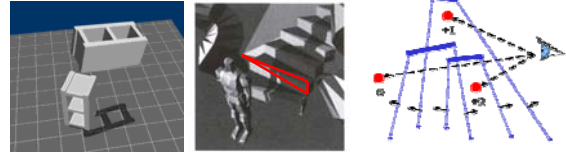
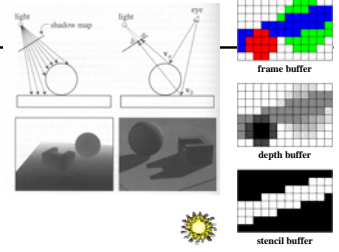


# Monte Carlo Rendering

## Last Time?

- Planar Shadows
- Projective Texture Shadows
- Shadow Maps
- Shadow Volumes
  - Stencil Buffer

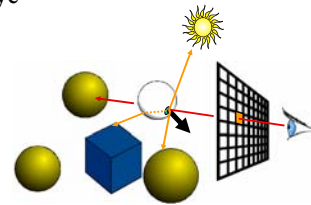


## Today

- **Does Ray Tracing Simulate Physics?**
- Monte-Carlo Integration
- Sampling
- Advanced Monte-Carlo Rendering

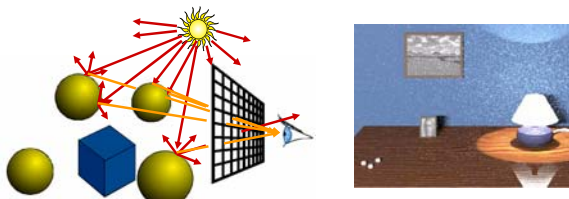
## Does Ray Tracing Simulate Physics?

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



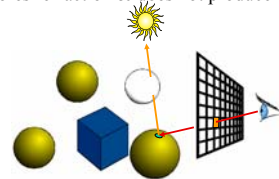
## Forward Ray Tracing

- 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



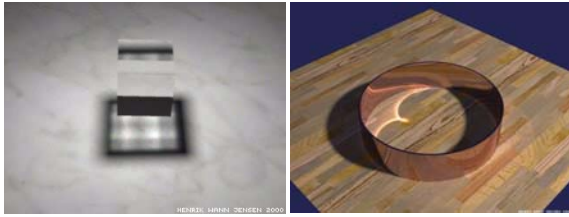
## Transparent Shadows?

- 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?

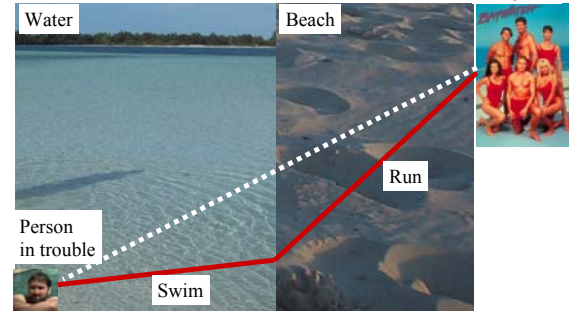


Images by Henrik Wann Jensen

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

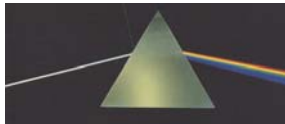
## Refraction and the Lifeguard Problem

- Running is faster than swimming



## What makes a Rainbow?

- Refraction is wavelength-dependent
  - Refraction increases as the wavelength of light decreases
  - violet and blue experience more bending than orange and red
- Usually ignored in graphics
- Rainbow is caused by refraction + internal reflection + refraction



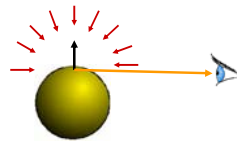
Pink Floyd, *The Dark Side of the Moon*

From "Color and Light in Nature" by Lynch and Livingstone



## 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

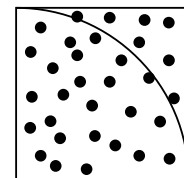


## Today

- Does Ray Tracing Simulate Physics?
- Monte-Carlo Integration
  - Probabilities and Variance
  - Analysis of Monte-Carlo Integration
- Sampling
- Advanced Monte-Carlo Rendering

## Monte-Carlo Computation of $\pi$

- Take a random point  $(x,y)$  in unit square
- Test if it is inside the  $\frac{1}{4}$  disc
  - Is  $x^2 + y^2 < 1$ ?
- Probability of being inside disc?
  - area of  $\frac{1}{4}$  unit circle / area of unit square
  - =  $\pi / 4$
- $\pi \approx 4 * \text{number inside disc} / \text{total number}$
- The error depends on the number or trials



## Convergence & Error

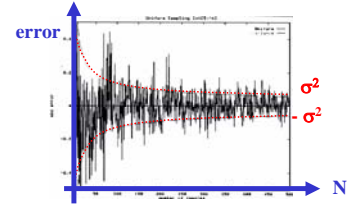
- Let's compute 0.5 by flipping a coin:
  - 1 flip: 0 or 1
    - average error = 0.5
  - 2 flips: 0, 0.5, 0.5 or 1
    - average error = 0.25
  - 4 flips: 0 (\*1), 0.25 (\*4), 0.5 (\*6), 0.75(\*4), 1(\*1)
    - average error = 0.1875
- Unfortunately, doubling the number of samples does not double accuracy

## Another Example:

$$I = \int_0^1 5x^4 dx$$

- We know it should be 1.0

- In practice with uniform samples:



## Review of (Discrete) Probability

- Random variable can take discrete values  $x_i$
- Probability  $p_i$  for each  $x_i$ 
  - $0 < p_i < 1, \sum p_i = 1$
- 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  $\sigma^2$ : 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  $\sigma$ :
  - 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  $\frac{1}{\sqrt{n}}$

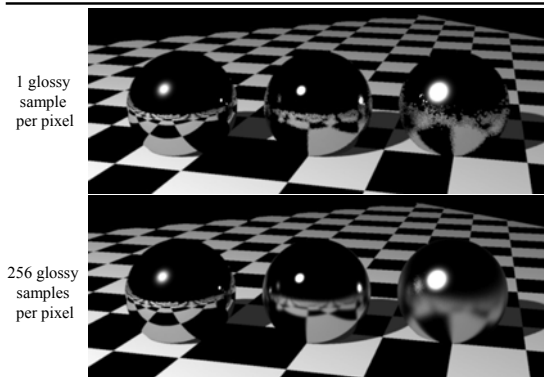
## Advantages of MC Integration

- 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?



## Today

- Does Ray Tracing Simulate Physics?
- Monte-Carlo Integration
- **Sampling**
  - **Stratified Sampling**
  - **Importance Sampling**
- Advanced Monte-Carlo Rendering

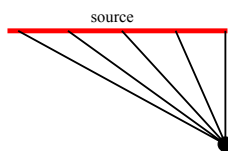
## Domains of Integration

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

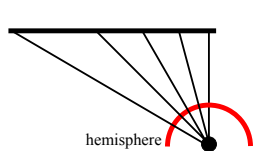
## Example: Light Source

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

Sampling the source uniformly

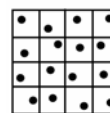
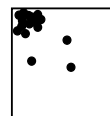


Sampling the hemisphere uniformly



## Stratified Sampling

- With uniform sampling, we can get unlucky
  - E.g. all samples in a corner
- To prevent it, subdivide domain  $\Omega$  into non-overlapping regions  $\Omega_i$ 
  - Each region is called a stratum
- Take one random samples per  $\Omega_i$



## Example

- Borrowed from Henrik Wann Jensen

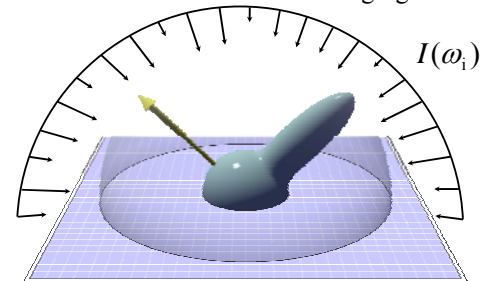
$f(x) = e^{\sin(3x^2)}$		$f(x) = e^{\sin(3x^2)}$	
N	I	N	I
1	2.75039	1	2.70457
10	1.9893	10	1.72858
100	1.79139	100	1.77925
1000	1.75146	1000	1.77606
10000	1.77313	10000	1.77610
100000	1.77862	100000	1.77610

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

**Stratified**  
 $O(1/N)$

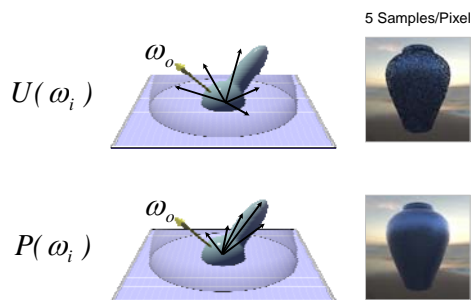
## Glossy Rendering

- Integrate over hemisphere
- BRDF times cosine times incoming light



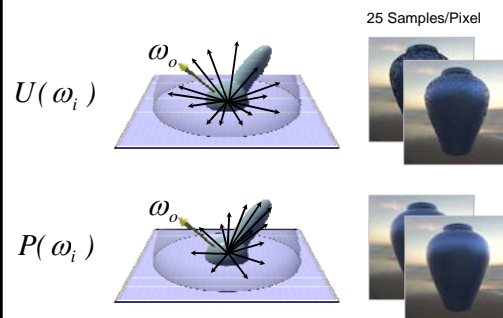
Slide from Jason Lawrence

## Sampling a BRDF



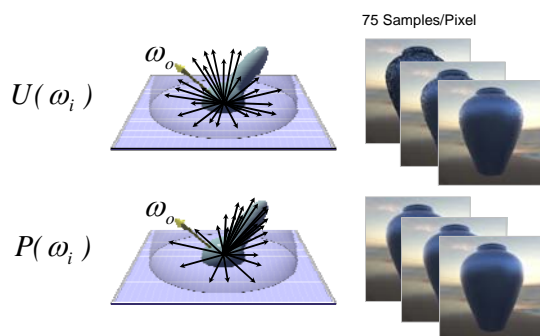
Slide from Jason Lawrence

## Sampling a BRDF



Slide from Jason Lawrence

## Sampling a BRDF

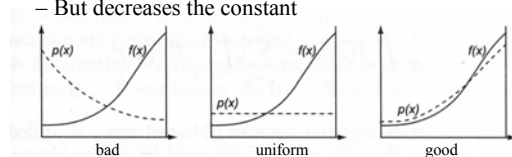


Slide from Jason Lawrence

## Importance Sampling

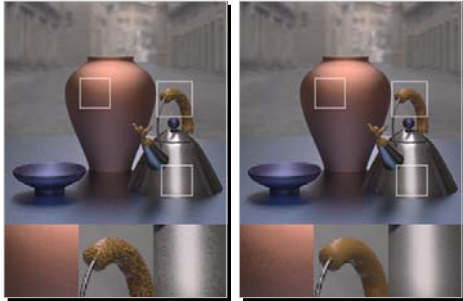
$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}$$

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



## Results

1200 Samples/Pixel



Traditional importance function

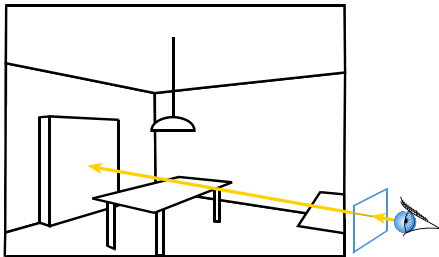
Better importance by Lawrence et al.

## Today

- Does Ray Tracing Simulate Physics?
- Monte-Carlo Integration
- Sampling
- **Advanced Monte-Carlo Rendering**

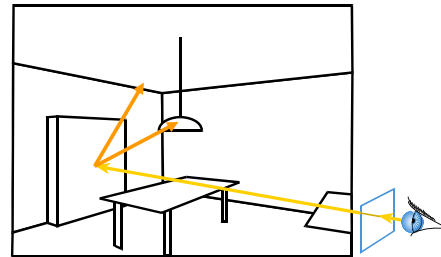
## Ray Casting

- **Cast a ray from the eye through each pixel**



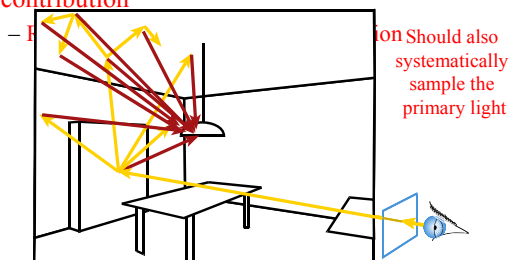
## Ray Tracing

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

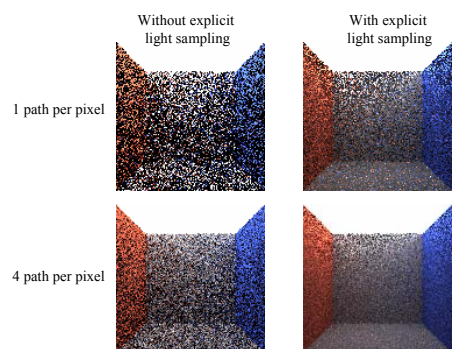


## Monte-Carlo Ray Tracing

- Cast a ray from the eye through each pixel
- **Cast random rays to accumulate radiance contribution**



## Importance of Sampling the Light



### Monte Carlo Path Tracing

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

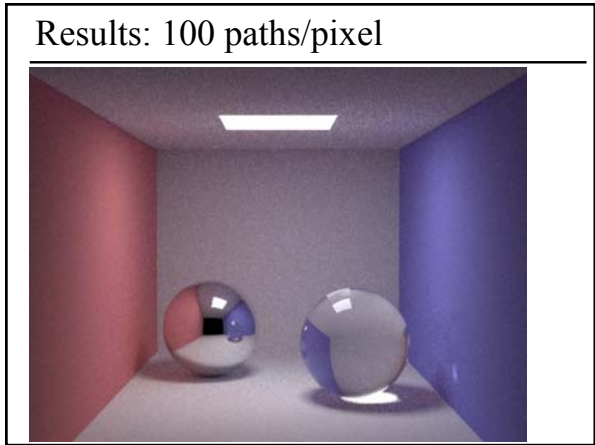
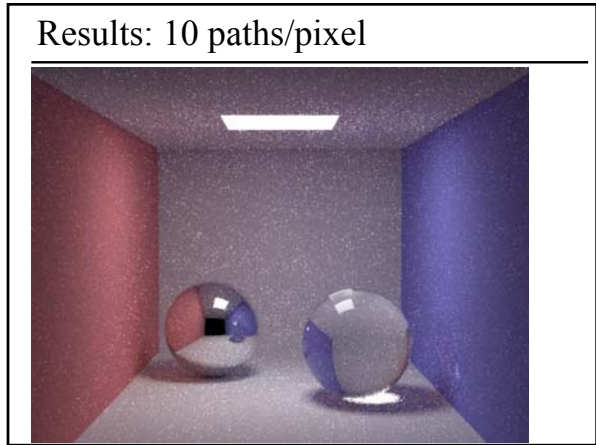
### Ray Tracing vs Path Tracing

2 bounces  
5 glossy samples  
5 shadow samples

How many rays cast per pixel?  
1 main 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?



### Reading for Today

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

Naive sampling strategy      Optimal sampling strategy

- Reading for Tuesday (3/27)

*Statistical Acceleration for Animated Global Illumination*, Meyer & Anderson, SIGGRAPH 2006

- Reading for Friday (3/30)

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