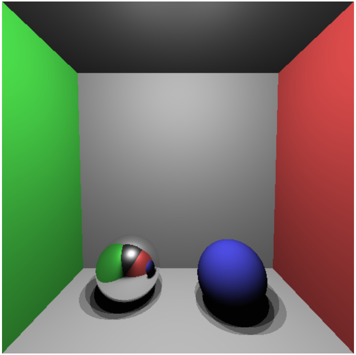
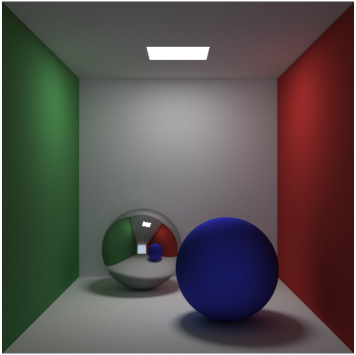
Lecture 17: Monte-Carlo Ray Tracing

October 29, 2019

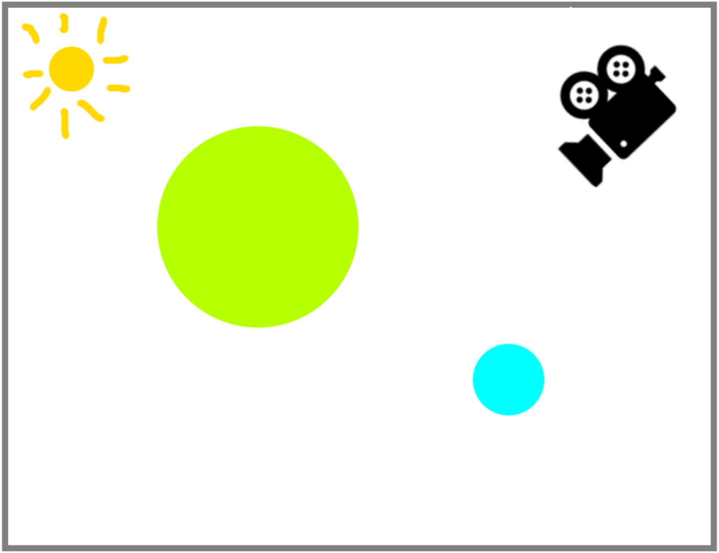
Today



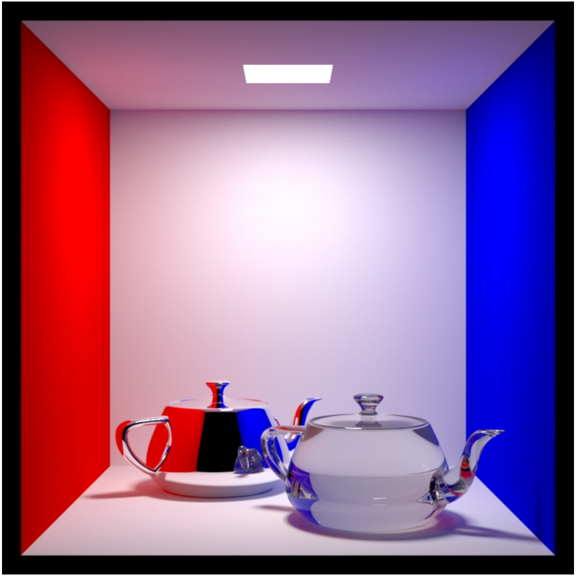
→



What color should the cyan circle appear to be?



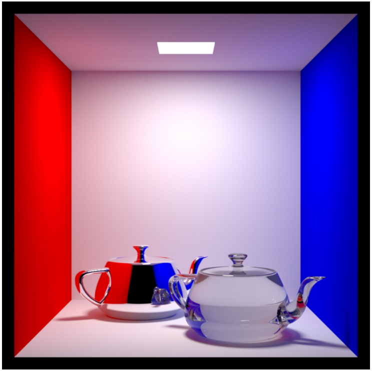
Direct vs. Indirect Lighting

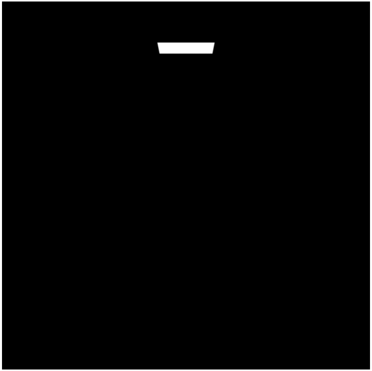


From

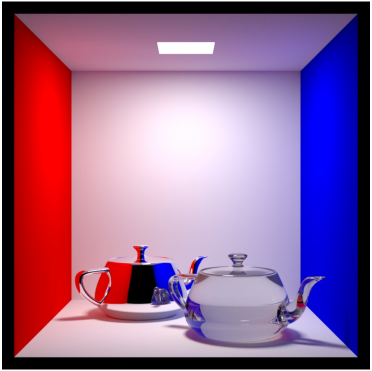
`https://graphics.pixar.com/library/PathTracedMovies/paper.pdf`

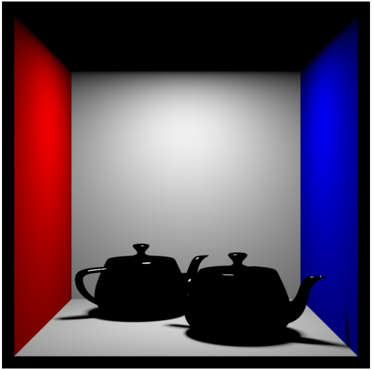
Direct vs. Indirect Lighting



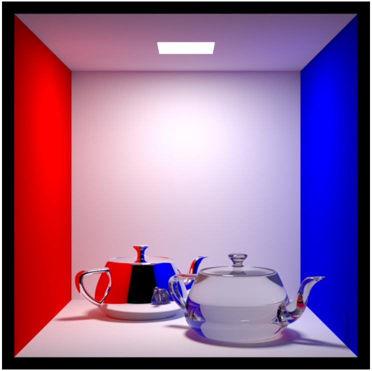


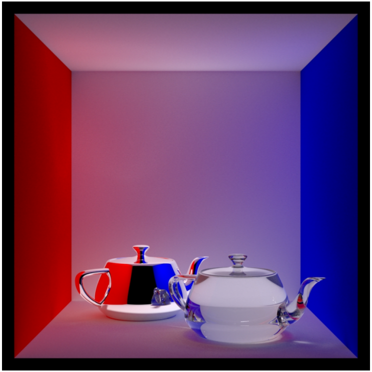
Direct vs. Indirect Lighting



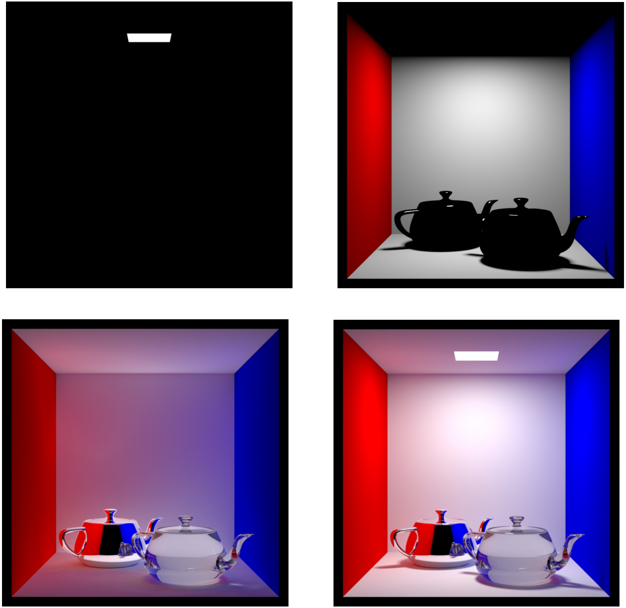


Direct vs. Indirect Lighting





Direct vs. Indirect Lighting



The Rendering Equation

$$
\begin{aligned}
L_o(x, \omega_o, \lambda, t) = & L_e(x, \omega_o, \lambda, t) + \\
& + \int_\Omega f_r(x, w_i, w_o, \lambda, t) L_i(x, w_i, \lambda, t)(\omega_i \cdot n)\mathrm{d}\omega_i
\end{aligned}
$$

The Rendering Equation

$$
\begin{aligned}
L_o(x, \omega_o, \lambda, t) = & L_e(x, \omega_o, \lambda, t) + \\
& + \int_\Omega f_r(x, w_i, w_o, \lambda, t) L_i(x, w_i, \lambda, t)(\omega_i \cdot n) \mathrm{d}\omega_i
\end{aligned}
$$

Light coming out $=$ Light emitted this direction$+$

$+$The amount of incoming light to this point that is reflected this direction

Monte Carlo Methods

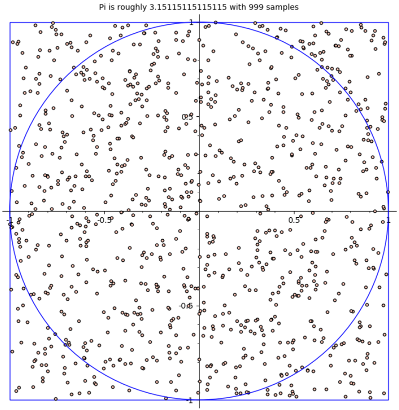
- Exact solutions are difficult (impossible?)
- Compute time is cheap
- Use many different samples to approximate true solution
- Shoot many rays per pixel and let them bounce “randomly”

Approximating Pi with Monte Carlo

Area of circle — πr^2

Area of square — $(2r)^2$

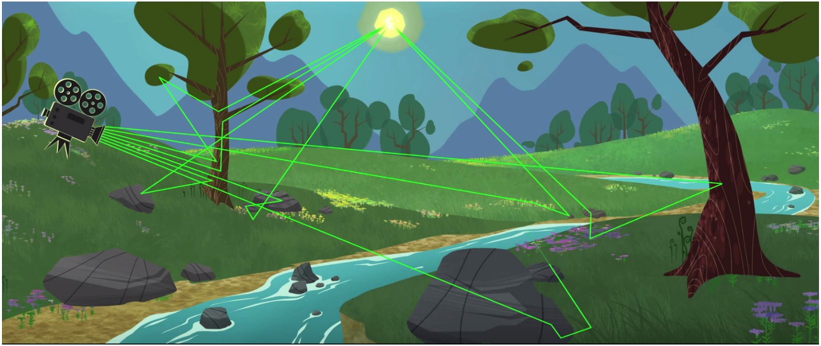
$$\frac{\text{\# samples in}}{\text{\# samples out}} \approx \frac{\pi}{4}$$



Illumination with Monte Carlo

From https://www.youtube.com/watch?v=frLwRLS_ZR0

Illumination with Monte Carlo



From https://www.youtube.com/watch?v=frLwRLS_ZR0

Illumination with Monte Carlo

Don't calculate illumination from each light at each point.
Instead:

1. Shoot many rays per pixel
2. Have each pixel bounce according to material properties
3. Keep track of the running albedo
4. When the ray collides with a light, return

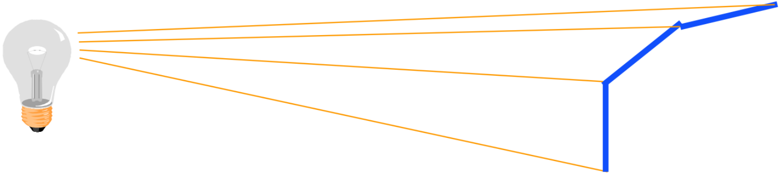
Bouncing Rays

- Specular reflection — keep the same
- Lambertian reflection — what to do?

Bouncing Rays

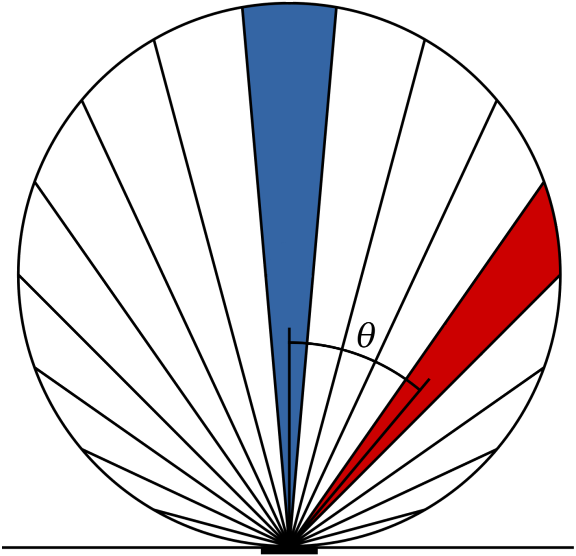
- Specular reflection — keep the same
- Lambertian reflection — what to do?

Recall from lecture eight:



Light per unit area arriving depends upon angle to light source.

Lambert's Cosine Law Revisited



Modified from `https://upload.wikimedia.org/wikipedia/commons/2/25/Lambert_Cosine_Law_1.svg`

Random Point in a Sphere

- Want to sample uniformly from points in the sphere
- Complex to do directly without bias
- Instead, sample from a cube and remove samples outside the sphere

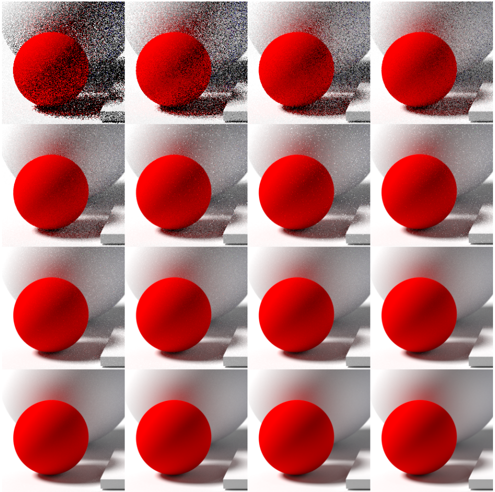
Other Implementation Details

- Ignore *Ka*, *Kd*, *Ks*, and *Kr* terms — only use the albedo and emittance of the object
- Have materials determine the direction of a bounced ray
- Lights must have volume to be collided with
- Lights may need emittance > 1

SageMath Implementation

Warning: slow

Accuracy Over Time



From https://upload.wikimedia.org/wikipedia/commons/e/ea/Path_tracing_sampling_values.png

Tricks for Efficiency

- Ray culling — ignore rays that have low albedo and boost the rest
- Early stopping — skip pixels that have converged
- Bidirectional path tracing — shoot rays from the camera and lights and connect them
- Explicit light sampling — cast rays directly towards the lights some proportion of the time
- AI denoising — render a partial image and feed it to an algorithm that gives a complete image