Variance Reduction

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

What Makes A Render Expensive

- **Number of Rays**
  - How many rays we trace into the scene
    - Measured as samples (rays) per pixel [spp]
- **Number of Ray Bounces**
  - How many times a ray bounces before it terminates
    - Measured as ray bounce or depth
- Choosing the right number is a difficult problem to solve
  - Don’t worry, we will tell you how many rays and how many bounces to cast :)

Star Wars VII: The Force Awakens (2015) Lucasfilm

Number Of Ray Samples

- **Number of Rays**
  - How many rays we trace into the scene
    - Measured as samples (rays) per pixel [spp]

- Increasing the number of rays increases the quality of the image
  - Anti-aliasing
  - Reduces black spots from terminating emission occlusion

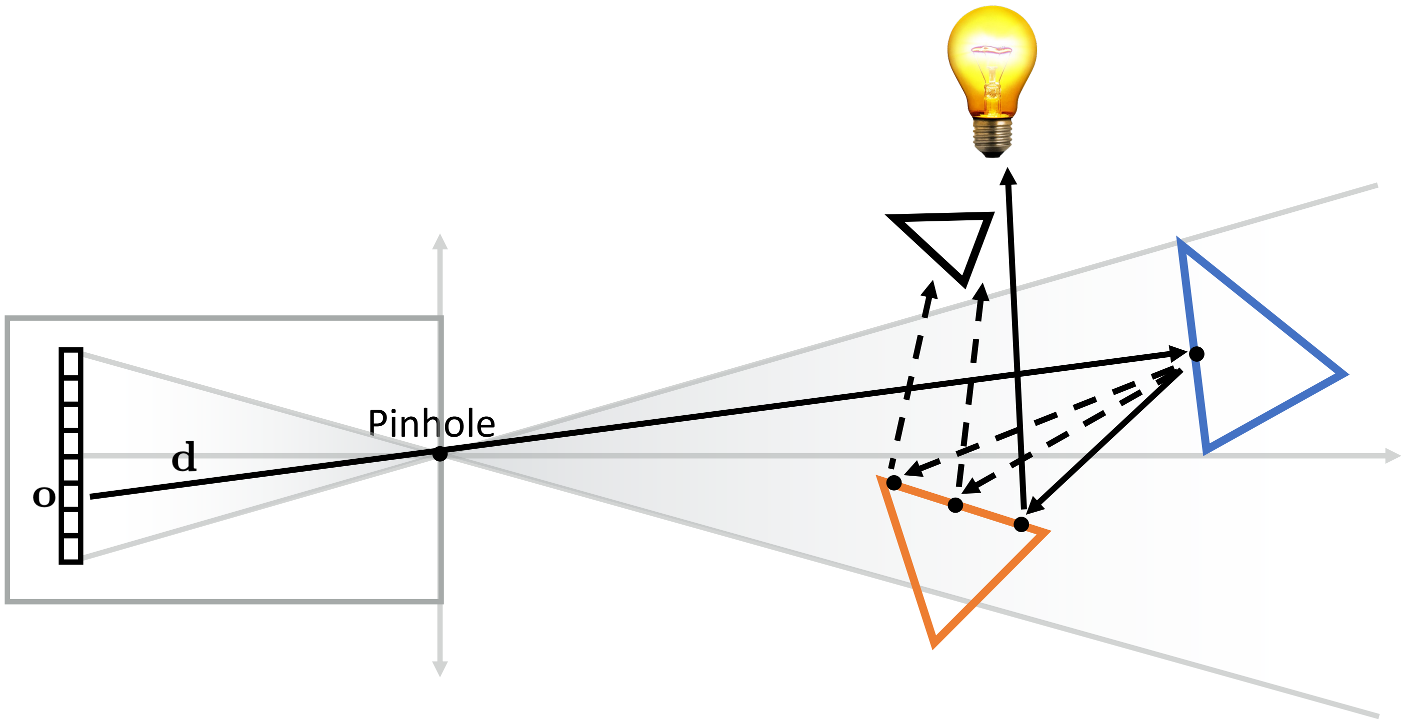
[1 spp]

[16 spp]

Number Of Ray Samples

- Having more rays similar to taking more samples in rasterization
  - Samples taken in a larger sample buffer and resolved into output buffer

- More likely for terminating ray to reach light source/not be occluded



Number Of Ray Bounces

- **Number of Ray Bounces**
    - How many times a ray bounces before it terminates
        - Measured as ray bounce or depth
- Increasing the number of ray bounces increases the quality of the image
    - Better color blending around images
    - More details reflected in specular images

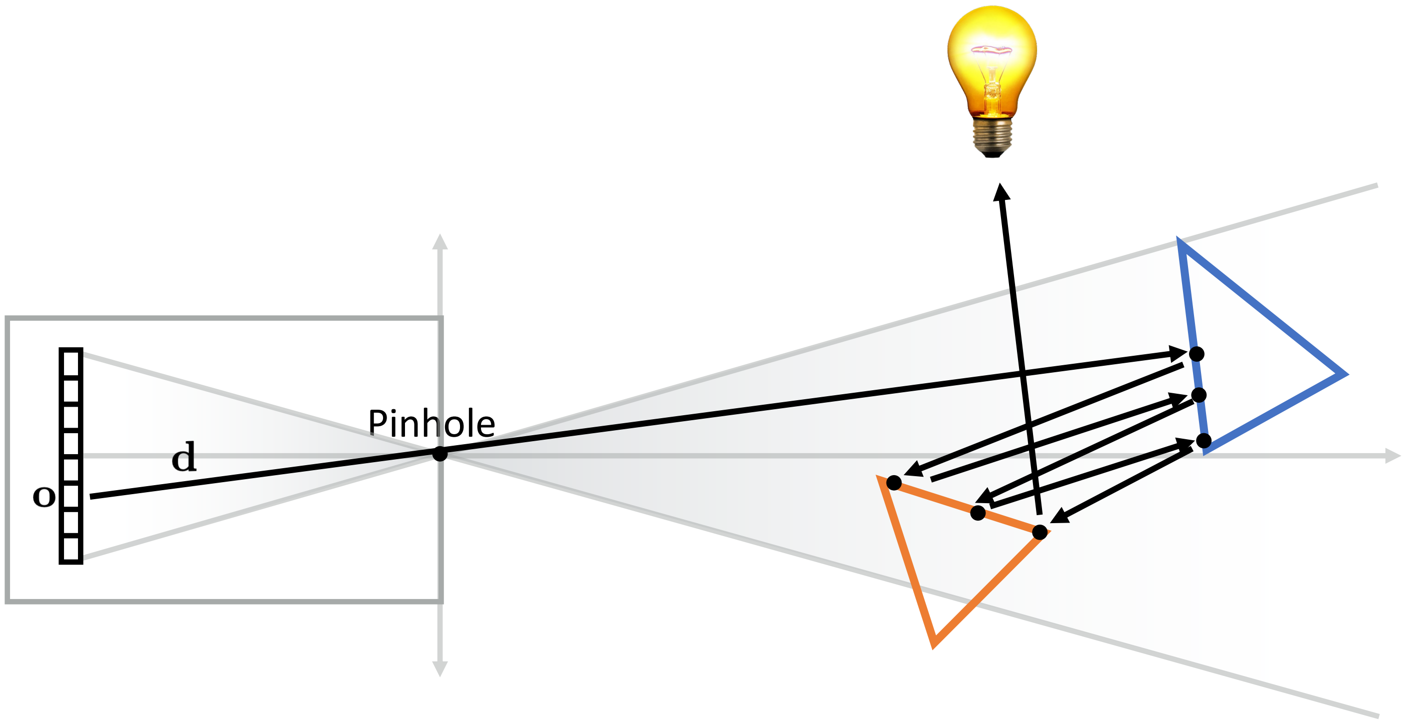
[2 depth]

[8 depth]

Number Of Ray Bounces

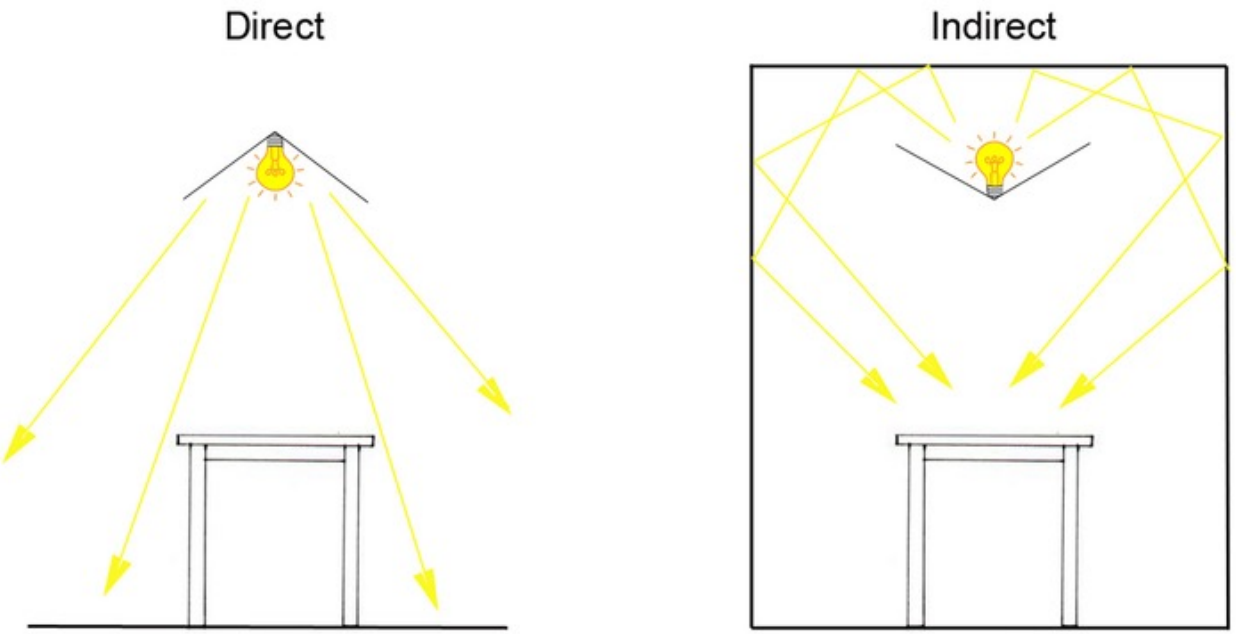
- Having more ray bouncing allows for better color blending
  - Final ray will be a larger mix of blue/orange than the original yellow

- Can render more interesting reflective and refractive paths with more bounces

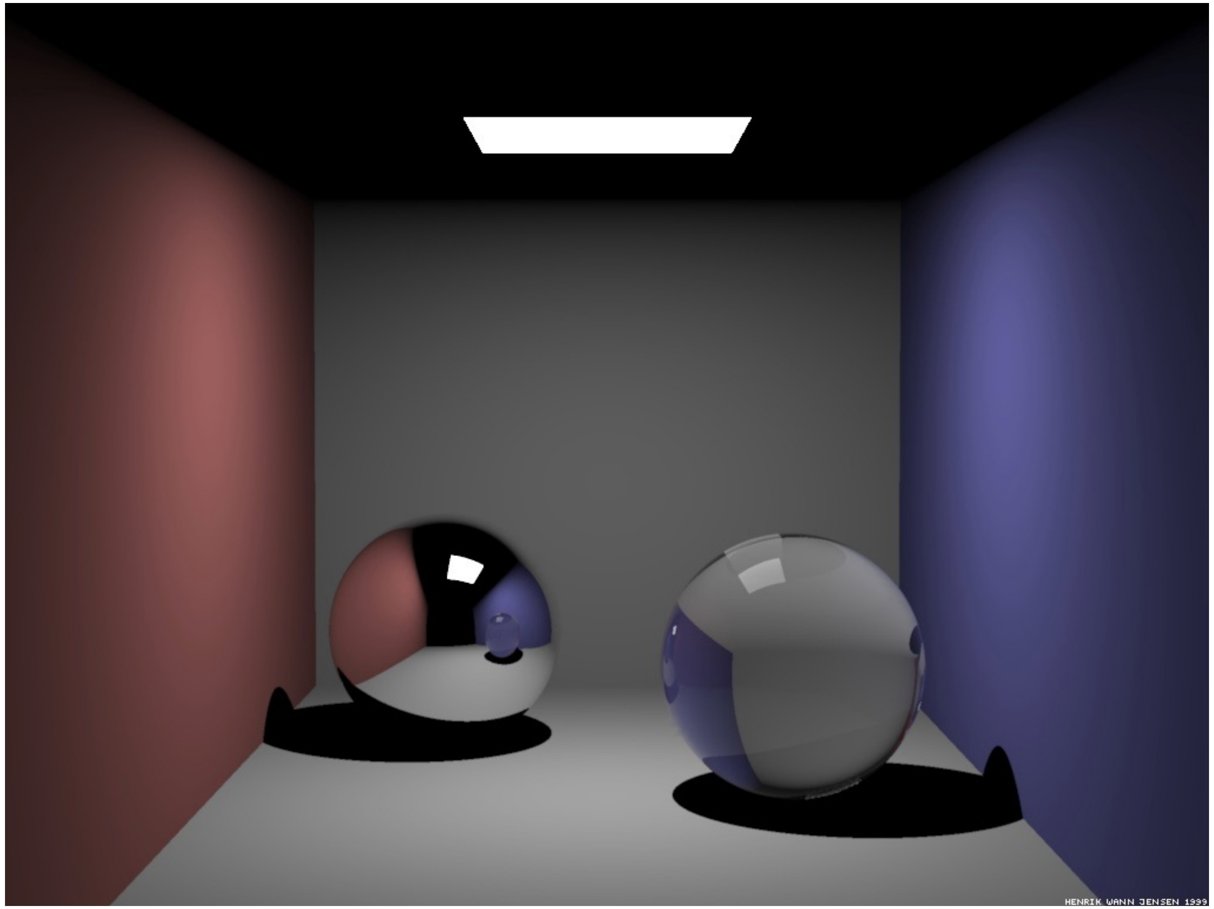


Direct VS Indirect Illumination

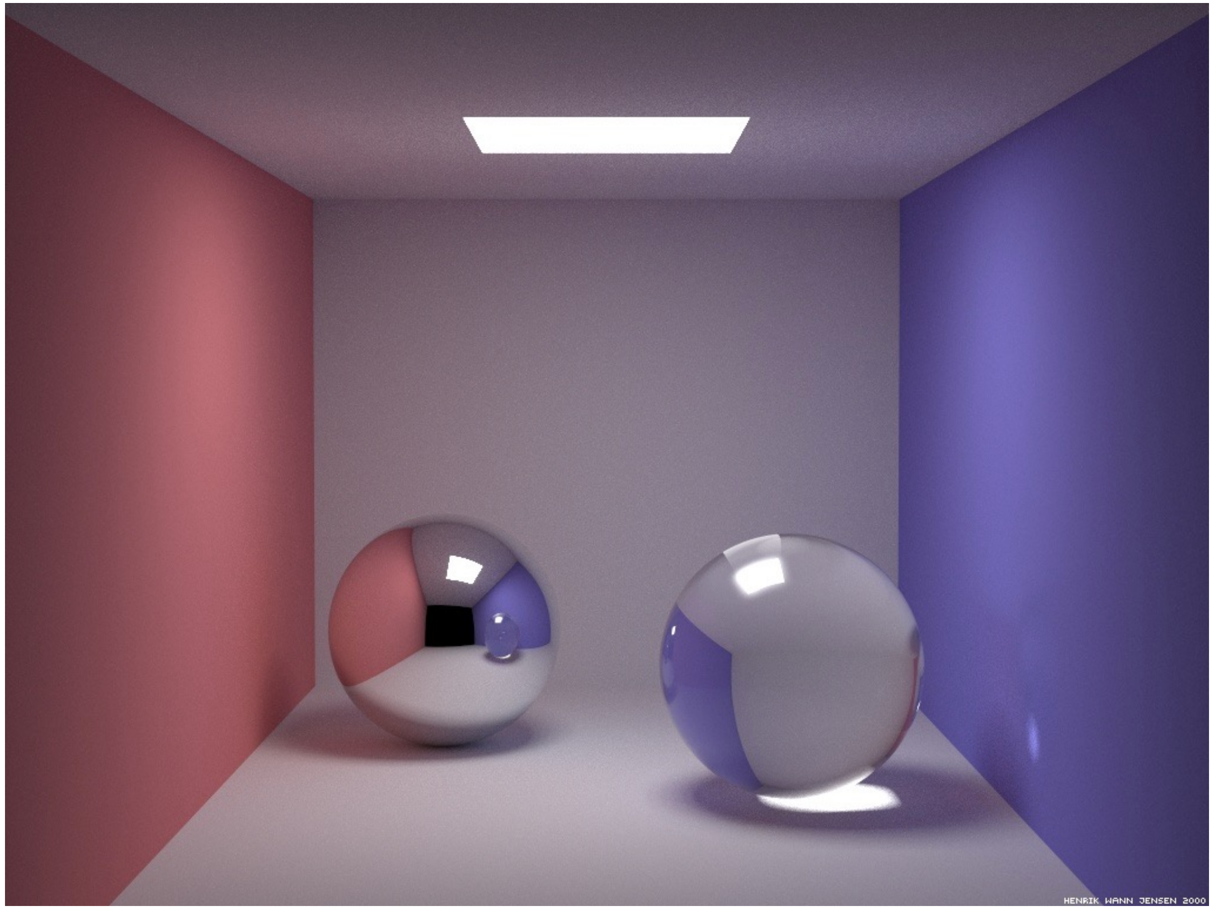
- Direct Illumination: Direct path from emitter to point
- Indirect Illumination: Multi-bounce path from emitter to point
- We use the term ‘bounce’ to describe how many piecewise linear rays we can stich together to form a path
  - Direct is 1-bounce
  - Indirect is N-bounce
    - Although some authors say Direct is 0-bounce [index at 0]



Direct VS Indirect Illumination



[Direct + Reflection + Refraction]**

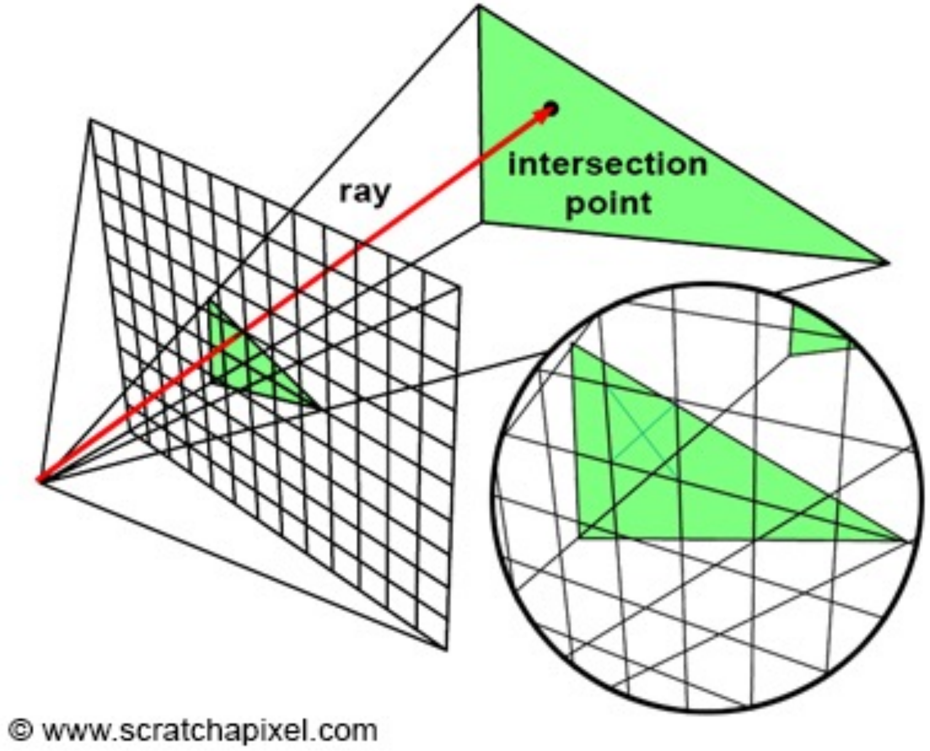


[Global]

**Normally can't do reflection & refraction in direct illumination

Wait a minute…
direct illumination looks like rasterization

Direct VS Rasterization



- We can think of rasterization as tracing a ray from a point in the output buffer to a shape in the scene
  - Recall even in rasterization, shapes have depth
  - We only care about the closest object we see (transparency disabled)
- Both rasterization and direct illumination only ever trace **one ray**!

Direct VS Indirect Illumination

Minecraft (2020) Microsoft

- Direct Illumination gives you efficiency
  - Very easy to render
  - Fairly straightforward in complexity
  - Comparable to rasterization in difficulty
  - Amendable to ray packeting
  - Easy to make real-time
- Indirect (Global) Illumination gives you quality
  - Some materials require multi-bounces
    - Ex: refraction
  - Ambient occlusion
  - Higher contrast
  - Samples converge to true values
- More bounces = ↓ efficiency, ↑ quality

So how do we take multiple samples?

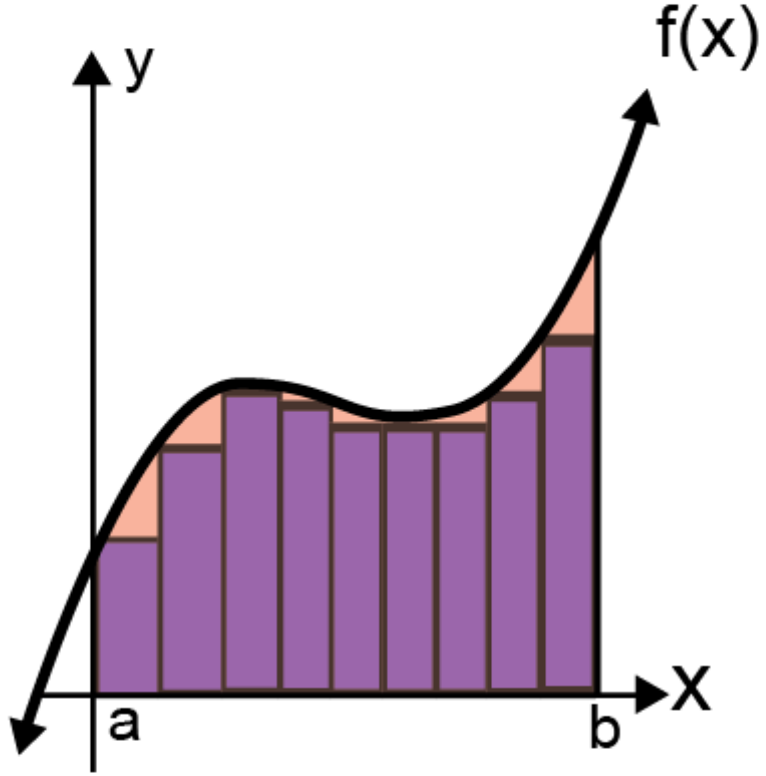
Continuous Vs. Discrete

- Our eyes see a continuous signal of energy
- Our digital cameras see a discrete signal of energy
  - Computers can only process discrete values
- Let the following integral be the true continuous signal of the scene

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

- We want to approximate the integral by taking multiple samples of our discrete scene function

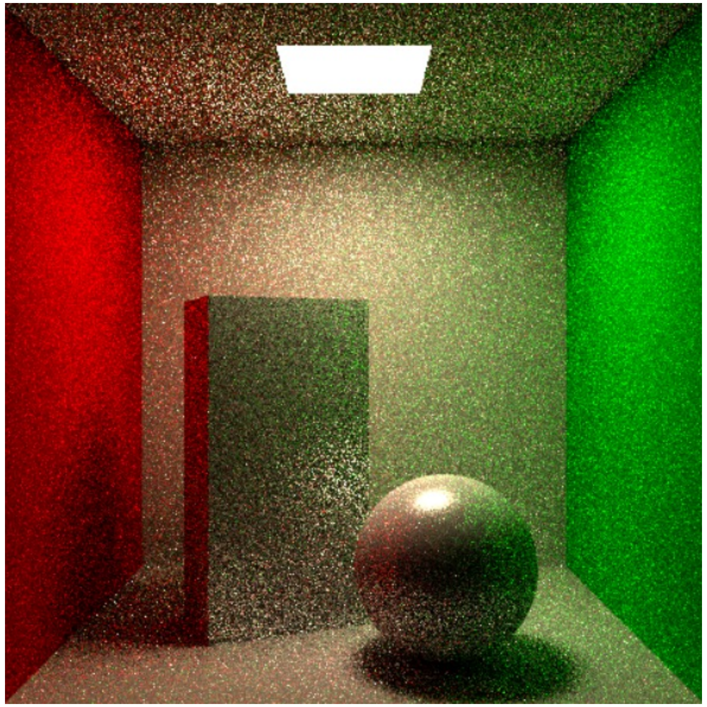
$$\frac{1}{N} \sum_{i=1}^{N} f_r(\mathbf{p}, \omega_i \rightarrow \omega_o) L_i(\mathbf{p}, \omega_i) \cos\theta$$



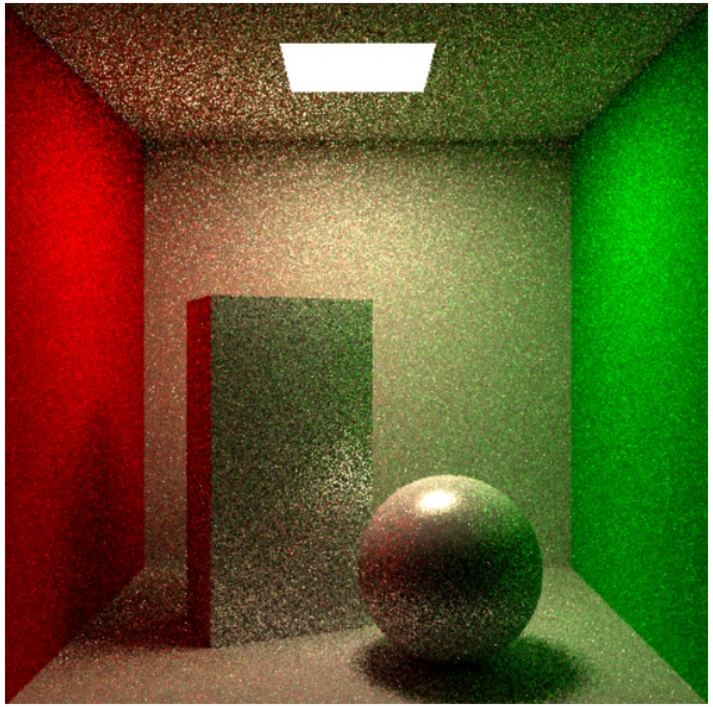
Sampling Rays

- **Issue:** Responsible for picking rays since we are no longer integrating over every possible ray direction in hemisphere
  - Some rays will be better than others

- **Idea:** pick rays from a PDF
  - **Uniform PDF:** ray sampled in uniformly random direction in hemisphere
  - **Cos-weighted PDF:** rays are more likely to be sampled in the direction of the normal



[uniform sampling]

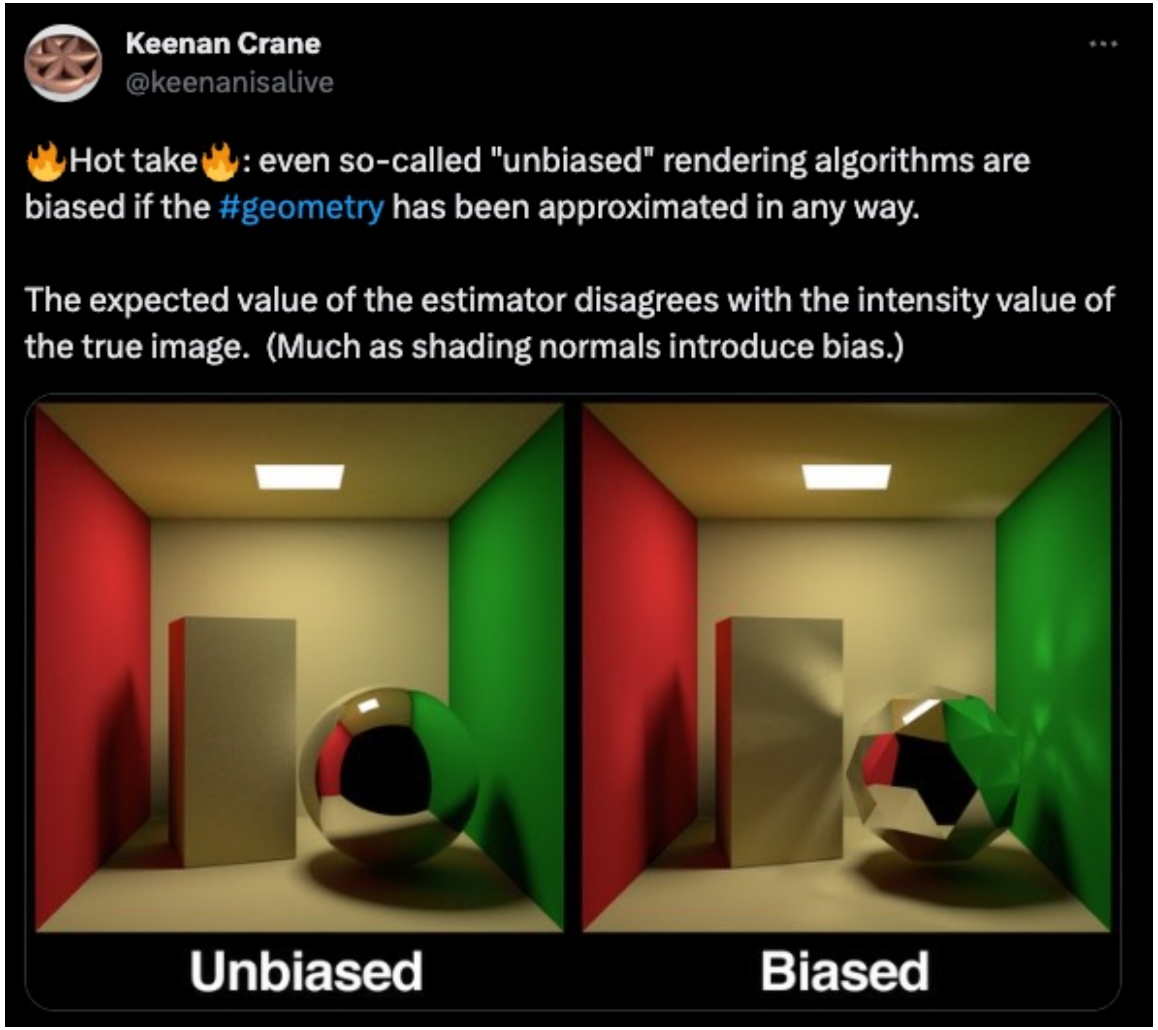


[cosine sampling]

But wait,
Isn’t taking non-uniform samples biased?

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

Biased vs. Unbiased Renderer

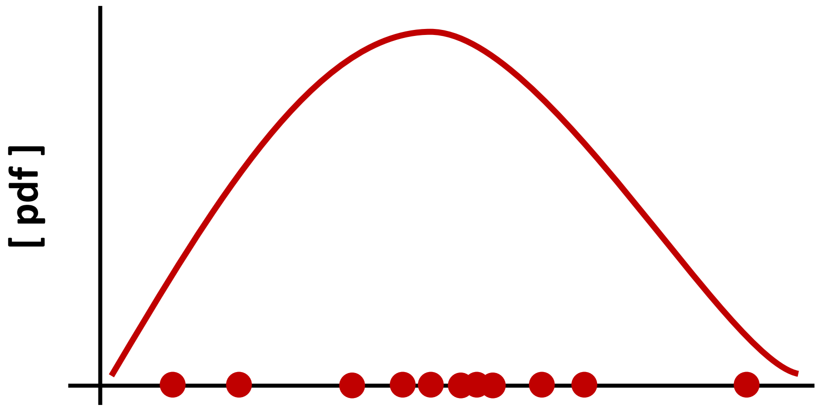


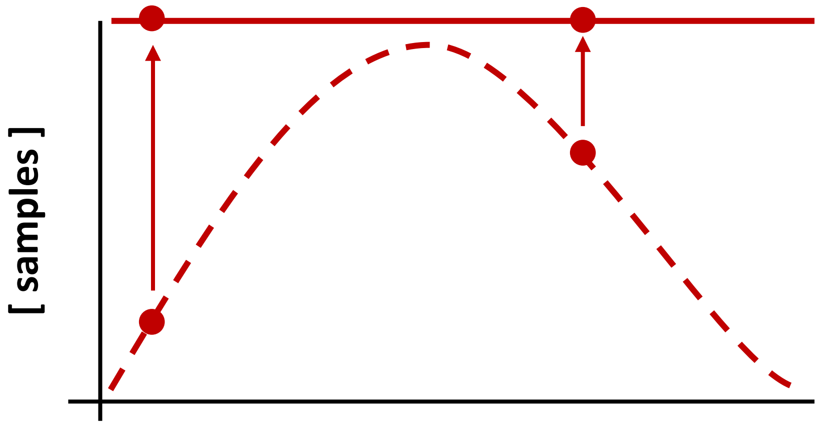
- An unbiased renderer tries to mimic the uniformity of real life
  - Does not introduce systematic bias into the rendering equation
  - Taking more samples will reduce error
  - Think of it as a ground truth
- A biased renderer will take shortcuts to make renders look better
  - Taking more samples 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**

Biased vs. Unbiased Example

- In a biased simulator, we draw samples proportional to the PDF
  - Much more samples drawn where PDF is high
  - Leads to under-sampling in low- PDF regions

- To turn a biased simulator unbiased, we can divide by the PDF of the sample
  - Samples with high PDFs are divided by a high value, **not increasing its contribution much**
  - Samples with a low PDF are divided by a low value, **increasing its contribution a lot**
    - Produces an unbiased sample set





The Monte Carlo Estimator

- Named **Monte Carlo** after the famous gambling location in Monaco because the method shares the same random characteristic as a roulette game

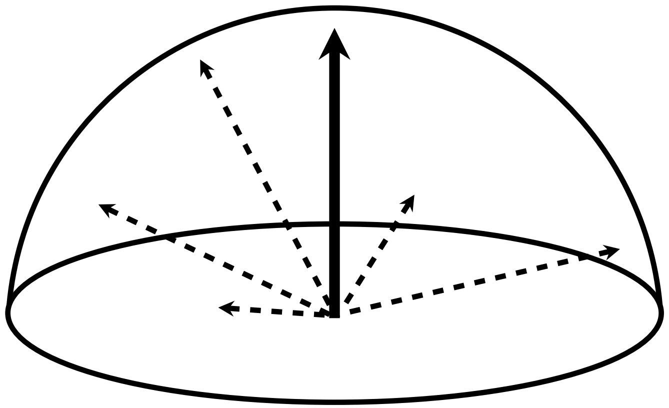
- Algorithm:
    - Sample a direction based on the PDF $p(w_j)$
    - Compute the incident radiance of the direction using the rendering equation
    - Divide by the PDF $p(w_j)$ so the sampler stays unbiased
    - Repeat, averaging the samples together

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

Monte Carlo Uniform Sampling

- Let $f(w)$ be the incident radiance [ignoring BRDF]
- Let $p(w)$ be the PDF of the sampled direction w

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



- Taking random samples leads to:

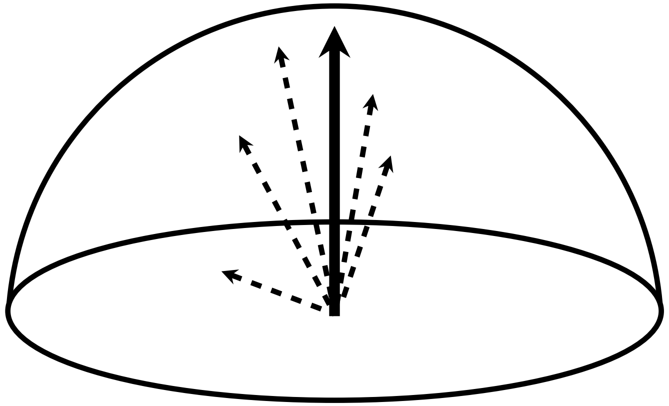
$$\int_\Omega f(\omega)\,\mathrm{d}\omega \approx \frac{1}{N}\sum_i^N \frac{f(\omega)}{p(\omega)} = \frac{1}{N}\sum_i^N \frac{L_i(\omega)\,\cos\theta}{1/2\pi} = \boxed{\frac{2\pi}{N}\sum_i^N L_i(\omega)\cos\theta}$$

- PDF is constant in all directions, just multiply by scalar 2π

Monte Carlo Cosine Sampling

- Let $f(w)$ be the incident radiance [ignoring BRDF]
- Let $p(w)$ be the PDF of the sampled direction w

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



- Taking random samples leads to:

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

- PDF removes the cosine term, we now get more radiance per sample!

How do we get a good sense of “how well” we did?

Variance

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

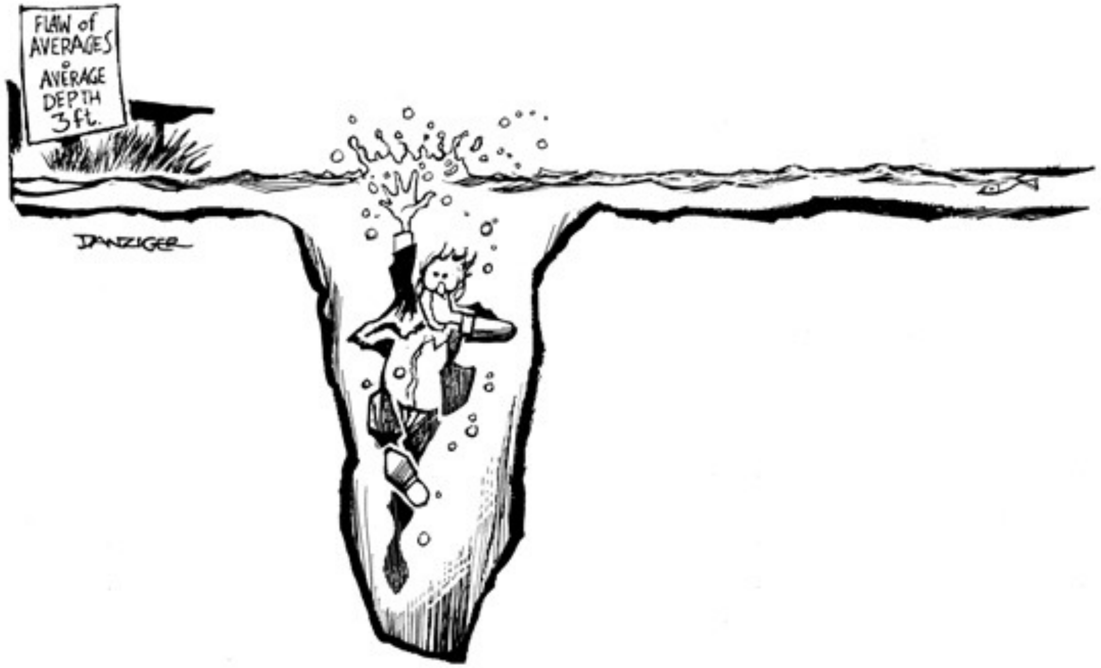
$$\mathrm{Var}(X) := E[(X - E[X])^2]$$

- Discrete:

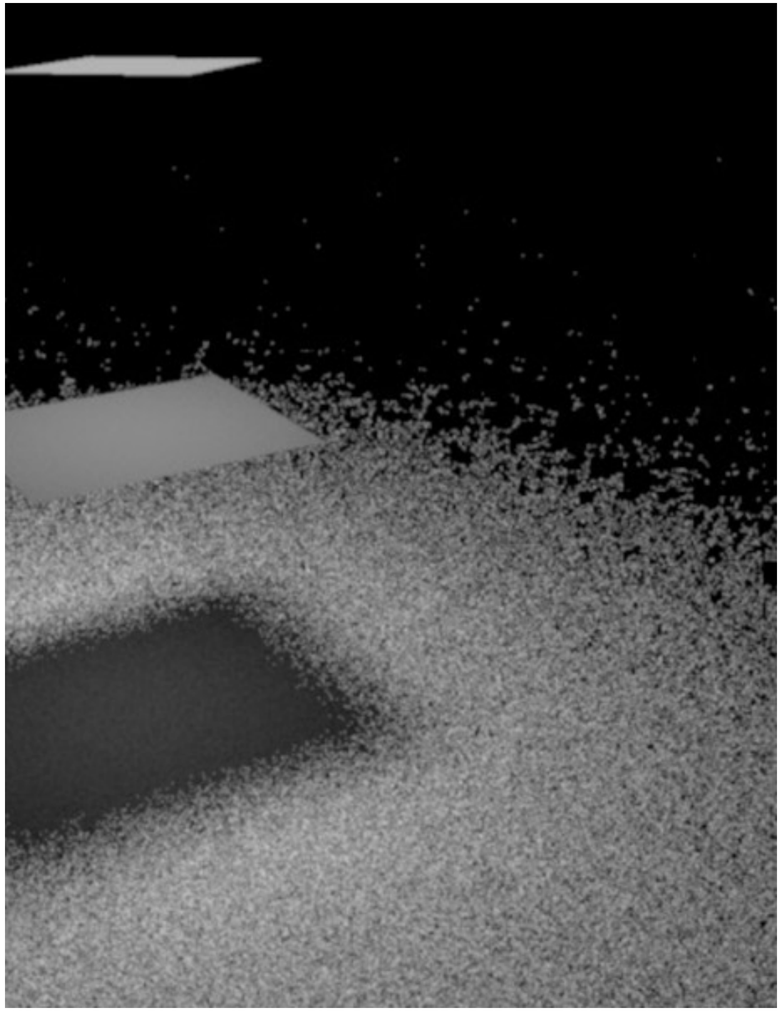
$$\sum_{i=1}^{n} p_i (x_i - \sum_j p_j x_j)^2$$

- Continuous:

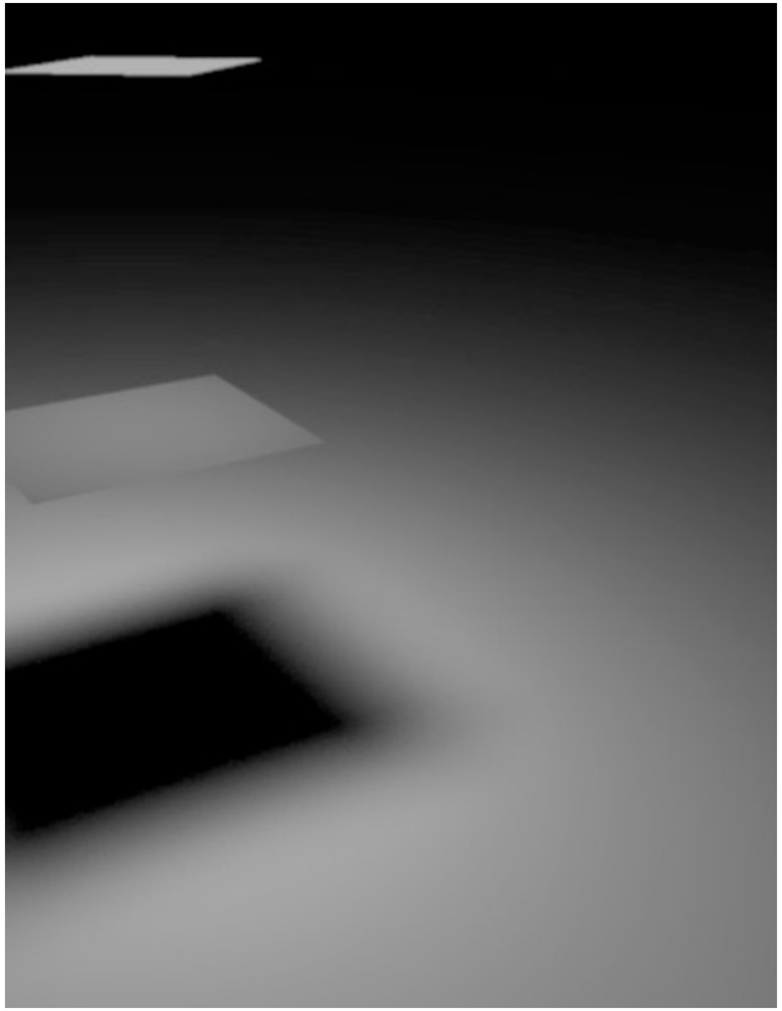
$$\int_\Omega p(x)(x - \int_\Omega y p(y)\ dy)^2\ dx$$



Variance In Rendering



[high variance]



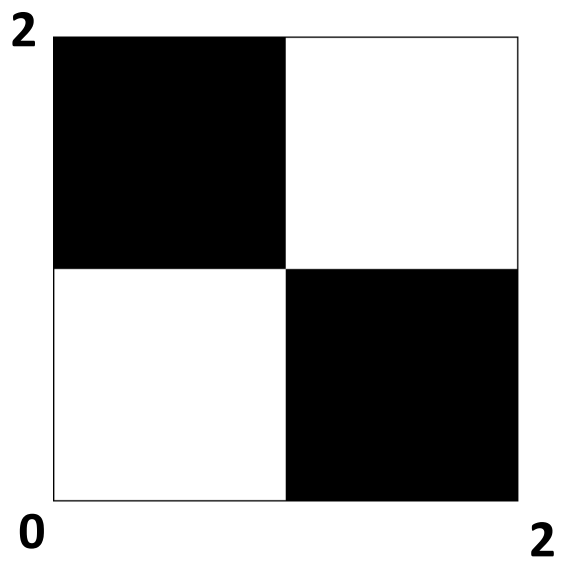
[low variance]

Variance Reduction Example

$$\Omega := [0,2] \times [0,2]$$

$$f(x,y) := \begin{cases} 1 & \lfloor x \rfloor + \lfloor y \rfloor \text{ is even,} \\ 0 & \text{otherwise} \end{cases}$$

$$I := \int_\Omega f(x,y)\ dxdy$$



- What's the expected value of the integrand?
  - Just by inspection: 1/2 (half black, half white)
- What's the variance?
  - $(1/2)(0-1/2)^2 + (1/2)(1-1/2)^2 = (1/2)(1/4) + (1/2)(1/4) = ¼$
- How do we reduce the variance?

Trick question!
You can't reduce the variance of an integrand.
Can only reduce variance of an estimator.

Bias & Consistency

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

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

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

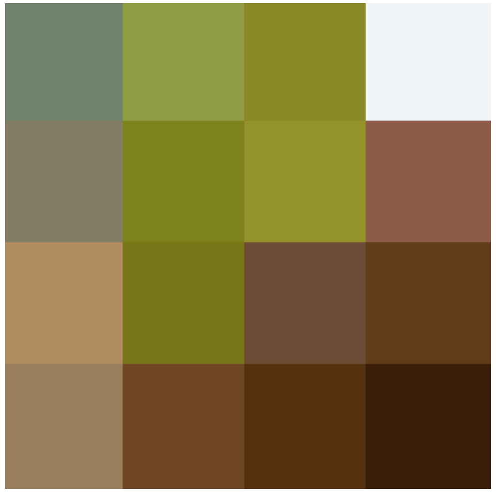
- **consistent != unbiased**

[biased] [unbiased]

Consistent Or Unbiased?

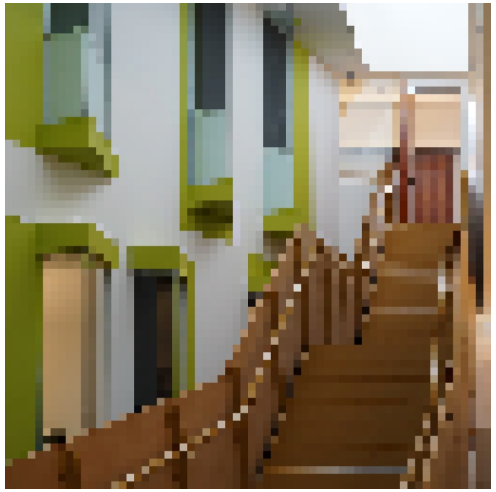
- Estimator for the integral over an image:
  - Take n = m x m samples at fixed grid points
  - Sum the contributions of each box
  - Let m go to ∞

- Is the estimator:
  - Consistent?
  - Unbiased?



[m = 4]

[m = 16]



[m = 64]

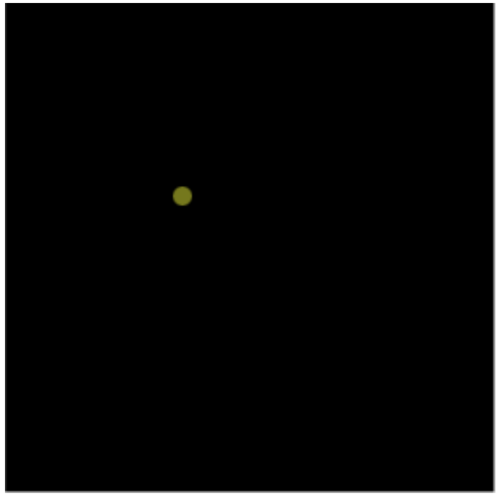
[m = ∞]

Consistent Or Unbiased?

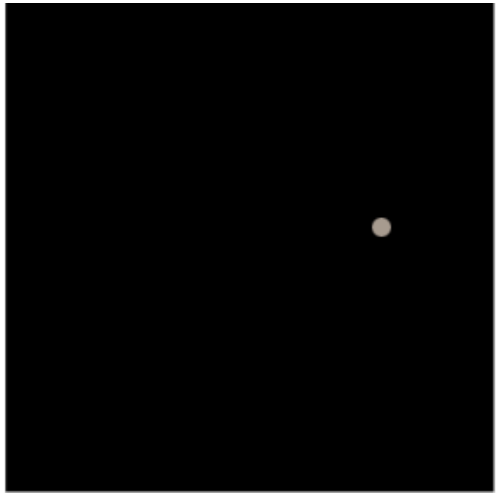
- Estimator for the integral over an image:
  - Take only a single random sample of the image (n=1)
  - Multiply it by the image area
  - Use this value as my estimate

- Is the estimator:
  - Consistent?
  - Unbiased?

- What if I let my estimator go to ∞

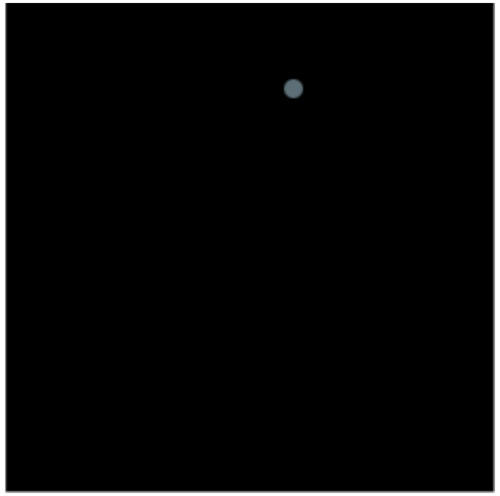


[m = 1]



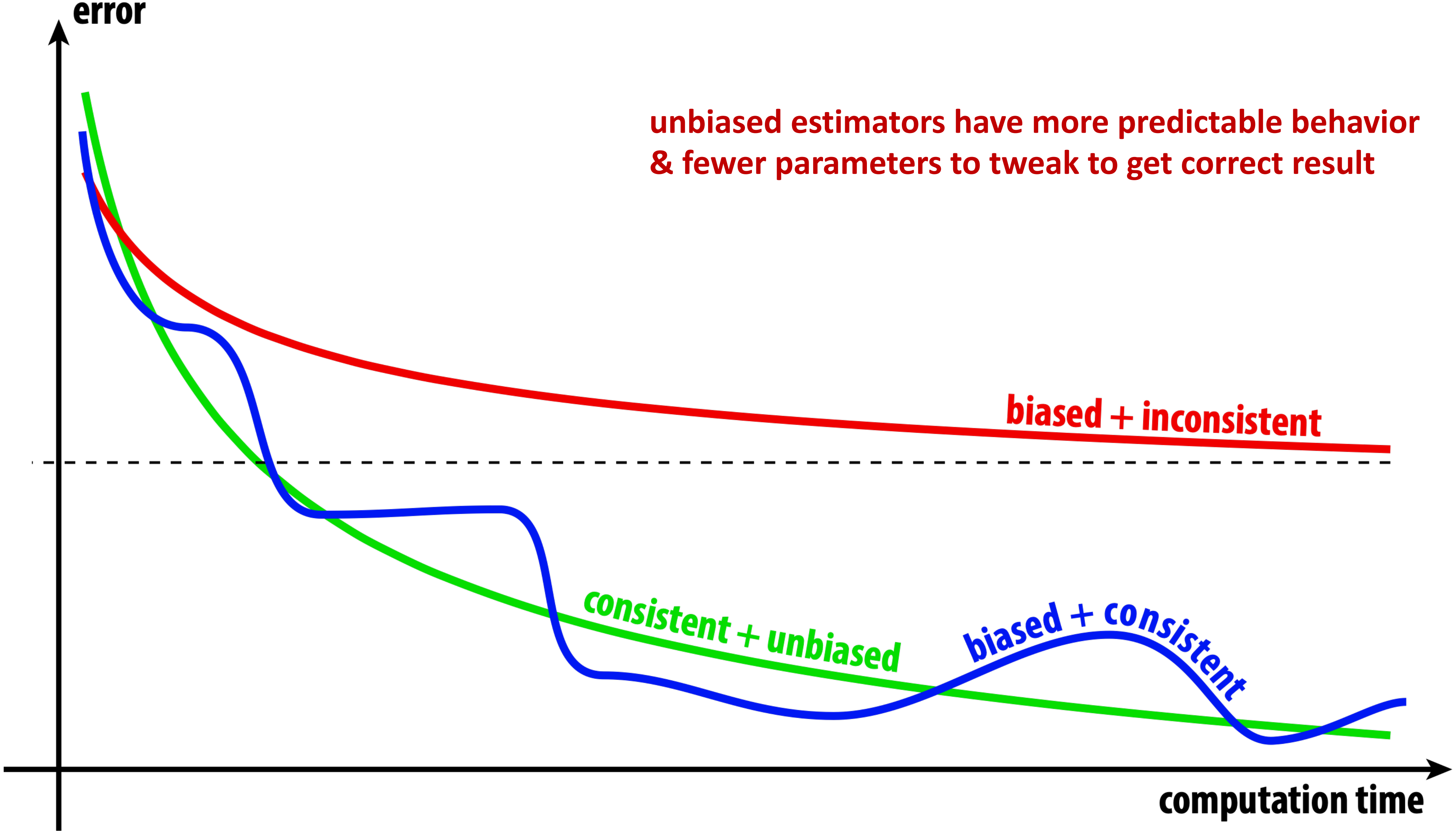
[m = 1]

[m = 1]



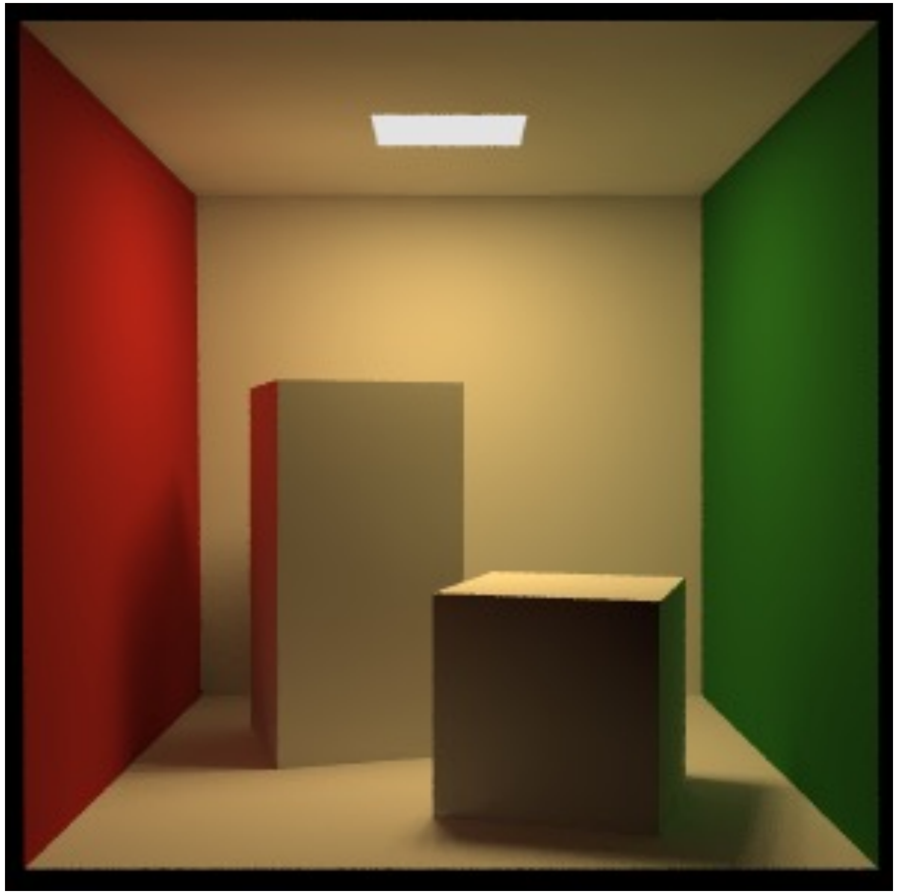
[m = 1]

Consistent Or Unbiased?



What is my true image?

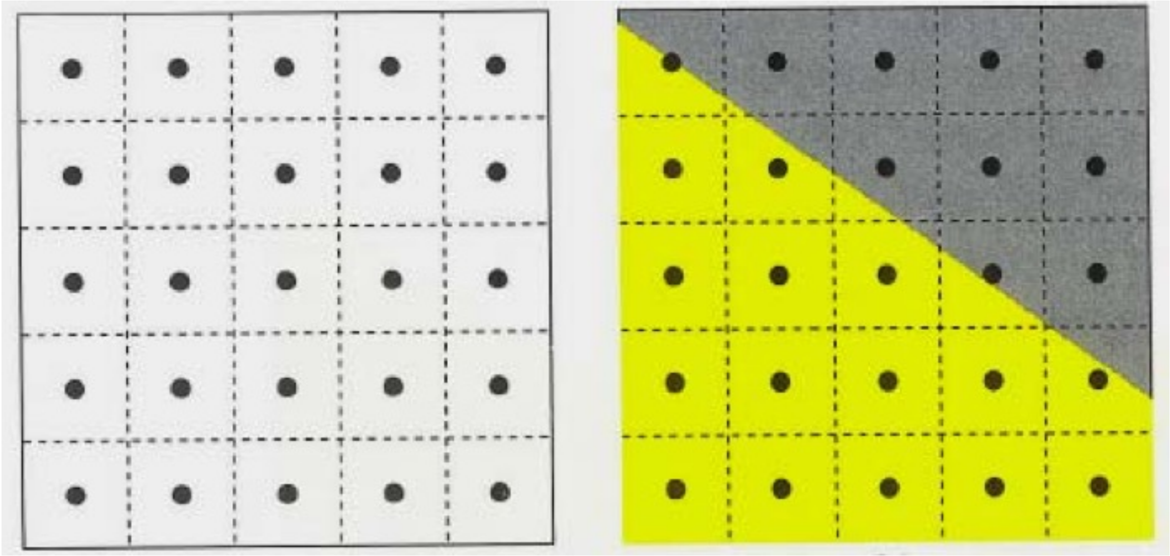
The Cornell Box

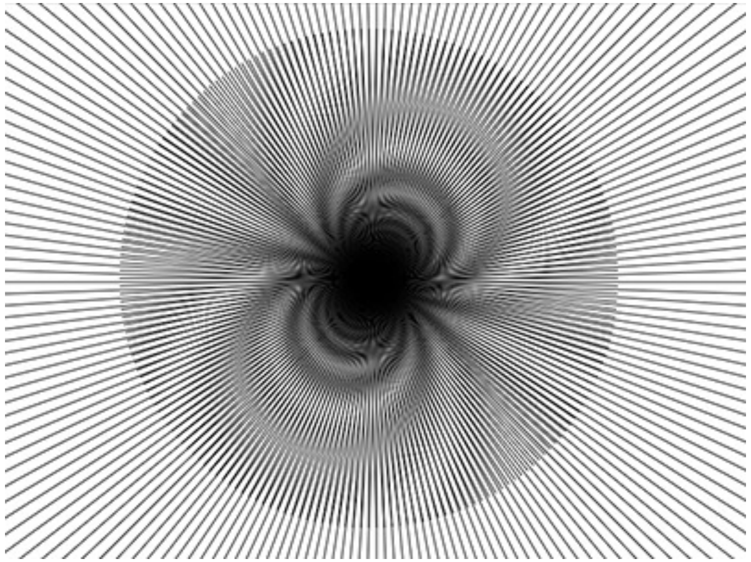


How do we take good samples?

Uniform Sampling

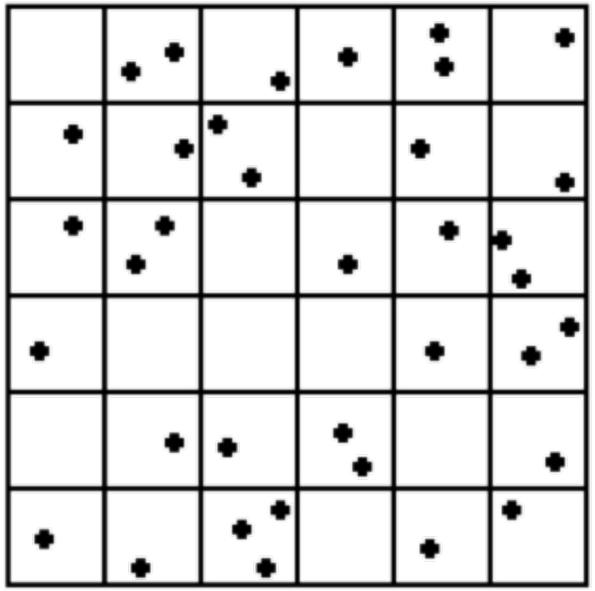
- Place samples uniformly apart in grid fashion
  - [+] Easy to compute
  - [-] We still have jagged edges, just at higher resolutions
  - [-] More samples needed
  - [-] Does not fix moiré pattern

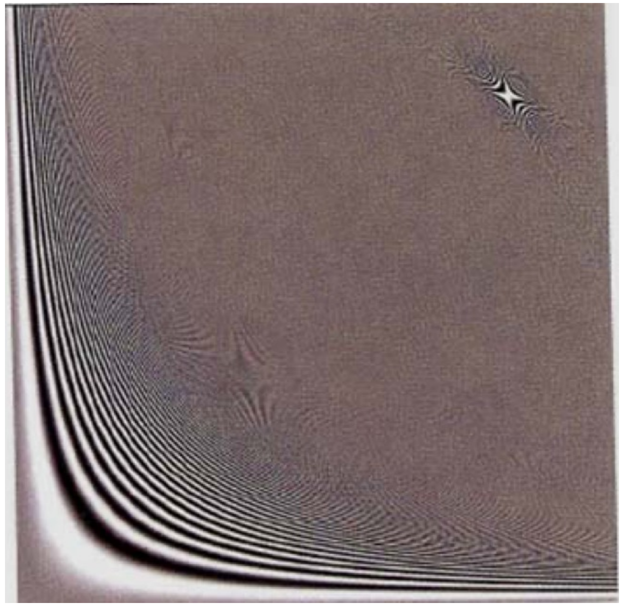


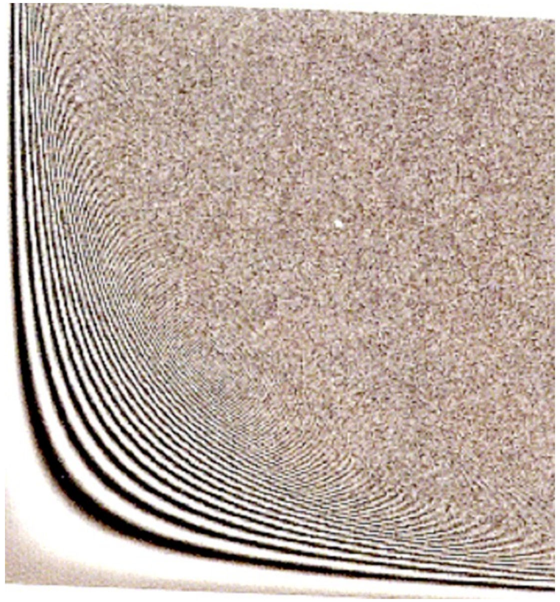


Random Sampling

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

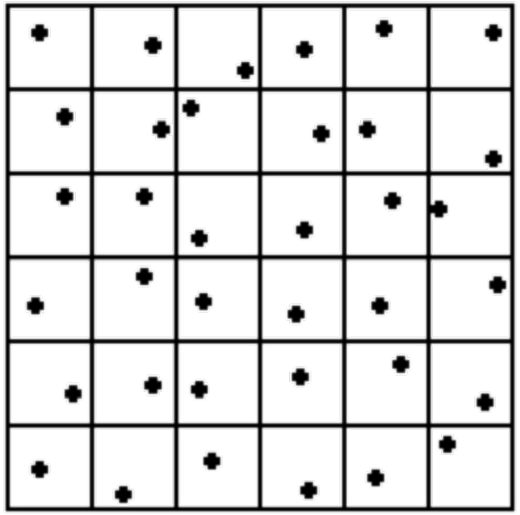


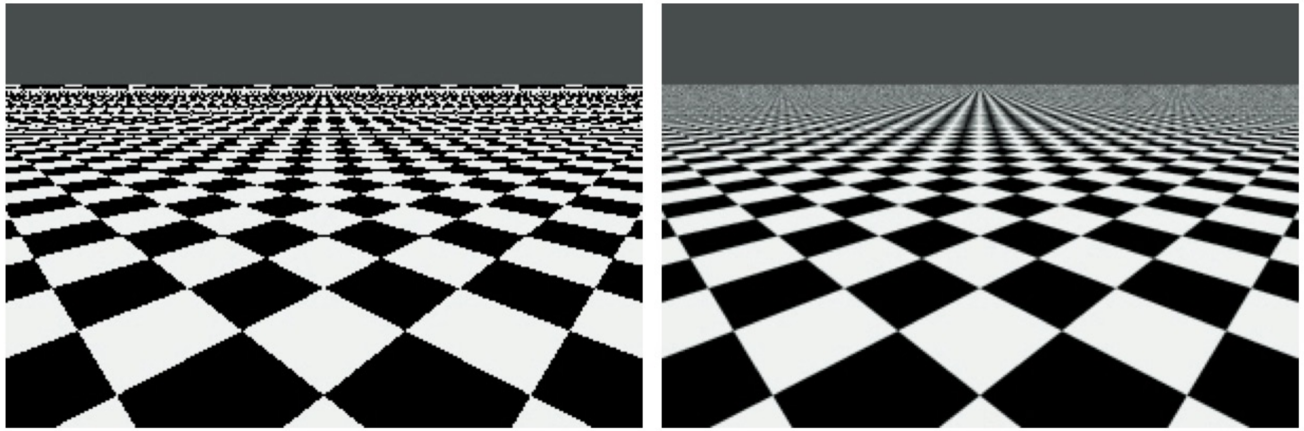




Jittered Sampling

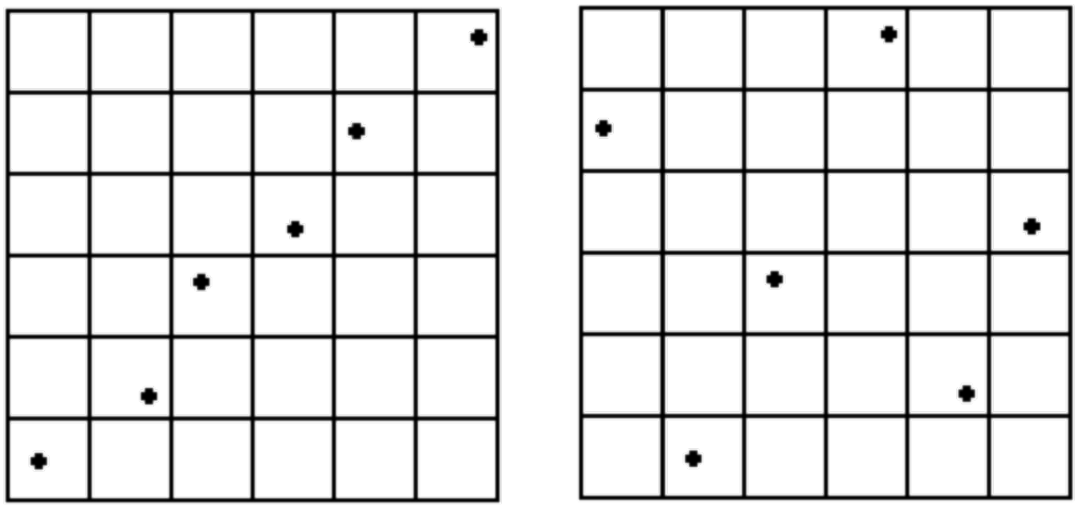
- Divide into N x N grid, place a sample randomly per grid cell
  - [+] Easy to compute
  - [+] A more constrained version of random sampling
  - [-] Ensures distance between samples, but not enough!





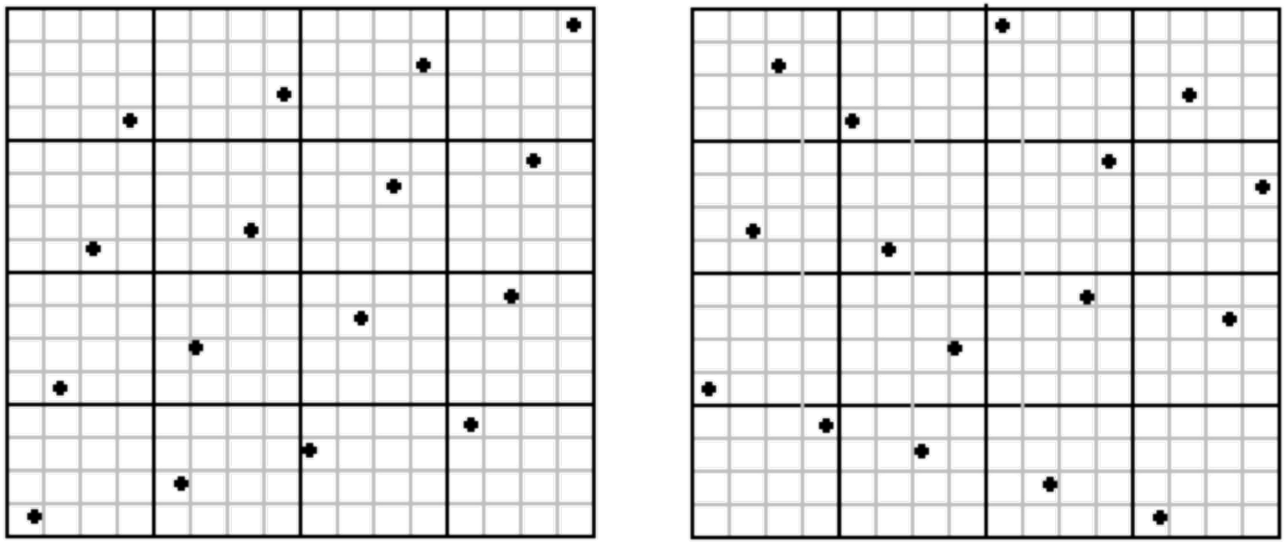
N-Rooks Sampling

- All samples start on the diagonal, randomly shuffle (x, y) coordinates until rooks condition satisfied
  - [+] 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



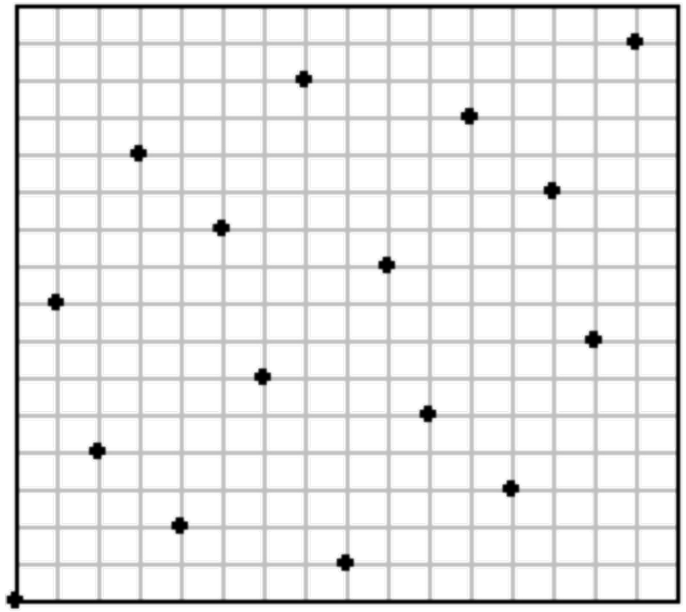
Hammersley Sampling

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

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

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

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



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*

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

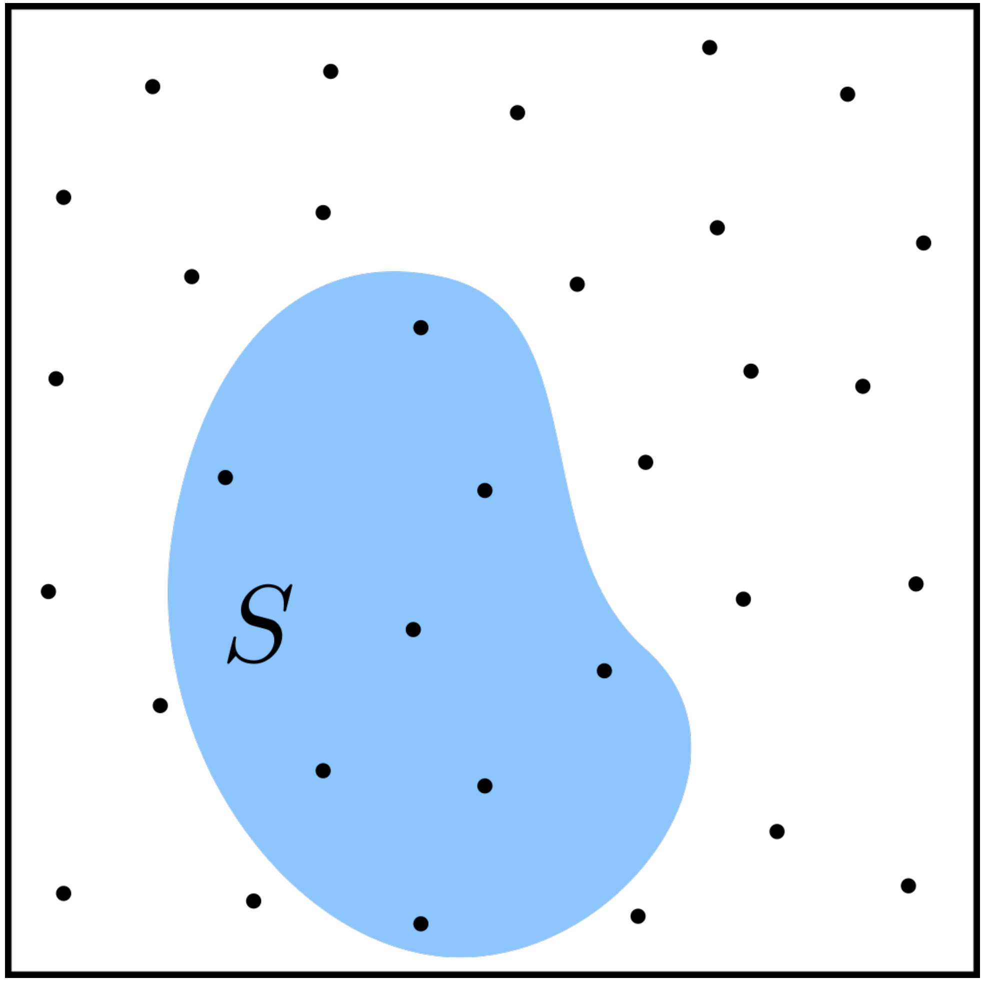
area of *S*

total # of samples in *X*

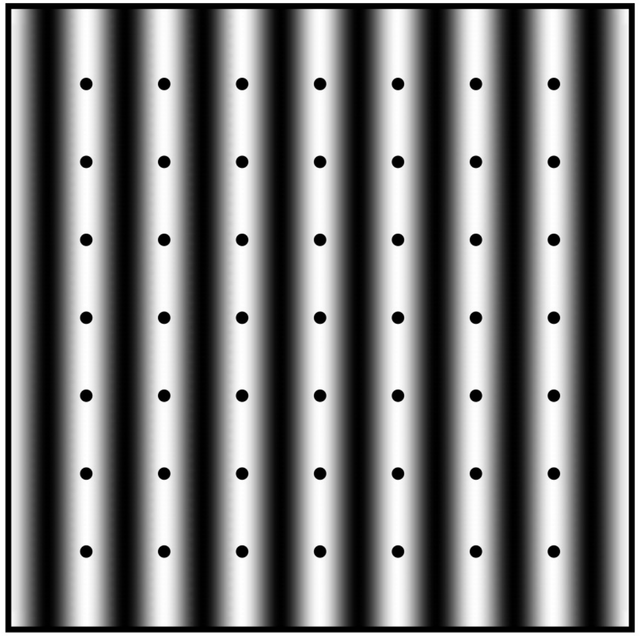
$$D(X) := \max_{S \in F} d_S(X)$$

overall discrepancy

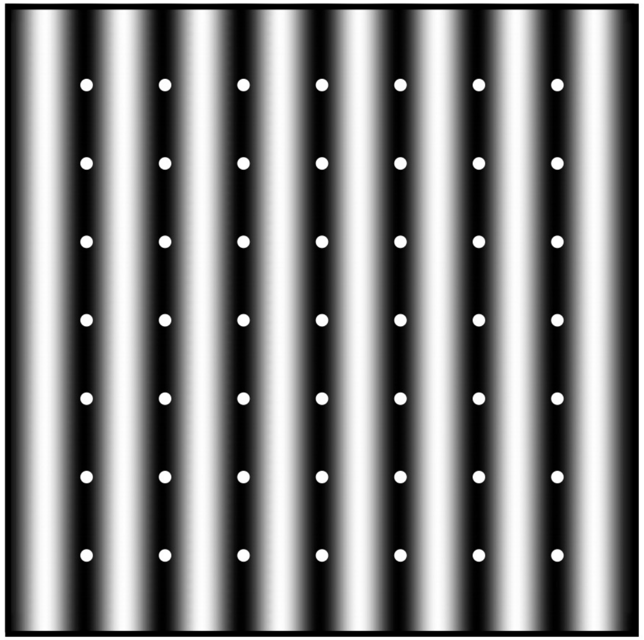
some family of regions *S* (box, disk, etc...)



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

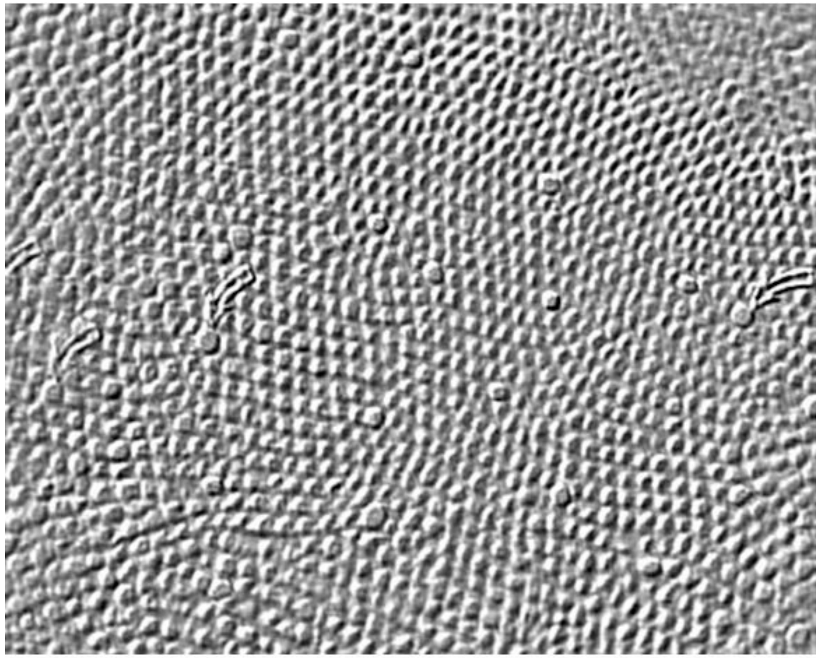
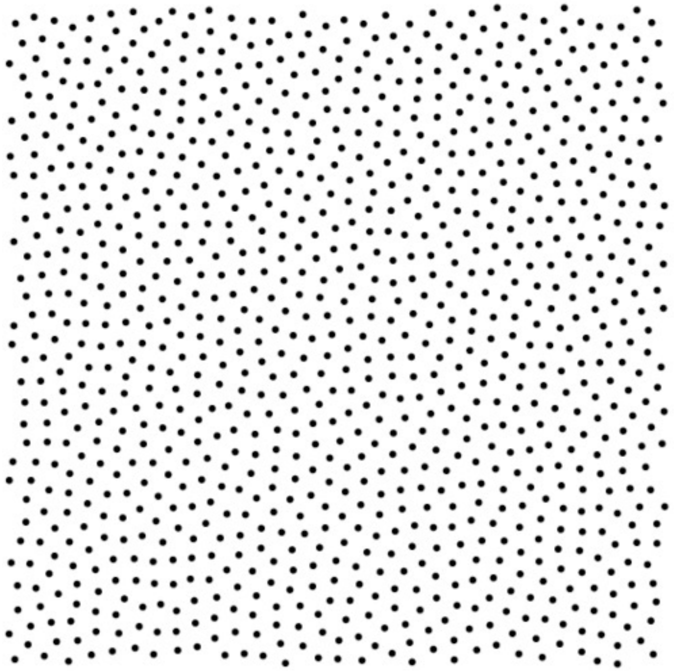


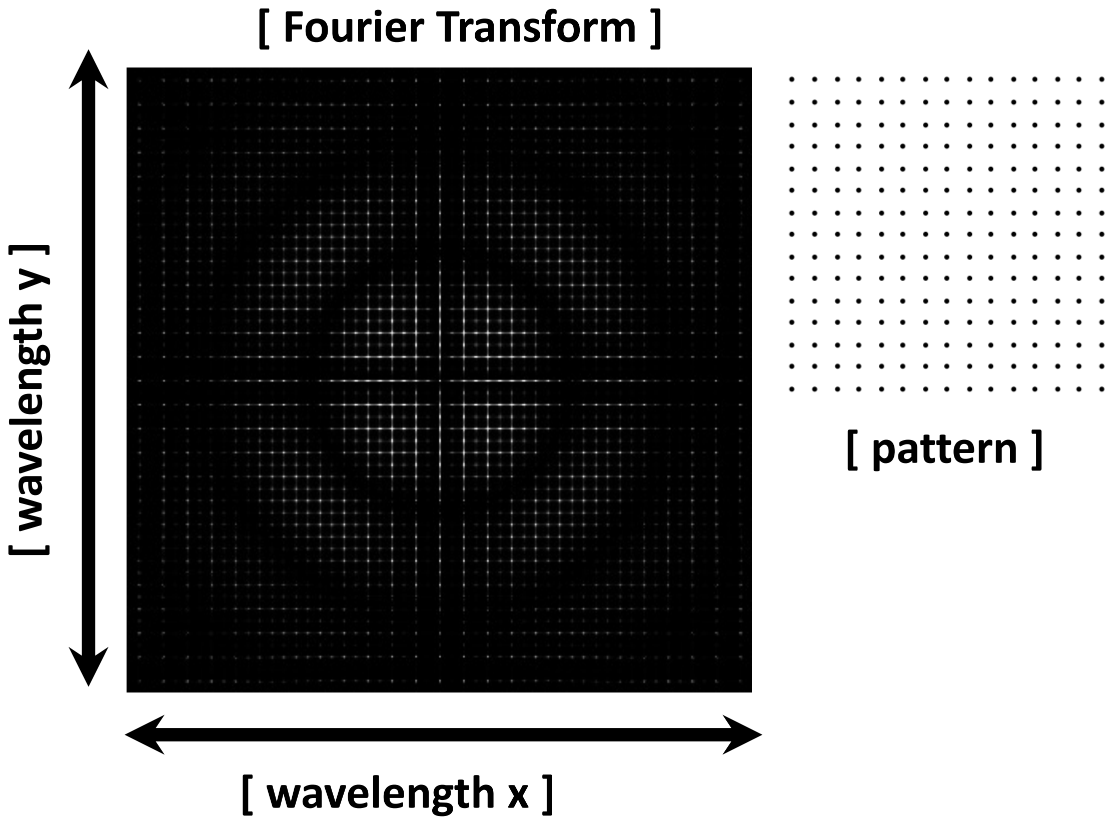
Fig. 13. Tangential section through the human fovea. Larger cones (arrows) are blue cones. From Ahnelt et al. 1987.

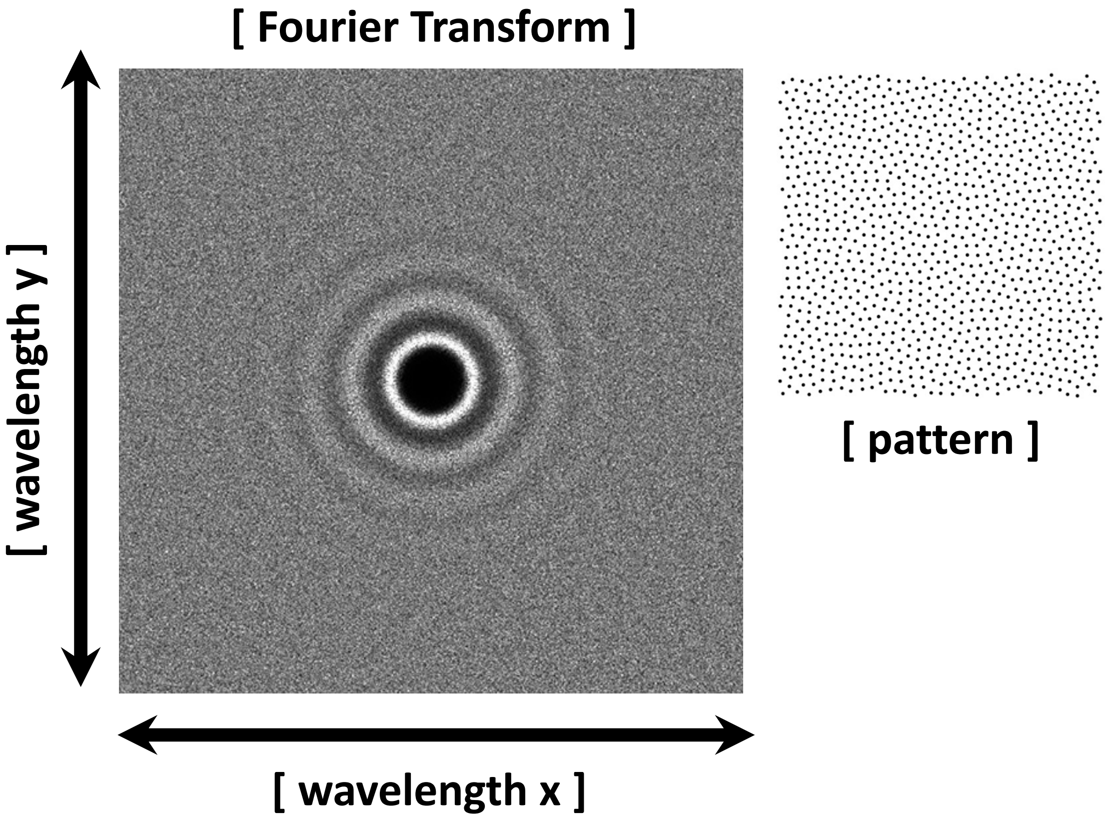


[“blue noise”]

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

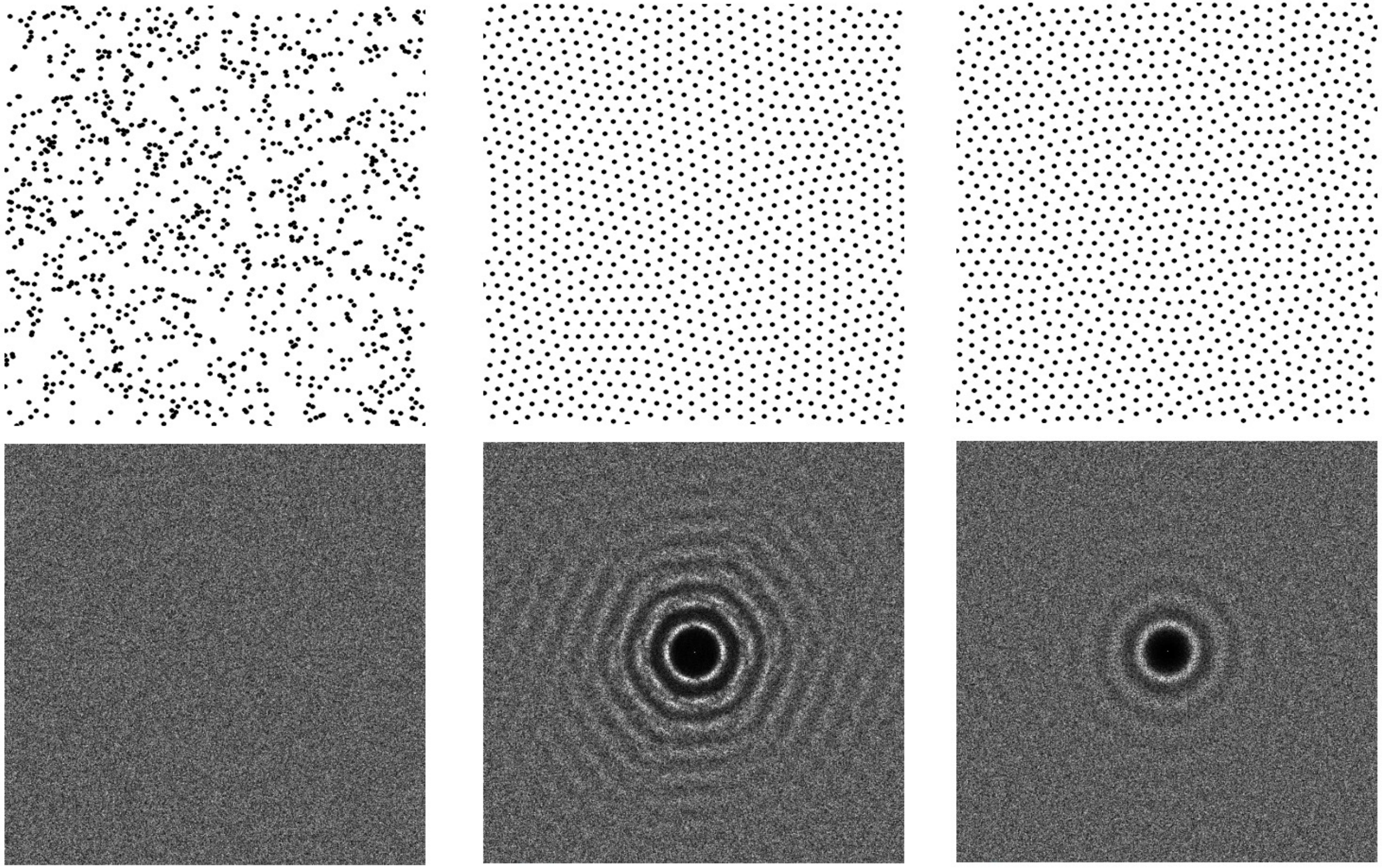
Blue Noise Fourier Transform





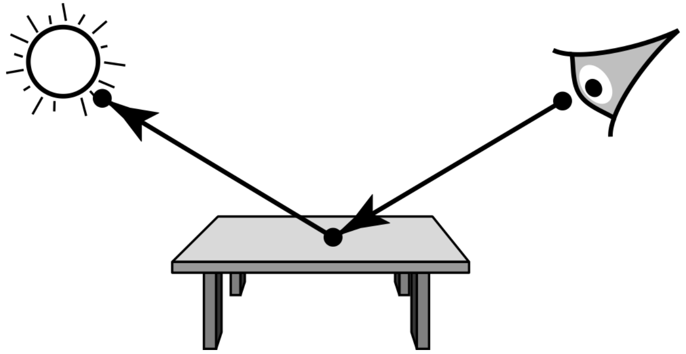
- Regular pattern has "spikes" at regular intervals
- Blue noise is spread evenly over all frequencies in all directions
  - Bright center "ring" corresponds to sample spacing

Blue Noise Fourier Transform



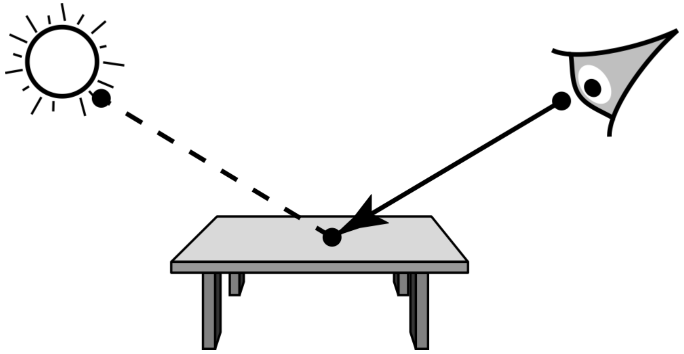
- ~~Monte-Carlo Sampling~~
- ~~Biased vs Unbiased Estimators~~
- Physically-Based Rendering Methods

Previous Methods



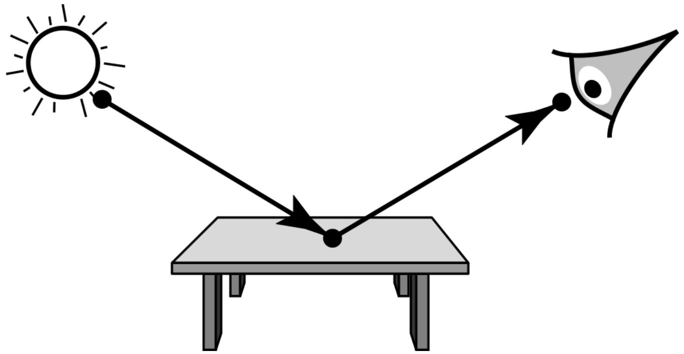
[backward path tracing]

fails: cannot intersect point lights



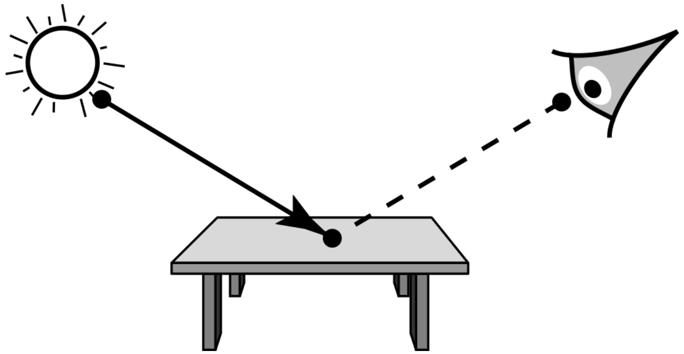
[backward path tracing + connect to light]

works: reaches point lights



[forward path tracing]

fails: cannot intersect pinhole camera

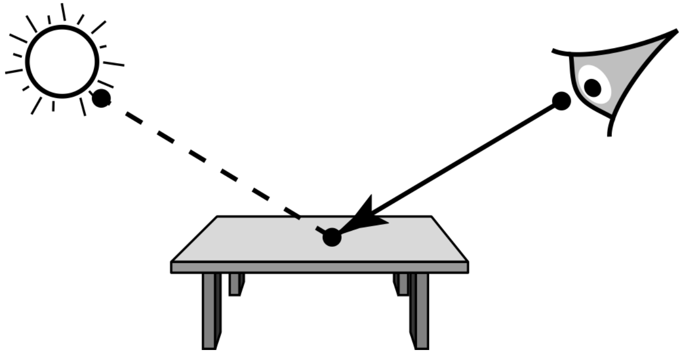


[forward path tracing + connect to camera]

works: reaches pinhole camera

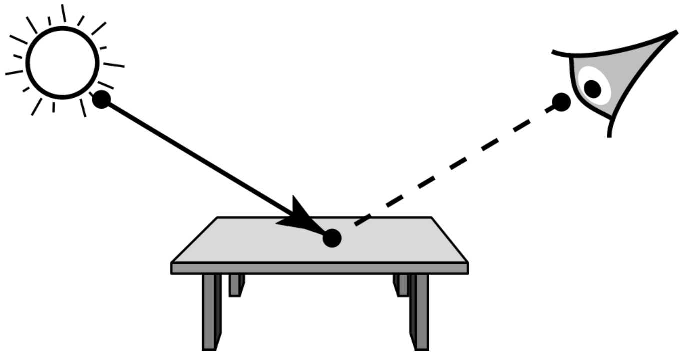
Path Tracing Can Be Biased

- 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



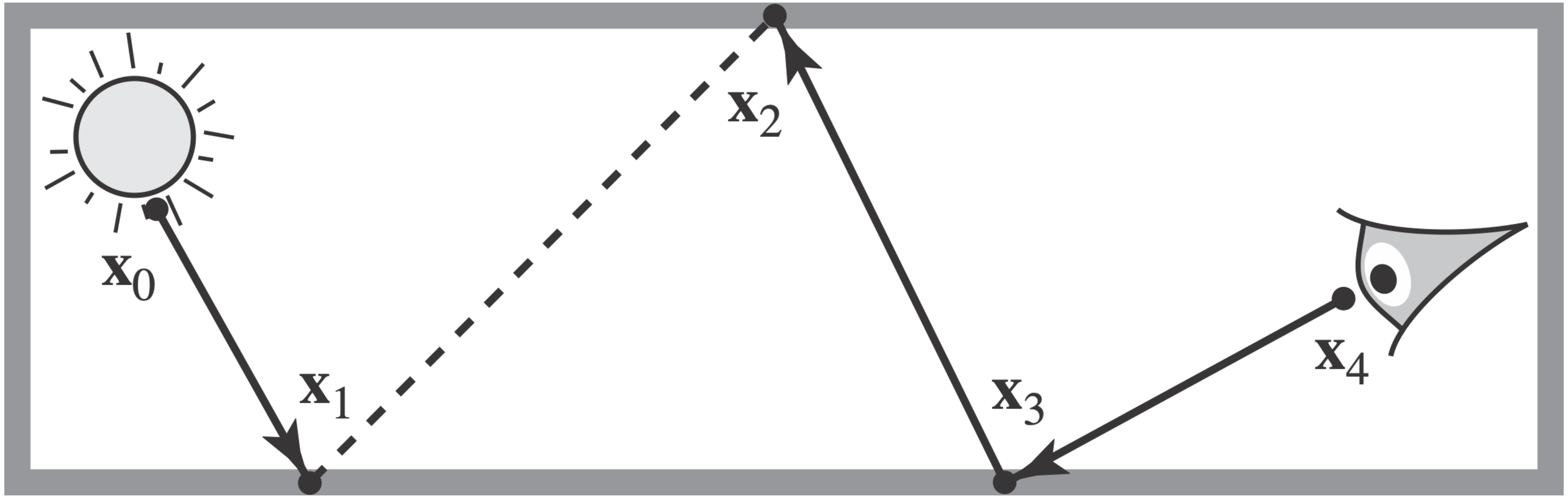
[forward path tracing + connect to camera]

works: reaches pinhole camera

Bidirectional Path Tracing

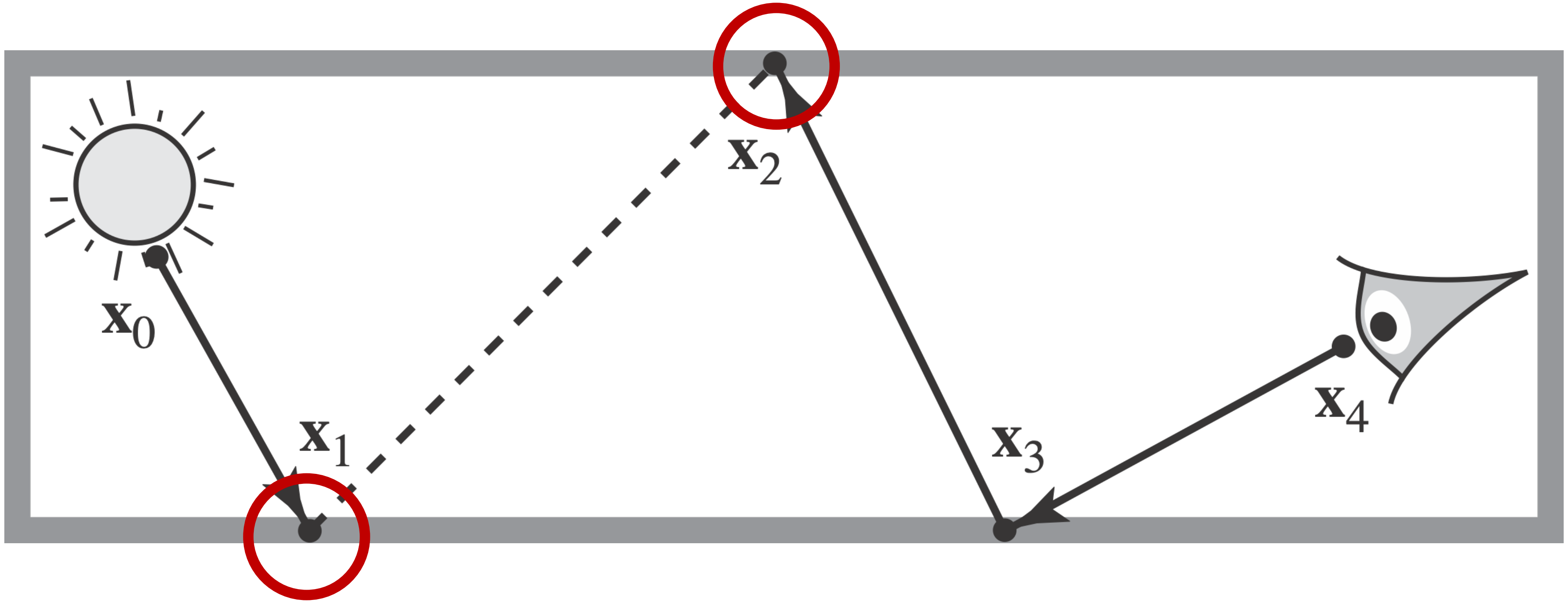
- If path tracing is so great, why not do it **twice**?
  - Main idea of bidirectional!
- Trace a ray from the camera into the scene
- Trace a ray from the light into the scene
  - Connect the rays at the end

- 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



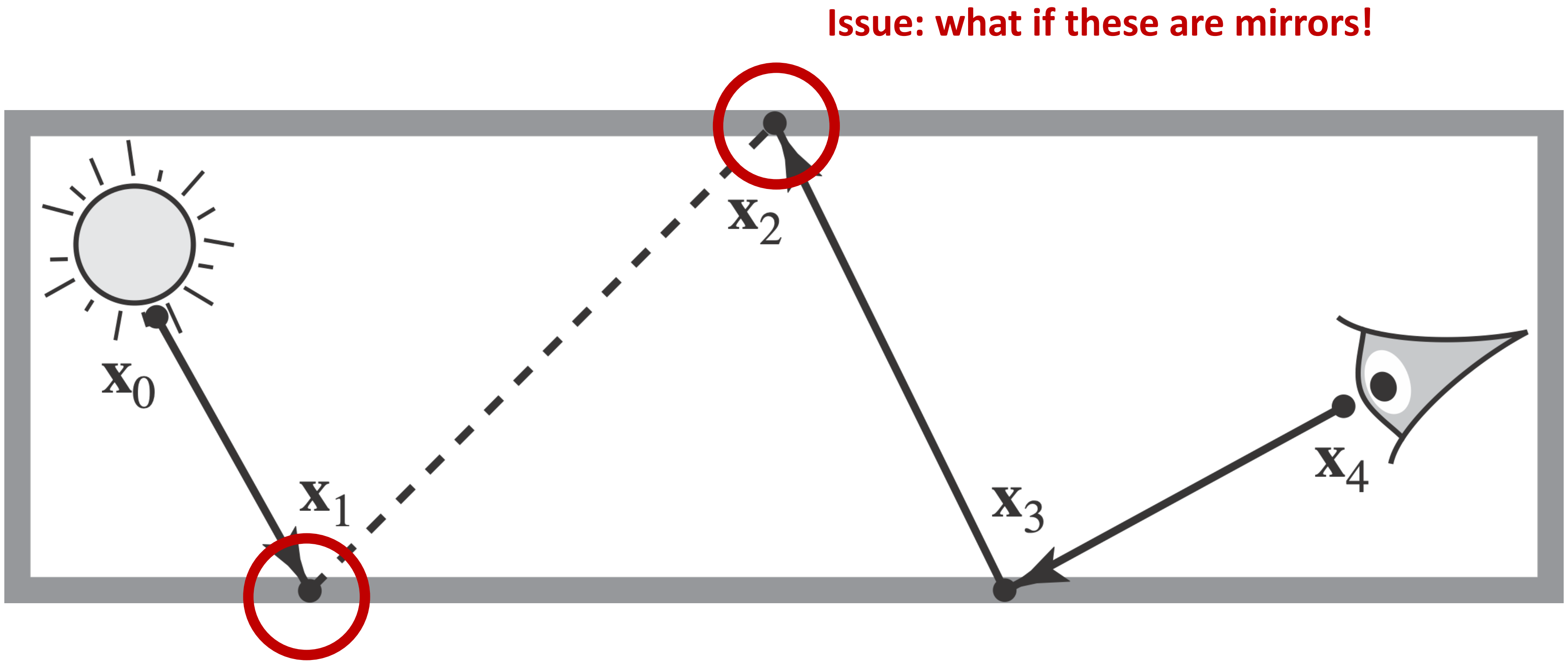
Bidirectional Path Tracing

Issue: what if these are mirrors!

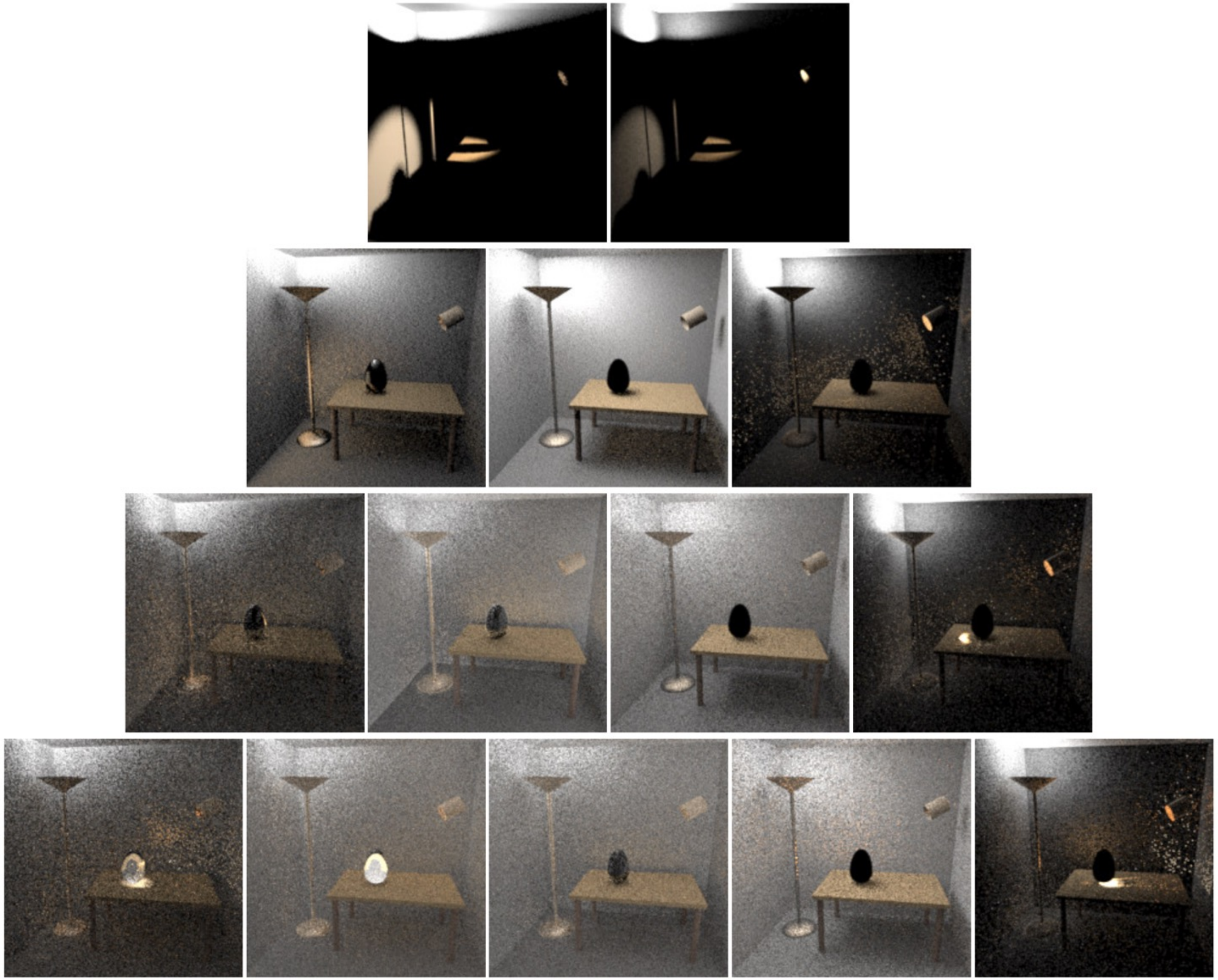


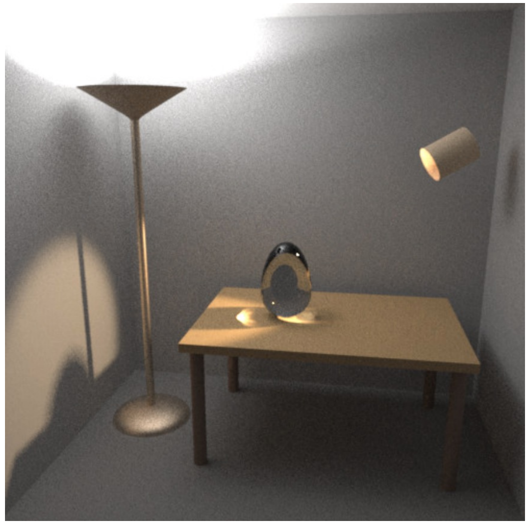
Bidirectional Path Tracing

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



Bidirectional Path Tracing

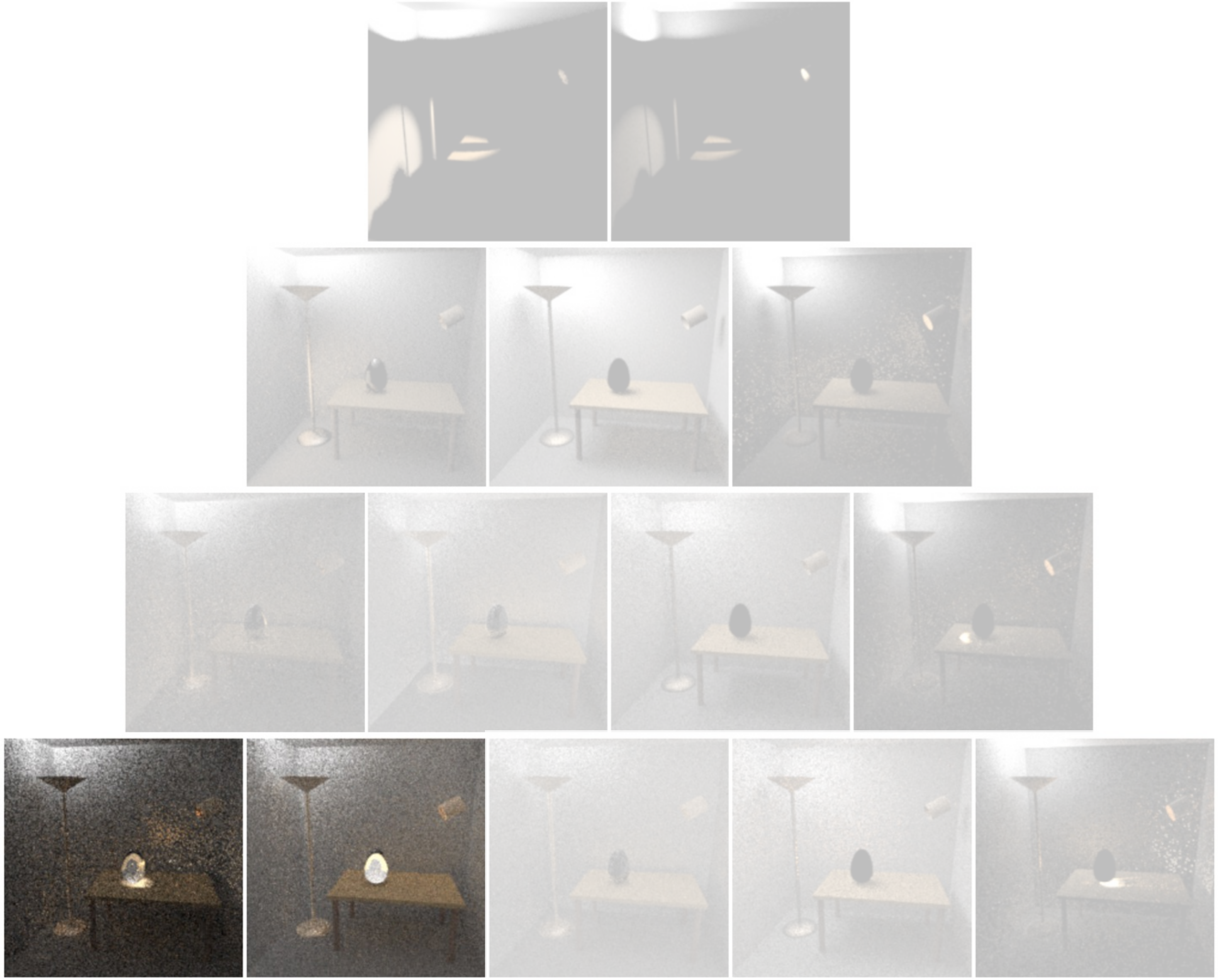


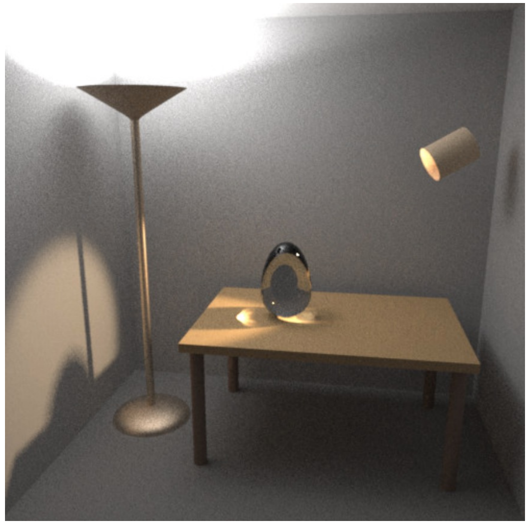


[final image]

- 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

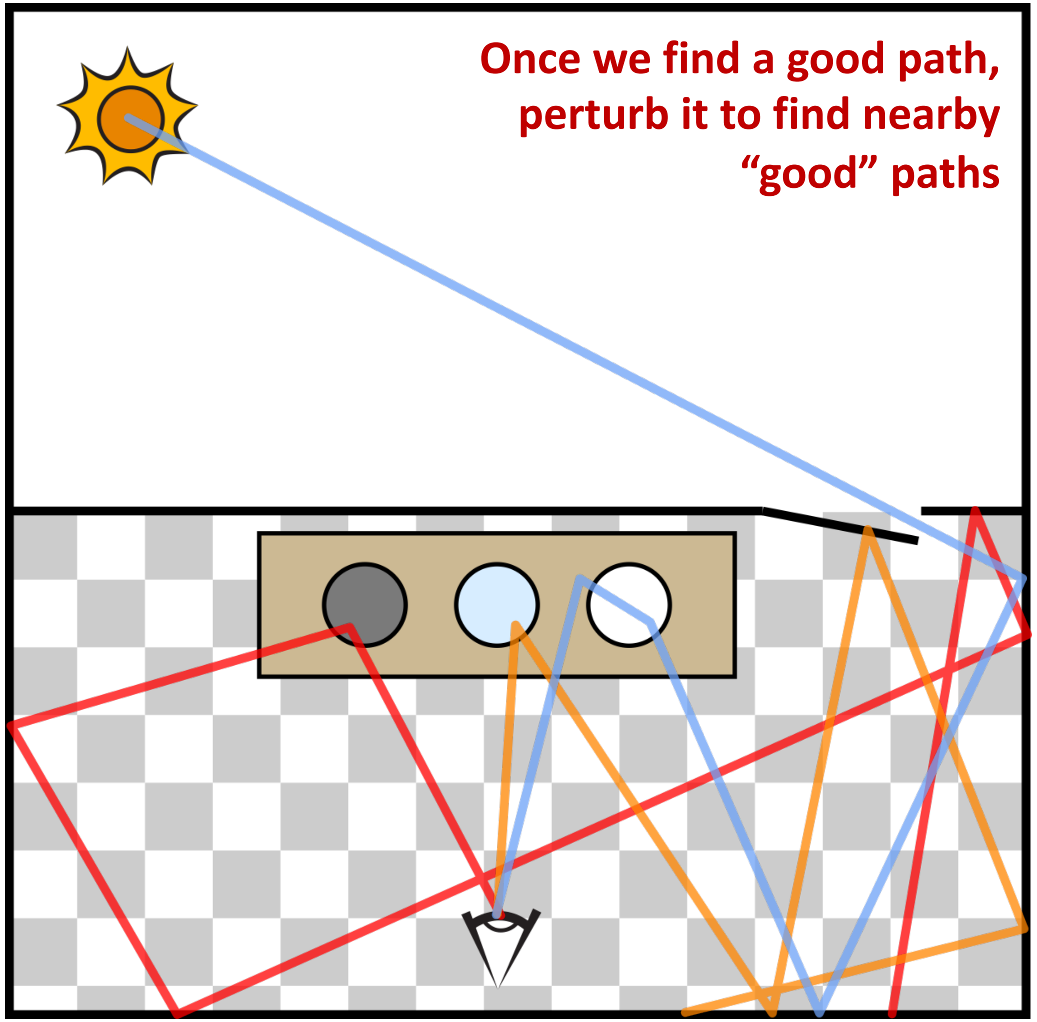


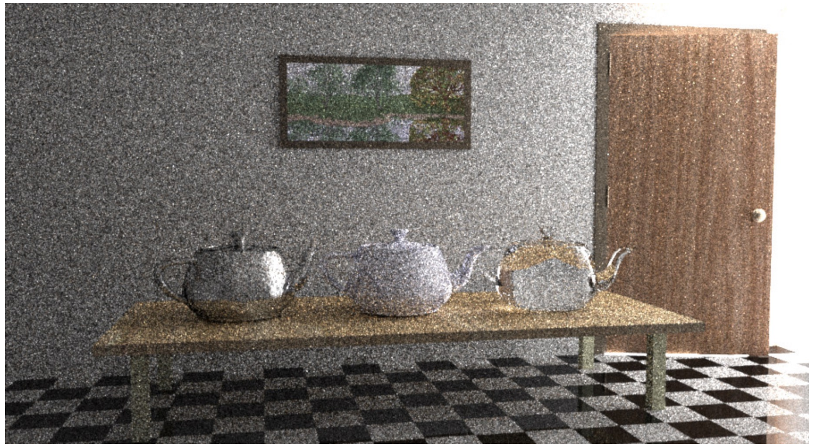


[final image]

- 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





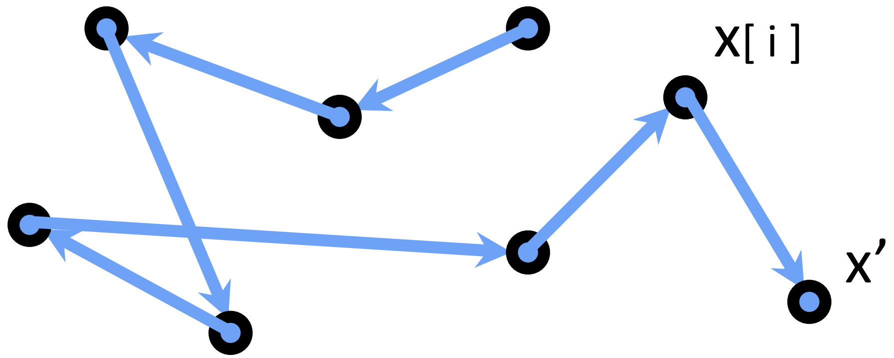
[Bidirectional Path Tracing]

[Metropolis Light Transport]

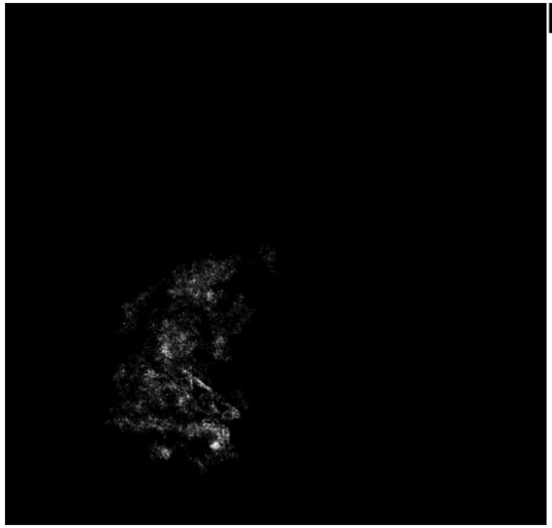
Metropolis Hasting Algorithm

- Recall in previous slide: *"Once we find a good path, perturb it to find nearby 'good' paths"*
- **Algorithm:** take random walk of dependent samples
  - If we are in an area where sampling yields high values, stay in or near the area, otherwise move far away
- If careful, sample distribution will 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') >> f(x[i]), then a is large
// and we increase chances of moving to x'
// if f(x') << 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]

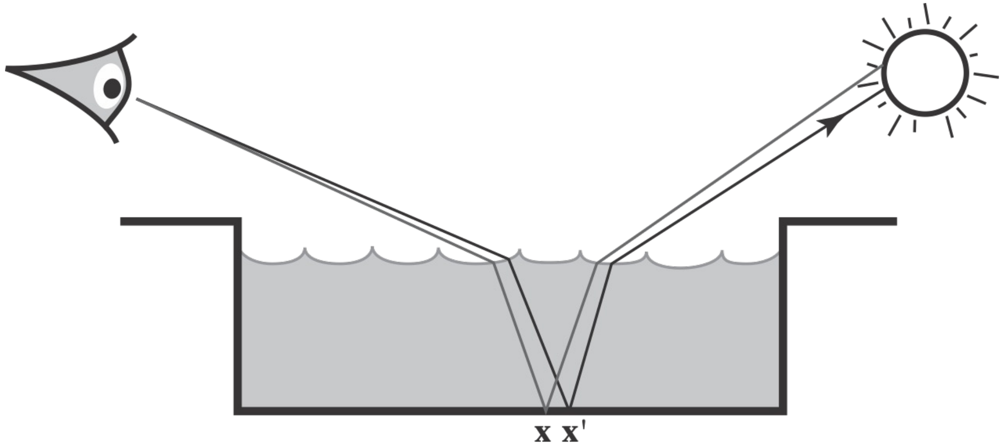
[long walk]

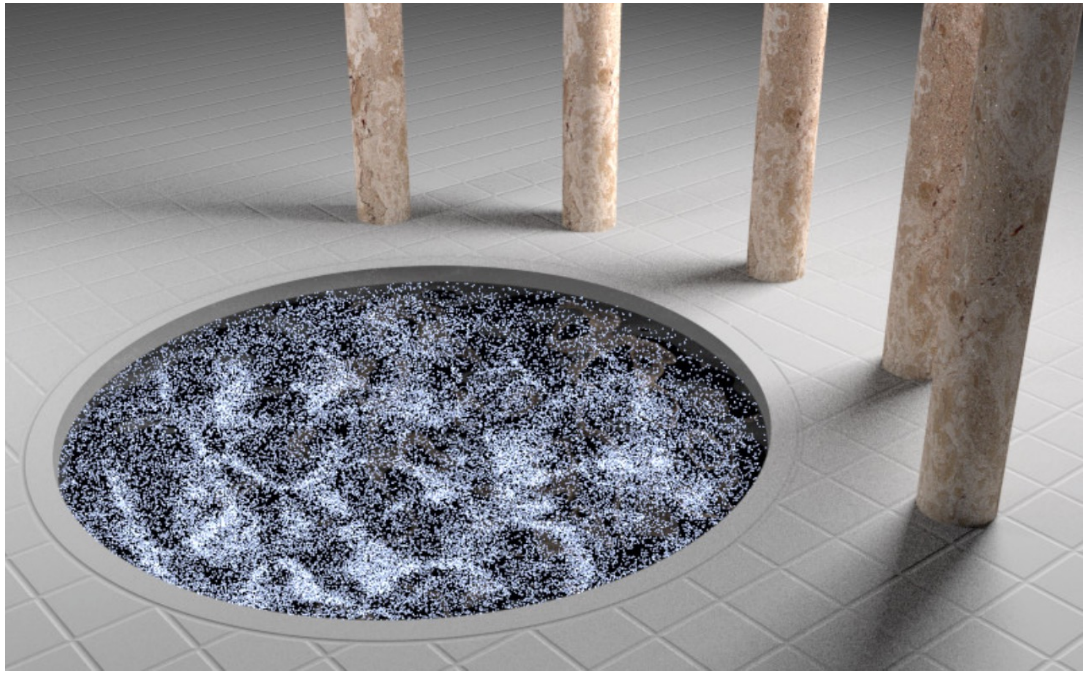
[original image]

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

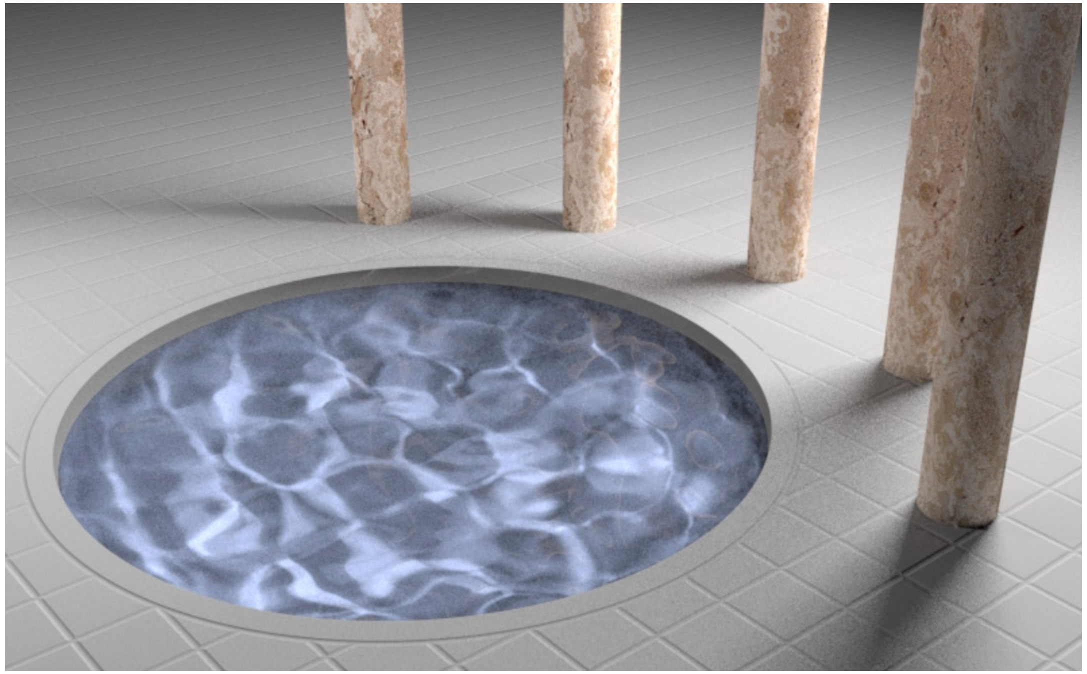
Metropolis Light Transport

- **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





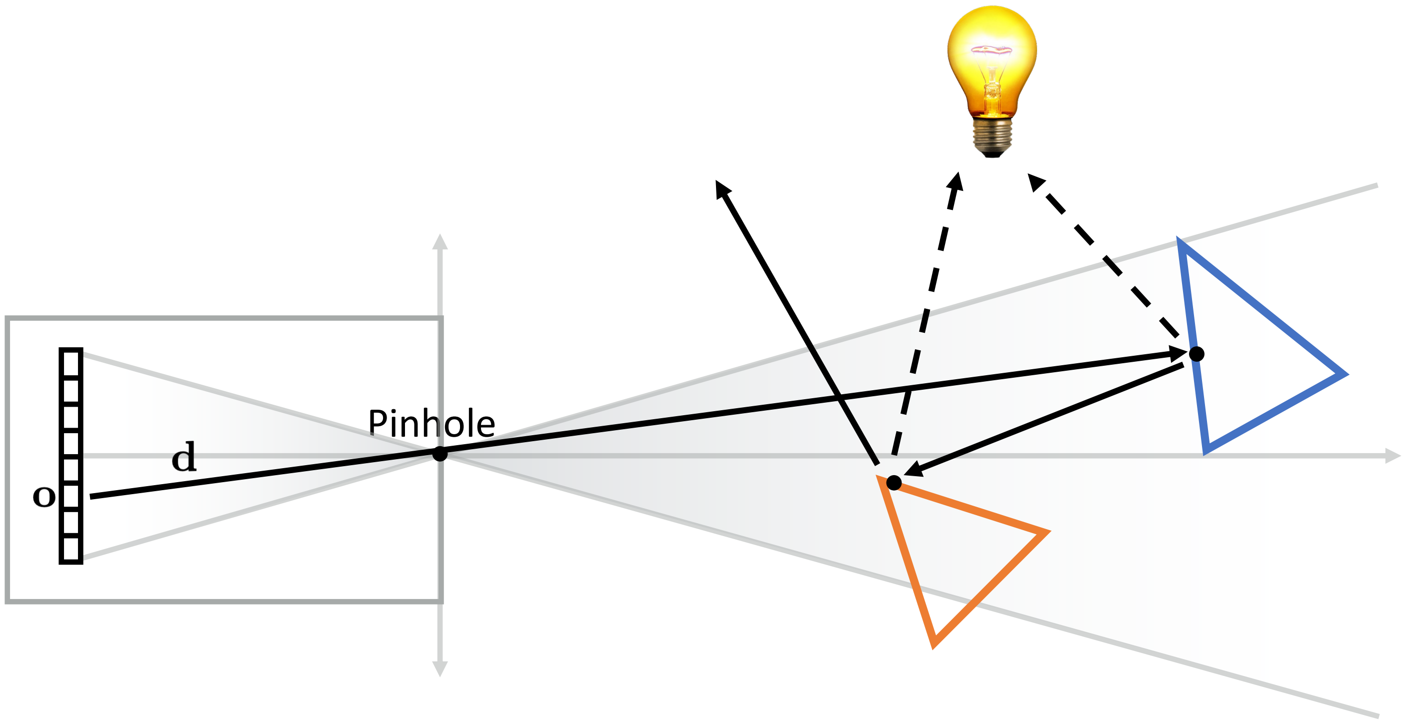
[Path Tracing]



[Metropolis Light Transport]

Next Event Estimation (NEE)

- Extension to Backwards Path Tracing
  - At each ray bounce, trace two new rays:
    - A ray generated by the BRDF
    - A ray towards the light
  - Integrate samples together
  - Can only be done for diffuse surfaces!
- No need to trace ray to light source explicitly on termination
  - Taken care of at each ray bounce
- **Issue:** averaging two high-variance methods leads to higher variance!

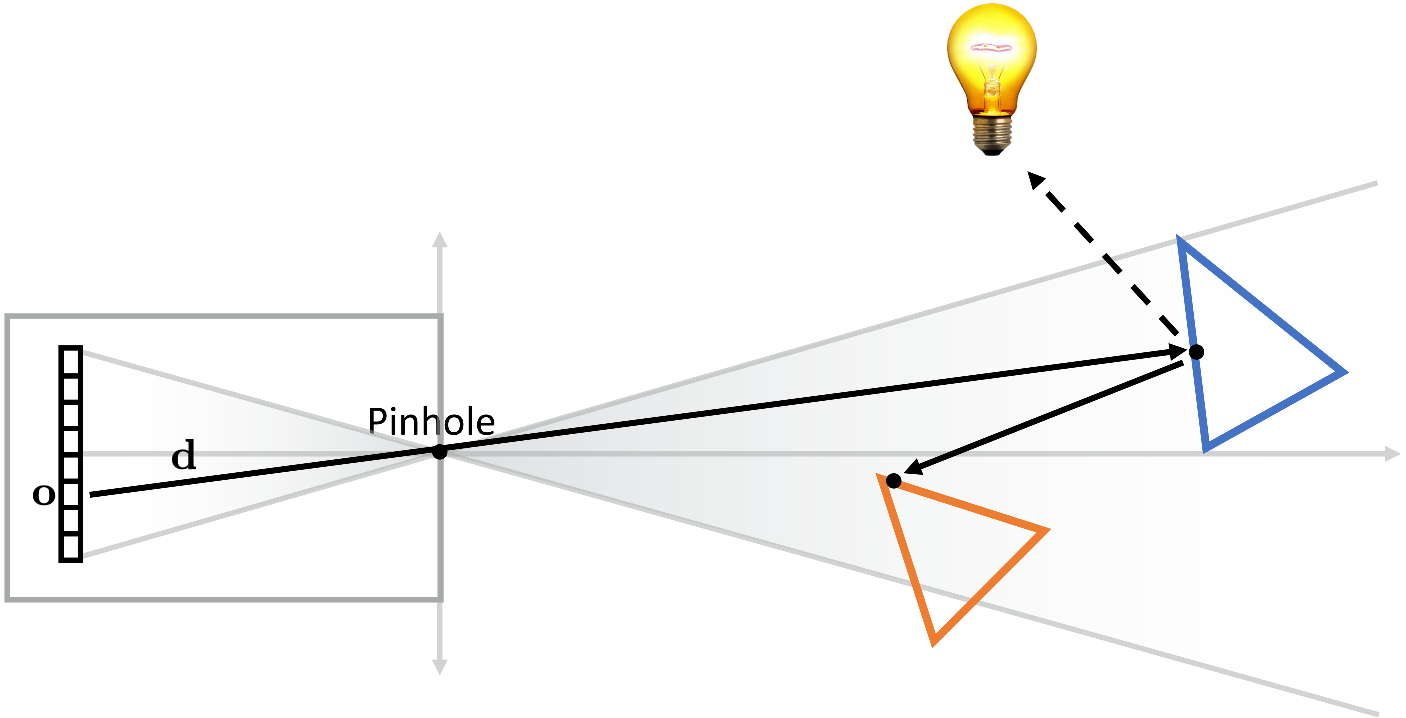


Single Sample Importance Sampling

- Extension to Backwards Path Tracing
  - **At each ray bounce, pick one:**
    - A ray generated by the BRDF
    - A ray towards the light
  - Can only be done for diffuse surfaces!

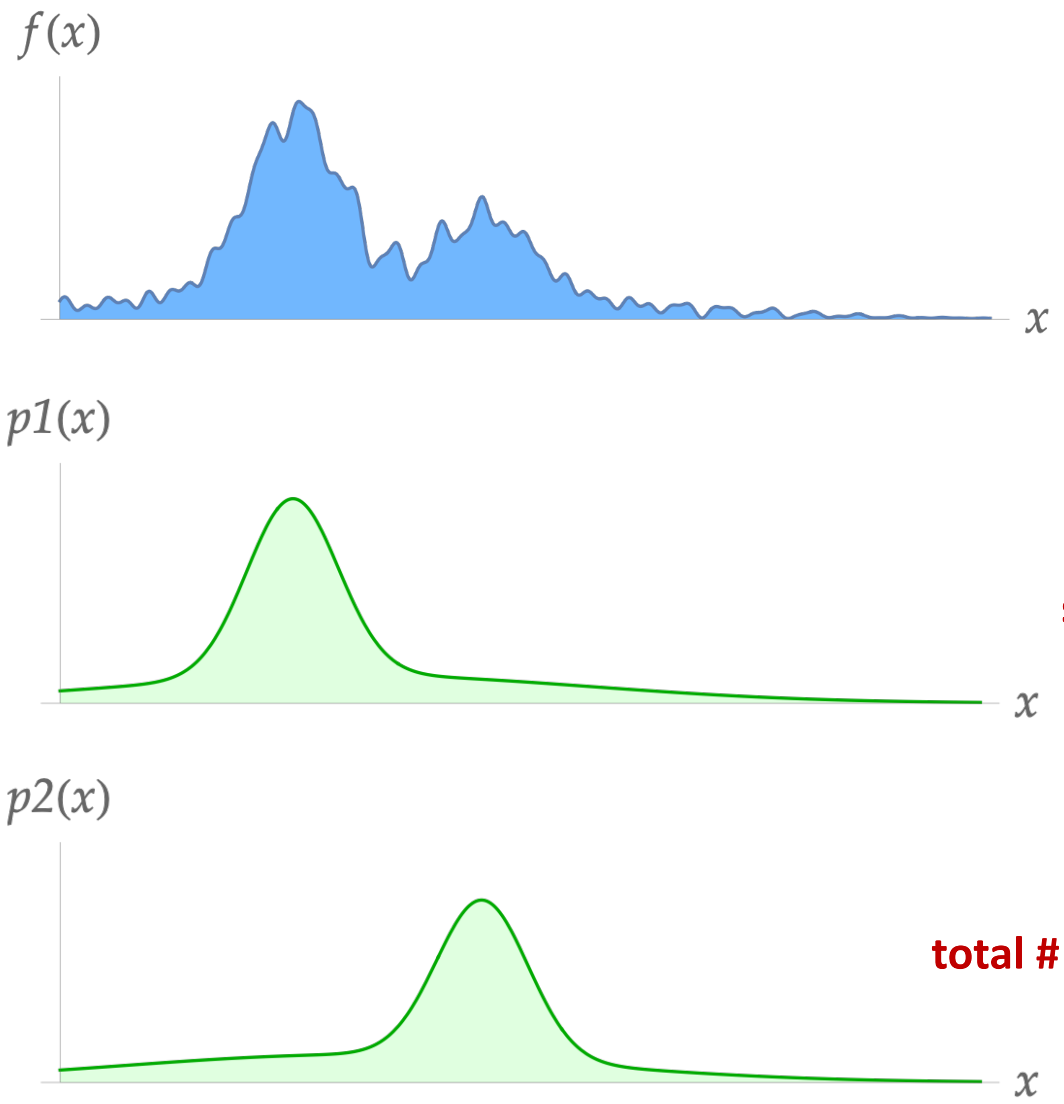
- Sample between rays with uniform probability
  - **Question:** what is the new pdf?

- You will implement this in Scotty3D



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

Multiple Importance Sampling



- Many possible importance sampling strategies
  - Which one should we use for a given integrand?
- **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

sum over samples

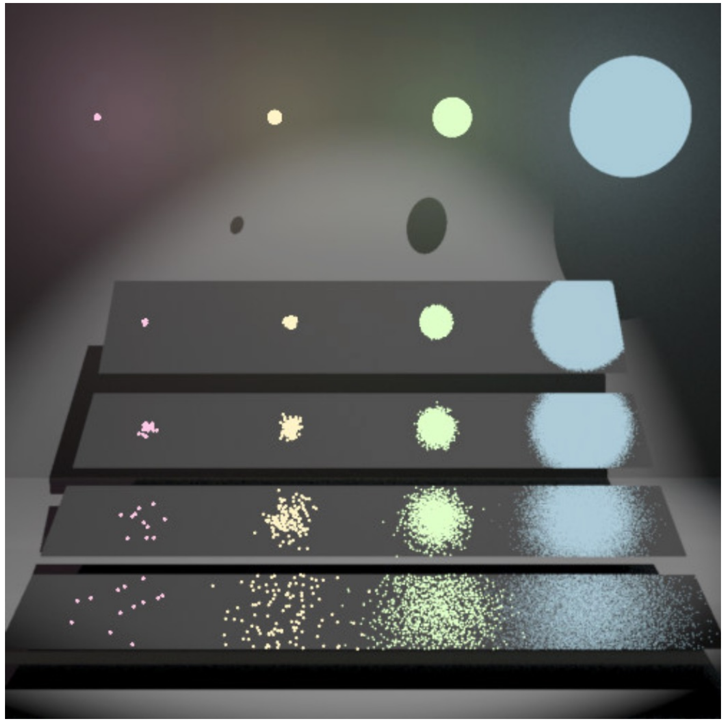
j^{th} sample taken with i^{th} strategy

total # of samples

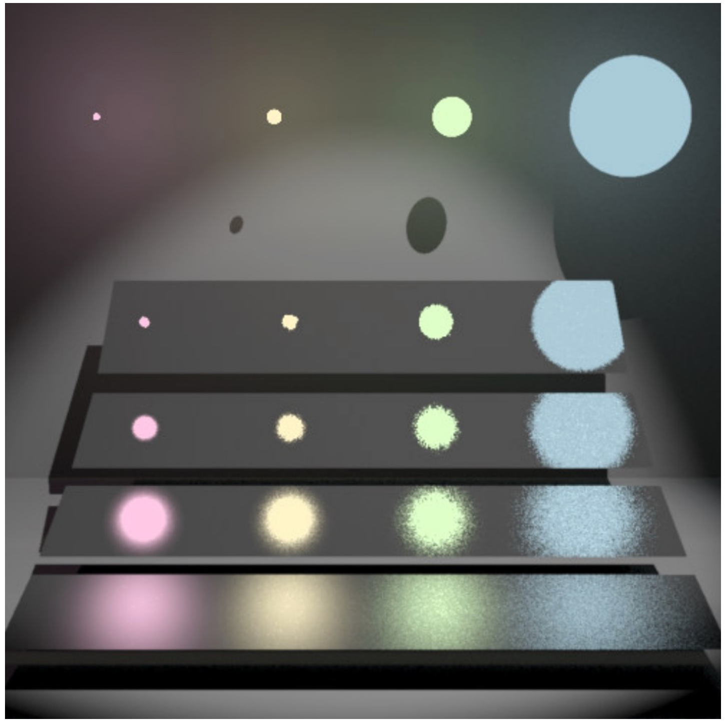
fraction of samples taken with k^{th} strategy

k^{th} strategy PDF

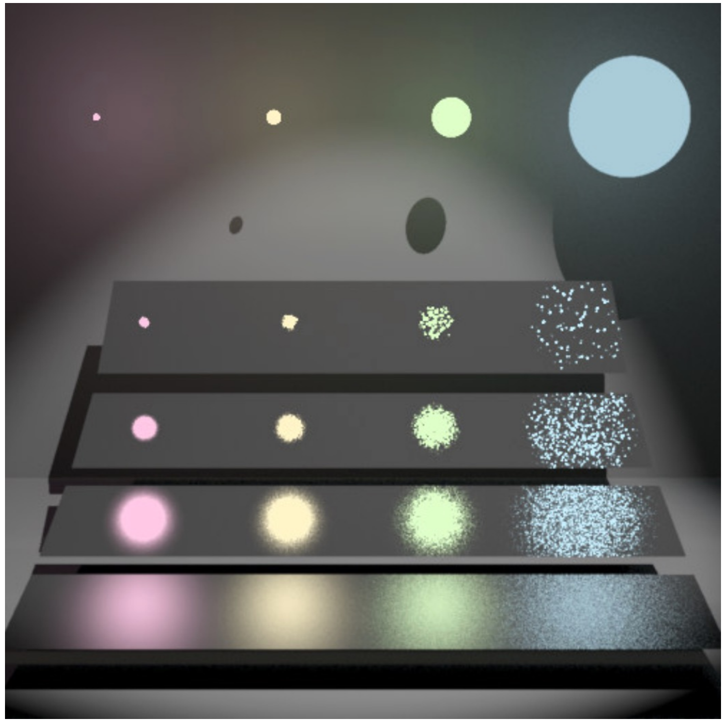
Multiple Importance Sampling



[sample materials]



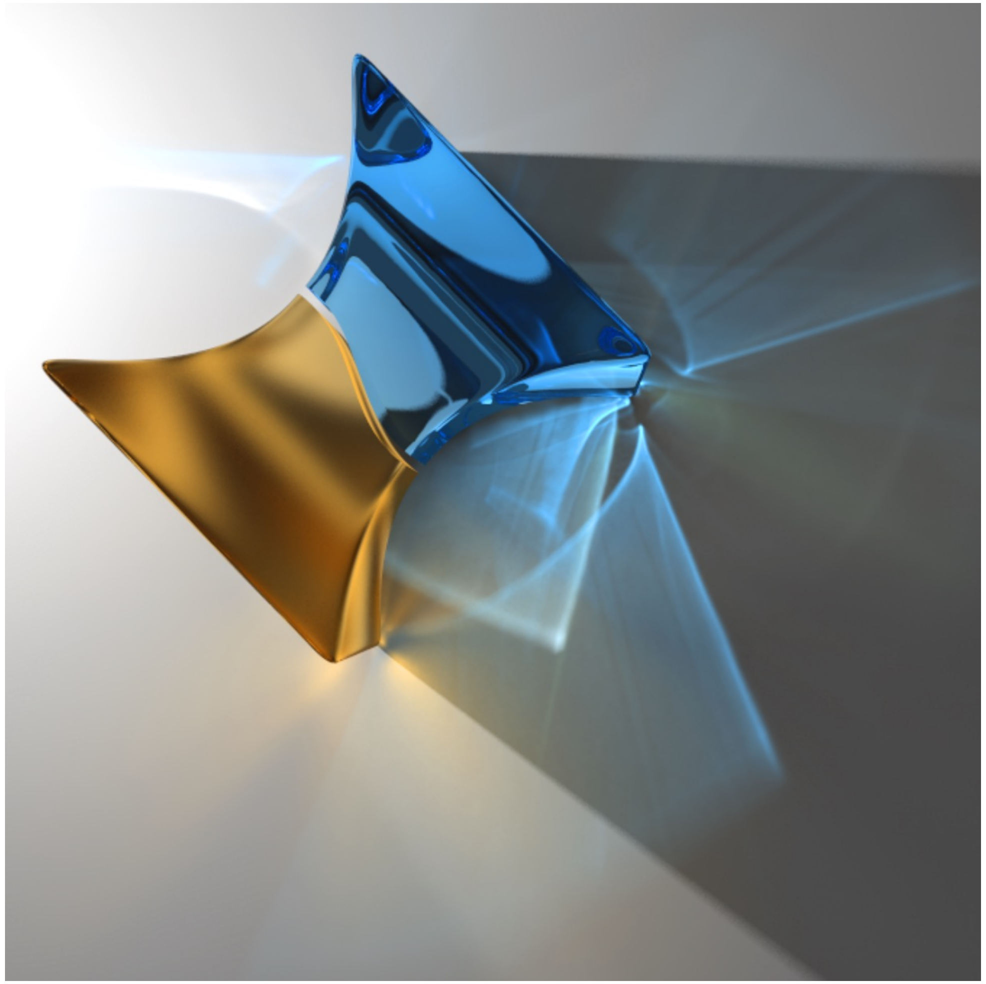
[sample both]



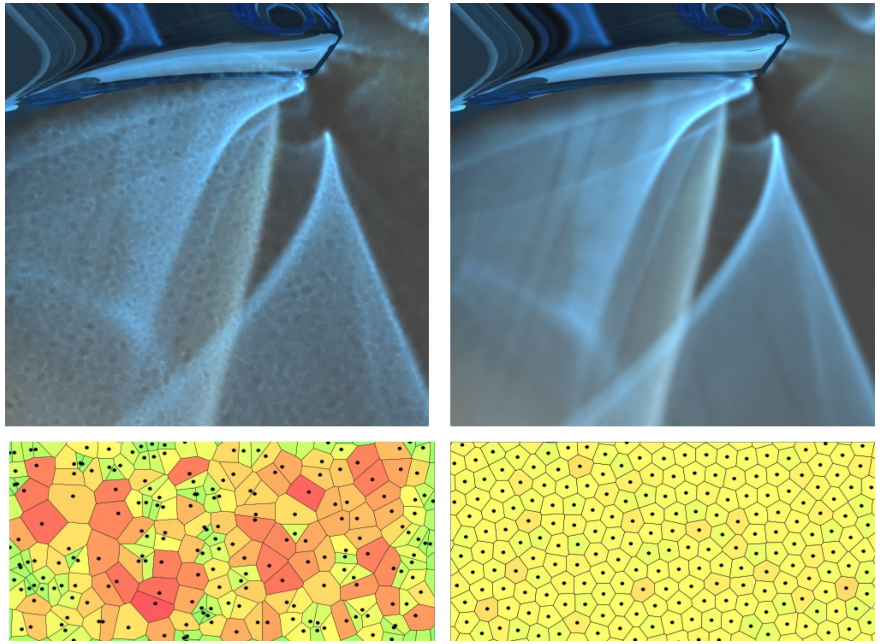
[sample lights]

- 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

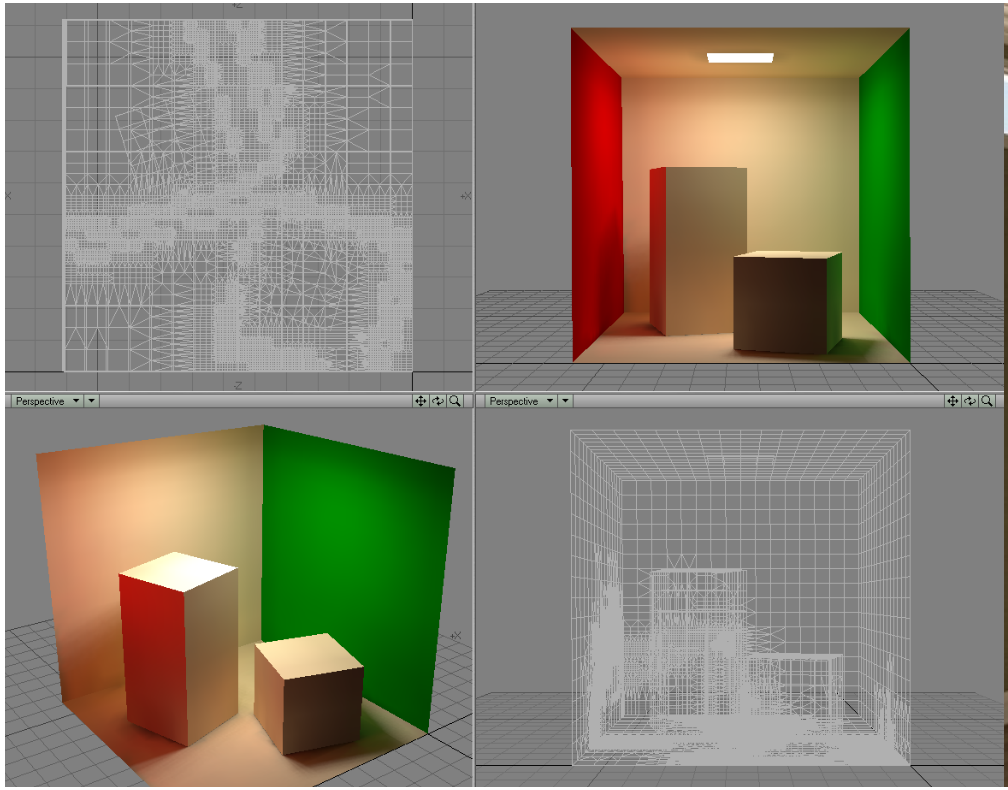


- Trace particles from light, deposit “photons” in kd-tree
  - Useful for, e.g., caustics, fog
- Voronoi diagrams can be used to improve photon distribution



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