

Global Illumination and Monte Carlo



ISTD 50.017 Graphics & Visualization
Sai-Kit Yeung
Note courtesy by Wojciech Matusik
with many slides from Fredo Durand and Jaakko Lehtinen

Today

- Lots of randomness!

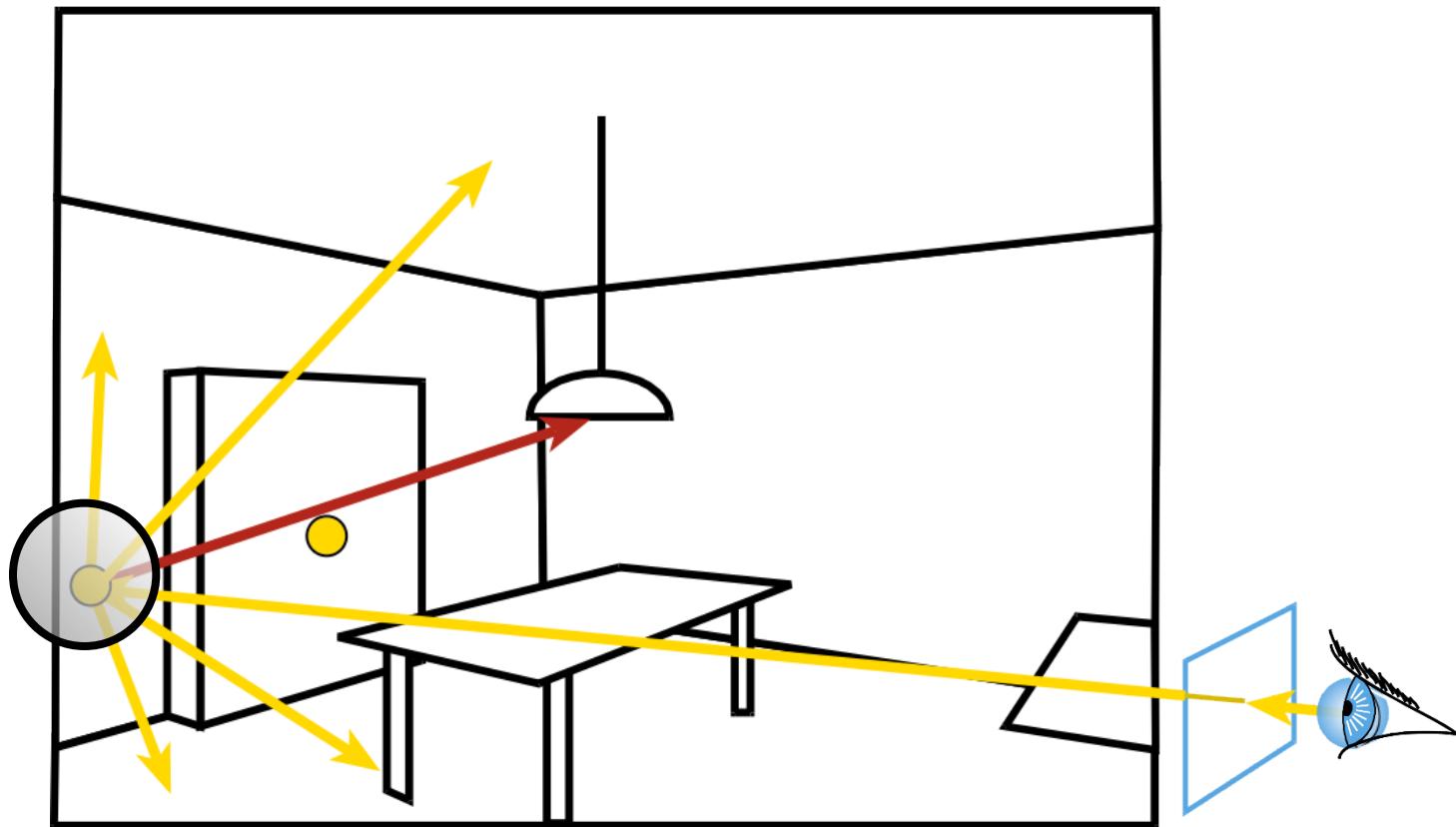
Today

- Global Illumination
 - Rendering Equation
 - Path tracing
- Monte Carlo integration
- Better sampling
 - importance
 - stratification

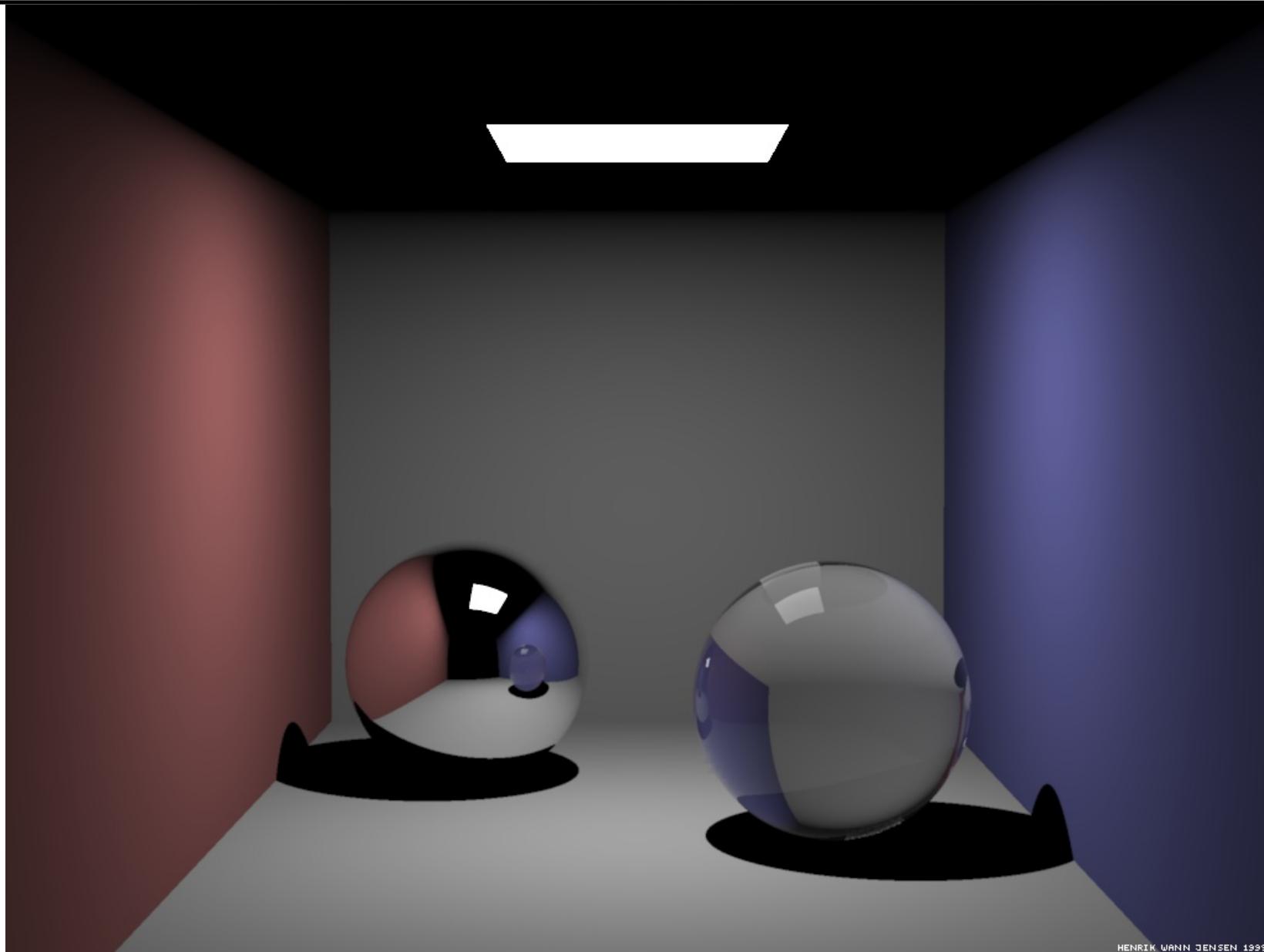


Global Illumination

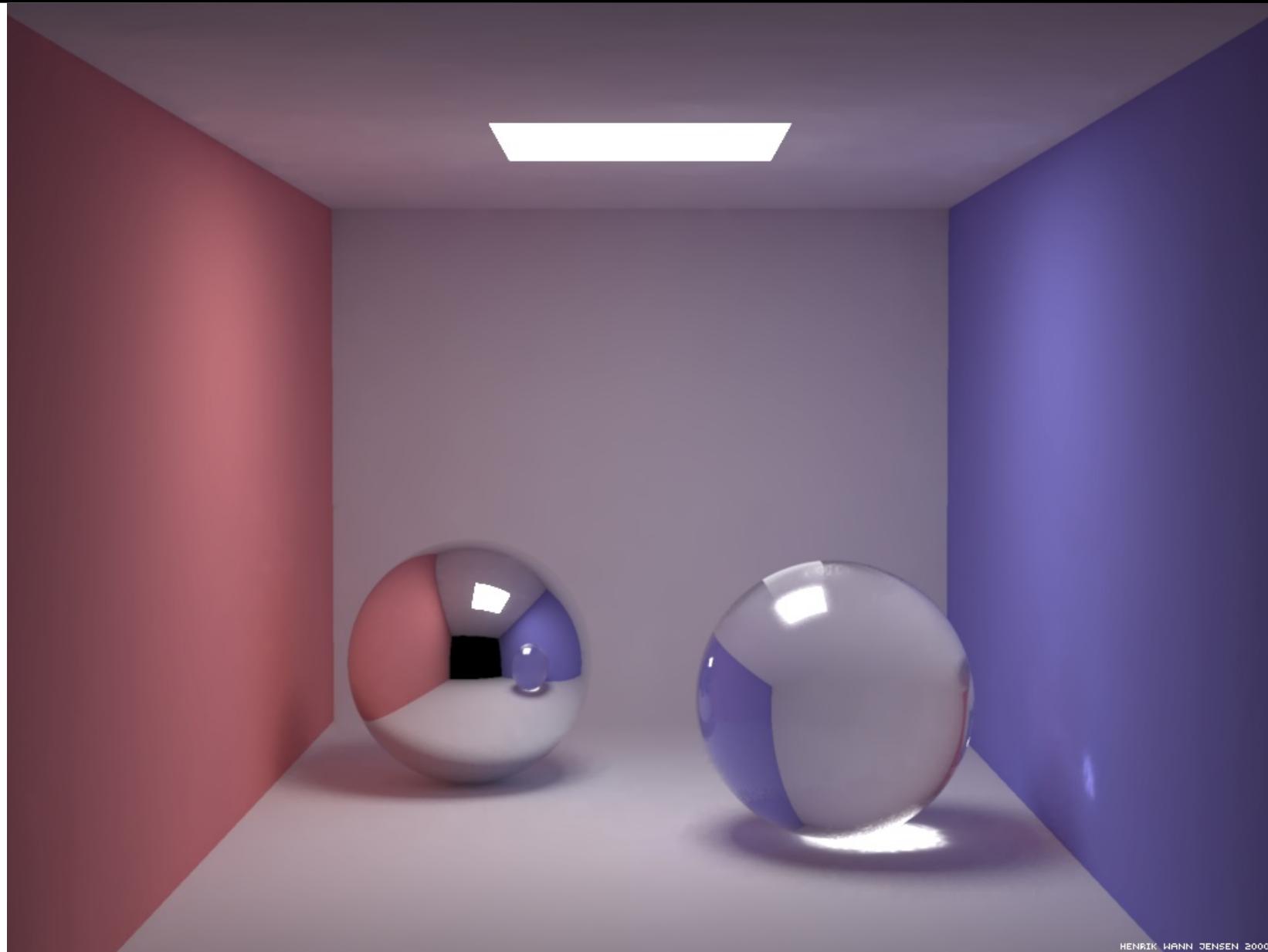
- So far, we've seen only direct lighting (red here)
- We also want indirect lighting
 - Full integral of all directions (multiplied by BRDF)
 - In practice, send tons of random rays



Direct Illumination

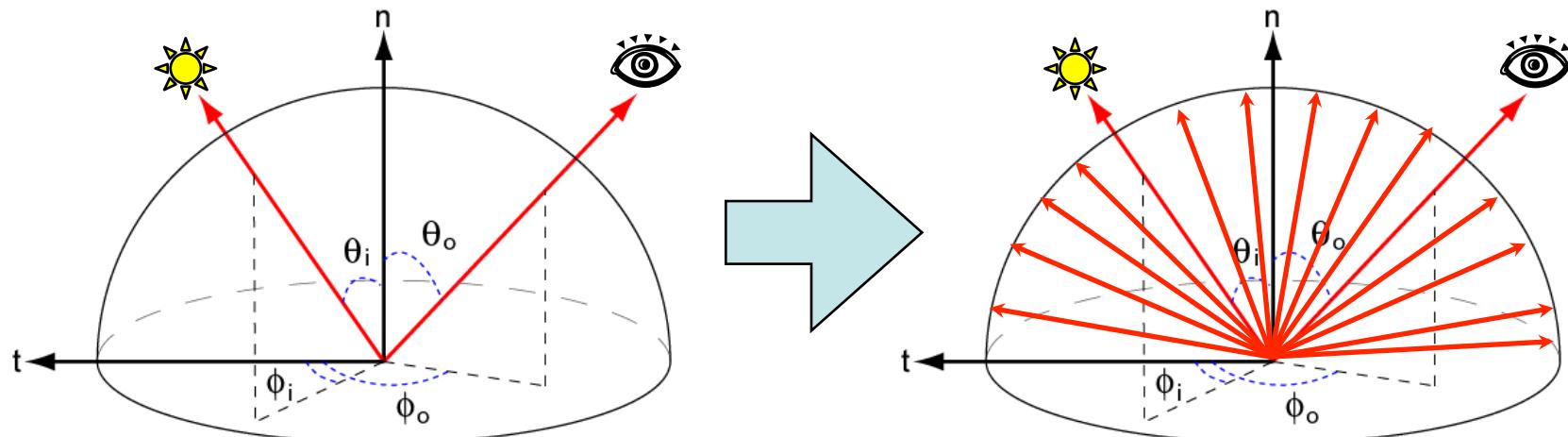


Global Illumination (with Indirect)



Global Illumination

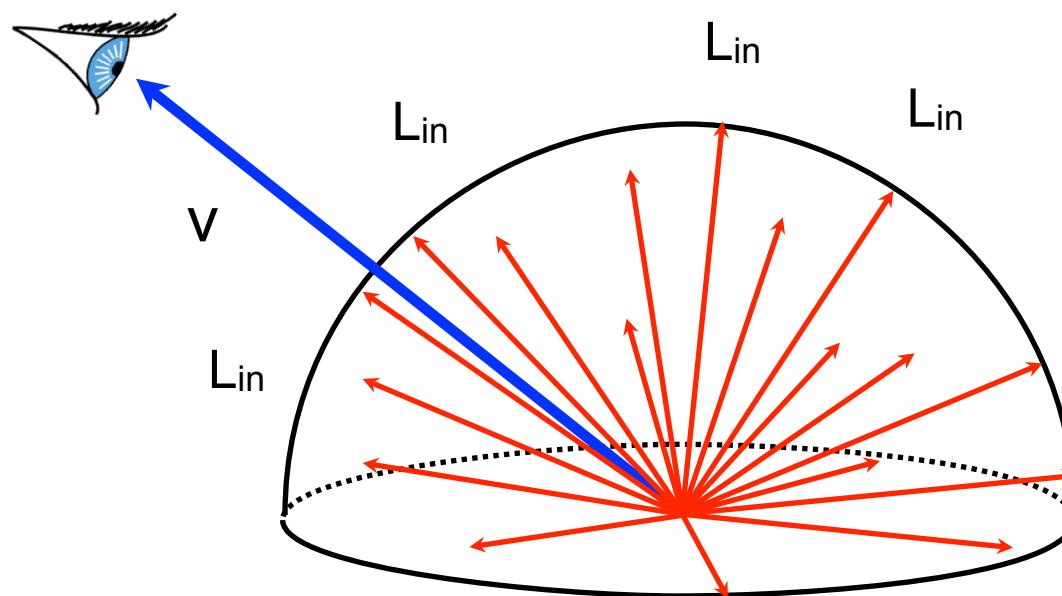
- So far, we only used the BRDF for point lights
 - We just summed over all the point light sources
- BRDF also describes how indirect illumination reflects off surfaces
 - Turns summation into integral over hemisphere
 - As if every direction had a light source



Reflectance Equation, Visually

$$L_{\text{out}}(x, \mathbf{v}) = \int_{\Omega} L_{\text{in}}(\mathbf{l}) f_r(x, \mathbf{l}, \mathbf{v}) \cos \theta \, d\mathbf{l}$$

outgoing light to direction \mathbf{v} Ω incident light from direction omega the BRDF cosine term

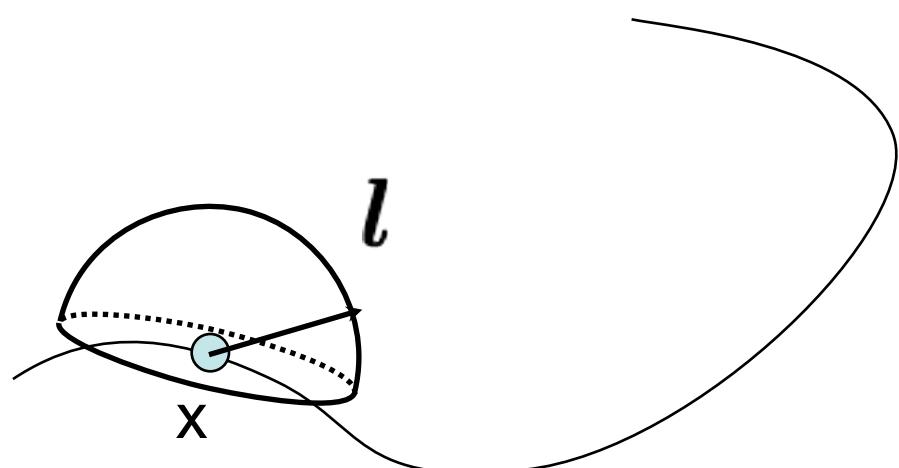


Sum (integrate) over every direction on the hemisphere, modulate incident illumination by BRDF

The Reflectance Equation

$$L_{\text{out}}(x, \mathbf{v}) = \int_{\Omega} L_{\text{in}}(\mathbf{l}) f_r(x, \mathbf{l}, \mathbf{v}) \cos \theta \, d\mathbf{l}$$

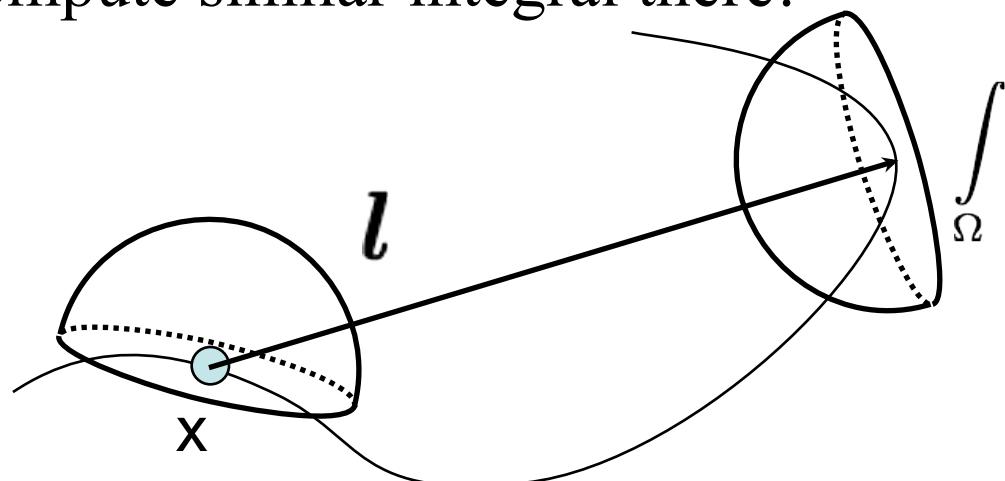
- Where does L_{in} come from?



The Reflectance Equation

$$L_{\text{out}}(x, \mathbf{v}) = \int_{\Omega} L_{\text{in}}(\mathbf{l}) f_r(x, \mathbf{l}, \mathbf{v}) \cos \theta \, d\mathbf{l}$$

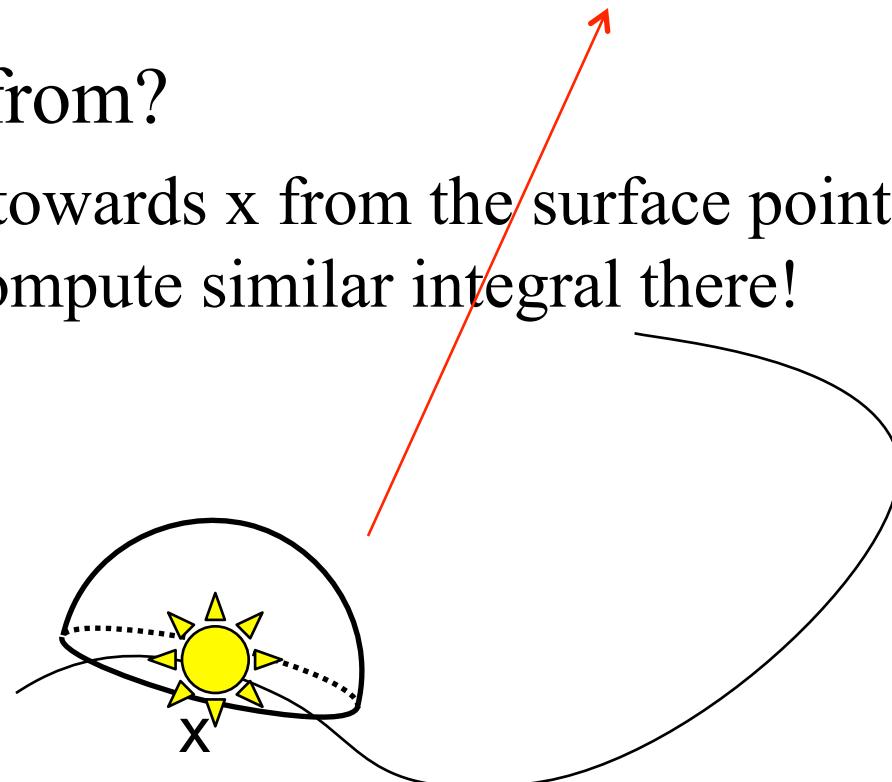
- Where does L_{in} come from?
 - It is the light reflected towards x from the surface point in direction $\mathbf{l} \implies$ must compute similar integral there!
 - Recursive!



The Rendering Equation

$$L_{\text{out}}(x, \mathbf{v}) = \int_{\Omega} L_{\text{in}}(\mathbf{l}) f_r(x, \mathbf{l}, \mathbf{v}) \cos \theta \, d\mathbf{l} + E_{\text{out}}(x, \mathbf{v})$$

- Where does L_{in} come from?
 - It is the light reflected towards x from the surface point in direction $\mathbf{l} \implies$ must compute similar integral there!
 - Recursive!
 - AND if x happens to be a light source, we add its contribution directly



The Rendering Equation

$$L_{\text{out}}(x, \mathbf{v}) = \int_{\Omega} L_{\text{in}}(\mathbf{l}) f_r(x, \mathbf{l}, \mathbf{v}) \cos \theta \, d\mathbf{l} + E_{\text{out}}(x, \mathbf{v})$$

- The rendering equation describes the appearance of the scene, including direct and indirect illumination
 - An “integral equation”, the unknown solution function L is both on the LHS and on the RHS inside the integral
 - Must either discretize or use Monte Carlo integration
 - Originally described by [Kajiya](#) and [Immel et al.](#) in 1986
 - “The rendering equation”, SIGGRAPH 86
 - Also, see book references towards the end

The Rendering Equation

- Analytic solution is usually impossible
- Lots of ways to solve it approximately
- Monte Carlo techniques use random samples for evaluating the integrals
 - We'll look at some simple method in a bit...
- Finite element methods discretize the solution using basis functions (again!)
 - Radiosity, wavelets, precomputed radiance transfer, etc.

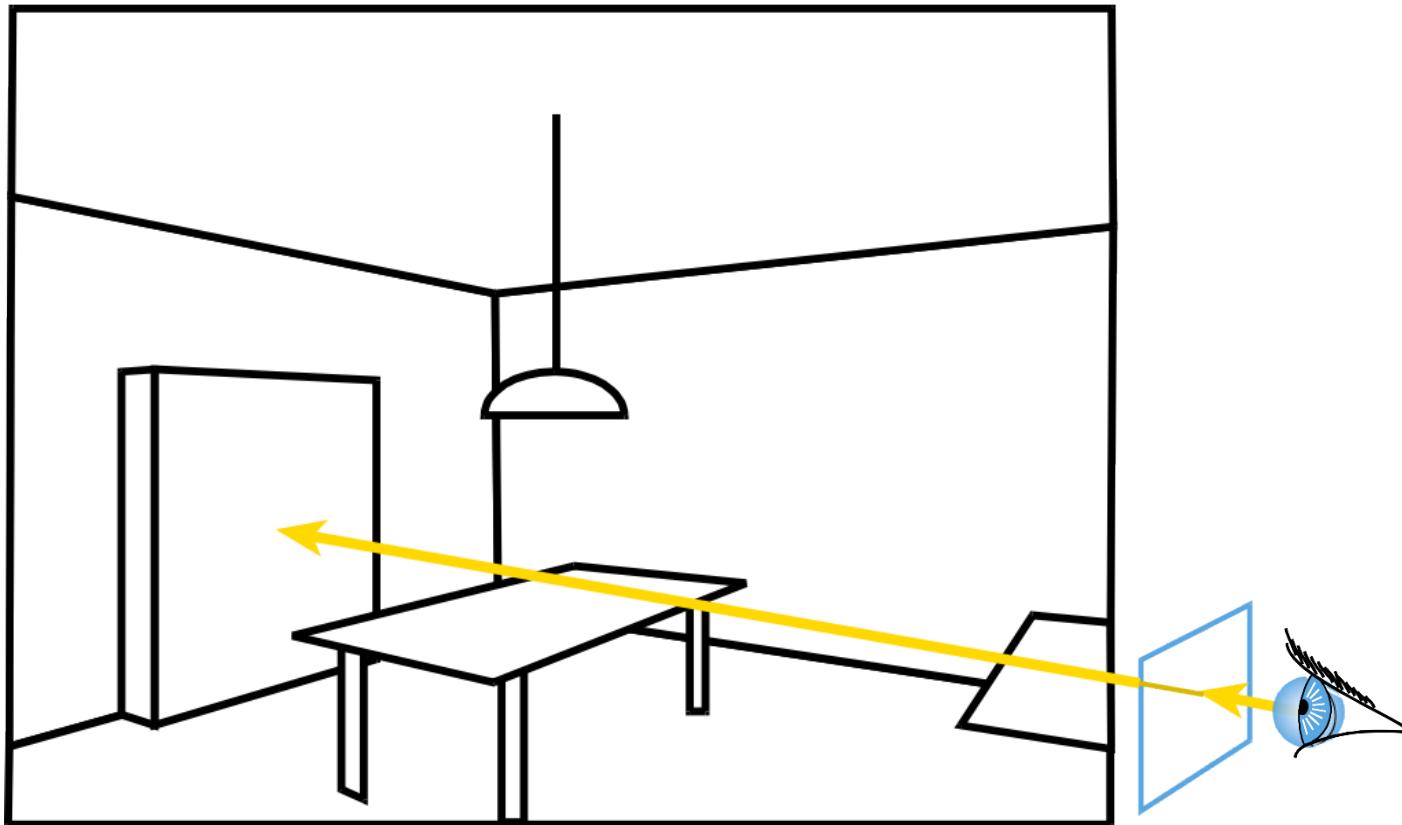
Questions?

How To Render Global Illumination?



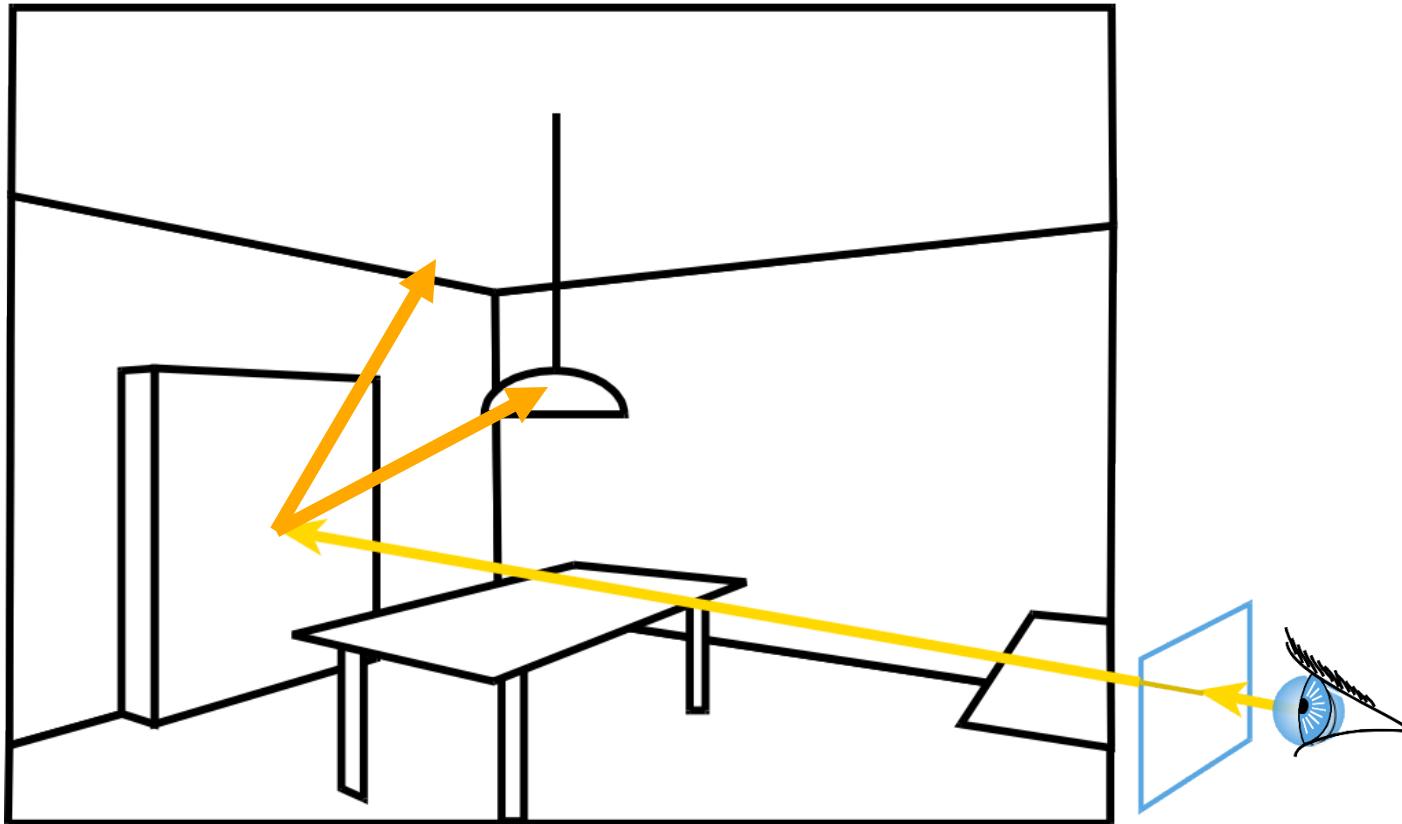
Ray Casting

- Cast a ray from the eye through each pixel



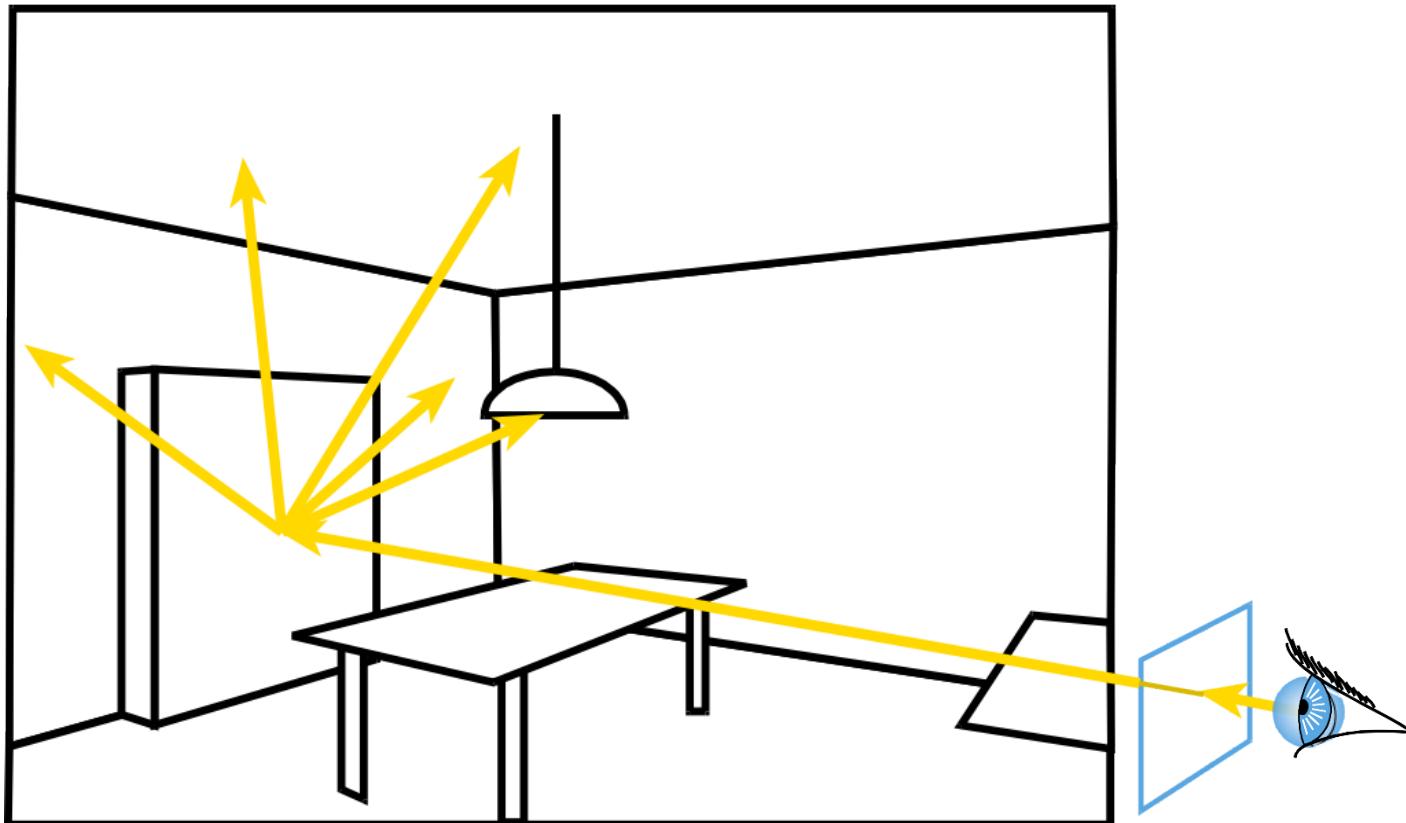
Ray Tracing

- Cast a ray from the eye through each pixel
- Trace secondary rays (shadow, reflection, refraction)



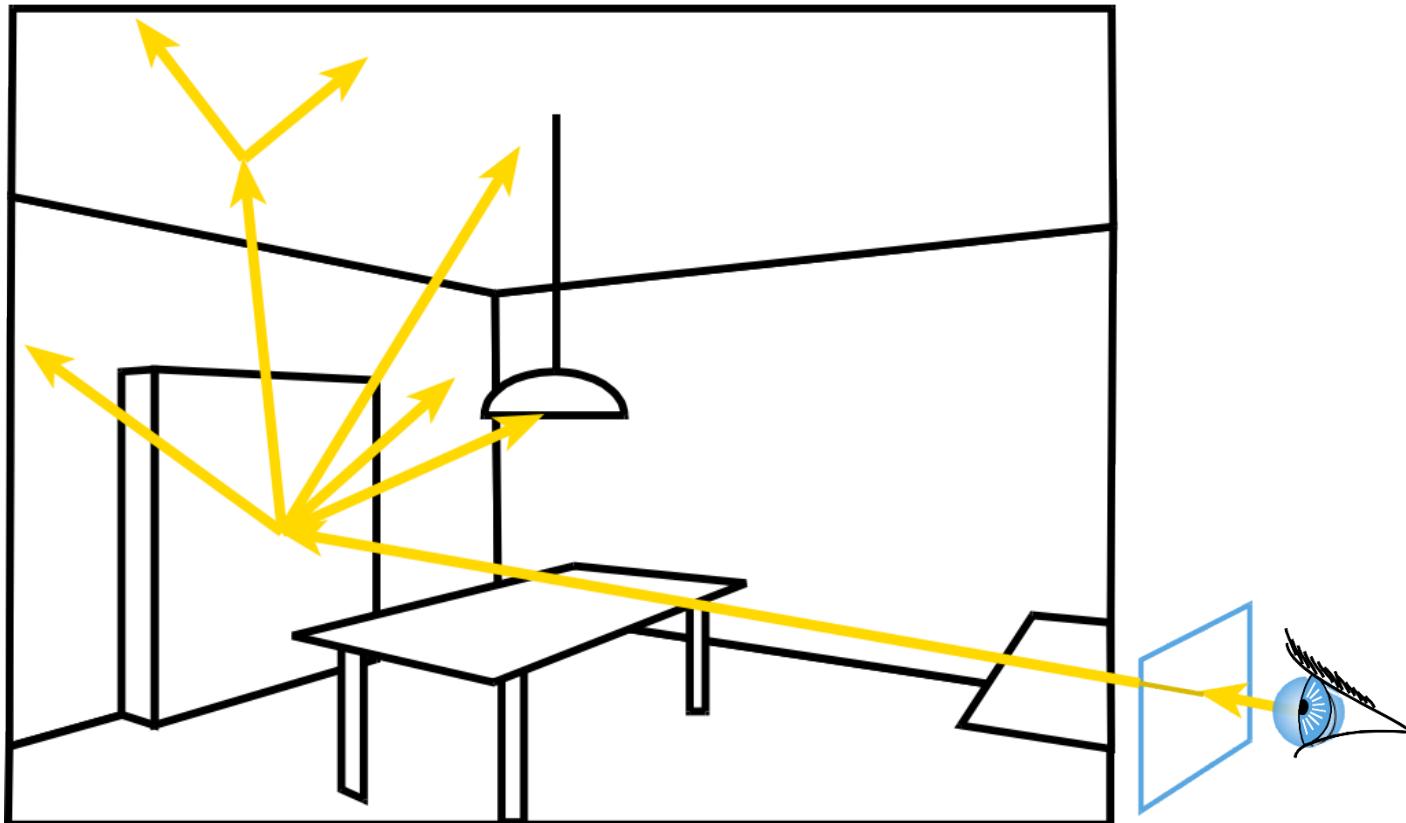
“Monte-Carlo Ray Tracing”

- Cast a ray from the eye through each pixel
- Cast random rays from the hit point to evaluate hemispherical integral using random sampling



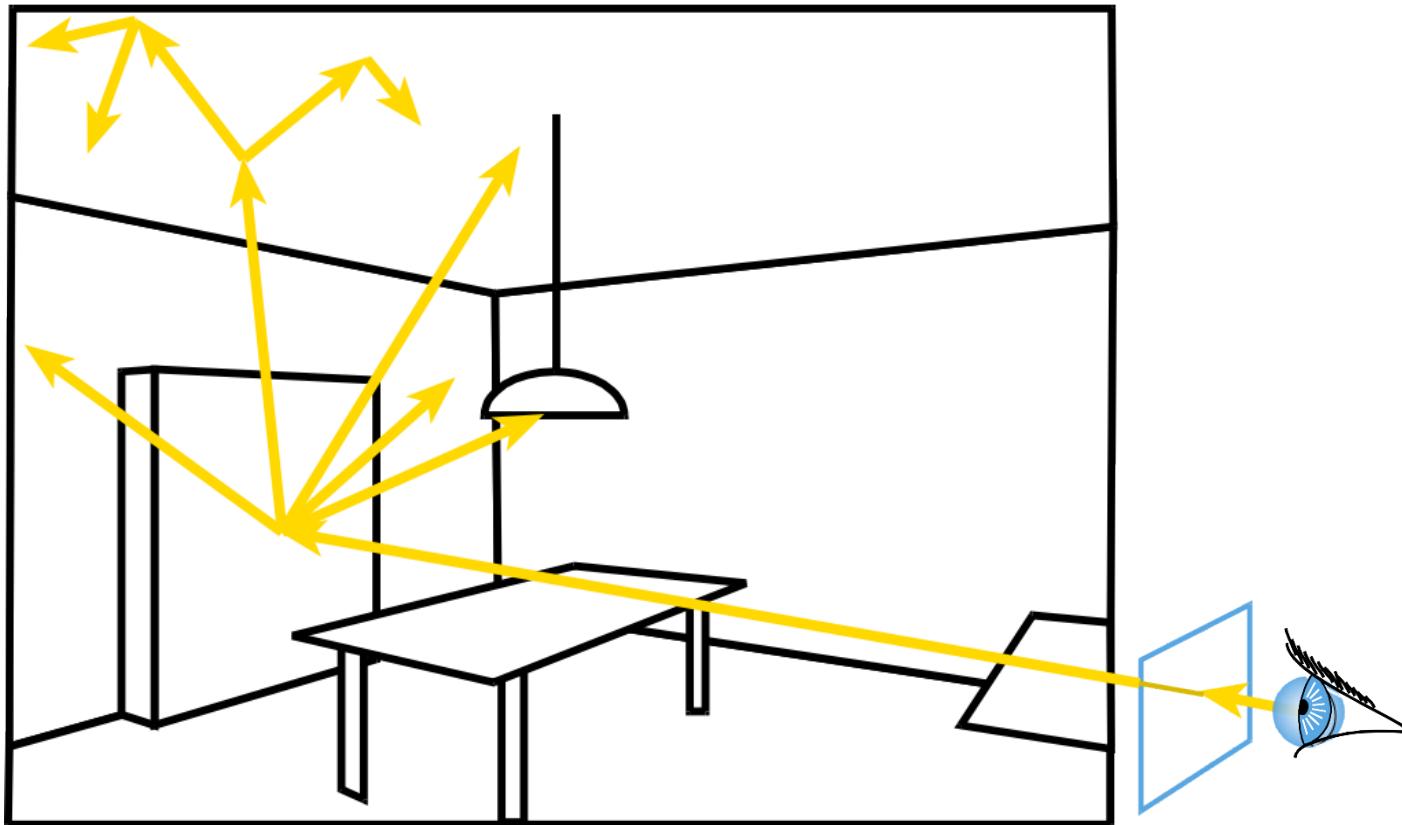
“Monte-Carlo Ray Tracing”

- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse



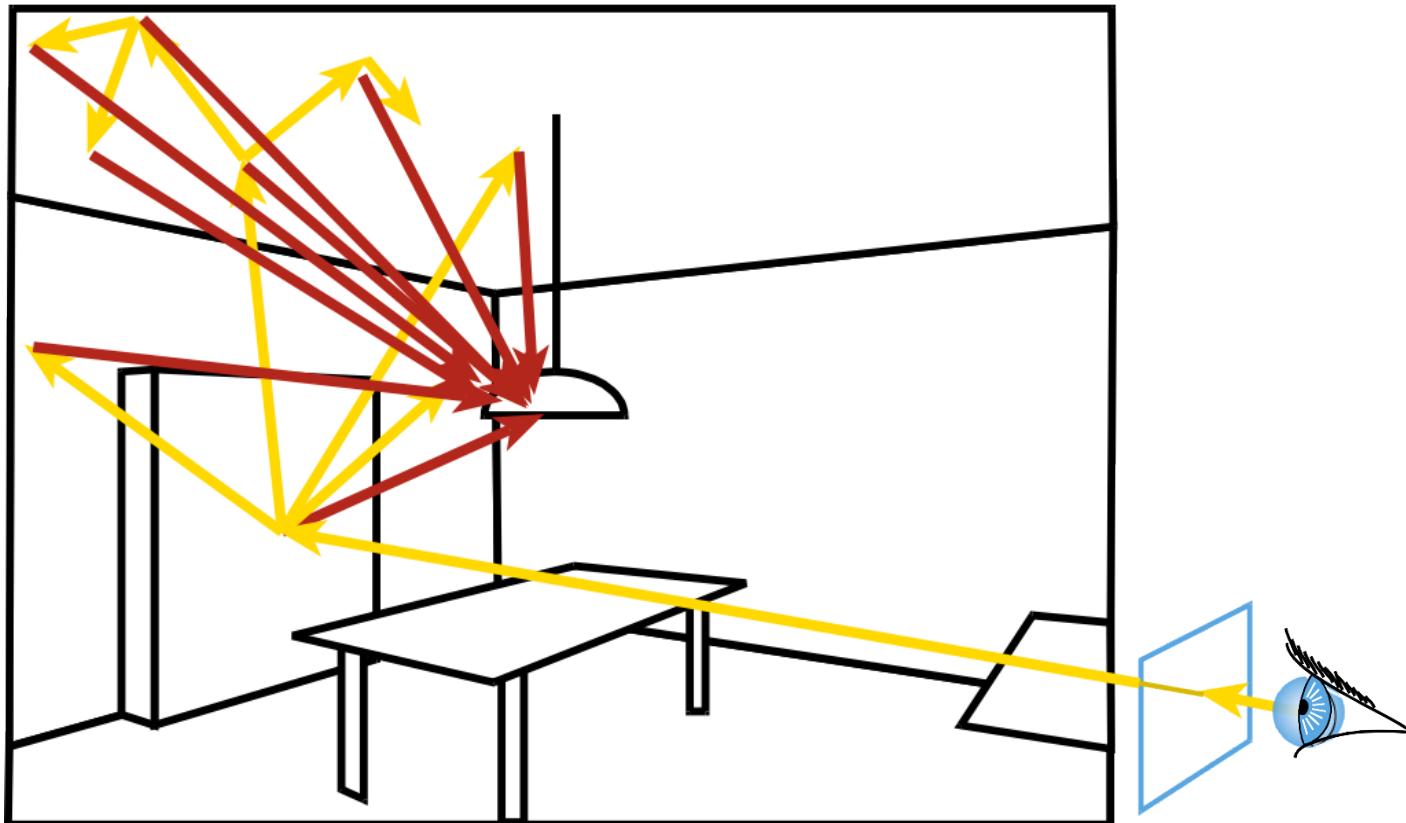
“Monte-Carlo Ray Tracing”

- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse



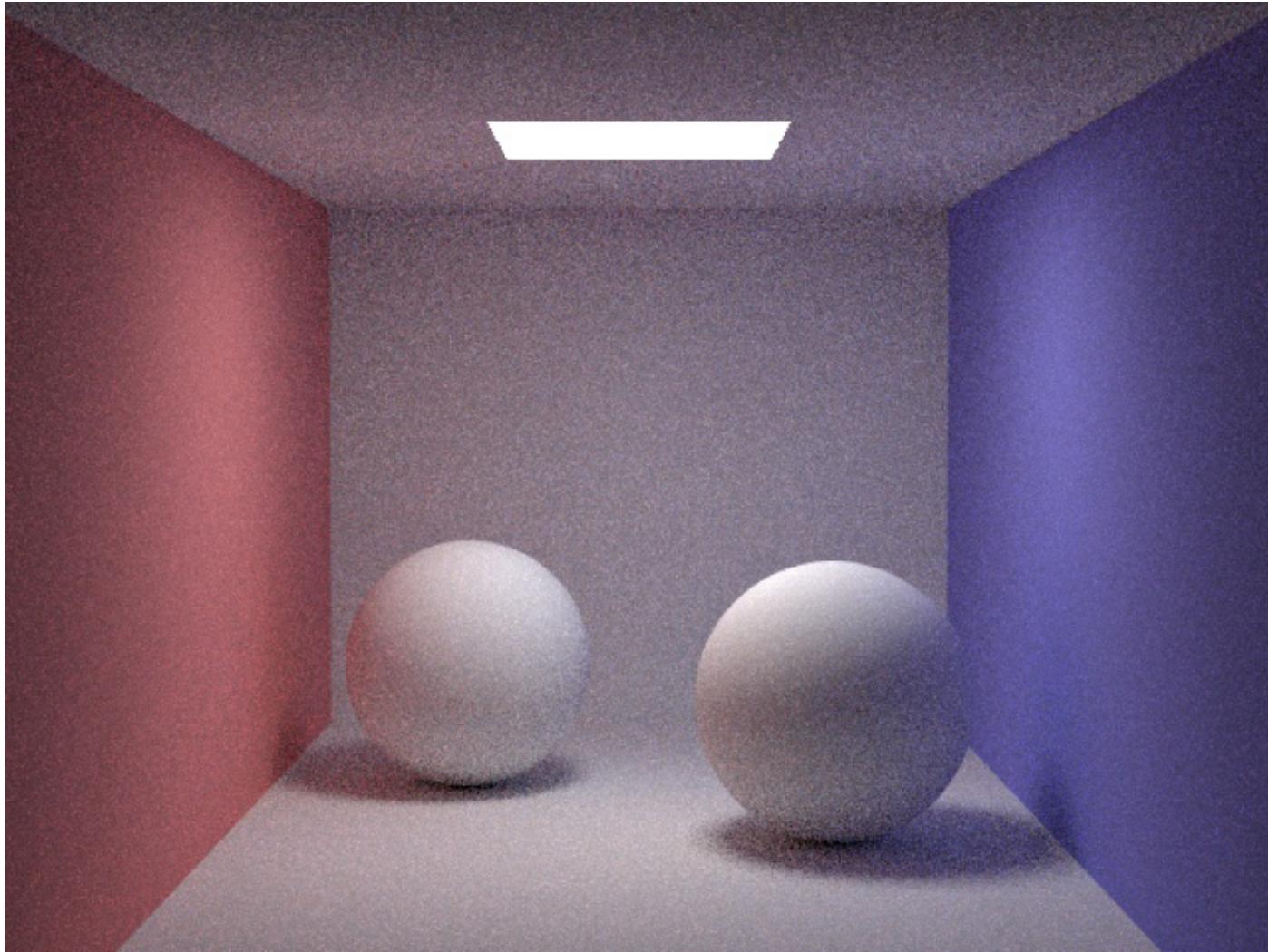
“Monte-Carlo Ray Tracing”

- Systematically sample light sources at each hit
 - Don't just wait the rays will hit it by chance



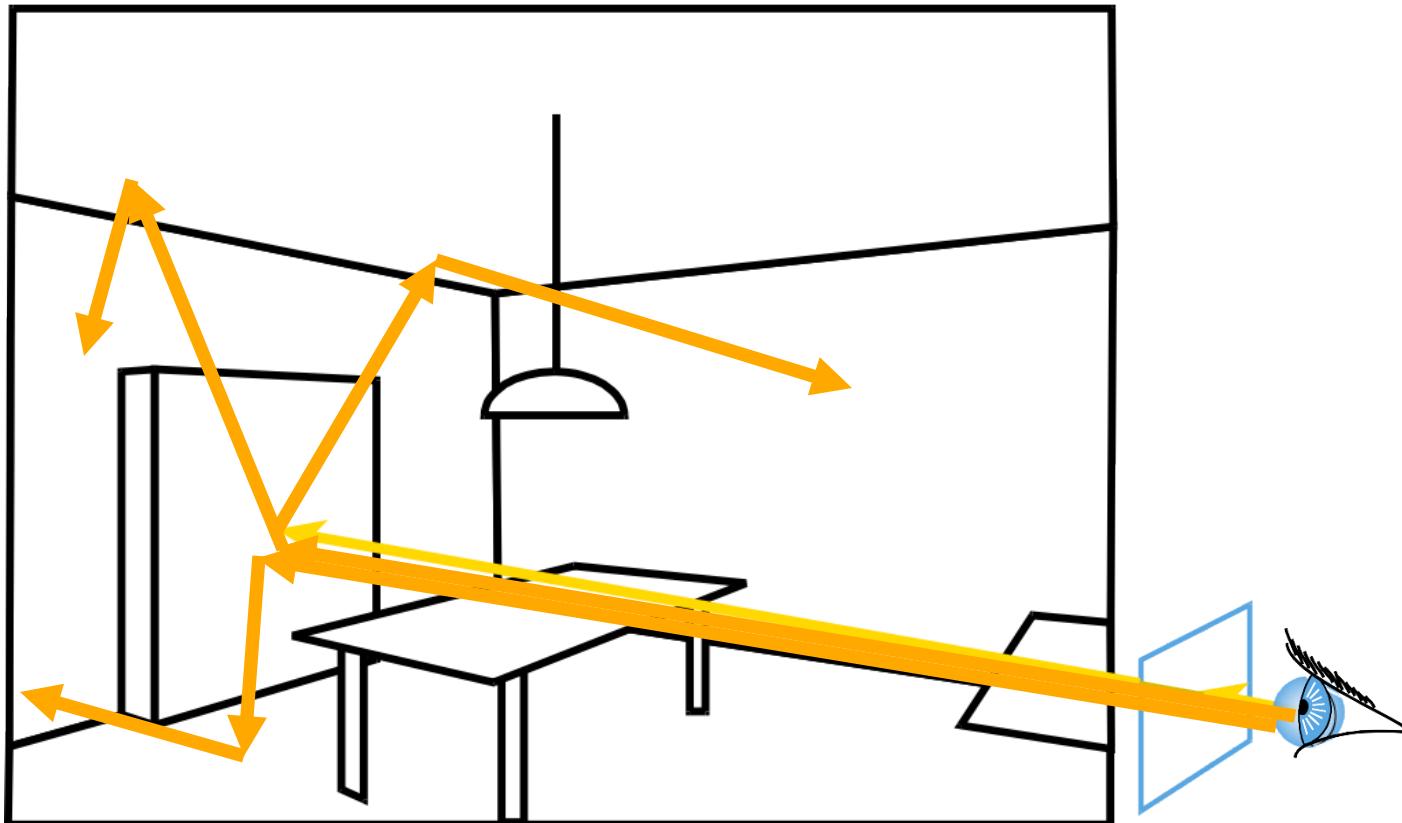
Results

Henrik Wann Jensen



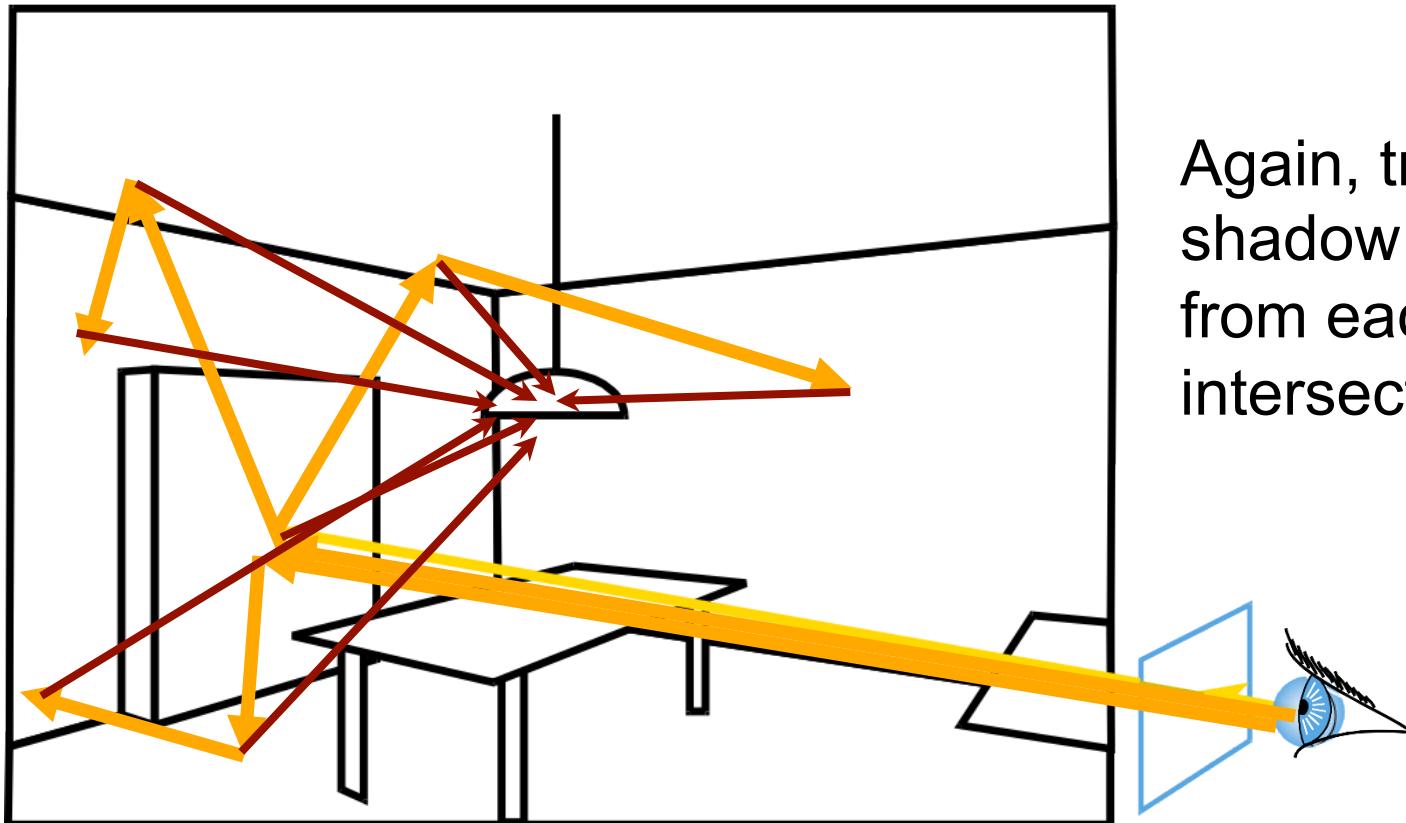
Monte Carlo Path Tracing

- Trace only one secondary ray per recursion
 - Otherwise number of rays explodes! (also, contribution decreases)
- But send many primary rays per pixel (antialiasing)



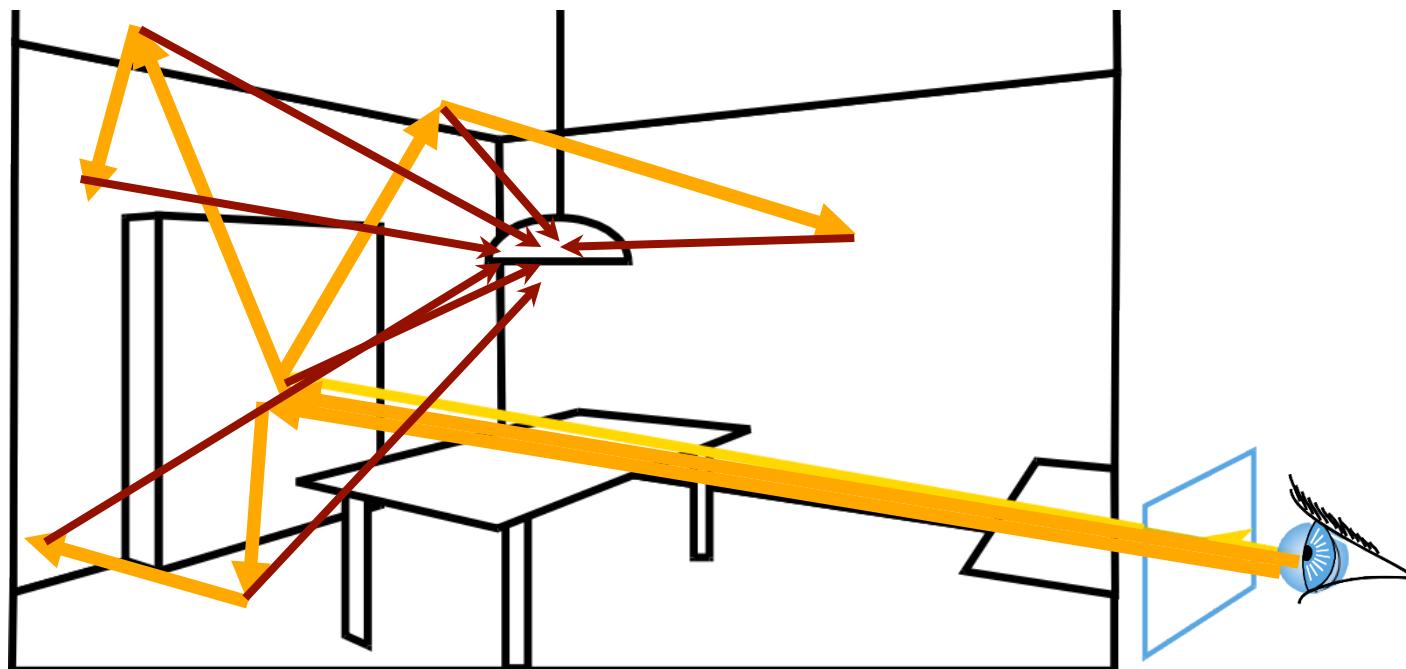
Monte Carlo Path Tracing

- Trace only one secondary ray per recursion
 - Otherwise number of rays explodes!
- But send many primary rays per pixel (antialiasing)



Monte Carlo Path Tracing

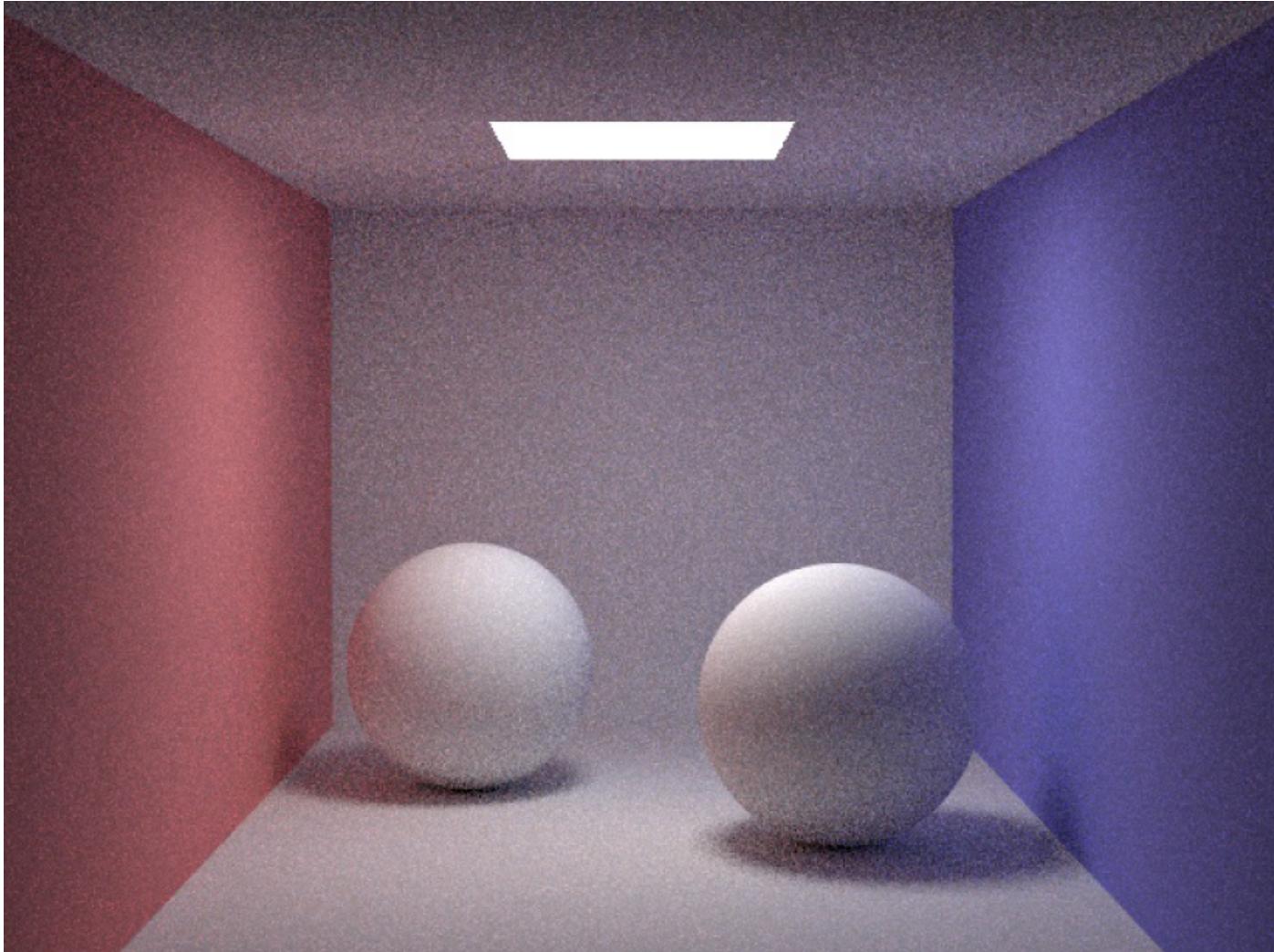
- We shoot one path from the eye at a time
 - Connect every surface point on the way to the light by a shadow ray
 - We are randomly sampling the space of all possible light paths between the source and the camera



Path Tracing Results

- 10 paths/pixel

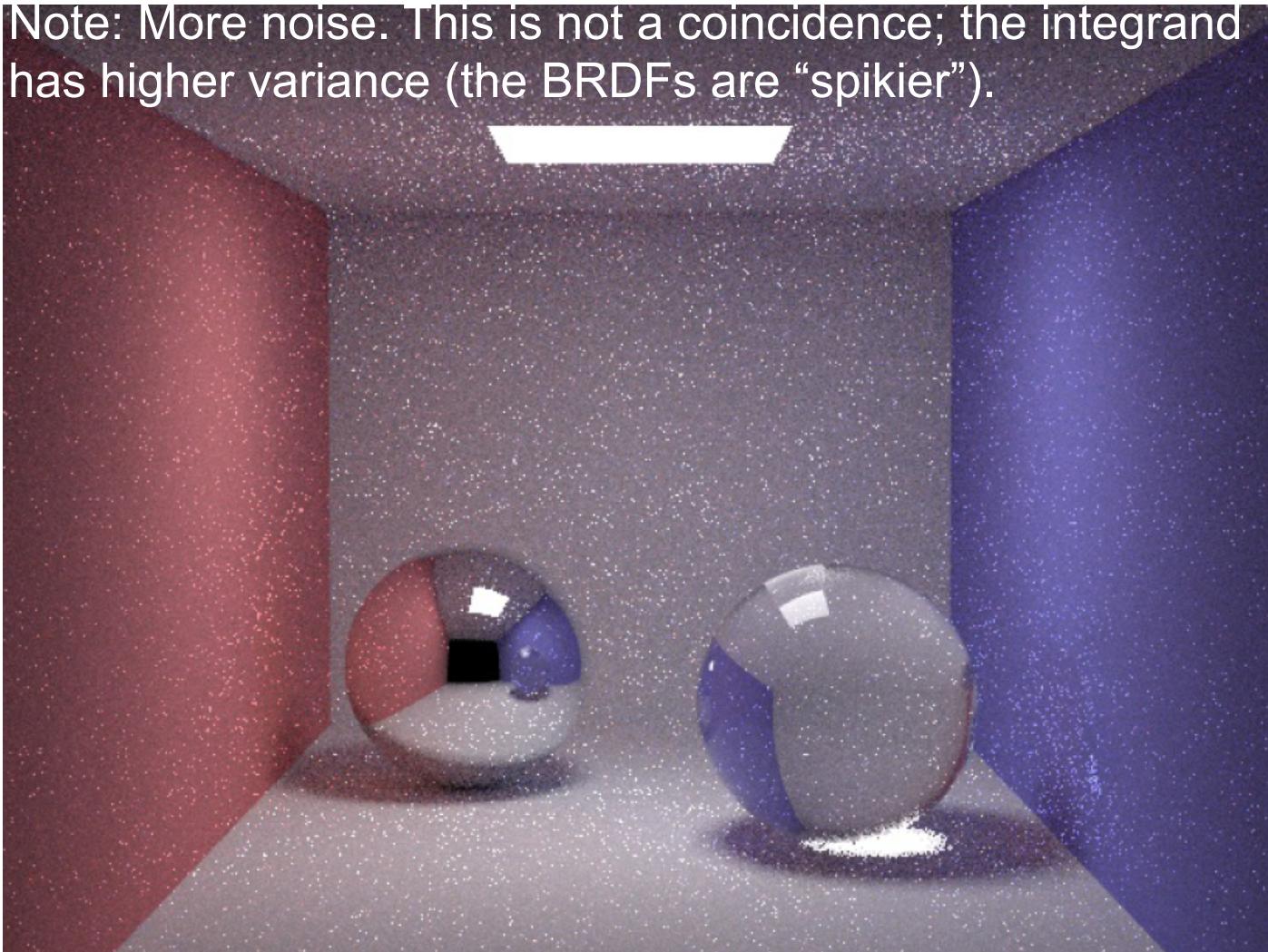
Henrik Wann Jensen



Path Tracing Results: Glossy Scene

- 10 paths/pixel

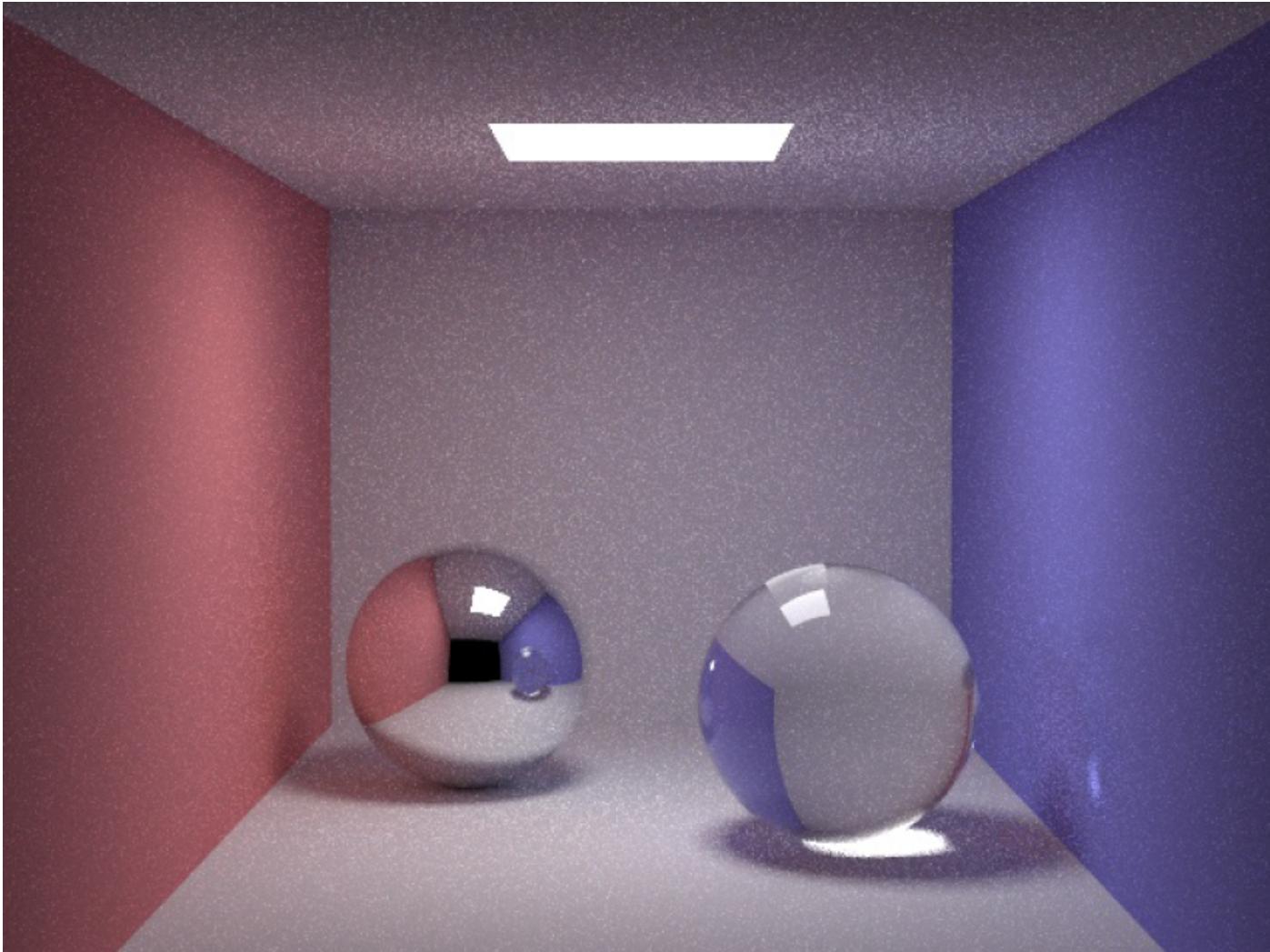
Note: More noise. This is not a coincidence; the integrand has higher variance (the BRDFs are “spikier”).



Path Tracing Results: Glossy Scene

- 100 paths/pixel

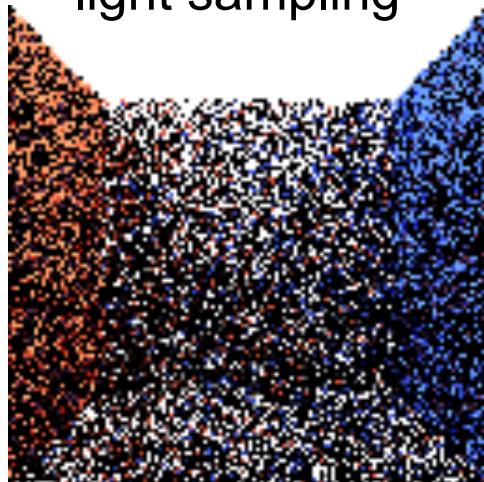
Henrik Wann Jensen



Importance of Sampling the Light

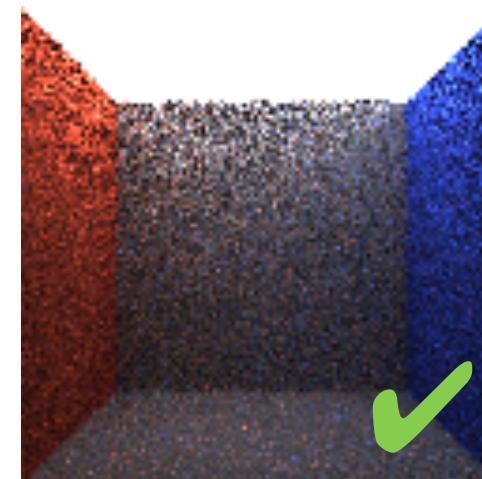
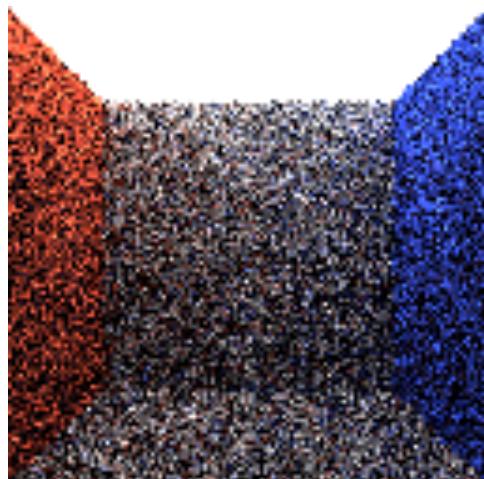
Without explicit
light sampling

1 path
per pixel



With explicit
light sampling

4 paths
per pixel



Why Use Random Numbers?

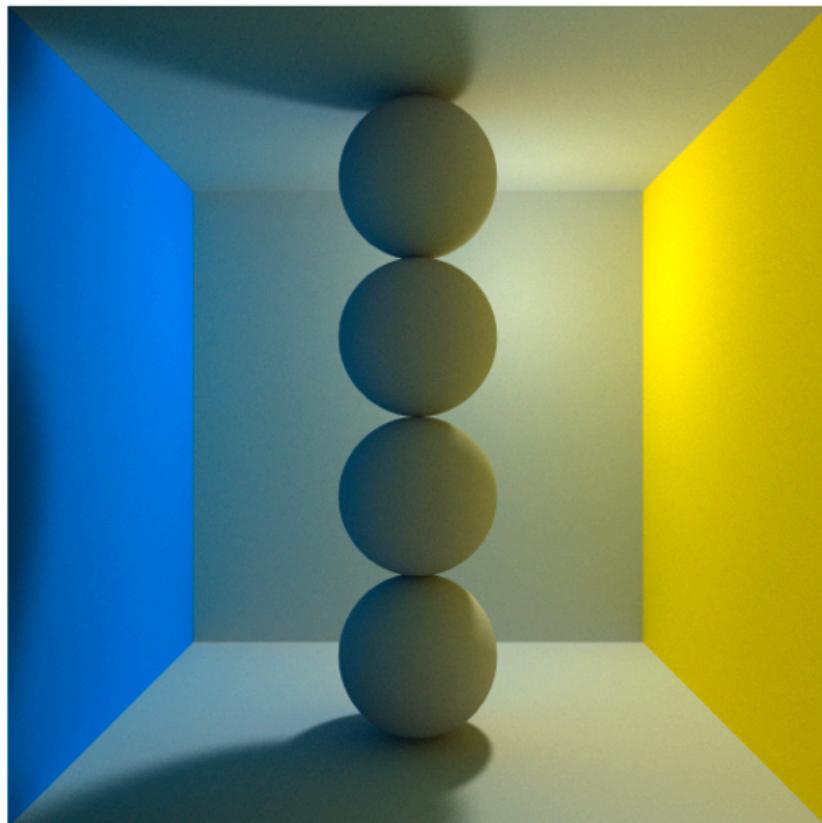
- Fixed random sequence
- We see the structure in the error



Henrik Wann Jensen

Demo

- <http://madebyevan.com/webgl-path-tracing/>



Select Light Add Sphere Add Cube

Material: Diffuse Environment: Cornell Box - Yellow and Blue

Load preset scene:

- Sphere Column
- Sphere Pyramid
- Sphere and Cube
- Cube and Spheres
- Table and Chair
- Stacks

For more demo/experimentation

- <http://www.mitsuba-renderer.org/>
- <http://www.pbrt.org/>
- http://www.luxrender.net/en_GB/index

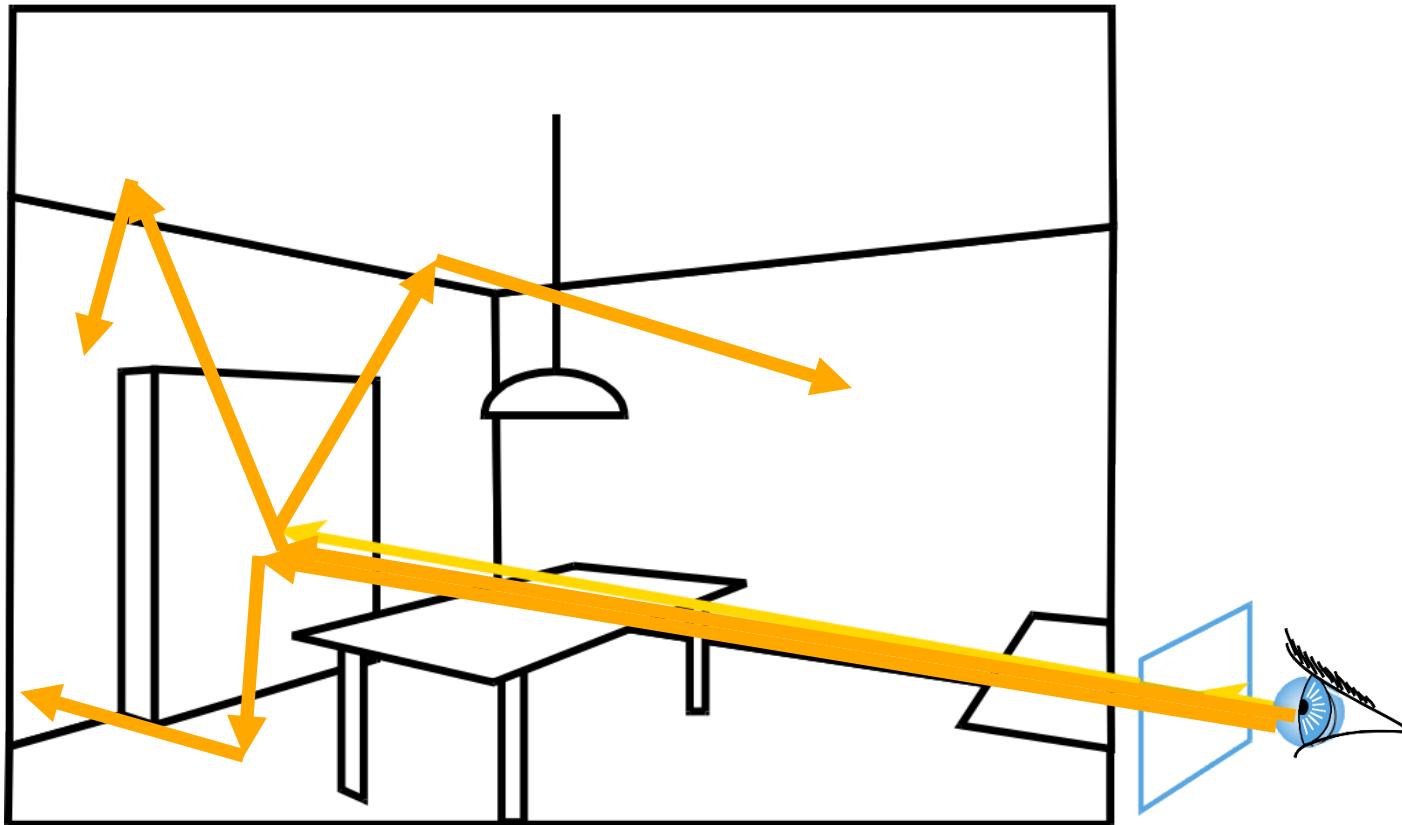
Questions?

- Vintage path tracing by Kajiya

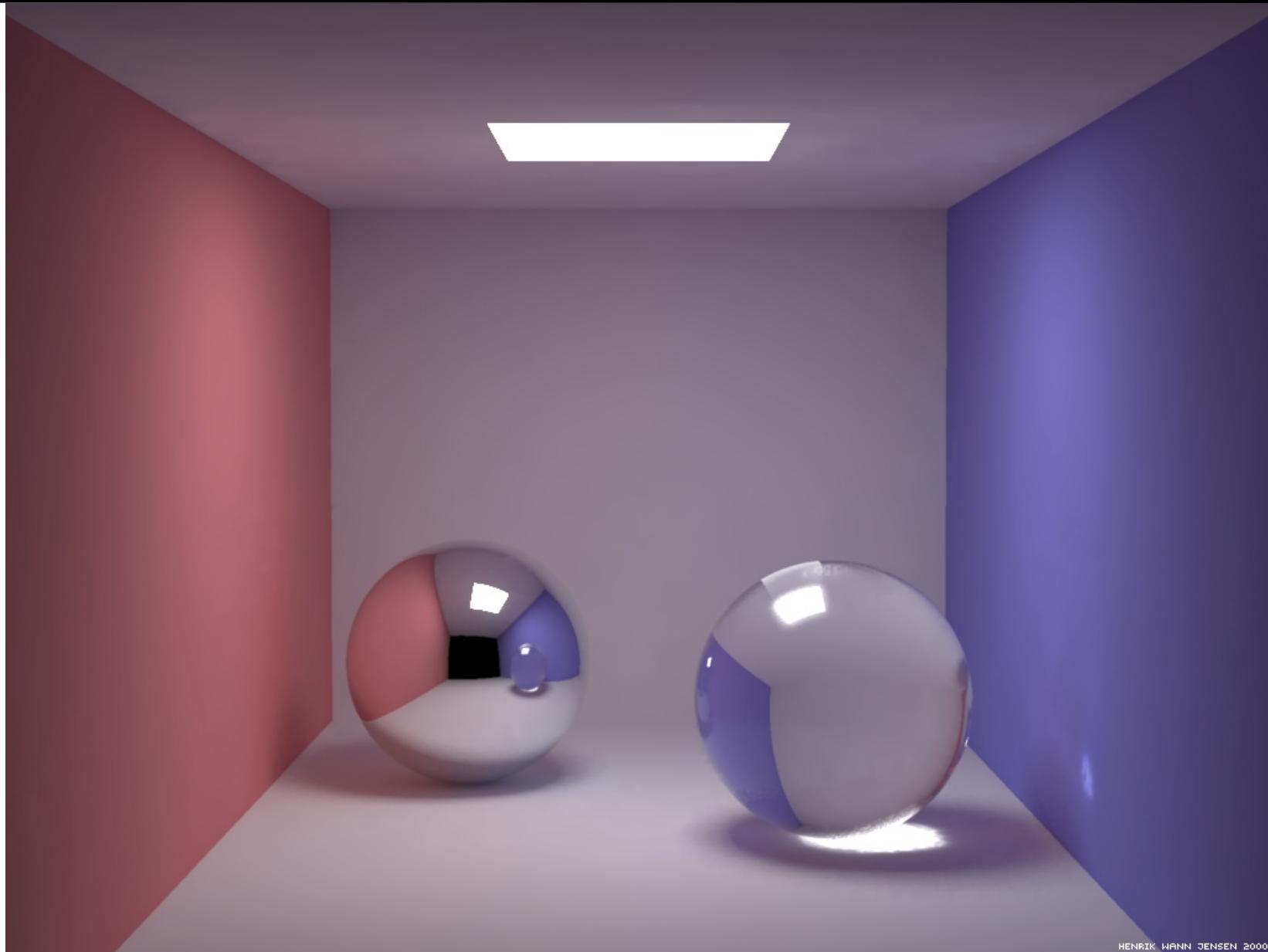


Path Tracing is costly

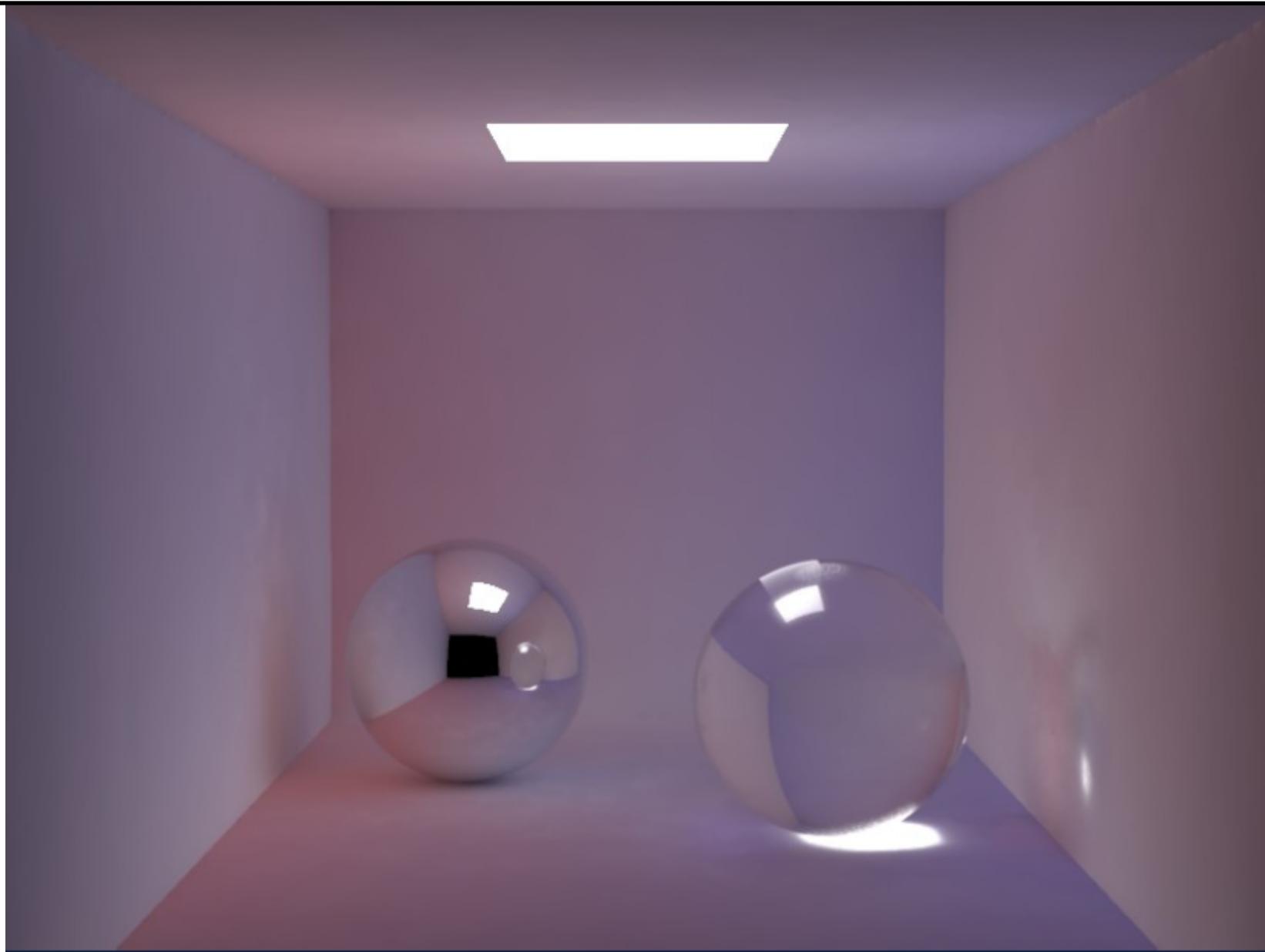
- Needs tons of rays per pixel!



Global Illumination (with Indirect)

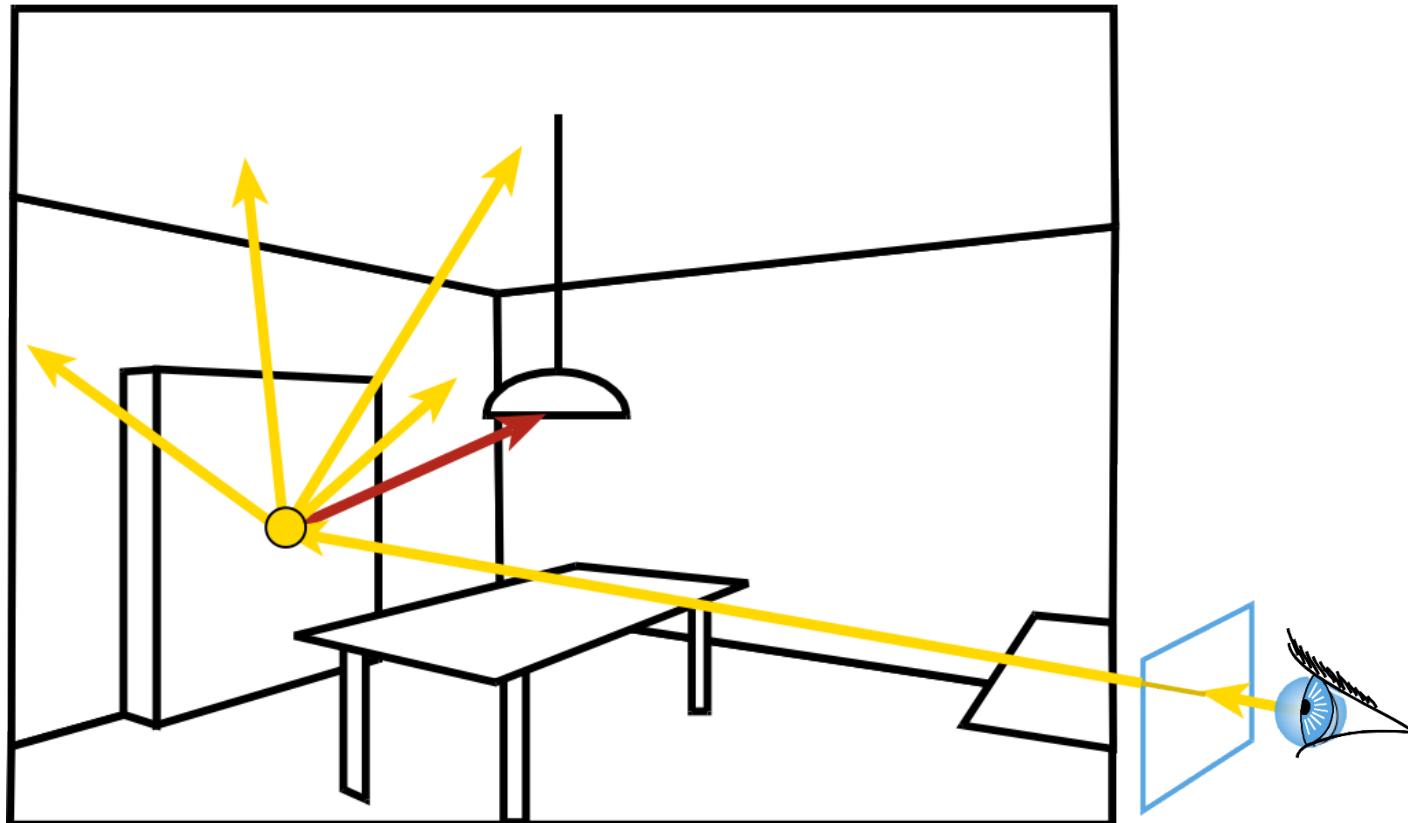


Indirect Lighting is Mostly Smooth



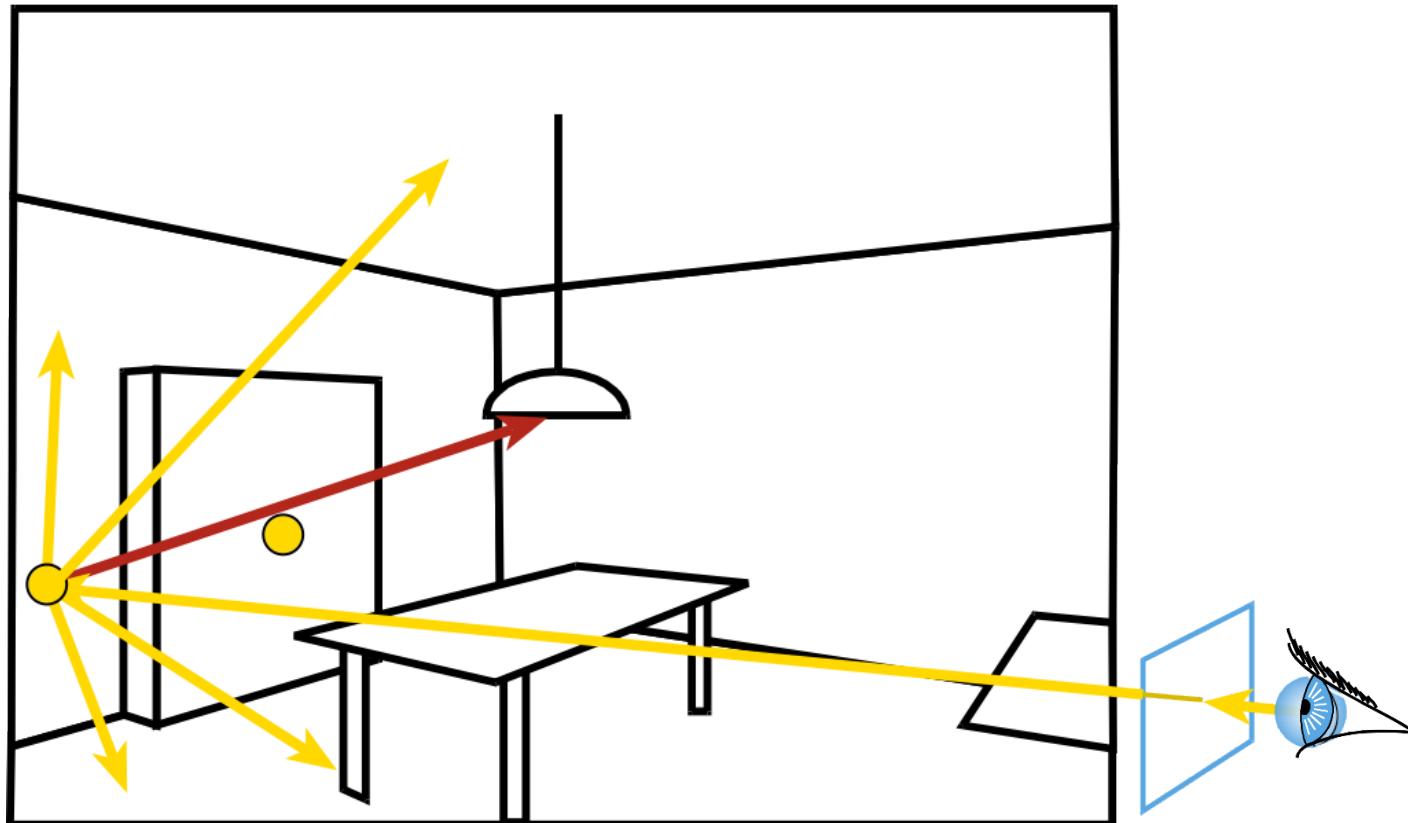
Irradiance Caching

- Indirect illumination is smooth



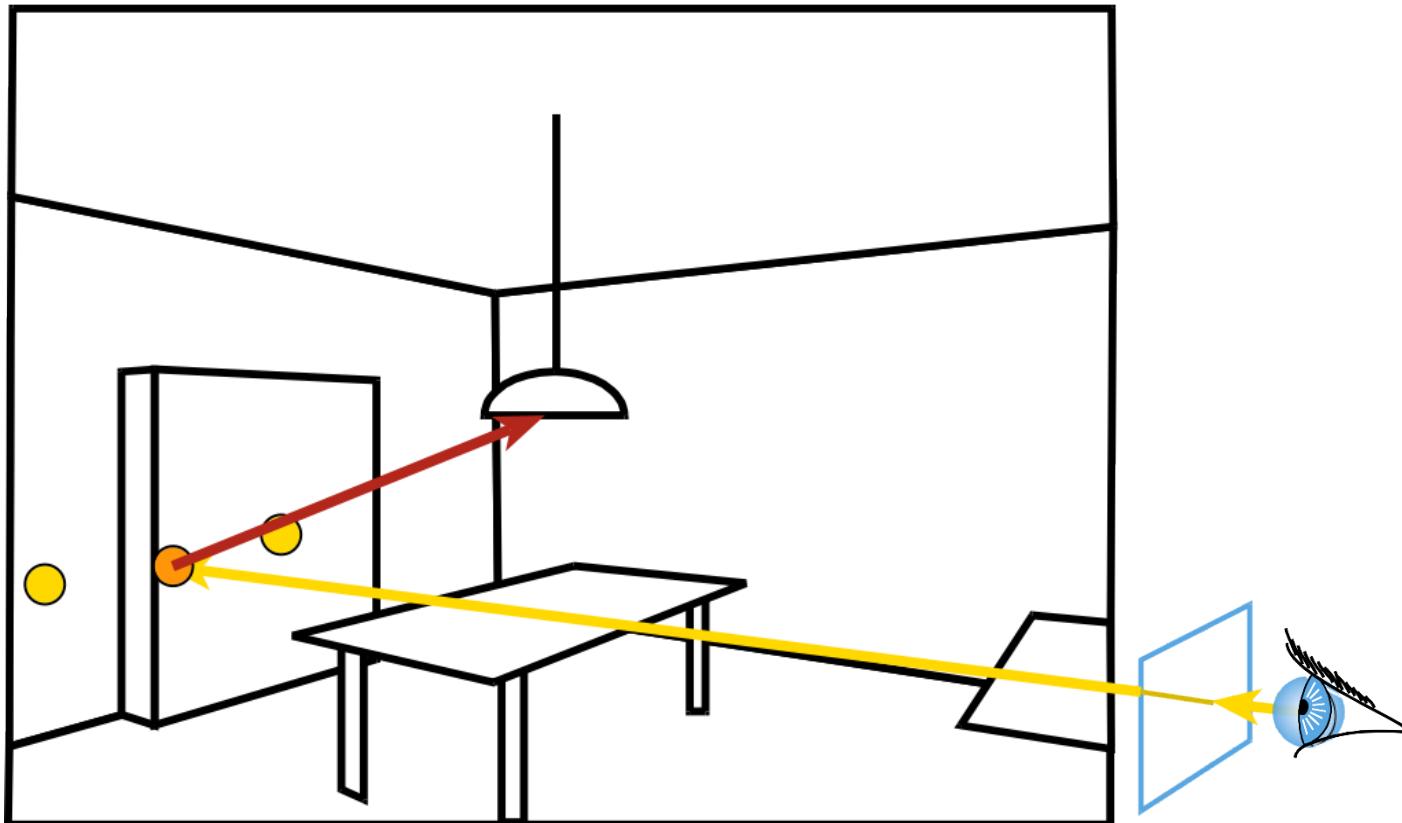
Irradiance Caching

- Indirect illumination is smooth



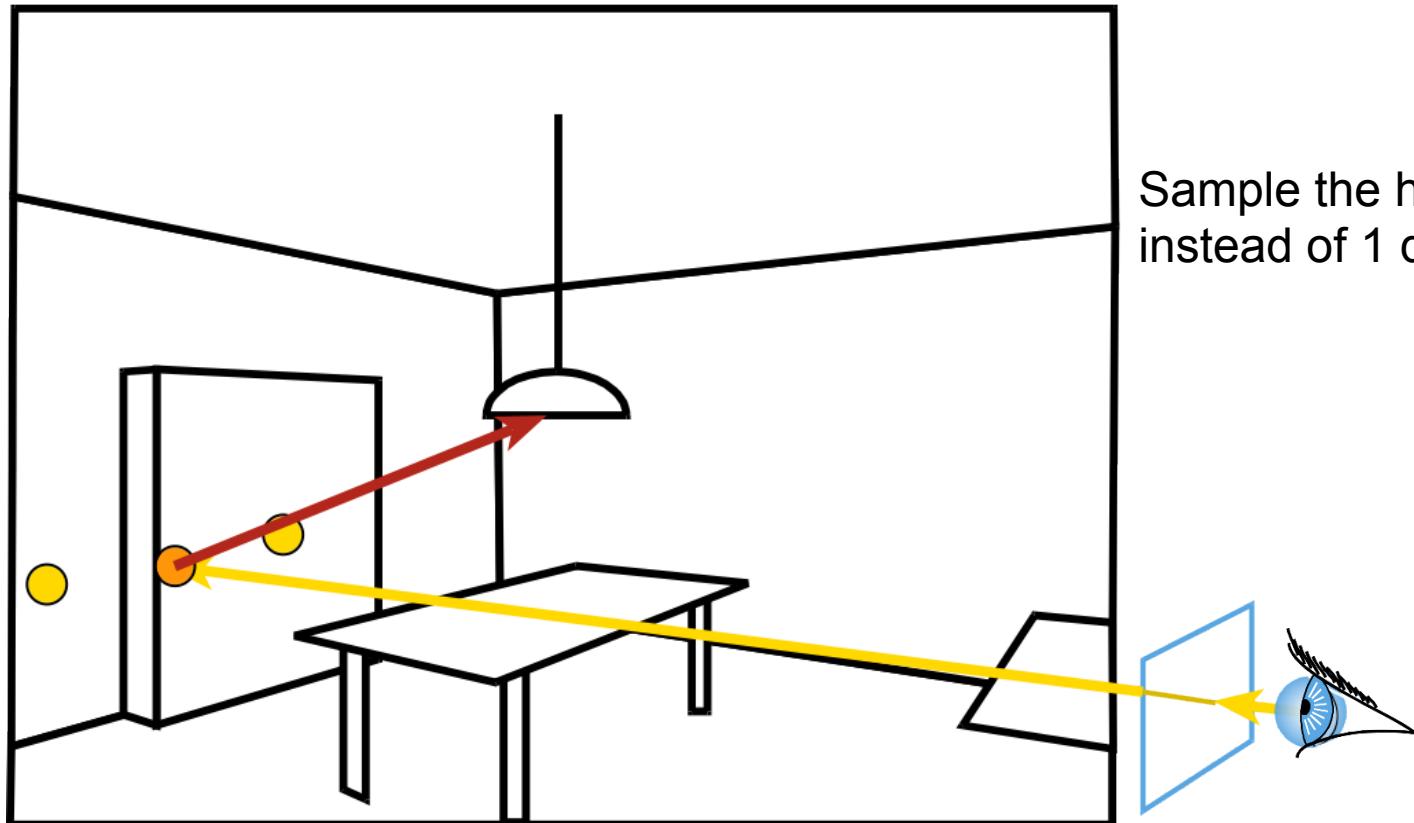
Irradiance Caching

- Indirect illumination is smooth
==> Sample sparsely, interpolate nearby values



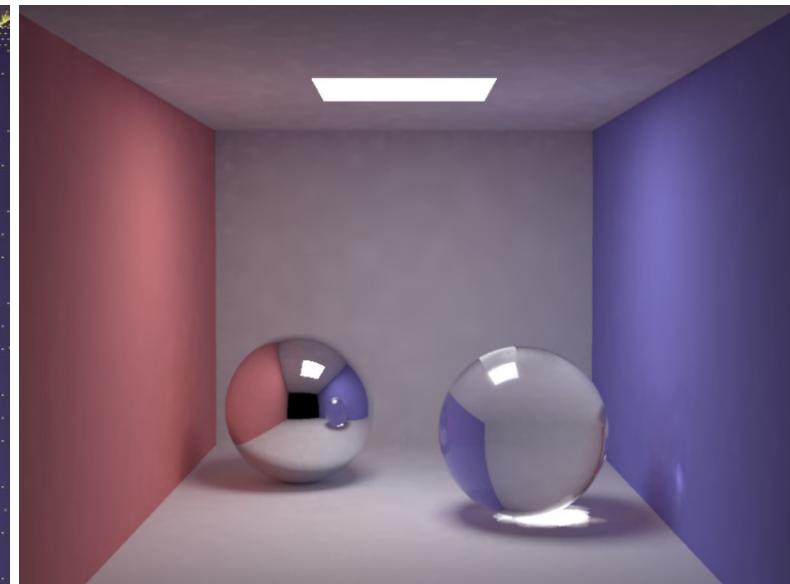
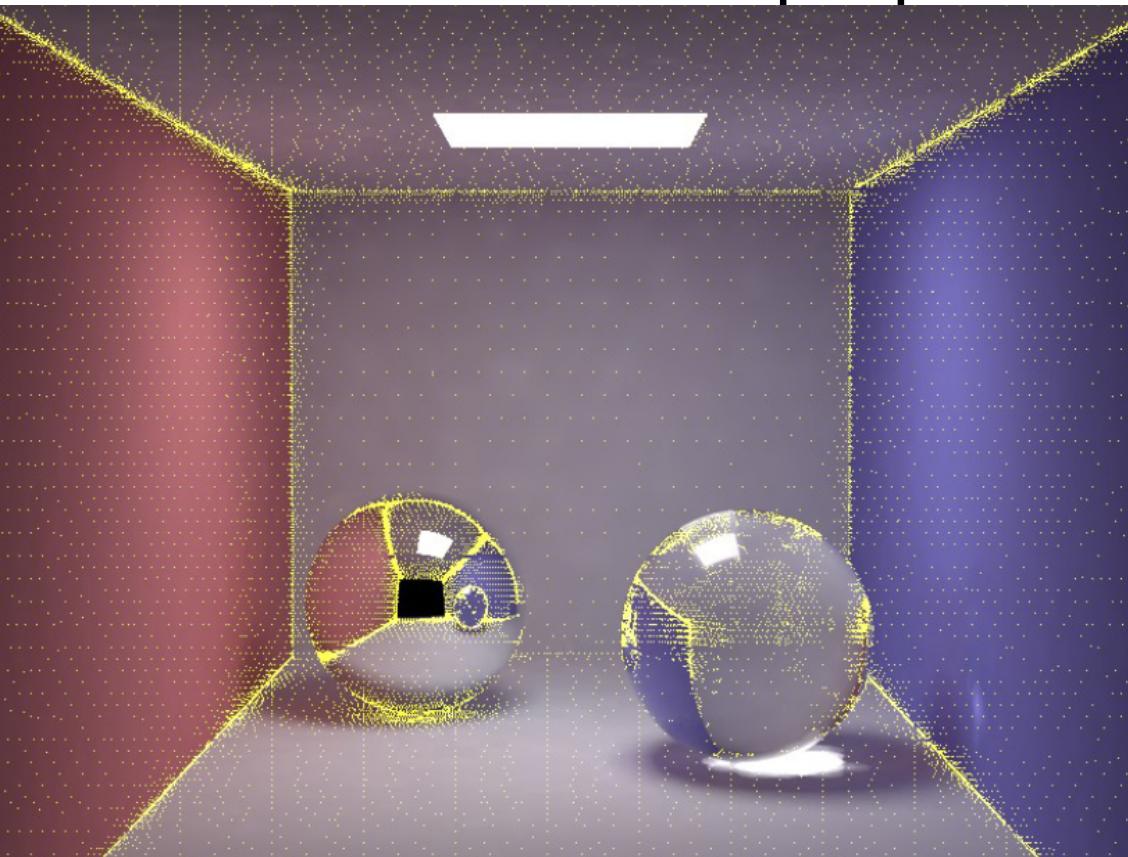
Irradiance Caching

- Store the indirect illumination
- Interpolate existing cached values
- But do full calculation for direct lighting



Irradiance Caching

- Yellow dots:
indirect diffuse sample points



The irradiance cache tries to adapt sampling density to expected frequency content of the indirect illumination (denser sampling near change in geometry)

Radiance by Greg Ward

- The inventor of irradiance caching
- <http://radsite.lbl.gov/radiance/>



What is it? • Reference • Gallery • Download • WWW •

Radiance

Synthetic Imaging System

Open Source Announcement!

Desktop Radiance

Radiance User Interface for Windows

[What is it?](#) | [Reference](#) | [Gallery](#) | [Download](#) | [WWW](#)
[Home](#) | [Webmaster](#) | [Building Technologies Department](#) | [EETD](#) | [BerkeleyLab](#)

Questions?



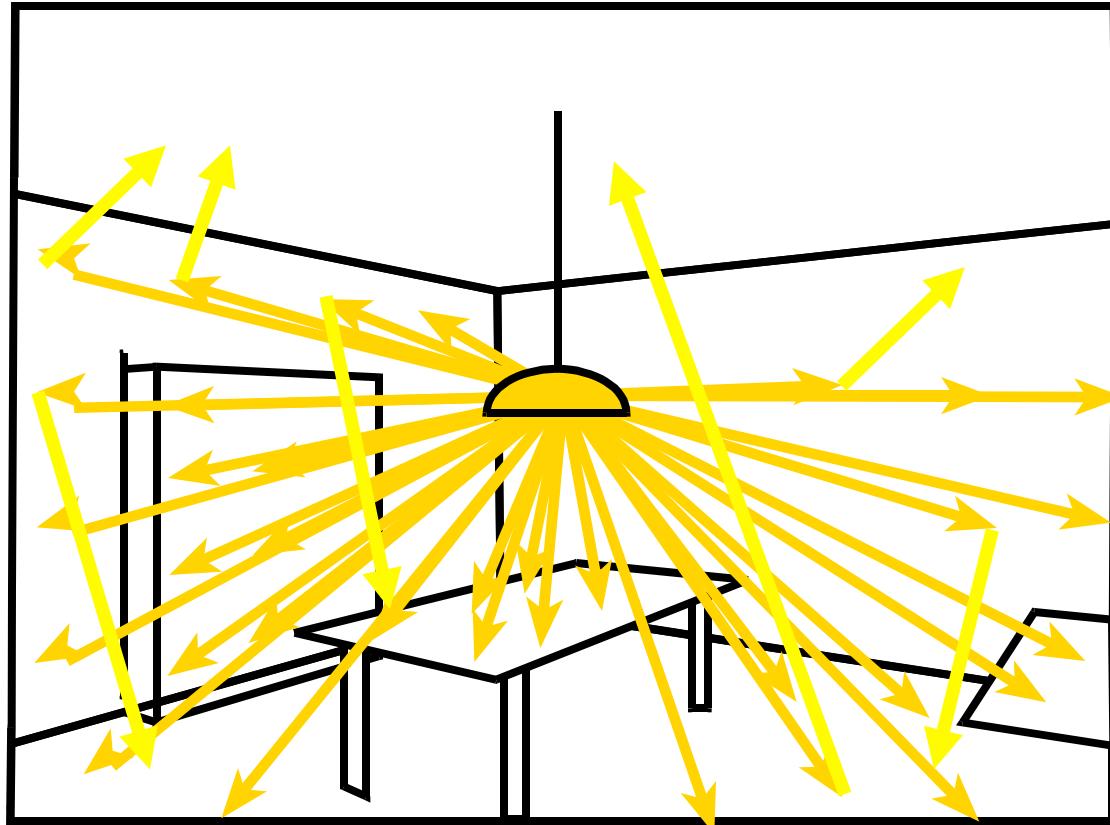
courtesy of Benjamin Brodau

pure www.purerender.com

Image: Pure

Photon Mapping

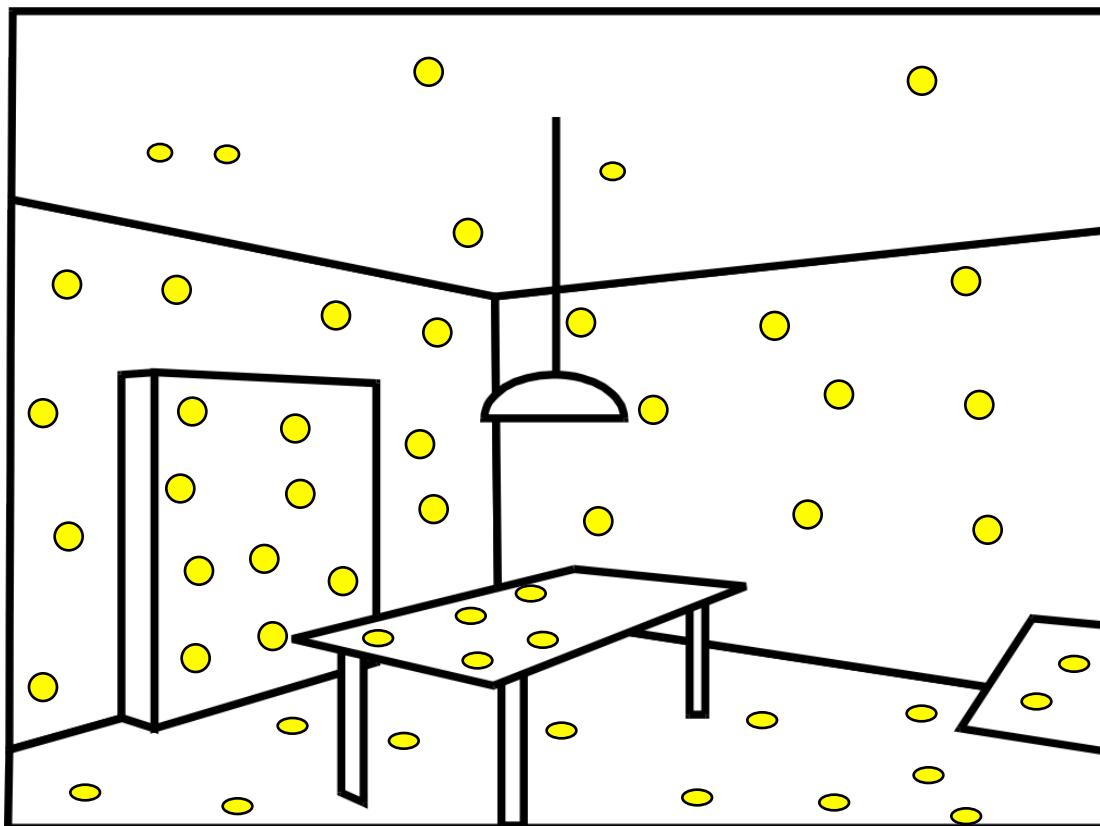
- Preprocess: cast rays from light sources, let them bounce around randomly in the scene
- Store “photons”



Henrik Wann Jensen

Photon Mapping

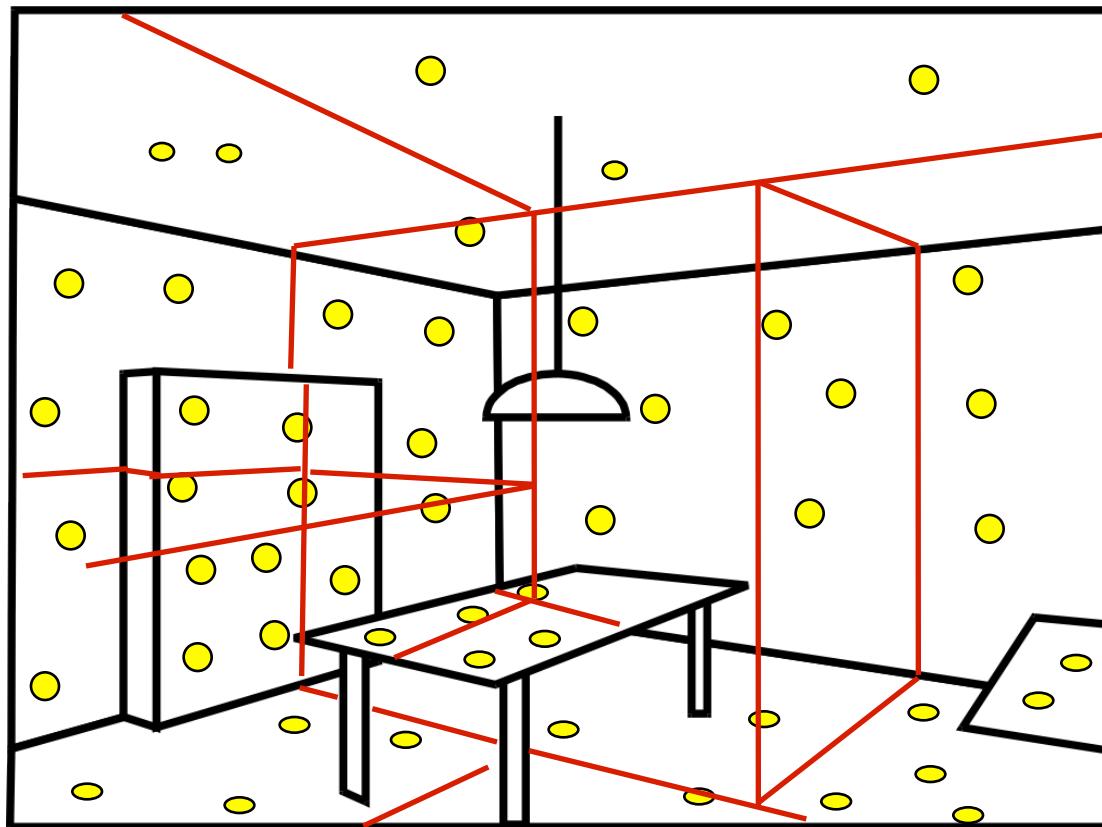
- Preprocess: cast rays from light sources
- Store photons (position + light power + incoming direction)



Approximate light distribution,
independent of any viewpoint

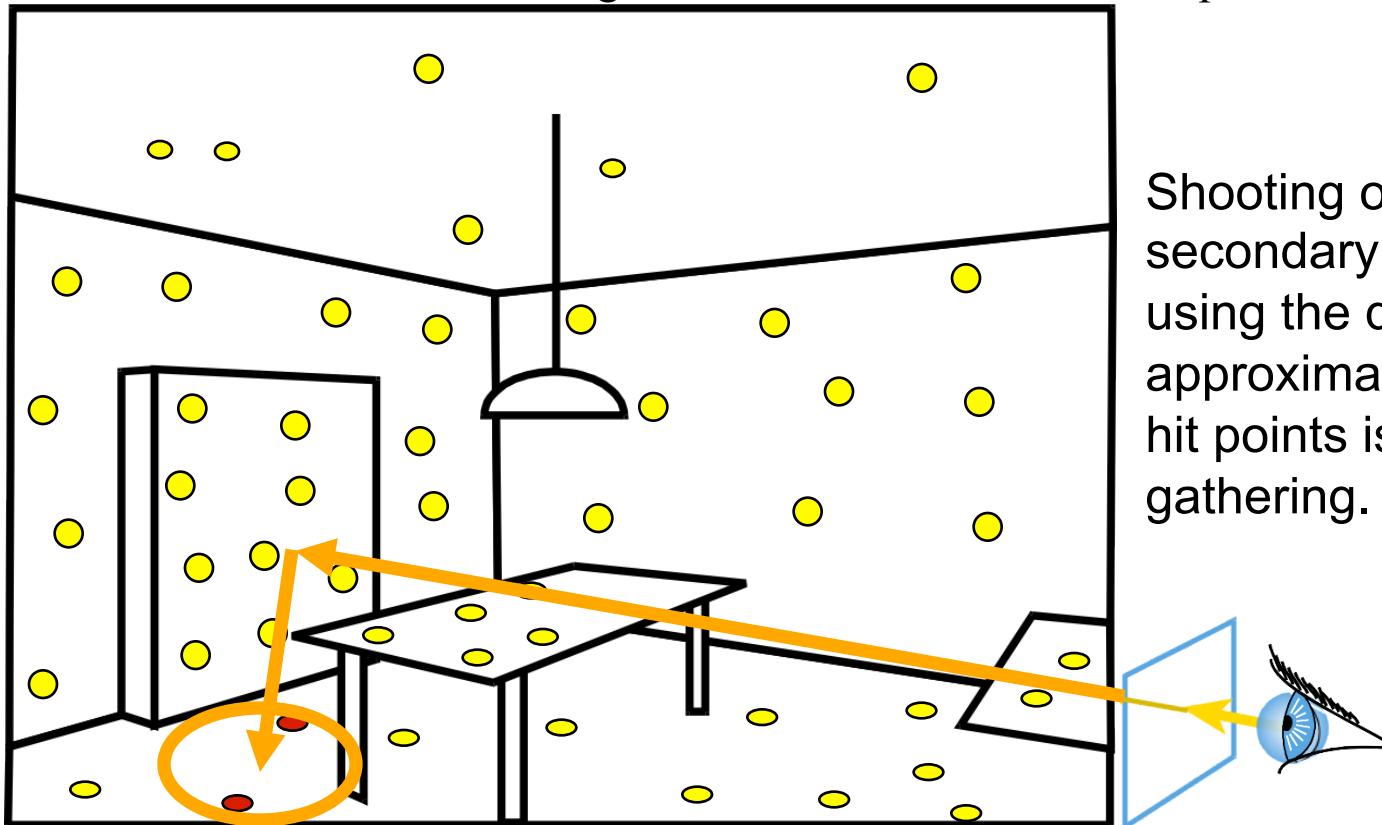
The Photon Map

- Efficiently store photons for fast access
- Use hierarchical spatial structure (kd-tree)



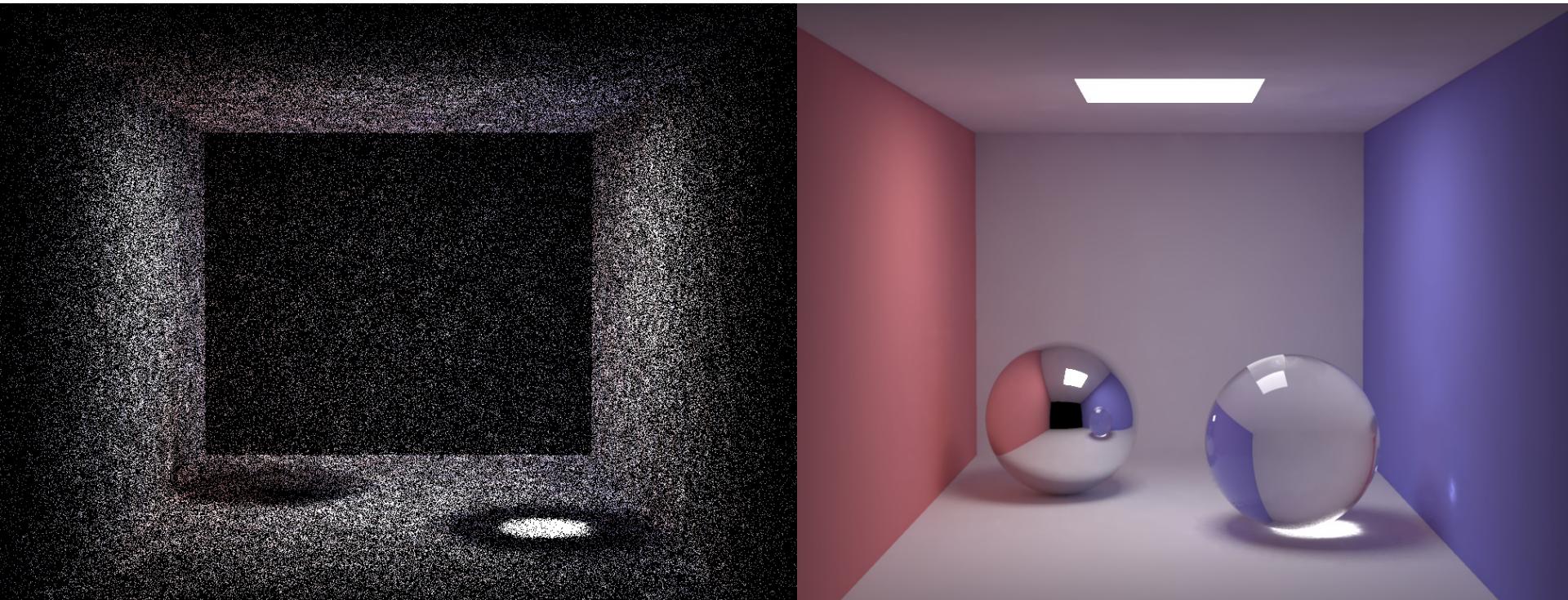
Photon Mapping - Rendering

- Cast primary rays
- For secondary rays
 - reconstruct irradiance using adjacent stored photon
 - Take the k closest photons
- Combine with irradiance caching and a number of other techniques



Shooting one bounce of secondary rays and using the density approximation at those hit points is called final gathering.

Photon Map Results



More Global Illumination Coolness

- Many materials exhibit subsurface scattering
 - Light doesn't just reflect off the surface
 - Light enters, scatters around, and exits at another point
 - Examples: Skin, marble, milk

Images: Jensen et al.



More Subsurface Scattering

Weyrich et al. 2006



Photograph



Rendering

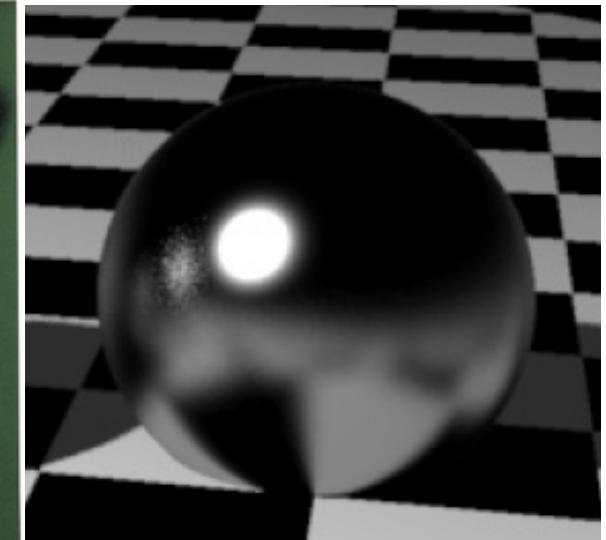
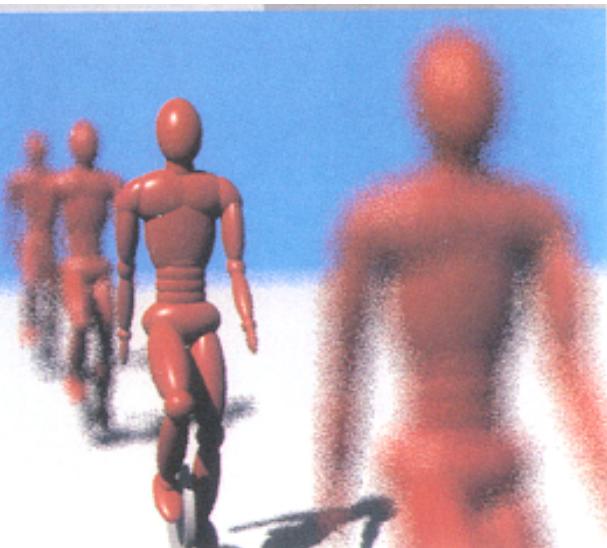
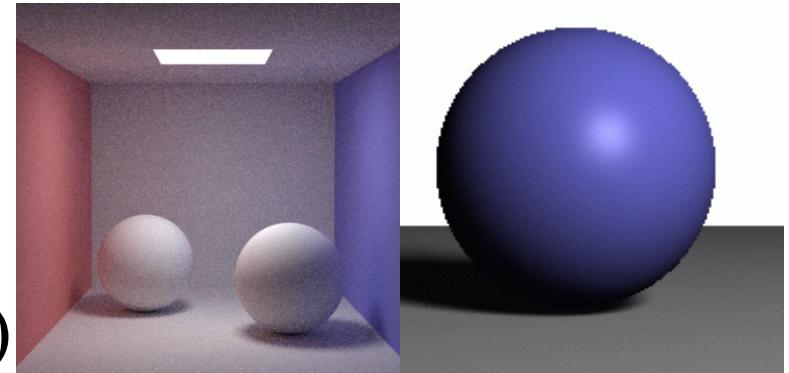
That Was Just the Beginning

- Tons and tons of other Monte Carlo techniques
 - Bidirectional Path Tracing
 - Shoot random paths not just from camera but also light, connect the path vertices by shadow rays
 - Metropolis Light Transport (SIGGRAPH 97')
- And Finite Element Methods
 - Use basis functions instead of random sampling
 - Radiosity (with hierarchies & wavelets)
 - Precomputed Radiance Transfer
- This would warrant a class of its own!

What Else Can We Integrate?

- Pixel: antialiasing
- Light sources: Soft shadows
- Lens: Depth of field
- Time: Motion blur
- BRDF: glossy reflection
- (Hemisphere: indirect lighting)

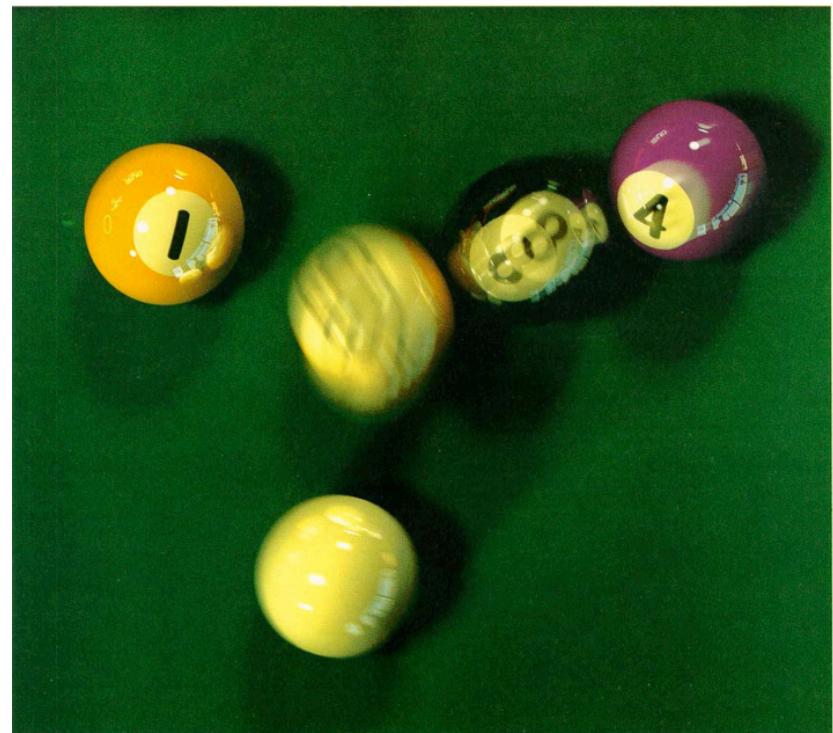
$$\int \int \int \int \int L(x, y, t, u, v) dx dy dt du dv$$



Domains of Integration

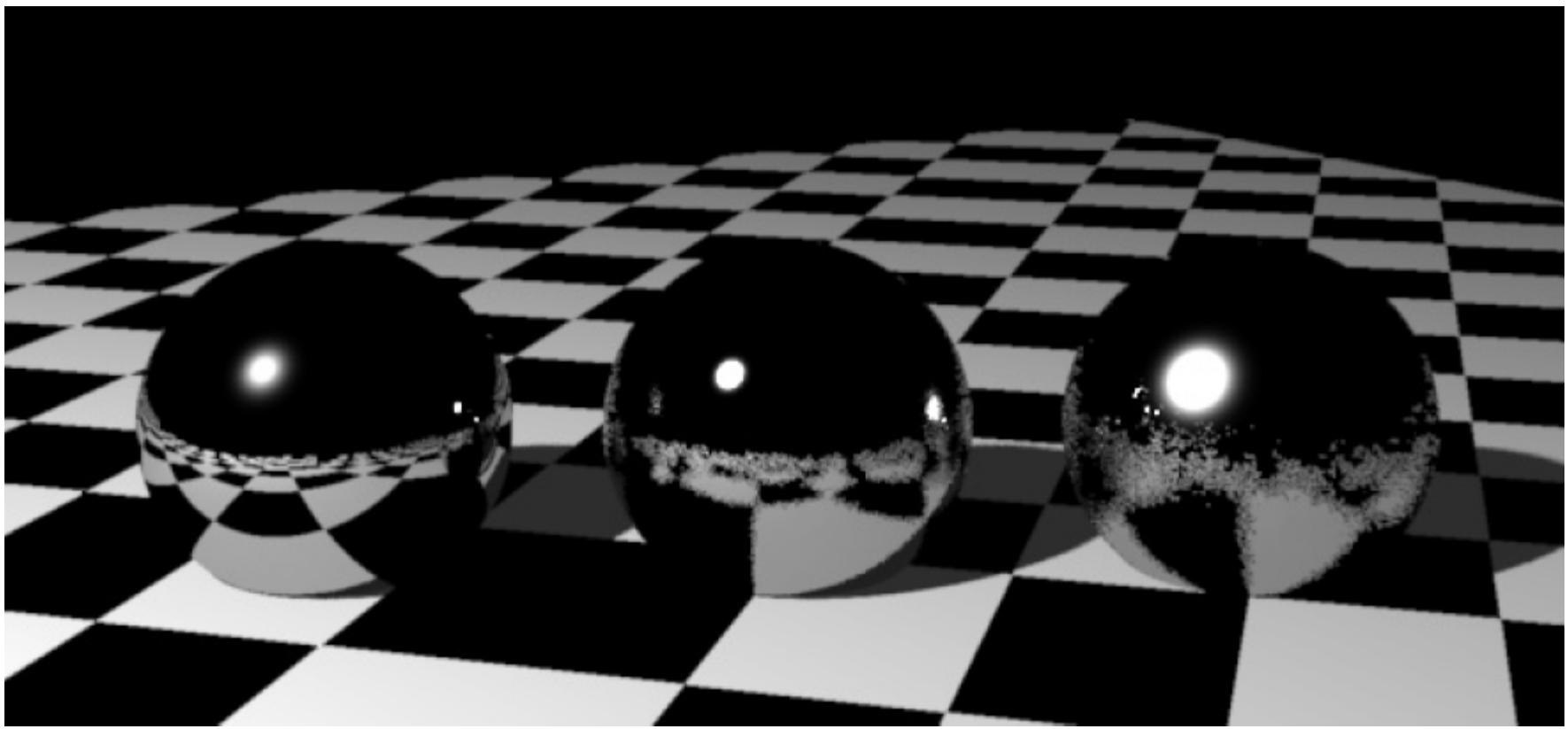
- Pixel, lens (Euclidean 2D domain)
 - Antialiasing filters, depth of field
- Time (1D)
 - Motion blur
- Hemisphere
 - Indirect lighting
- Light source
 - Soft shadows

Famous motion blur image
from [Cook et al. 1984](#)



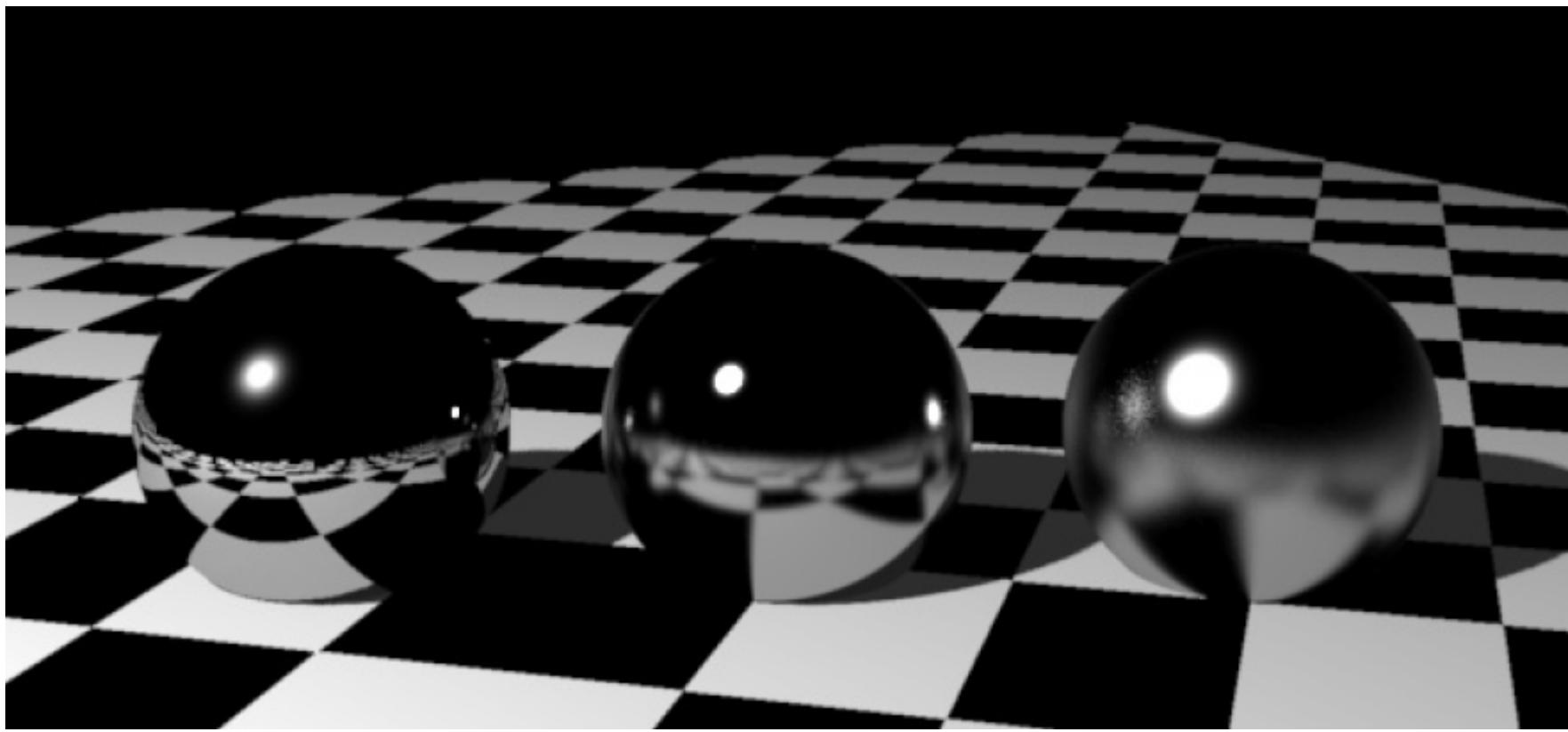
Motivational Eye Candy

- Rendering glossy reflections
- Random reflection rays around mirror direction
 - 1 sample per pixel



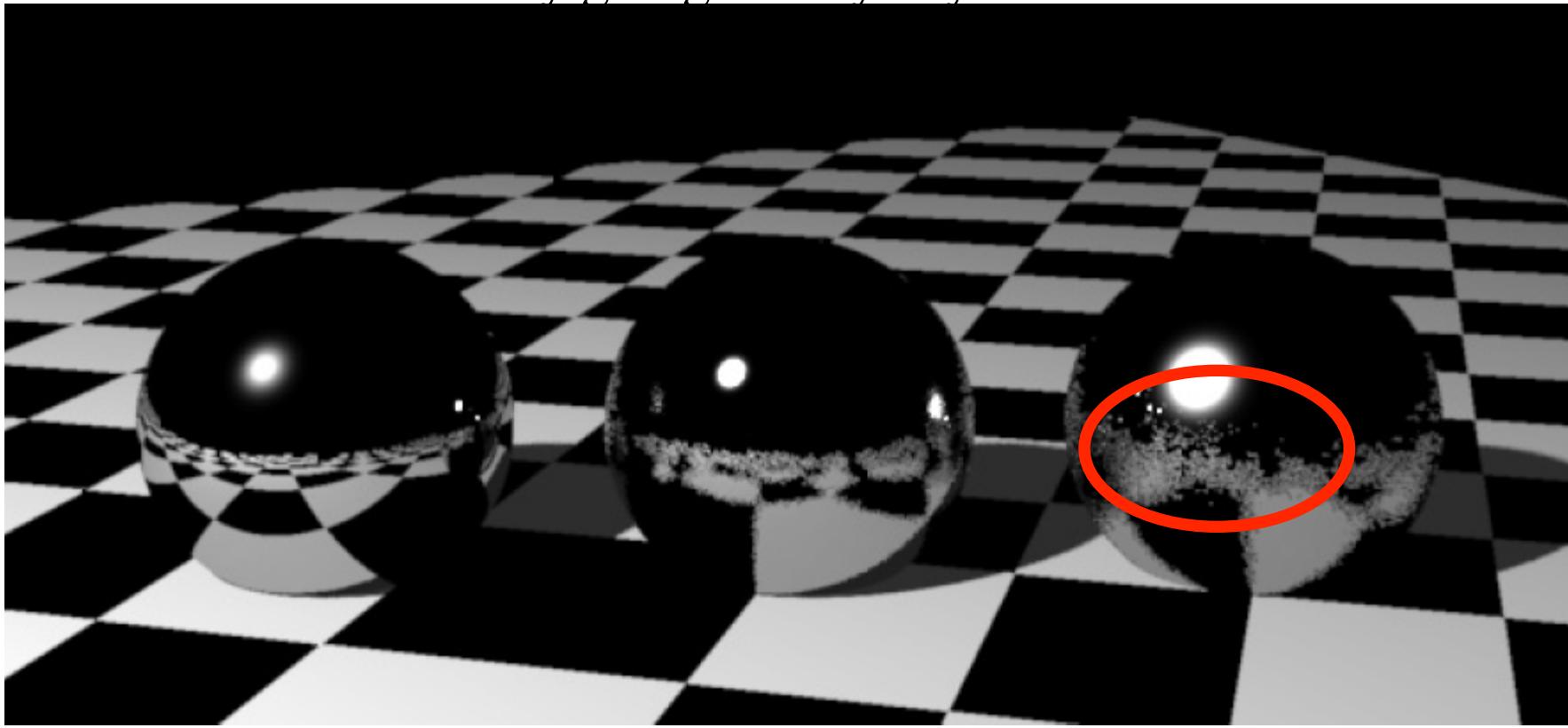
Motivational Eye Candy

- Rendering glossy reflections
- Random reflection rays around mirror direction
 - 256 samples per pixel



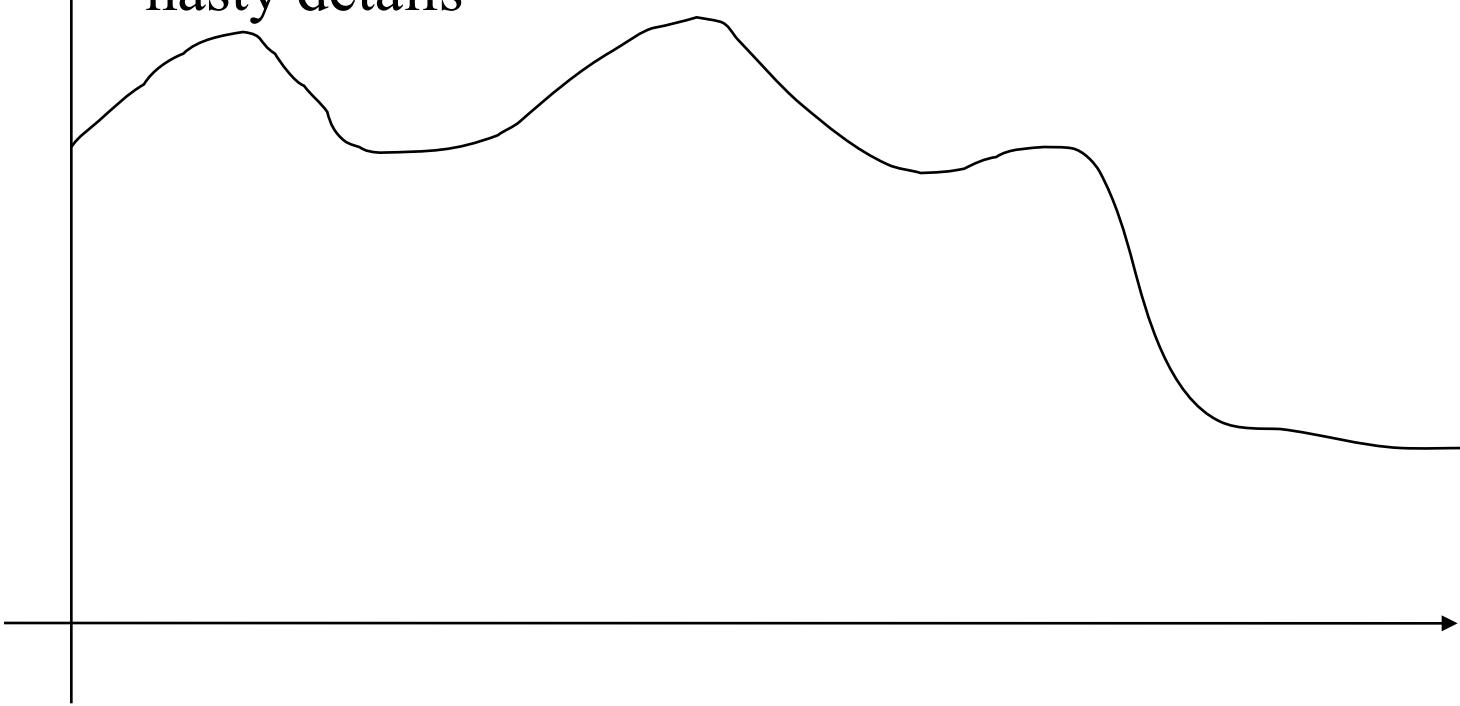
Error/noise Results in Variance

- We use random rays
 - Run the algorithm again → get different image
- What is the noise/variance/standard deviation?
 - And what's really going on anyway?



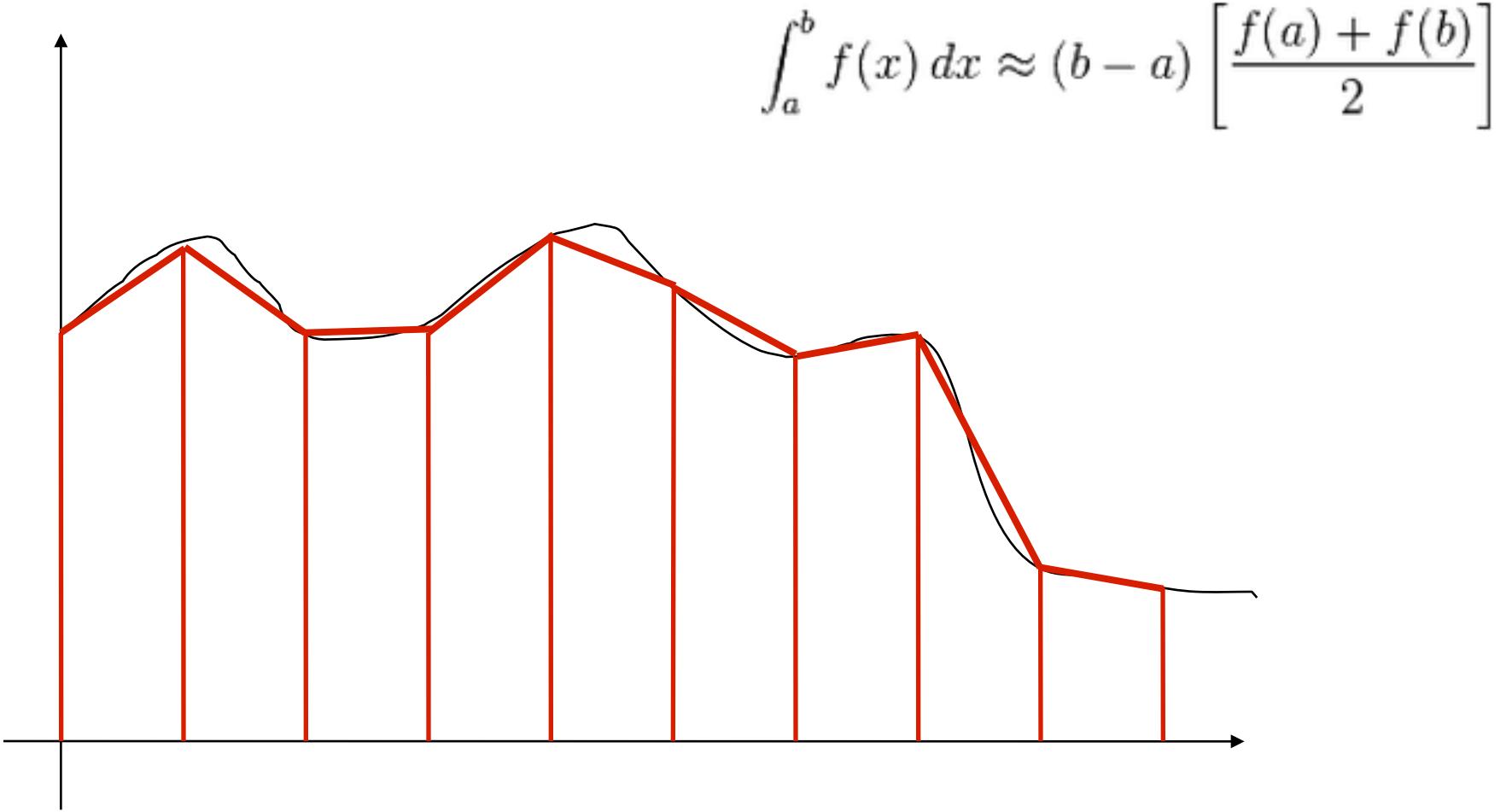
Integration

- Compute integral of arbitrary function
 - e.g. integral over area light source, over hemisphere, etc.
- Continuous problem → we need to discretize
 - Analytic integration never works because of visibility and other nasty details



Integration

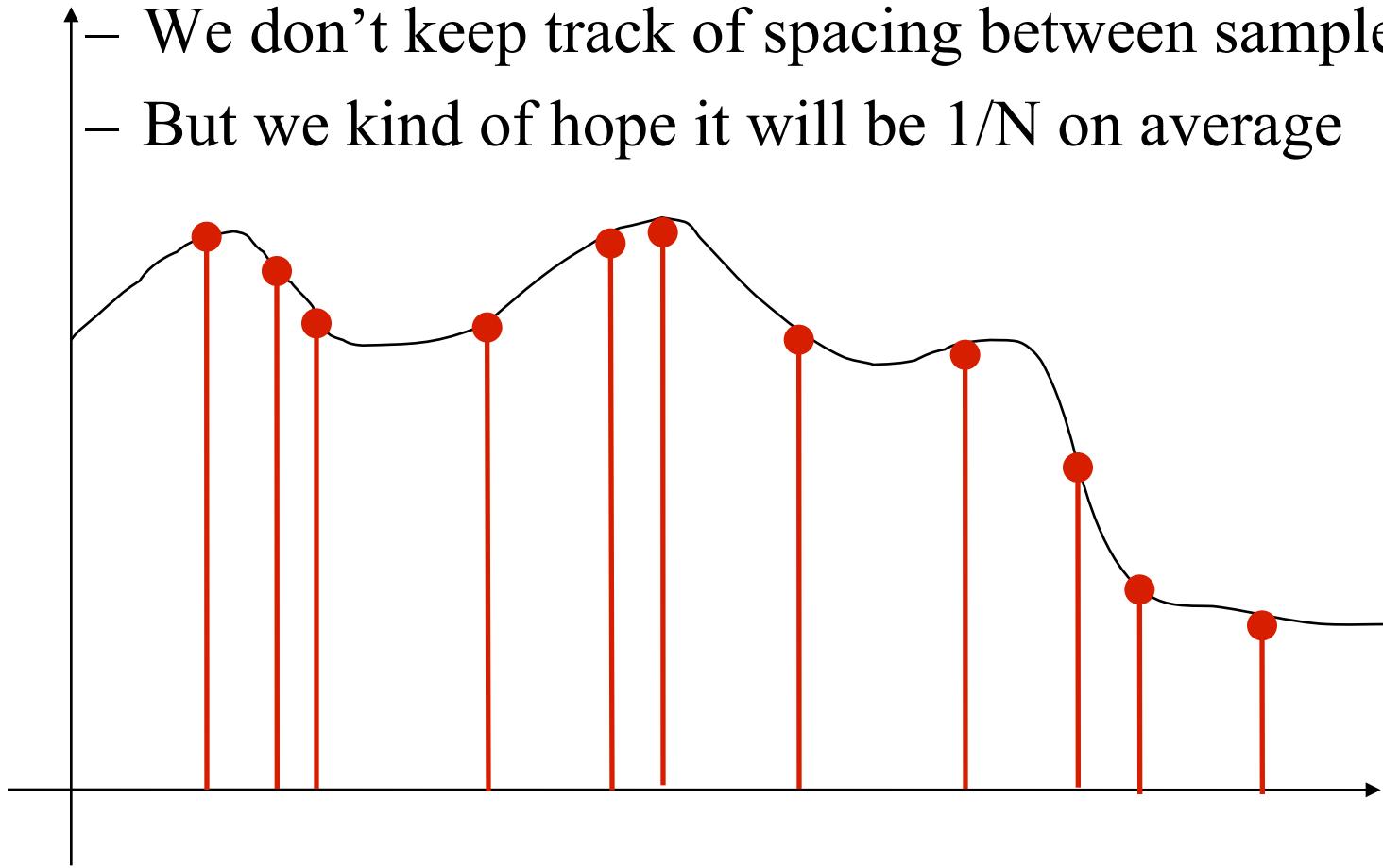
- You know trapezoid, Simpson's rule, etc.



Monte Carlo Integration

- Monte Carlo integration: use random samples and compute average

- We don't keep track of spacing between samples
 - But we kind of hope it will be $1/N$ on average



Monte Carlo Integration

$$\int_S f(x) dx \approx \frac{\text{Vol}(S)}{N} \sum_{i=1}^N f(x_i)$$

- S is the integration domain
 - $\text{Vol}(S)$ is the volume (measure) of S
 - $\{x_i\}$ are independent, identically distributed (i.i.d) random points in S
- $$\text{Vol}(S) = \int_S dx$$

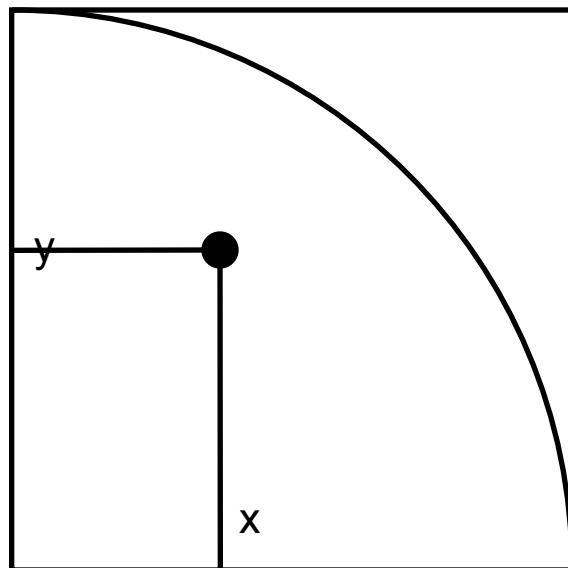
Monte Carlo Integration

$$\int_S f(x) dx \approx \frac{\text{Vol}(S)}{N} \sum_{i=1}^N f(x_i)$$

- S is the integration domain
 - Vol(S) is the volume (measure) of S
- $\{x_i\}$ are independent, identically distributed (i.i.d) random points in S
- The integral is the average of f times the volume of S
- Variance is proportional to $1/N$
 - Avg. error is proportional $1/\sqrt{N}$
 - To halve error, need 4x samples

Monte Carlo Computation of π

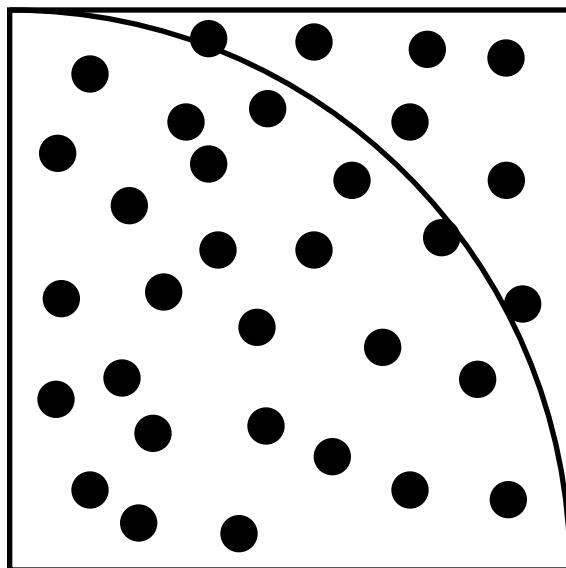
- Take a square
- Take a random point (x,y) in the square
- Test if it is inside the $\frac{1}{4}$ disc $(x^2+y^2 < 1)$
- The probability is $\pi / 4$



Integral of the function that is one inside the circle, zero outside

Monte Carlo Computation of π

- The probability is $\pi / 4$
- Count the inside ratio $n = \# \text{ inside} / \text{total } \# \text{ trials}$
- $\pi \approx n * 4$
- The error depends on the number of trials



Demo

```
def piMC(n):  
    success = 0  
    for i in range(n):  
        x=random.random()  
        y=random.random()  
        if x*x+y*y<1: success = success+1  
    return 4.0*float(success)/float(n)
```

Why Not Use Simpson Integration?

- You're right, Monte Carlo is not very efficient for computing π
- When is it useful?
 - High dimensions: Convergence is independent of dimension!
 - For d dimensions, Simpson requires N^d domains (!!)
 - Similar explosion for other quadratures (Gaussian, etc.)

Advantages of MC Integration

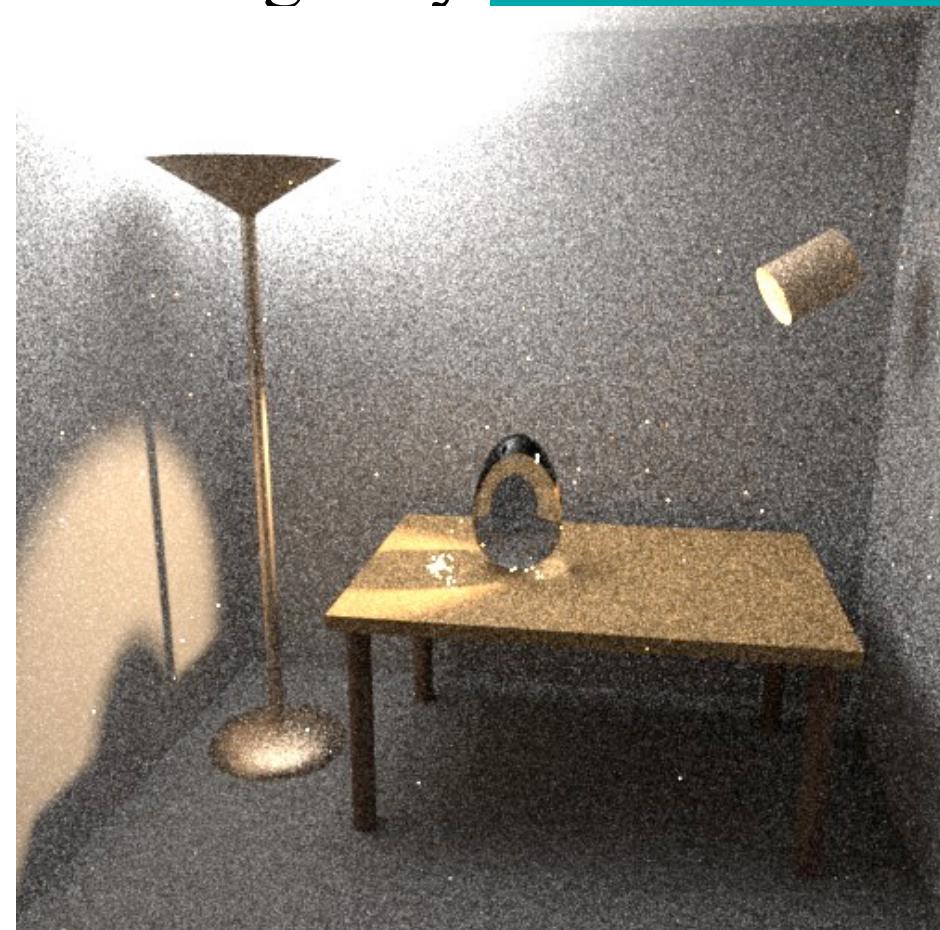
- Few restrictions on the integrand
 - Doesn't need to be continuous, smooth, ...
 - Only need to be able to evaluate at a point
- Extends to high-dimensional problems
 - Same convergence
- Conceptually straightforward
- Efficient for solving at just a few points

Disadvantages of MC

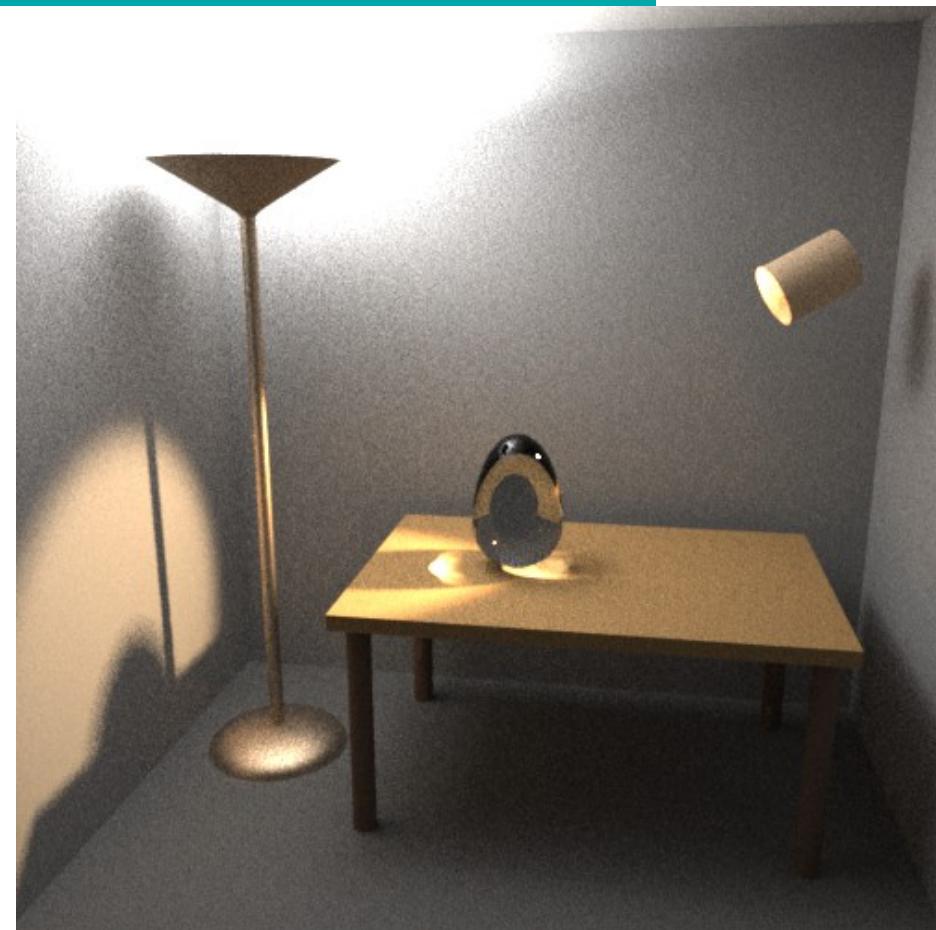
- Noisy
- Slow convergence
- Good implementation is hard
 - Debugging code
 - Debugging math
 - Choosing appropriate techniques

Questions?

- Images by Veach and Guibas, SIGGRAPH 95



Naïve sampling strategy



Optimal sampling strategy

Hmmh...

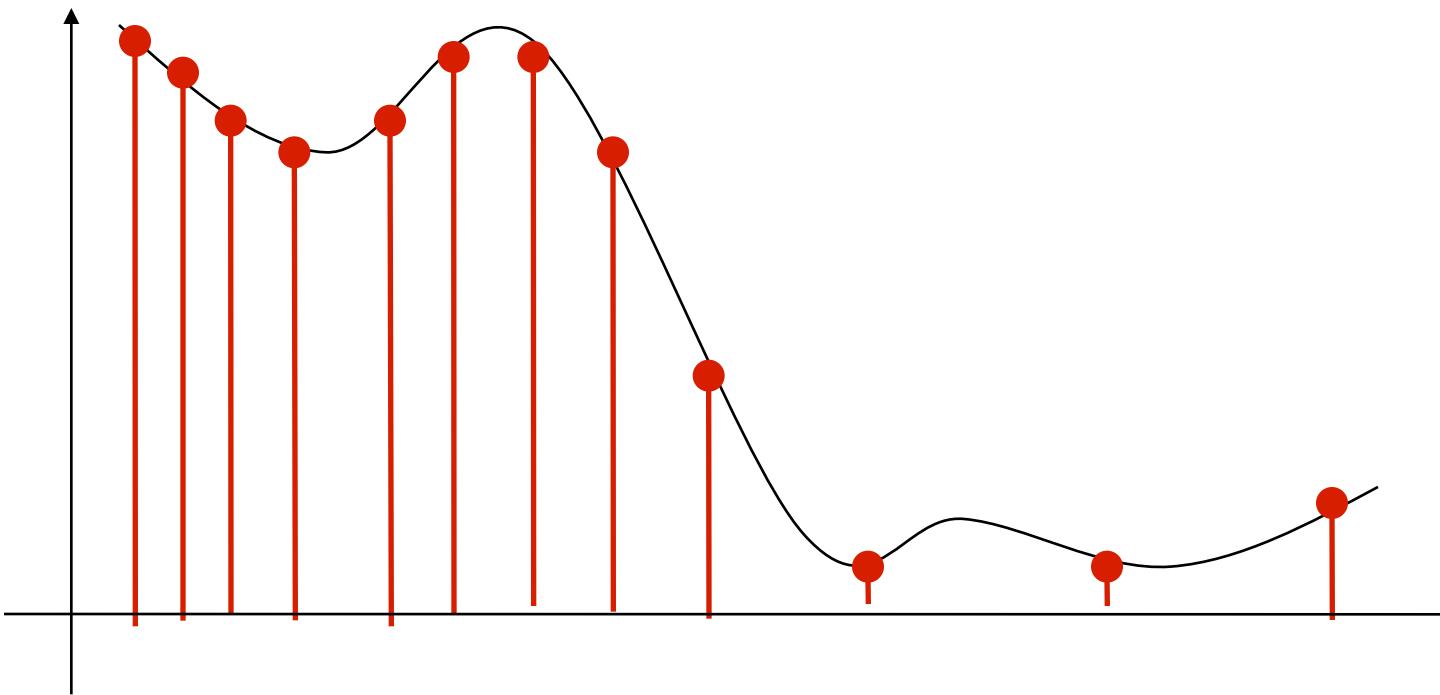
- Are uniform samples the best we can do?

Smarter Sampling

Sample a non-uniform probability

Called “importance sampling”

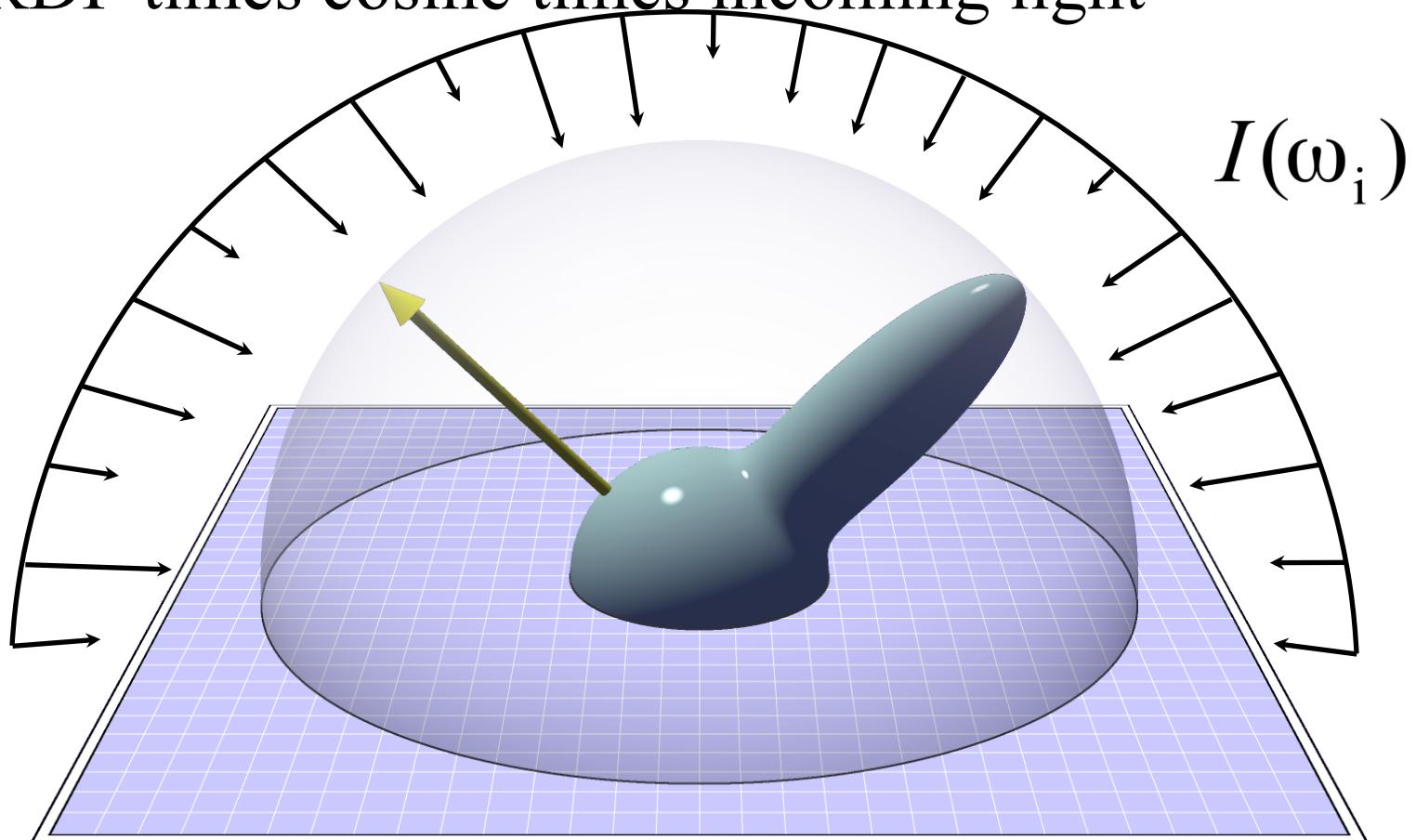
Intuitive justification: Sample more in places where there are likely to be larger contributions to the integral



Example: Glossy Reflection

Slide courtesy of Jason Lawrence

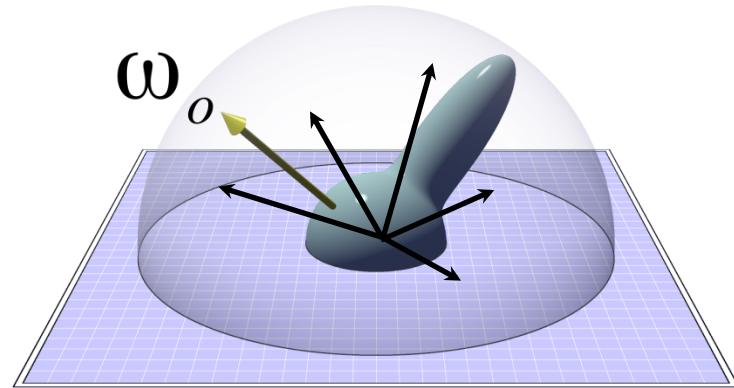
- Integral over hemisphere
- BRDF times cosine times incoming light



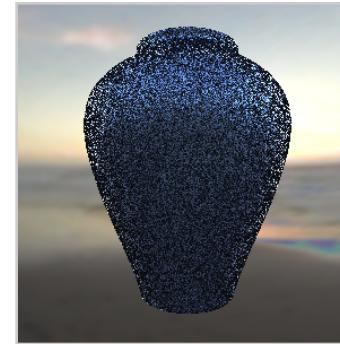
Sampling a BRDF

Slide courtesy of Jason Lawrence

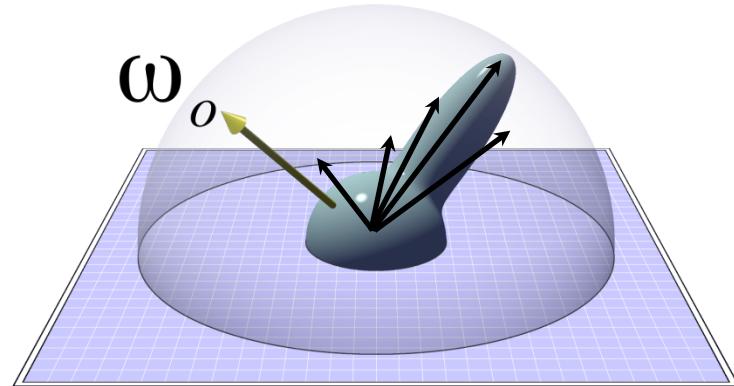
$U(\omega_i)$



5 Samples/Pixel



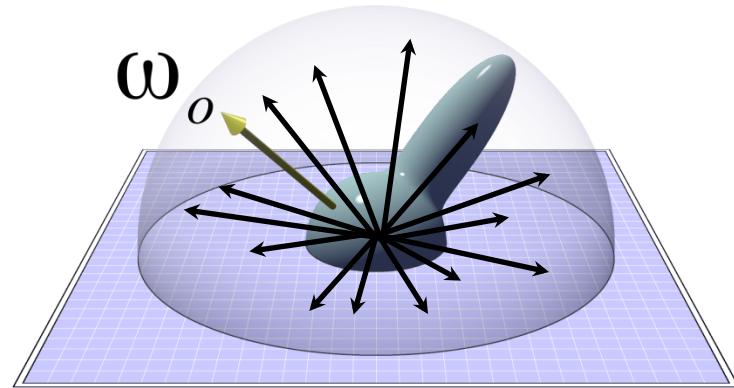
$P(\omega_i)$



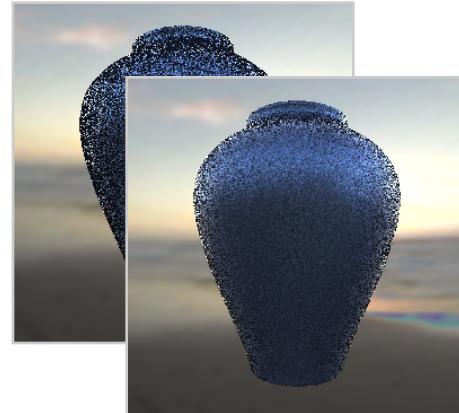
Sampling a BRDF

Slide courtesy of Jason Lawrence

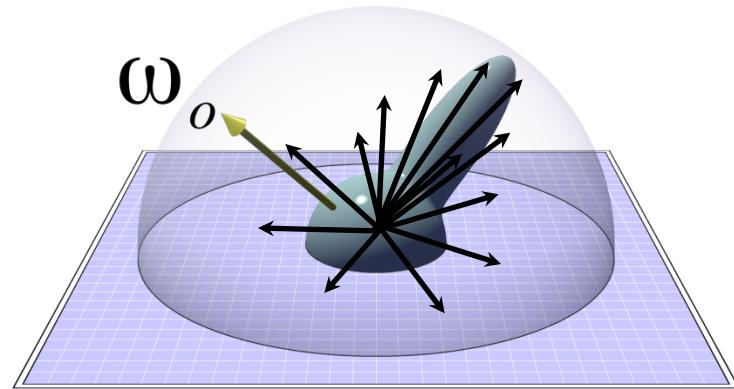
$$U(\omega_i)$$



25 Samples/Pixel



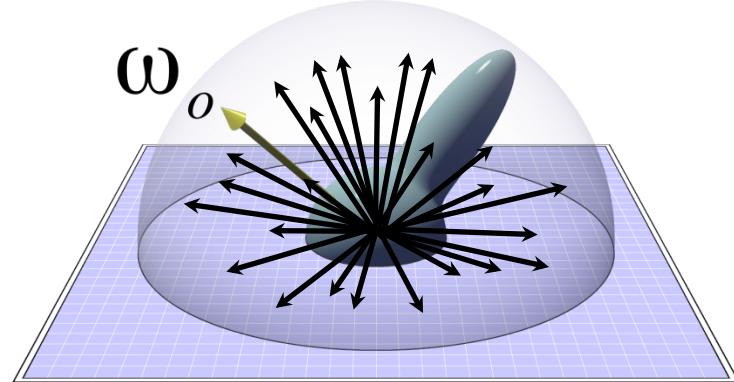
$$P(\omega_i)$$



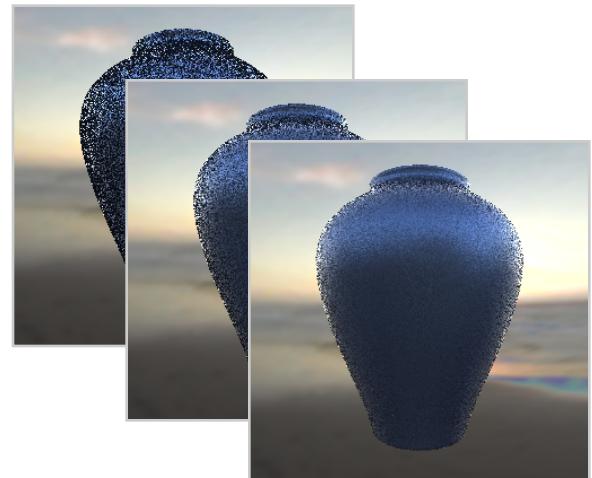
Sampling a BRDF

Slide courtesy of Jason Lawrence

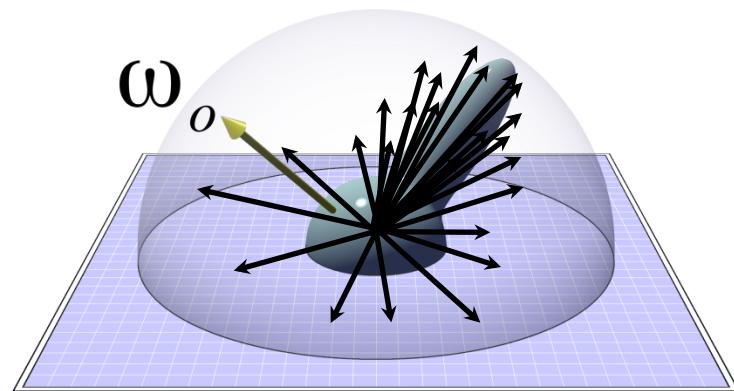
$$U(\omega_i)$$



75 Samples/Pixel



$$P(\omega_i)$$

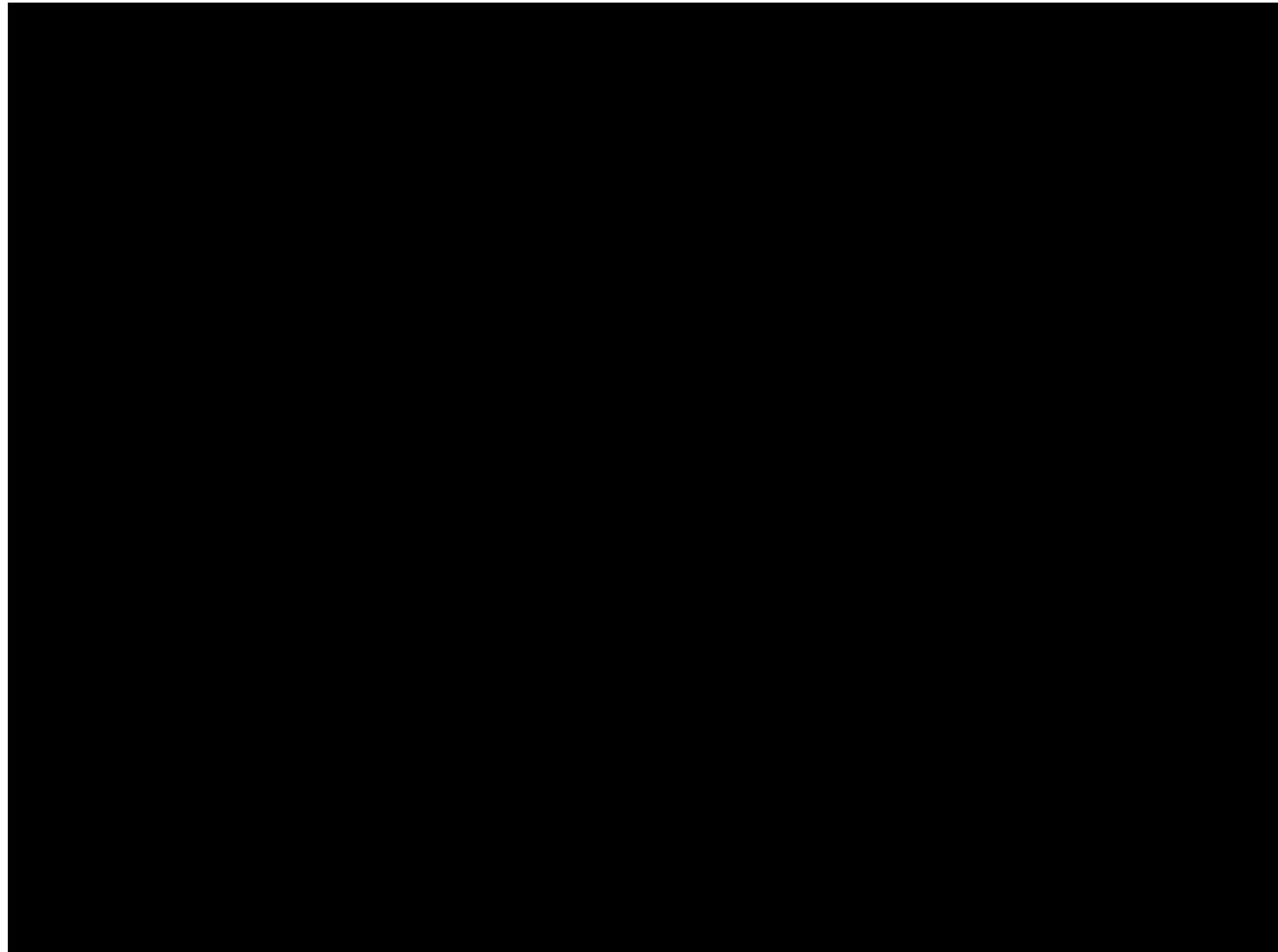


Importance Sampling Math

$$\int_S f(x) dx \approx \frac{\text{Vol}(S)}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}$$

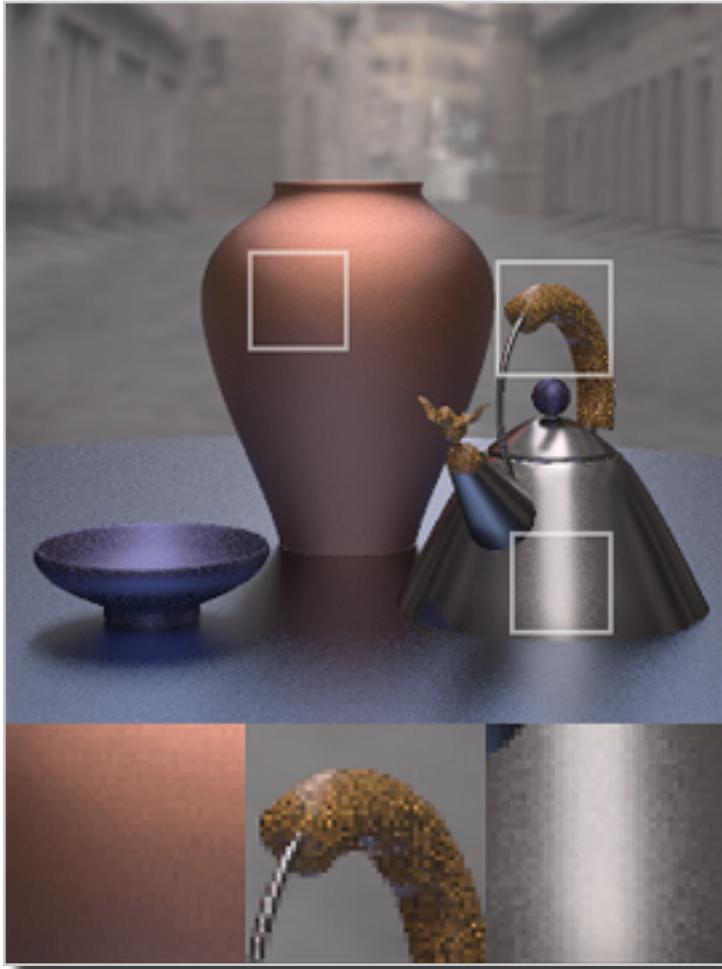
- Like before, but now $\{x_i\}$ are not uniform but drawn according to a probability distribution p
 - Uniform case reduces to this with $p(x) = \text{const.}$
- The problem is designing p_s that are easy to sample from and mimic the behavior of f

Monte Carlo Path Tracing

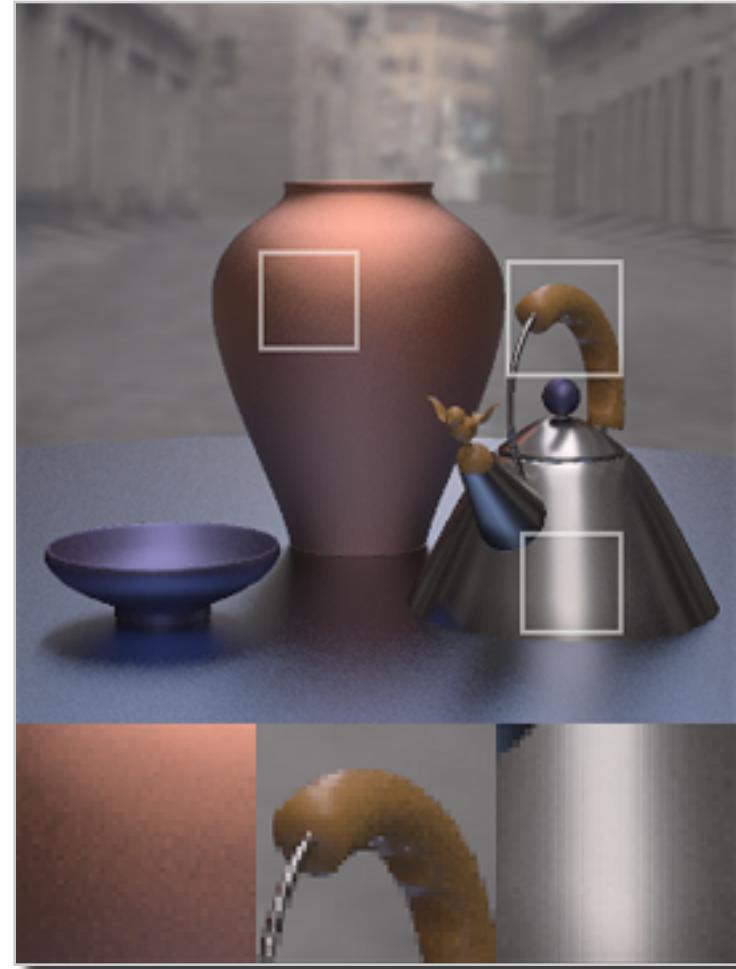


Questions?

1200 Samples/Pixel



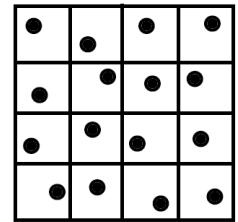
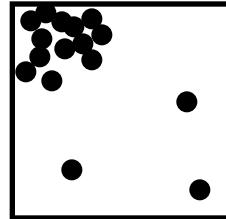
Traditional importance function



Better importance by Lawrence et al.

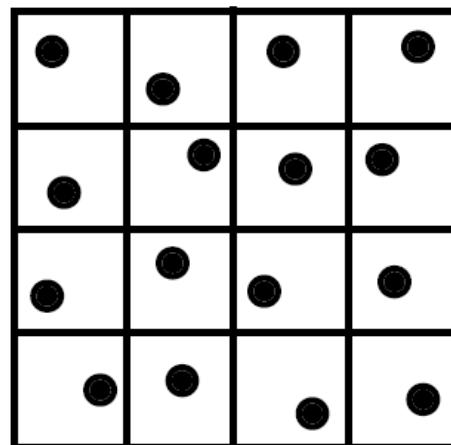
Stratified Sampling

- With uniform sampling, we can get unlucky
 - E.g. all samples clump in a corner
 - If we don't know anything of the integrand, we want a relatively uniform sampling
 - Not regular, though, because of aliasing!
- To prevent clumping, subdivide domain Ω into non-overlapping regions Ω_i
 - Each region is called a stratum
- Take one random sample per Ω_i



Stratified Sampling Example

- When supersampling, instead of taking $K \times K$ regular sub-pixel samples, do random jittering within each $K \times K$ sub-pixel

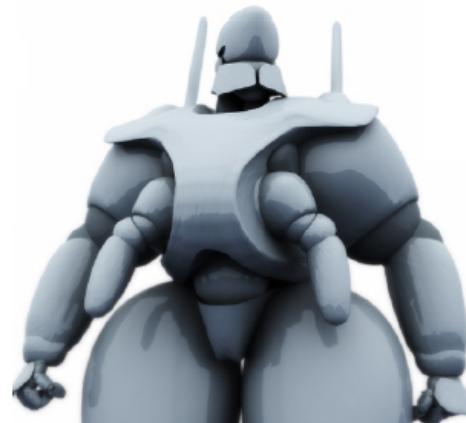
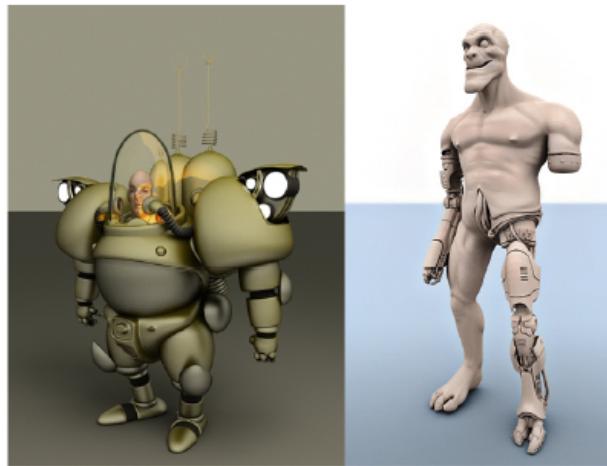
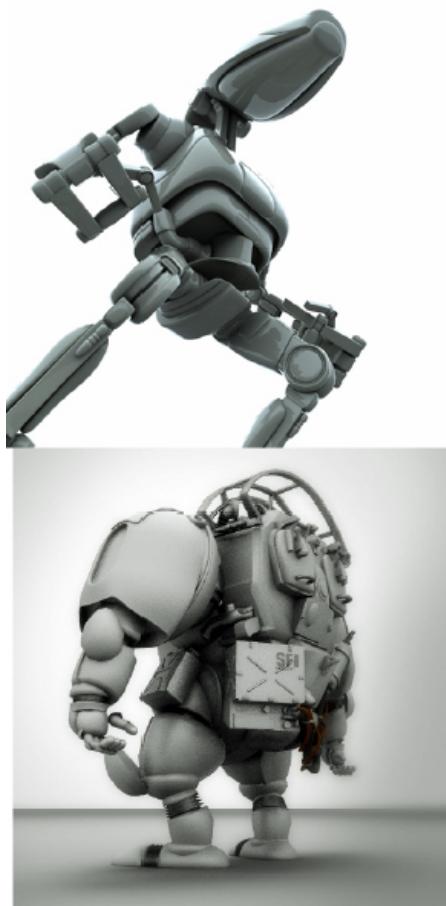


Stratified Sampling Analysis

- Cheap and effective
- But mostly for low-dimensional domains
 - Again, subdivision of N-D needs N^d domains like trapezoid, Simpson's, etc.!
- With very high dimensions, Monte Carlo is pretty much the only choice

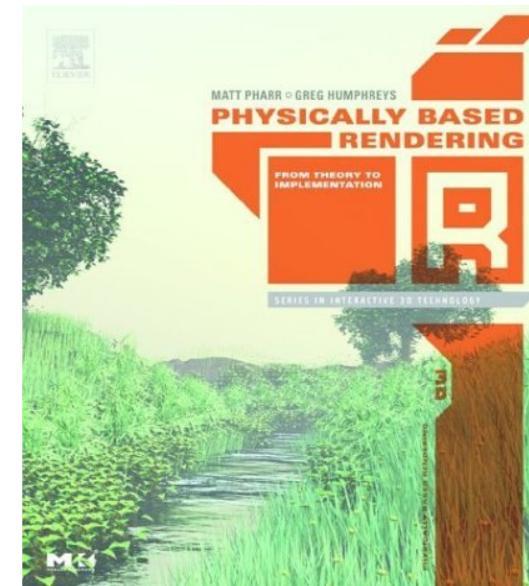
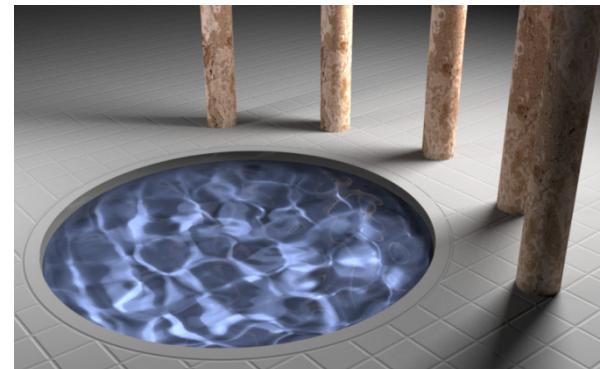
Questions?

- Image from the ARNOLD Renderer by Marcos Fajardo



For Further Information...

- Eric Veach's PhD dissertation
http://graphics.stanford.edu/papers/veach_thesis/
- Physically Based Rendering
by Matt Pharr, Greg Humphreys



References

Global Illumination in Unity 5

<http://blogs.unity3d.com/2014/09/18/global-illumination-in-unity-5/>

Advanced Global
Illumination



Philip Dutré
Philippe Bekaert
Kavita Bala

REALISTIC
RAY TRACING

PETER SHIRLEY



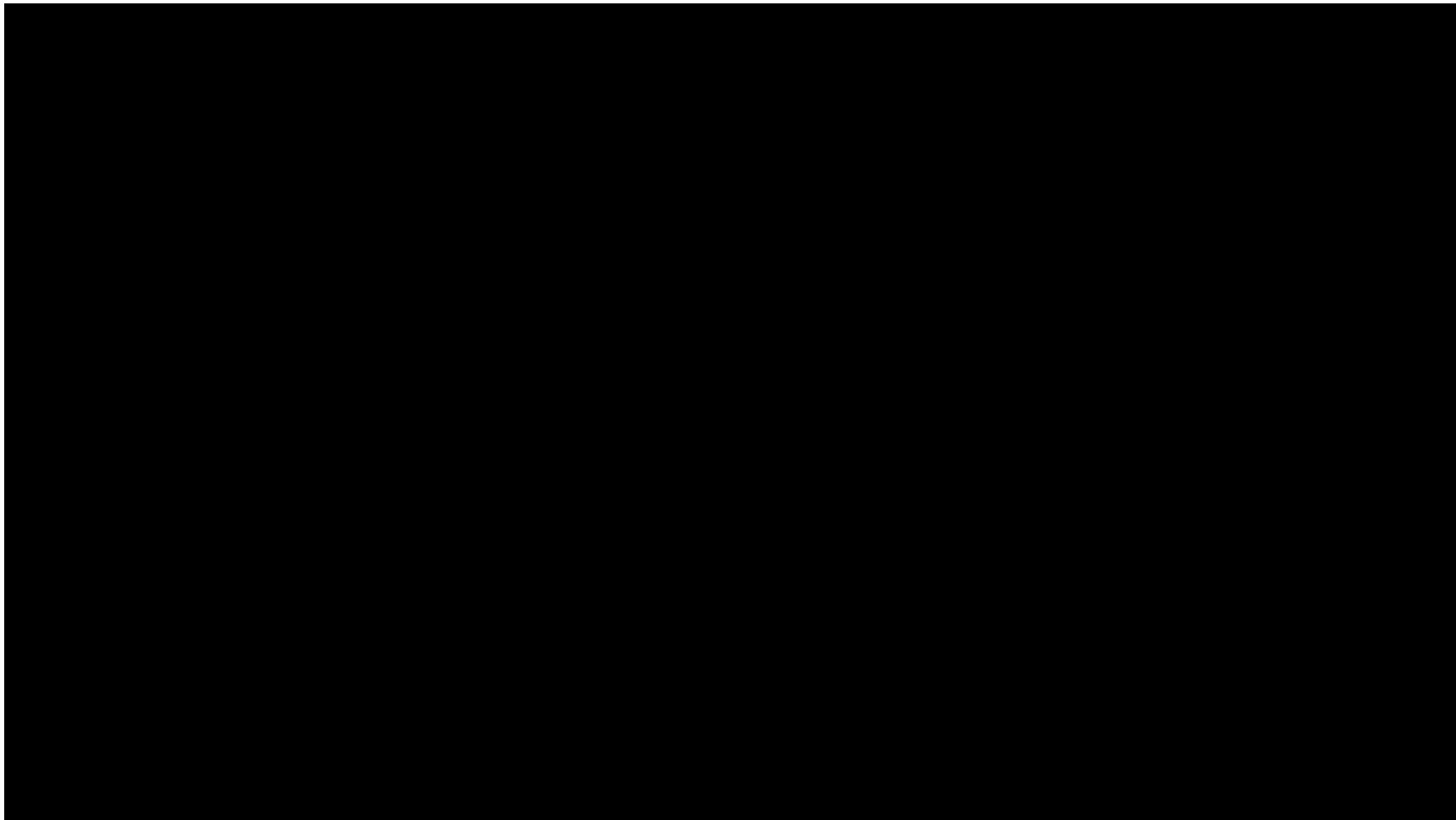
Henrik Wann Jensen

Realistic Image
Synthesis
Using Photon
Mapping



Foreword by Pat Hanrahan

Unity 5



That's All for today



Image: Fournier and
Reeves, SIGGRAPH 86