Lecture 15:

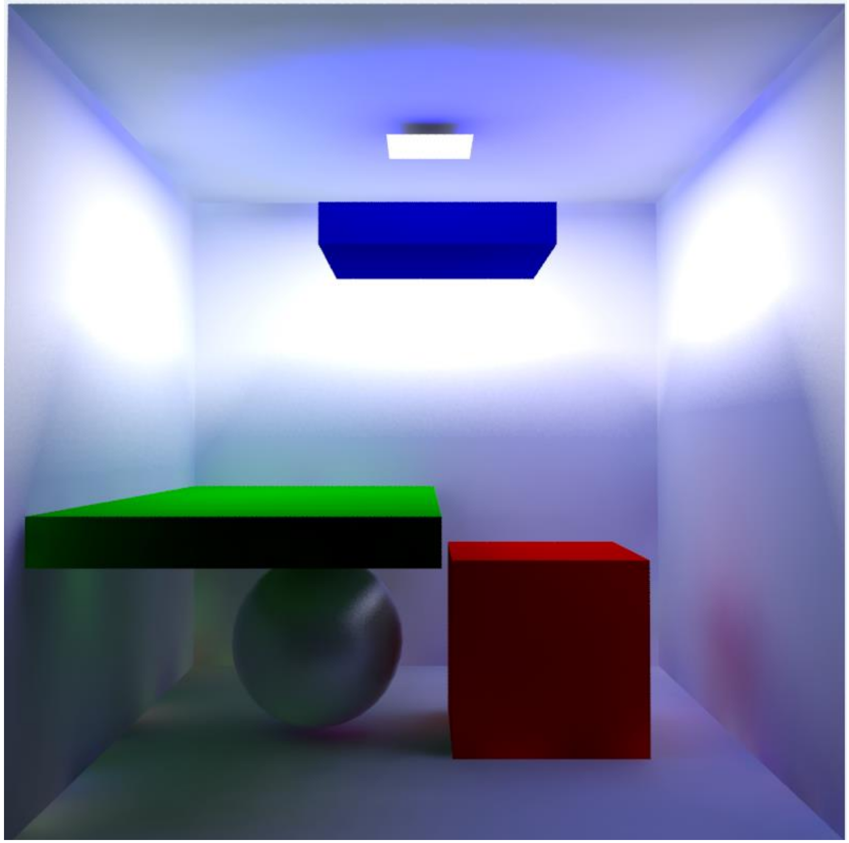
Global Illumination

Contents

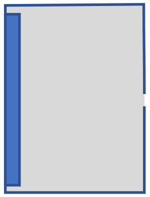
1. Introduction
2. The Rendering Equation
3. Monte Carlo Integration
4. Path Tracing
5. Ambient Occlusion
6. Radiosity Equation

Why bother with physically correct rendering?

- As opposed to making up shaders that look good
  - When something goes wrong, you can reason about why, and how to fix it
  - It is easy to hack a material shader that looks realistic for specific lighting conditions, but extremely difficult to hack something that allways looks realistic
- Path-tracing may take longer to converge
- But we will know early if something looks wrong



Where does an image come from?



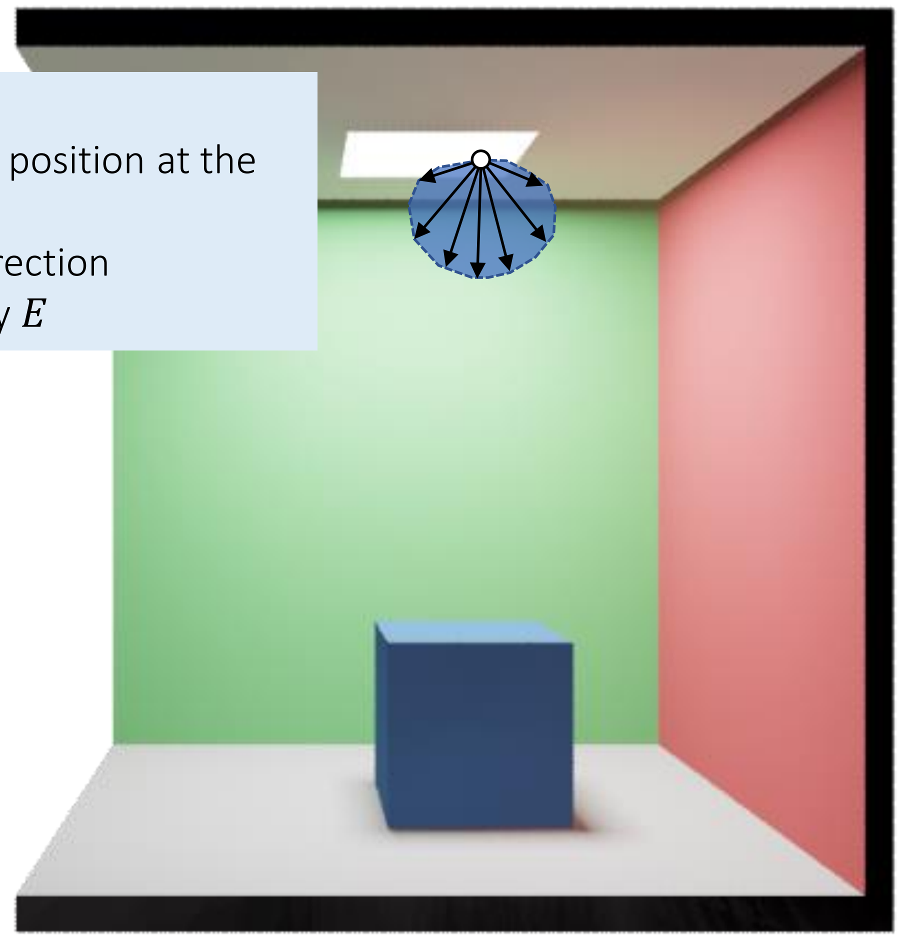
Pinhole
Camera

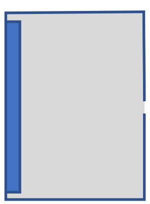
Where does an image come from?

- Position, direction and energy come from properties of the light

Photon emitted

- From some random position at the light source
- To some random direction
- Carries some energy E

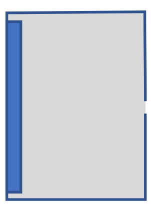




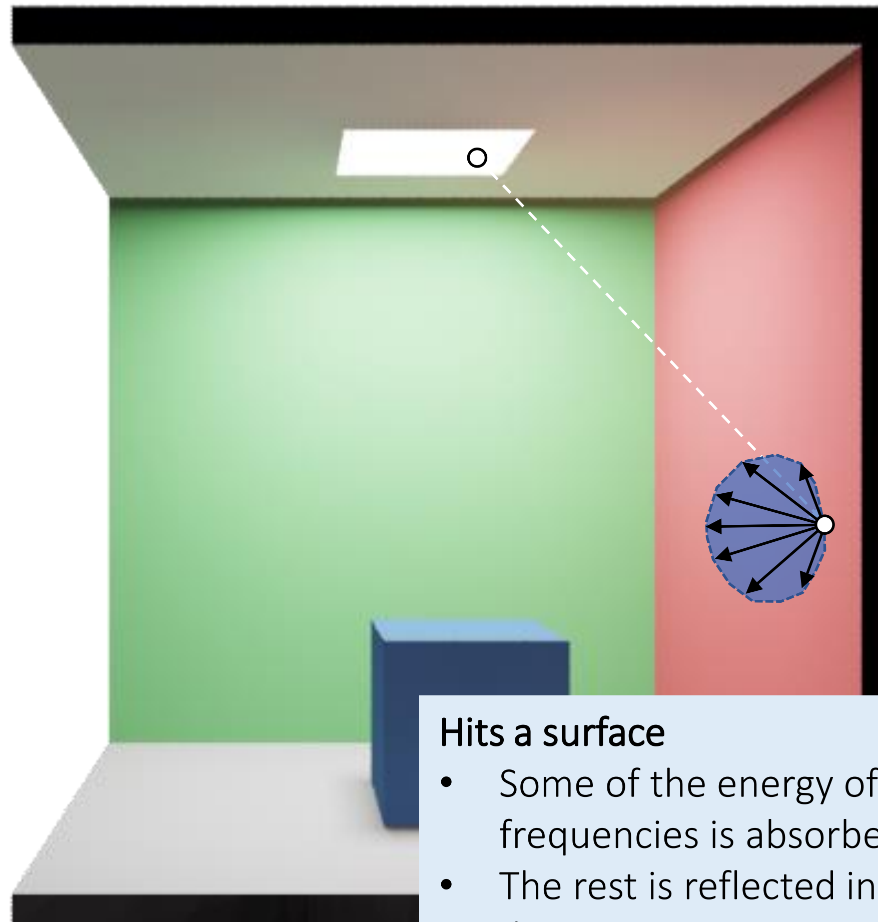
Pinhole Camera

Where does an image come from?

- The amount of absorbed energy is a material property
- The direction in which it is reflected depends on the BRDF of the material
- Here, the material is red and diffuse, so green and blue frequencies will be absorbed and the reddish photon may bounce of in most any direction



Pinhole Camera

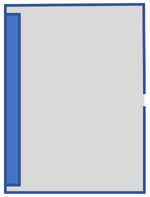


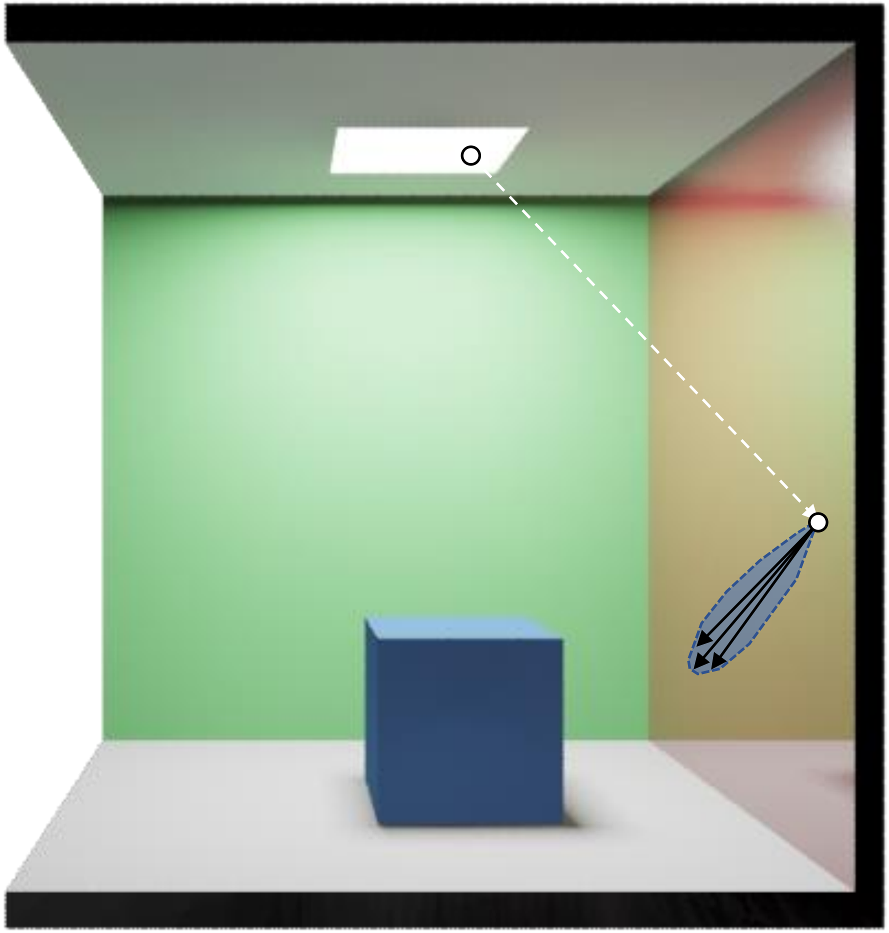
Hits a surface

- Some of the energy of some frequencies is absorbed
- The rest is reflected in some direction

Where does an image come from?

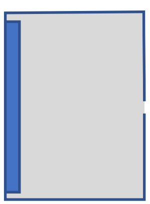
- Here, the surface is more of a mirror so little energy will be absorbed and the photon is more likely to bounce in the perfect reflection direction

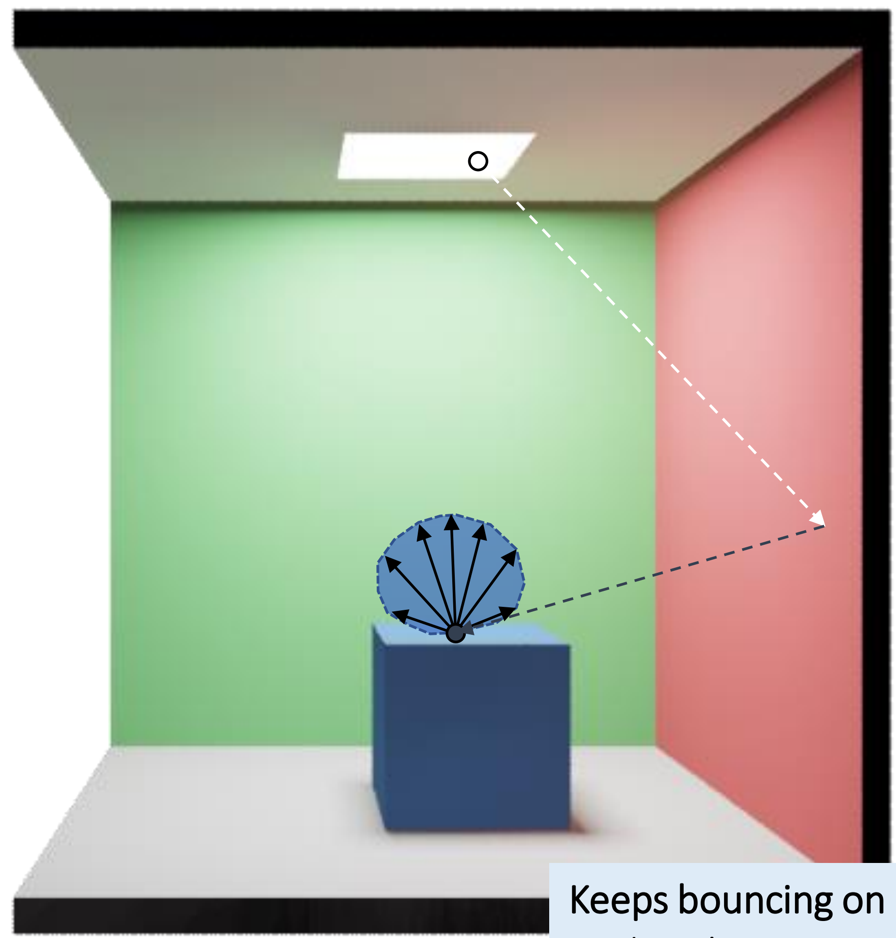




Where does an image come from?

- The photon will keep bouncing on the surfaces…

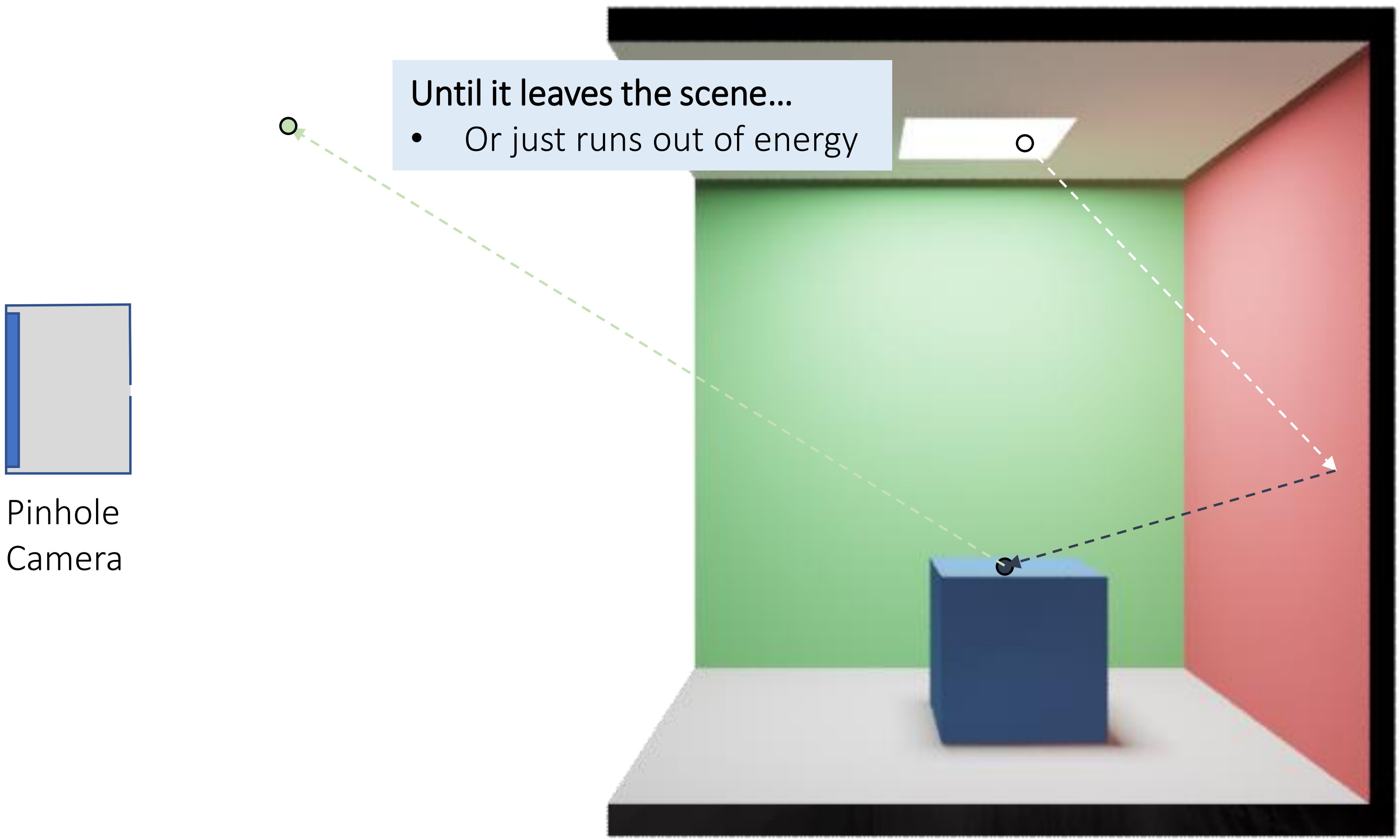




Keeps bouncing on surfaces
- Loosing energy at each hit

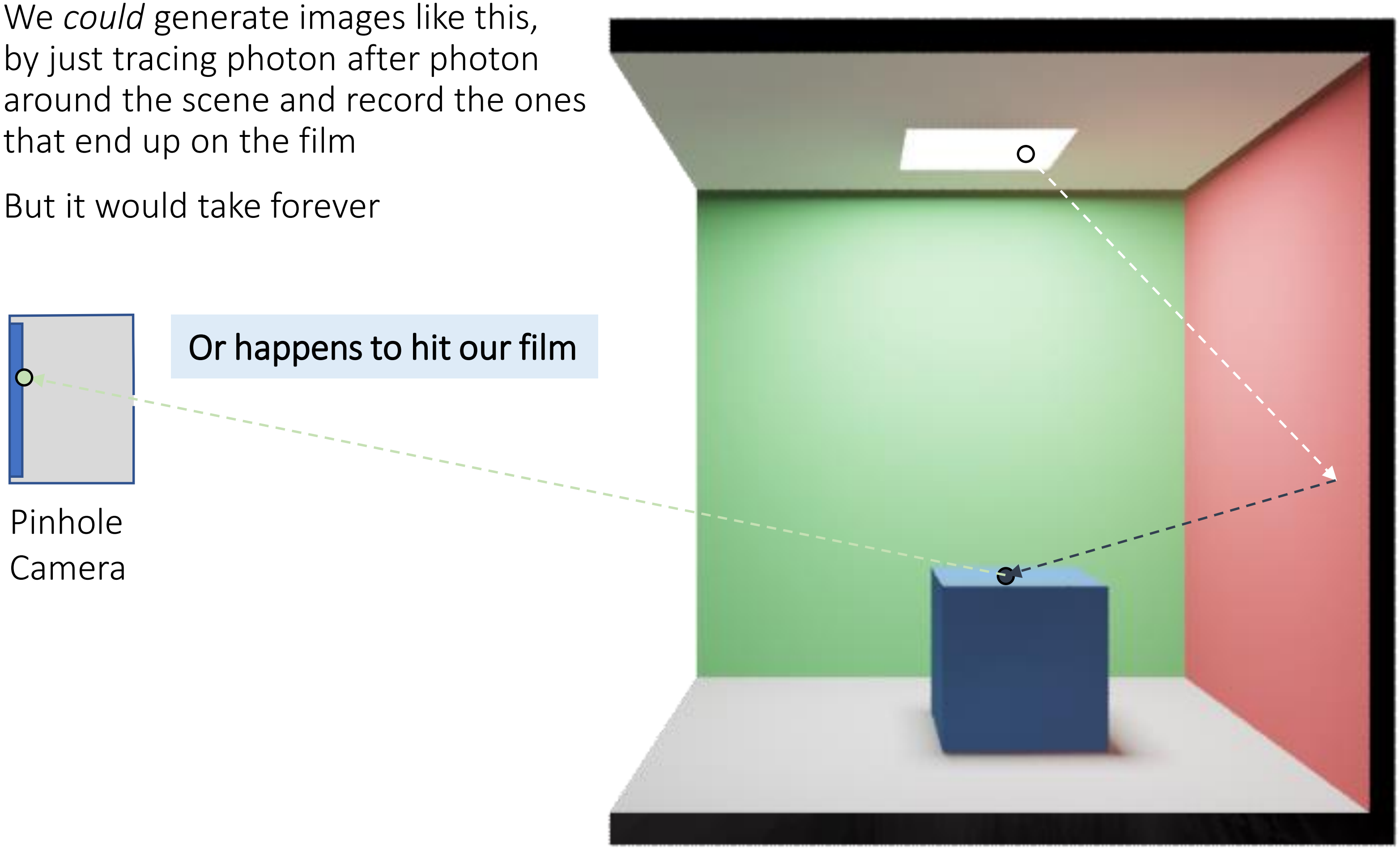
Where does an image come from?

- We could keep following this photon until it left the scene, or ran out of energy



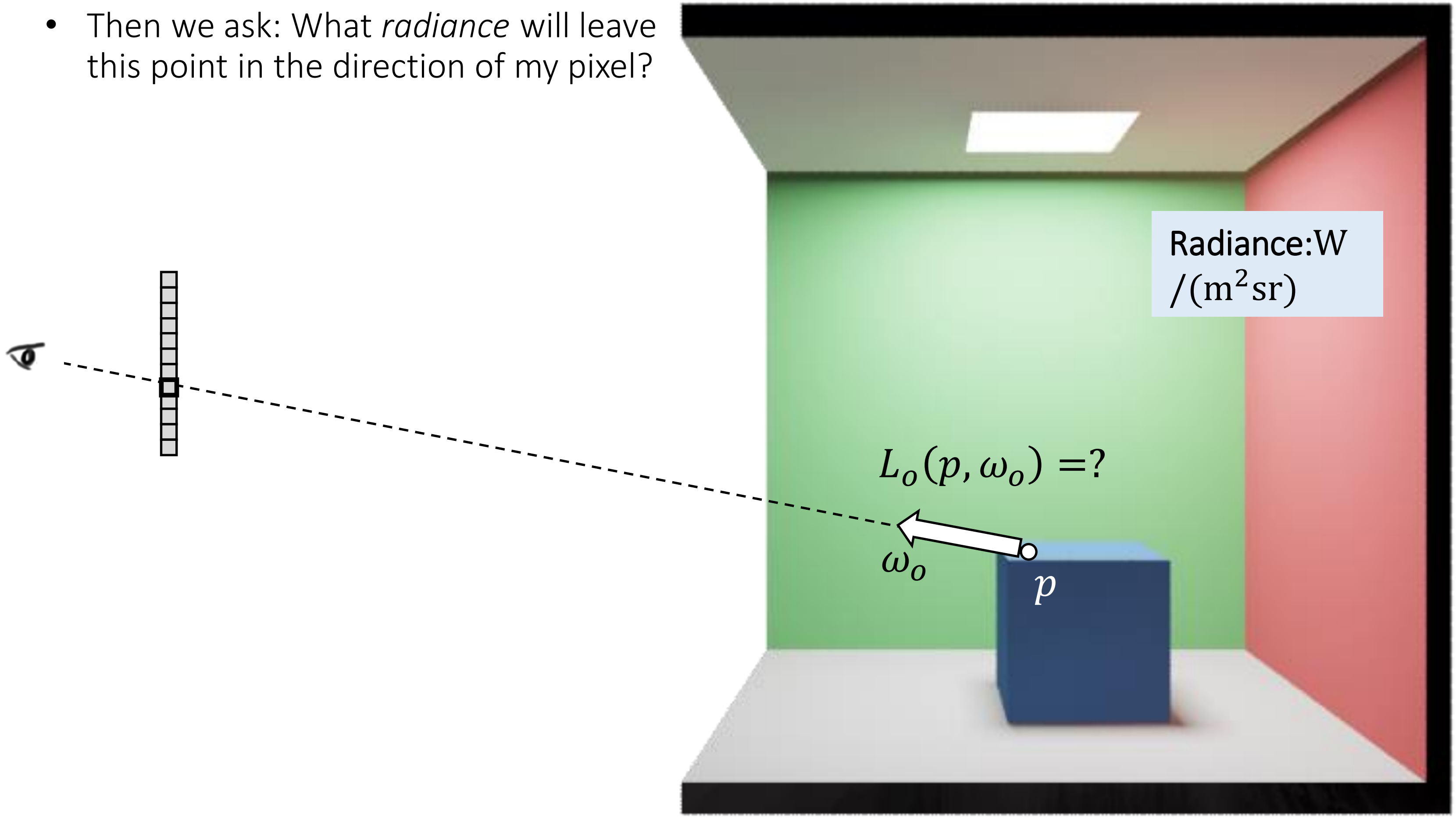
Where does an image come from?

- Or, by some chance it happens to reflect off a surface, go through the little pinhole and land on our film contributing to a pixel value
- We *could* generate images like this, by just tracing photon after photon around the scene and record the ones that end up on the film
- But it would take forever

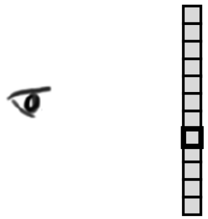


Backward Light Tracing

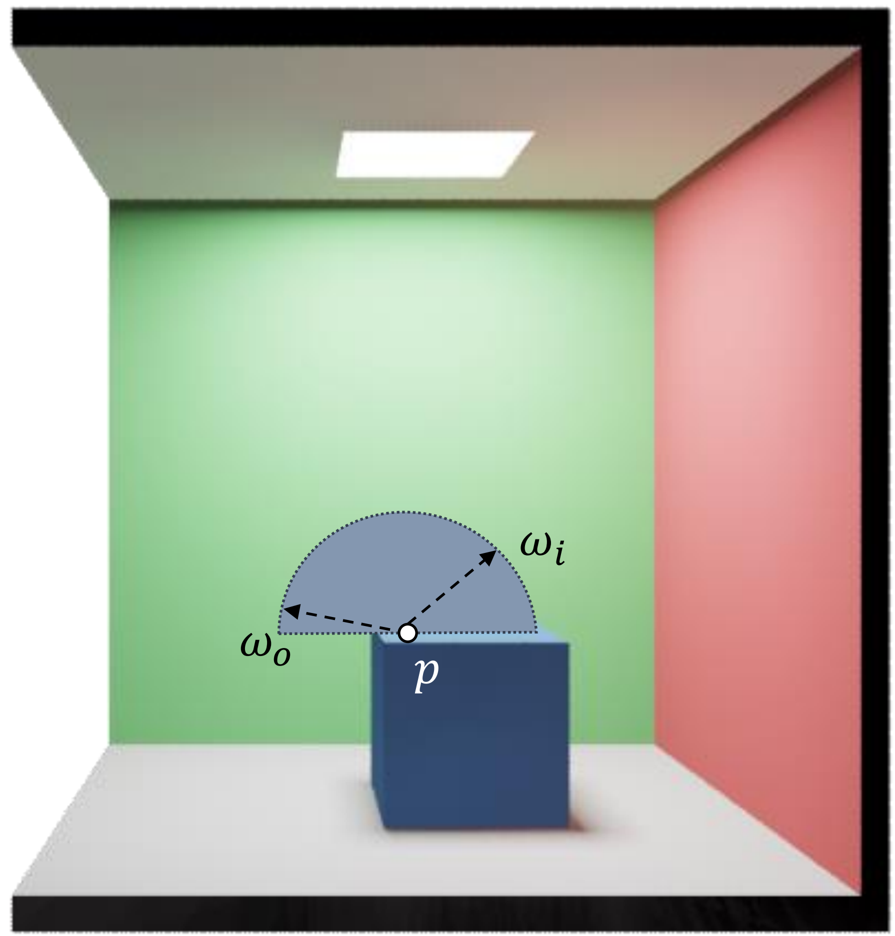
- Since most of the paths traced would have been wasted, we do this simulation backwards instead
- So, we shoot a ray *through* the pixel we are interested in, until we hit a surface
- Then we ask: What *radiance* will leave this point in the direction of my pixel?



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

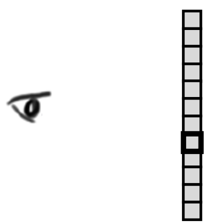


Integrate over all directions ω_i in the hemisphere around p

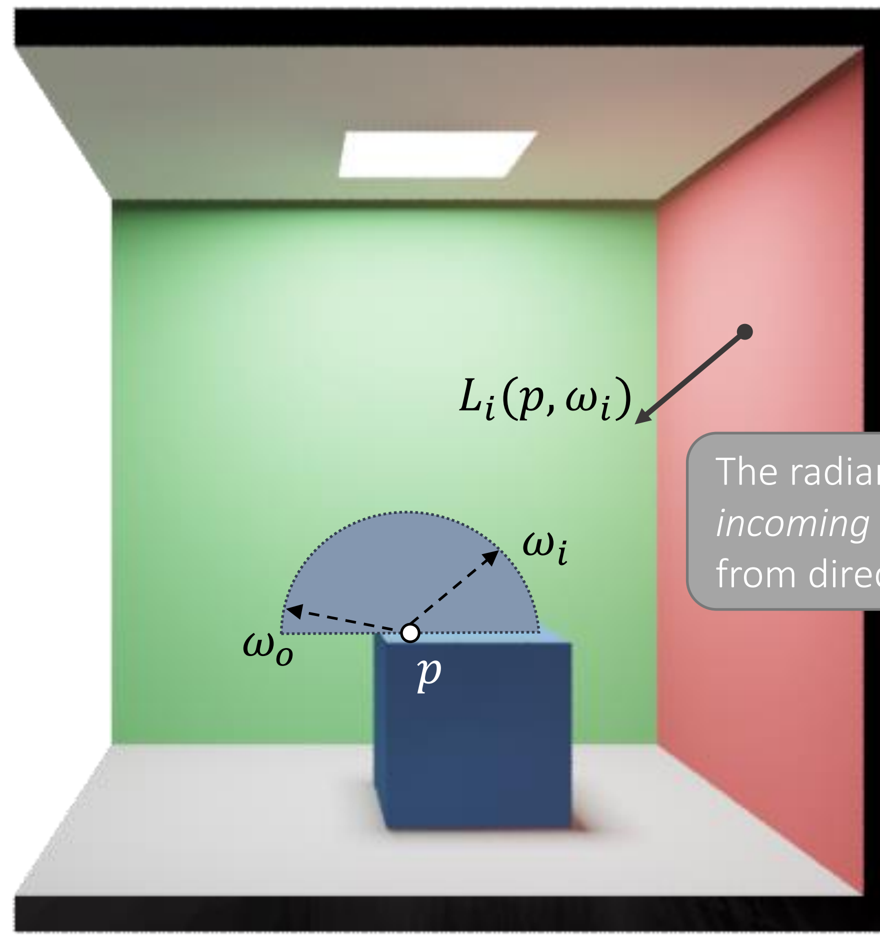


$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{H^2(\vec{n})} f_r(p,\omega_i,\omega_o) L_i(p,\omega_i) \cos\theta_i \, d\omega_i$$

The Rendering Equation



Integrate over all directions ω_i in the hemisphere around p

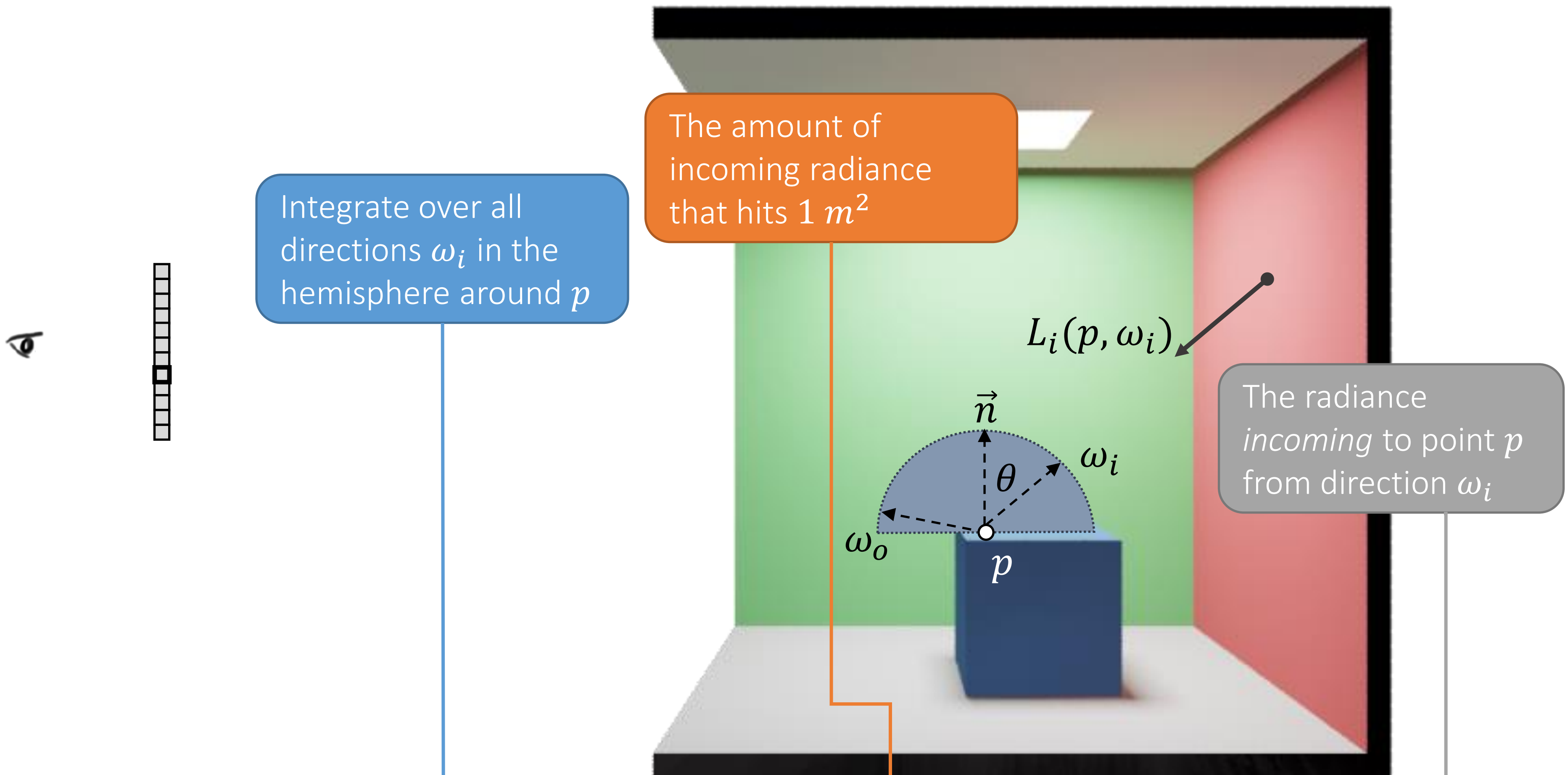


The radiance *incoming* to point p from direction ω_i

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

The cosine term

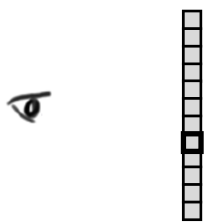
- Tells us which amount of the incoming radiance from direction ω_i will land on a unit surface area
- Independent of the material



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

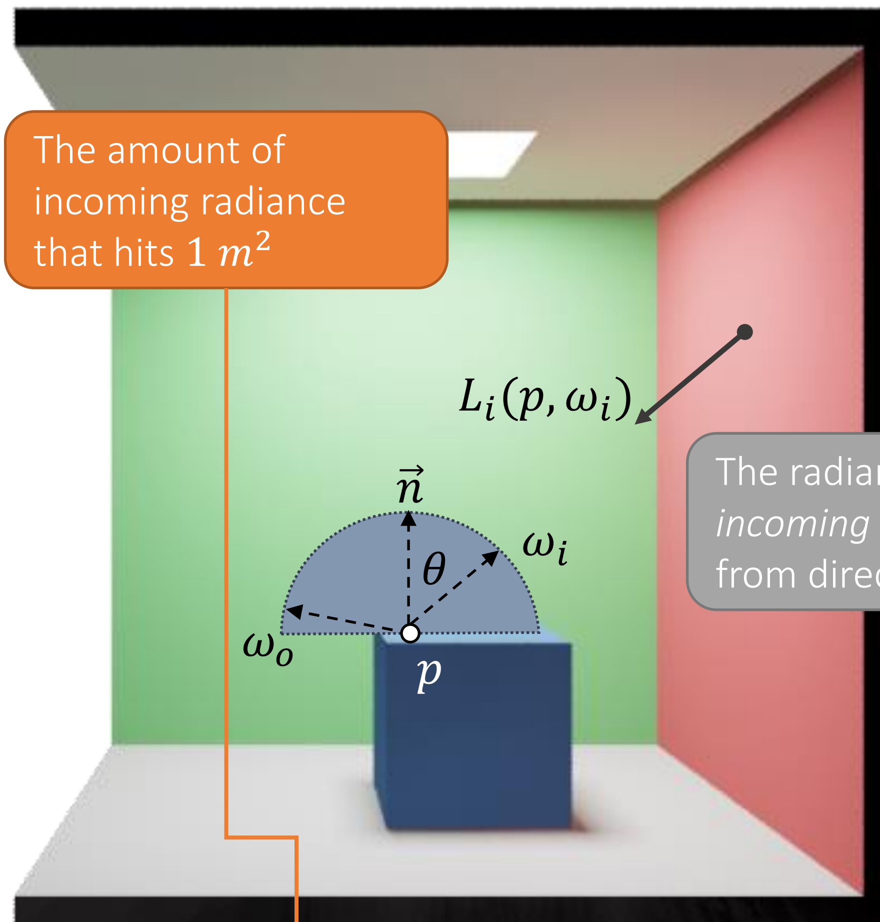
The BRDF

- A mirror surface will only have high values when $\omega_i = \omega_o$
- A diffuse surface has a constant BRDF



Integrate over all directions ω_i in the hemisphere around p

The amount of incoming radiance that hits 1 m^2



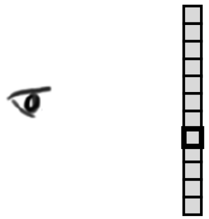
The radiance *incoming* to point p from direction ω_i

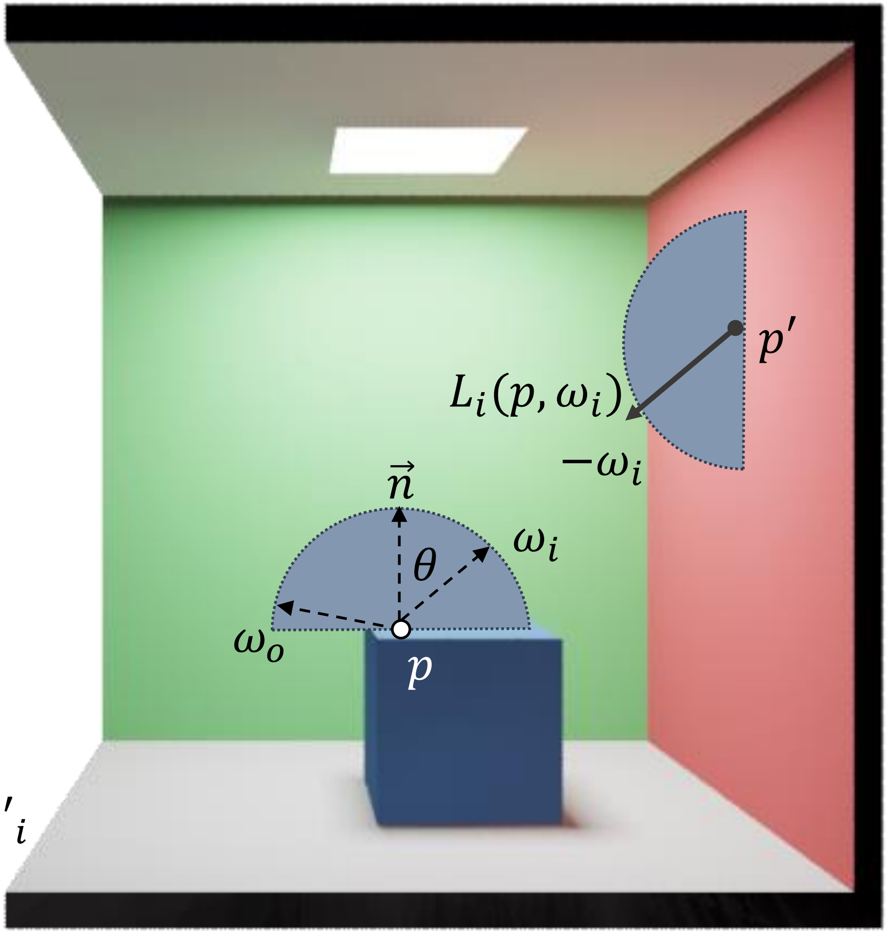
The BRDF tells us what fraction of the radiance incoming from ω_i will be reflected in direction ω_o

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

The BRDF

- We (usually) can not solve this equation analytically, since it depends on a scene which we have no nice mathematical description of





$$L_i(p, \omega_i) = L_o(p', -\omega_i)$$

$$L_o(p', -\omega_i) = L_e(p', -\omega_i) + \int_{H^2(\vec{n})} f_r(p', \omega'_i, -\omega_i) L_i(p', \omega'_i) \cos\theta'_i \, d\omega'_i$$

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

Numerical Integration

- But we *can* estimate the value of *any* integral using Monte-Carlo integration
- We then take N random samples over the domain of the integral

$$L_o(p,\omega_o) \approx L_e(p,\omega_o) + \frac{1}{N}\sum_{i=0}^{N} \frac{f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i}{p(\omega_i)}$$

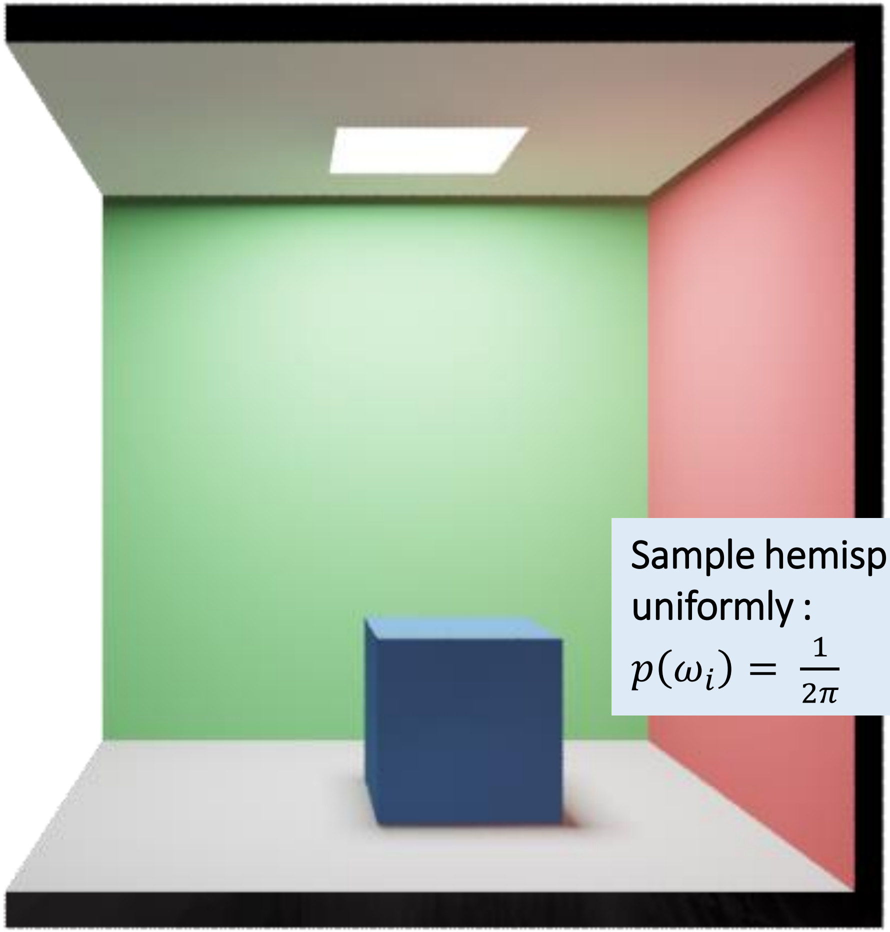
Numerical Integration

- This is an *unbiased* estimator, so the *expected value* will be exactly the radiance we are after

$$L_o(p,\omega_o) = \mathbb{E}\left[L_e(p,\omega_o) + \frac{1}{N}\sum_{i=0}^{N}\frac{f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i}{p(\omega_i)}\right]$$

Numerical Integration

- This is an *unbiased* estimator, so the *expected value* will be exactly the radiance we are after
- Even if we only take ONE sample
- And even if we sample the hemisphere perfectly uniformly (making the PDF constant)

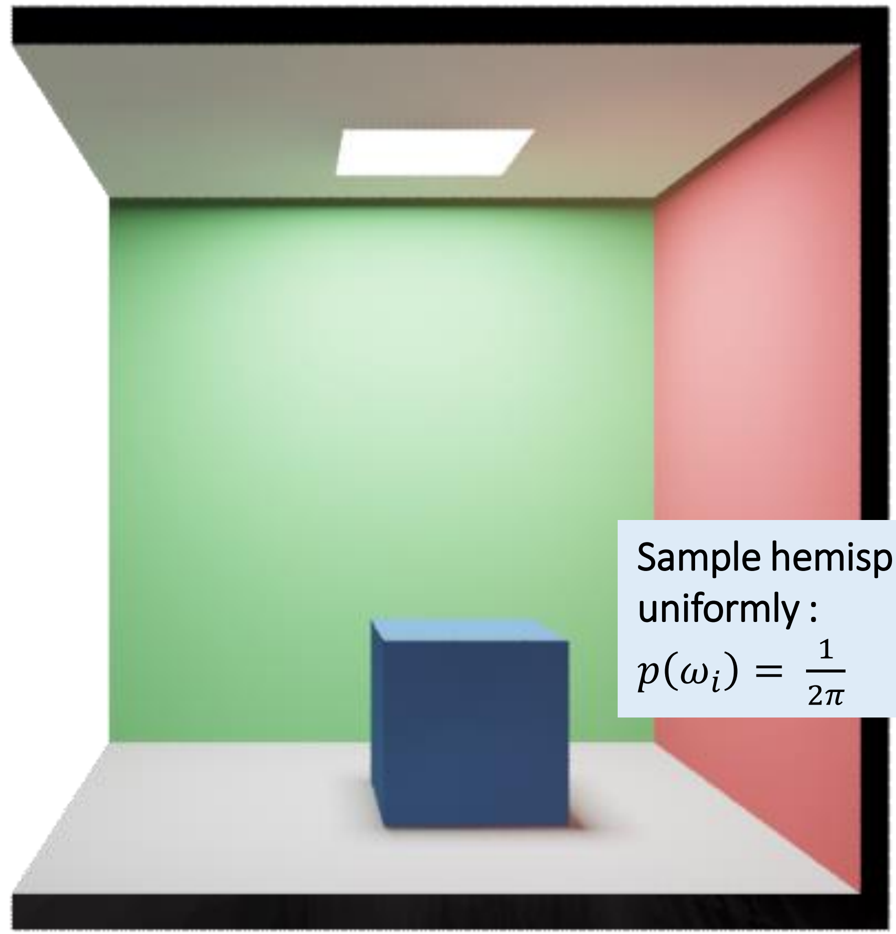


Sample hemisphere uniformly : $p(\omega_i) = \frac{1}{2\pi}$

$$L_o(p,\omega_o) = \mathbb{E}\left[L_e(p,\omega_o) + \frac{1}{1}\sum_{i=0}^{1}\frac{f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i}{p(\omega_i)}\right]$$

Numerical Integration

- Now we have a very simple estimator for the correct outgoing radiance from p
- Since it is *unbiased* we can evaluate this for every pixel, time and time again and the average will converge towards the correct value

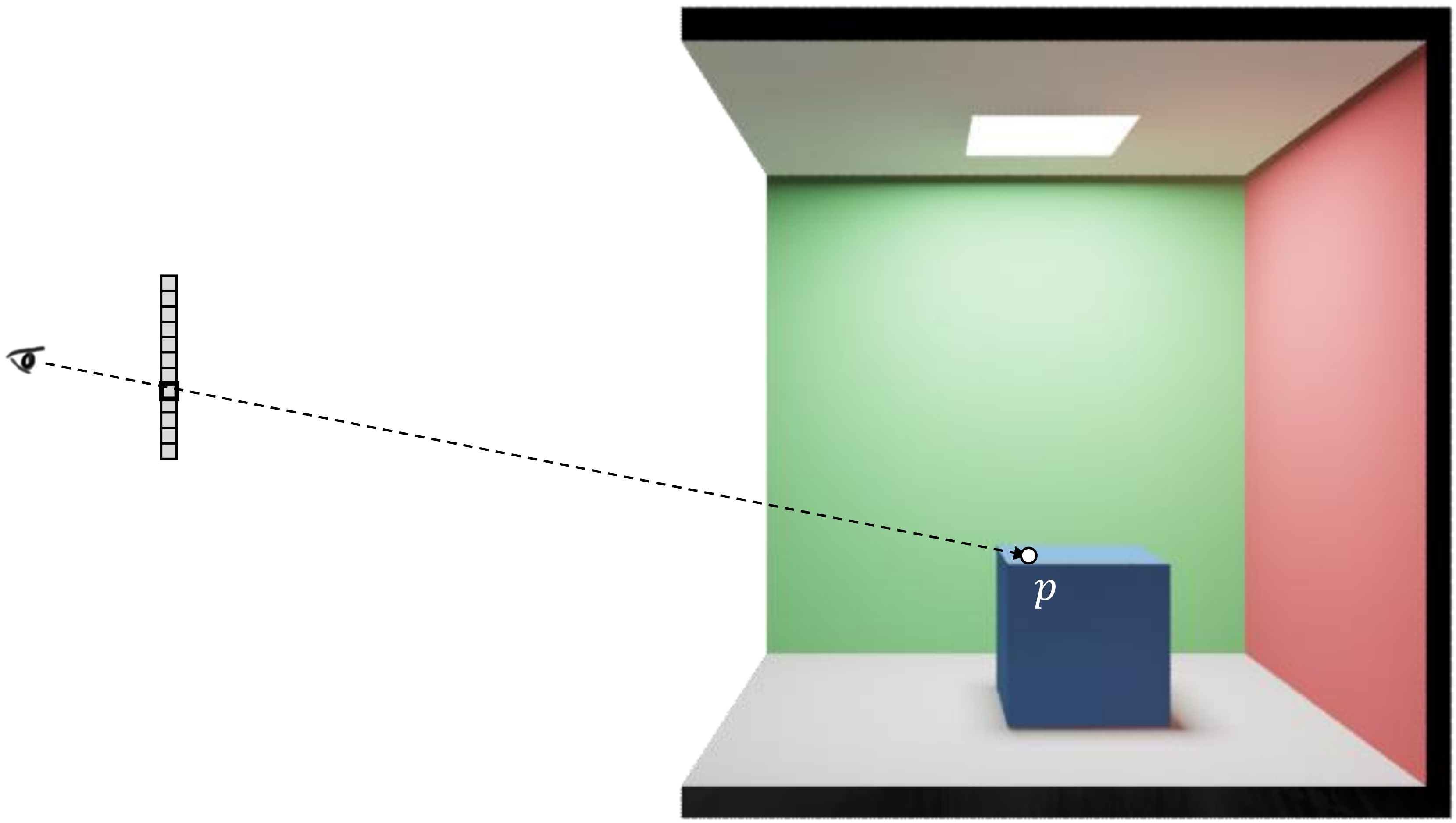


Sample hemisphere uniformly : $p(\omega_i) = \frac{1}{2\pi}$

$$L_o(p, \omega_o) = \mathbb{E}[L_e(p, \omega_o) + 2\pi f_r(p, \omega_i, \omega_o) L_i(p, \omega_i) \cos\theta_i]$$

Basic path tracing algorithm

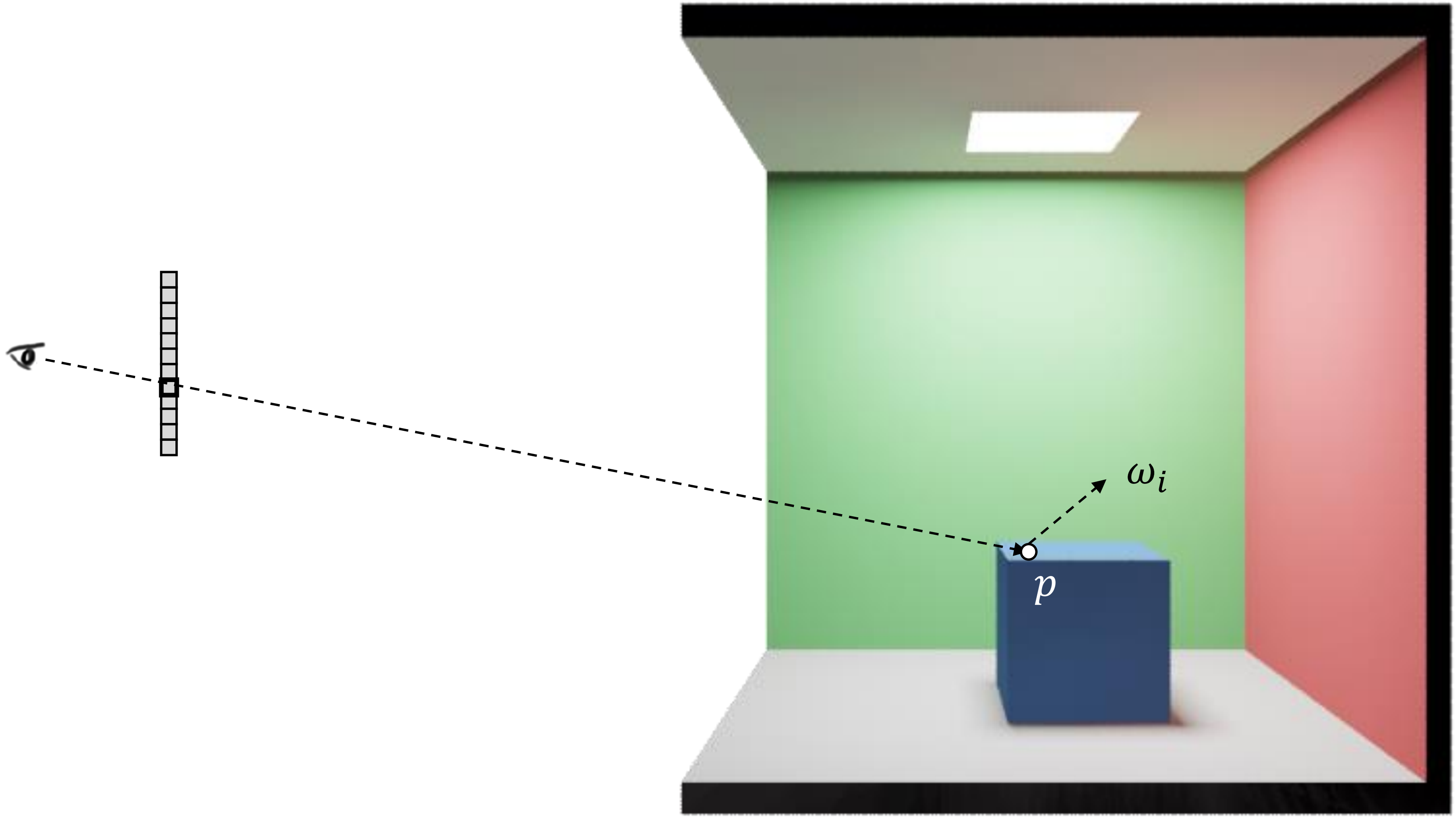
- Trace a ray through the pixel, to find the first intersection point p



$$L_o(p,\omega_o) \approx L_e(p,\omega_o) + 2\pi f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i$$

Basic path tracing algorithm

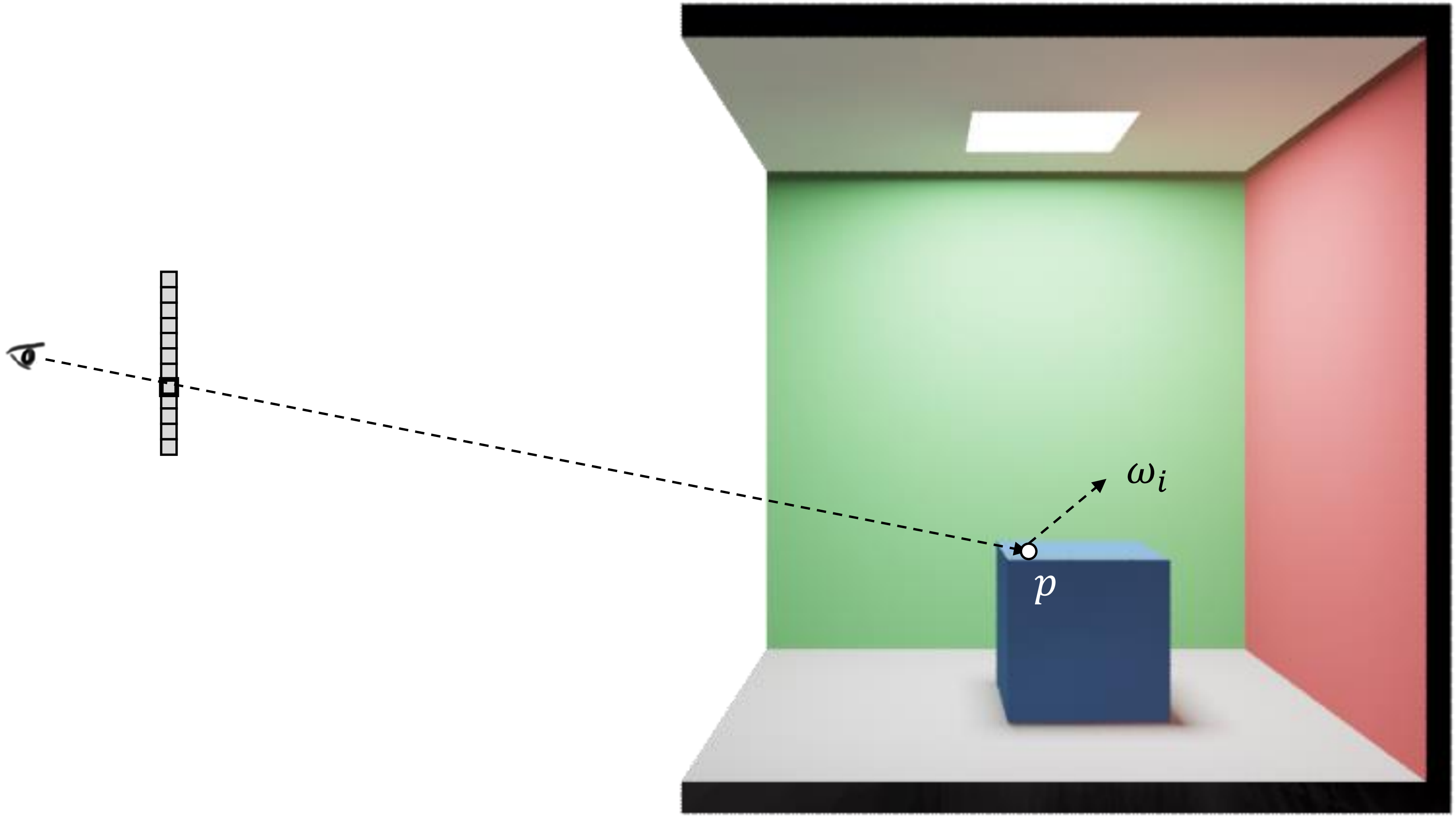
- Trace a ray through the pixel, to find the first intersection point p
- Choose a random direction ω_i on the hemisphere



$$L_o(p,\omega_o) \approx L_e(p,\omega_o) + 2\pi f_r(p,\omega_i,\omega_o) L_i(p,\omega_i)\cos\theta_i$$

Basic path tracing algorithm

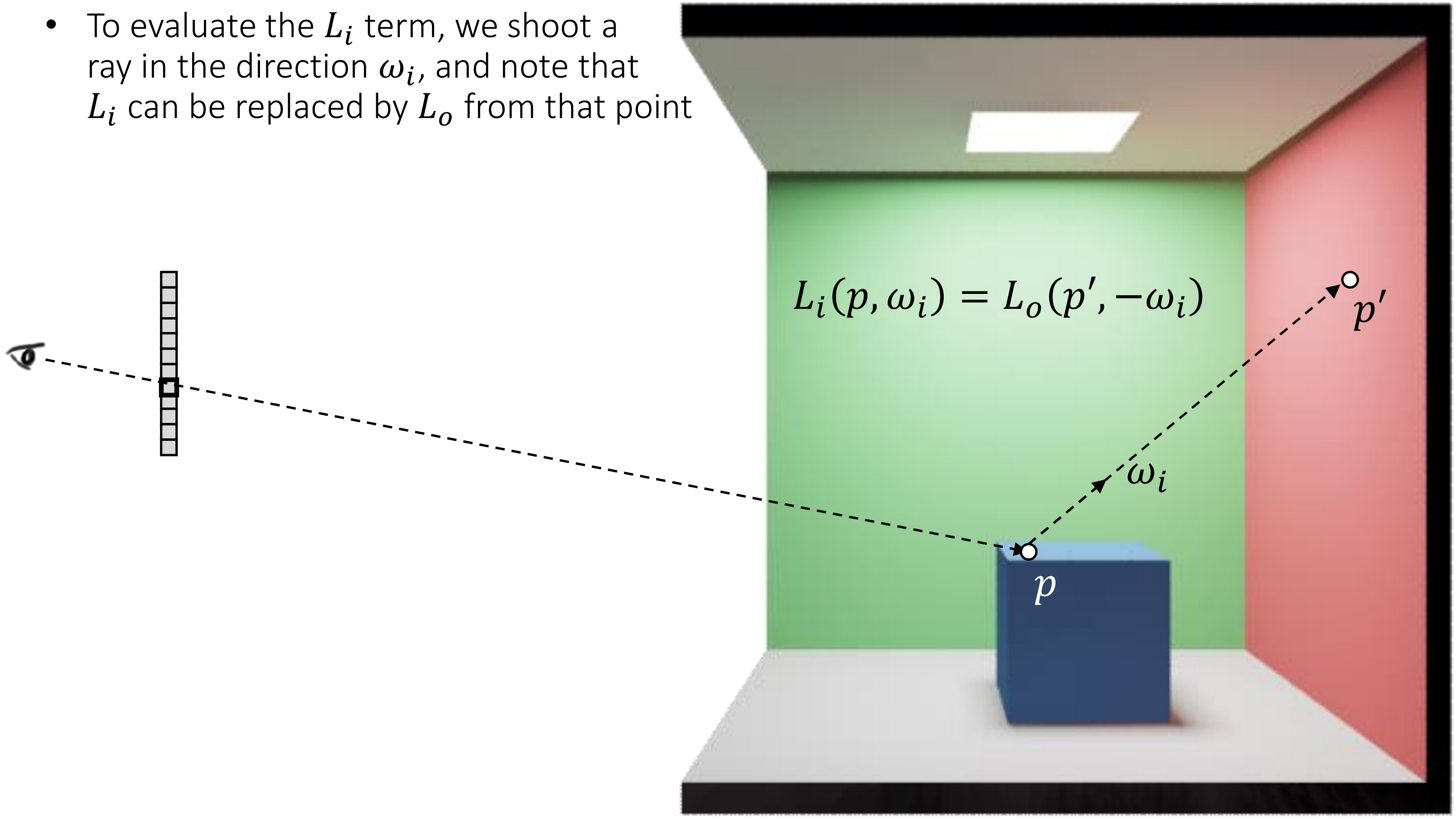
- Trace a ray through the pixel, to find the first intersection point p
- Choose a random direction ω_i on the hemisphere



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o) L_i(p,\omega_i) \cos\theta_i$$

Basic path tracing algorithm

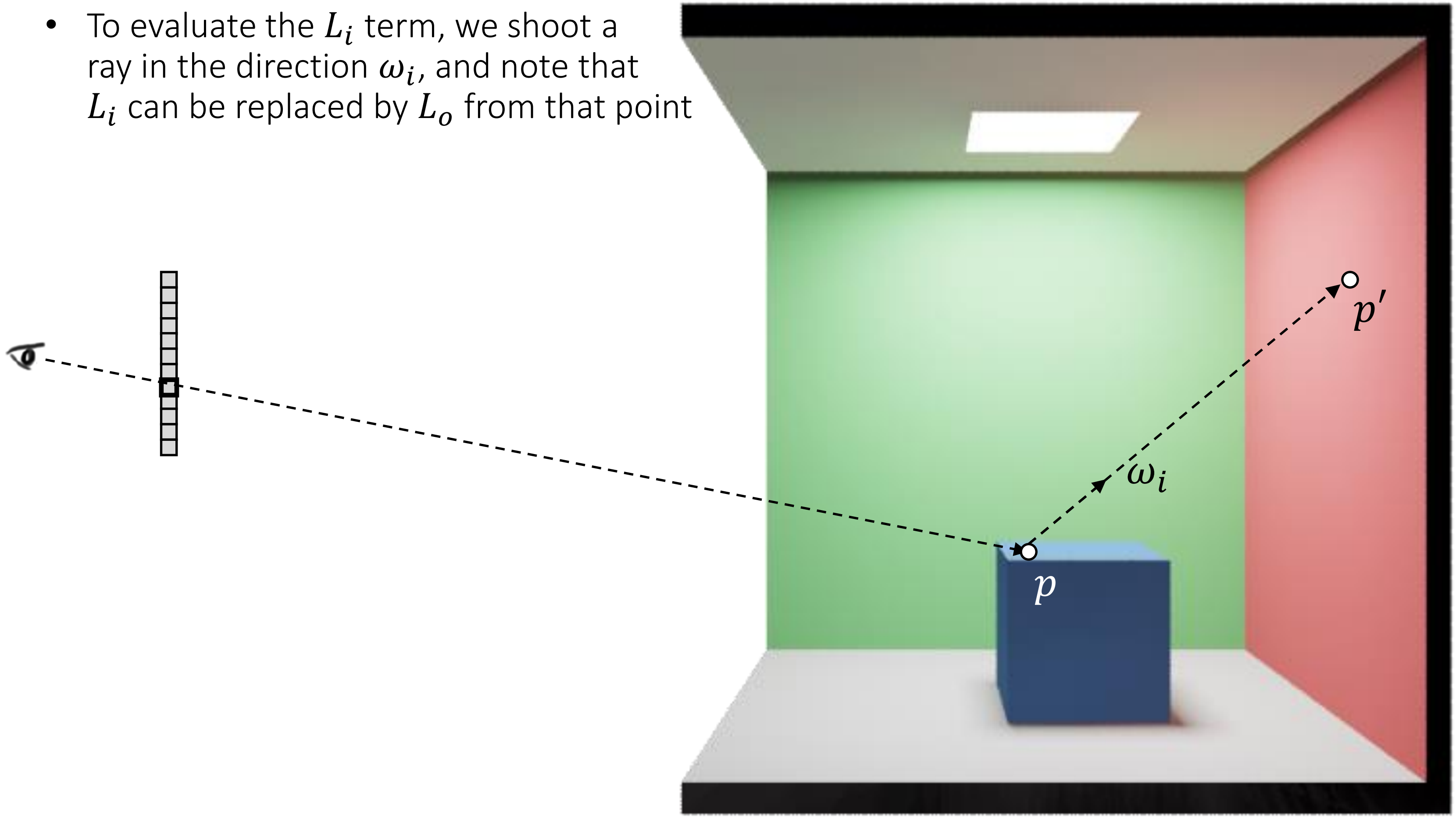
- Trace a ray through the pixel, to find the first intersection point p
- Choose a random direction ω_i on the hemisphere
- To evaluate the L_i term, we shoot a ray in the direction ω_i, and note that L_i can be replaced by L_o from that point



$$L_o(p, \omega_o) \approx 0 + 2\pi f_r(p, \omega_i, \omega_o) L_o(p', -\omega_i) \cos\theta_i$$

Basic path tracing algorithm

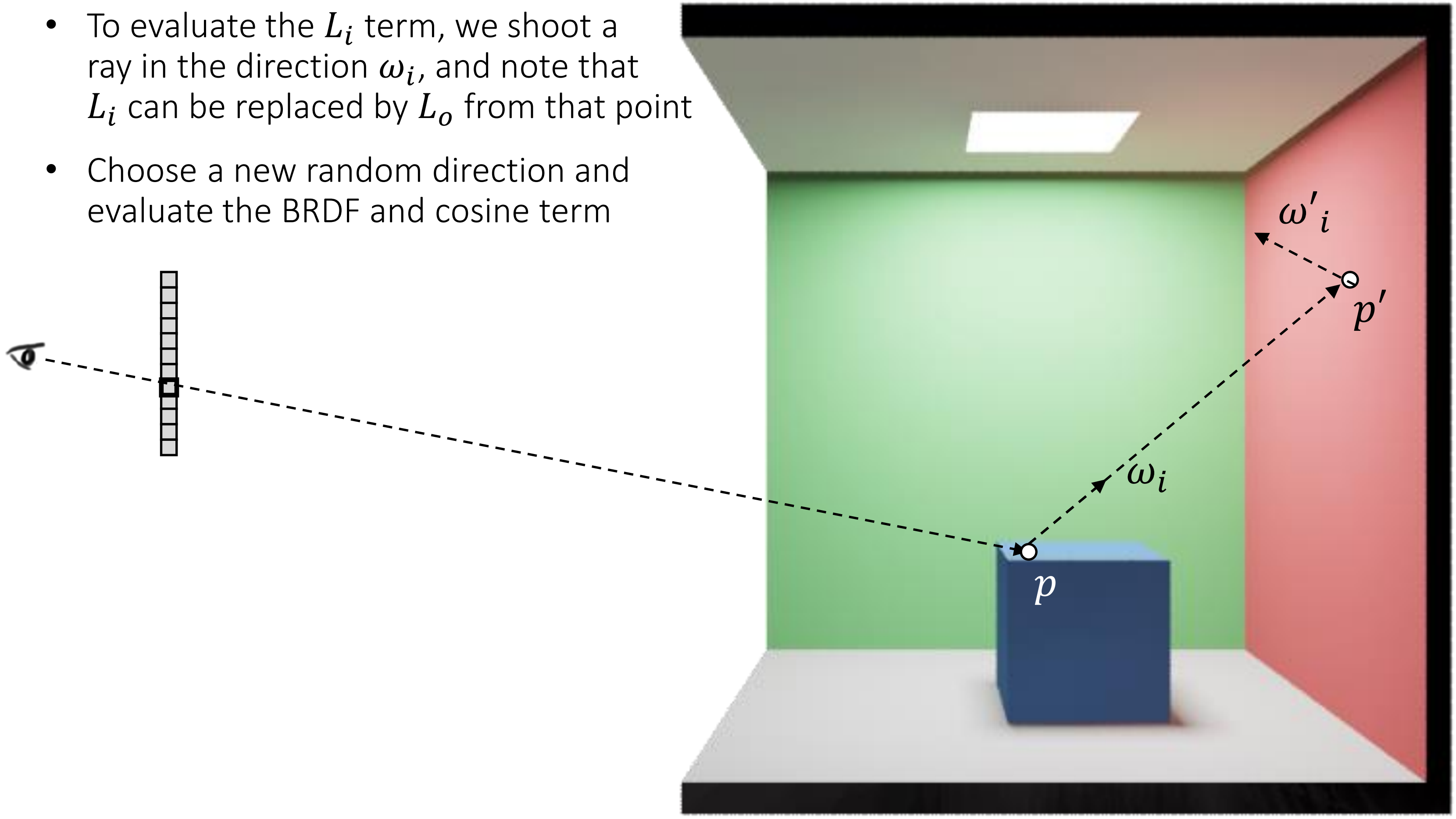
- Trace a ray through the pixel, to find the first intersection point p
- Choose a random direction ω_i on the hemisphere
- To evaluate the L_i term, we shoot a ray in the direction ω_i, and note that L_i can be replaced by L_o from that point



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o)[L_e(p',-\omega_i) + 2\pi f_r(p',\omega'_i,-\omega_i)L_i(p',\omega'_i)\cos\theta'_i]\cos\theta_i$$

Basic path tracing algorithm

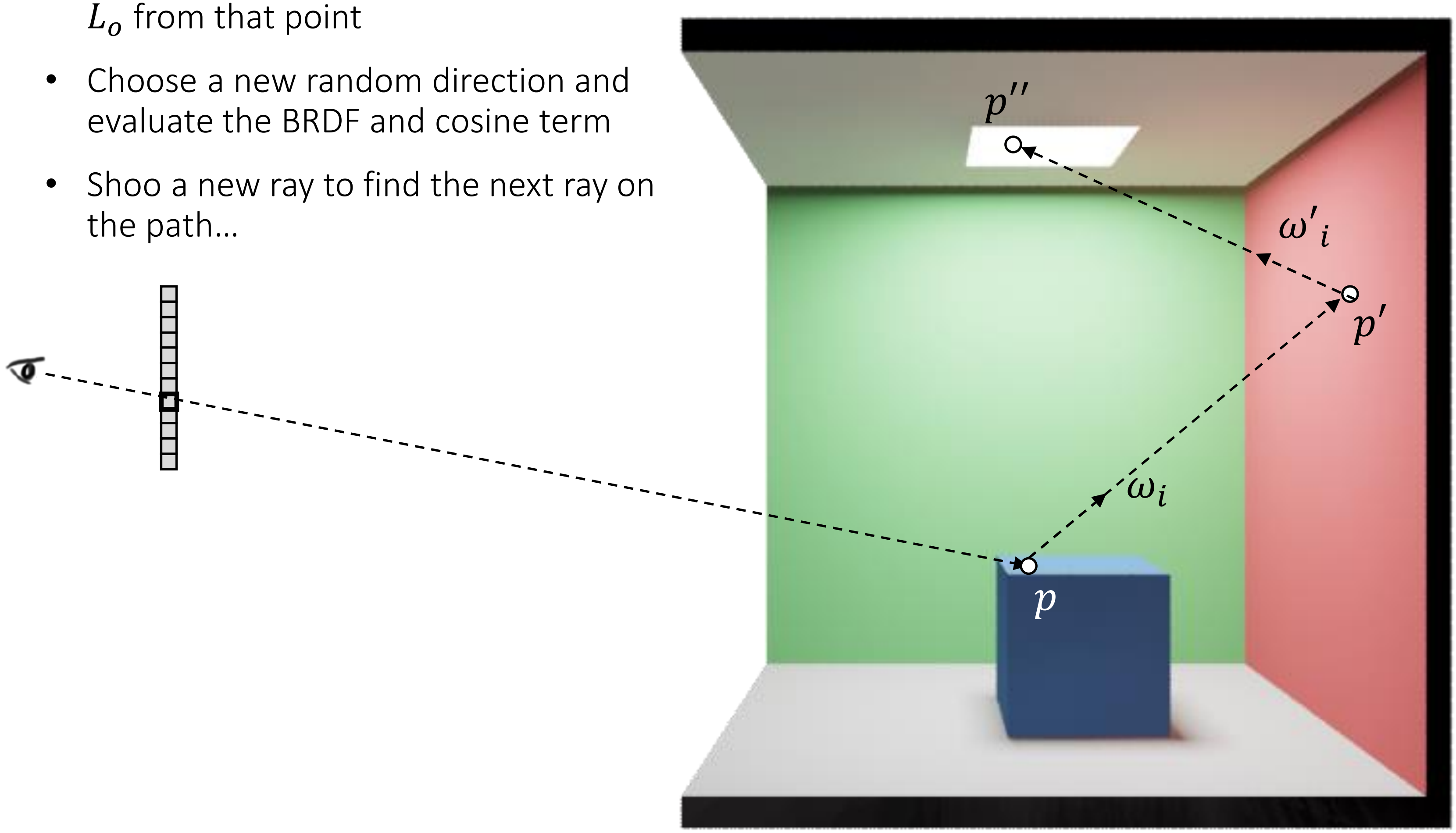
- Trace a ray through the pixel, to find the first intersection point p
- Choose a random direction ω_i on the hemisphere
- To evaluate the L_i term, we shoot a ray in the direction ω_i, and note that L_i can be replaced by L_o from that point
- Choose a new random direction and evaluate the BRDF and cosine term



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o)[0 + 2\pi f_r(p',\omega'_i,-\omega_i)L_i(p',\omega'_i)\cos\theta'_i]\cos\theta_i$$

Basic path tracing algorithm

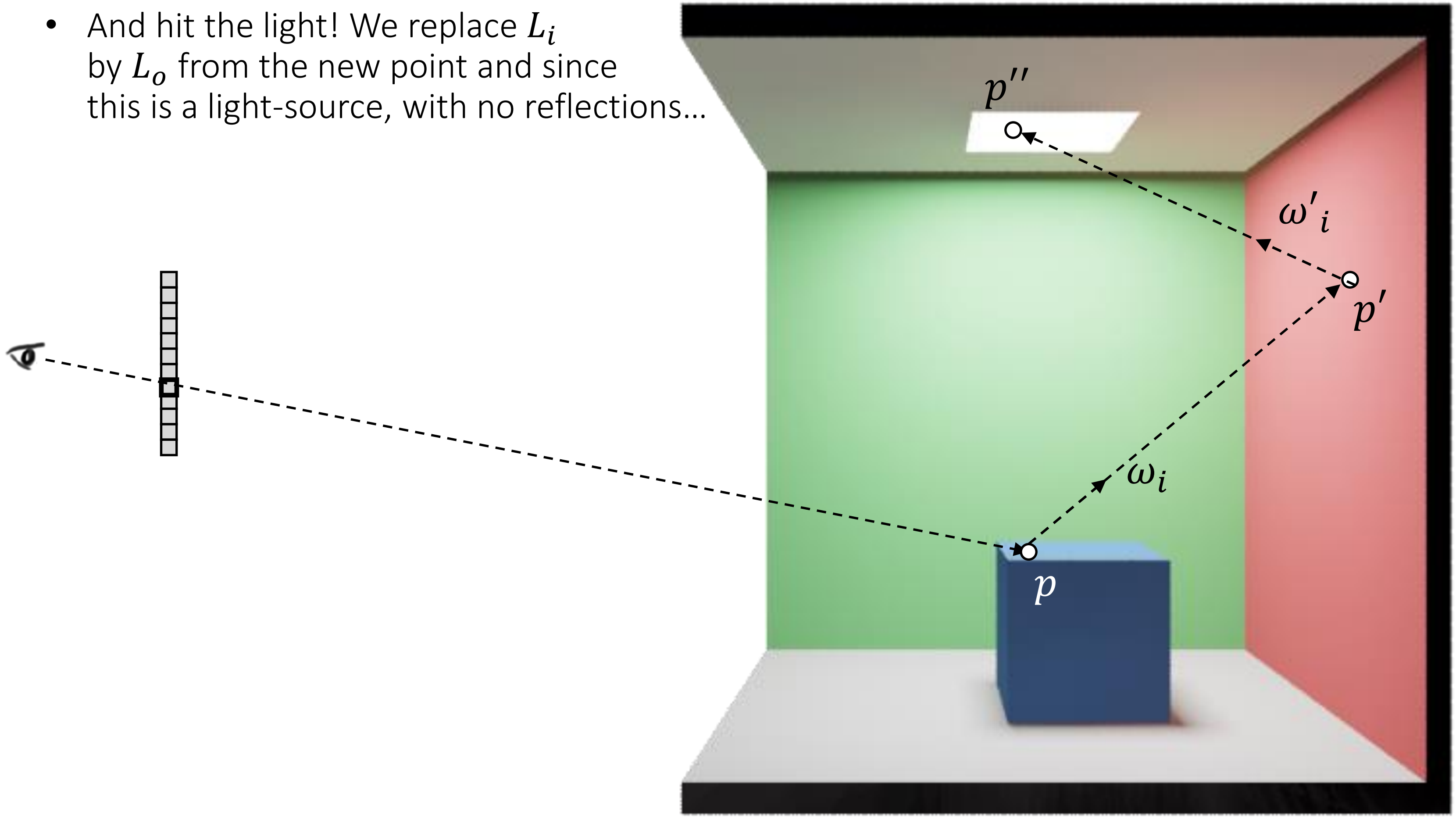
- Choose a random direction ω_i on the hemisphere
- To evaluate the L_i term, we shoot a ray in the direction ω_i, and note that L_i can be replaced by L_o from that point
- Choose a new random direction and evaluate the BRDF and cosine term
- Shoo a new ray to find the next ray on the path…



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o)[0 + 2\pi f_r(p',\omega'_i,-\omega_i)L_i(p',\omega'_i)\cos\theta'_i]\cos\theta_i$$

Basic path tracing algorithm

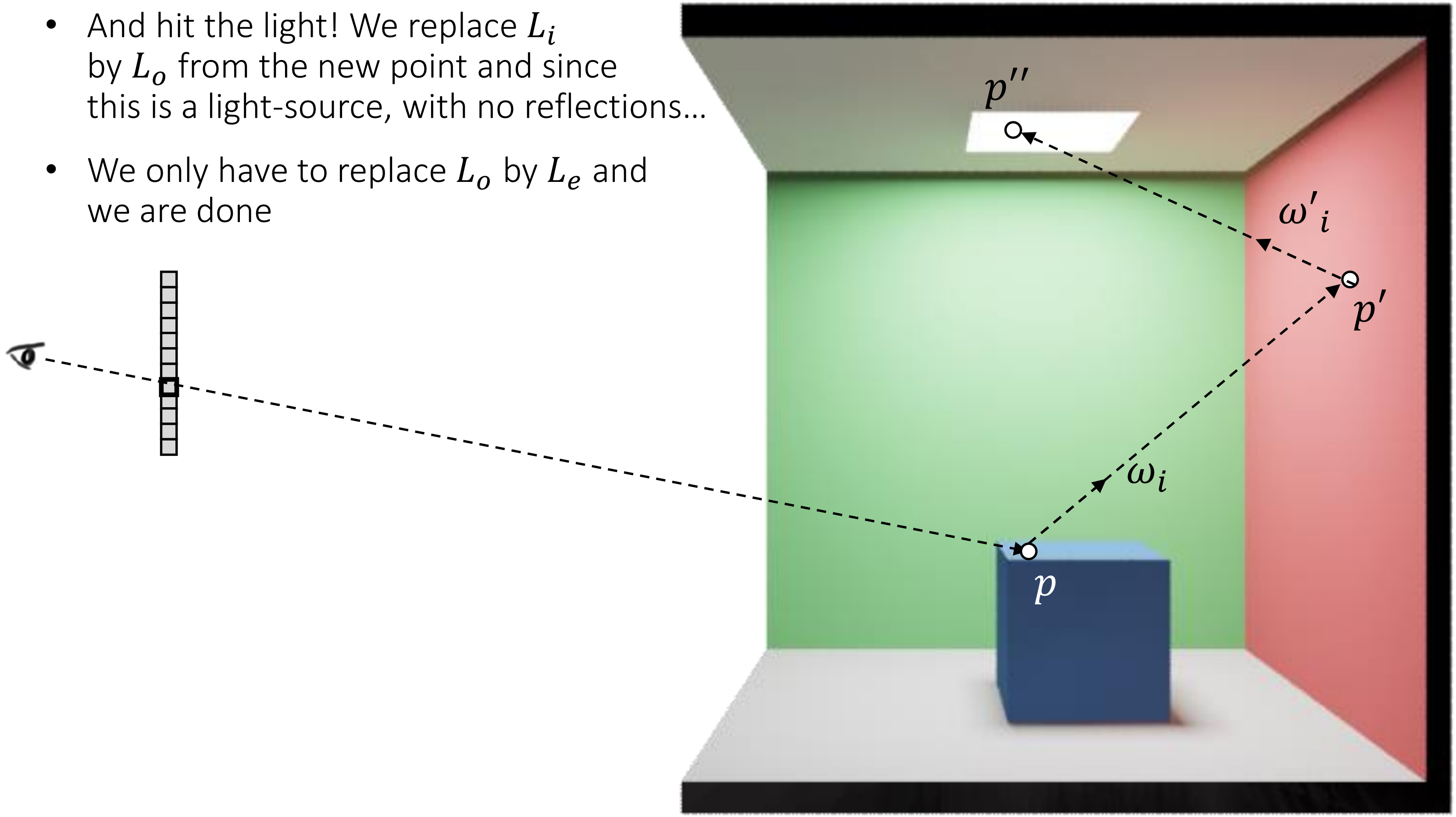
- Choose a new random direction and evaluate the BRDF and cosine term
- Shoo a new ray to find the next ray on the path…
- And hit the light! We replace L_i by L_o from the new point and since this is a light-source, with no reflections…



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o)[0 + 2\pi f_r(p',\omega'_i,-\omega_i)L_o(p'',-\omega'_i)\cos\theta'_i]\cos\theta_i$$

Basic path tracing algorithm

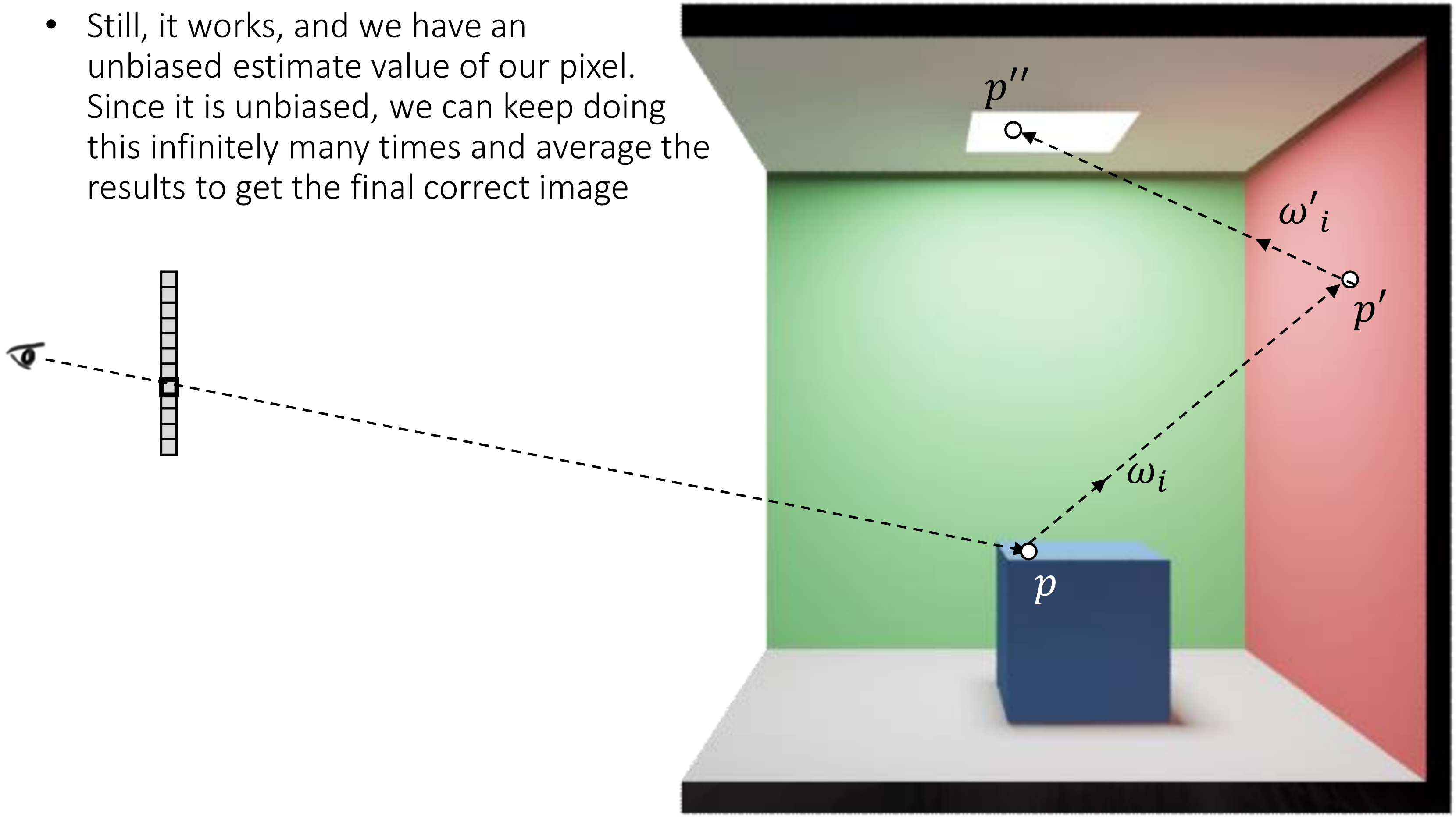
- Choose a new random direction and evaluate the BRDF and cosine term
- Shoo a new ray to find the next ray on the path…
- And hit the light! We replace L_i by L_o from the new point and since this is a light-source, with no reflections…
- We only have to replace L_o by L_e and we are done



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o)[0 + 2\pi f_r(p',\omega'_i,-\omega_i) L_e(p'',-\omega'_i) \cos\theta'_i] \cos\theta_i$$

Basic path tracing algorithm

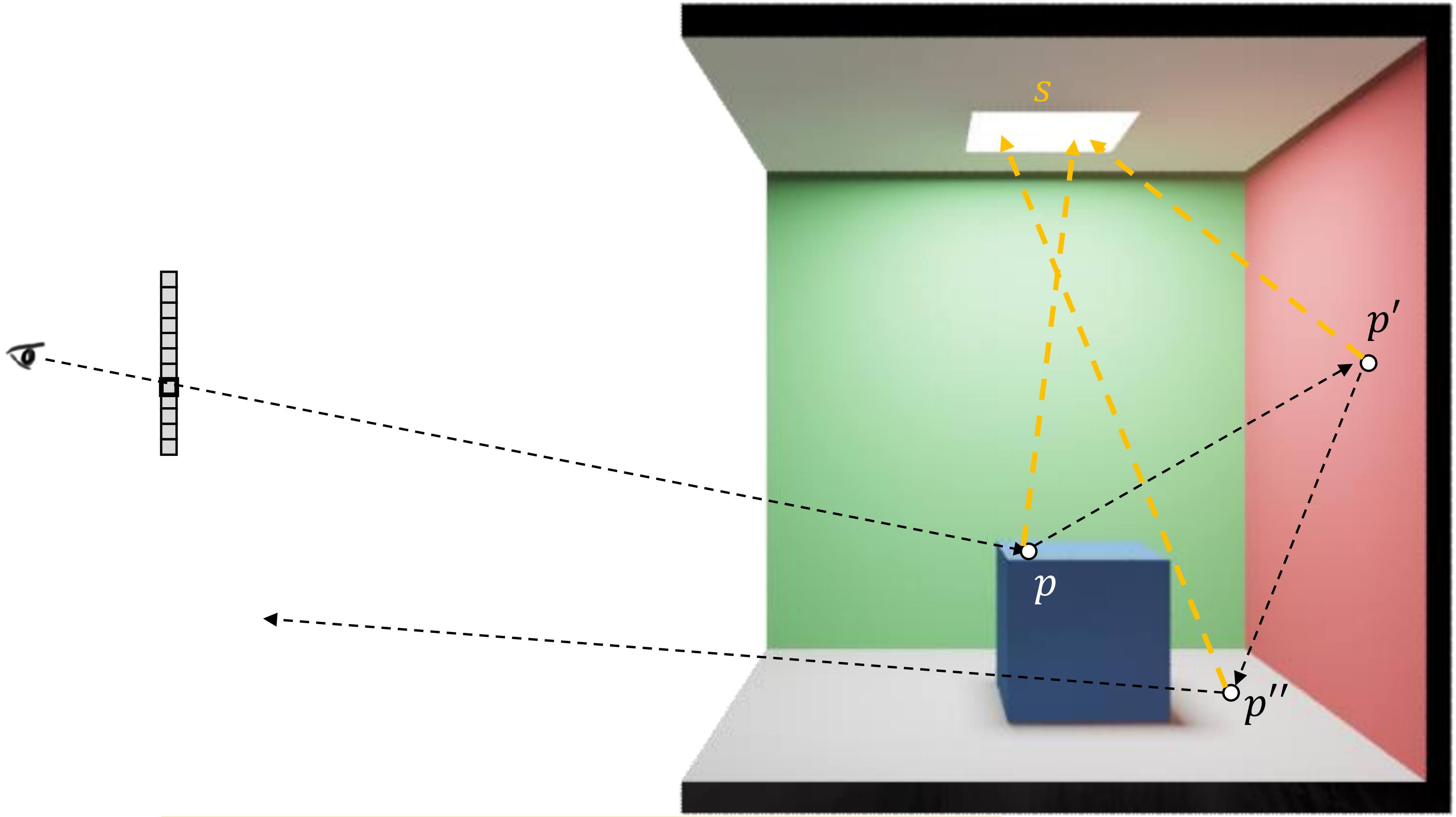
- **The problem** is that it was pure luck that we hit a light-source so soon
- For the majority of paths, the contribution will be close to zero before we ever hit a light…
- Still, it works, and we have an unbiased estimate value of our pixel. Since it is unbiased, we can keep doing this infinitely many times and average the results to get the final correct image



$$L_o(p,\omega_o) \approx 0 + 2\pi f_r(p,\omega_i,\omega_o)[0 + 2\pi f_r(p',\omega'_i,-\omega_i)L_e(p'',-\omega'_i)\cos\theta'_i]\cos\theta_i$$

Basic path tracing algorithm

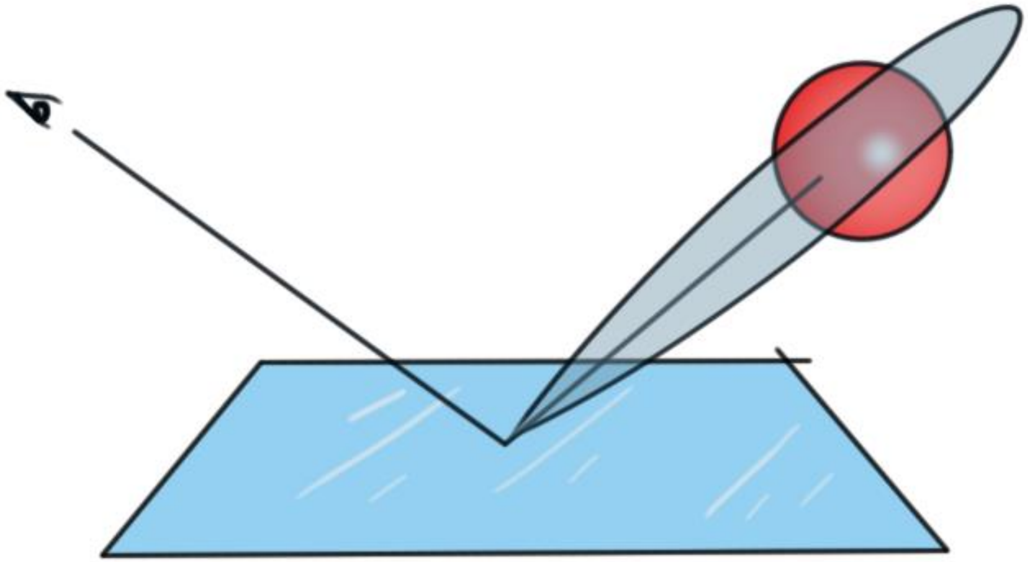
- **The problem** is that it was pure luck that we hit a light-source so soon
- **Soluition:** separate direct and indirect illumination



$$L_o(p,\omega_o) = \underbrace{\int_S f_r(p, p \rightarrow q, \omega_o) L_e(s, s \rightarrow p) G(p,s) V(p,s) ds}_{\text{direct}} + \underbrace{\int_{H^2(\vec{n})} f_r(p, \omega_i, \omega_o) L_i(p, \omega_i) \cos\theta_i d\omega_i}_{\text{indirect}}$$

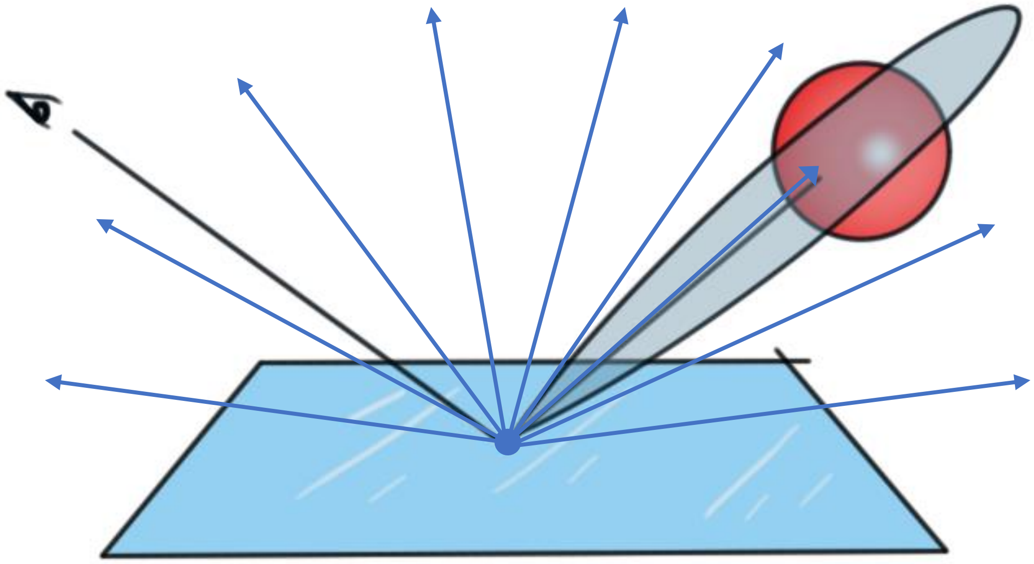
Importance Sampling

- So far we have sampled incoming light uniformly over the hemisphere. Why is that a bad idea?



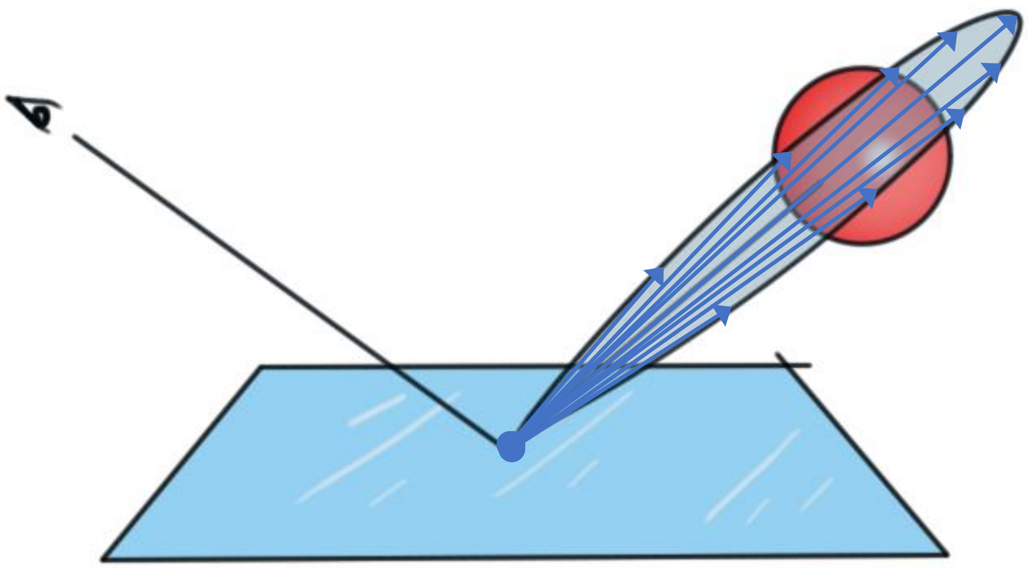
Importance Sampling

- So far we have sampled incoming light uniformly over the hemisphere. Why is that a bad idea?



Importance Sampling

- So far we have sampled incoming light uniformly over the hemisphere. Why is that a bad idea?
- We want to shoot more samples where the function we are integrating is high!
- One common type of importance sampling is to create a distribution that resembles the BRDF



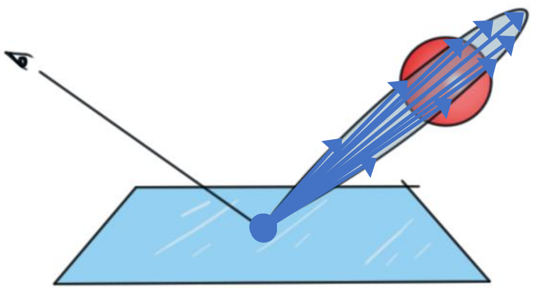
Importance Sampling

$$L_o(p,\omega_o) \approx L_e(p,\omega_o) + \frac{1}{N}\sum_{i=0}^{N}\frac{f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i}{p(\omega_i)}$$

- We need to make sure our PDF is not low where the function we are sampling can be high
  - Or we will accumulate samples with extremely high variance

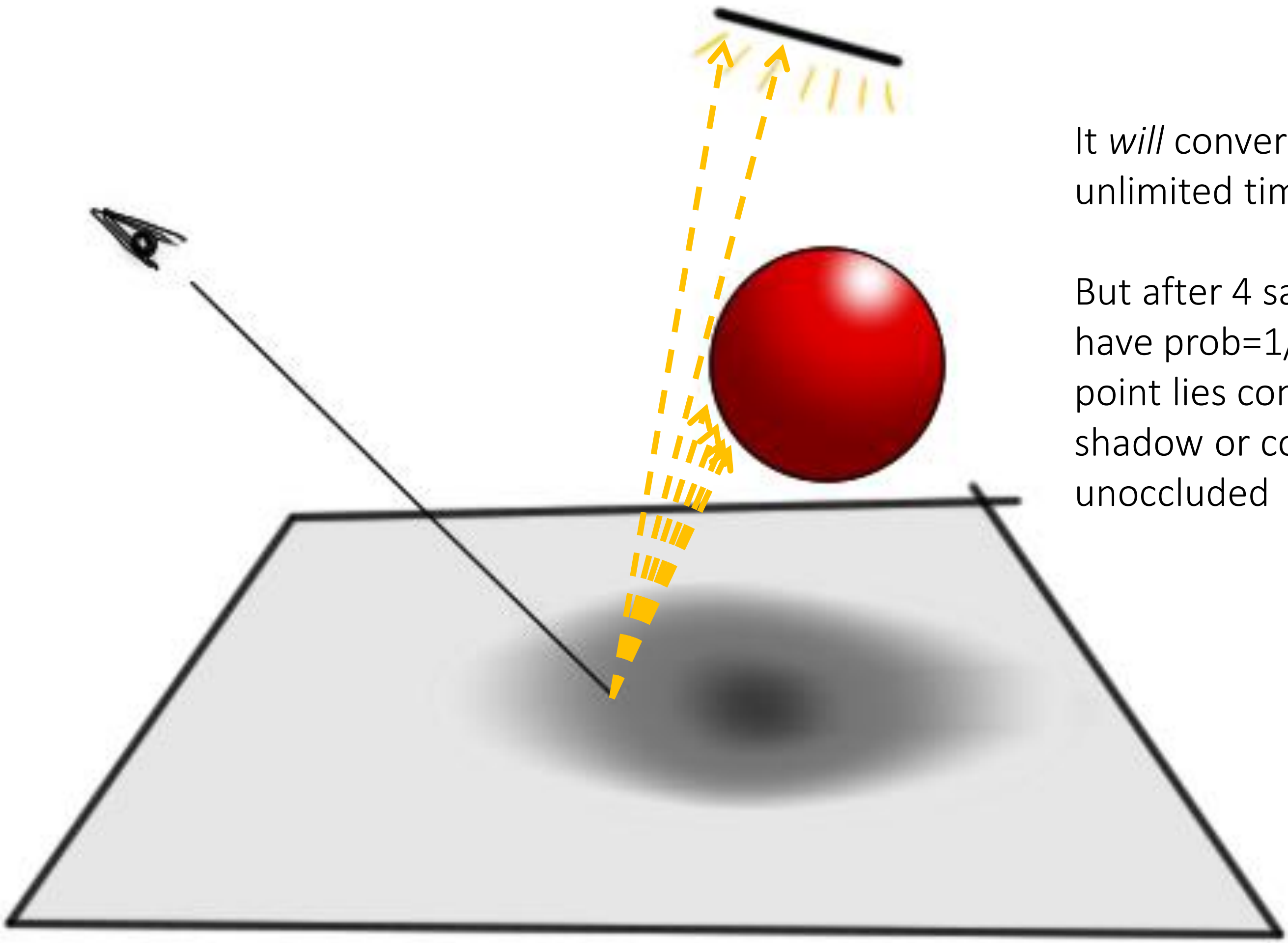
- **Example:** We can always generate samples with cosine distribution:

$$L_o(p,\omega_o) \approx L_e(p,\omega_o) + \frac{1}{N}\sum_{i=0}^{N}\frac{f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i}{\frac{\cos\theta_i}{\pi}} = \frac{\pi}{N}\sum_{i=0}^{N} f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)$$



Stratified Sampling

- Another standard variance reduction method
- When just choosing samples randomly over the domain, they may “clump” and take a long while to converge

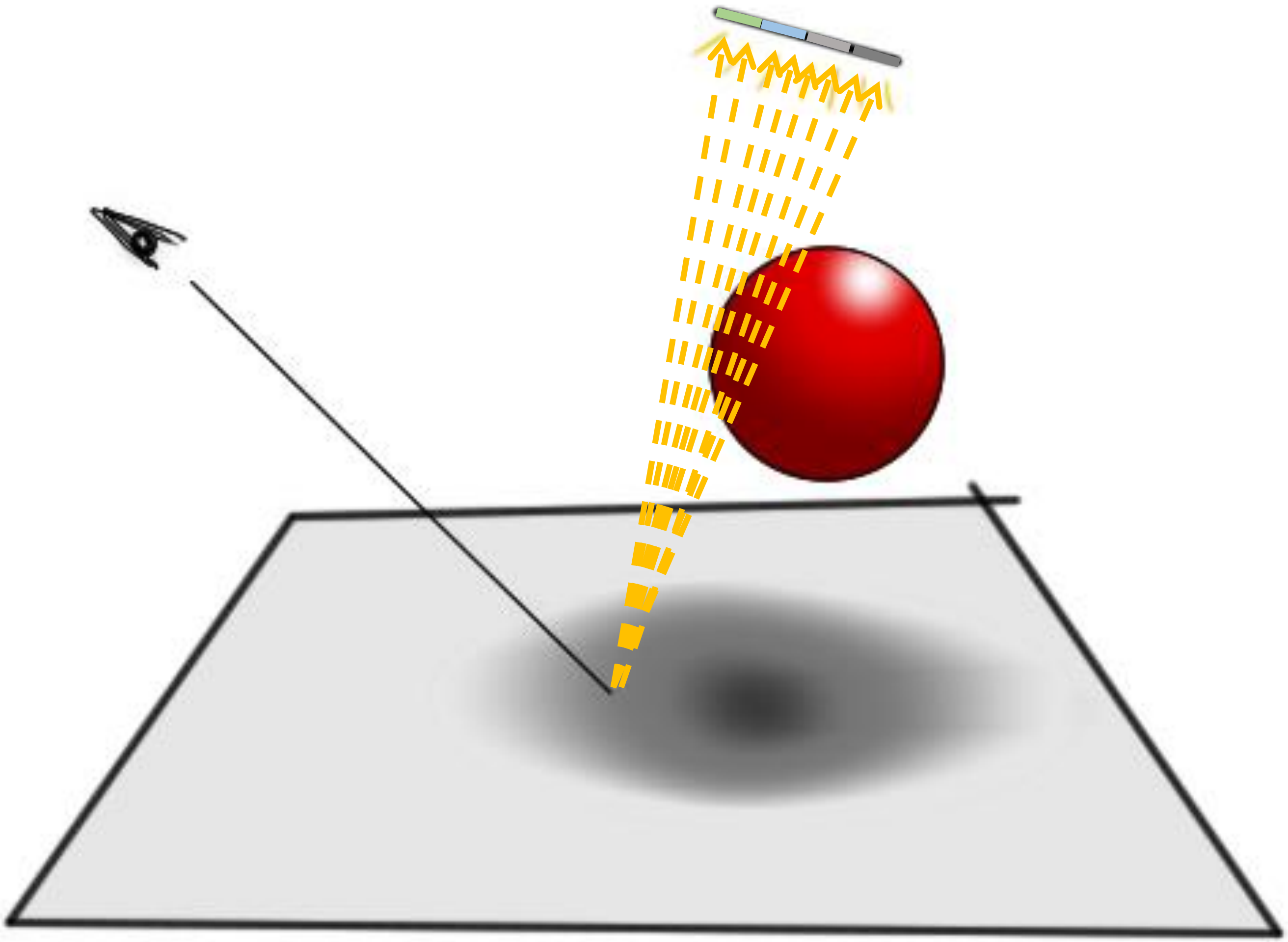


It *will* converge to 0.5 after unlimited time

But after 4 samples it may have prob=1/8 that the point lies completely in shadow or completely unoccluded

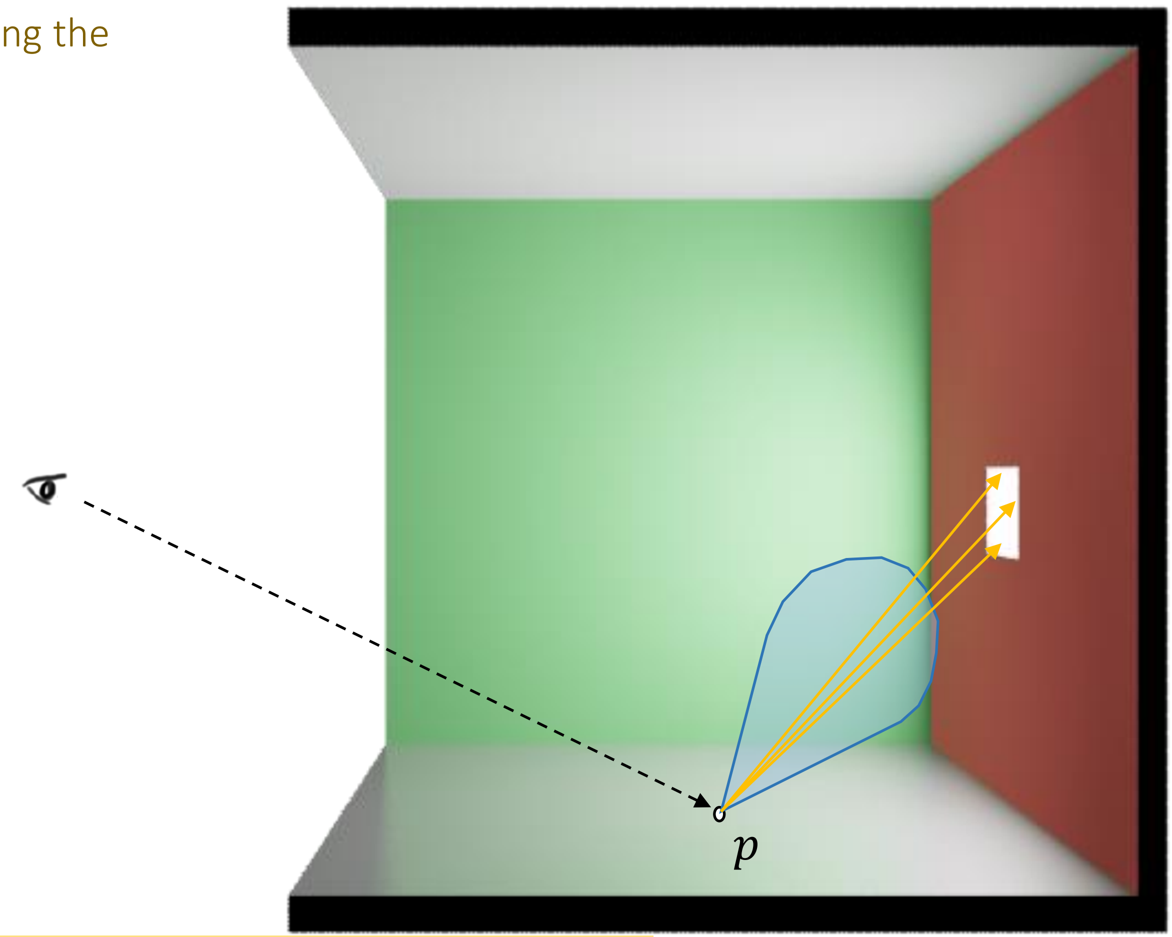
Divide domain into “strata”

- Don’t sample one strata again until all others have been sampled once



Multiple Importance Sampling

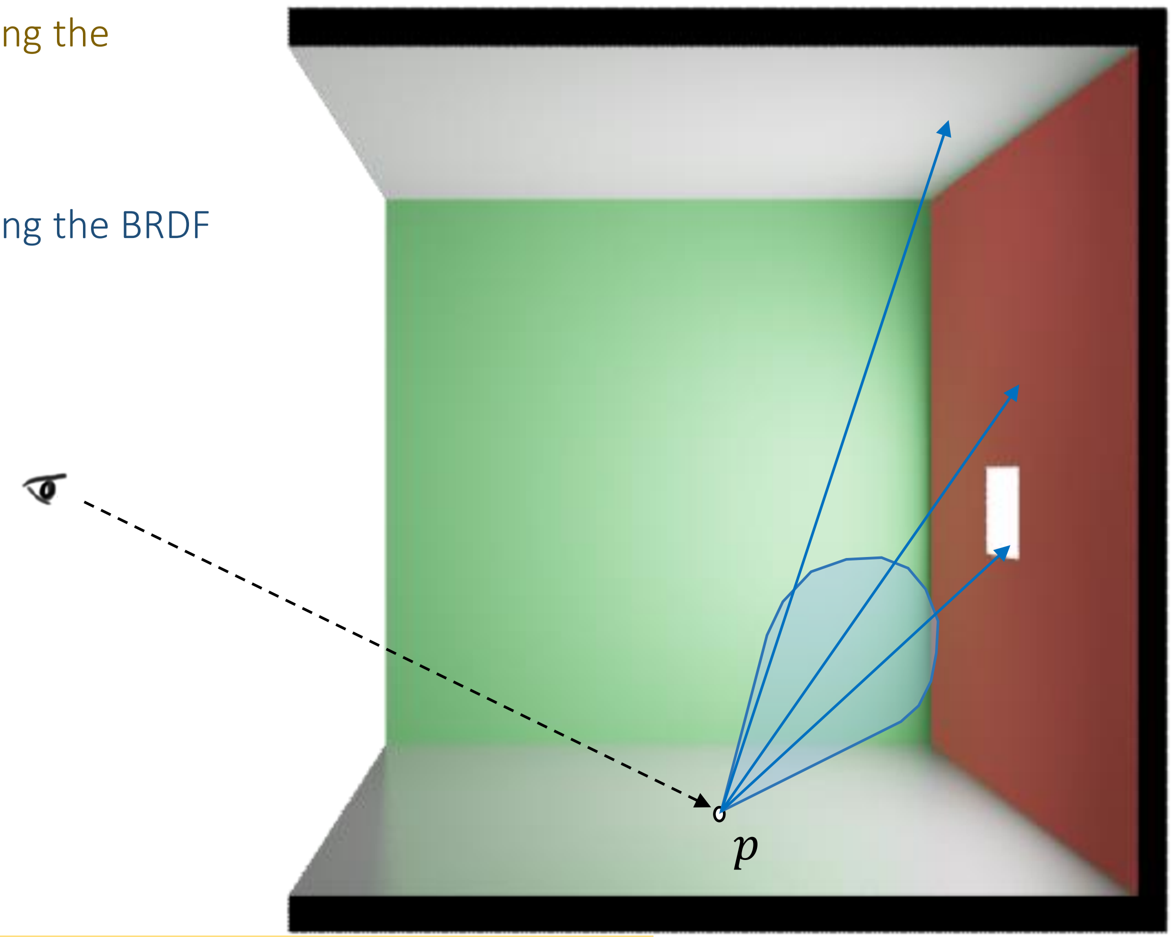
- Small light source, diffuse surface
- Direct Illumination
  - Stochastic sampling the light sources



$$L_o(p,\omega_o) = \underbrace{\int_S f_r(p, p\to q, \omega_o) L_e(s, s\to p) G(p,s) V(p,s) ds}_{\text{direct}} + \underbrace{\int_{H^2(\vec{n})} f_r(p,\omega_i,\omega_o) L_i(p,\omega_i) \cos\theta_i d\omega_i}_{\text{indirect}}$$

Multiple Importance Sampling

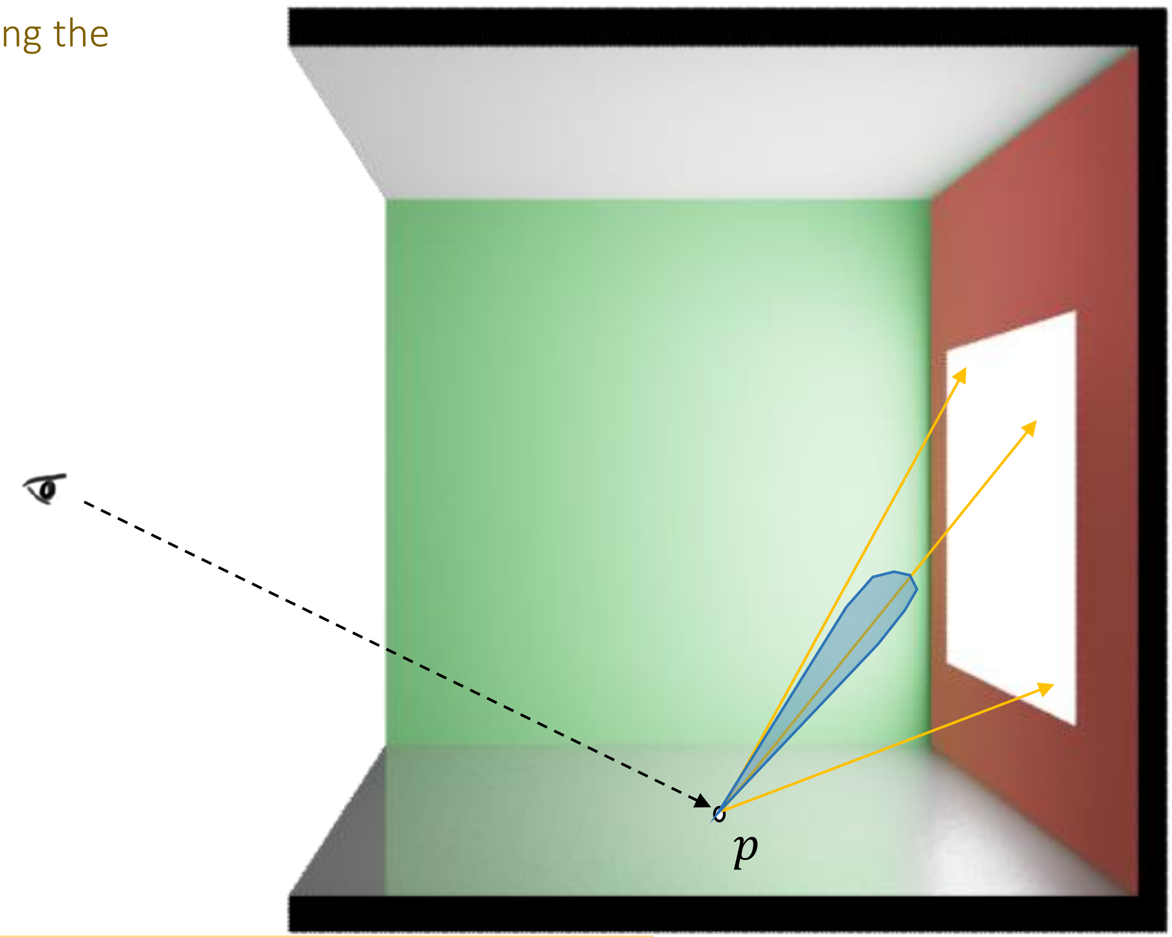
- Small light source, diffuse surface
- Direct Illumination
  - Stochastic sampling the light sources
- Indirect Illumination
  - Stochastic sampling the BRDF



$$L_o(p,\omega_o) = \underbrace{\int_S f_r(p, p \rightarrow q, \omega_o) L_e(s, s \rightarrow p) G(p,s) V(p,s) ds}_{\text{direct}} + \underbrace{\int_{H^2(\vec{n})} f_r(p,\omega_i,\omega_o) L_i(p,\omega_i) \cos\theta_i d\omega_i}_{\text{indirect}}$$

Multiple Importance Sampling

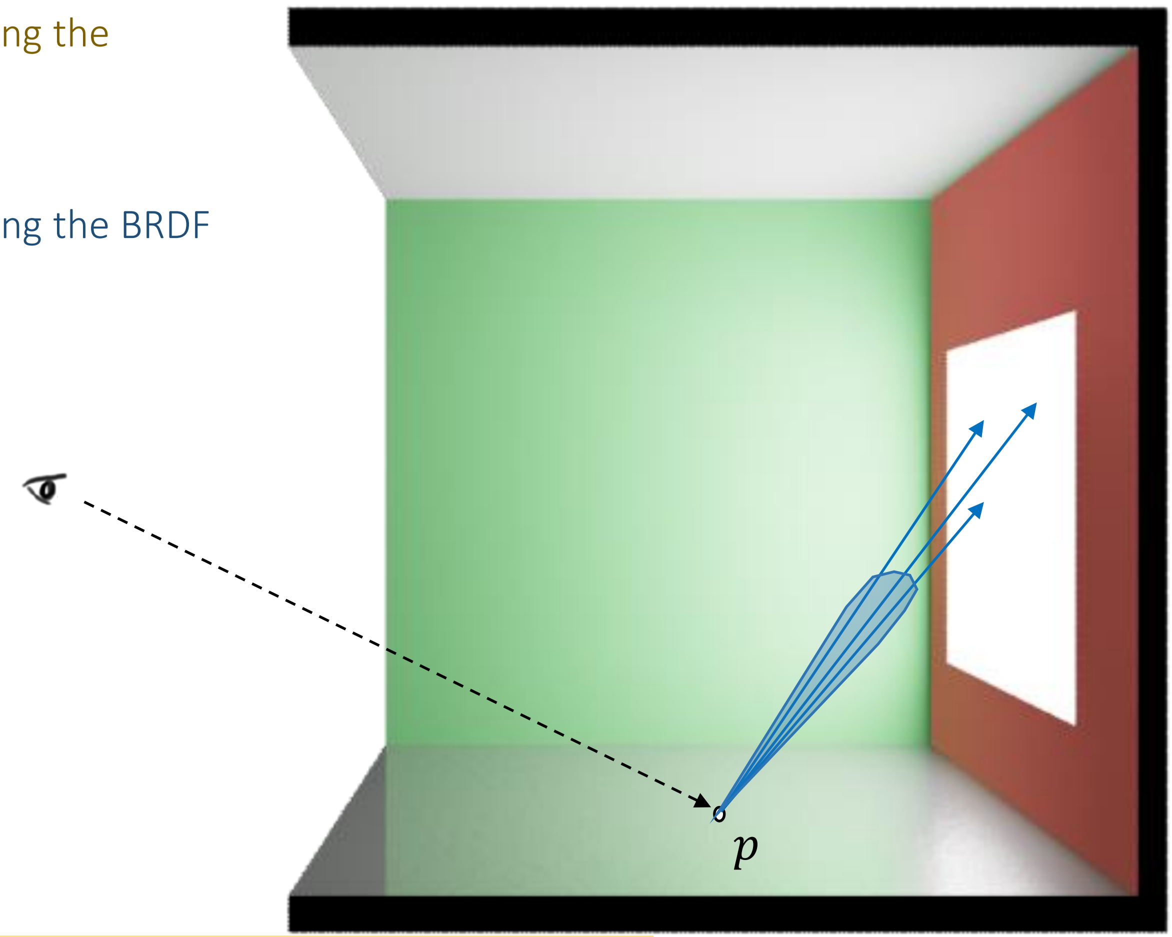
- **Problem:** large light source, specullar surface
- Direct Illumination
  - Stochastic sampling the light sources



$$L_o(p,\omega_o) = \underbrace{\int_S f_r(p,p\to q,\omega_o)L_e(s,s\to p)G(p,s)V(p,s)ds}_{\text{direct}} + \underbrace{\int_{H^2(\vec{n})} f_r(p,\omega_i,\omega_o)L_i(p,\omega_i)\cos\theta_i d\omega_i}_{\text{indirect}}$$

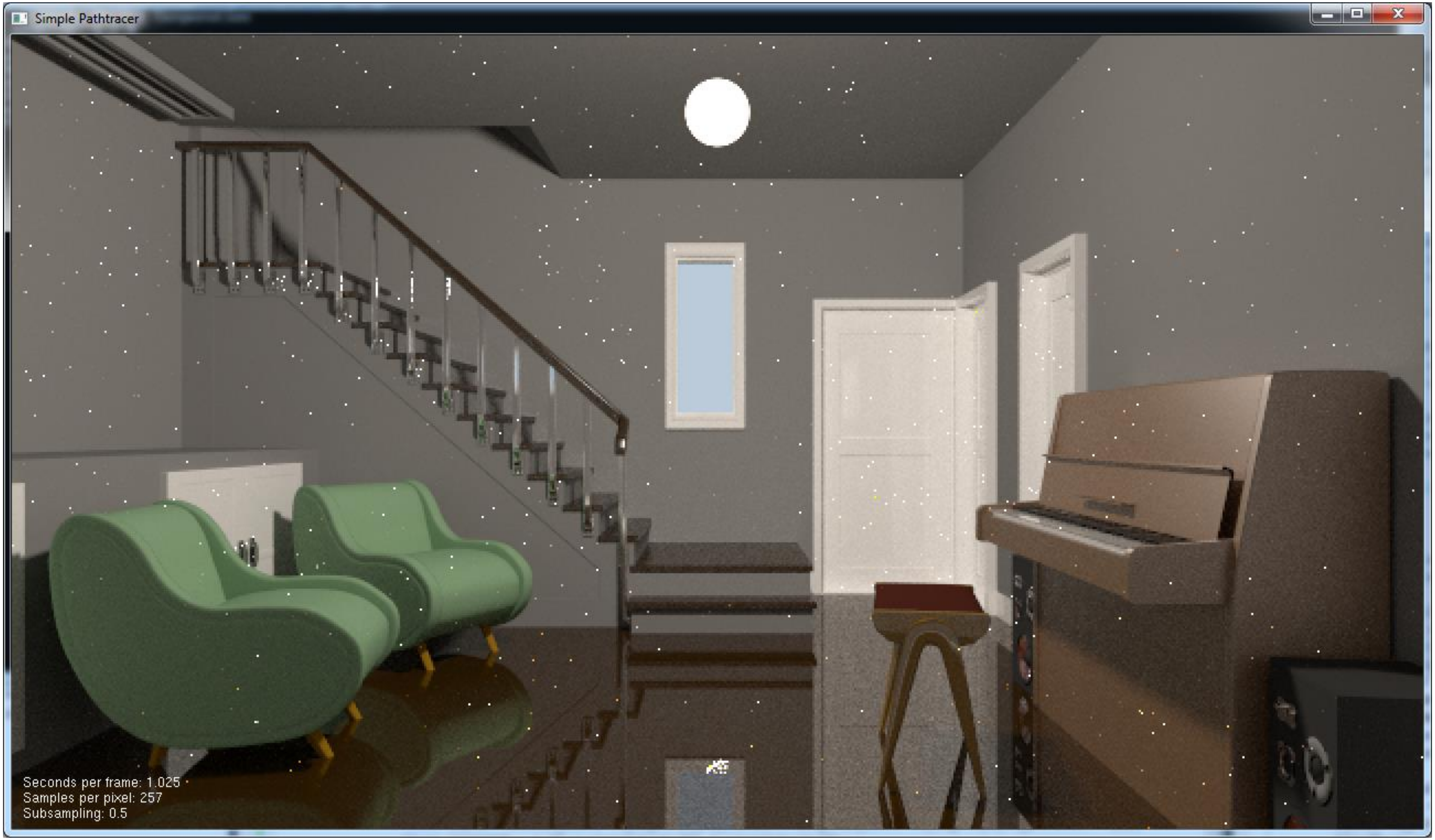
Multiple Importance Sampling

- **Problem:** large light source, specullar surface
- Direct Illumination
  - Stochastic sampling the light sources
- Indirect Illumination
  - Stochastic sampling the BRDF



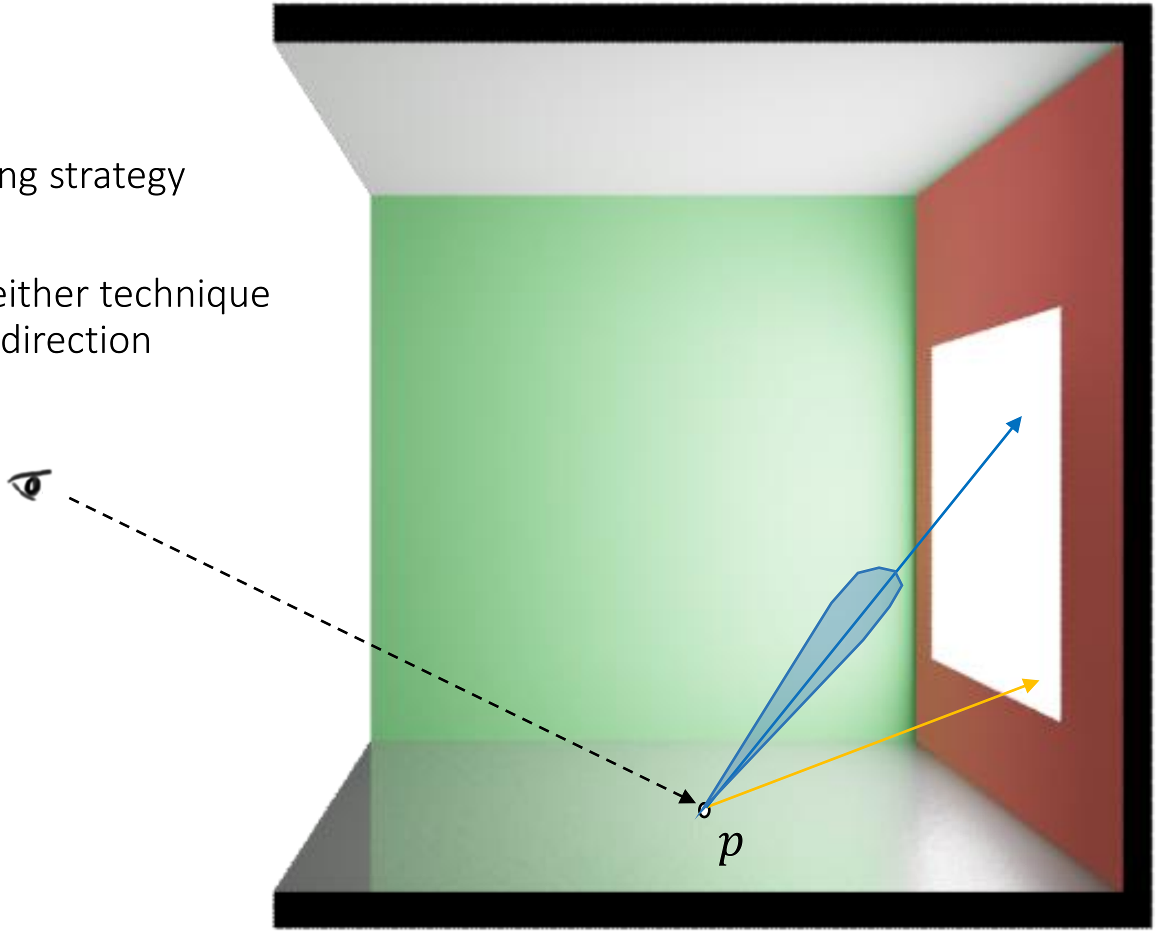
$$L_o(p,\omega_o) = \underbrace{\int_S f_r(p, p \to q, \omega_o) L_e(s, s \to p) G(p,s) V(p,s) ds}_{\text{direct}} + \underbrace{\int_{H^2(\vec{n})} f_r(p, \omega_i, \omega_o) L_i(p, \omega_i) \cos\theta_i d\omega_i}_{\text{indirect}}$$

Multiple Importance Sampling

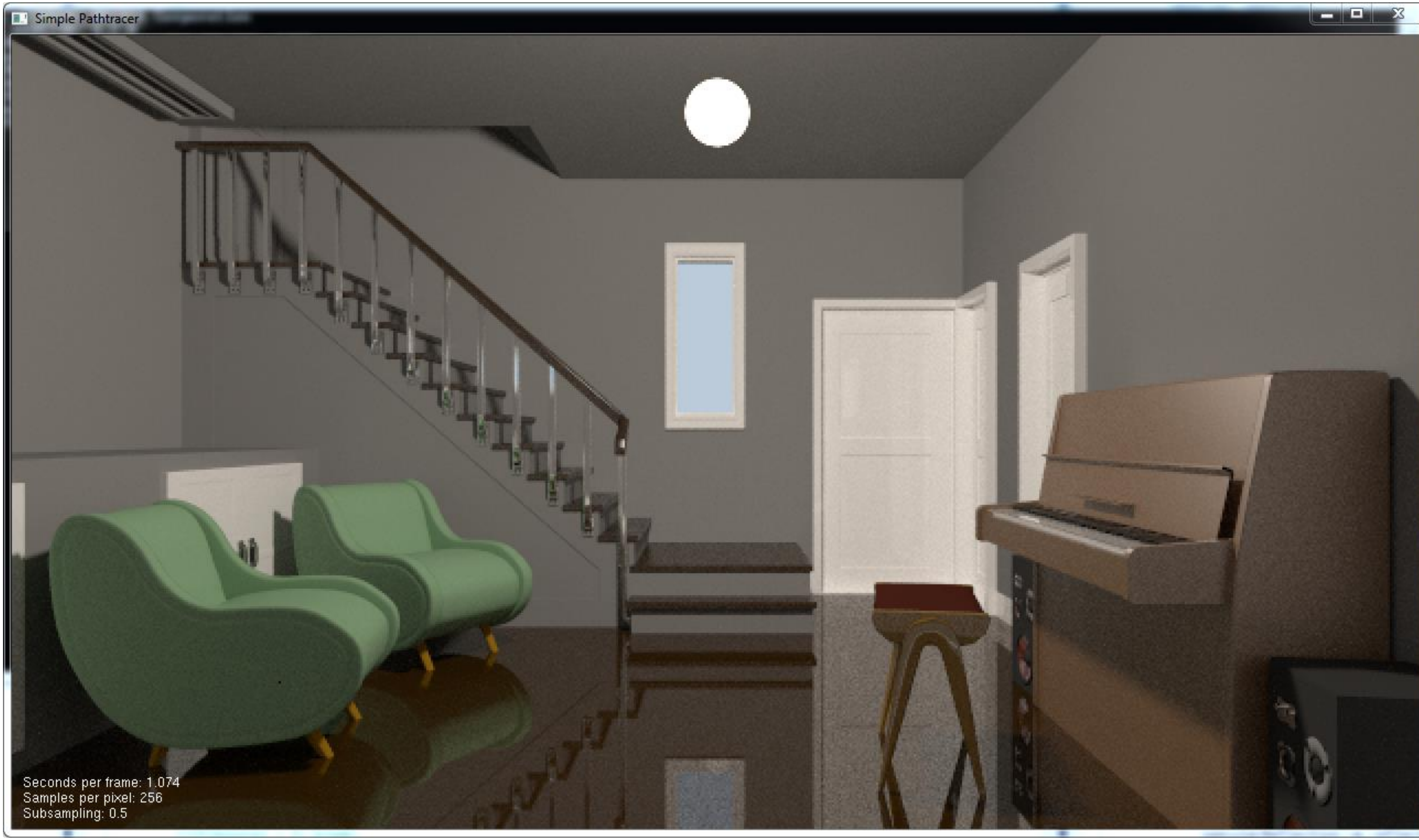


Multiple Importance Sampling

- **Problem:** large light source, specullar surface
- Solution: Sample both BRDF and the light source
  - Sample BRDF first
  - Then light
  - PDF of this sampling strategy is (weighted) sum
  - Only very low if neither technique is likely to choose direction

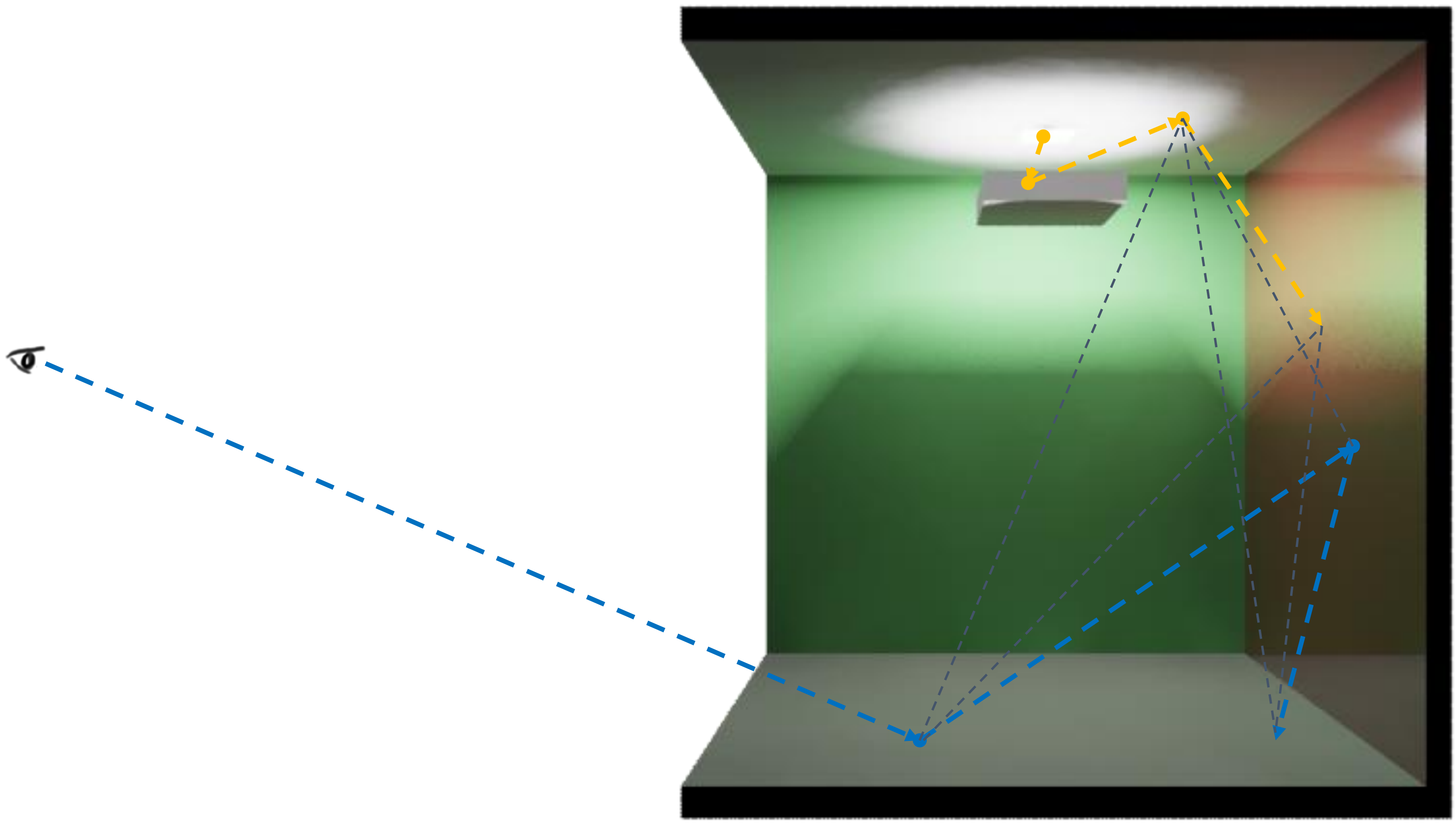


Multiple Importance Sampling



Bidirectional path tracing is a combination of

- Shooting rays from the light sources and creating paths in the scene
- Gathering rays from a point on a surface

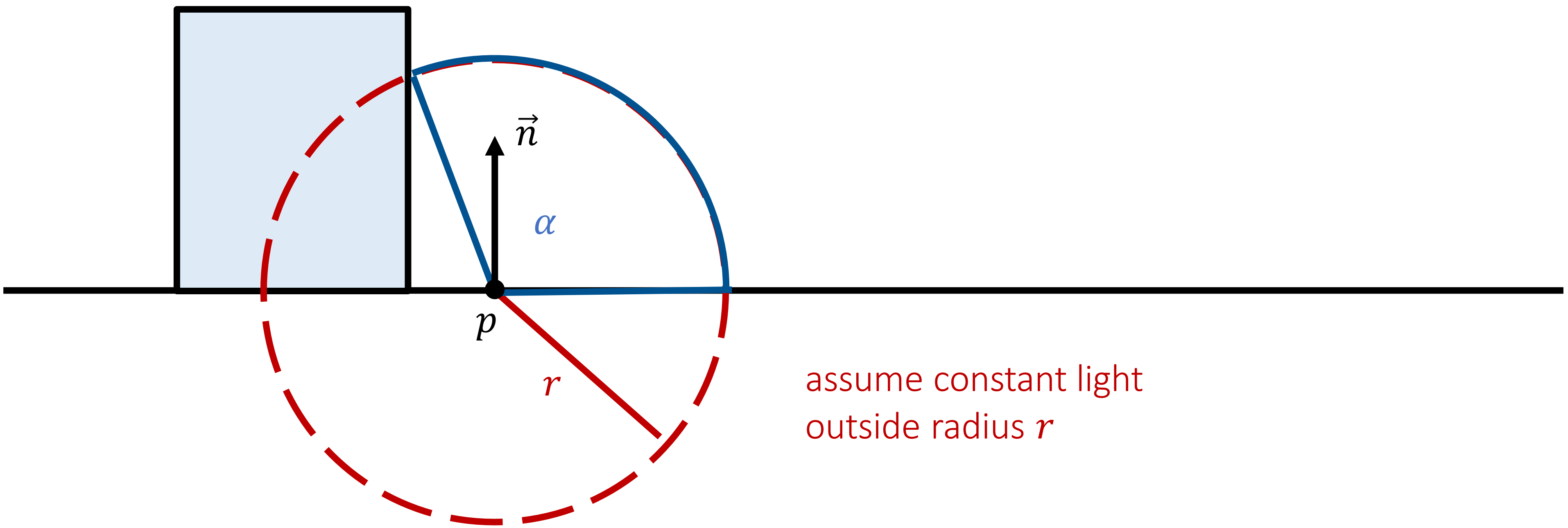


Bidirectional path tracing

Ambient Occlusion

Calculates shadows against assumed constant ambient illumination

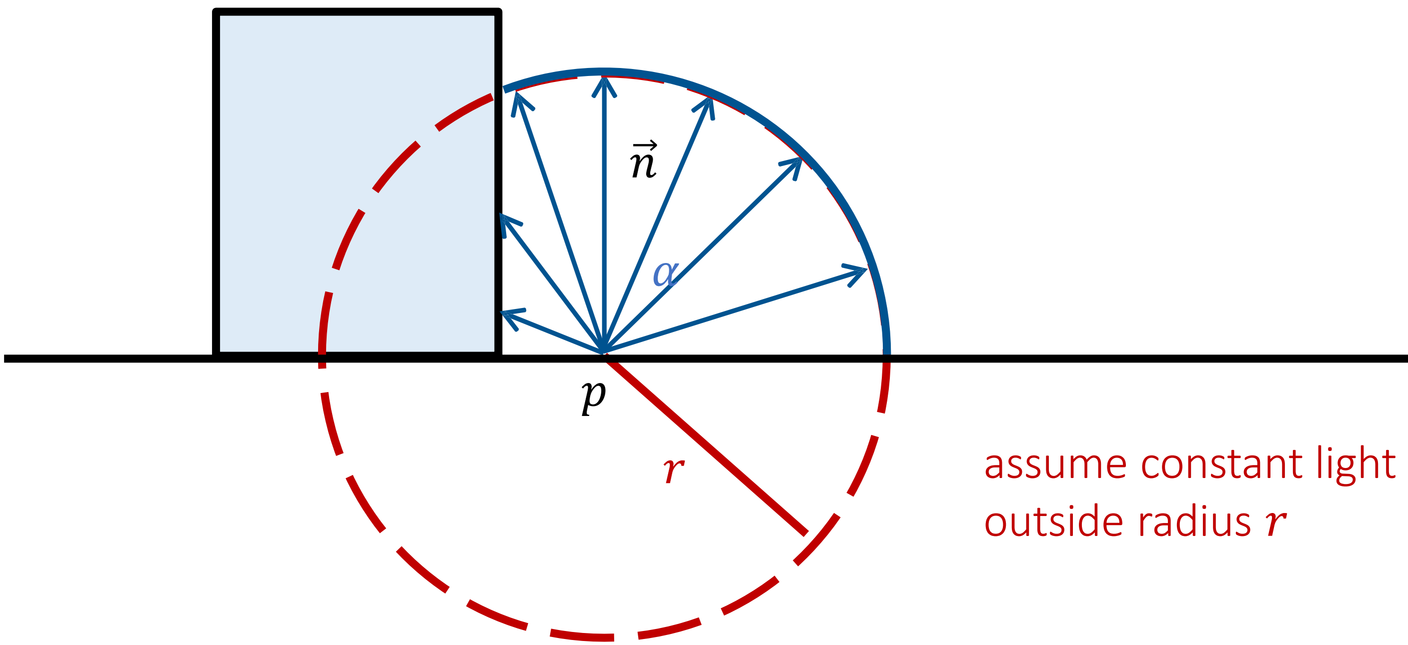
- **Idea**: in most environments, multiple light bounces lead to a very smooth component in the overall illumination
- For this component, incident light on a point is proportional to the part of the environment (opening angle) visible from the point
- Describes well contact shadows, dark corners



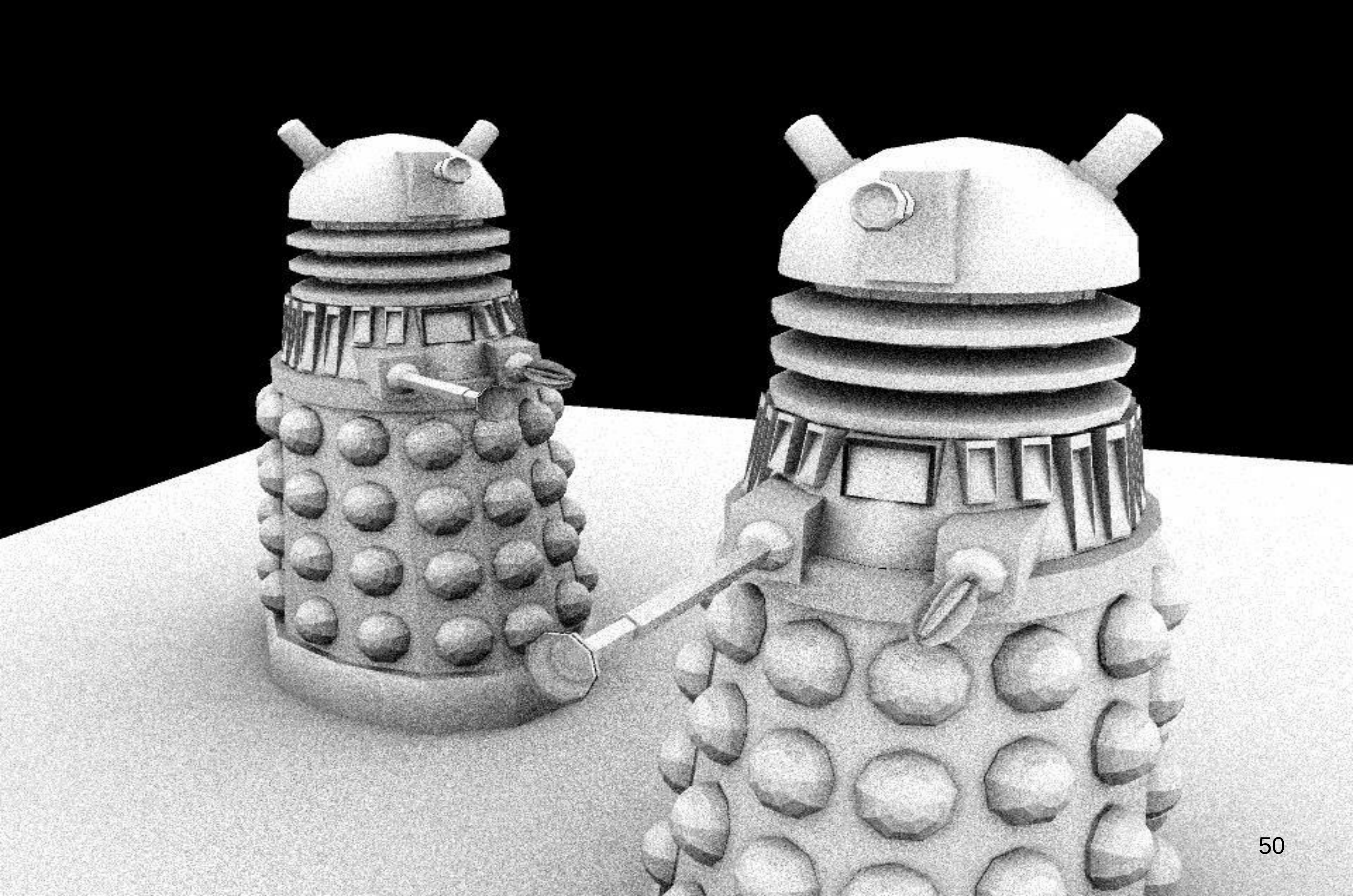
Example: visibility map

Computation using Ray-Tracing is straightforward

- Start at point p
- Sample N directions $(\omega_1, \omega_2, \ldots, \omega_N)$ from upper hemisphere (*e.g.* using cosine-weighted hemisphere sampler)
- Transform the samples from their coordinate to the object's coordinate system
- Shot shadow rays from p to ω_i with maximum length r (*i.e.*: `ray.t = r`)
- Count how many directions are occluded



Ambient Occlusion



Ambient Occlusion

Without ambient occlusion

With ambient occlusion

Ambient Occlusion