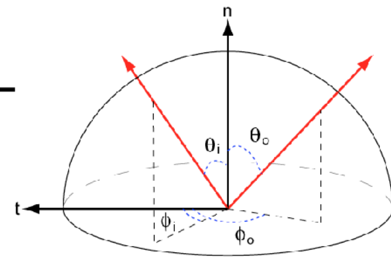
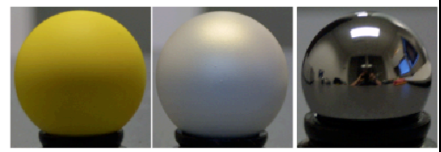
The Rendering Equation
& Monte Carlo Ray Tracing

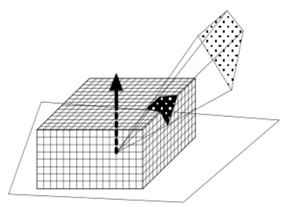
Last Time?

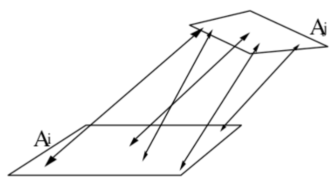


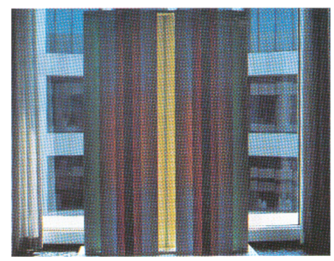
- Local Illumination
  - BRDF
  - Ideal Diffuse Reflectance
  - Ideal Specular Reflectance
  - The Phong Model



- Radiosity Equation/Matrix
- Calculating the Form Factors





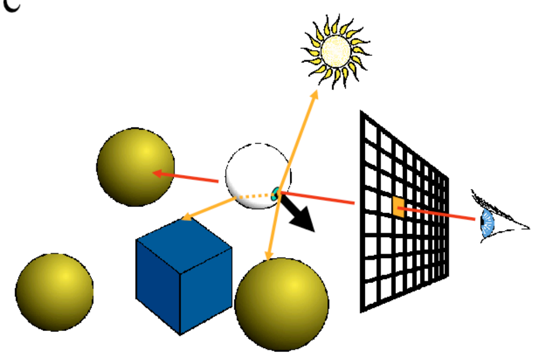


Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

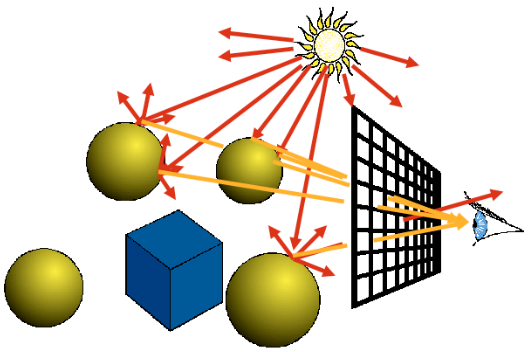
Does Ray Tracing Simulate Physics?

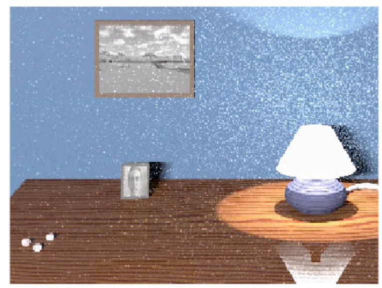
- No…. traditional ray tracing is also called *"backward" ray tracing*
- In reality, photons actually travel from the light to the eye



Forward Ray Tracing

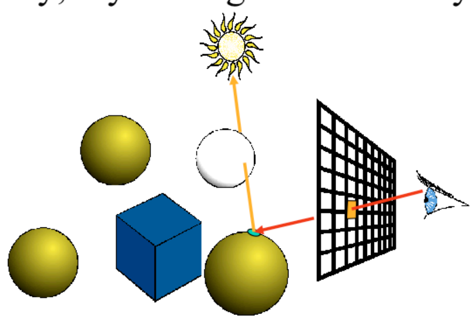
- Start from the light source
  - But very, very low probability to reach the eye
- What can we do about it?
  - Always send a ray to the eye…. still not efficient



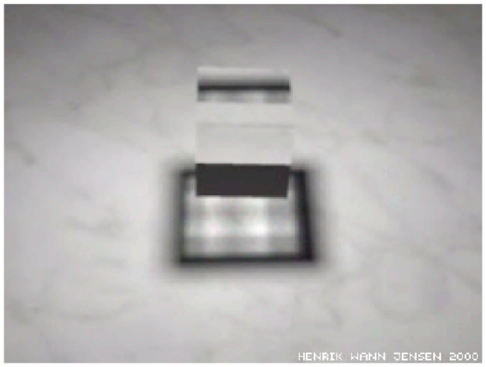


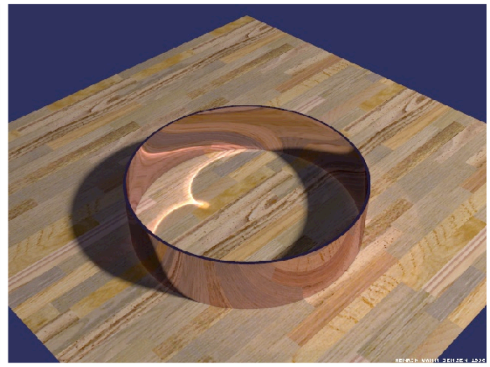
Transparent Shadows?

- What to do if the shadow ray sent to the light source intersects a transparent object?
  - Pretend it's opaque?
  - Multiply by transparency color?
    (ignores refraction & does not produce caustics)
- Unfortunately, ray tracing is full of dirty tricks



Is this Traditional Ray Tracing?



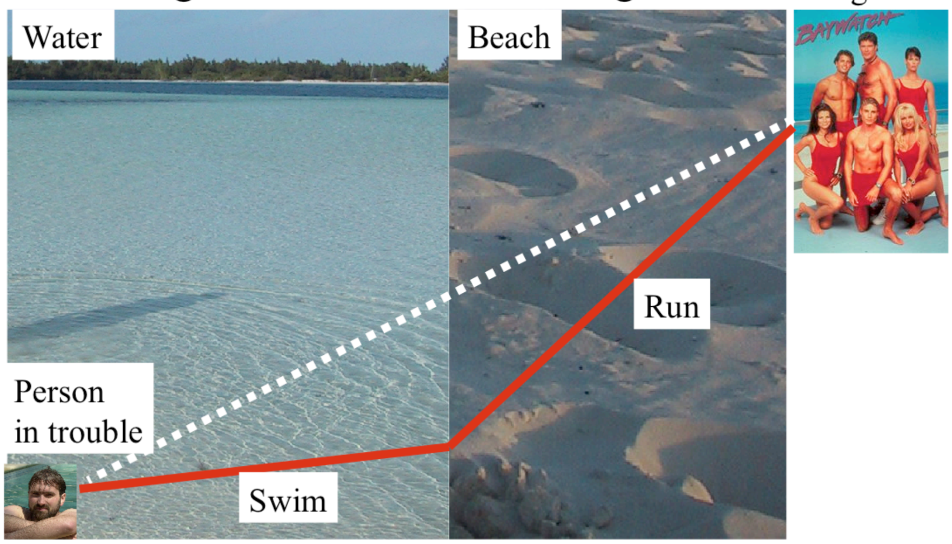


Images by Henrik Wann Jensen

- No, Refraction and complex reflection for illumination are not handled properly in traditional (backward) ray tracing

Refraction and the Lifeguard Problem

- Running is faster than swimming

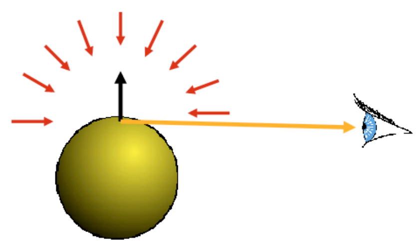


Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

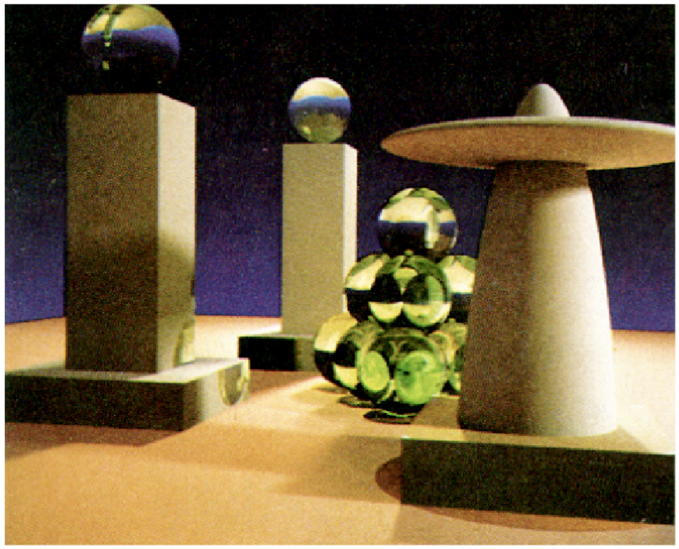
The Rendering Equation

- Clean mathematical framework for light-transport simulation
- At each point, outgoing light in one direction is the integral of incoming light in all directions multiplied by reflectance property

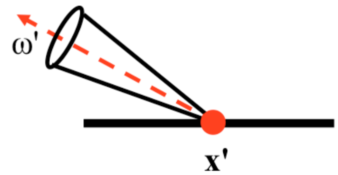


Reading for Today:

- "The Rendering Equation", Kajiya, SIGGRAPH 1986



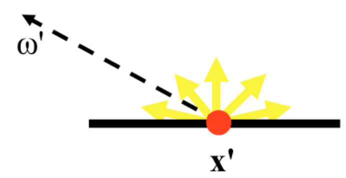
The Rendering Equation



$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\, dA$$

L (x',ω') is the radiance from a point on a surface in a given direction ω'

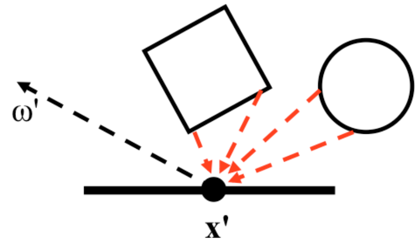
The Rendering Equation



$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\,dA$$

E(x',ω') is the emitted radiance from a point: E is non-zero only if x' is emissive (a light *source*)

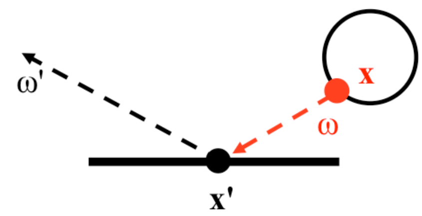
The Rendering Equation



$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\, dA$$

Sum the contribution from all of the other surfaces in the scene

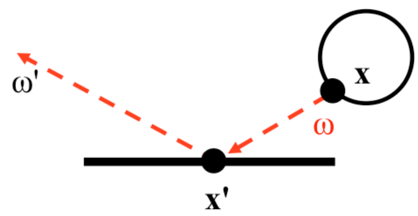
The Rendering Equation



$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\,dA$$

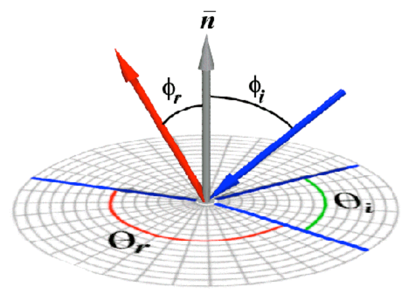
For each x, compute L(x, ω), the radiance at point x in the direction ω (from x to x')

The Rendering Equation

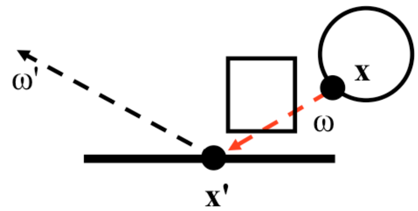


$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega') L(x,\omega) G(x,x') V(x,x')\, dA$$

scale the contribution by $\rho_{x'}(\omega,\omega')$, the reflectivity (BRDF) of the surface at x'



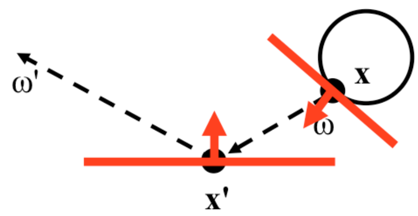
The Rendering Equation



$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\,dA$$

For each x, compute V(x,x'),
the visibility between x and x':
1 when the surfaces are unobstructed
along the direction ω, 0 otherwise

The Rendering Equation

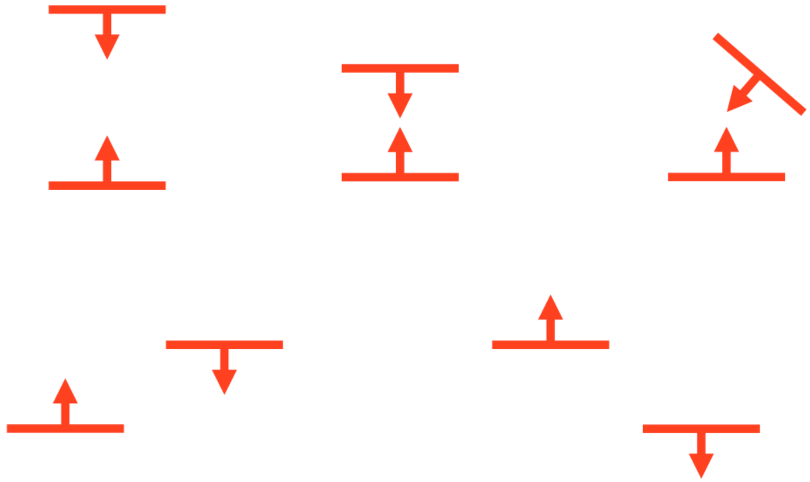


$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\, dA$$

For each x, compute G(x, x'), which describes the on the geometric relationship between the two surfaces at x and x'

Intuition about G(x,x')?

- Which arrangement of two surfaces will yield the greatest transfer of light energy? Why?



Rendering Equation ➔ Radiosity

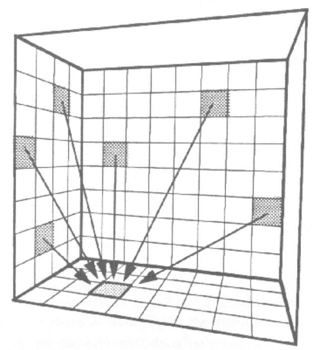
$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega') L(x,\omega) G(x,x') V(x,x')\, dA$$

Radiosity assumption:
perfectly diffuse surfaces (not directional)

$$B_{x'} = E_{x'} + \rho_{x'} \int B_x \; G(x,x') V(x,x')$$

discretize

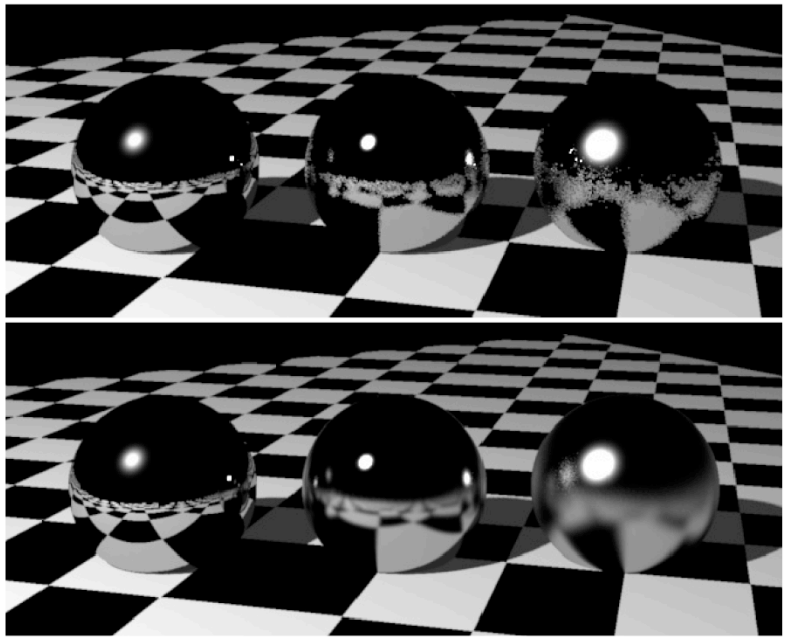
$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$



Questions?

1 glossy sample per pixel

256 glossy samples per pixel

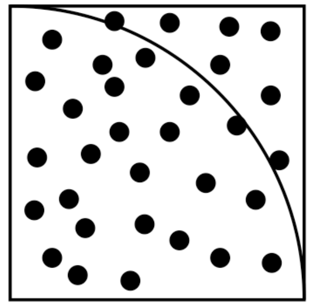


Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
  - Probabilities and Variance
  - Analysis of Monte-Carlo Integration
- Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

Monte-Carlo Computation of π

- Take a random point (x,y) in unit square
- Test if it is inside the ¼ disc
  - Is $x^2 + y^2 < 1$?
- Probability of being inside disc?
  - area of ¼ unit circle / area of unit square = π /4



- $\pi \approx 4$ * number inside disc / total number
- The error depends on the number or trials

Convergence & Error

- Let's compute 0.5 by flipping a coin:
  - 1 flip: 0 or 1
    → average error = 0.5
  - 2 flips: 0, 0.5, 0.5 or 1
    → average error = 0. 25
  - 4 flips: 0 (*1),0.25 (*4), 0.5 (*6), 0.75(*4), 1(*1)
    → average error = 0.1875
- Unfortunately, doubling the number of samples does not double accuracy

Review of (Discrete) Probability

- Random variable can take discrete values x_i
- Probability p_i for each x_i

  $0 < p_i < 1, \quad \Sigma\, p_i = 1$
- Expected value $$E(x) = \sum_{i=1}^{n} p_i x_i$$
- Expected value of function of random variable
  - $f(x_i)$ is also a random variable

$$E[f(x)] = \sum_{i=1}^{n} p_i f(x_i)$$

Variance & Standard Deviation

- Variance σ^2: deviation from expected value
- Expected value of square difference

$$\sigma^2 = E[(x - E[x])^2] = \sum_i (x_i - E[x])^2 p_i$$

- Also

$$\sigma^2 = E[x^2] - (E[x])^2$$

- Standard deviation σ:
  square root of variance (notion of error, RMS)

Monte Carlo Integration

- Turn integral into finite sum
- Use *n* random samples
- As *n* increases…
  - Expected value remains the same
  - Variance decreases by *n*
  - Standard deviation (error) decreases by $\frac{1}{\sqrt{n}}$
- Thus, converges with $\frac{1}{\sqrt{n}}$

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 Integration

- Noisy
- Slow convergence
- Good implementation is hard
  - Debugging code
  - Debugging math
  - Choosing appropriate techniques
- Punctual technique, no notion of smoothness of function (e.g., between neighboring pixels)

Questions?

- "A Theoretical Framework for Physically Based Rendering", Lafortune and Willems, Computer Graphics Forum, 1994.

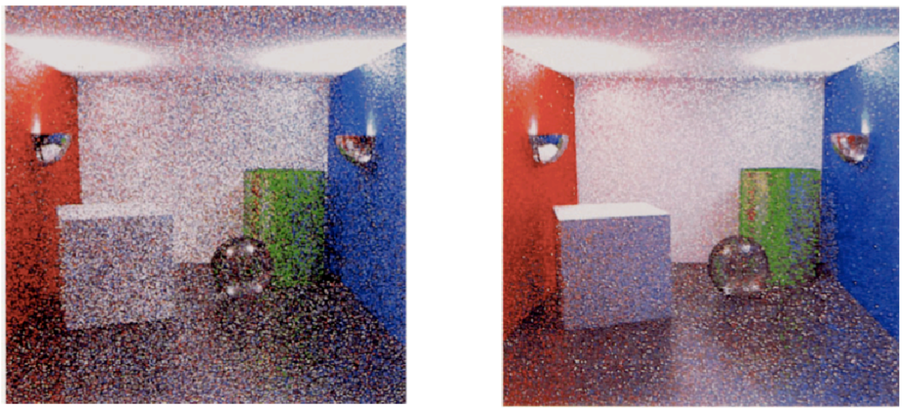


Figure B: *An indirectly illuminated scene rendered using path tracing and bidirectional path tracing respectively. The latter method results in visibly less noisefor the same amount of work.*

Today

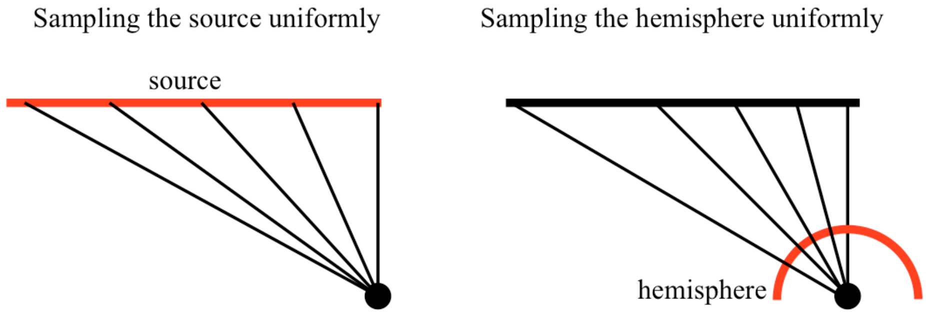
- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
  - Stratified Sampling
  - Importance Sampling
- Monte-Carlo Ray Tracing vs. Path Tracing

Domains of Integration

- Pixel, lens (Euclidean 2D domain)
- Time (1D)
- Hemisphere
  - – Work needed to ensure *uniform* probability

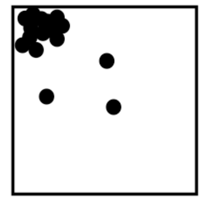
Example: Light Source

- We can integrate over surface *or* over angle
- But we must be careful to get probabilities and integration measure right!

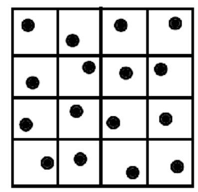


Stratified Sampling

- With uniform sampling, we can get unlucky
  - E.g. all samples in a corner



- To prevent it, subdivide domain Ω into non-overlapping regions Ω_i
  - Each region is called a stratum



- Take one random samples per Ω_i

Stratified Sampling Example

$f(x) = e^{\sin(3x^2)}$

N	I
1	2.75039
10	1.9893
100	1.79139
1000	1.75146
10000	1.77313
100000	1.77862

Unstratified

$O(1/\sqrt{N})$

$f(x) = e^{\sin(3x^2)}$

N	I
1	2.70457
10	1.72858
100	1.77925
1000	1.77606
10000	1.77610
100000	1.77610

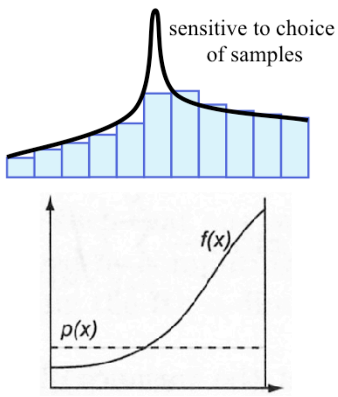
Stratified

$O(1/N)$

Slide from Henrik Wann Jensen

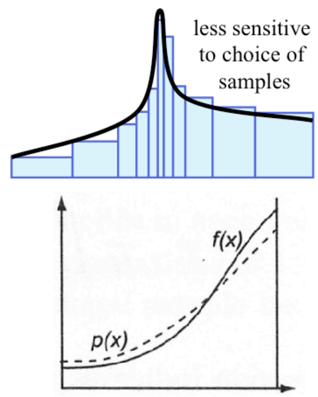
Sampling

uniform sampling
(or uniform random)



all samples
weighted equally

dense sampling where
function has greater magnitude

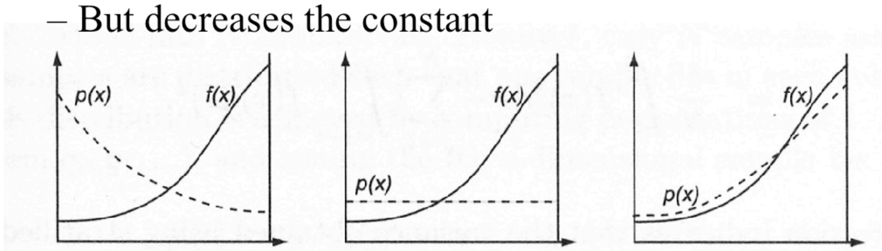


weights (width) for dense
samples are reduced

Importance Sampling

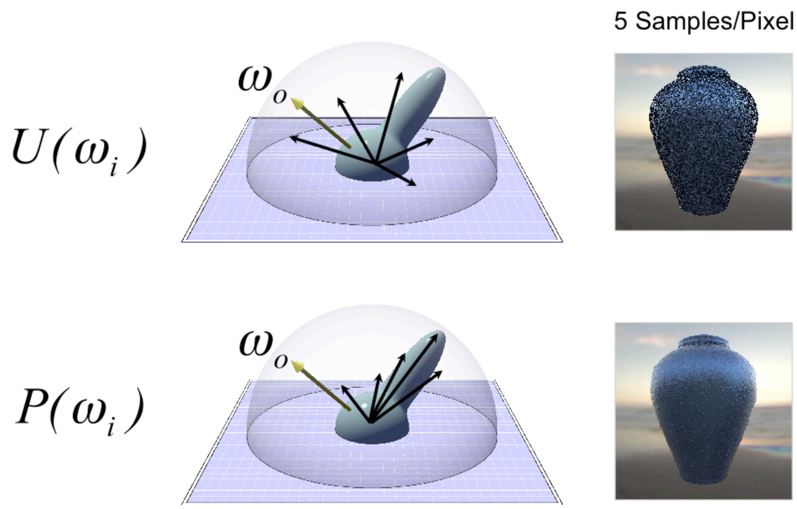
$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

- Choose p wisely to reduce variance
  - Want to use a p that resembles f
  - Does not change convergence rate (still sqrt)
  - But decreases the constant



bad uniform good

Uniform vs. Importance Sampling



Slide from Jason Lawrence

(1) No closed form solution for integral so must sample hemisphere and form estimate of integral.

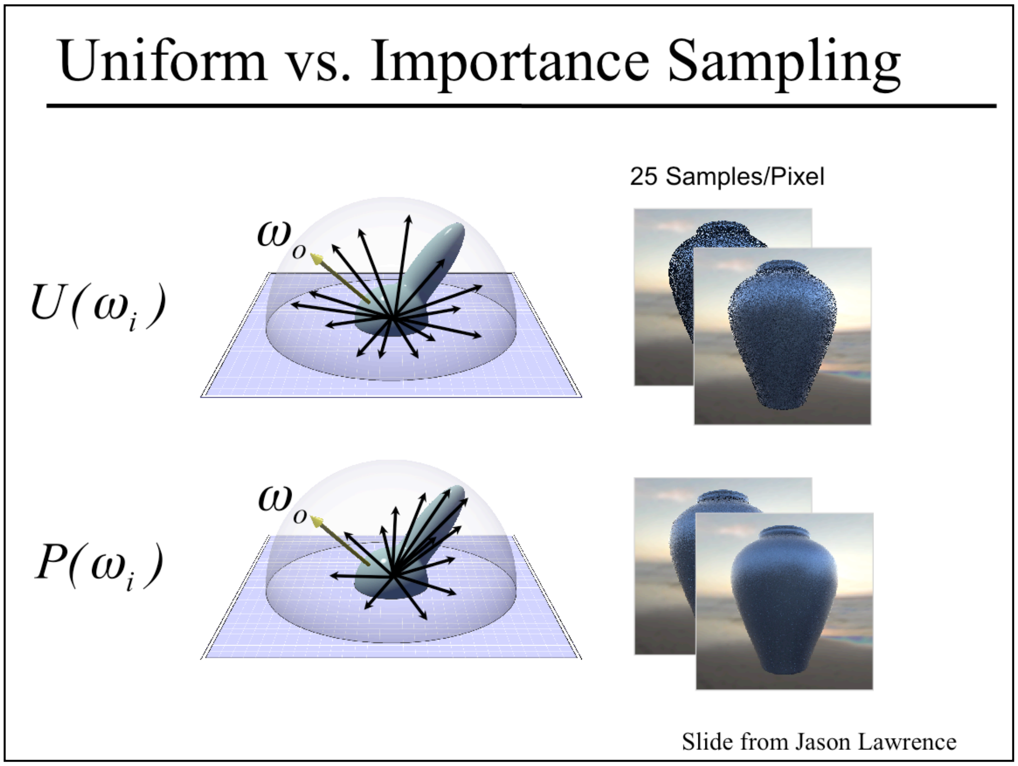
(2) Uniform distribution.

(3) Select directions uniformly at each pixel and estimate integral. Notion of variance in this estimator since it approximates integral.

(4) Perform sampling at each pixel: variance in estimator appears as noise in rendered image.

(5) Importance sampling: select samples from distribution that match shape of BRDF.

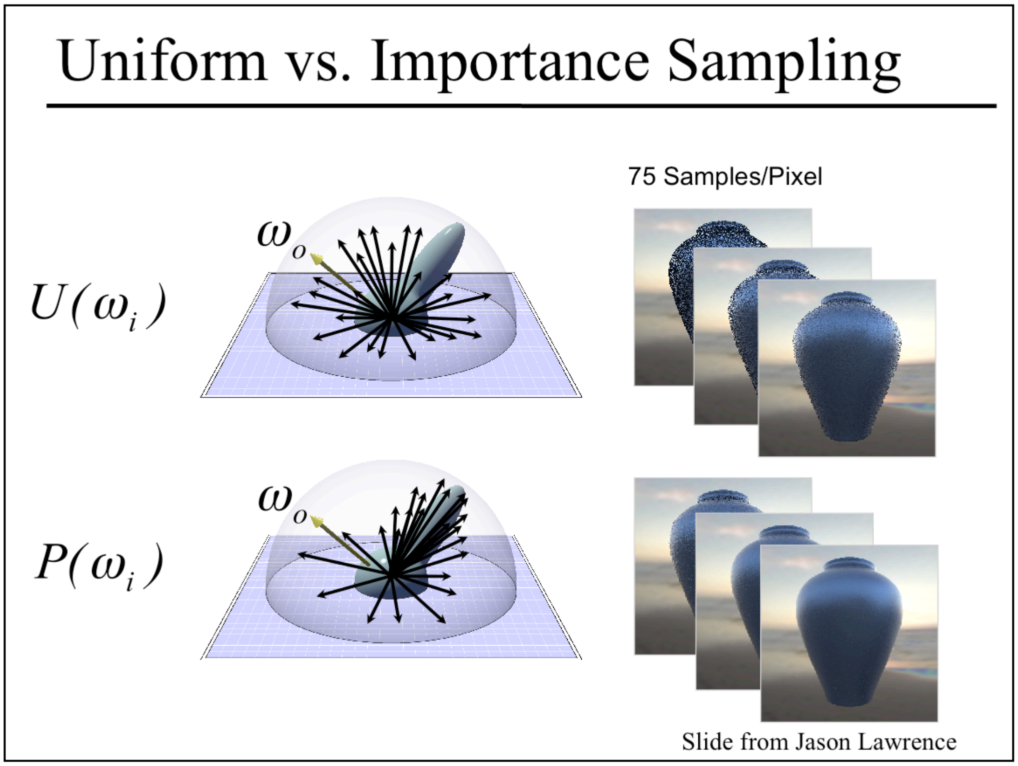
(6) This reduces variance in estimate of integral and noise in the image for fixed sample count.



(1) As we increase the sample count both distributions converge.

(2) Bottom distribution will converge to actual solution with fewer samples – less time.

(3) Idea of our work: find P for any BRDF.



(1) As we increase the sample count both distributions converge.

(2) Bottom distribution will converge to actual solution with fewer samples – less time.

(3) Idea of our work: find P for any BRDF.

Questions?

Naïve sampling strategy

Optimal sampling strategy

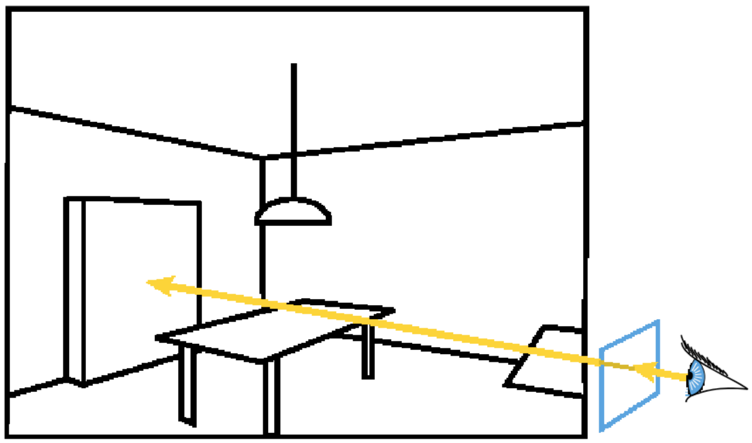
Veach & Guibas "Optimally Combining Sampling
Techniques for Monte Carlo Rendering" SIGGRAPH 95

Today

- Does Ray Tracing Simulate Physics?
- The Rendering Equation
- Monte-Carlo Integration
- Sampling
- Monte-Carlo Ray Tracing & Path Tracing

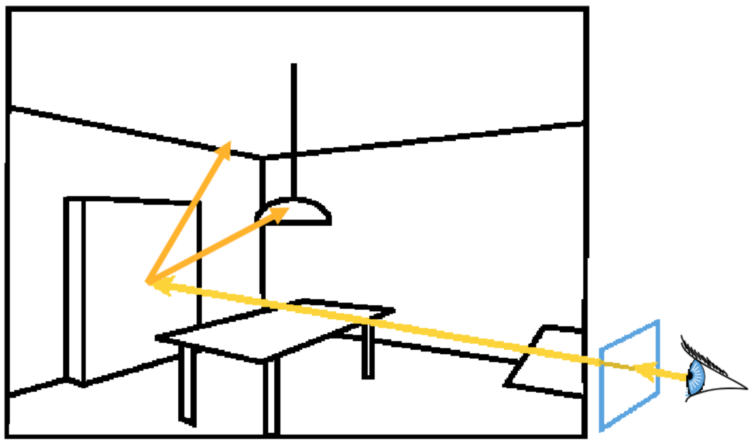
Ray Casting

- Cast a ray from the eye through each pixel



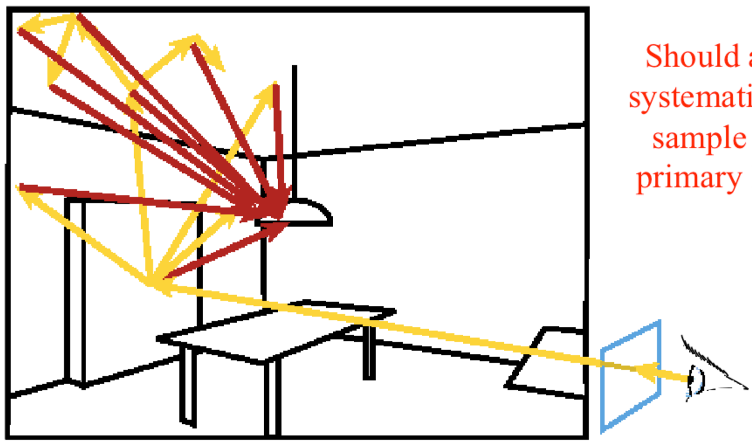
Ray Tracing

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



Monte-Carlo Ray Tracing

- Cast a ray from the eye through each pixel
- Cast random rays to accumulate radiance contribution
  - Recurse to solve the Rendering Equation



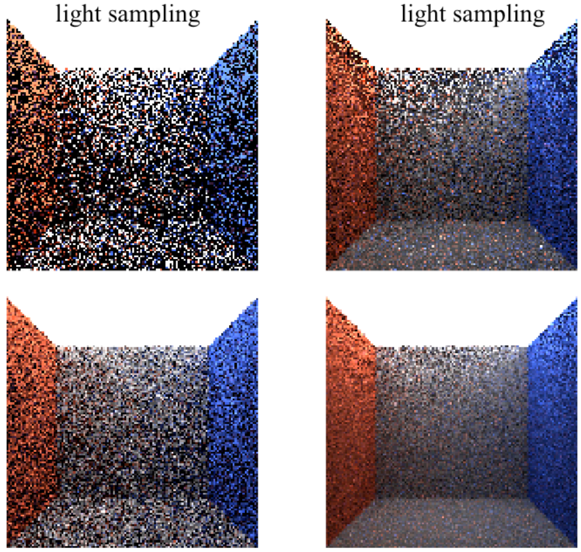
Should also systematically sample the primary light

Importance of Sampling the Light

Without explicit light sampling | With explicit light sampling

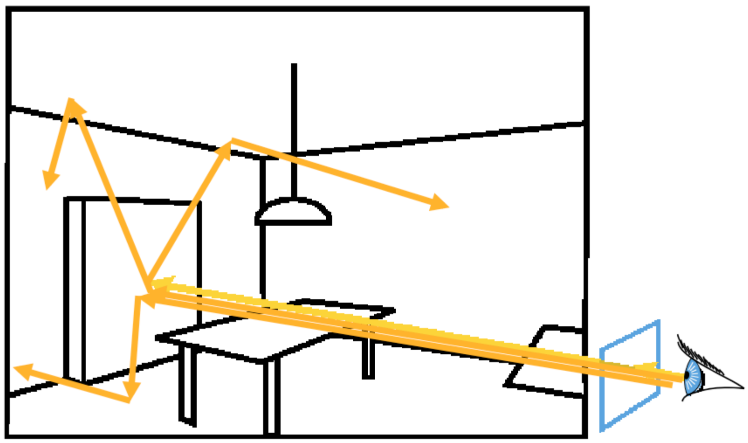
1 path per pixel

4 path per pixel

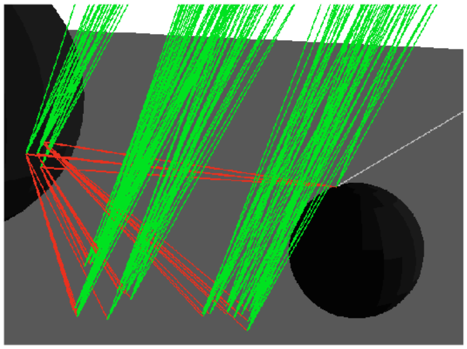


Monte Carlo Path Tracing

- Trace only one secondary ray per recursion
- But send many primary rays per pixel
  (performs antialiasing as well)



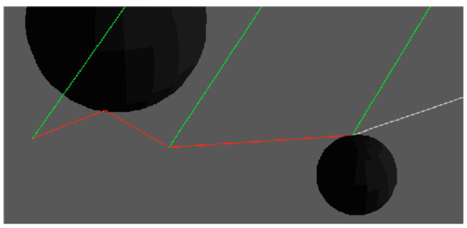
Ray Tracing vs Path Tracing



2 bounces
5 glossy samples
5 shadow samples

How many rays cast per pixel?

1main ray + 5 shadow rays +
5 glossy rays + 5x5 shadow rays +
5*5 glossy rays + 5x5x5 shadow rays
= 186 rays



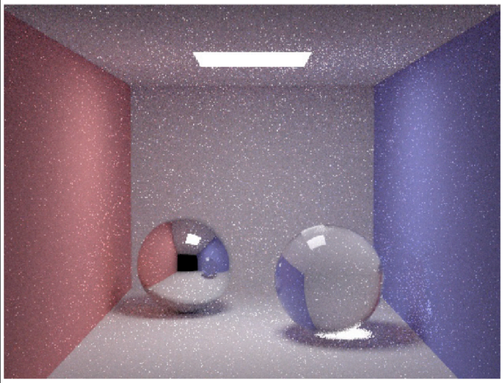
How many 3 bounce paths can we trace per pixel for the same cost?

186 rays / 8 ray casts per path
= ~23 paths

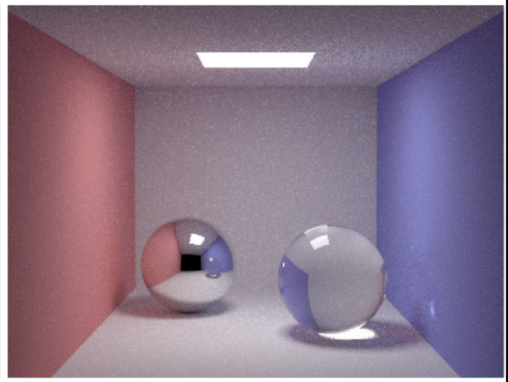
Which will probably have less error?

Questions?

10 paths/pixel



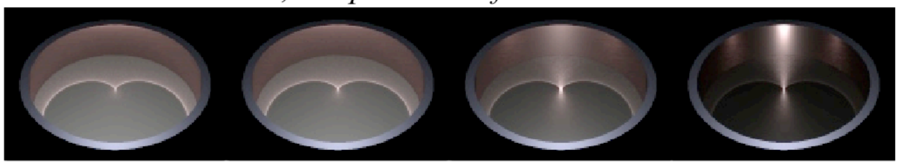
100 paths/pixel



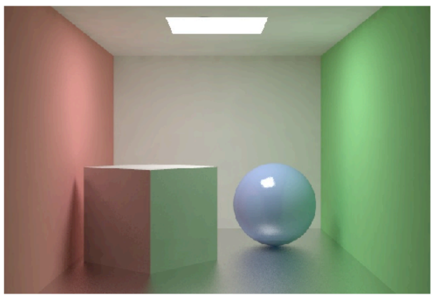
Images from Henrik Wann Jensen

Readings for Friday (3/22) *pick one*:

- "Rendering Caustics on Non-Lambertian Surfaces",
  Henrik Wann Jensen, *Graphics Interface* 1996.



- "Global Illumination using Photon Maps",
  Henrik Wann Jensen, *Rendering Techniques* 1996.



Raytracing & Epsilon

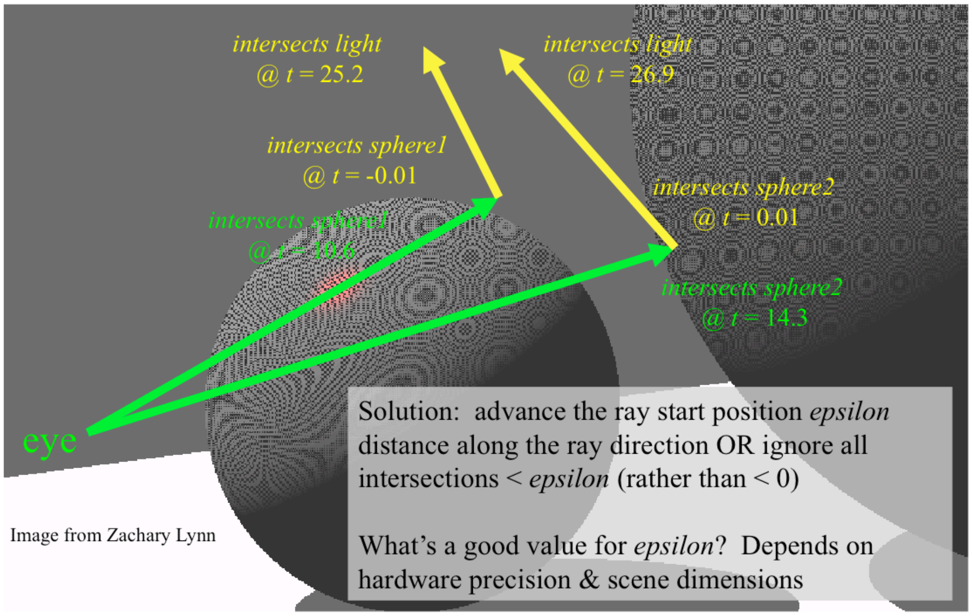


Image from Zachary Lynn

Solution: advance the ray start position *epsilon* distance along the ray direction OR ignore all intersections < *epsilon* (rather than < 0)

What's a good value for *epsilon*? Depends on hardware precision & scene dimensions