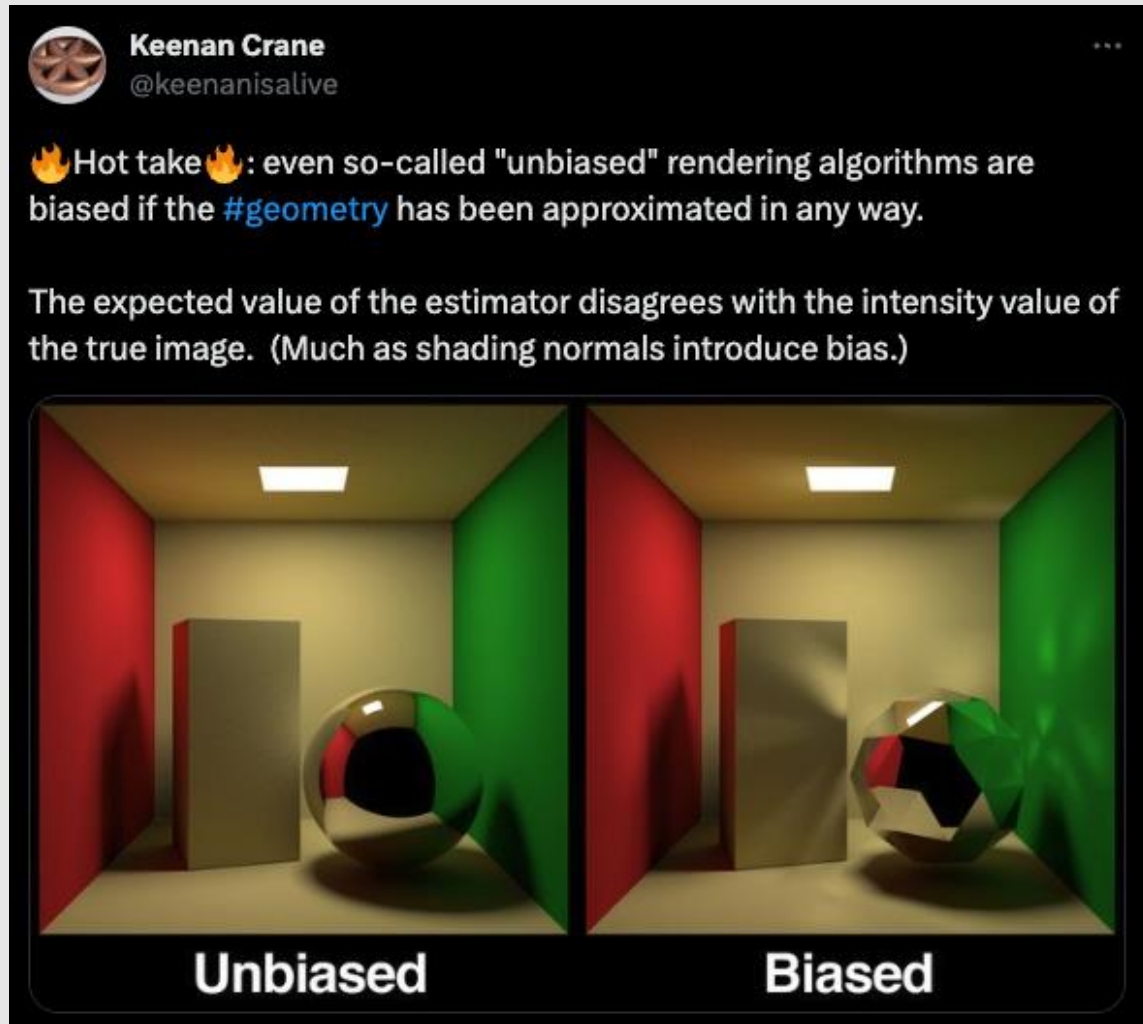


More Variance Reduction

- ~~Monte Carlo Sampling~~
- **Biased vs Unbiased Estimators**
- ~~Physically-Based Rendering Methods~~

Biased vs. Unbiased Renderer



- An **unbiased** renderer tries to mimic the uniformity of real life
 - Does not introduce systematic bias
 - Taking more samples will reduce error
 - Approaches ground truth with infinite sampling
- A **biased** renderer will take shortcuts to make renders look better
 - Taking more samples may introduce even more signal than the original image
 - Usually faster rendering/less samples
 - Can seek out more difficult paths
- When comparing render methods, makes more sense to compare **unbiased methods**

The Monte Carlo Estimator

- Named **Monte Carlo** after the famous gambling location in Monaco
 - Shares the same random characteristic as a roulette game
- **Algorithm:**
 - Sample a direction based on the PDF $p(\omega_j)$
 - Compute the incident radiance of the direction
 - Divide by the PDF $p(\omega_j)$ to make unbiased
 - Repeat, averaging the samples together

$$\frac{1}{N} \sum_{j=1}^N \frac{f_r(\mathbf{p}, \omega_j \rightarrow \omega_r) L_i(\mathbf{p}, \omega_j) \cos \theta_j}{p(\omega_j)}$$

Note! We no longer multiply our average by the size of the domain. Dividing by the PDF takes care of that for us. Why? Because the PDF must integrate to 1 over the entire domain.

Variance

- **Variance** is how far we are from the average, on average

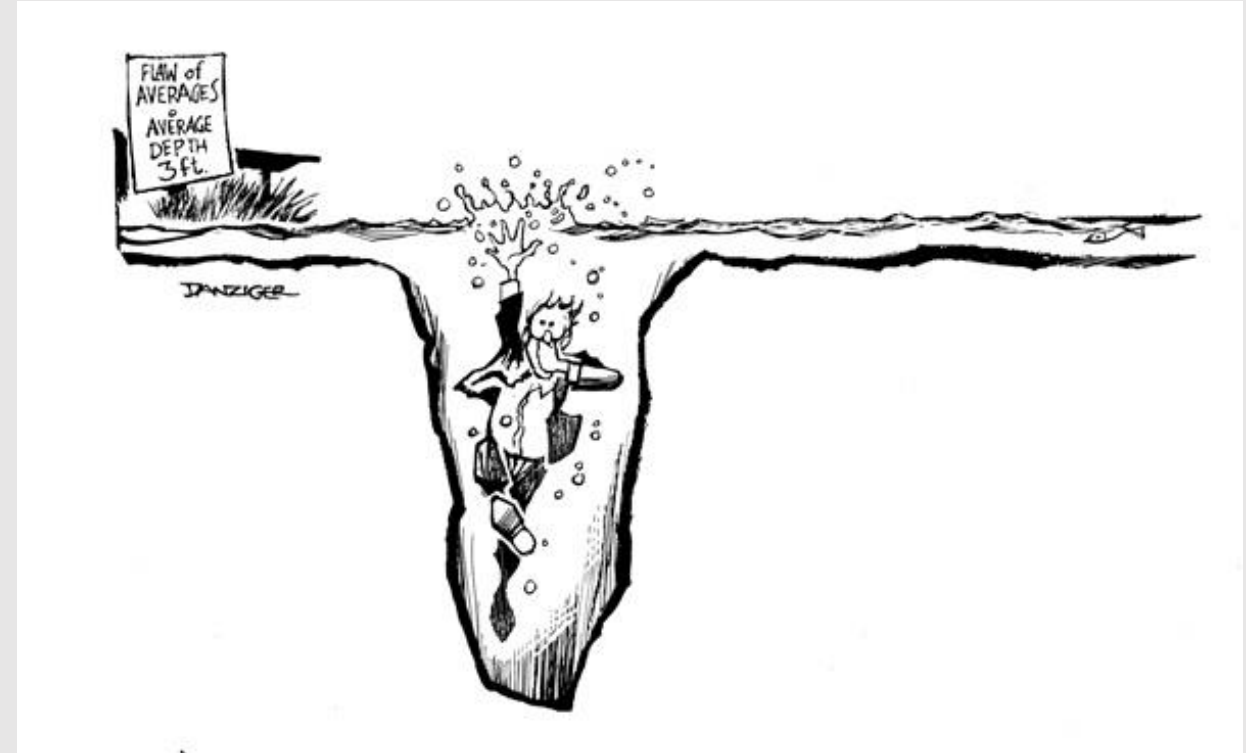
$$\text{Var}(X) := E[(X - E[X])^2]$$

- Discrete:

$$\sum_{i=1}^n p_i (x_i - \sum_j p_j x_j)^2$$

- Continuous:

$$\int_{\Omega} p(x) (x - \int_{\Omega} yp(y) dy)^2 dx$$



Bias & Consistency

- An estimator is **consistent** if it converges to the correct answer:

$$\lim_{n \rightarrow \infty} P(|I - \hat{I}_n| > 0) = 0$$

near infinite # of samples

- An estimator is **unbiased** if it is correct on average:

$$E[I - \hat{I}_n] = 0$$

even if just 1 sample

- consistent != unbiased**



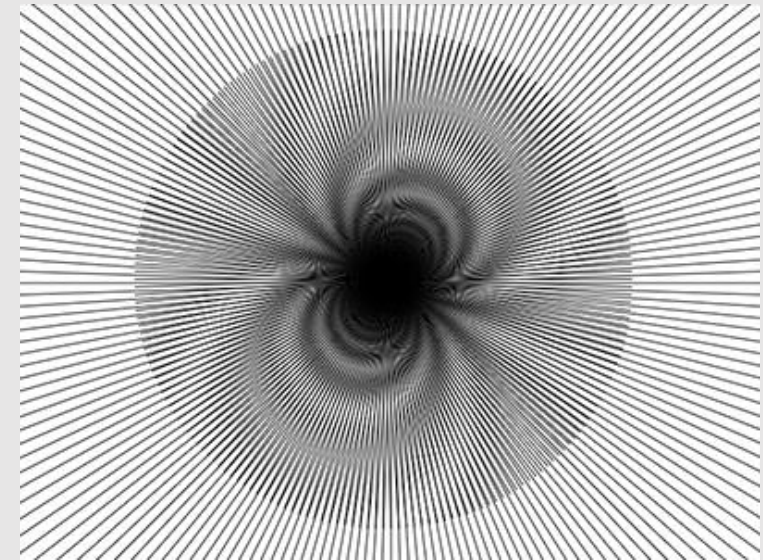
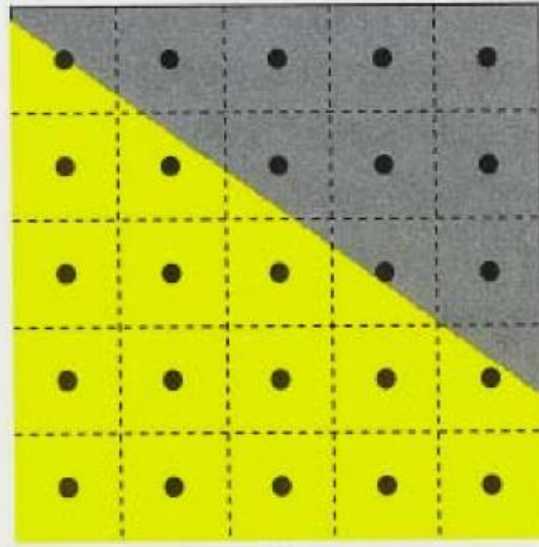
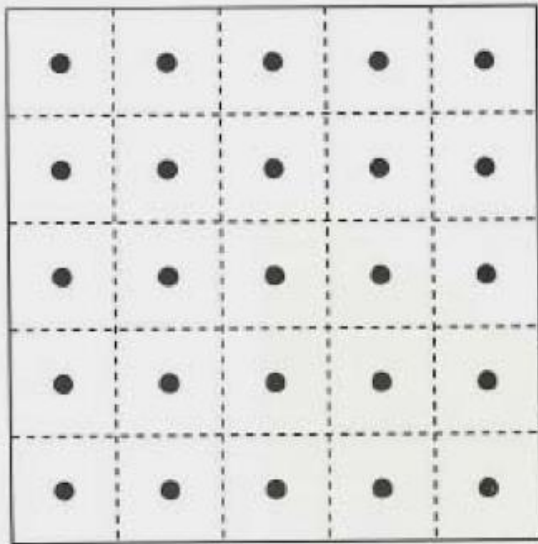
[biased]

[unbiased]

How do we take good samples?

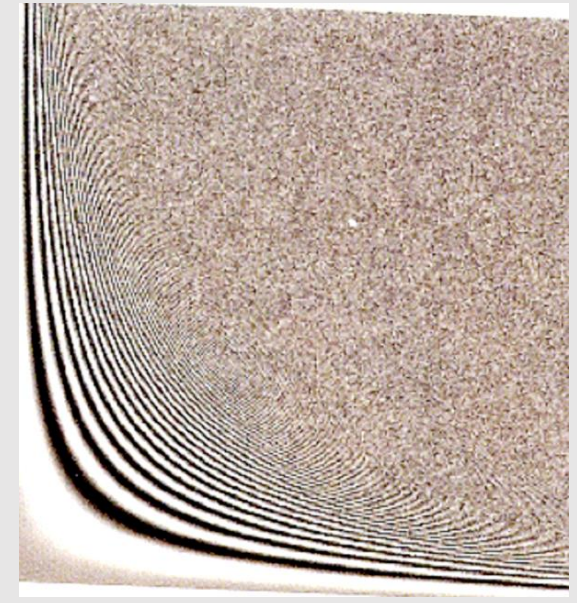
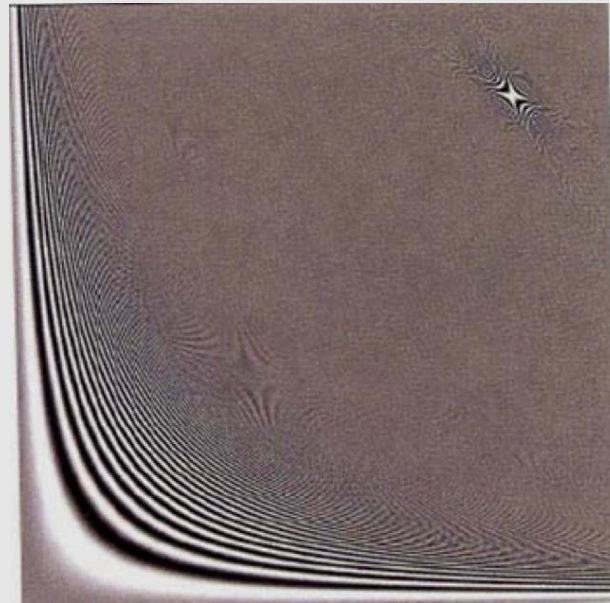
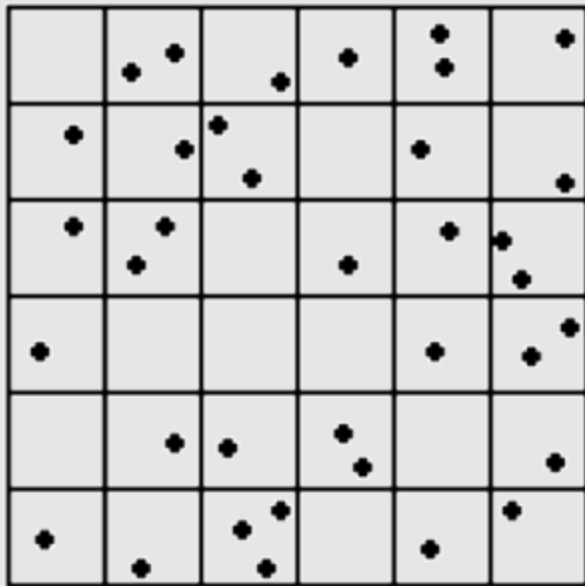
Uniform Sampling

- Place samples uniformly apart in grid fashion
 - [+] Easy to compute
 - [-] We still have jagged edges, just at higher resolutions
 - [-] More samples needed
 - [-] Does not fix moiré pattern



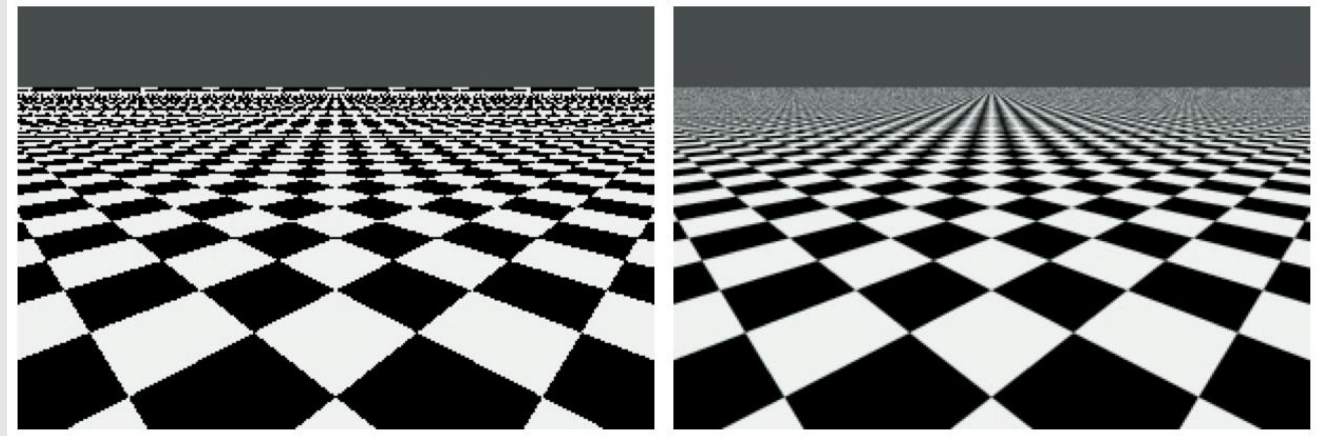
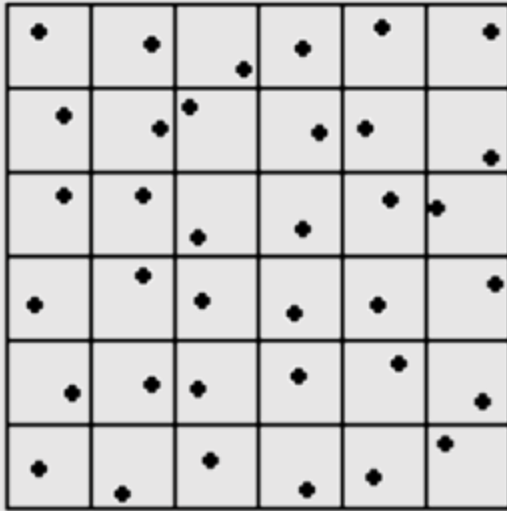
Random Sampling

- Place samples randomly
 - [+] Easy to compute
 - [-] Introduces noise, noticeable at low resolutions
 - [-] Lack of distance between samples



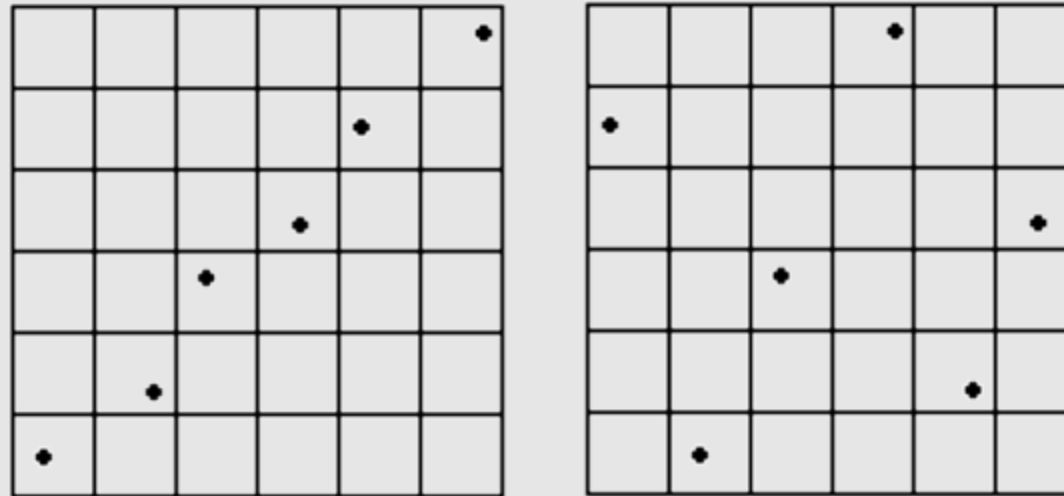
Jittered Sampling

- Divide into $N \times N$ grid, place a sample randomly per grid cell
 - [+] Easy to compute
 - [+] A more constrained version of random sampling
 - [-] Ensures distance between samples, but not enough!



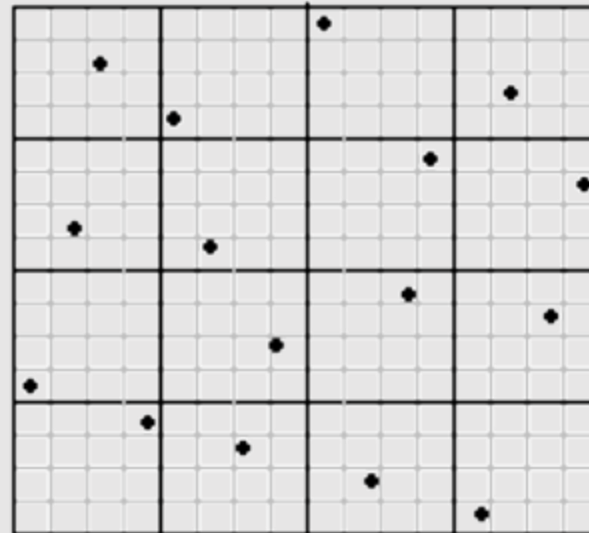
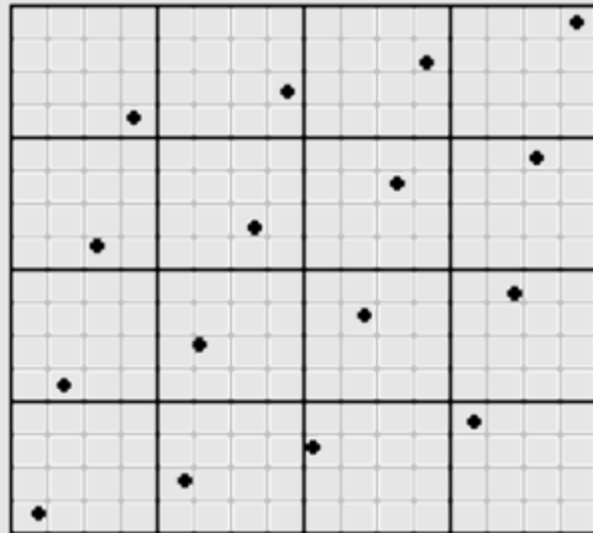
N-Rooks Sampling

- All samples start on the diagonal, randomly shuffle (x, y) coordinates until rooks condition satisfied (no 2 samples lie on the same column or row)
 - [+] Provides good sample sparsity
 - [-] Expensive to compute
 - [-] Possibility of not terminating



Multi-Jittered Sampling

- Jittering + n-rook sampling
 - [+] Provides good sample sparsity
 - [+] Easier to satisfy rook condition
 - [-] Expensive to compute



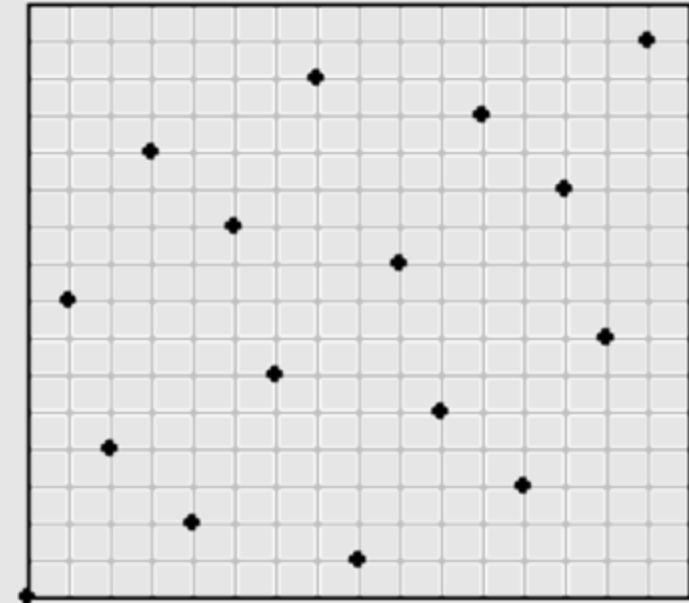
Hammersley Sampling

- Sample according to a fixed, well formed distribution
 - [+] Can pre-compute results
 - [+] Evenly distributed in 2D space
 - [-] No randomness in results

$$\Phi_2(i) \in [0, 1] = \sum_{j=0}^n a_j(i) 2^{-j-1} = a_0 2^{-1} + a_1 2^{-2} + \dots$$

$$1101 \Rightarrow 0.1011 = 1/2 + 1/8 + 1/16 = 11/16 = 0.6975$$

$$p_j = (x_i, y_i) = \left[\frac{i}{n}, \Phi_2(i) \right]$$



Low-Discrepancy Sampling

- In general, number of samples should be **proportional to area**
- **Discrepancy** measures deviation from this ideal

discrepancy of sample points X in a region S

of samples in S

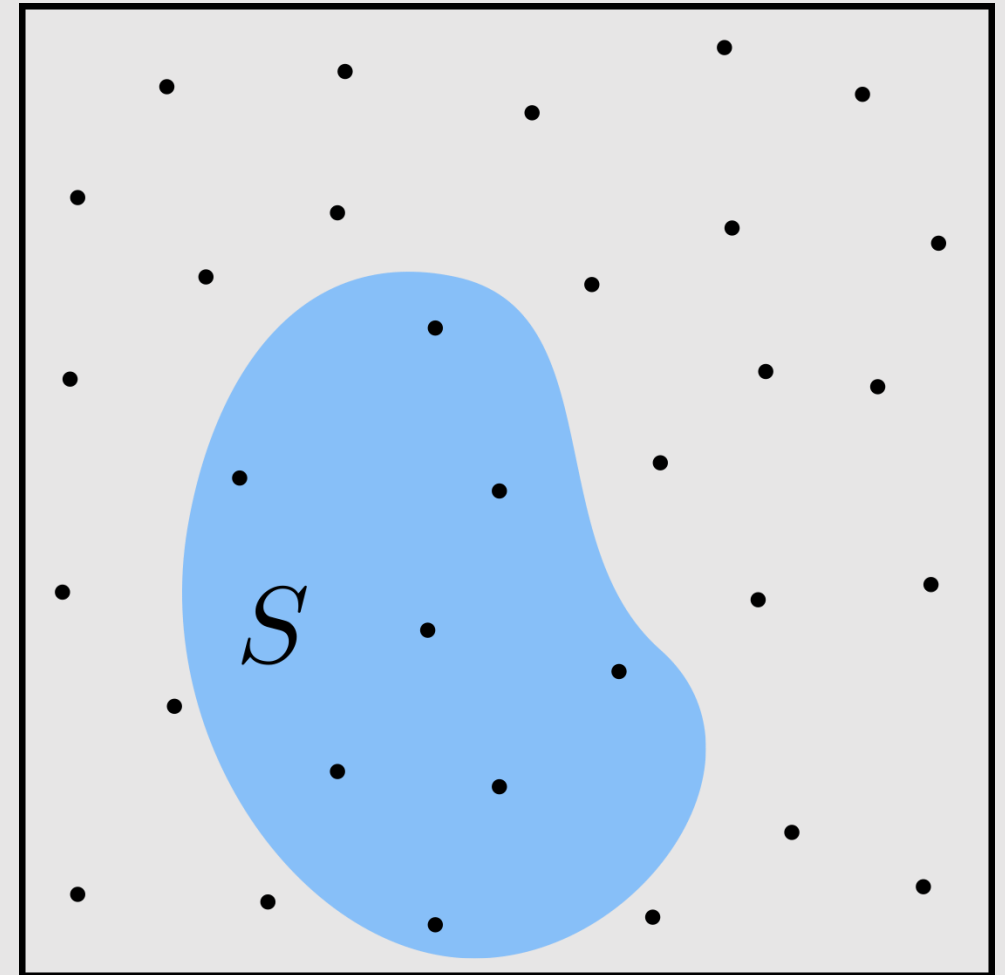
$$d_S(X) := \left| A(S) - \frac{n(S)}{|X|} \right|$$

area of S total # of samples in X

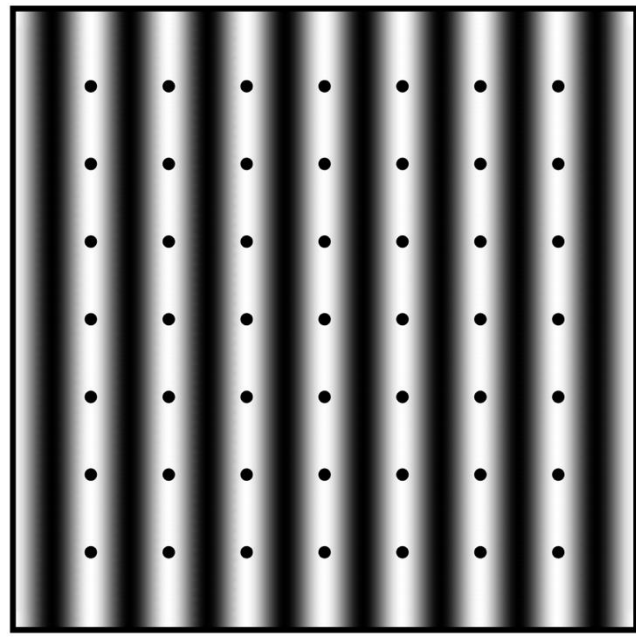
$$D(X) := \max_{S \in \mathcal{F}} d_S(X)$$

overall discrepancy

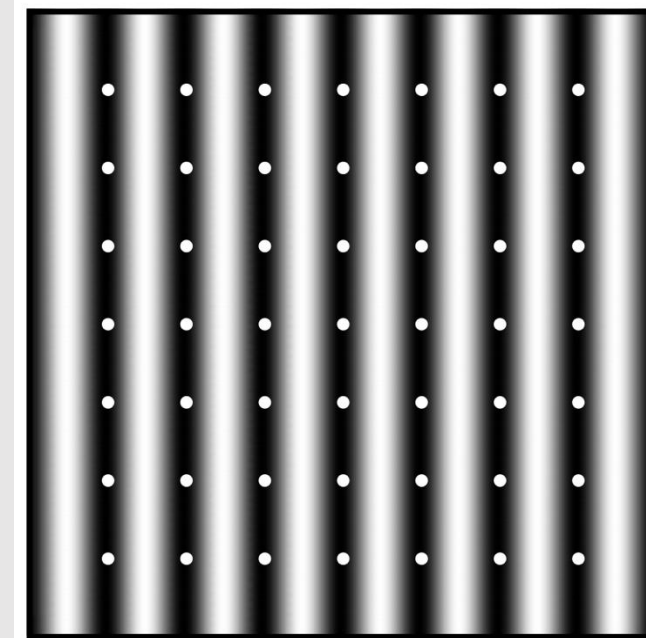
some family of regions S (box, disk, etc...)



Low-Discrepancy Sampling



$$\frac{1}{n} \sum_{i=1}^n f(x_i) = 1$$



$$\frac{1}{n} \sum_{i=1}^n f(x_i) = 0$$

- A uniform grid has the lowest discrepancy
 - But even low-discrepancy patterns can exhibit poor behavior
 - We want patterns to be **anisotropic** (no preferred direction)

Blue Noise

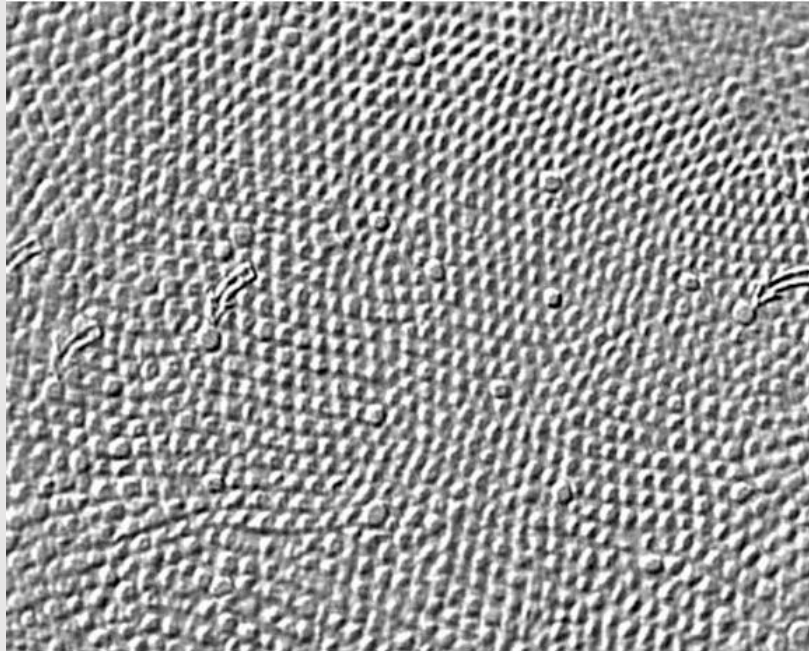
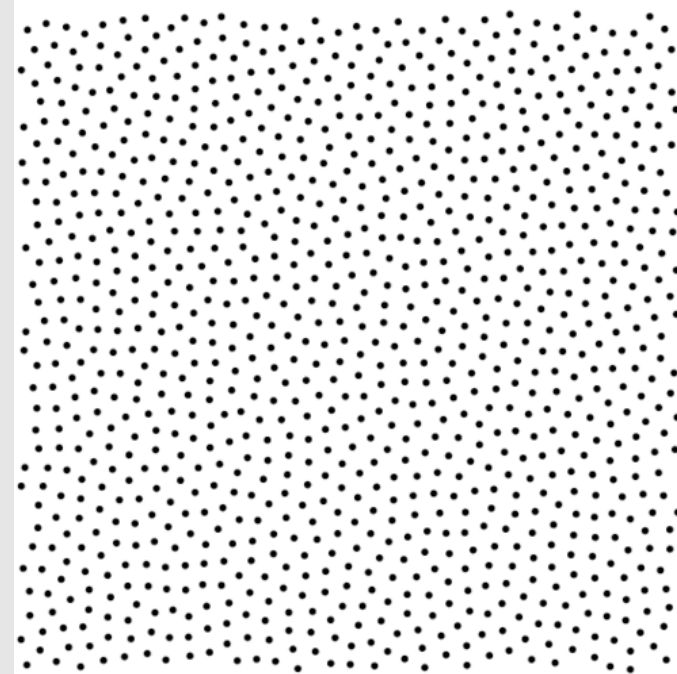


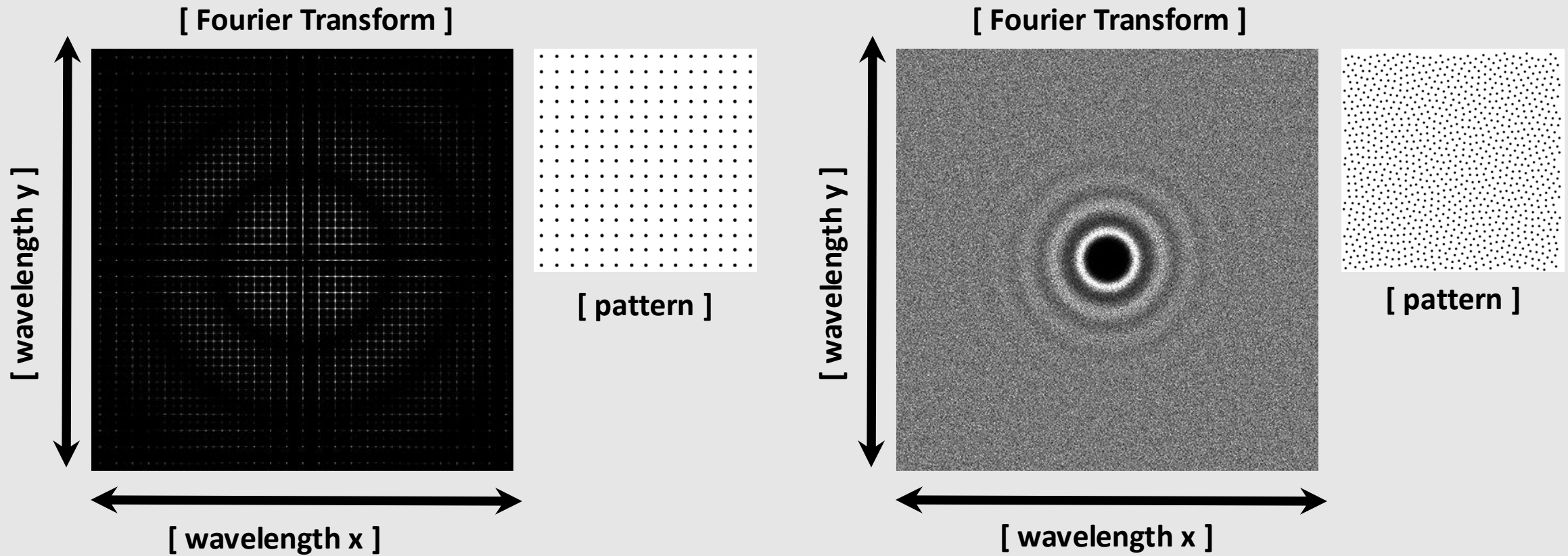
Fig. 13. Tangential section through the human fovea. Larger cones (arrows) are blue cones. From Ahnelt et al. 1987.



["blue noise"]

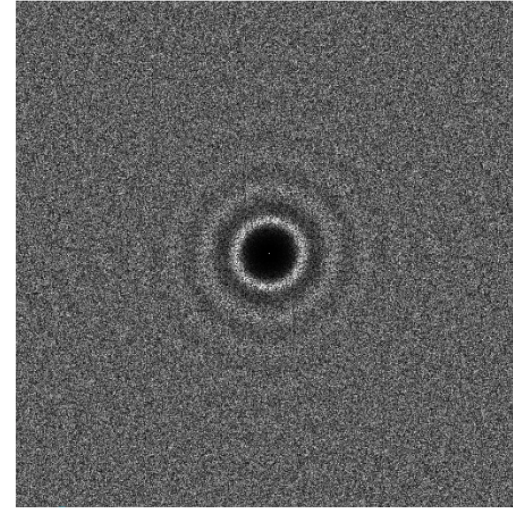
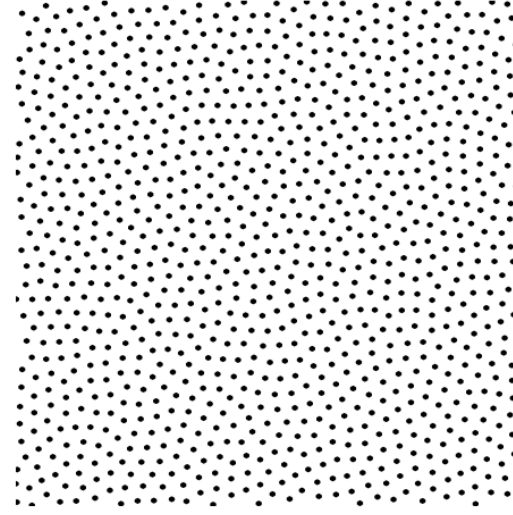
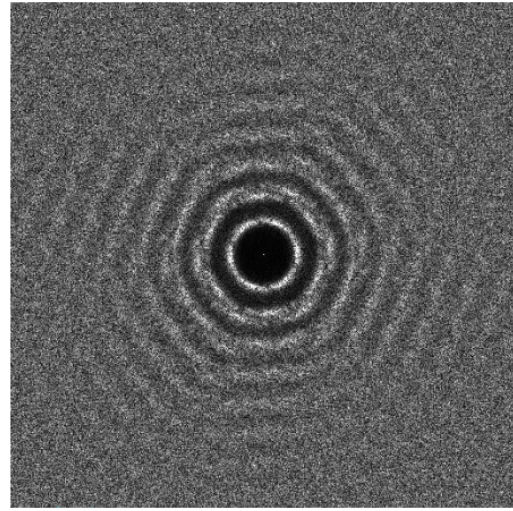
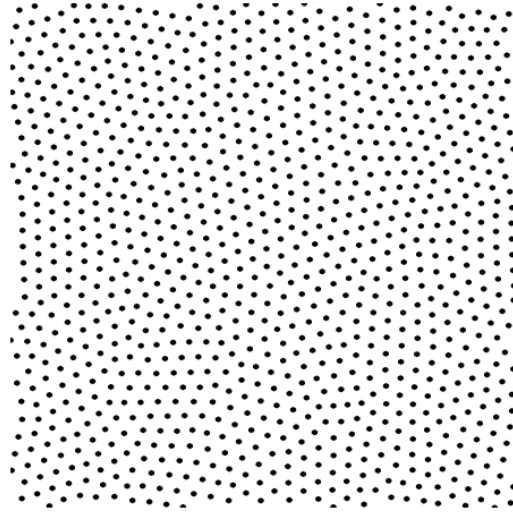
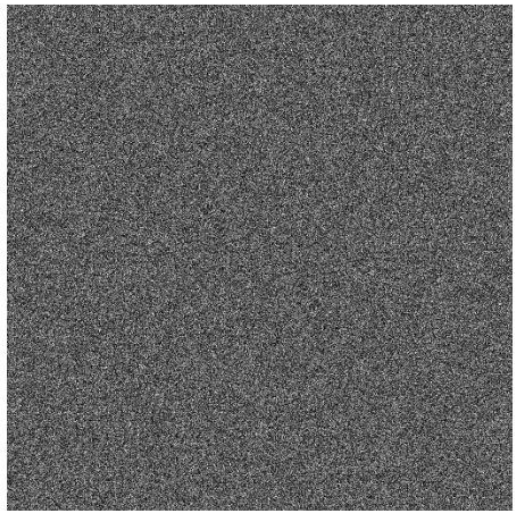
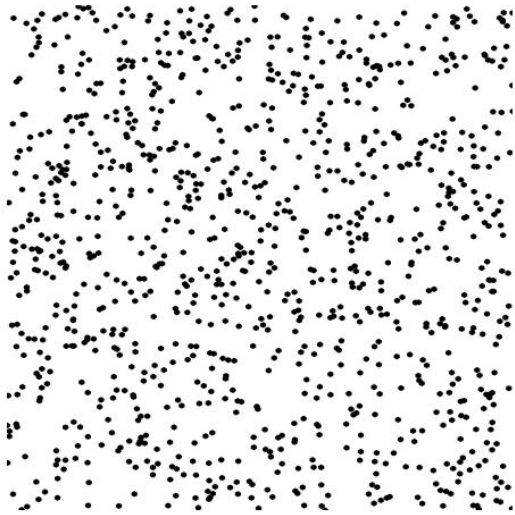
- Monkey retina exhibits **blue noise** pattern [Yellott 1983]
 - No preferred directions (**anisotropic**)

Blue Noise Fourier Transform



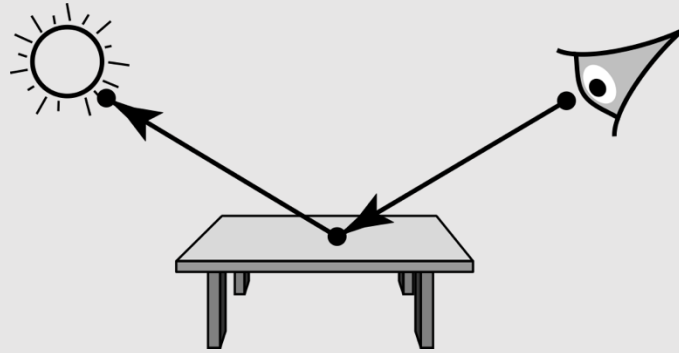
- Regular pattern has “spikes” at regular intervals
- Blue noise is spread evenly over all frequencies in all directions
 - Bright center “ring” corresponds to sample spacing

Blue Noise Fourier Transform



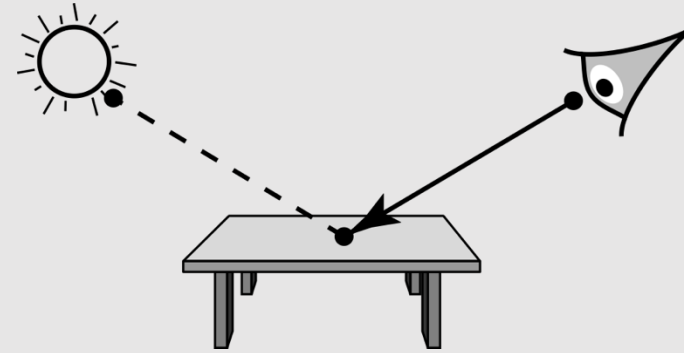
- ~~Monte Carlo Sampling~~
- ~~Biased vs Unbiased Estimators~~
- Physically-Based Rendering Methods

Previous Methods



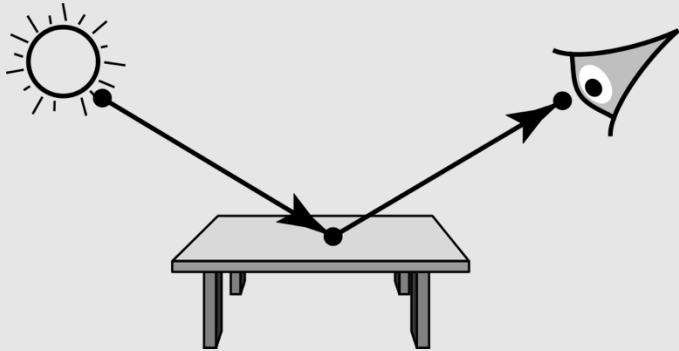
[backward path tracing]

Fails: cannot intersect point lights



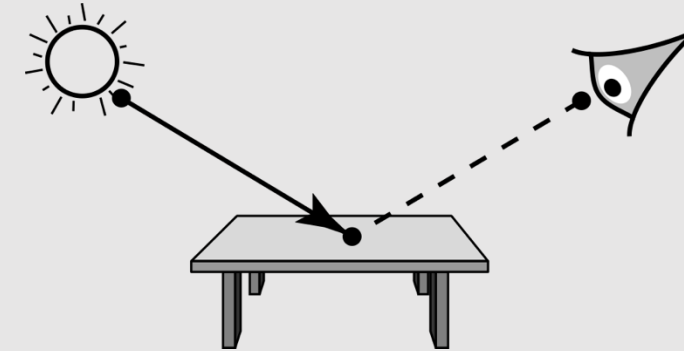
[backward path tracing + connect to light]

Works: reaches point lights



[forward path tracing]

Fails: cannot intersect pinhole camera

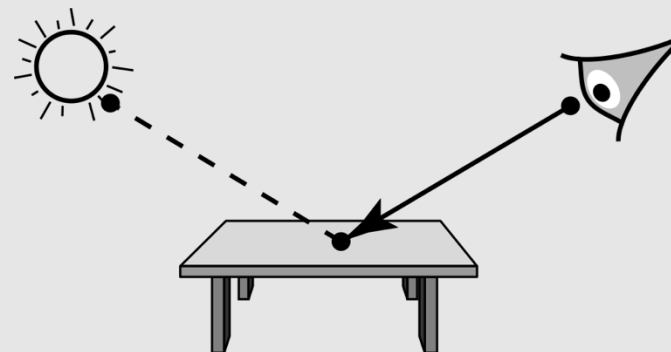


[forward path tracing + connect to camera]

Works: reaches pinhole camera

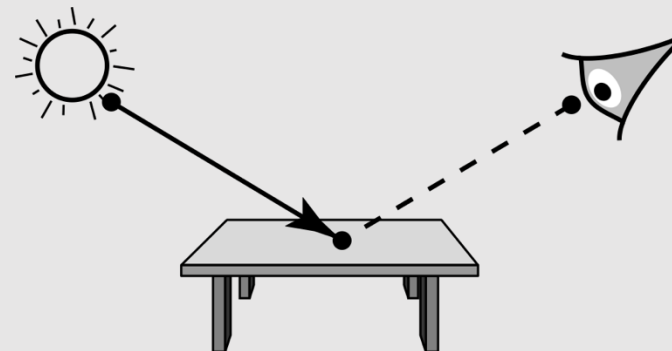
Path Tracing Can Be Biased

- Deliberately connect terminating rays to light (forward) or camera (backward)
- Probability of sampling a ray that hits a non-volume source (point light, pinhole camera) is 0
 - We bias our renderer by choosing those rays



[backward path tracing + connect to light]

works: reaches point lights

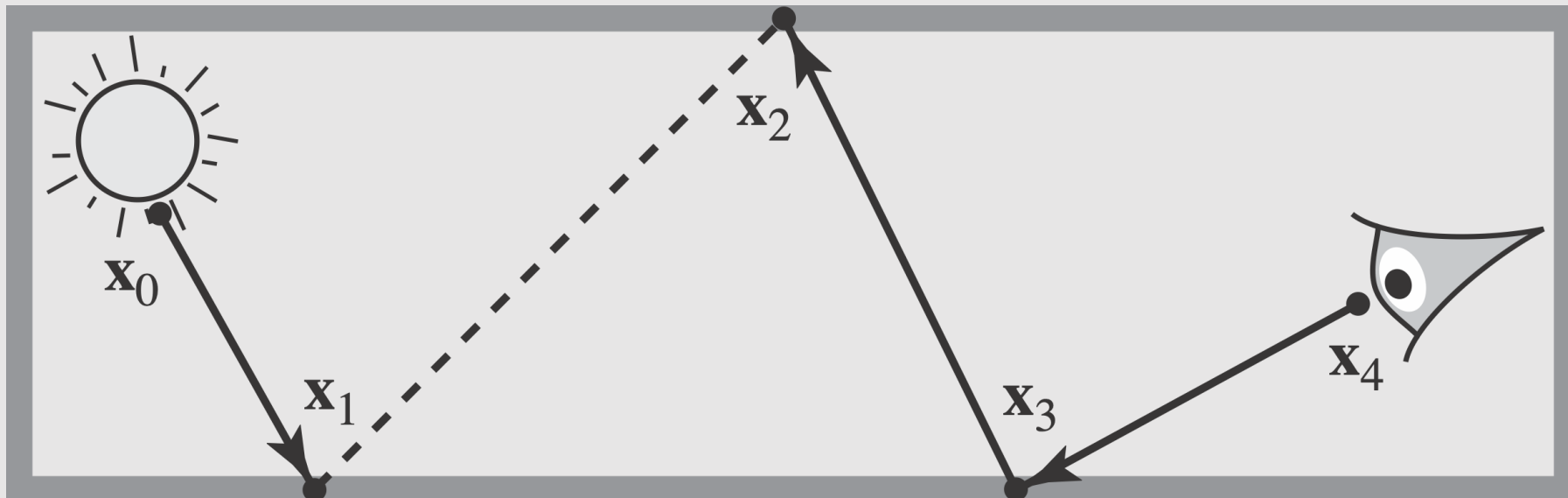


[forward path tracing + connect to camera]

works: reaches pinhole camera

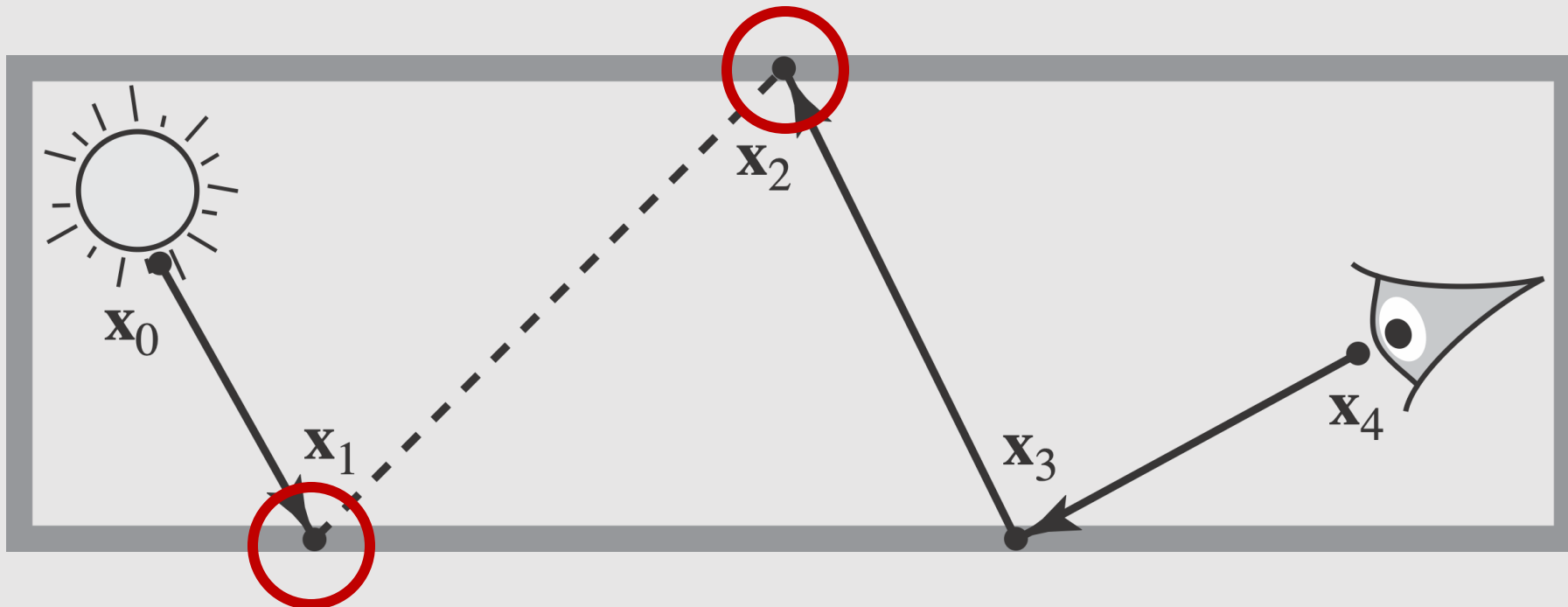
Bidirectional Path Tracing

- If path-tracing is so great, why not do it **twice**?
 - Main idea of bidirectional!
- Trace a ray from the camera into the scene
- Trace a ray from the light into the scene
 - Connect the rays at the end
- Unbiased algorithm
 - No longer trying to connect rays through non-volume sources
- Can set different lengths per ray
 - Example: Forward $m = 2$, Backward $m = 1$



Bidirectional Path Tracing

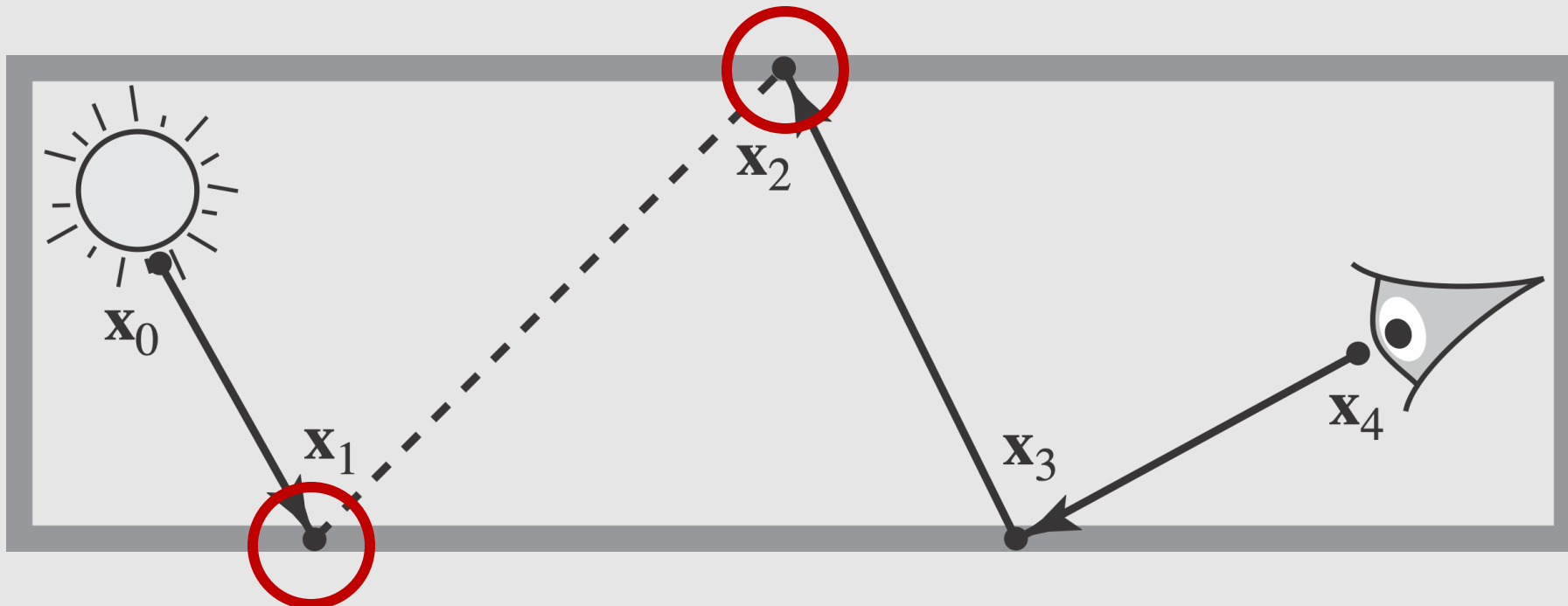
Issue: what if these are mirrors!



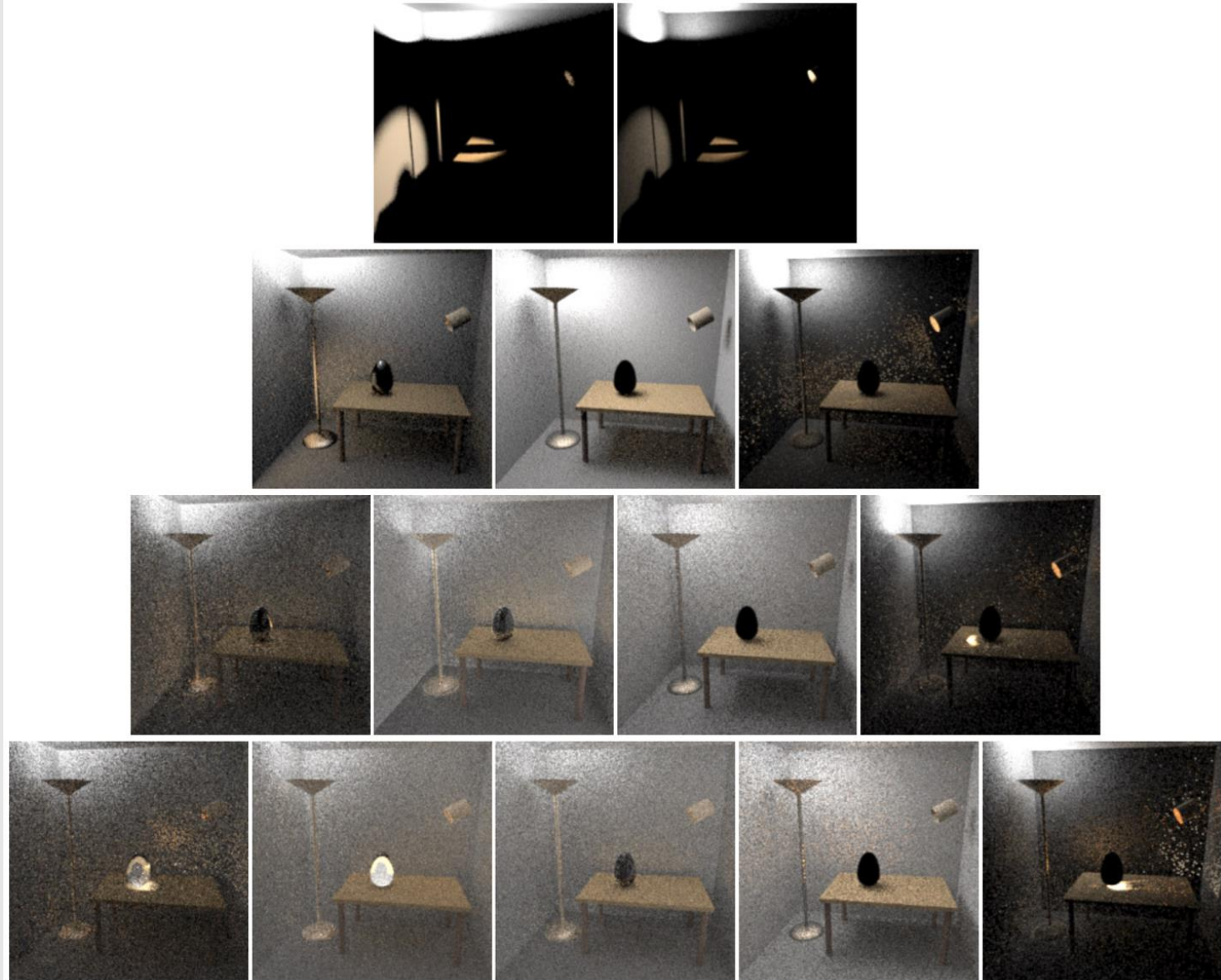
Bidirectional Path Tracing

- In cases of mirrors, we cannot choose any ray path
- Instead, continue tracing rays until diffuse surfaces are reached on both rays

Issue: what if these are mirrors!



Bidirectional Path Tracing



[final image]

- Each row shows path length
- As we move over images in a row, we decrease forward ray depth and increase a backward ray depth
 - Overall length kept constant per row

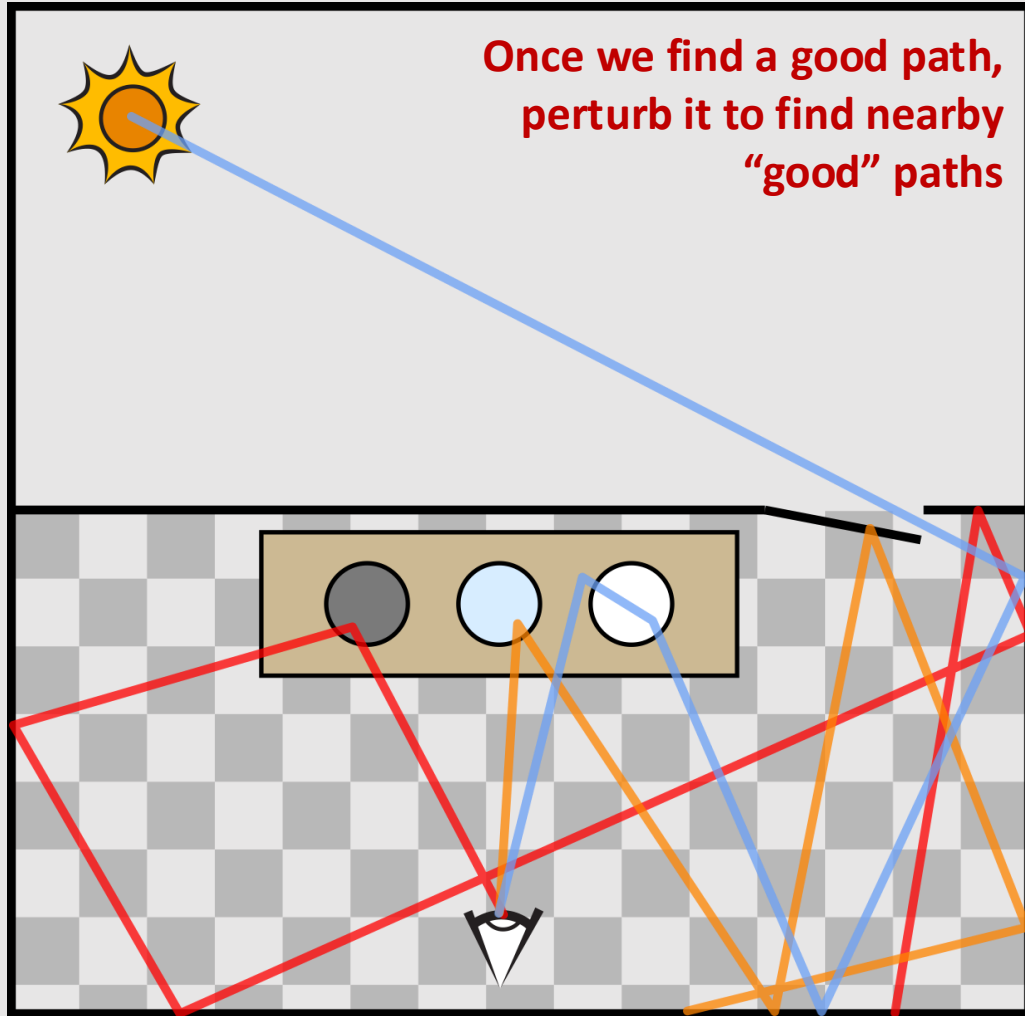
Bidirectional Path Tracing



[final image]

- Not easy to tell which path lengths work well for a scene!
 - The glass egg is illuminated at specific path lengths for forward and backward rays

Good Paths Are Hard To Find



[Bidirectional Path Tracing]

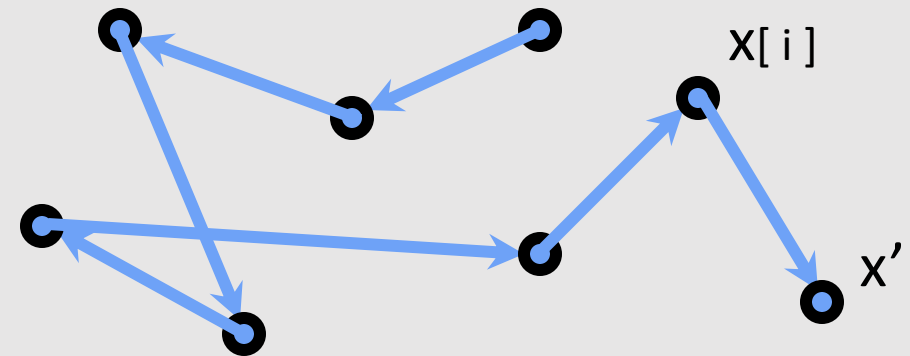


[Metropolis Light Transport]

Metropolis Hasting Algorithm

- “Once we find a good path, perturb it to find nearby ‘good’ paths” – previous slide
- **Algorithm:** take random walk of dependent samples
 - If in an area where sampling yields high values, stay in or near the area
 - Otherwise move far away
- Sample distribution should be proportional to integrand
 - Make sure mutations are “ergodic” (reach whole space)
 - Need to take a long walk, so initial point doesn’t matter

```
float r = rand();  
// if  $f(x') \gg f(x[i])$ , then a is large  
// and we increase chances of moving to  $x'$   
// if  $f(x') \ll f(x[i])$ , then a is small  
// and we increase chances of staying at  $x$   
float a = f(x')/f(x[i]);  
if (r < a)  
    x[i+1] = x';  
else  
    x[i+1] = x;
```



Metropolis Hasting: Sampling An Image



[short walk]

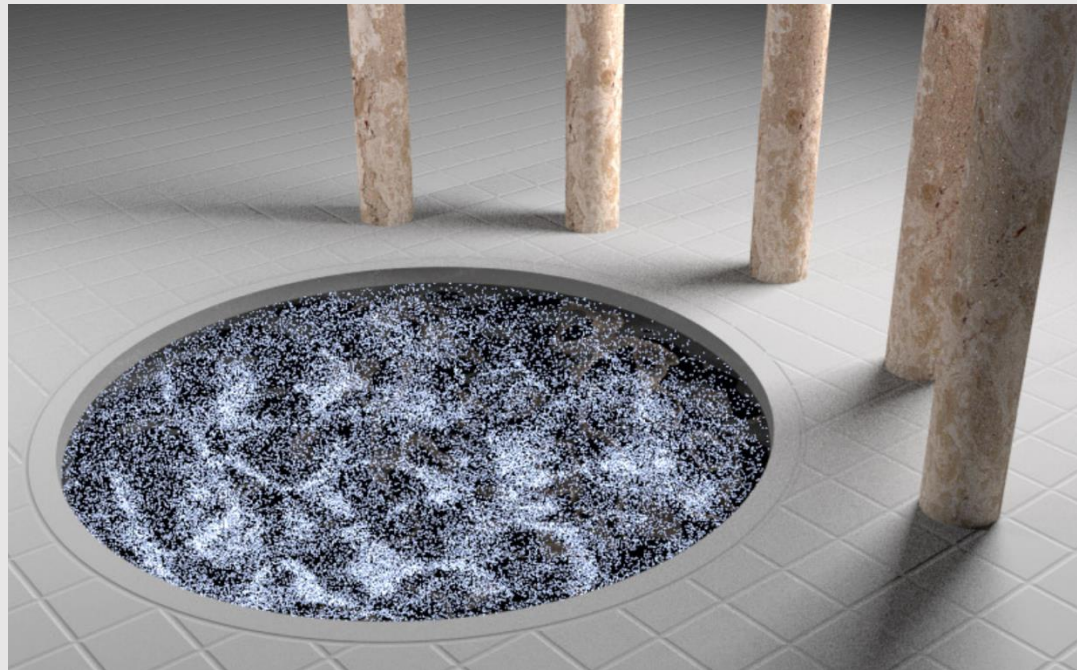
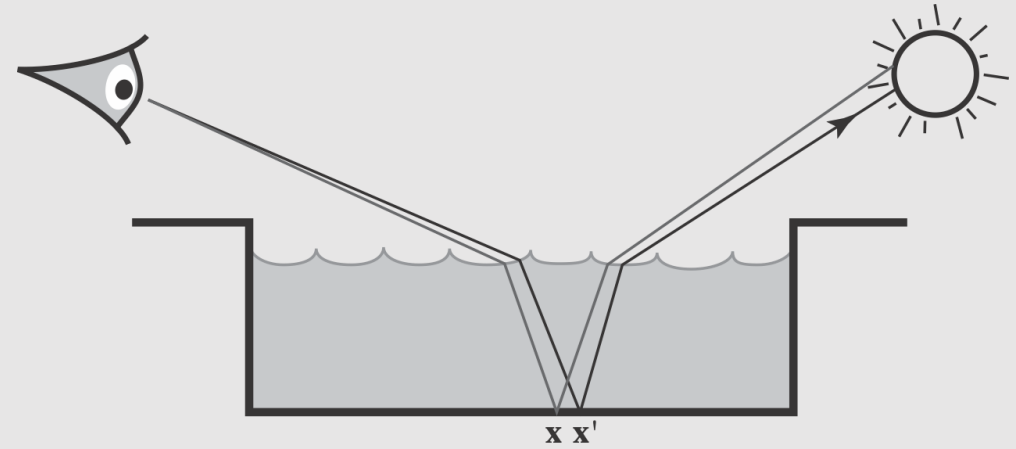
[long walk]

[original image]

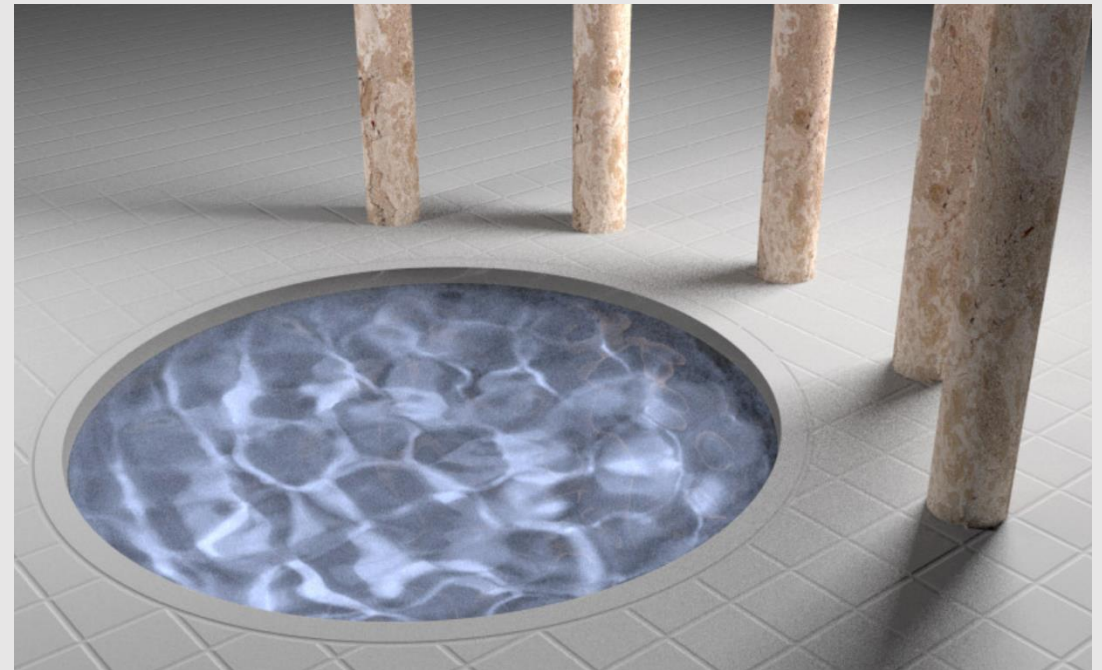
- Want to take samples proportional to image density f
- Occasionally jump to a random point (ergodicity)
- Transition probability is 'relative darkness'
 - $f(x')/f(x_i)$

Metropolis Light Transport

- **Similar idea:** mutate good paths
- Water causes paths to refract a lot
 - Small mutations allows renderer to find contributions faster
- Path Tracing and MLT rendered in the same time



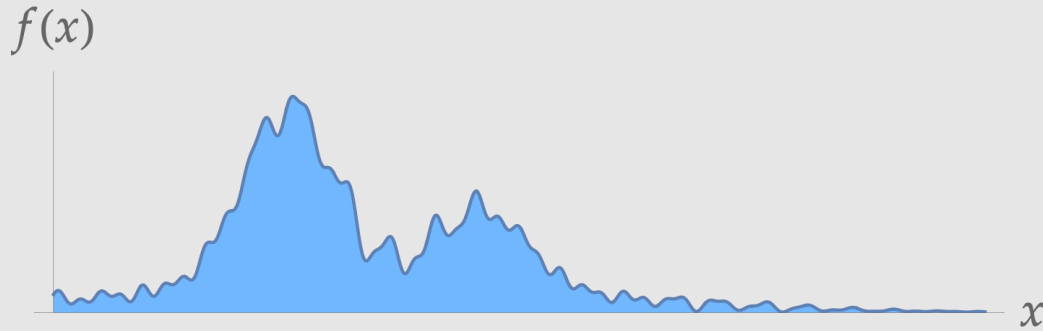
[Path Tracing]



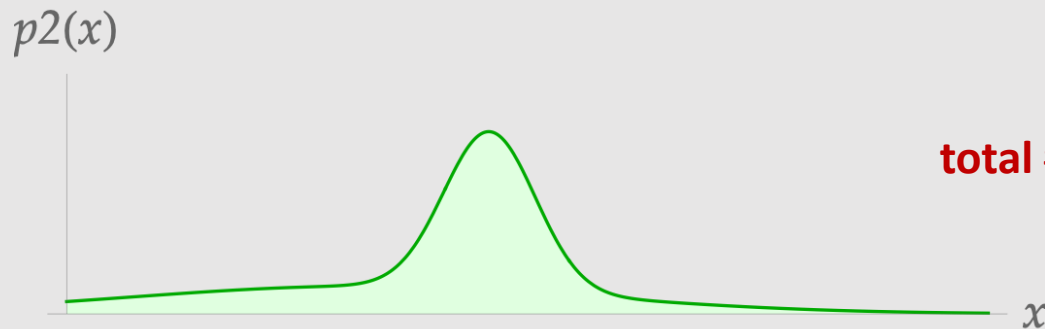
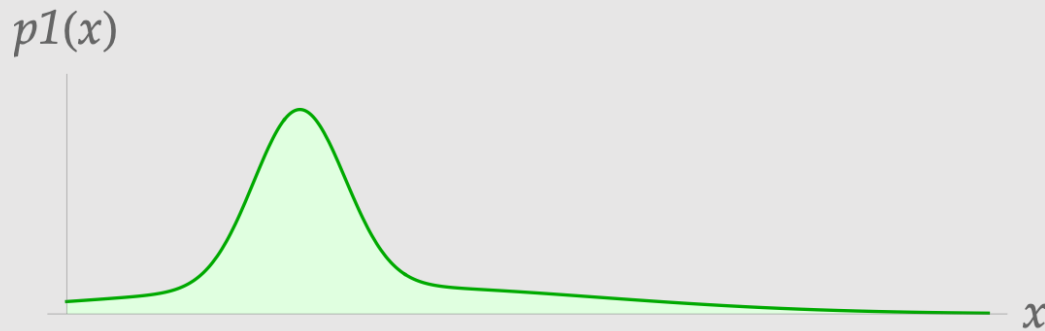
[Metropolis Light Transport]

If there are so many good sampling methods,
why not combine them?

Multiple Importance Sampling



- **Multiple Importance Sampling:** combine strategies to preserve strengths of all of them
 - Think of it as taking multiple rays/samples at each bounce



$$\frac{1}{N} \sum_{i=1}^n \sum_{j=1}^{n_i} \frac{f(x_{ij})}{\sum_k c_k p_k(x_{ij})}$$

sum over strategies (points to the outer sum)

sum over samples (points to the inner sum)

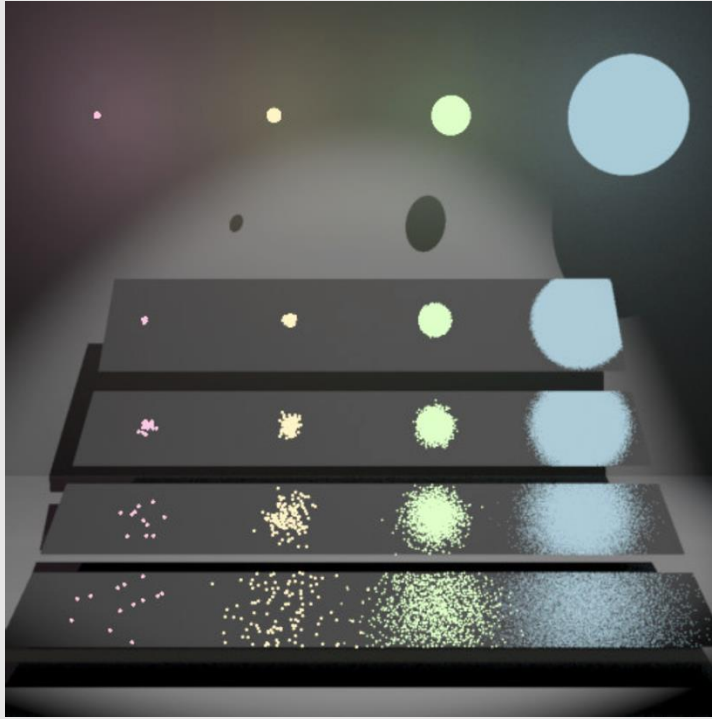
j^{th} sample taken with i^{th} strategy (points to x_{ij})

total # of samples (points to N)

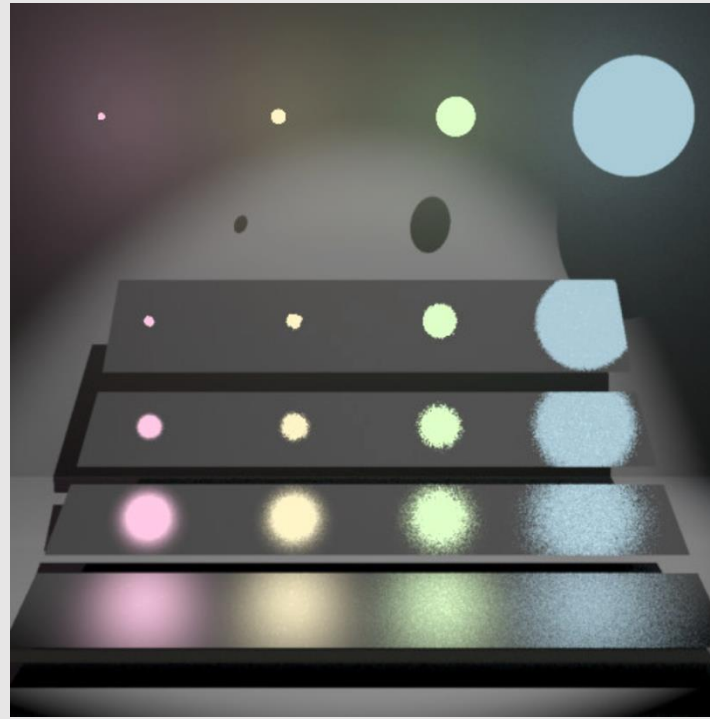
fraction of samples taken with k^{th} strategy (points to c_k)

k^{th} strategy PDF (points to $p_k(x_{ij})$)

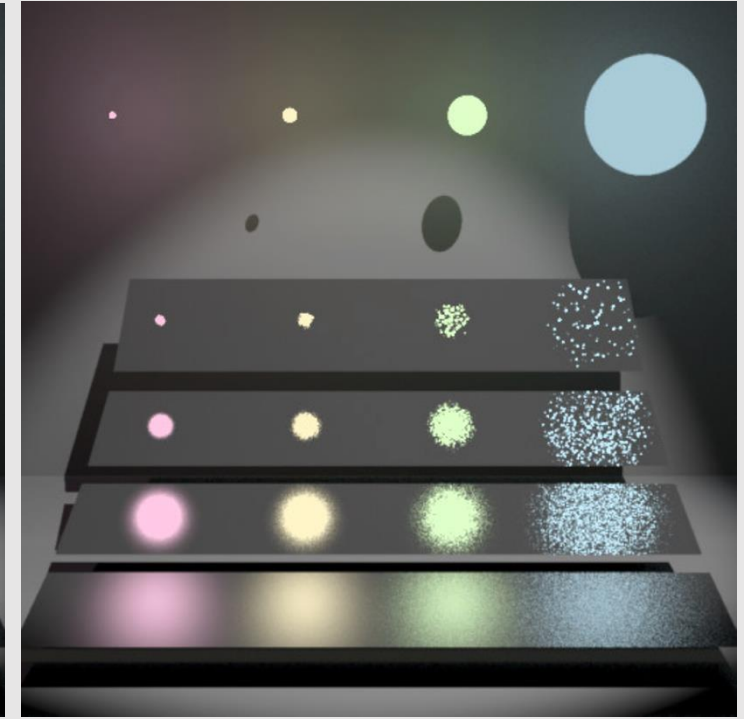
Multiple Importance Sampling



[sample materials]



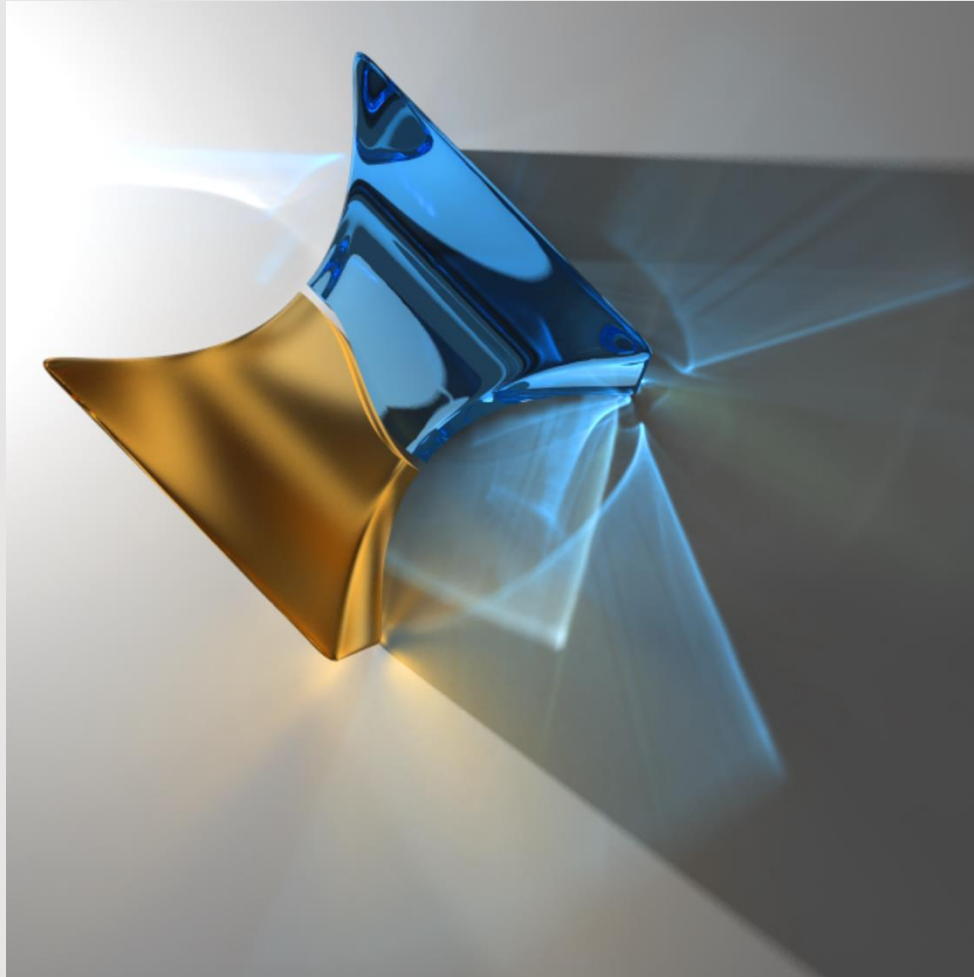
[sample both]



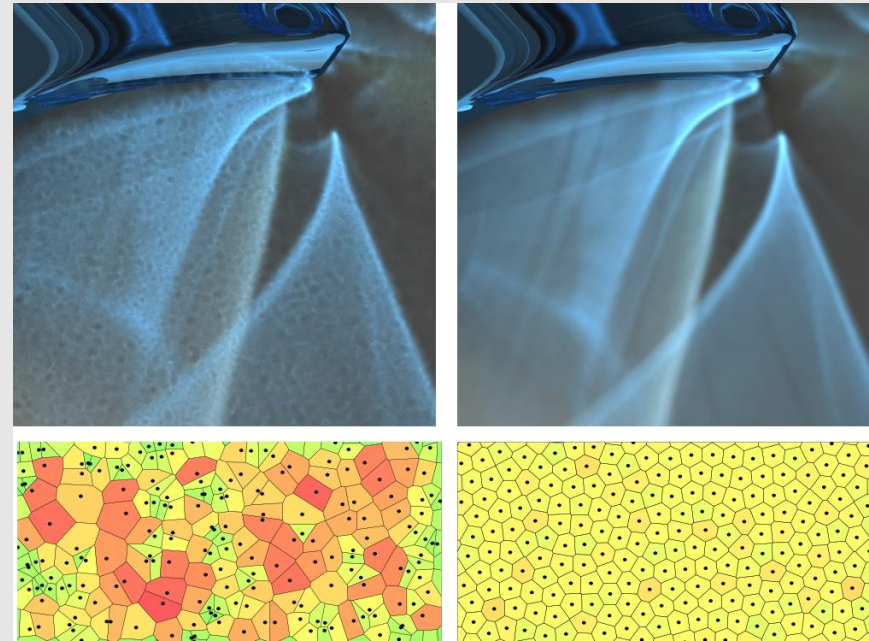
[sample lights]

- Normally need to pick next ray bounce as hitting a material or hitting light
 - MIS allows us to take both rays and average them together
 - At each bounce, trace a ray as normal, and another ray to the light

Photon Mapping

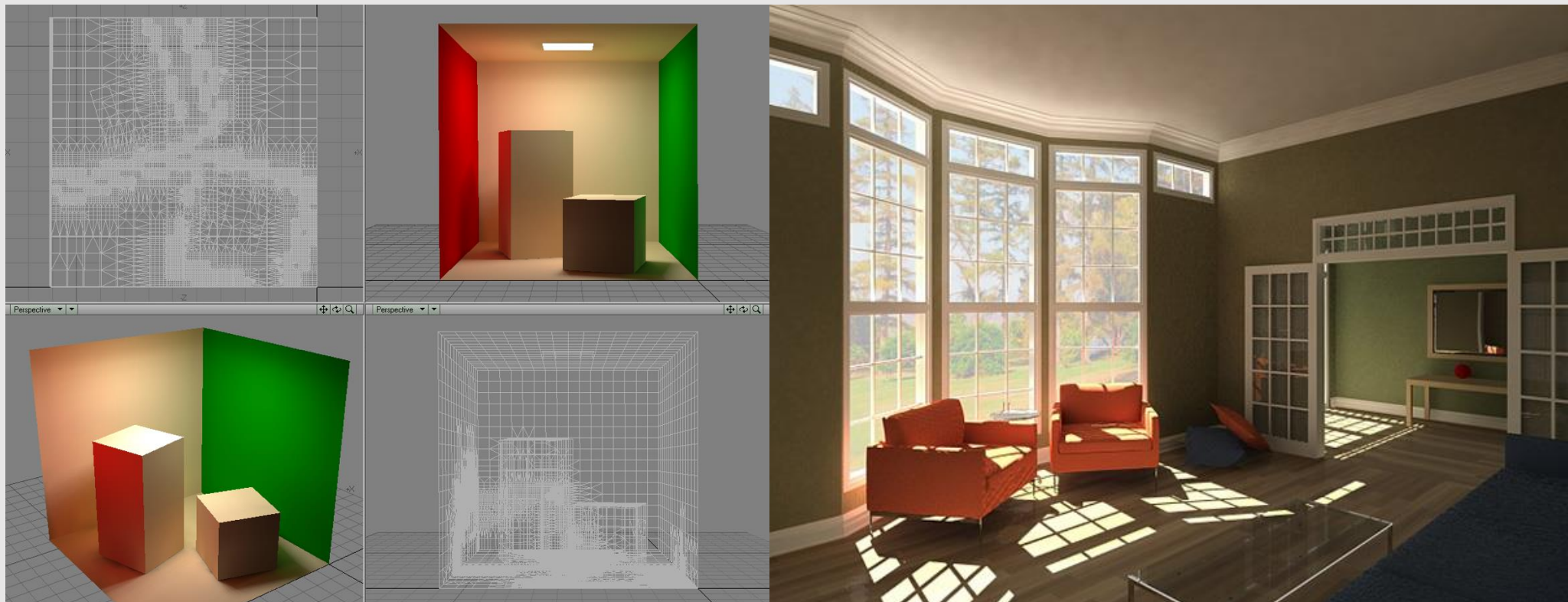


- Trace particles from light, deposit “photons” in KD-tree
 - Useful for, e.g., caustics, fog
- Voronoi diagrams can improve photon distribution
 - **Careful:** poor Voronoi resolution causes aliasing!



Finite Element Radiosity

- Transport light between patches in scene
- Solve large linear system for equilibrium distribution
 - Good for diffuse lighting; hard to capture other light paths
 - Light paths travel in groups
 - Difficult when light diverges



Rendering Algorithm Chart

method	consistent?	unbiased?
Rasterization	no	no
Path Tracing	almost	almost
Bidirectional Path Tracing	yes	yes
Metropolis Light Transport	yes	yes
Photon Mapping	yes	no
Finite Element Radiosity	no	no