CSC 433/533
Computer Graphics

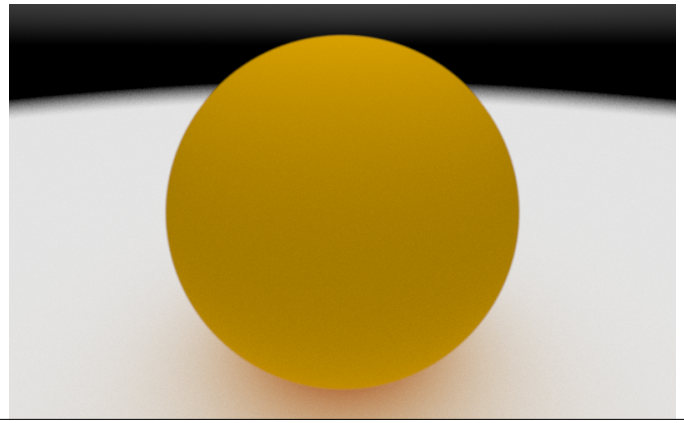
Alon Efrat
Credit: Joshua Levine

Lecture 13
Global Illumination

Oct. 9, 2019

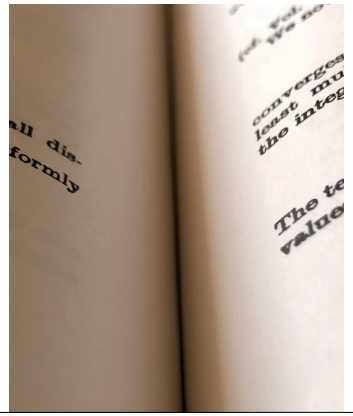
Indirect Lighting

- The world isn't split in "light sources" vs "non-light sources"



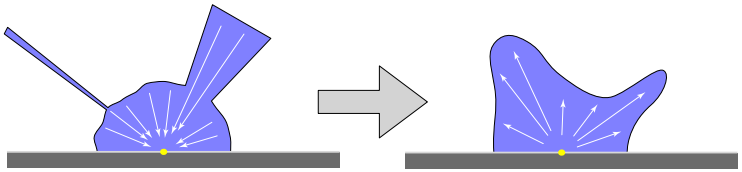
Indirect Lighting

- The world isn't split in "light sources" vs "non-light sources"



Light reflection: full picture

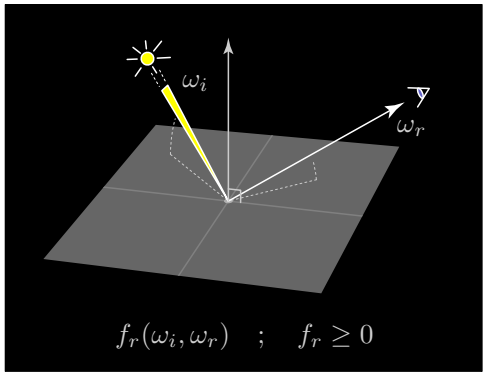
- **when writing a shader, think like a bug standing on the surface**
  - bug sees an incident distribution of light arriving at the surface
  - physics question: what is the outgoing distribution of light?



incident distribution
(function of direction)

reflected distribution
(function of direction)

BRDF



Bidirectional Reflectance Distribution Function

Surface illumination integral (as sum)

- **BRDF tells you how light from a single direction is reflected**
- **Light coming from a small source behaves similarly**
- **What about light coming from everywhere?**
  - approximate incoming light with many small sources on a sphere (the little bug can't tell the difference…)
  - reflected light is sum of reflected light due to each source (each source has its size Ω_k, brightness L_k, and direction ω_k)

$$L_r(\omega_r) = \sum_k \Omega_k L_k f_r(\omega_k, \omega_r) |\omega_k \cdot \mathbf{n}|$$

reflected light in direction ω_r | "intensity" of light source k | BRDF | cosine factor

Surface illumination integral

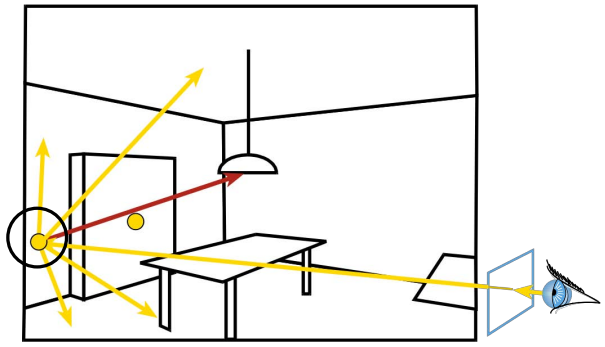
- **Take the limit as the little area sources get smaller**
  - collection of separate brightnesses L_k becomes a function $L_i(\omega_i)$
  - size of sources turns into an integration measure $d\sigma$

$$L_r(\omega_r) = \int_{S^2_+} L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}| d\sigma(\omega_i)$$

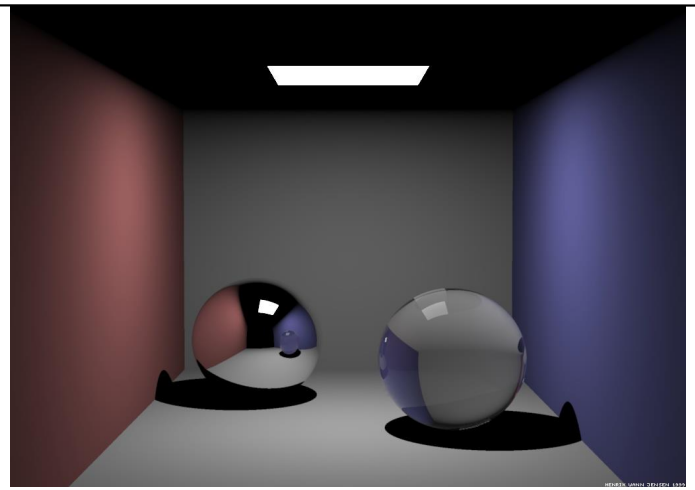
"The light reflected to direction ω_r is the integral, over the positive unit hemisphere, of the incoming light times the BRDF times the incoming cosine factor, with respect to surface area."

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

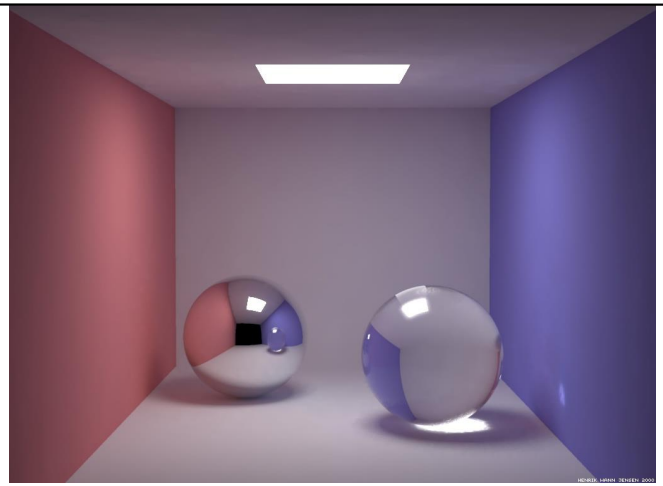


Direct Illumination



Courtesy of Henrik Wann Jensen. Used with permission.

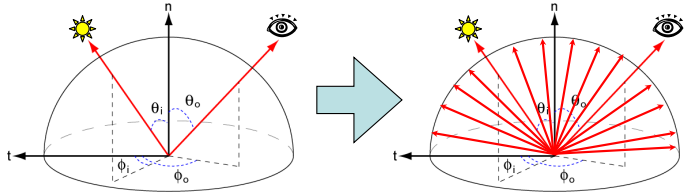
Global Illumination (with Indirect)



Courtesy of Henrik Wann Jensen. Used with permission.

Global Illumination

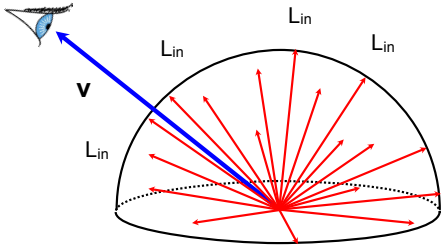
- So far, we only used the BRDF for point lights
  - We just summed over all the point light sources
- BRDF also describes how indirect illumination reflects off surfaces
  - Turns summation into integral over hemisphere
  - As if *every* direction had a light source



Reflectance Equation, Visually

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

outgoing light to direction v — incident light from direction omega — the BRDF — cosine term

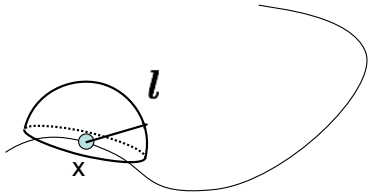


Sum (integrate) over every direction on the hemisphere, modulate incident illumination by BRDF

The Reflectance Equation

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

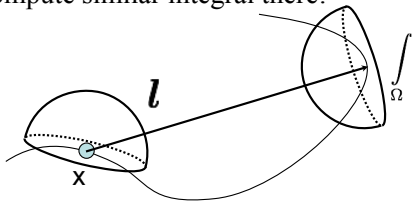
- Where does L_{in} come from?



The Reflectance Equation

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

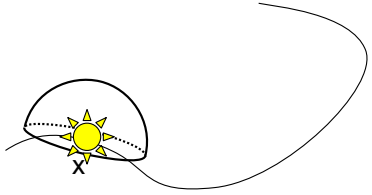
- Where does L_{in} come from?
  - It is the light reflected towards x from the surface point in direction ***l*** ==> must compute similar integral there!
    - Recursive!



The Rendering Equation

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

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



The Rendering Equation

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

- 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

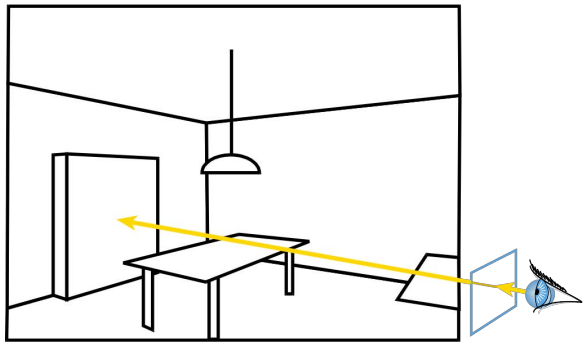
The Rendering Equation

- 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.

How to Compute Global Illumination

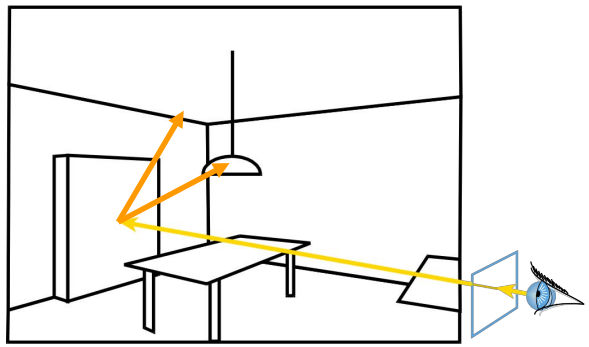
Ray Casting

- Cast a ray from the eye through each pixel



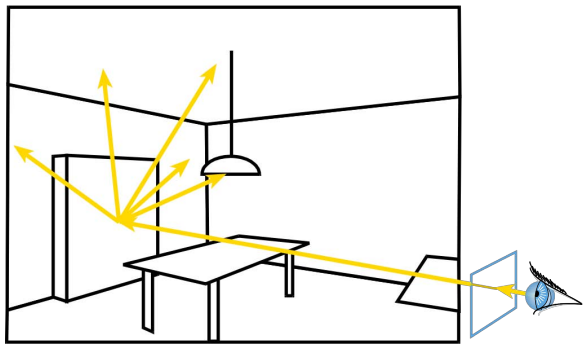
Ray Tracing

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



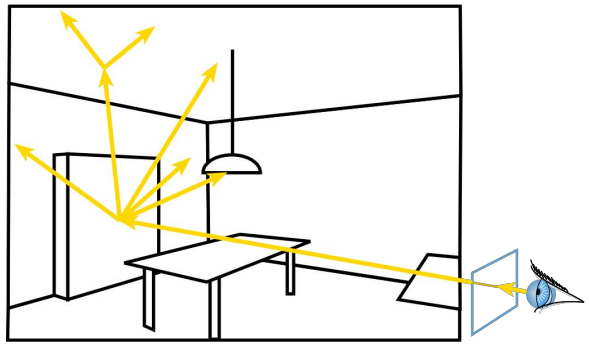
“Monte-Carlo Ray Tracing”

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



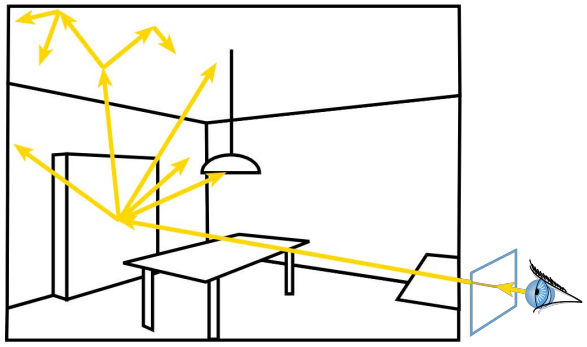
“Monte-Carlo Ray Tracing”

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



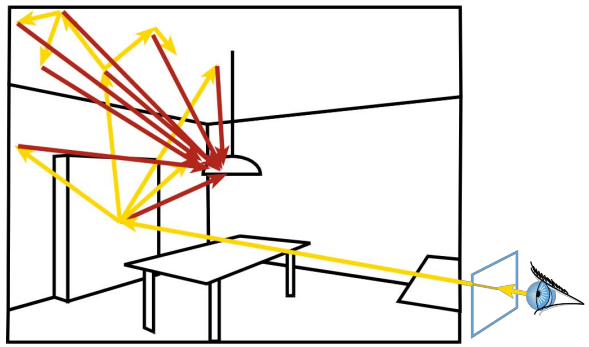
“Monte-Carlo Ray Tracing”

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

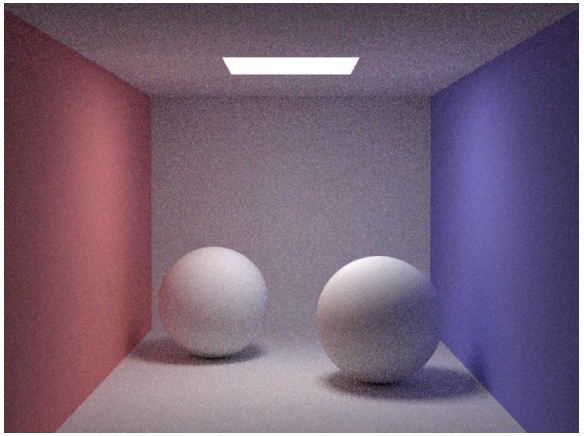


“Monte-Carlo Ray Tracing”

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



Results

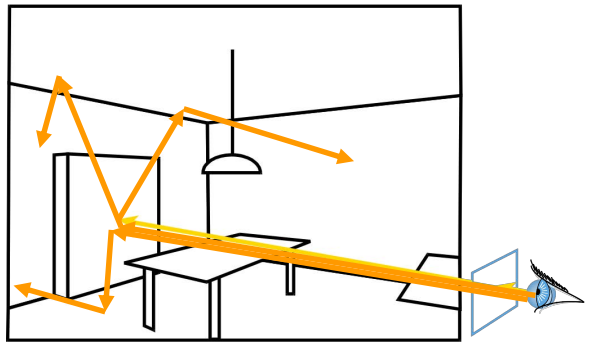


Henrik Wann Jensen

Courtesy of Henrik Wann Jensen. Used with permission.

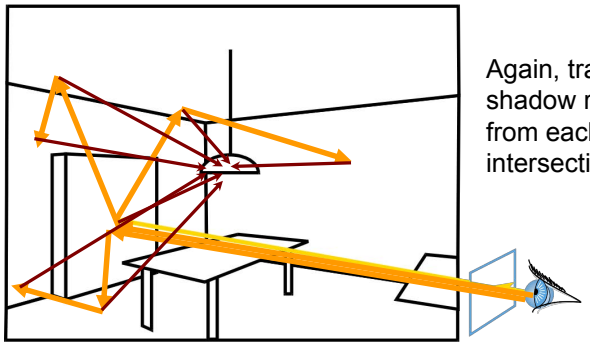
Monte Carlo Path Tracing

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



Monte Carlo Path Tracing

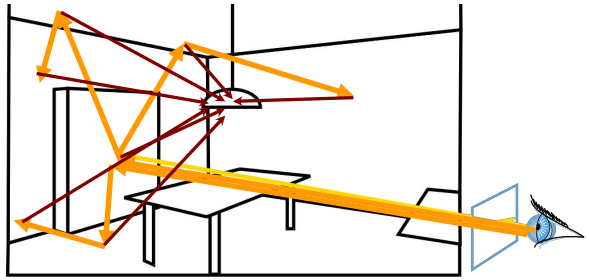
- Trace only one secondary ray per recursion
  - Otherwise number of rays explodes!
- But send many primary rays per pixel (antialiasing)



Again, trace shadow rays from each intersection

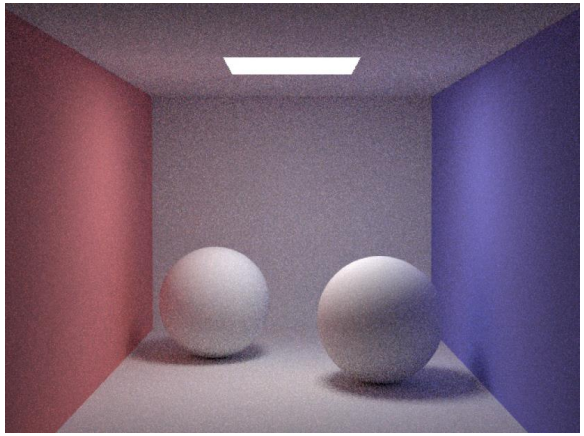
Monte Carlo Path Tracing

- 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

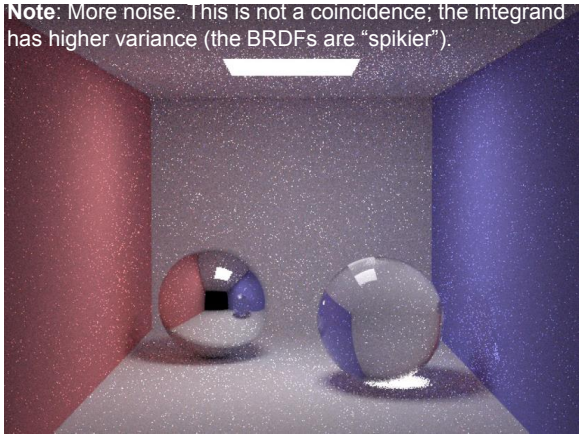
- 10 paths/pixel



Henrik Wann Jensen

Path Tracing Results: Glossy Scene

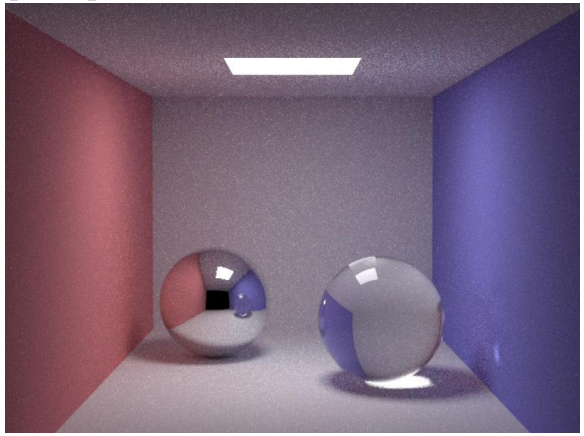
- 10 paths/pixel



Henrik Wann Jensen

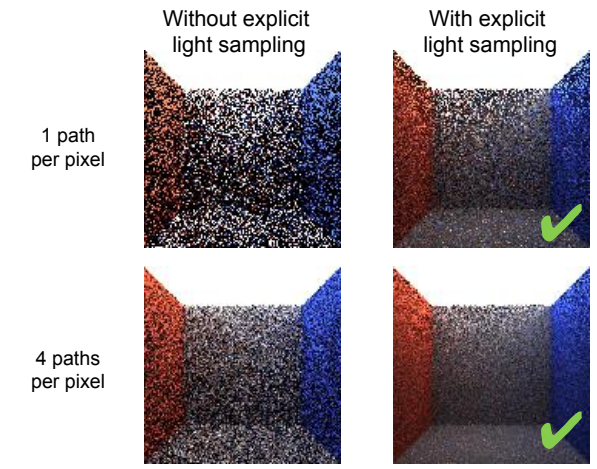
Path Tracing Results: Glossy Scene

- 100 paths/pixel



Henrik Wann Jensen

Importance of Sampling the Light



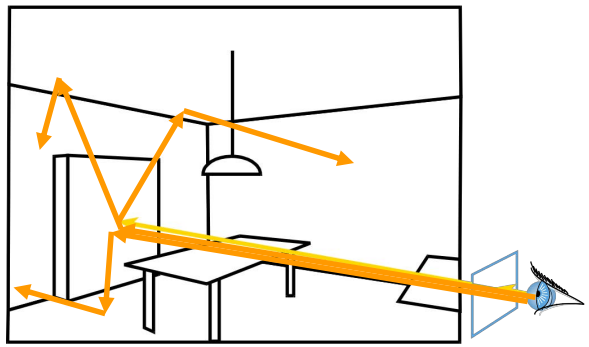
Questions?

- Vintage path tracing by Kajiya

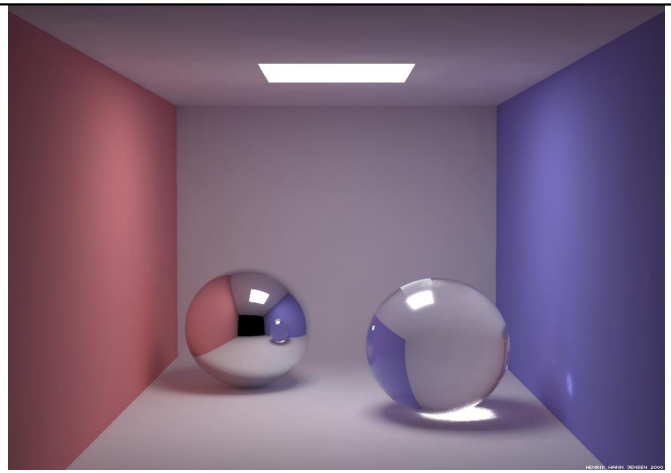
Irradiance Caching

Path Tracing is costly

- Needs tons of rays per pixel!

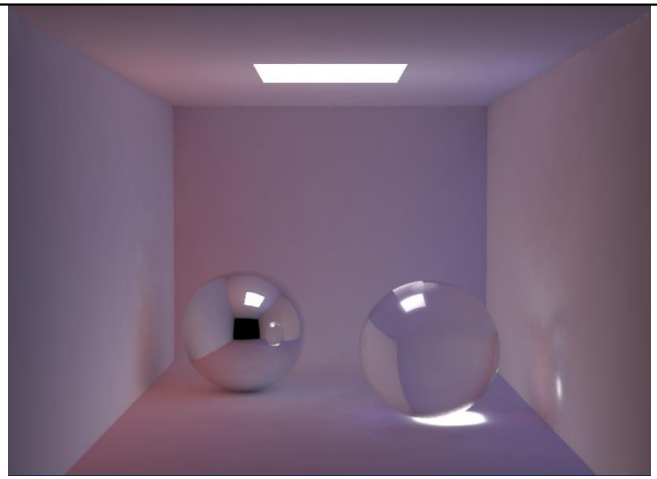


Global Illumination (with Indirect)



Courtesy of Henrik Wann Jensen. Used with permission.

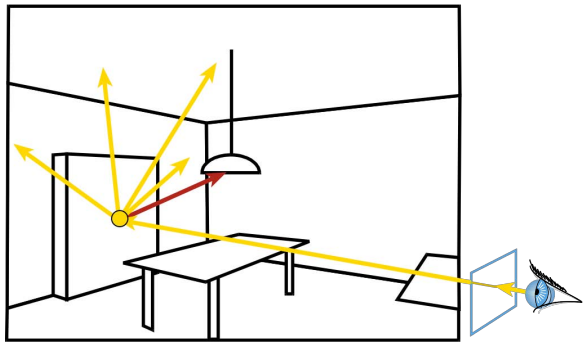
Indirect Lighting is Mostly Smooth



Courtesy of Henrik Wann Jensen. Used with permission.

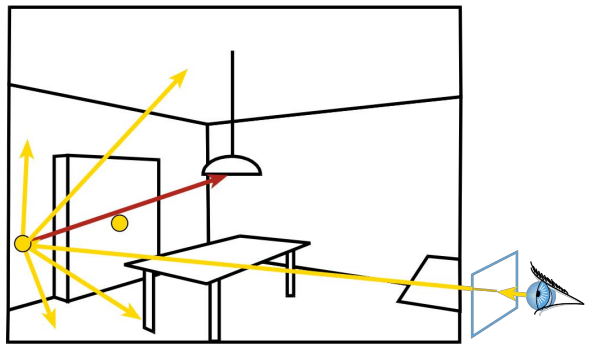
Irradiance Caching

- Indirect illumination is smooth



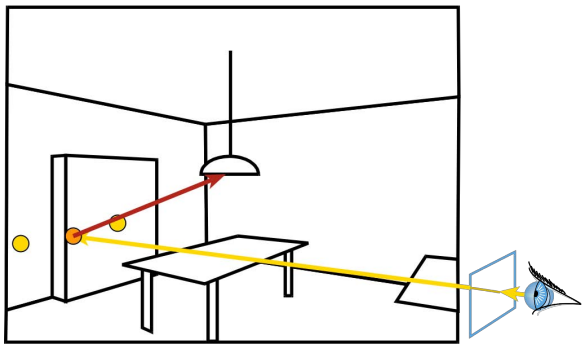
Irradiance Caching

- Indirect illumination is smooth



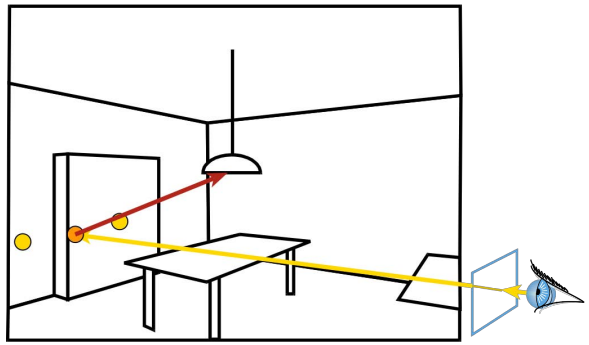
Irradiance Caching

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



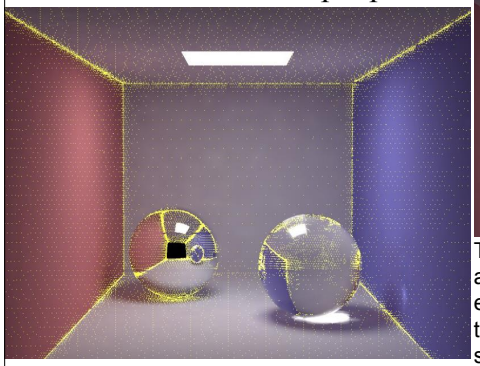
Irradiance Caching

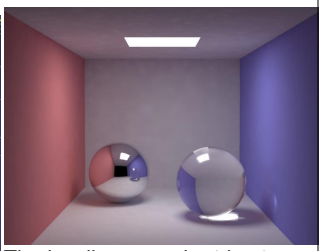
- Store the indirect illumination
- Interpolate existing cached values
- But do full calculation for direct lighting



Irradiance Caching

- Yellow dots:
  indirect diffuse sample points



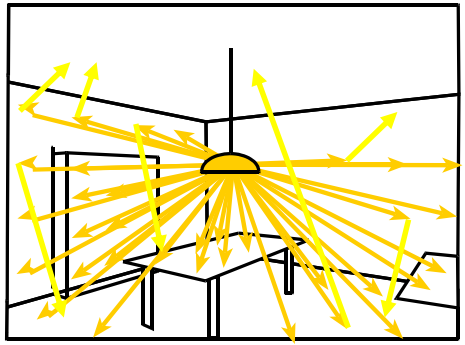


The irradiance cache tries to adapt sampling density to expected frequency content of the indirect illumination (denser sampling near geometry)

Photon Mapping

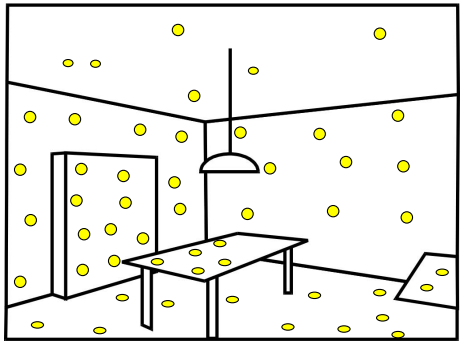
Photon Mapping

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



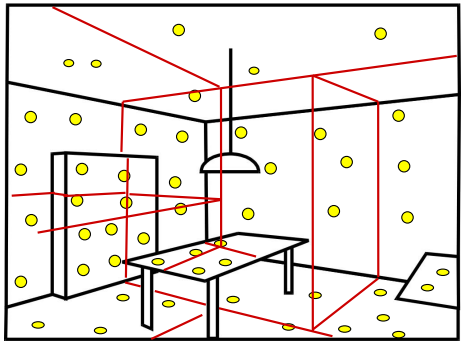
Photon Mapping

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



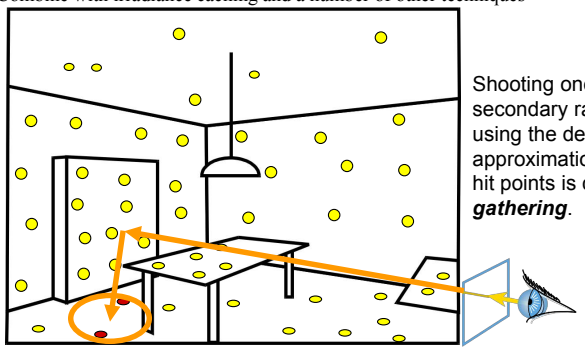
The Photon Map

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



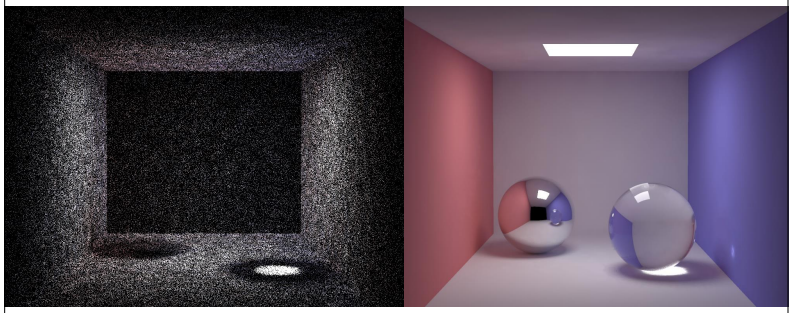
Photon Mapping - Rendering

- 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



Courtesy of Henrik Wann Jensen. Used with permission.

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.

Courtesy of Henrik Wann Jensen. Used with permission.

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
- 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!

Questions?

- Images by Veach and Guibas, SIGGRAPH 95

Naïve sampling strategy

Optimal sampling strategy

Lec14 Required Reading

- FOCG, Ch. 12.1