omega$$



**Estimator:**

$$X_i \sim p(\omega)$$

$$Y_i = f(X_i)$$

$$Y_i = L(p, \omega_i) \cos \theta_i$$

$$F_N = \frac{2\pi}{N} \sum_{i=1}^N Y_i$$

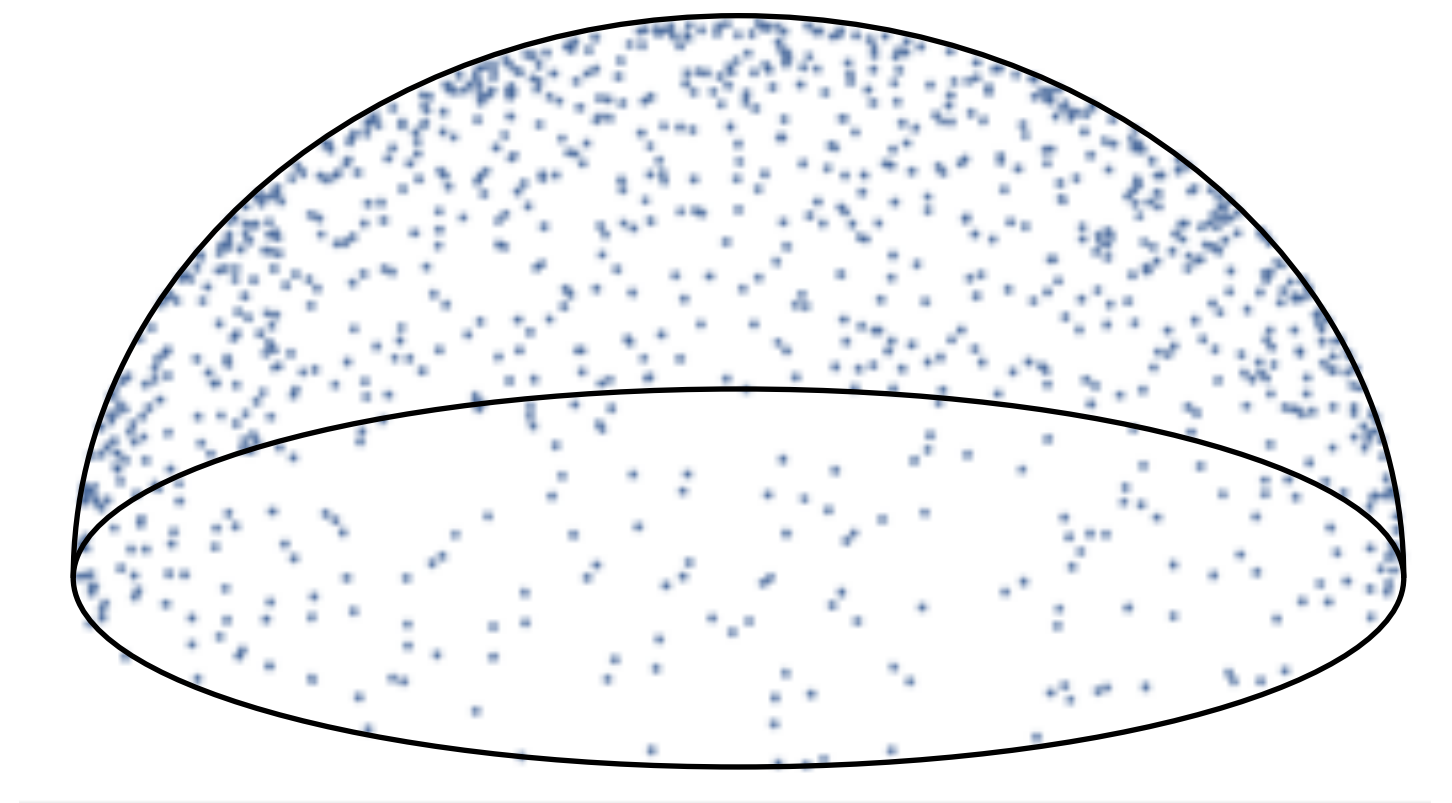
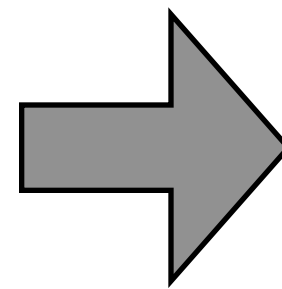
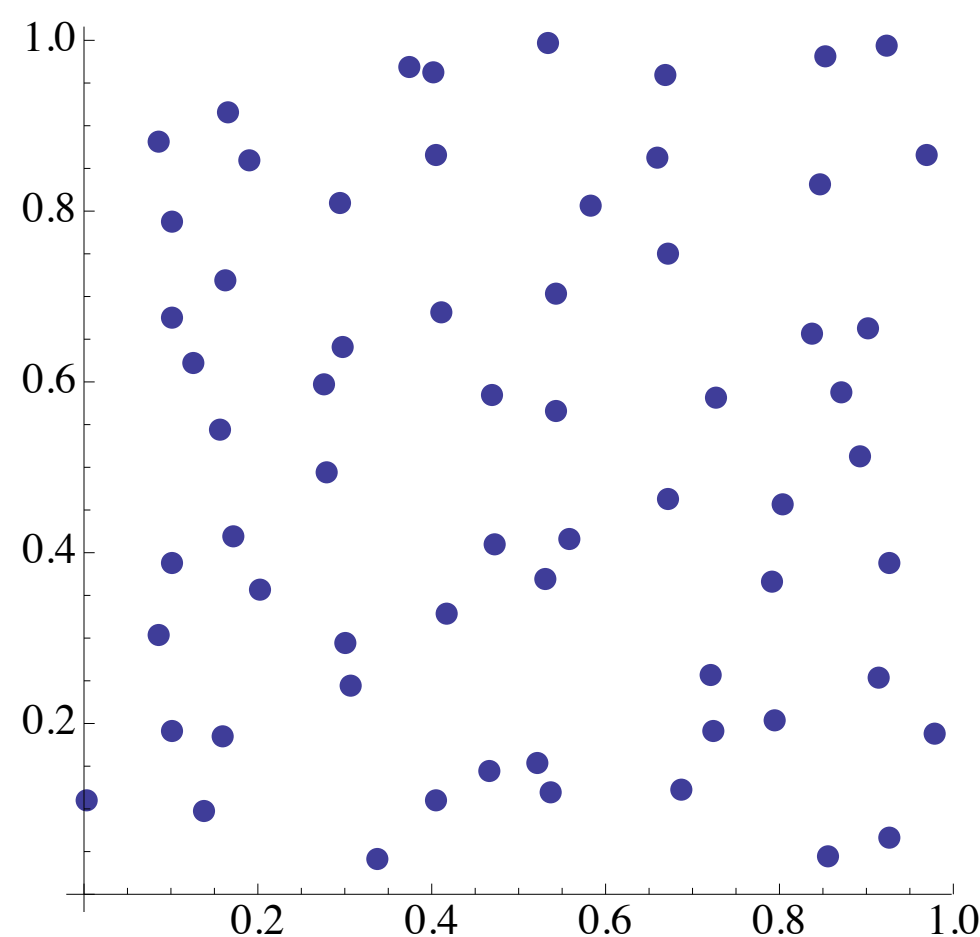
# Aside: Picking points on unit hemisphere

How do we uniformly sample directions from the hemisphere?

One way: use rejection sampling. (How?)

Another way: “warp” two values in  $[0,1]$  via the inversion method:

$$(\xi_1, \xi_2) = (\sqrt{1 - \xi_1^2} \cos(2\pi\xi_2), \sqrt{1 - \xi_1^2} \sin(2\pi\xi_2), \xi_1)$$



**Exercise: derive from the inversion method**

# Direct lighting—uniform sampling (algorithm)

Uniformly-sample hemisphere of directions with respect to solid angle

$$p(\omega) = \frac{1}{2\pi}$$

$$E(p) = \int L(p, \omega) \cos \theta \, d\omega$$

Given surface point  $p$

A ray tracer evaluates radiance along a ray  
(see `Raytracer::trace_ray()` in `raytracer.cpp`)

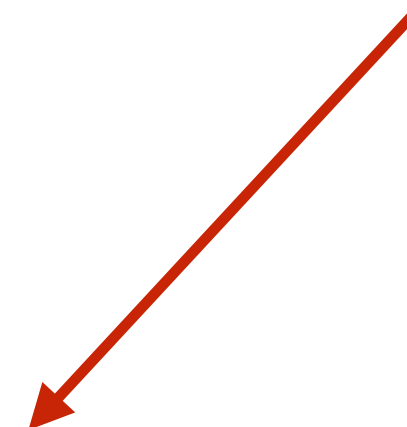
For each of  $N$  samples:

Generate random direction:  $\omega_i$

Compute incoming radiance arriving  $L_i$  at  $p$  from direction:  $\omega_i$

Compute incident irradiance due to ray:  $dE_i = L_i \cos \theta_i$

Accumulate  $\frac{2\pi}{N} dE_i$  into estimator



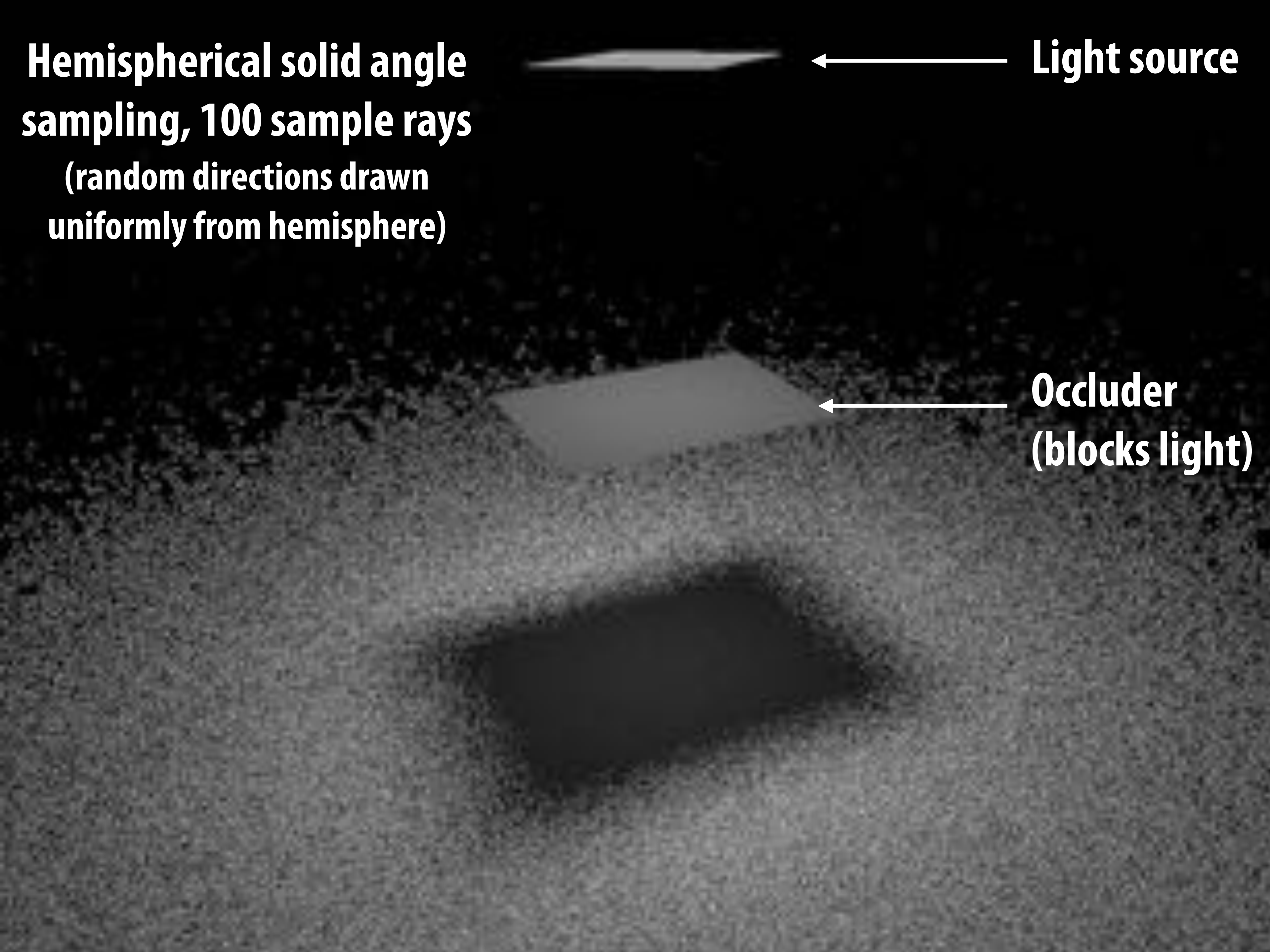
**Hemispherical solid angle  
sampling, 100 sample rays  
(random directions drawn  
uniformly from hemisphere)**



**Light source**

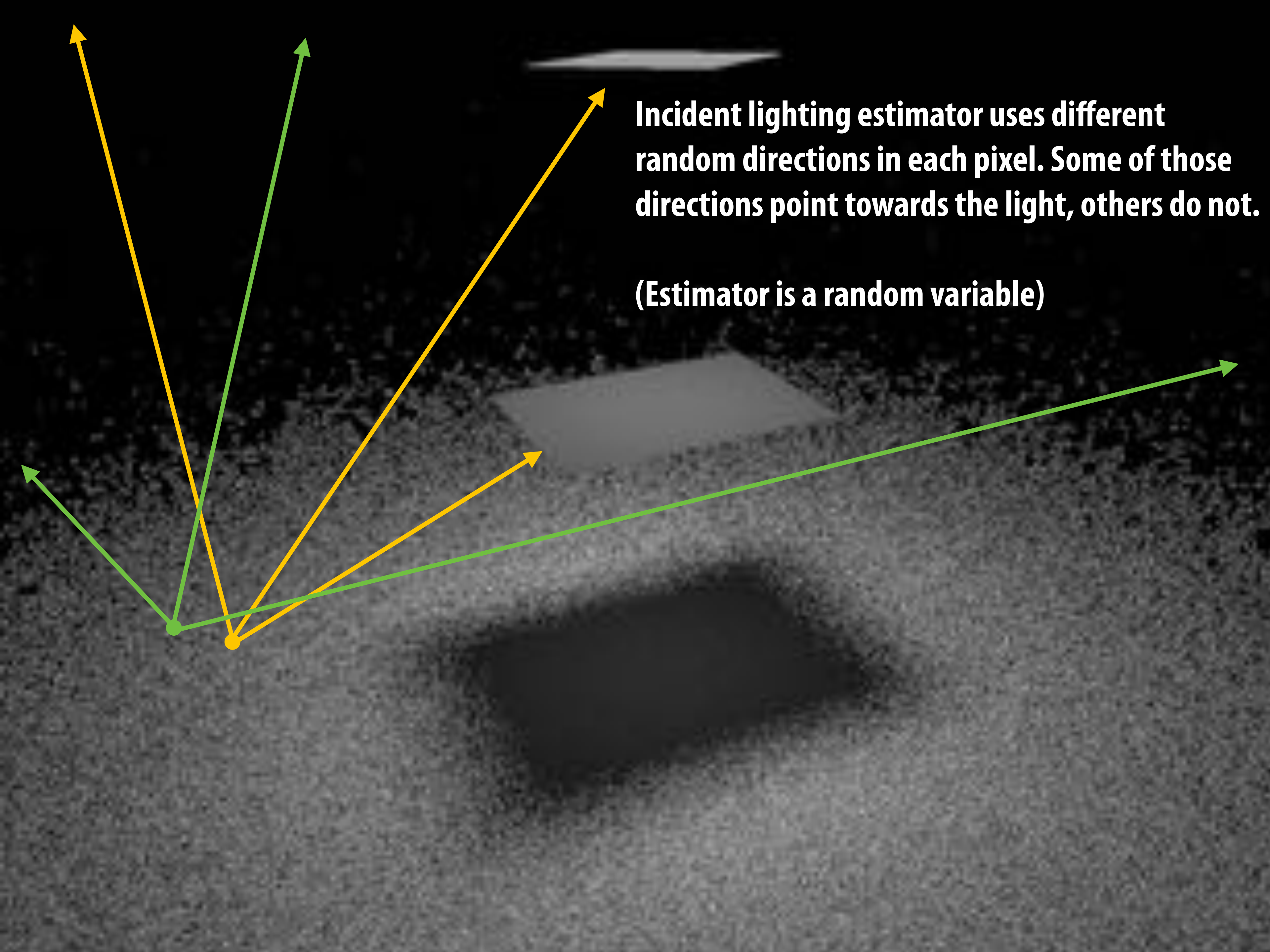


**Occluder  
(blocks light)**





**Why is the image in the previous slide “noisy”?**



**Incident lighting estimator uses different random directions in each pixel. Some of those directions point towards the light, others do not.**

**(Estimator is a random variable)**

**How can we reduce noise?**

**One idea: just take more samples!**

**Another idea:**

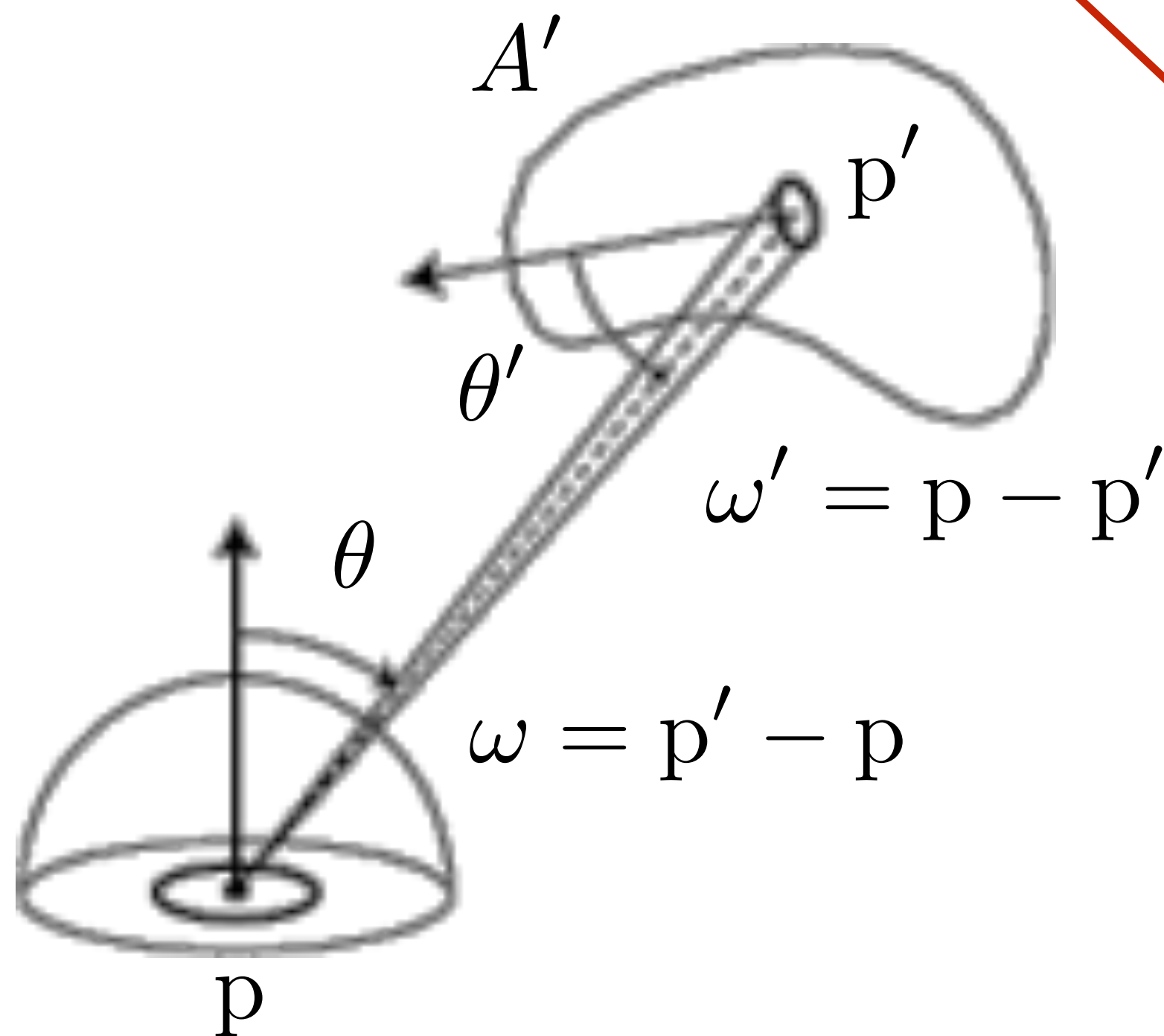
- **Don't need to integrate over entire hemisphere of directions (incoming radiance is 0 from most directions).**
- **Just integrate over the area of the light (directions where incoming radiance is non-zero) and weight appropriately**

# Direct lighting: area integral

$$E(p) = \int L(p, \omega) \cos \theta \, d\omega \quad \leftarrow \text{Previously: just integrate over all directions}$$

$$E(p) = \int_{A'} L_o(p', \omega') V(p, p') \frac{\cos \theta \cos \theta'}{|p - p'|^2} \, dA' \quad \leftarrow \text{Change of variables to integrate over area of light *}$$

$$d\omega = \frac{dA}{|p' - p|^2} = \frac{dA' \cos \theta}{|p' - p|^2}$$



**Binary visibility function:**  
1 if  $p'$  is visible from  $p$ , 0 otherwise  
(accounts for light occlusion)

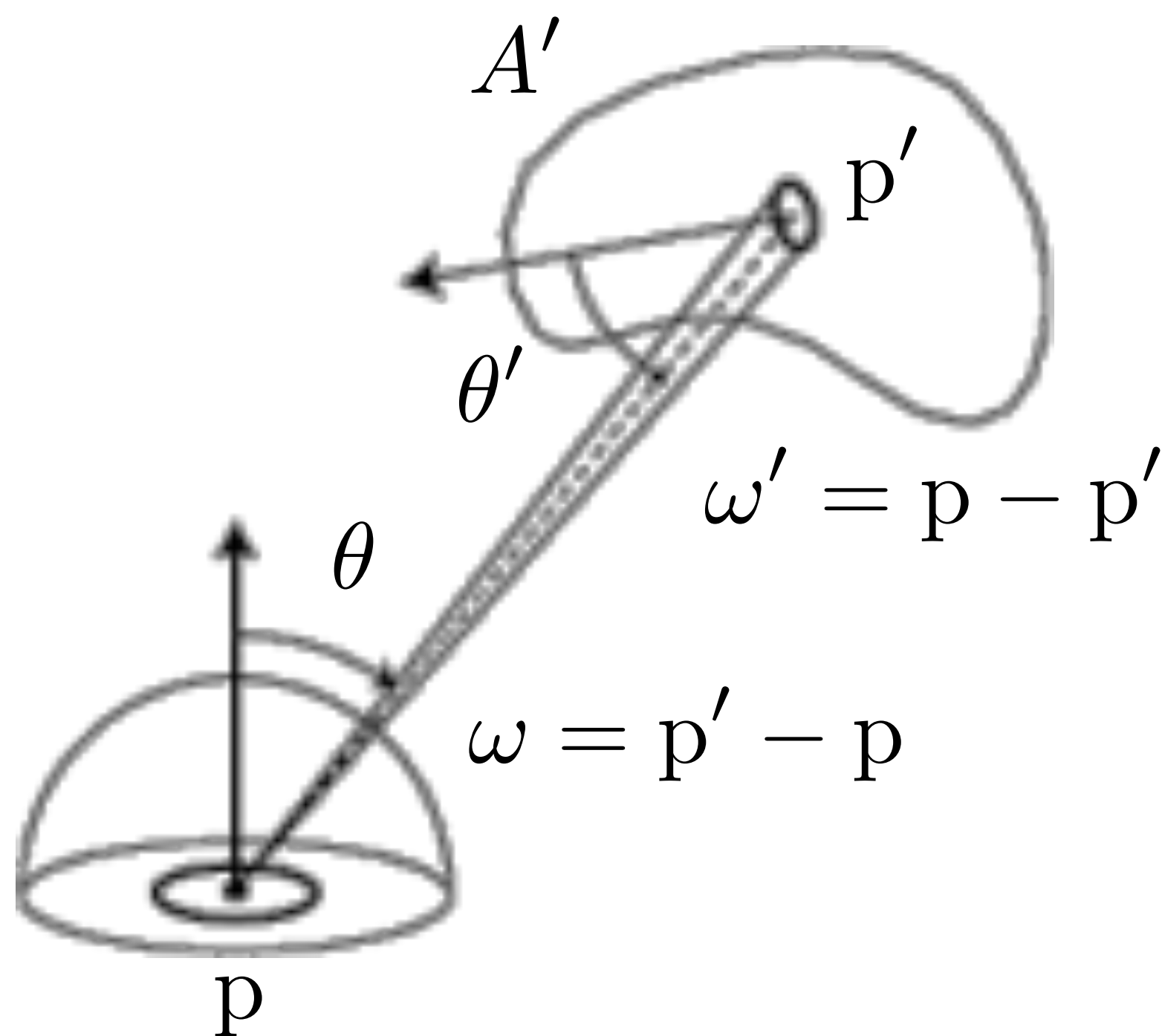
**Outgoing radiance from light point  $p$ , in direction  $w'$  towards  $p$**



# Direct lighting: area integral

$$E(p) = \int_{A'} L_o(p', \omega') V(p, p') \frac{\cos \theta \cos \theta'}{|p - p'|^2} dA'$$

**Sample shape uniformly by area  $A'$**

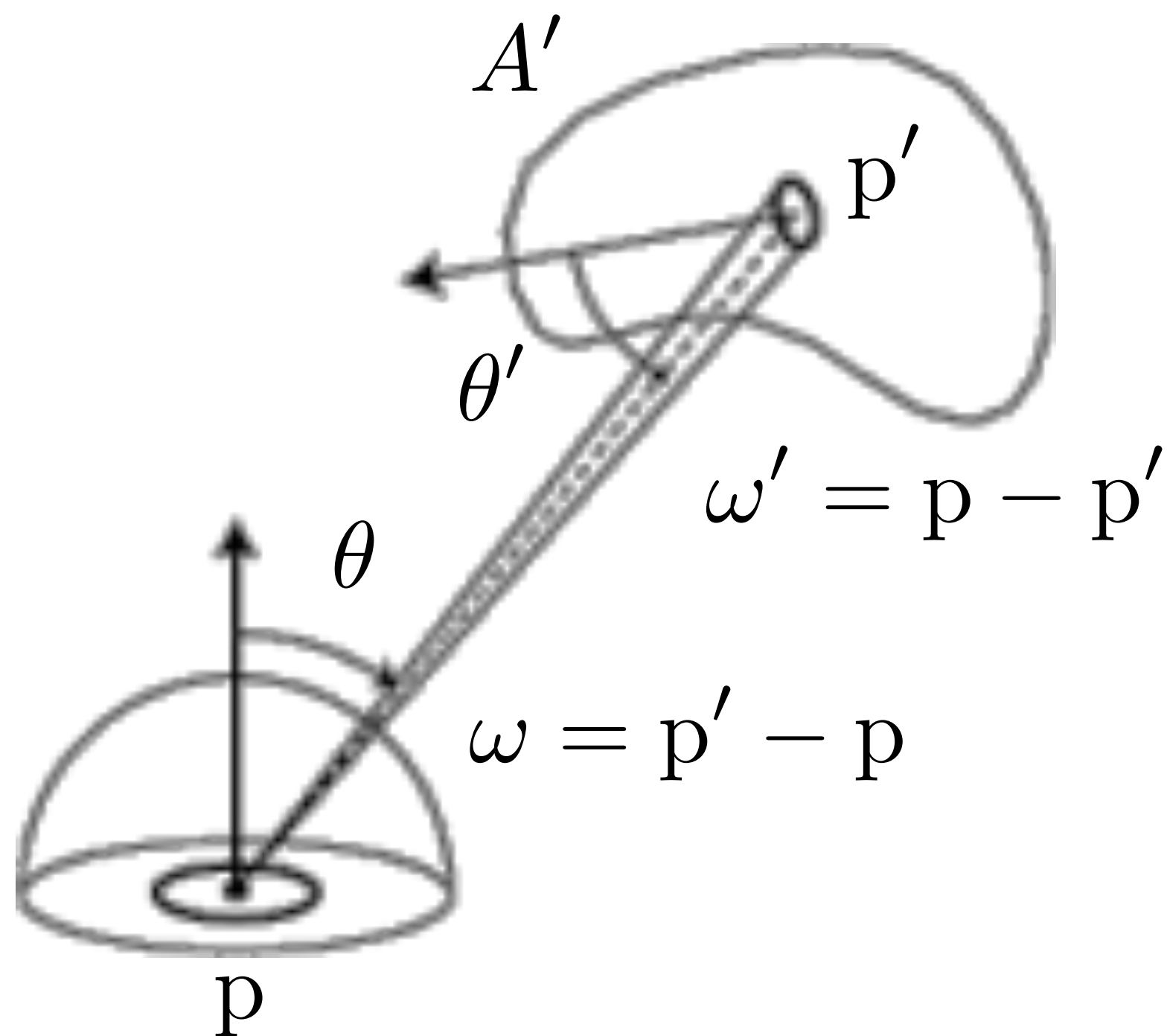


$$\int_{A'} p(p') dA' = 1$$

$$p(p') = \frac{1}{A'}$$

# Direct lighting: area integral

$$E(p) = \int_{A'} L_o(p', \omega') V(p, p') \frac{\cos \theta \cos \theta'}{|p - p'|^2} dA'$$



**Probability:**

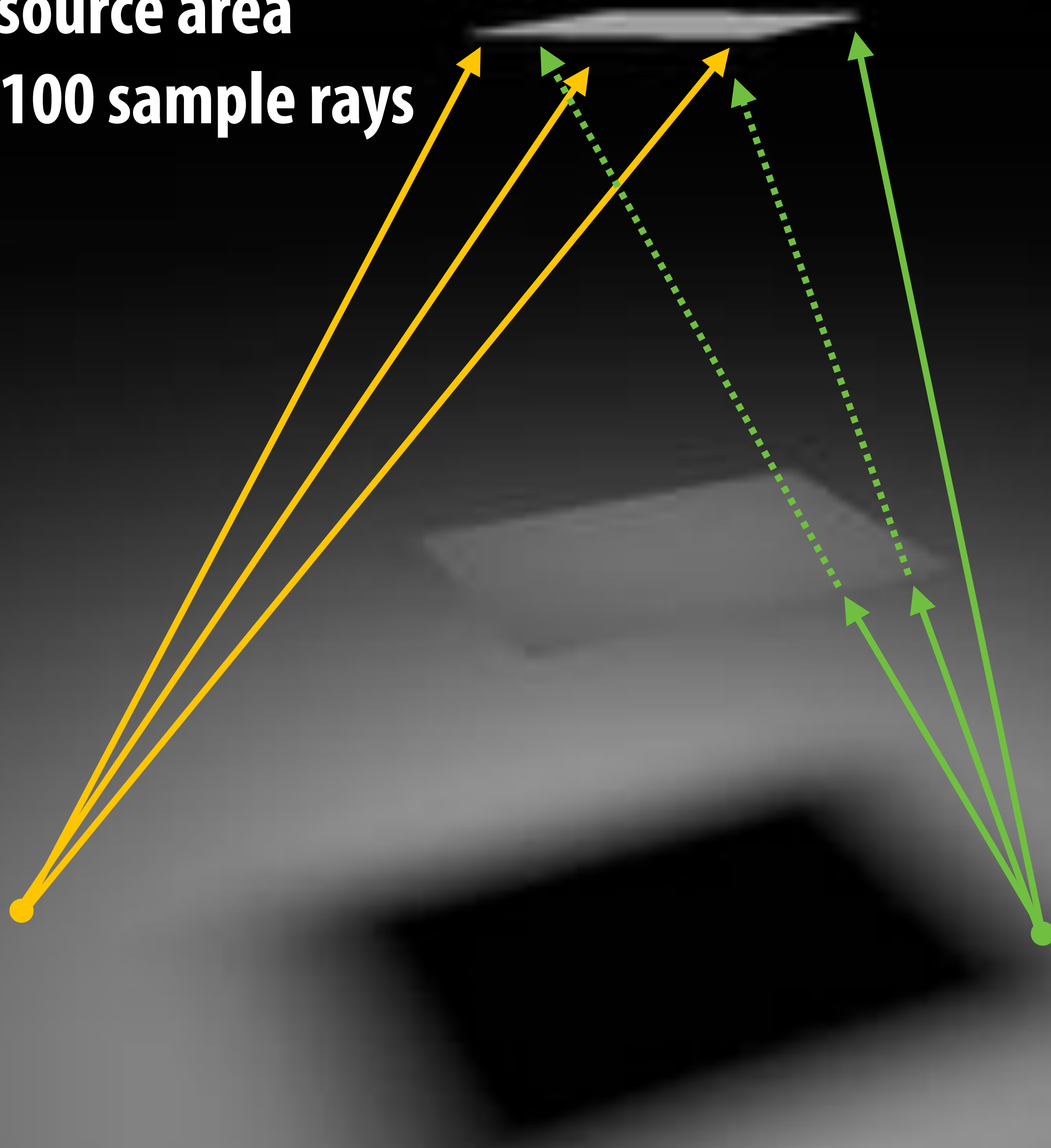
$$p(p') = \frac{1}{A'}$$

**Estimator**

$$Y_i = L_o(p'_i, \omega'_i) V(p, p'_i) \frac{\cos \theta_i \cos \theta'_i}{|p - p'_i|^2}$$

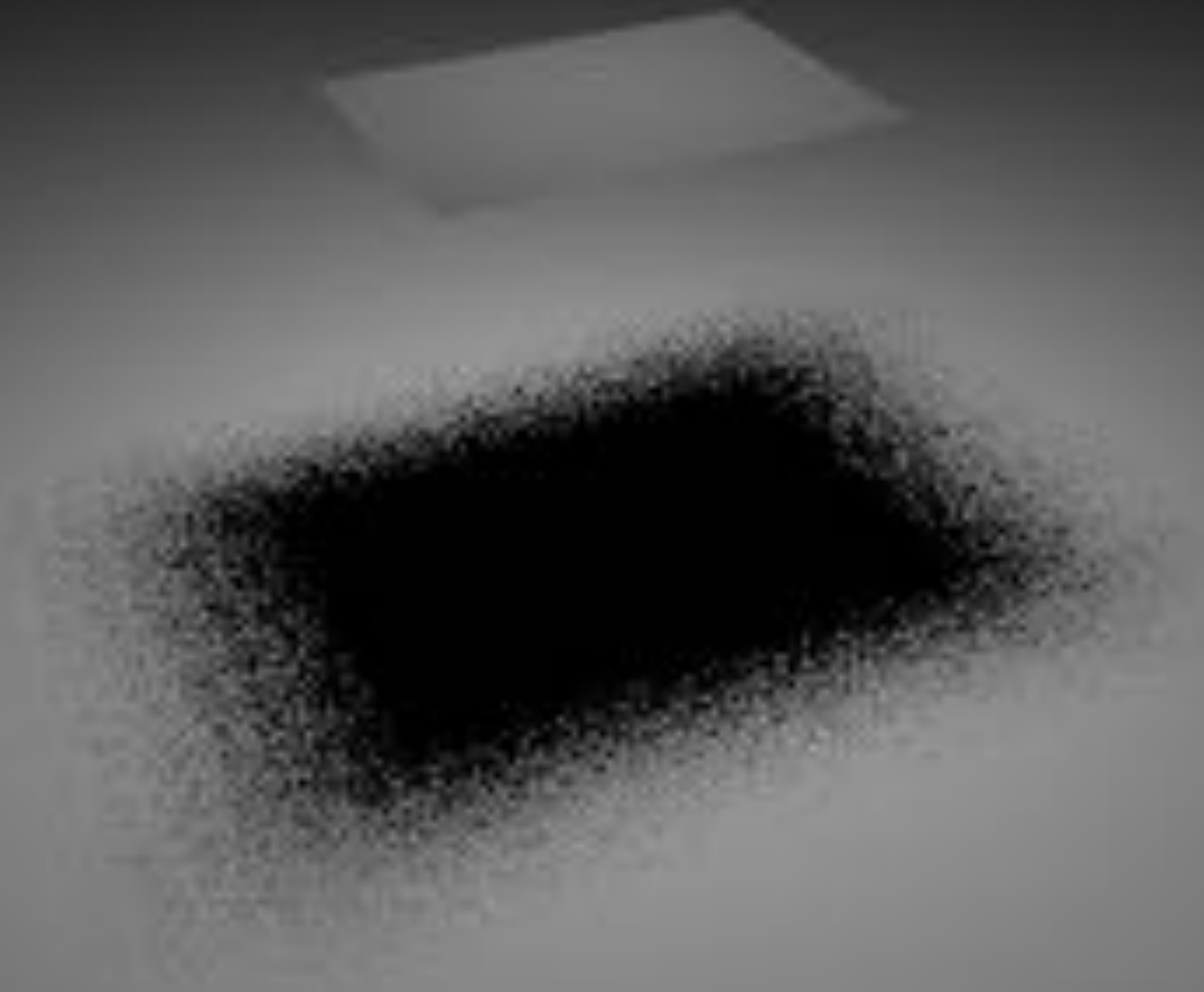
$$F_N = \frac{A'}{N} \sum_{i=1}^N Y_i$$

**Light source area  
sampling, 100 sample rays**

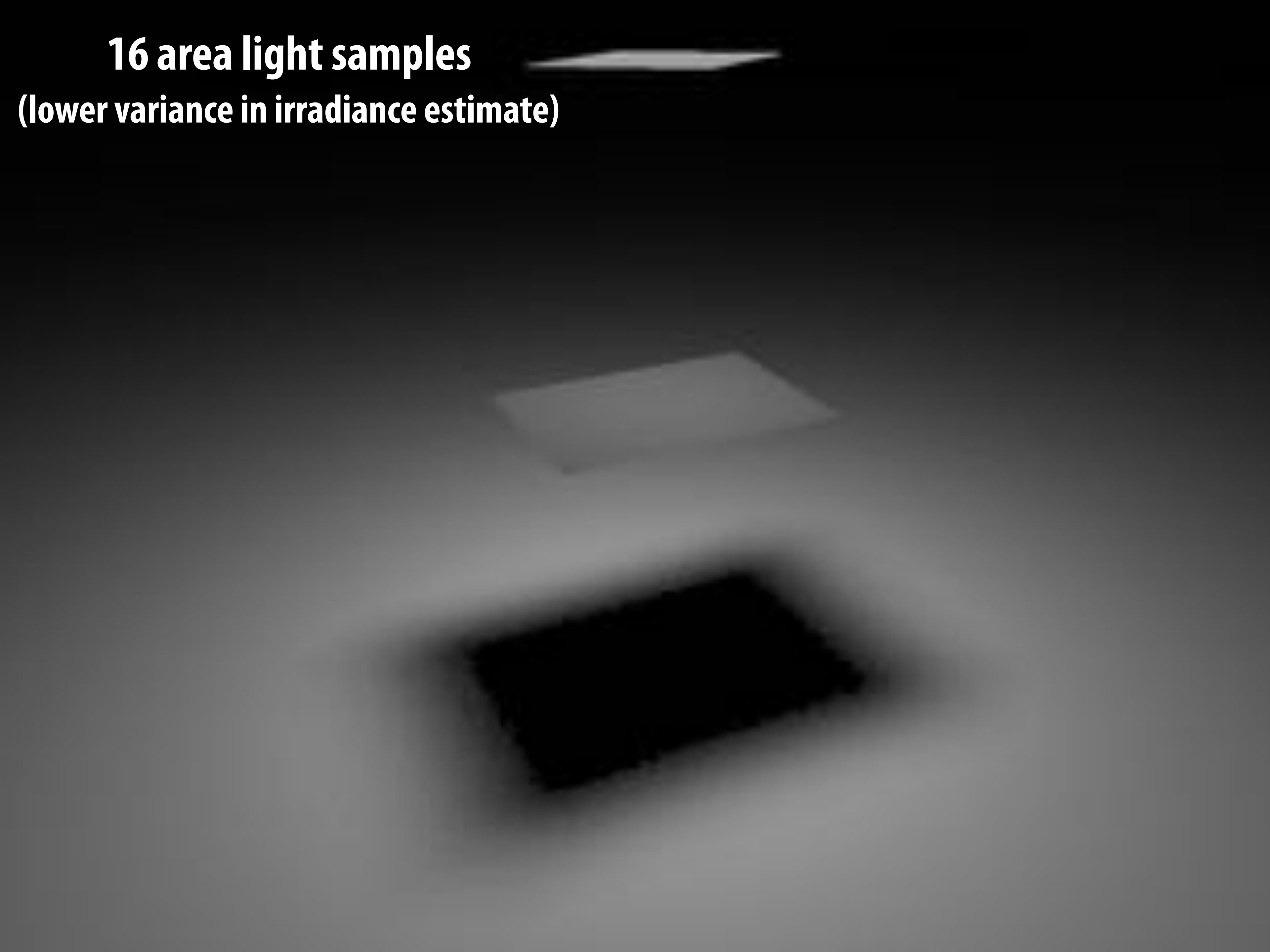


**If no occlusion is present, all directions chosen in computing estimate “hit” the light source.  
(Choice of direction only matters if portion of light is occluded from surface point p.)**

**1 area light sample**  
**(high variance in irradiance estimate)**



**16 area light samples**  
**(lower variance in irradiance estimate)**





# Comparing different techniques

- **Variance in an estimator manifests as noise in rendered images**
- **Estimator efficiency measure:**

$$\text{Efficiency} \propto \frac{1}{\text{Variance} \times \text{Cost}}$$

- **If one integration technique has twice the variance of another, then it takes twice as many samples to achieve the same variance**
- **If one technique has twice the cost of another technique with the same variance, then it takes twice as much time to achieve the same variance**

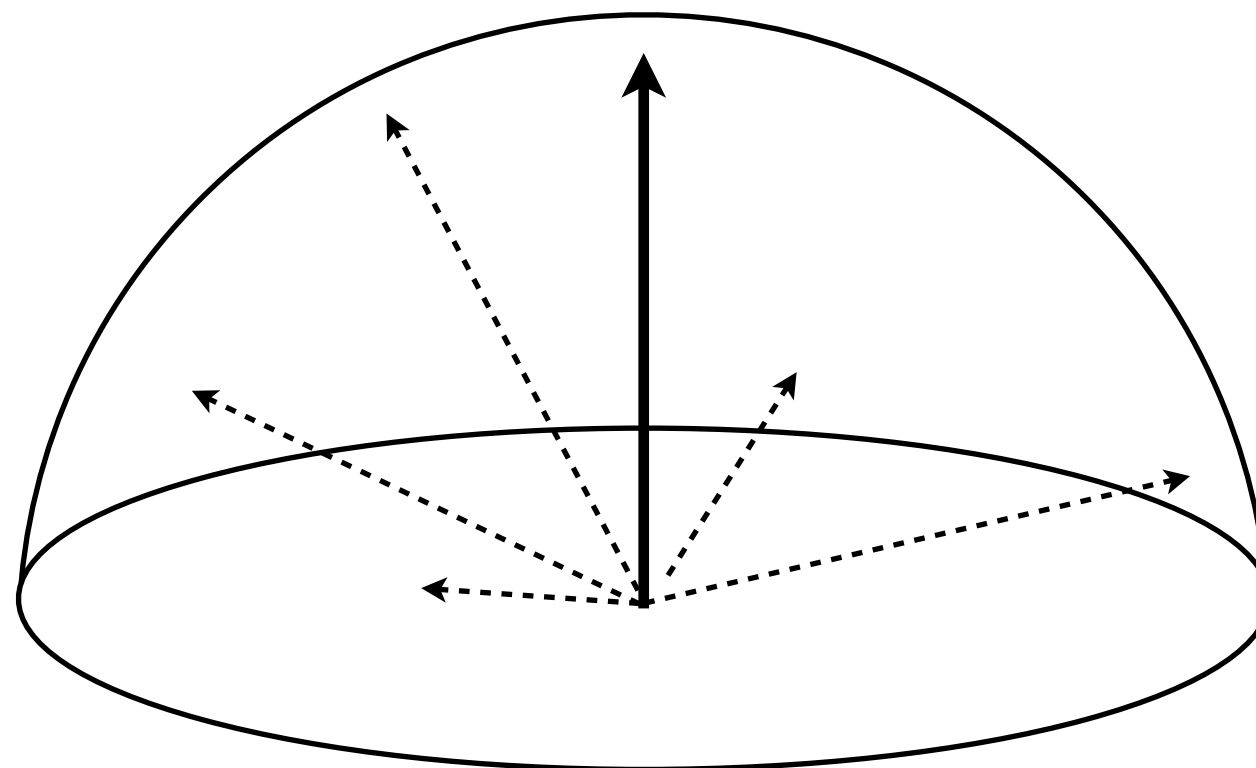
# Example—Cosine-Weighted Sampling

Consider uniform hemisphere sampling in irradiance estimate:

$$f(\omega) = L_i(\omega) \cos \theta \qquad p(\omega) = \frac{1}{2\pi}$$

$$(\xi_1, \xi_2) = (\sqrt{1 - \xi_1^2} \cos(2\pi\xi_2), \sqrt{1 - \xi_1^2} \sin(2\pi\xi_2), \xi_1)$$

$$\int_{\Omega} f(\omega) d\omega \approx \frac{1}{N} \sum_i^N \frac{f(\omega)}{p(\omega)} = \frac{1}{N} \sum_i^N \frac{L_i(\omega) \cos \theta}{1/2\pi} = \frac{2\pi}{N} \sum_i^N L_i(\omega) \cos \theta$$



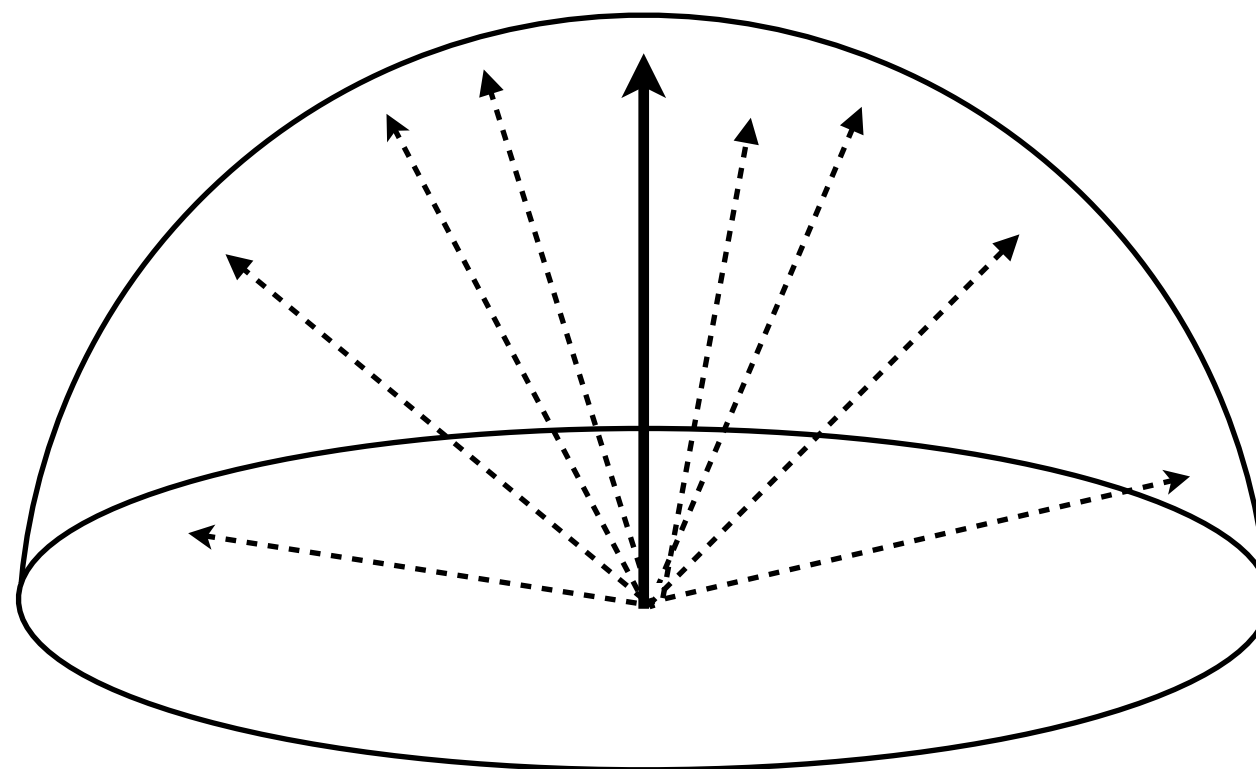
# Example—Cosine-Weighted Sampling

Cosine-weighted hemisphere sampling in irradiance estimate:

$$f(\omega) = L_i(\omega) \cos \theta \qquad p(\omega) = \frac{\cos \theta}{\pi}$$

$$\int_{\Omega} f(\omega) \, d\omega \approx \frac{1}{N} \sum_i^N \frac{f(\omega)}{p(\omega)} = \frac{1}{N} \sum_i^N \frac{L_i(\omega) \cos \theta}{\cos \theta / \pi} = \frac{\pi}{N} \sum_i^N L_i(\omega)$$

**Idea: bias samples toward directions where  $\cos \theta$  is large**  
(if  $L$  is constant, then these are the directions that contribute most)

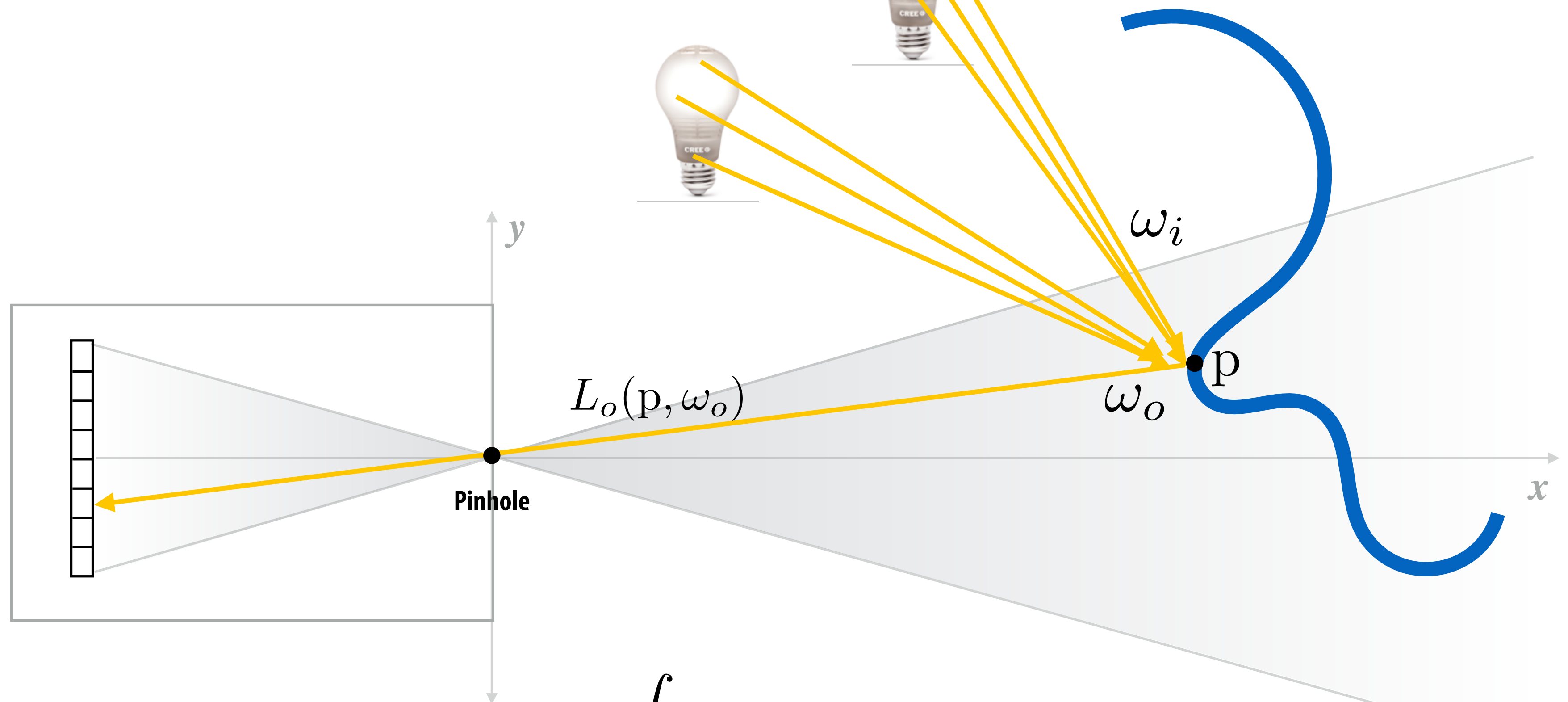


**So far we've considered light coming directly from light sources, scattered once.**

**How do we use Monte Carlo integration to get the final color values for each pixel?**



# Monte Carlo + Rendering Equation

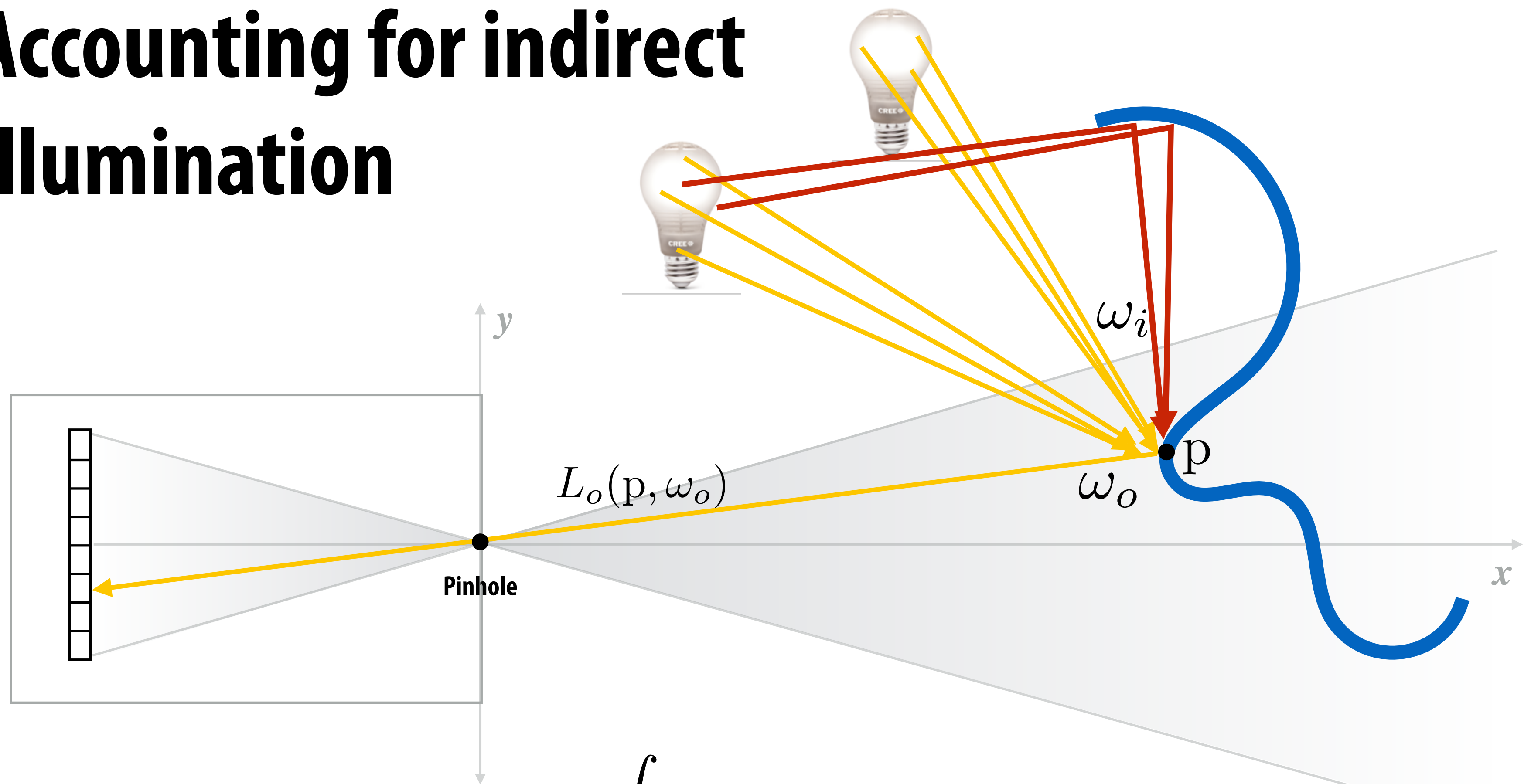


$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{H^2} f_r(p, \omega_i \rightarrow \omega_o) L_i(p, \omega_i) \cos \theta_i d\omega_i$$

**Need to know incident radiance.**

**So far, have only computed incoming radiance from scene light sources.**

# Accounting for indirect illumination



$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{H^2} f_r(p, \omega_i \rightarrow \omega_o) L_i(p, \omega_i) \cos \theta_i d\omega_i$$

**Incoming light energy from direction  $\omega_i$  may be due to light reflected off another surface in the scene (not an emitter)**

# Path tracing: indirect illumination

$$\int_{H^2} f_r(\omega_i \rightarrow \omega_o) L_{o,i}(tr(p, \omega_i), -\omega_i) \cos \theta_i d\omega_i$$

- **Sample incoming direction from some distribution (e.g. proportional to BRDF):**

$$\omega_i \sim p(\omega)$$

- **Recursively call path tracing function to compute incident indirect radiance**





• p

**Direct illumination**





• p

**One-bounce global illumination**





• p

**Two-bounce global illumination**





• p

**Four-bounce global illumination**





• p

**Eight-bounce global illumination**





• p

**Sixteen-bounce global illumination**

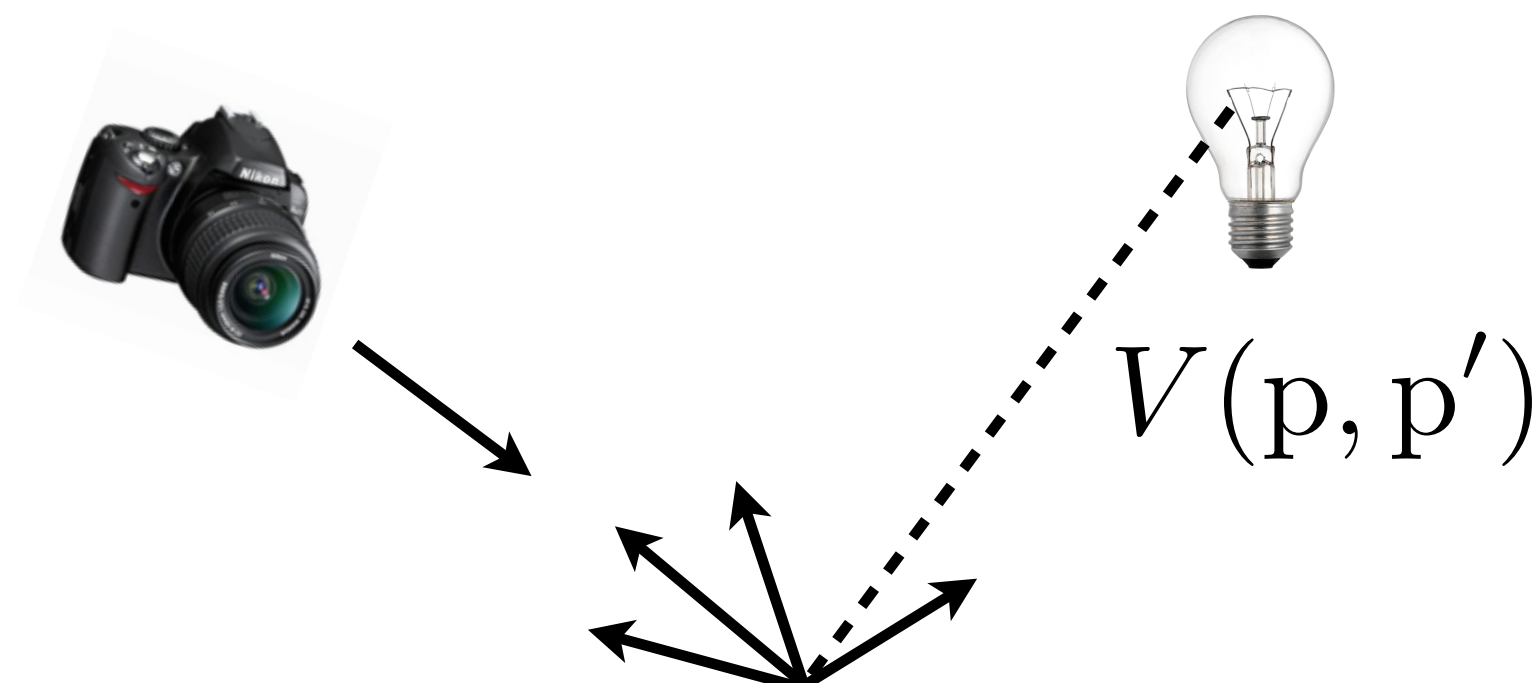


**Wait a minute...**  
**When do we stop?!**

# Russian roulette

- Idea: want to avoid spending time evaluating function for samples that make a **small contribution** to the final result
- Consider a low-contribution sample of the form:

$$L = \frac{f_r(\omega_i \rightarrow \omega_o) L_i(\omega_i) V(p, p') \cos \theta_i}{p(\omega_i)}$$



# Russian roulette

$$L = \frac{f_r(\omega_i \rightarrow \omega_o) L_i(\omega_i) V(p, p') \cos \theta_i}{p(\omega_i)}$$



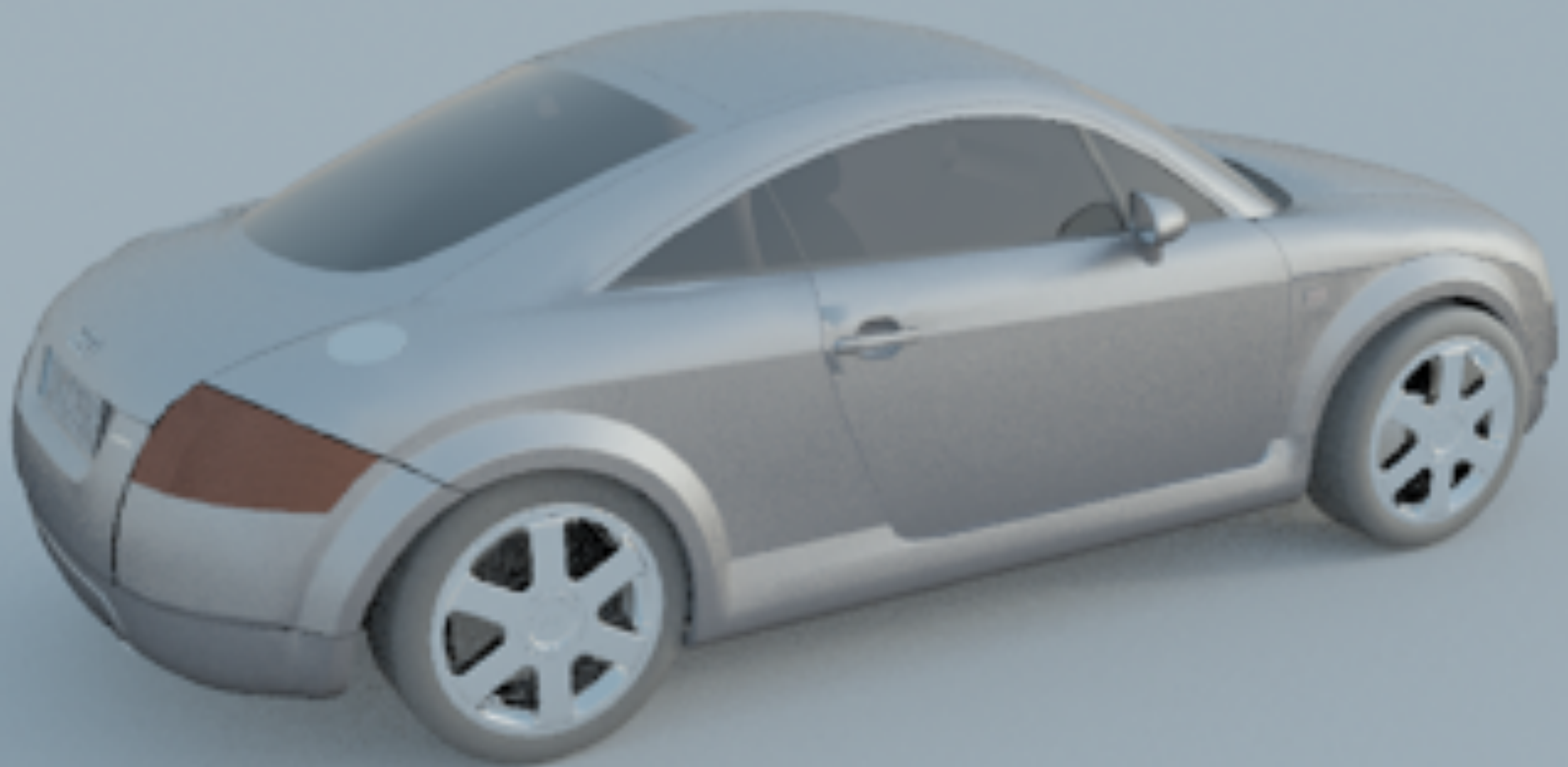
$$L = \left[ \frac{f_r(\omega_i \rightarrow \omega_o) L_i(\omega_i) \cos \theta_i}{p(\omega_i)} \right] V(p, p')$$

- If tentative contribution (in brackets) is small, total contribution to the image will be small regardless of  $V(p, p')$
- Ignoring low-contribution samples introduces systematic error
  - No longer converges to correct value!
- Instead, randomly discard low-contribution samples in a way that leaves estimator unbiased

# Russian roulette

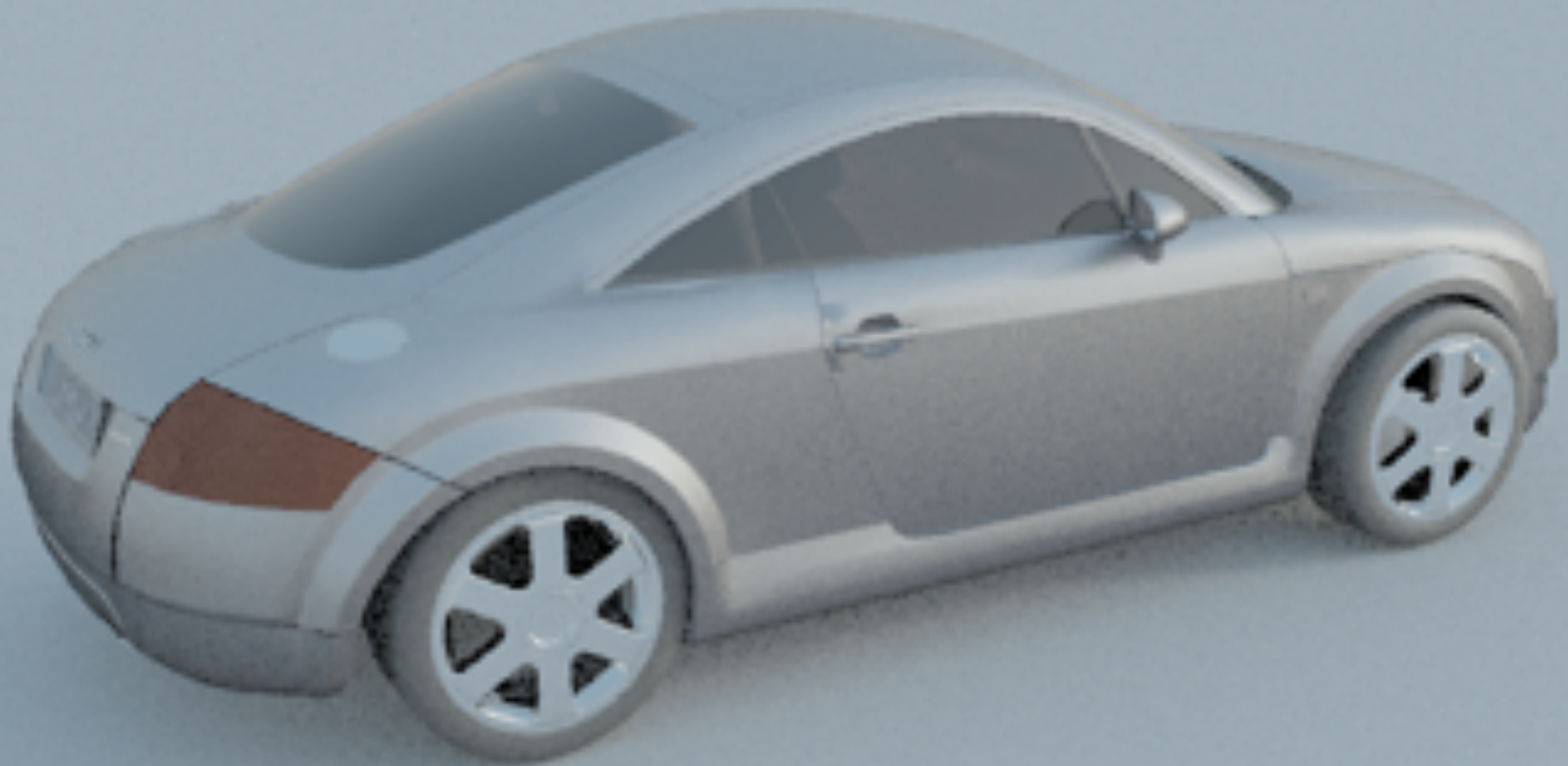
- **New estimator: evaluate original estimator with probability  $p_{rr}$ , reweight. Otherwise ignore.**
- **Same expected value as original estimator:**

$$p_{rr} E \left[ \frac{X}{p_{rr}} \right] + E[(1 - p_{rr})0] = E[X]$$

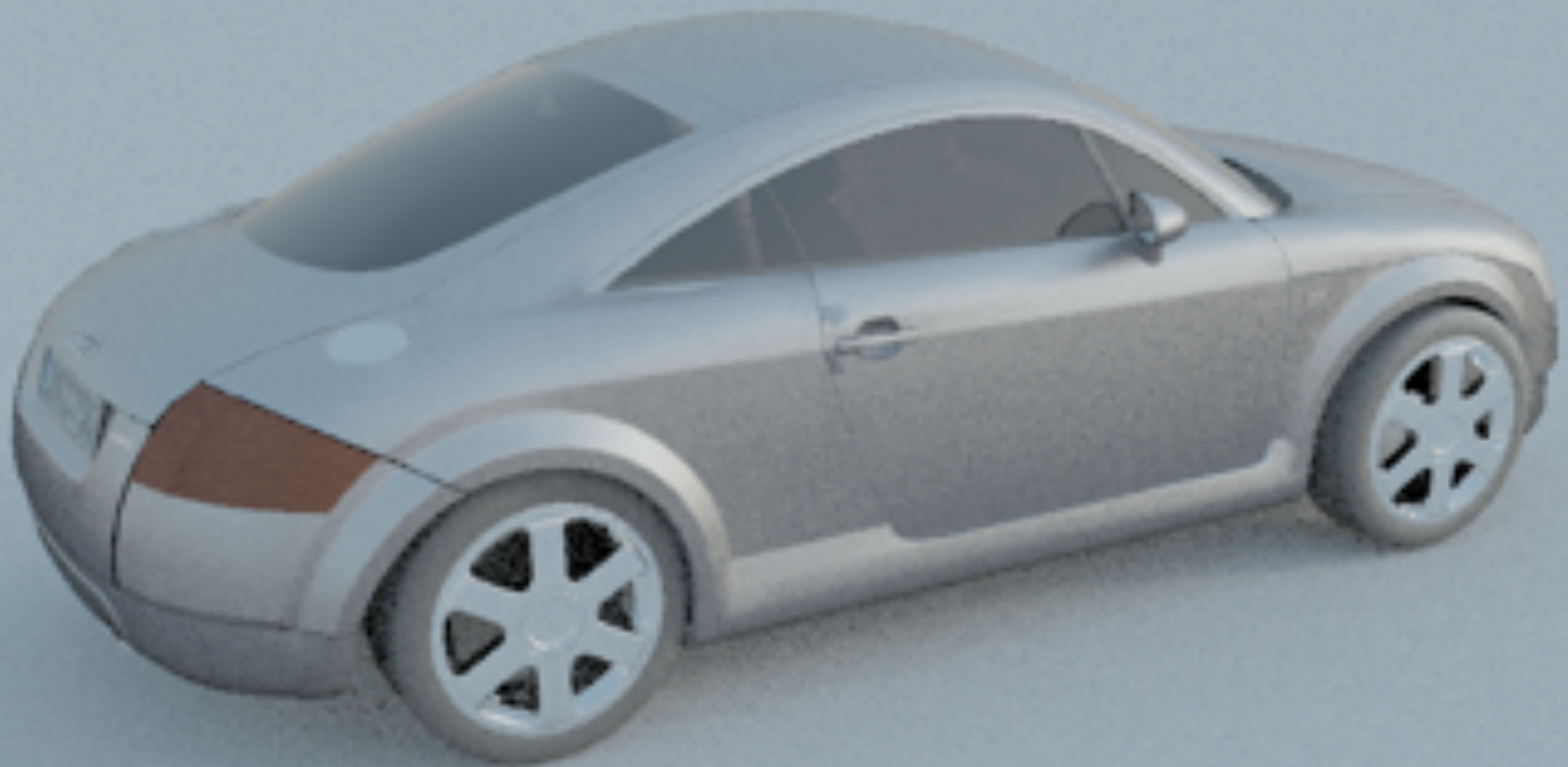


**No Russian roulette: 6.4 seconds**



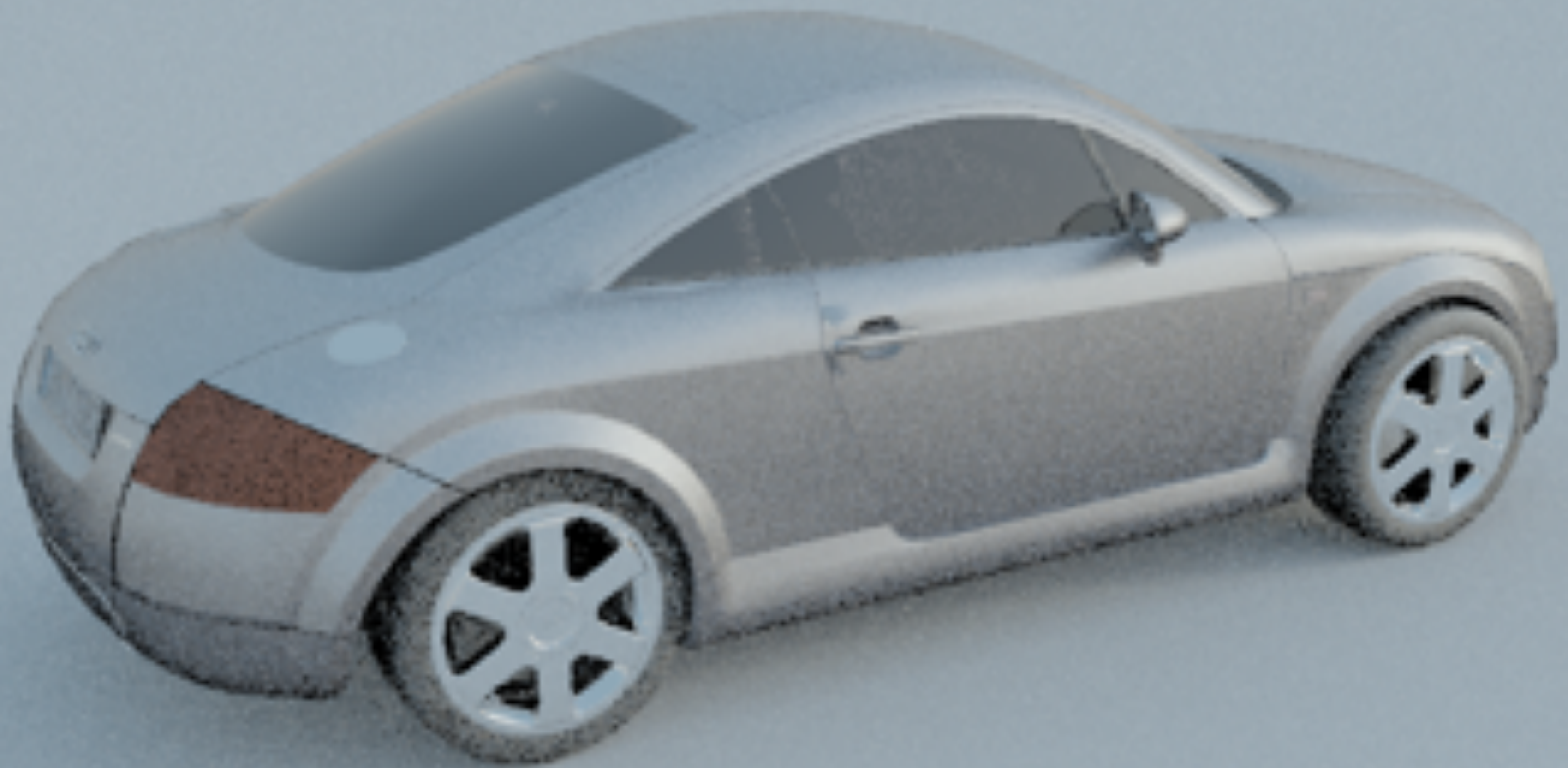


**Russian roulette: terminate 50% of all contributions with  
luminance less than 0.25: 5.1 seconds**



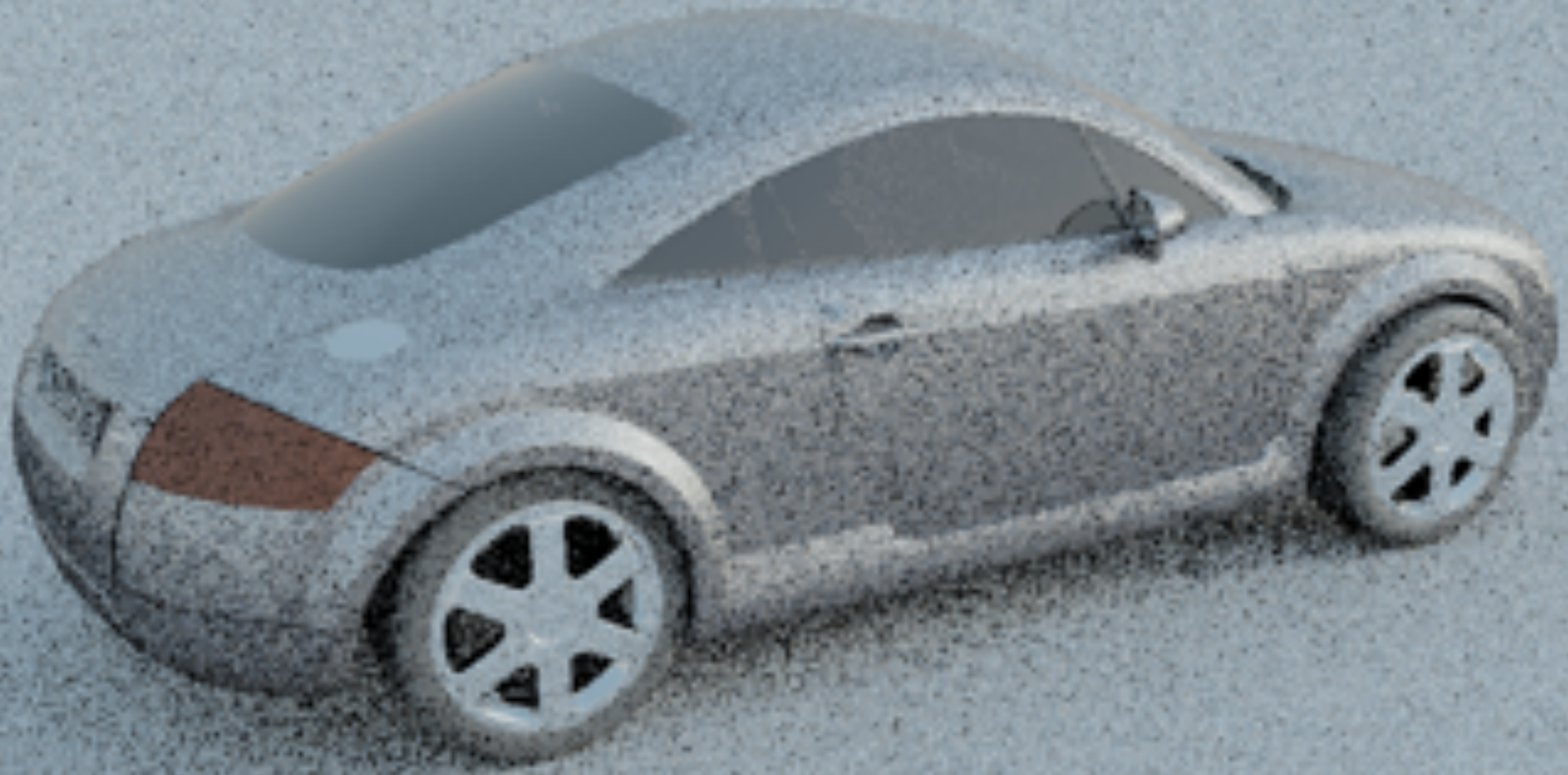
**Russian roulette: terminate 50% of all contributions with  
luminance less than 0.5: 4.9 seconds**





**Russian roulette: terminate 90% of all contributions with  
luminance less than 0.125: 4.8 seconds**



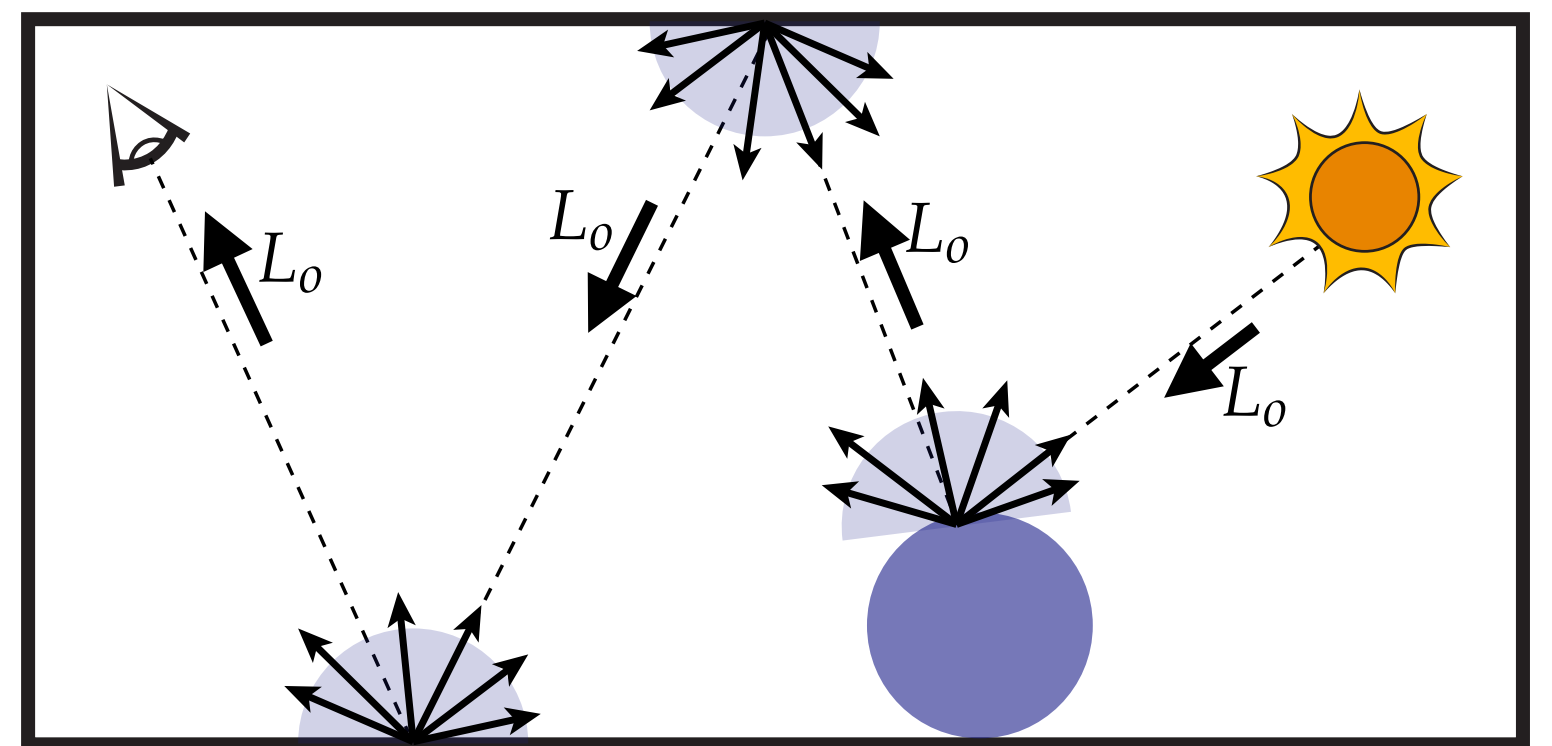


**Russian roulette: terminate 90% of all contributions with  
luminance less than 1: 3.6 seconds**



# Monte Carlo Ray Tracing—Summary

- Light hitting a point (e.g., pixel) described by rendering equation
  - Expressed as recursive integral
  - Can use Monte Carlo to estimate this integral
  - Need to be intelligent about how to sample!



$$L_o(\mathbf{p}, \omega_o) = L_e(\mathbf{p}, \omega_o) + \int_{\mathcal{H}^2} f_r(\mathbf{p}, \omega_i \rightarrow \omega_o) L_i(\mathbf{p}, \omega_i) \cos \theta \, d\omega_i$$



# Next time:

- **Variance reduction—how do we get the most out of our samples?**

