COL781: Computer Graphics

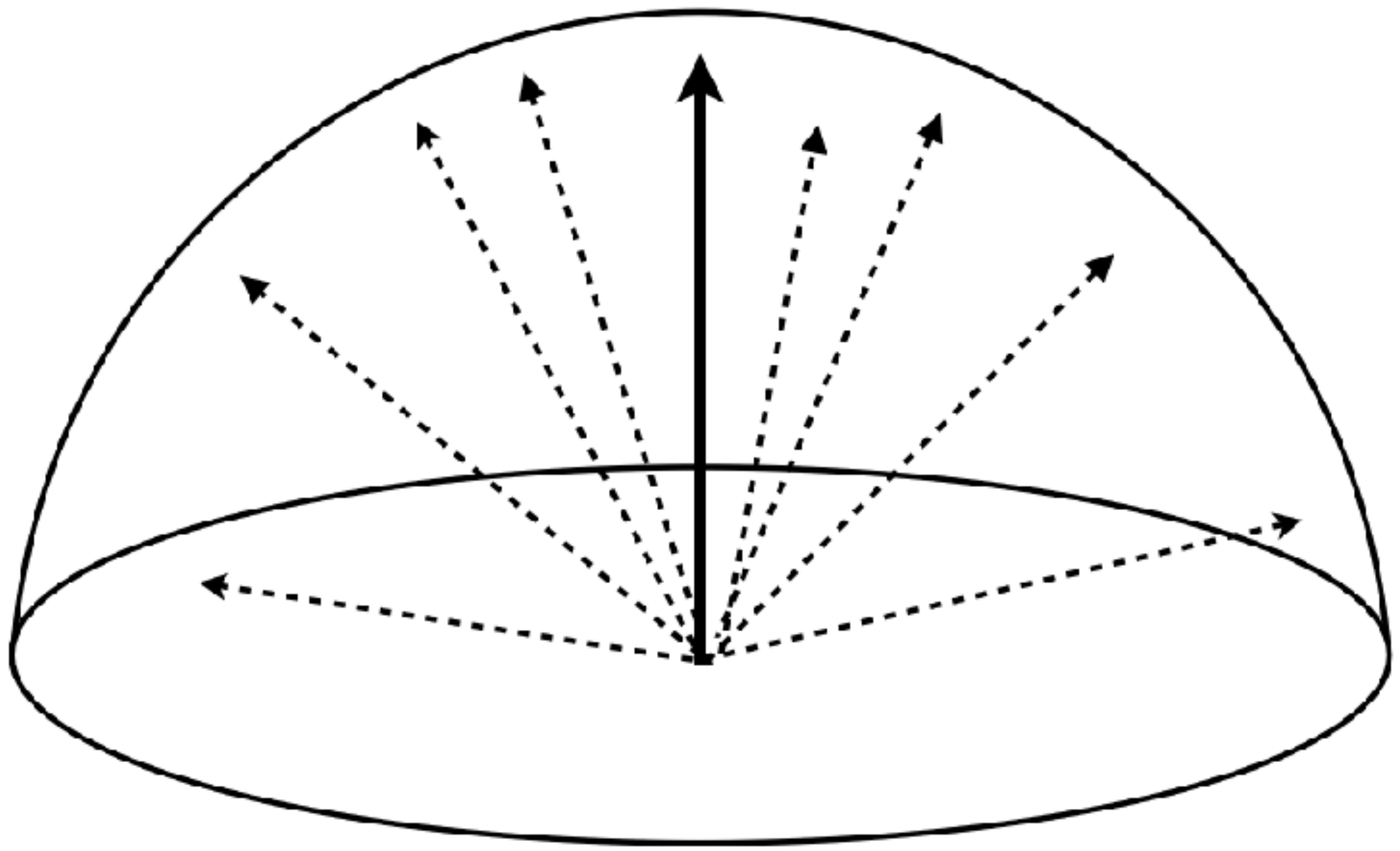
26. Bidirectional Methods

Christensen et al. 2018

Homework exercise

Find a way to sample directions on the hemisphere according to the cosine-weighted distribution, $p(\boldsymbol{\omega}) = \cos(\theta)/\pi$.

(A very nice geometrical approach exists, but a straightforward application of inversion sampling should also work.)



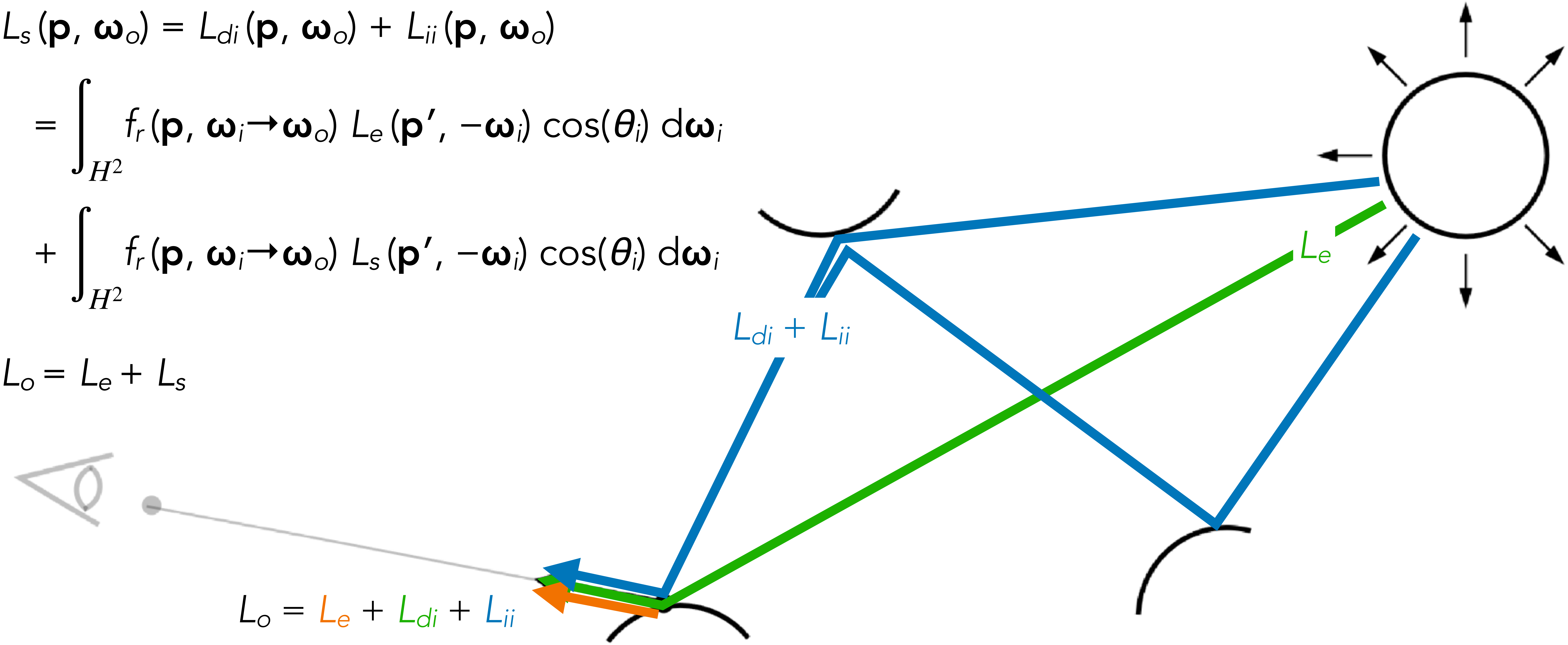
Next event estimation

$$L_s(\mathbf{p}, \boldsymbol{\omega}_o) = L_{di}(\mathbf{p}, \boldsymbol{\omega}_o) + L_{ii}(\mathbf{p}, \boldsymbol{\omega}_o)$$

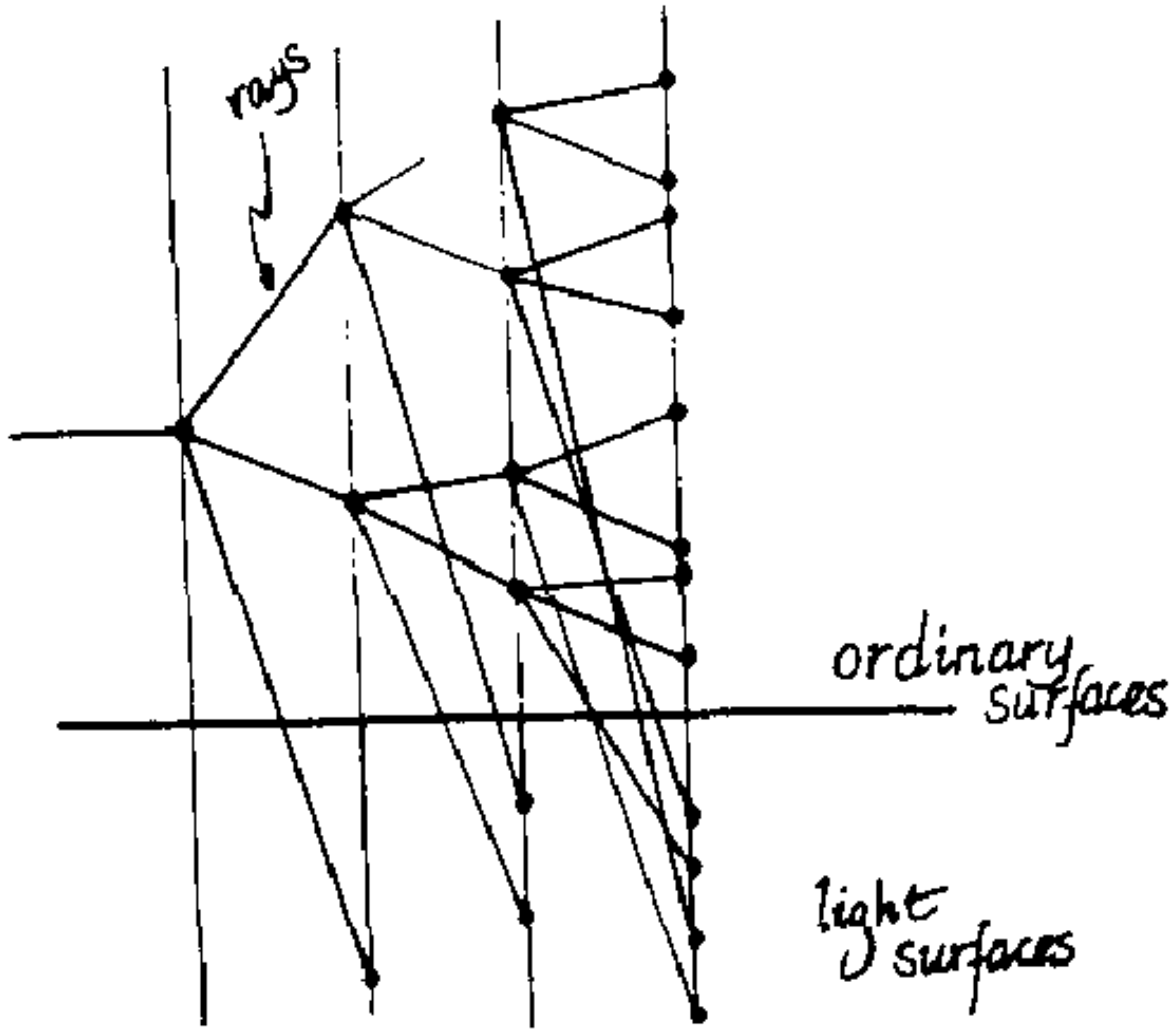
$$= \int_{H^2} f_r(\mathbf{p}, \boldsymbol{\omega}_i \rightarrow \boldsymbol{\omega}_o)\, L_e(\mathbf{p}', -\boldsymbol{\omega}_i) \cos(\theta_i)\, d\boldsymbol{\omega}_i$$

$$+ \int_{H^2} f_r(\mathbf{p}, \boldsymbol{\omega}_i \rightarrow \boldsymbol{\omega}_o)\, L_s(\mathbf{p}', -\boldsymbol{\omega}_i) \cos(\theta_i)\, d\boldsymbol{\omega}_i$$

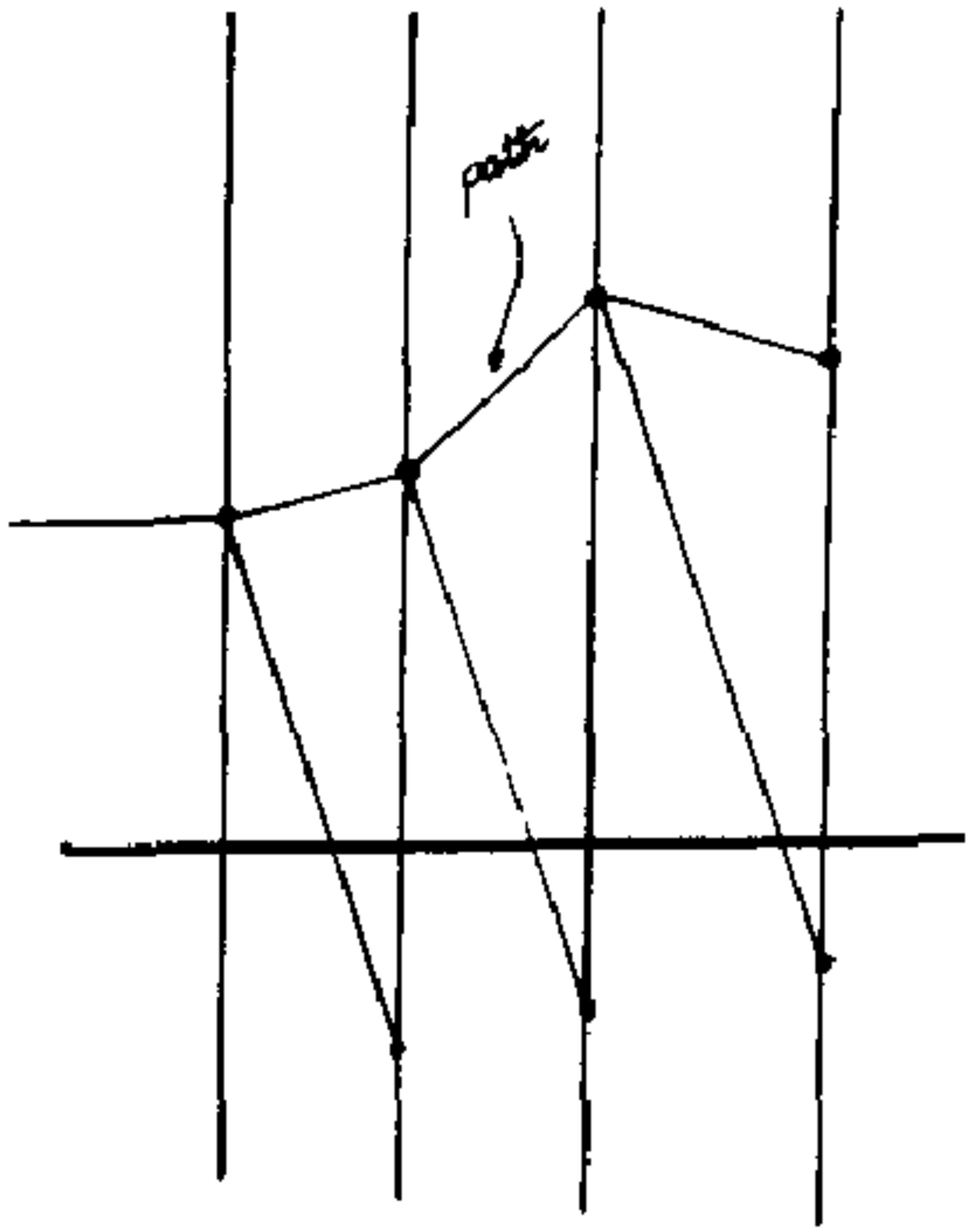
$$L_o = L_e + L_s$$



Path tracing in a nutshell



Ray tracing



Path tracing

…

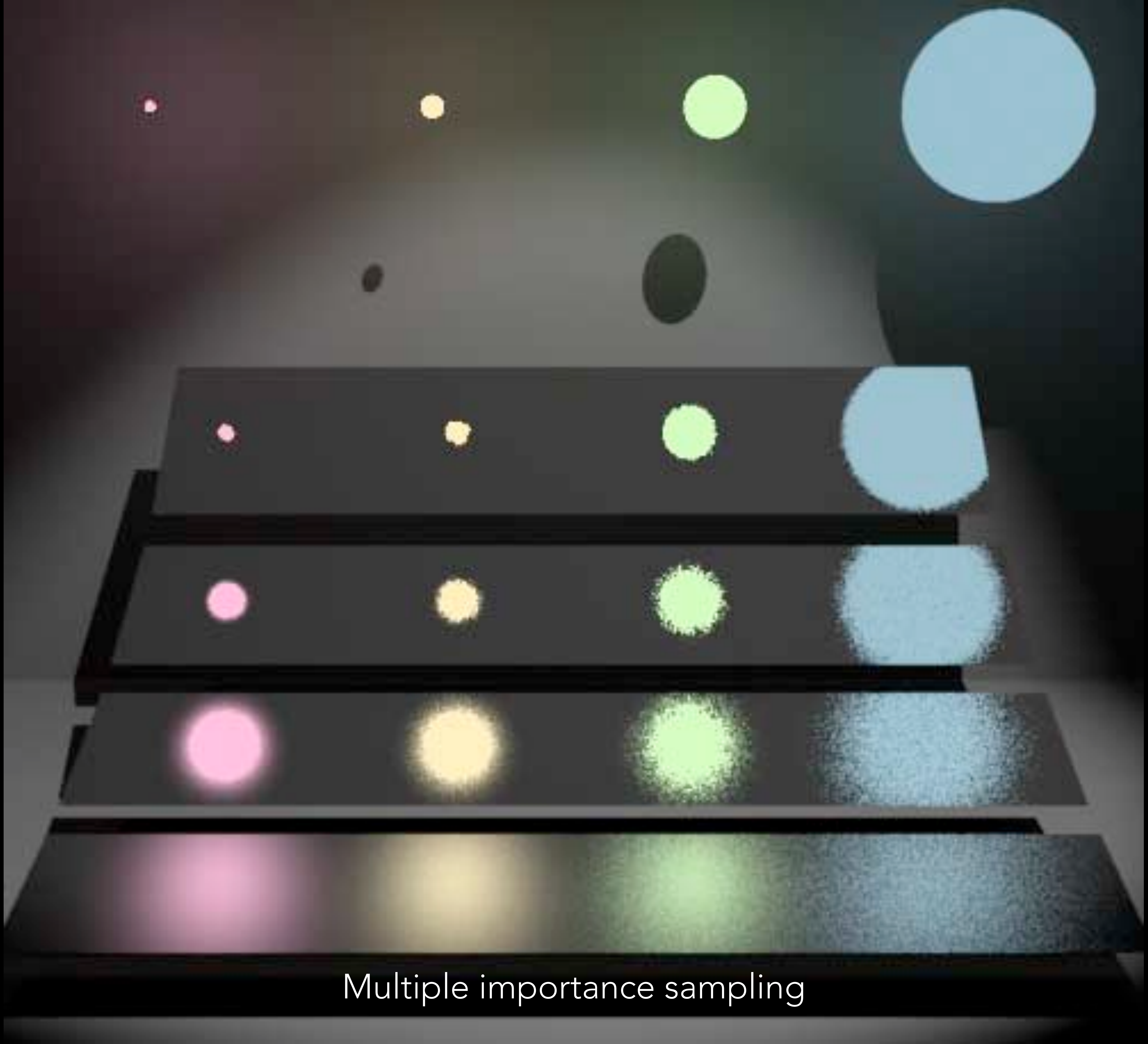
+ Russian roulette

+ importance sampling

From Kajiya's original paper, "The Rendering Equation" (1986)

Limitations and extensions

Point vs. area lights, diffuse vs. glossy surfaces



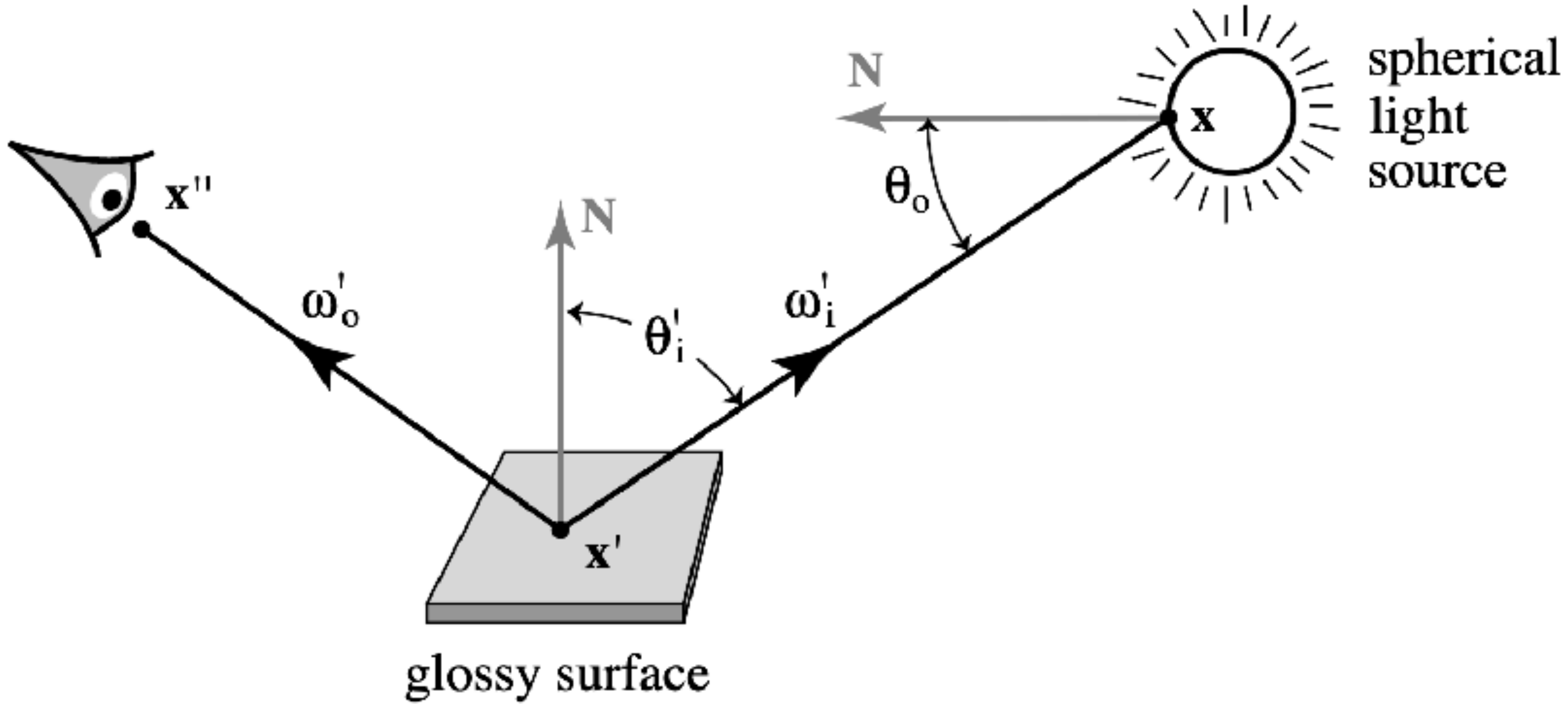
Multiple importance sampling

Veach and Guibas 1995

Multiple importance sampling (Veach and Guibas 1995)

Say I want to integrate $f(x) = f_1(x)\, f_2(x)$
(e.g. f_1 = BRDF, f_2 = incident radiance).

I know probability distributions p_1 and p_2 for importance sampling f_1 and f_2, but not for f.



- Can't just sample according to $p_1(x)\, p_2(x)$ (Why not?)

- Shouldn't just average results of both strategies (too much variance)

Multiple importance sampling: sample from both p_1 and p_2, weight each sample carefully to reduce variance

Caustics

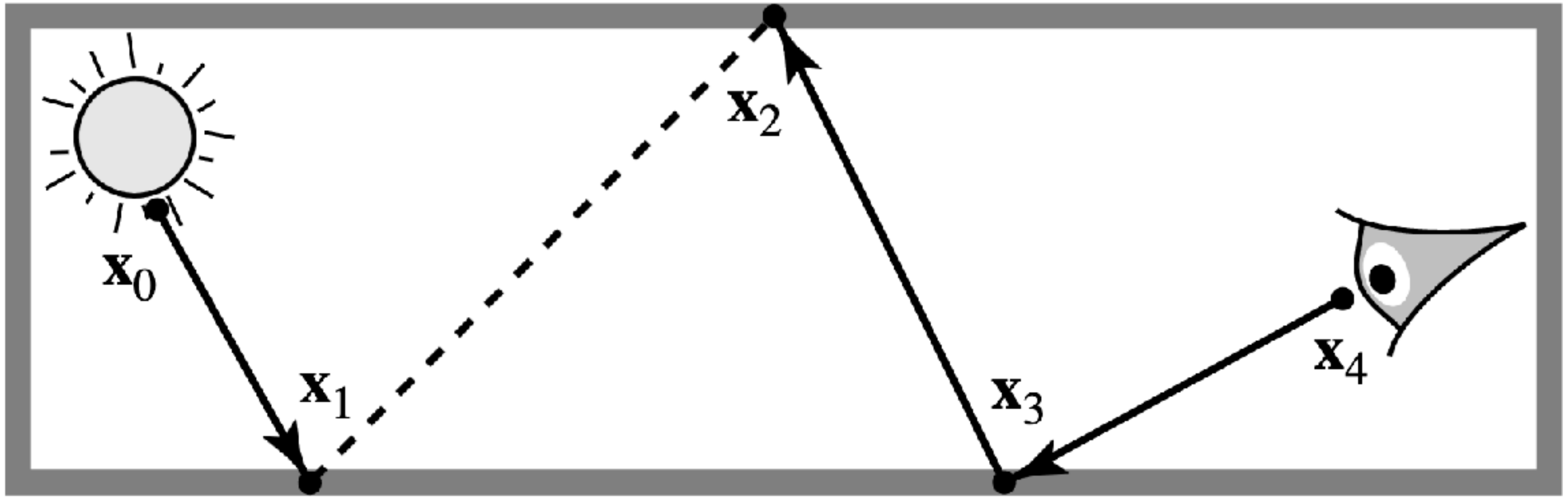
Henrik Wann Jensen

Hard-to-find light paths

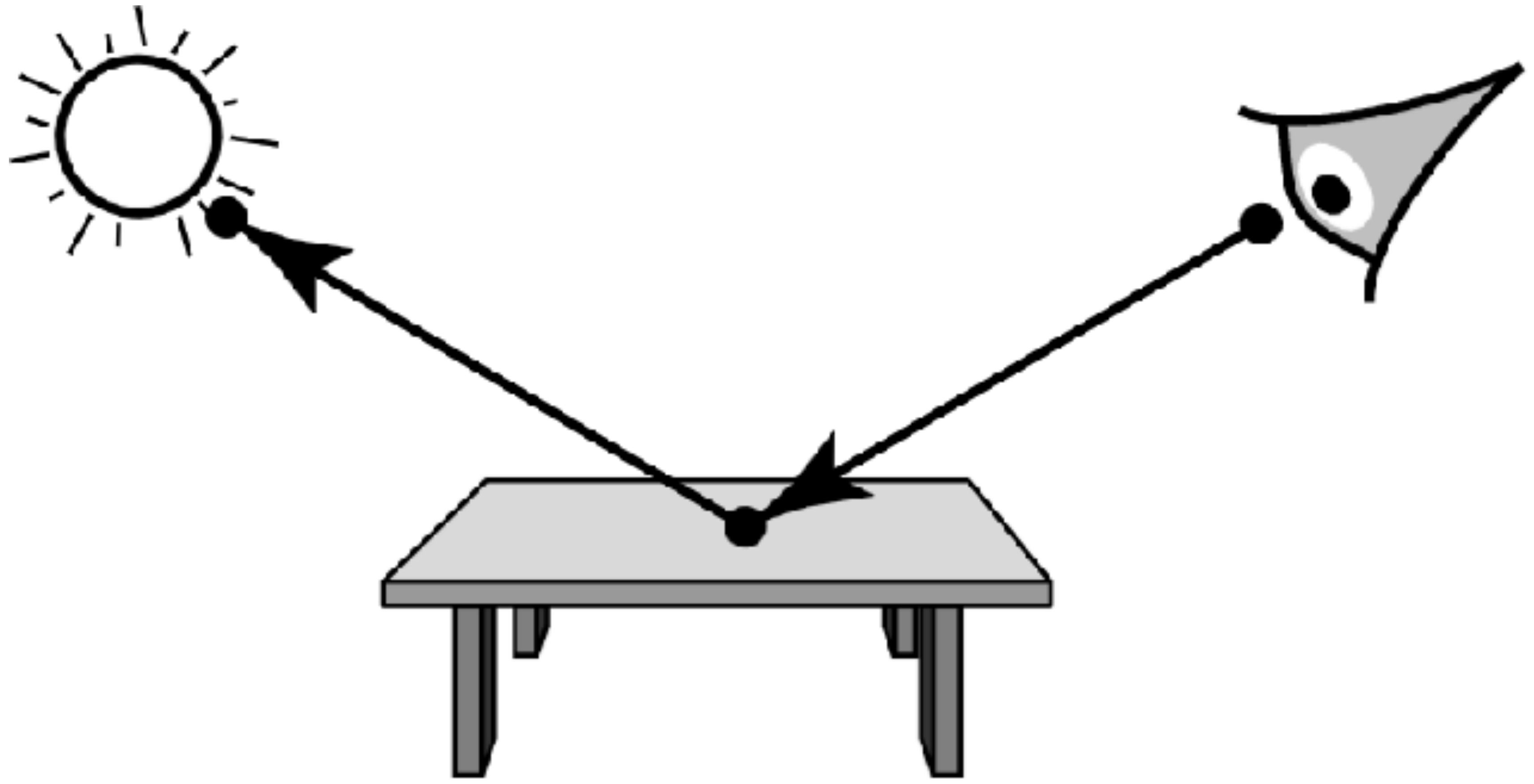
Veach and Guibas 1995

Bidirectional path tracing (Lafortune & Willems 1993, Veach & Guibas 1994)

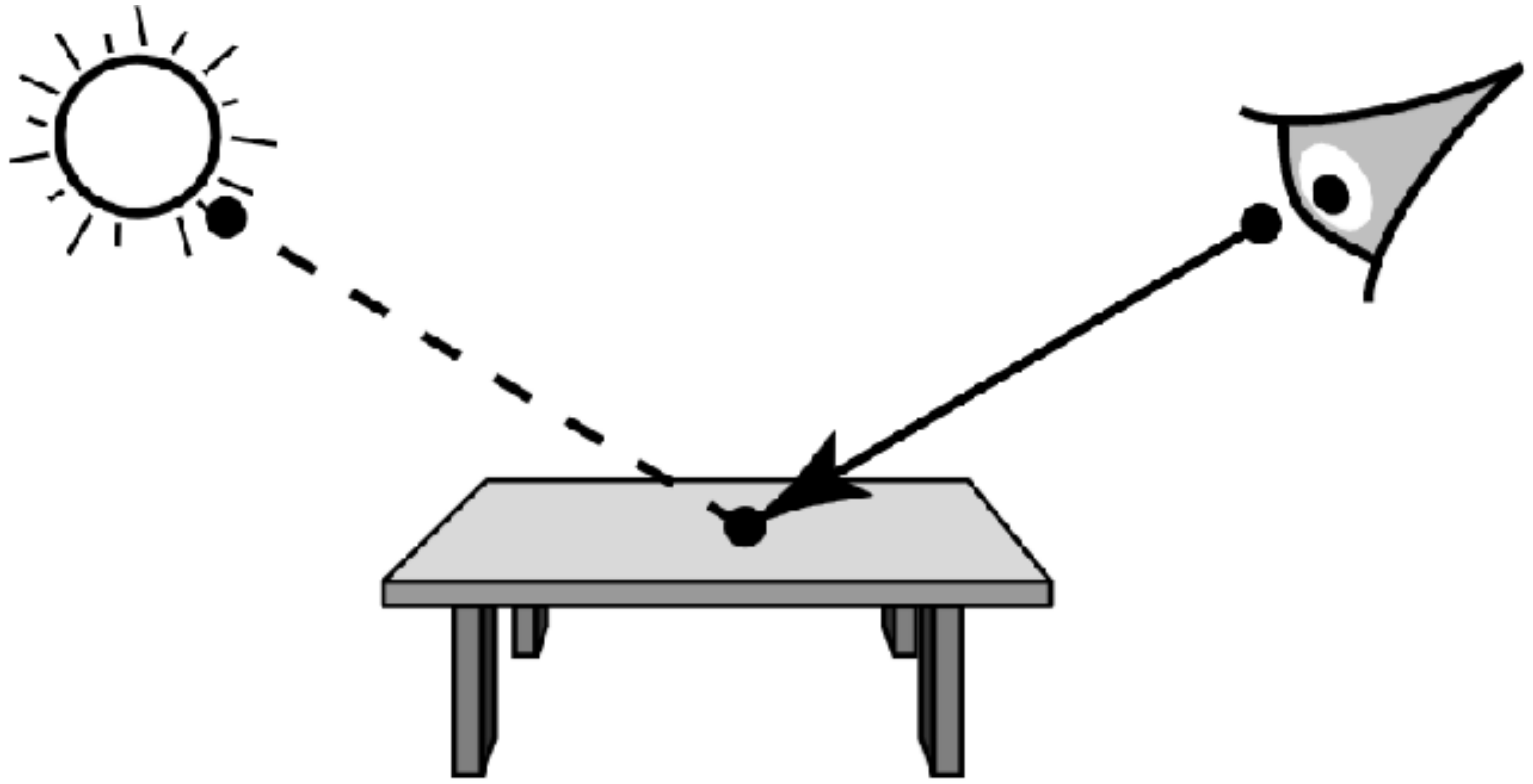
Trace subpaths from the light source and from the eye, then join them together



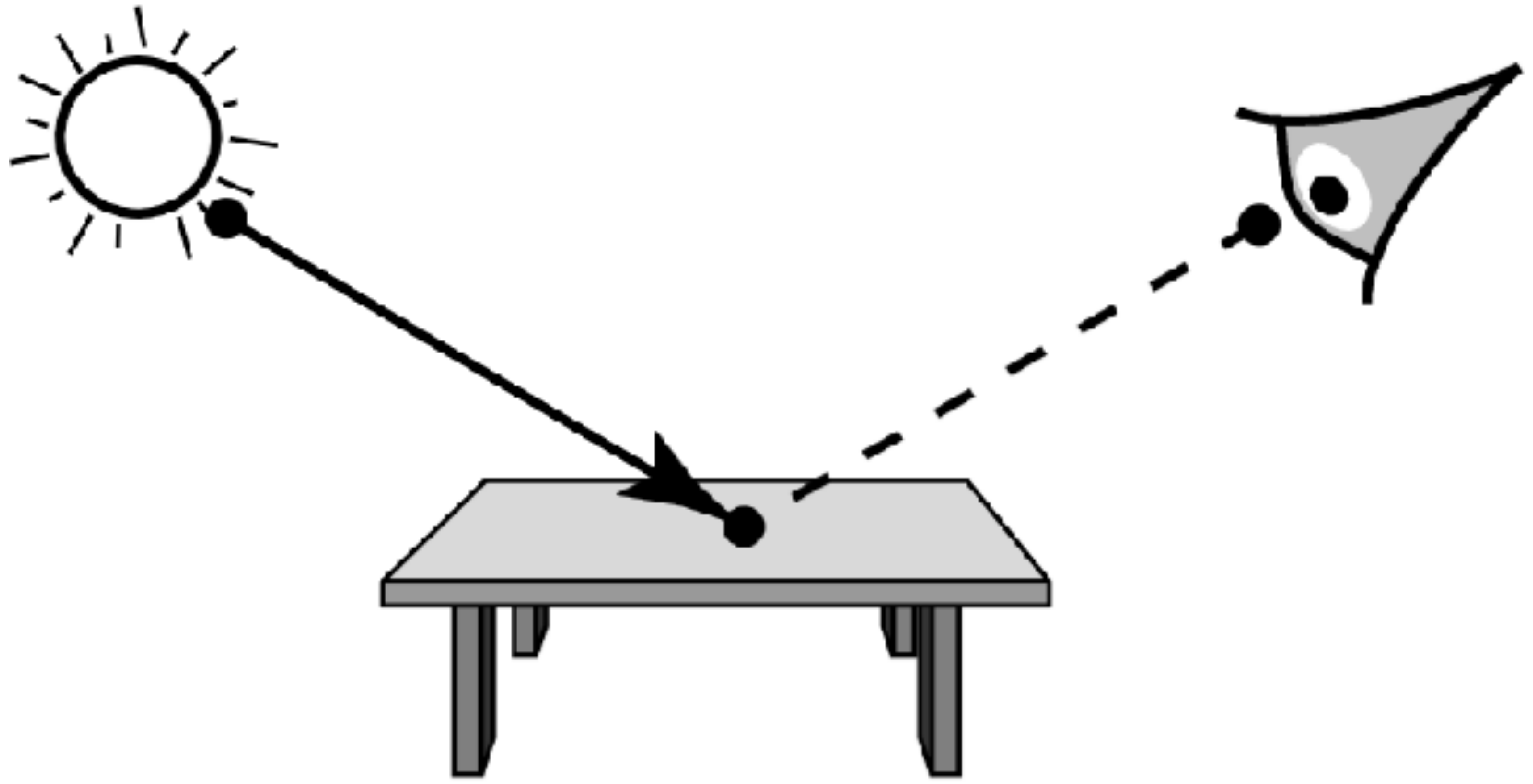
Can be much more efficient if bounces near light source are harder to sample
(e.g. light source is only seen through a glass surface)



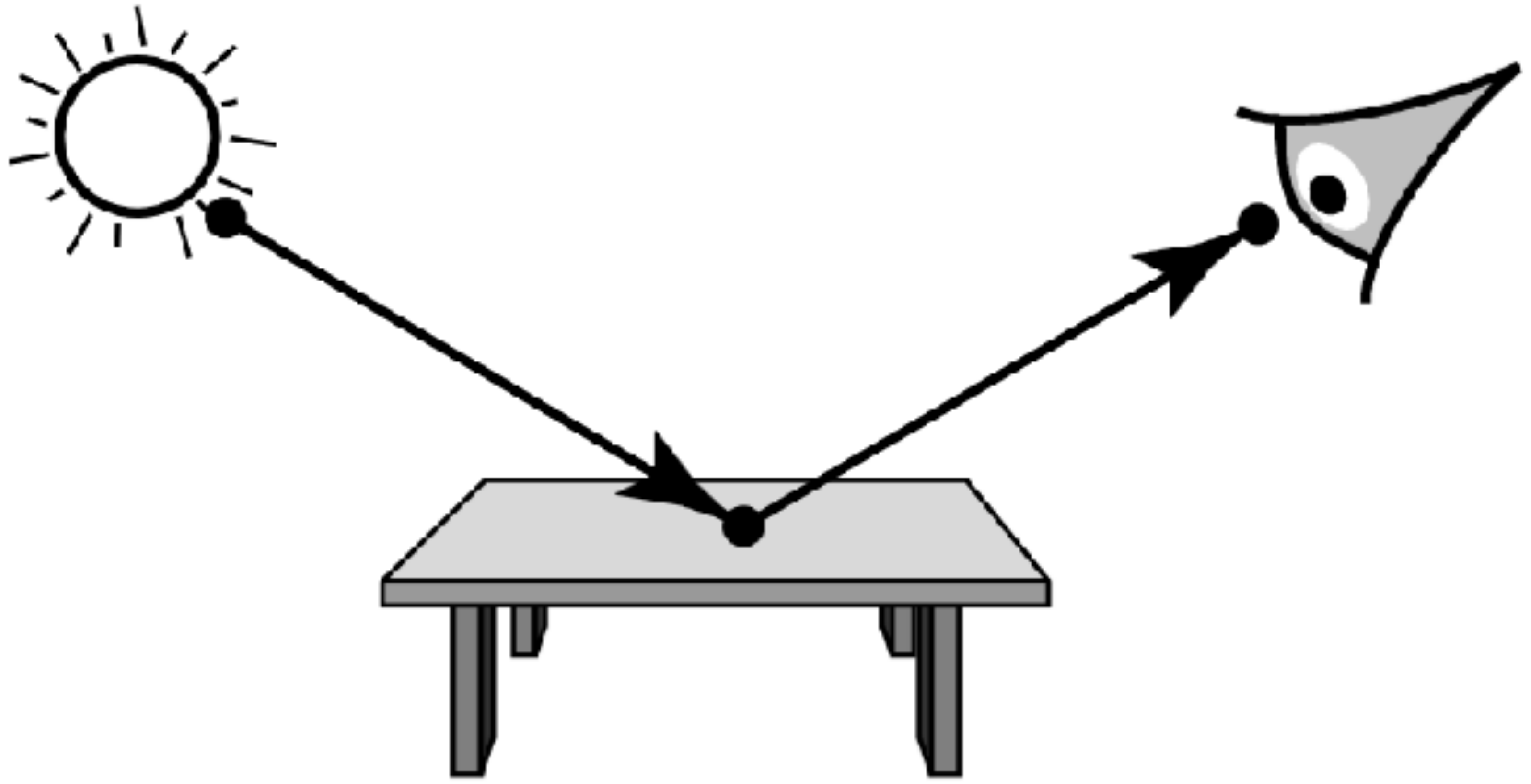
(a) $s = 0, t = 3$



(b) $s = 1, t = 2$



(c) $s = 2, t = 1$



(d) $s = 3, t = 0$

Path tracing
56 spp

Bidirectional path tracing
25 spp (equal computation time)

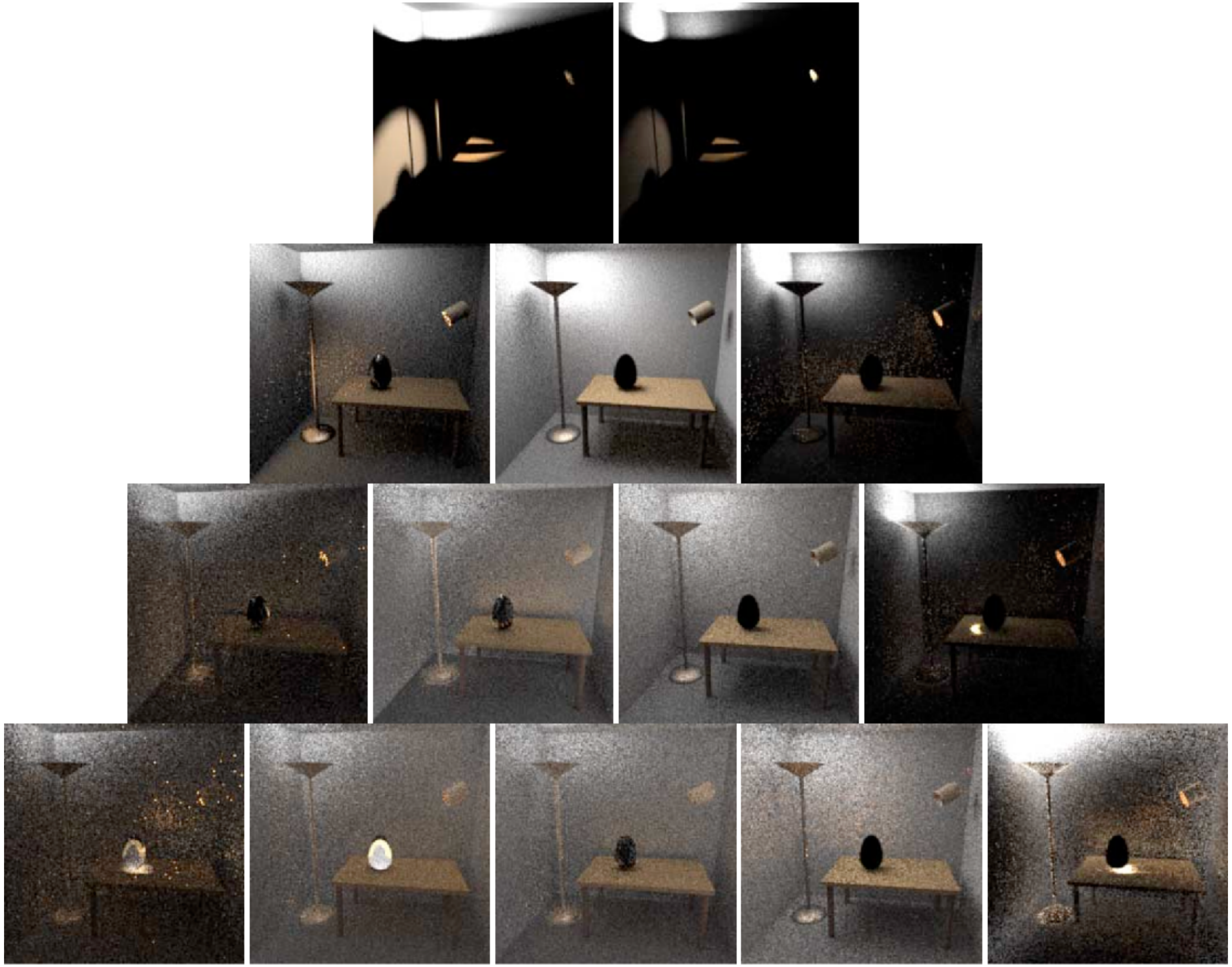
Veach and Guibas 1995

1-bounce paths

2-bounce paths

3-bounce paths

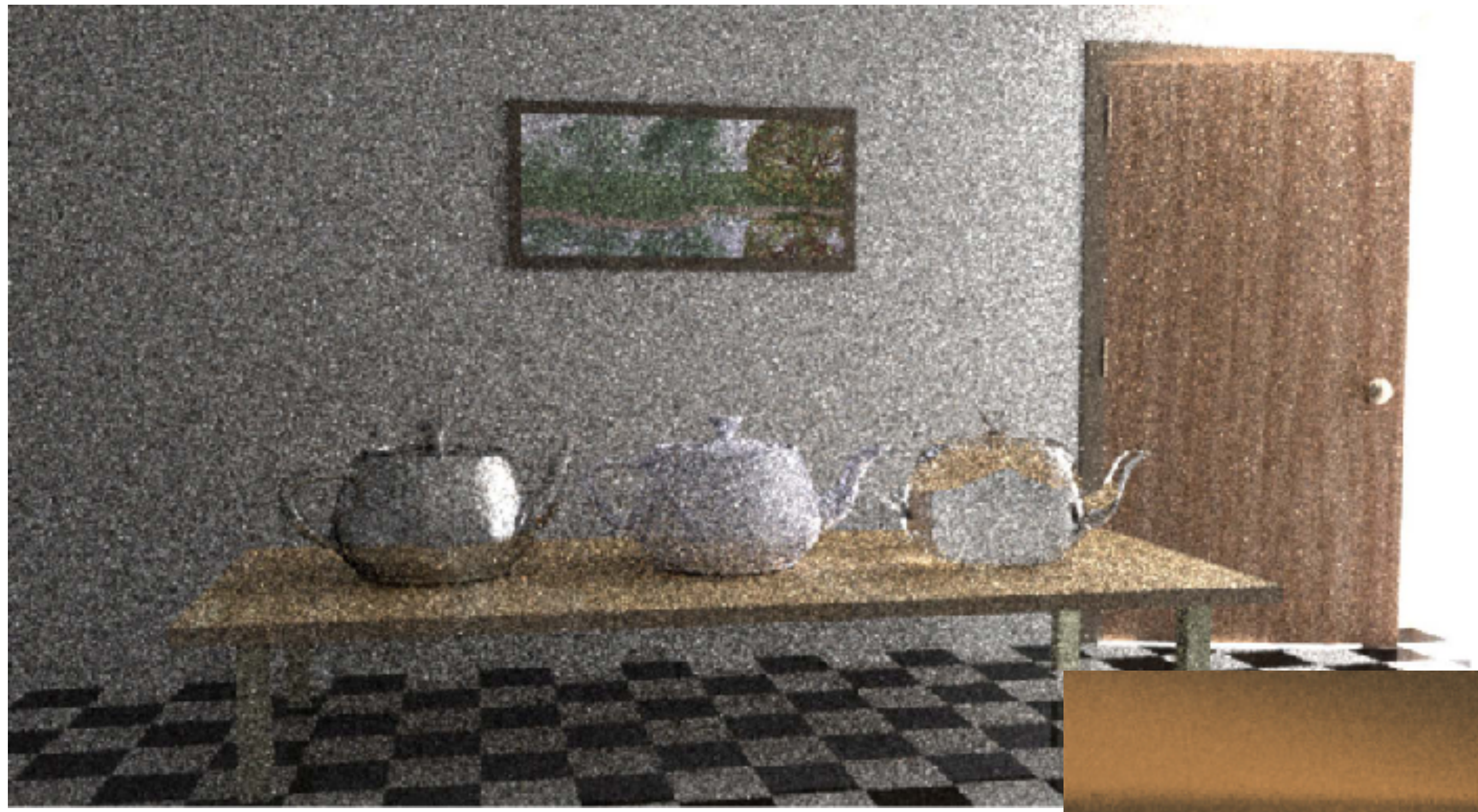
4-bounce paths

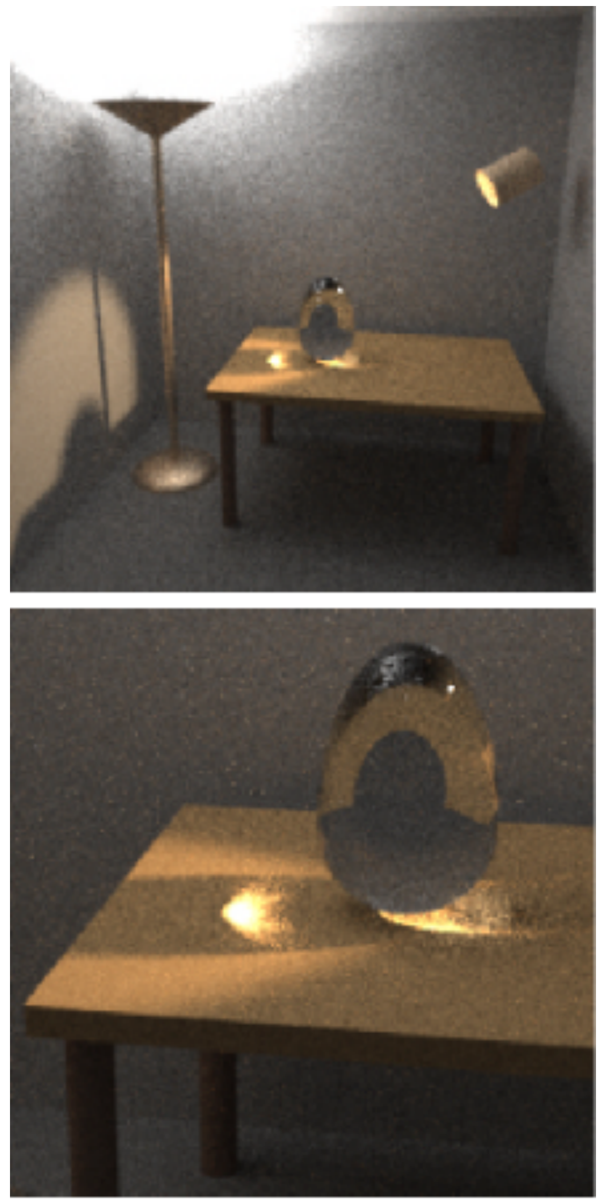


from eye only

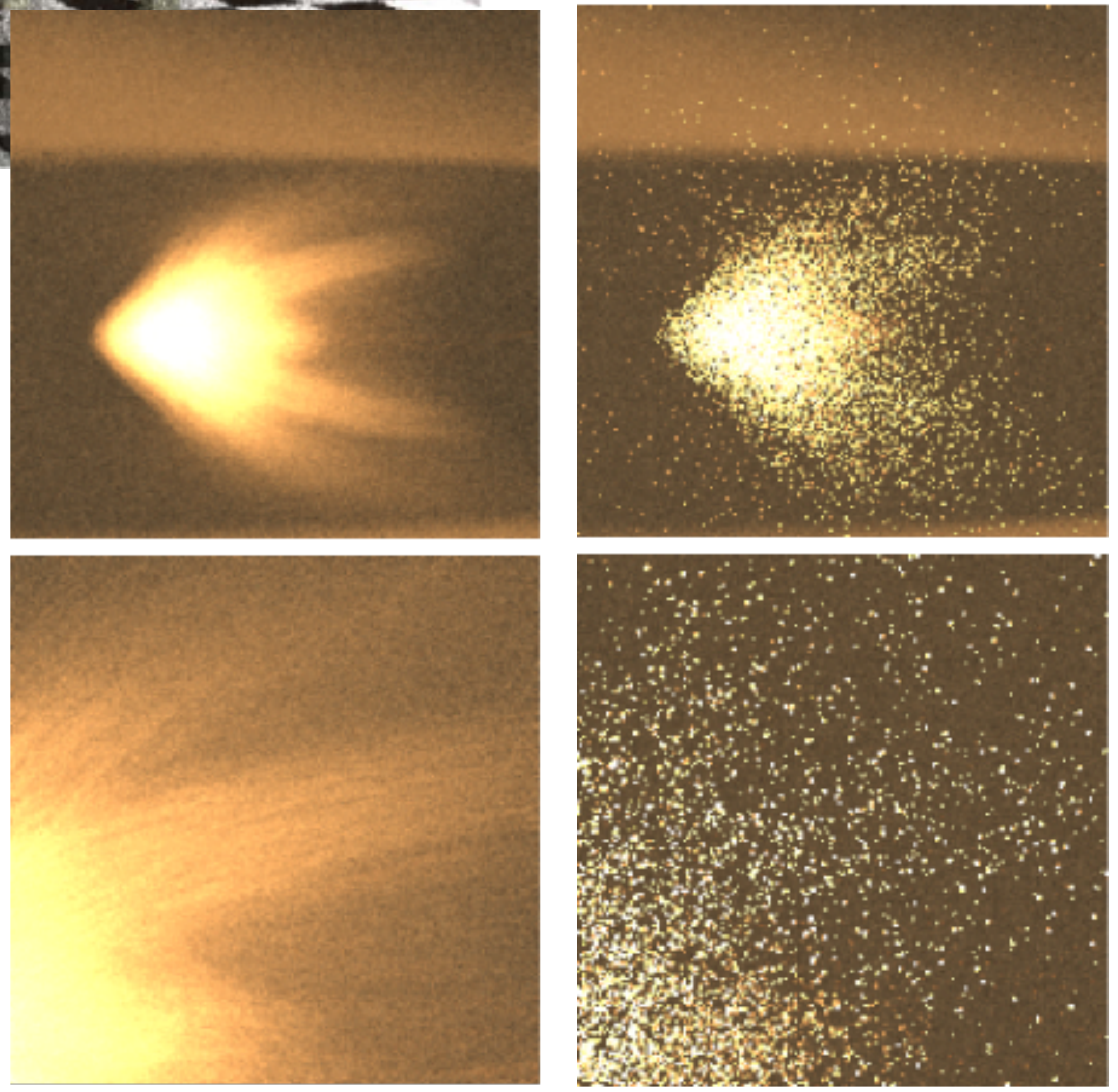
from light only

Metropolis light transport (Veach & Guibas 1997)





In some scenes, only a small fraction of possible paths contribute significantly to the final image



Standard Monte Carlo: pick each sample independently

Metropolis-Hastings sampling algorithm: perturb previous sample to find a nearby high-value sample

Metropolis light transport: MH applied to light paths!


Bidirectional path tracing
40 samples per pixel
Veach and Guibas 1997

Metropolis light transport
~250 samples per pixel

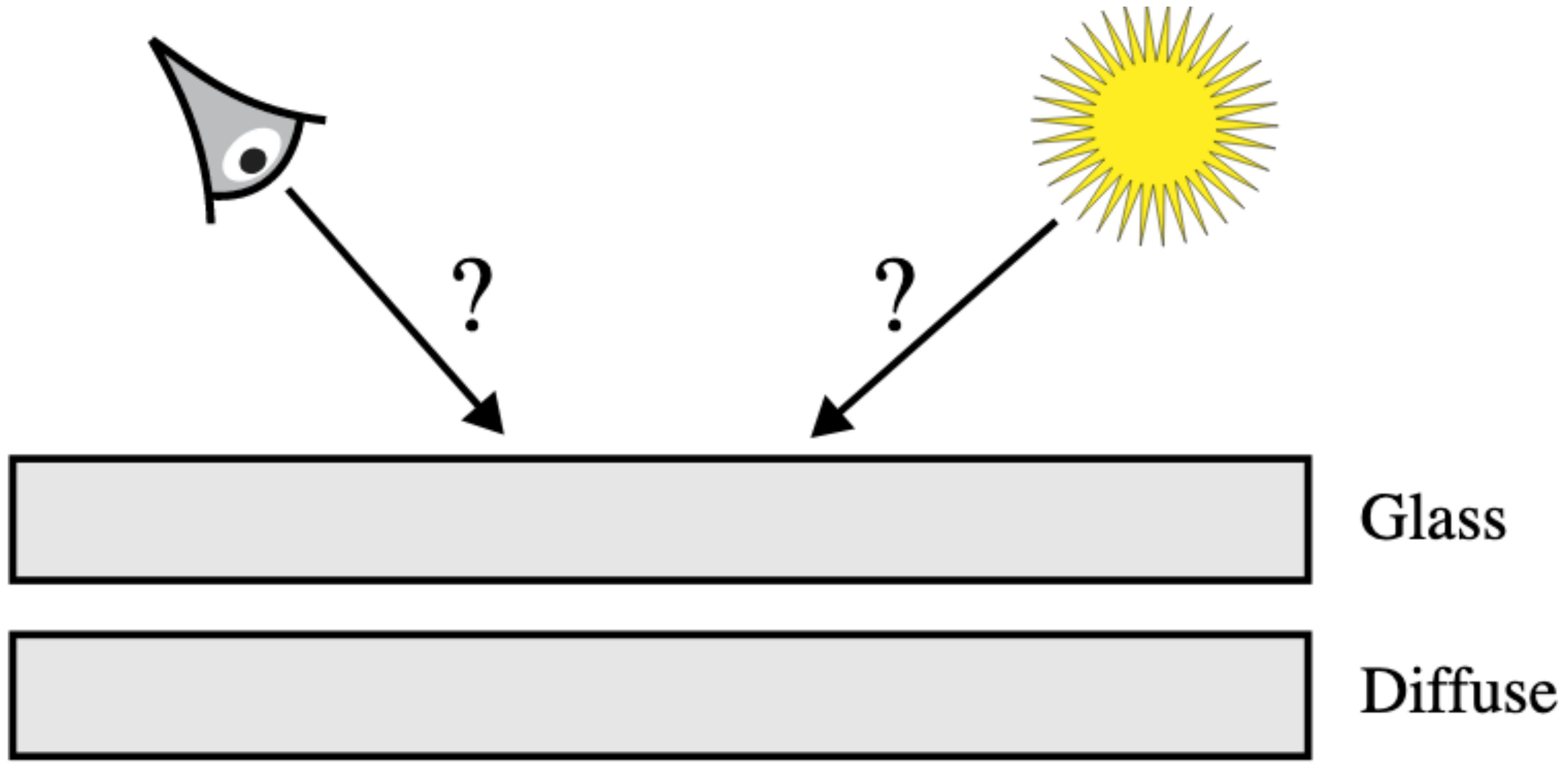
Veach and Guibas 1997

Light path notation

Light *L*, diffuse *D*, specular *S*, eye *E*

In general, we want to sample all paths *L*(*D*|*S*)**E*

- Direct illumination: *LDE*
- Ray-traced reflections: LDS^+E
- Diffuse indirect light: LD^+DE
- Caustics: LS^+DE



LS^+DS^+E

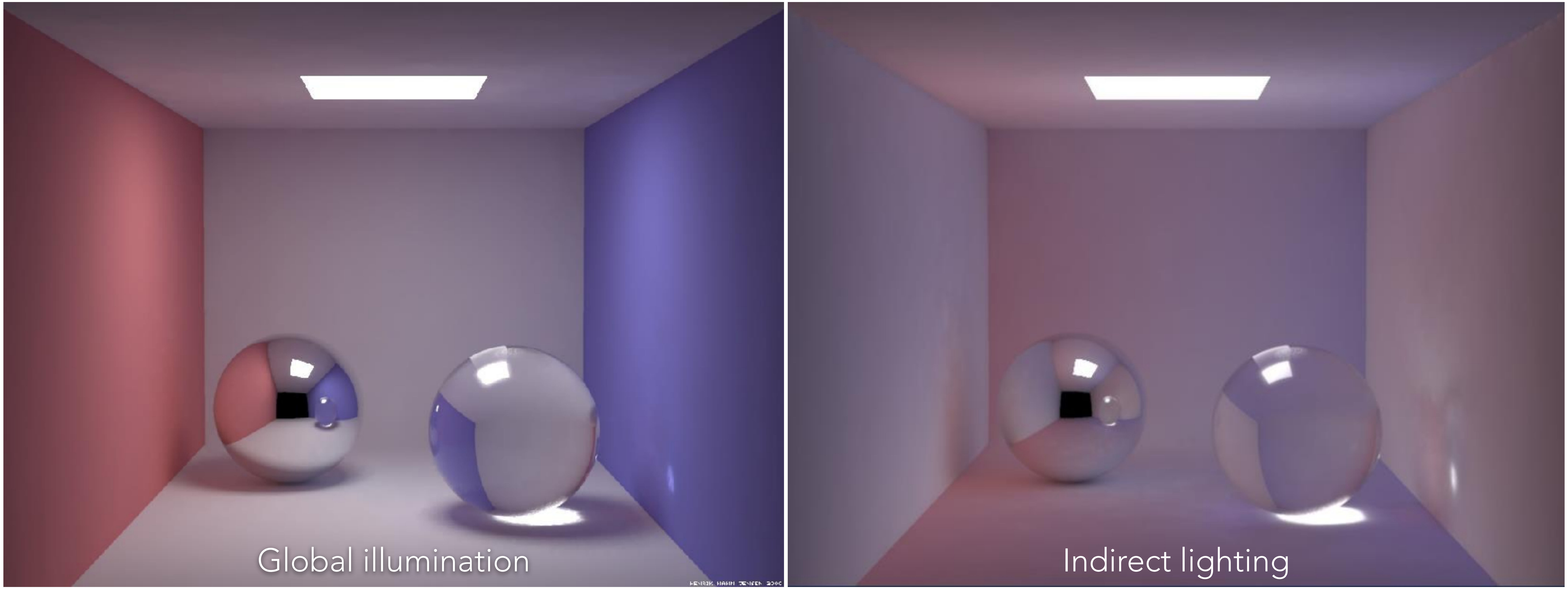
Introduced by Paul Heckbert in 1990.

What paths are still hard to sample?

Path tracing BDPT MPT Photon mapping PPM

Path tracing methods consider only one path at a time. Each path only affects one pixel.

But indirect lighting is mostly smooth! Why not store it and reuse to light other points?



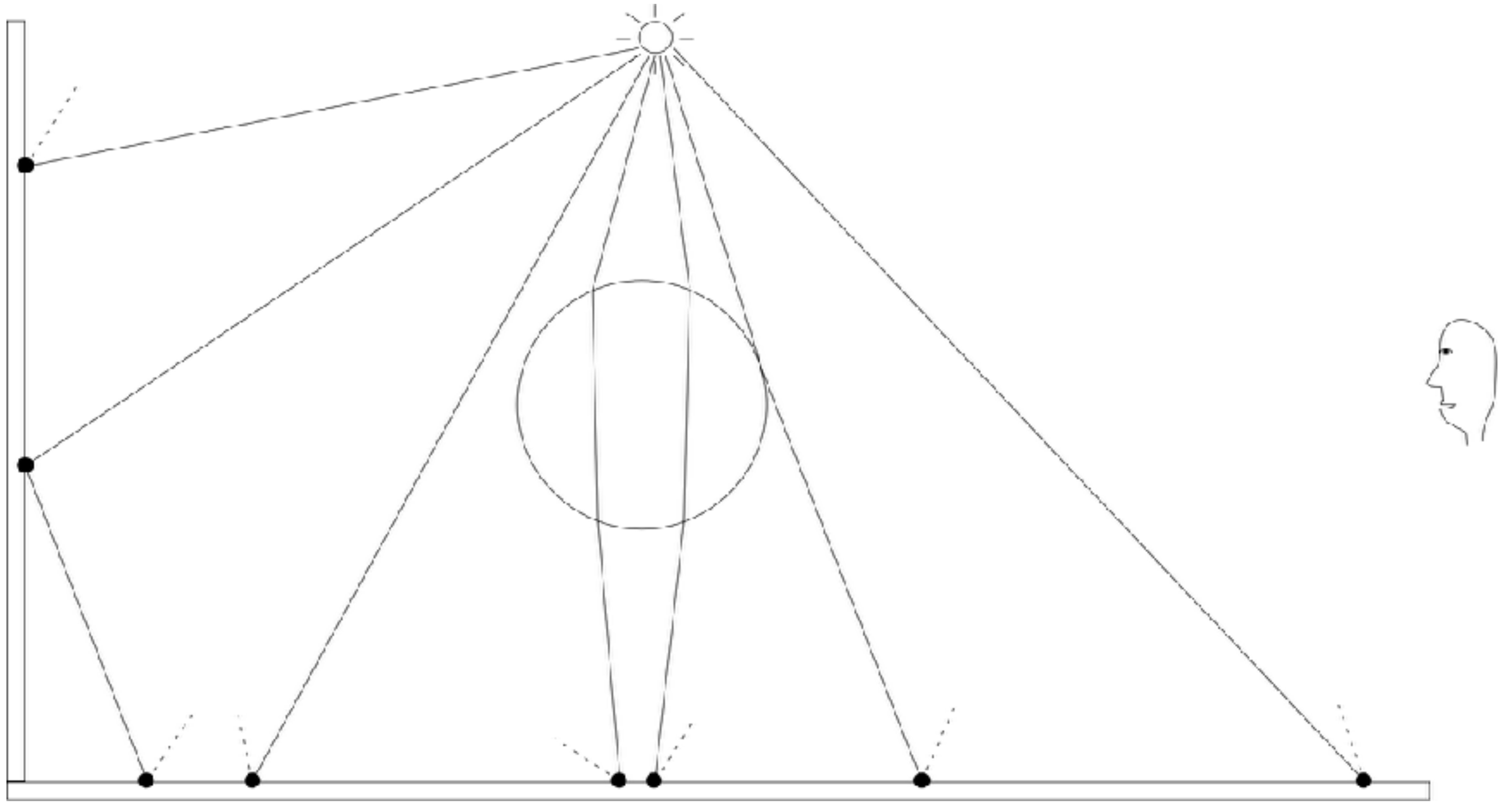
Henrik Wann Jensen

Photon mapping

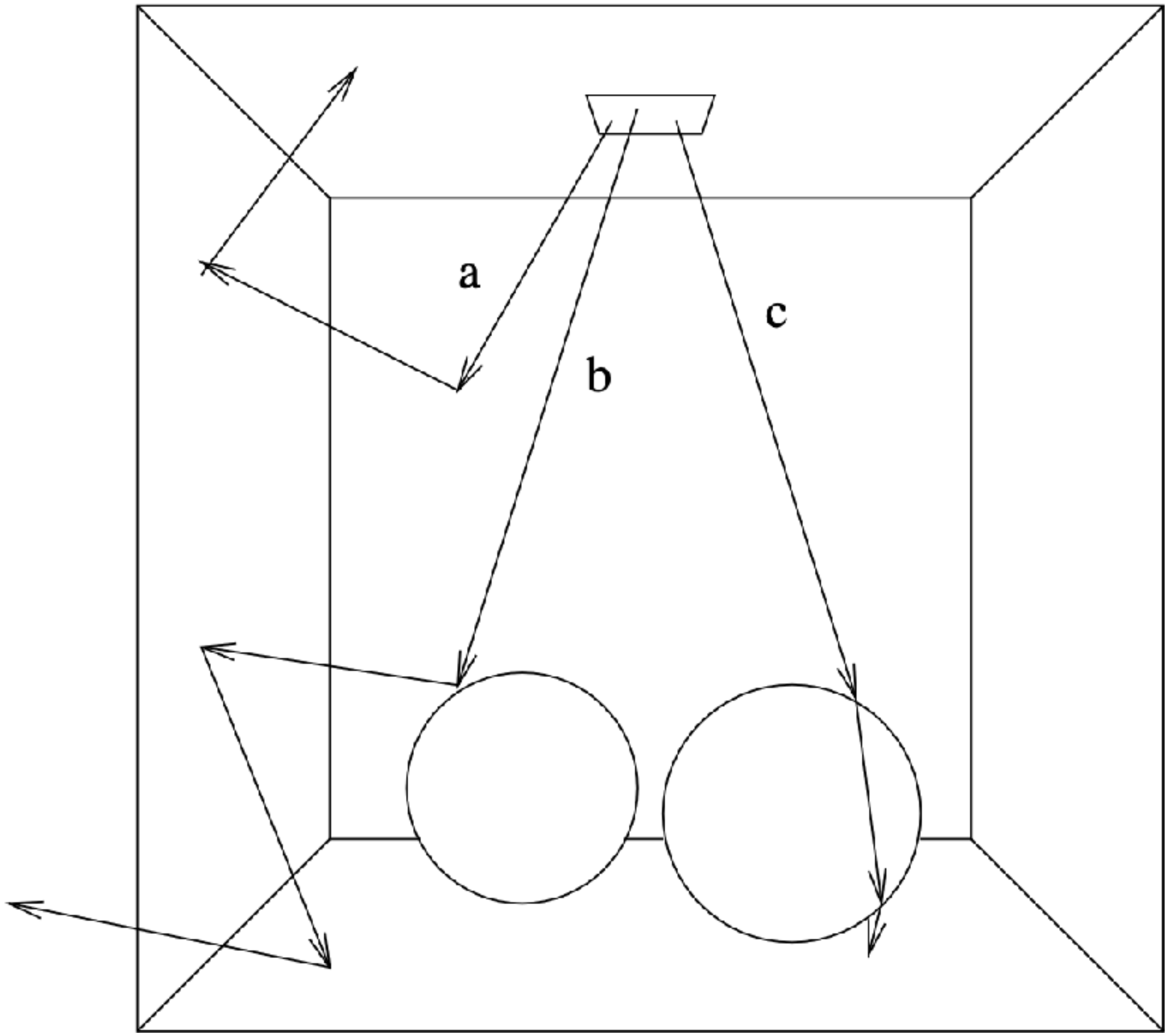
Phase 1:

Trace packets of light energy ("photons") from the light source and bounce them around the scene

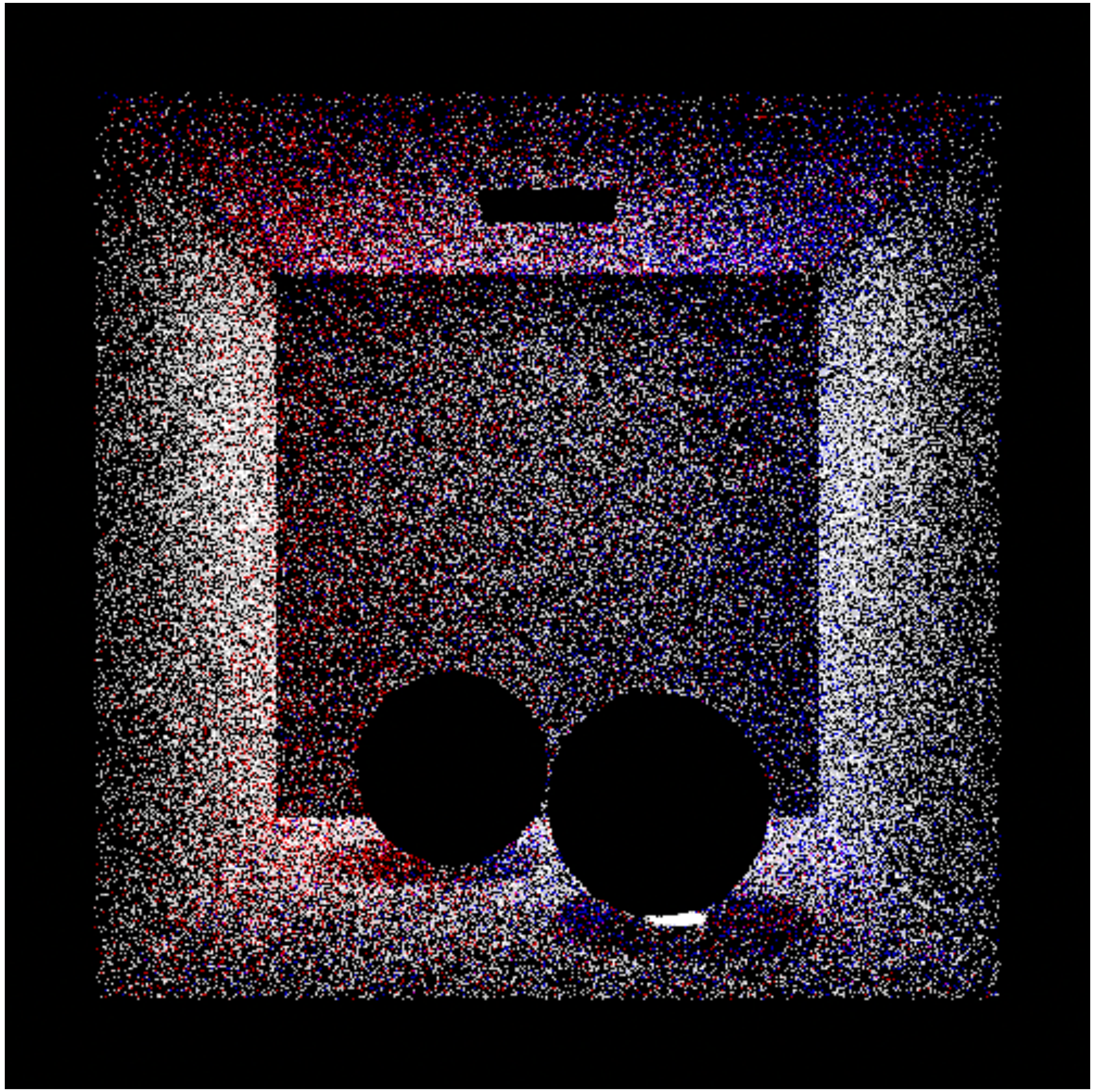
At each surface hit, store the position, incident direction, light power



Henrik Wann Jensen



Henrik Wann Jensen



Henrik Wann Jensen

Photon mapping

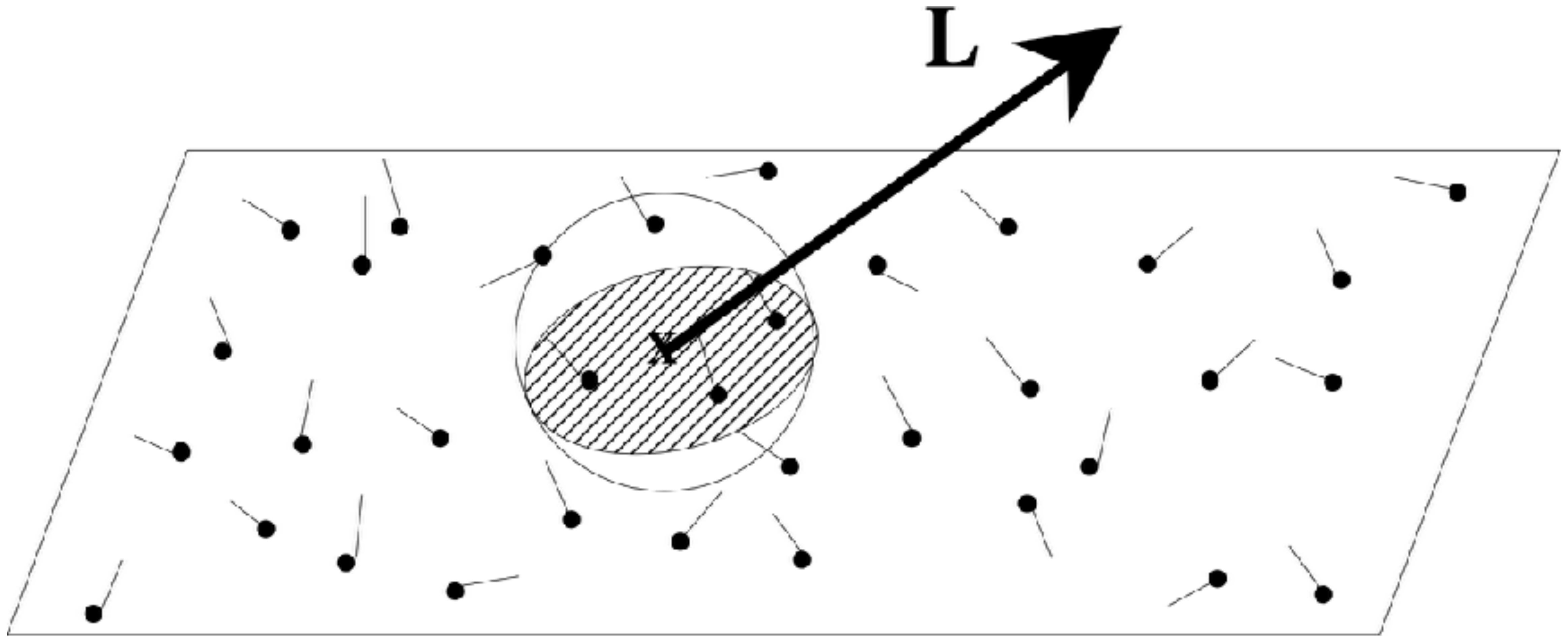
Phase 2:

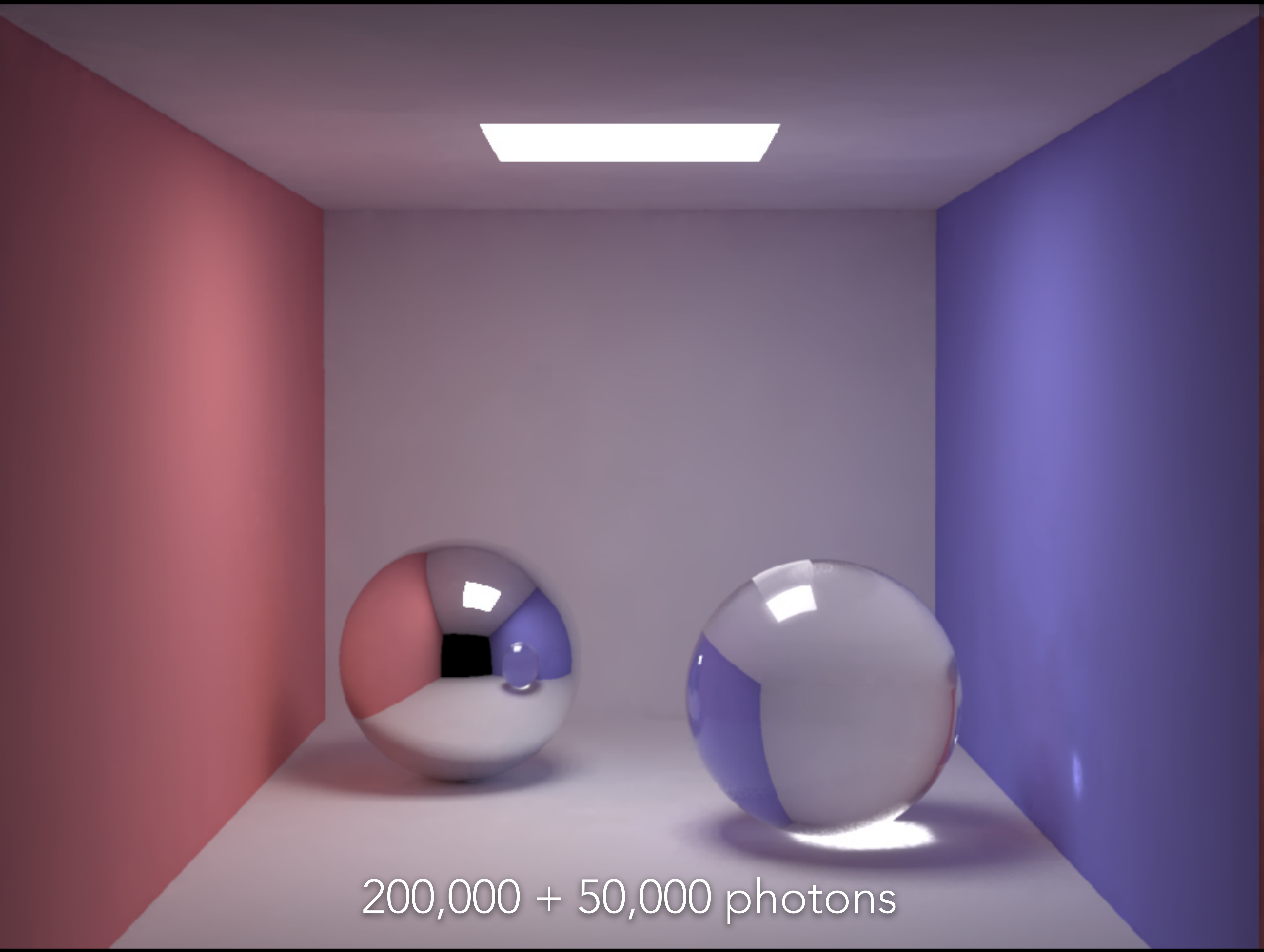
At diffuse surface, estimate incident illumination as weighted sum of nearest *N* photons

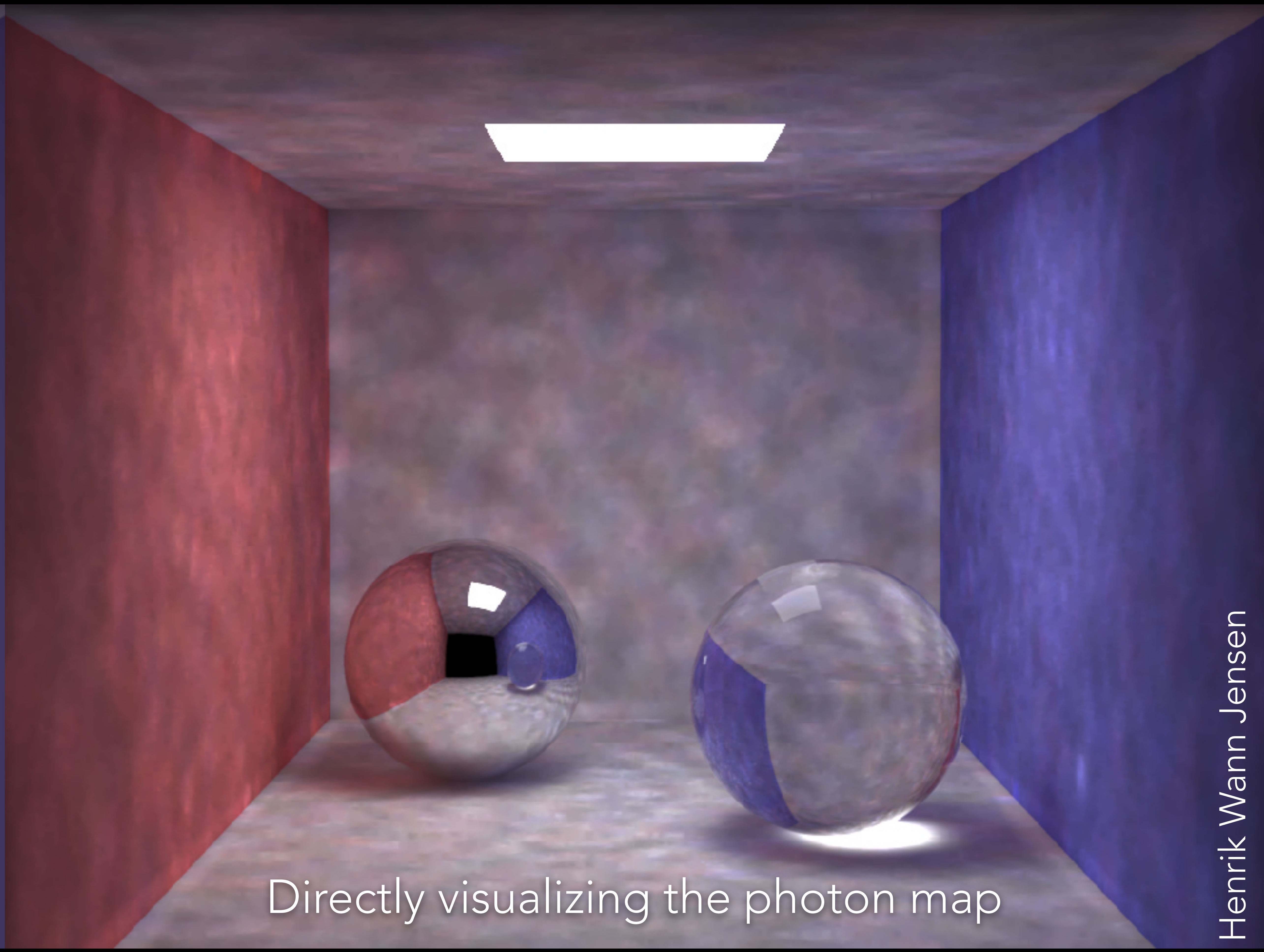
At specular surface, keep ray tracing until you hit a diffuse surface. (Can be useful to do this for diffuse surfaces as well: “final gathering”)

Other optimizations:

- Compute direct illumination directly
- Extra photon map specifically for caustics







Henrik Wann Jensen


Increasing number of photons
Physically Based Rendering

Photon mapping is biased (for any fixed #photons, caustics are blurry)

…but it is consistent (as #photons → ∞, converges to correct image).

Requires extra memory to store all the photons!

- Solved in more recent work: progressive photon mapping (PPM), stochastic PPM

Lots of other algorithms we're not covering…

- Virtual point lights, lightcuts
- Unified path space / vertex connection and merging
- …