Global Illumination

Course web page:

http://goo.gl/EB3aA

Outline

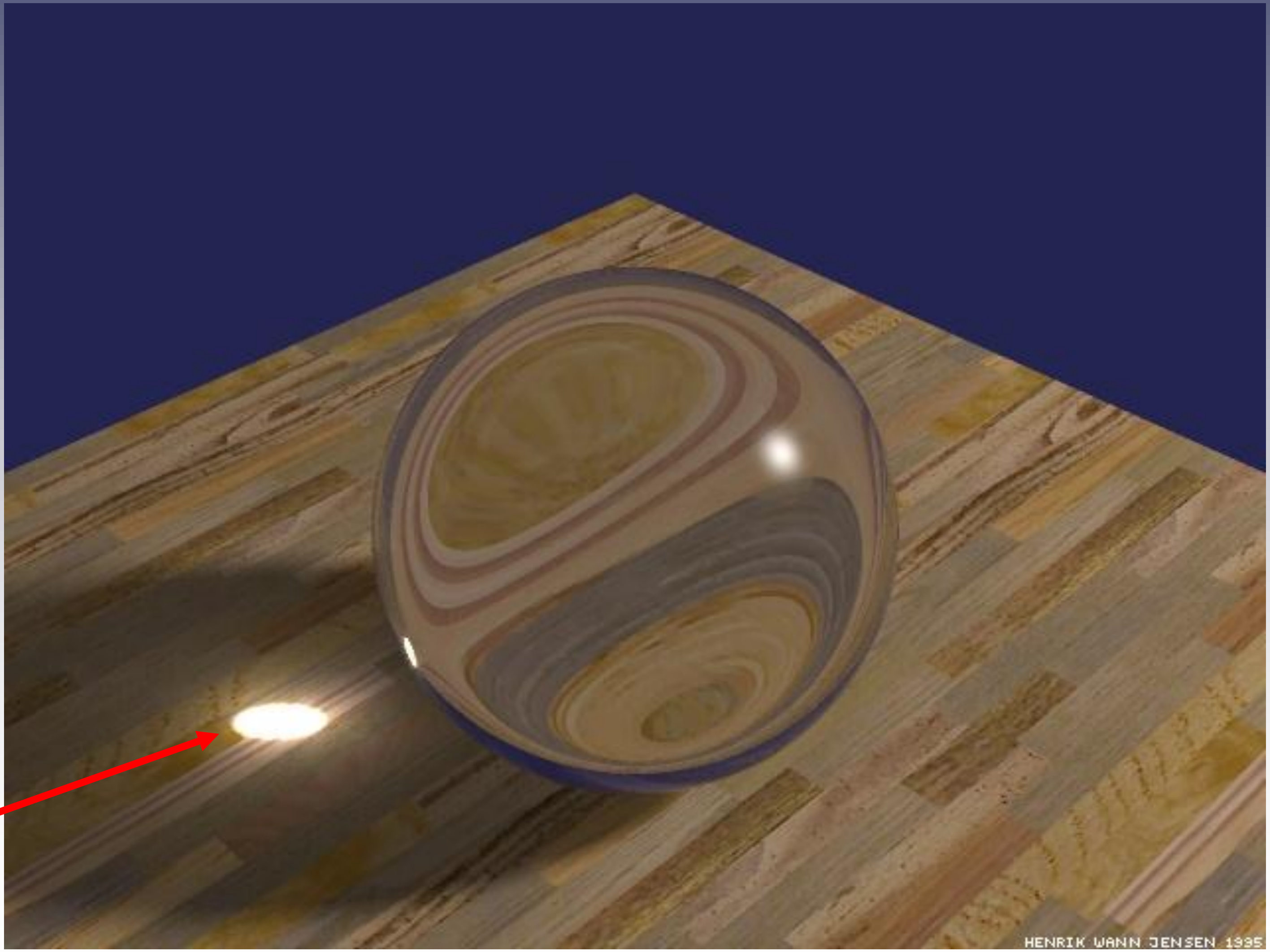
- HW #3
- Caustics
- Bidirectional ray tracing
- Photon mapping

HW #3 (due next Thursday, Apr. 26)

- Basic requirements
  - Complete shade_ray_**diffuse**()
  - Complete shade_ray_local(), which adds **specular and shadow** effects
  - Complete **reflection** component of shade_ray_recursive()
  - Add **sphere intersection testing** in intersect_ray_sphere()
  - Scene complexity and creativity
- Grad student requirements
  - Add support for **refraction** in shade_ray_recursive()
  - Add some version of adaptive supersampling, glossy reflection, ambient occlusion, or another advanced **distributed-ray** technique
  - Implement **bounding volumes** around objects to speed intersection calculations

Can Ray Tracing Do This?

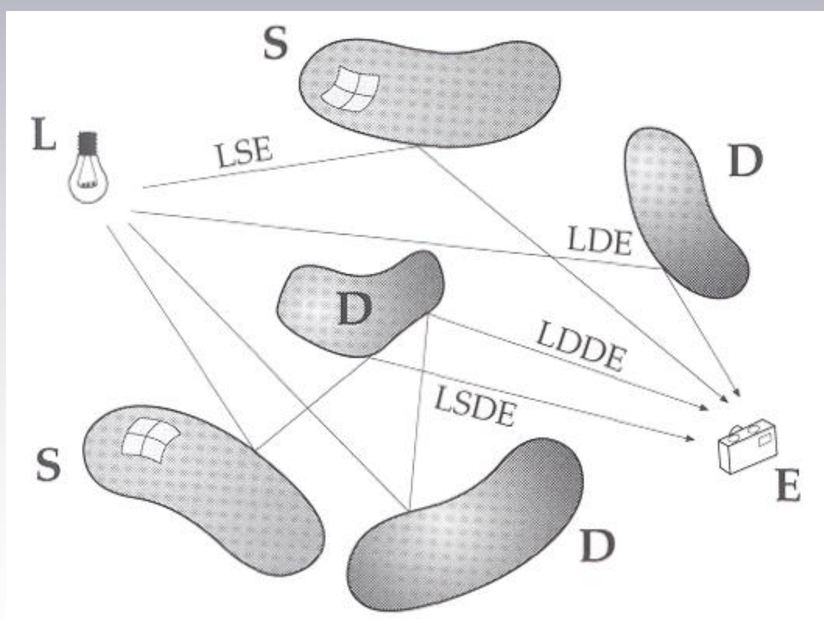
caustic



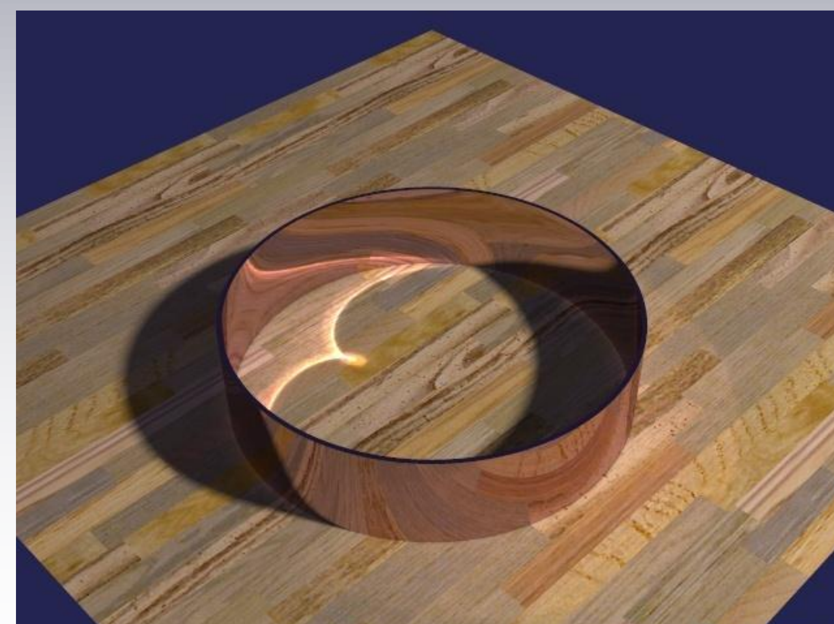
courtesy of H. Wann Jensen

Caustics

- Definition: (Concentrated) specular reflection/refraction onto a diffuse surface
  - In simplest form, follow an **LSDE** path
- Standard ray tracing **cannot** handle caustics—only paths described by **LDS*E**



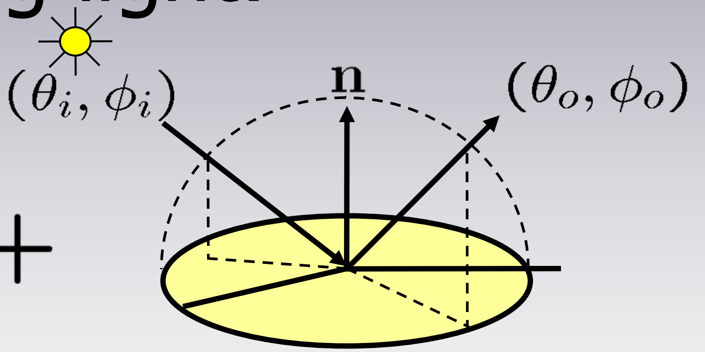
from Sillion & Puech



courtesy of H. Wann Jensen

More about caustics

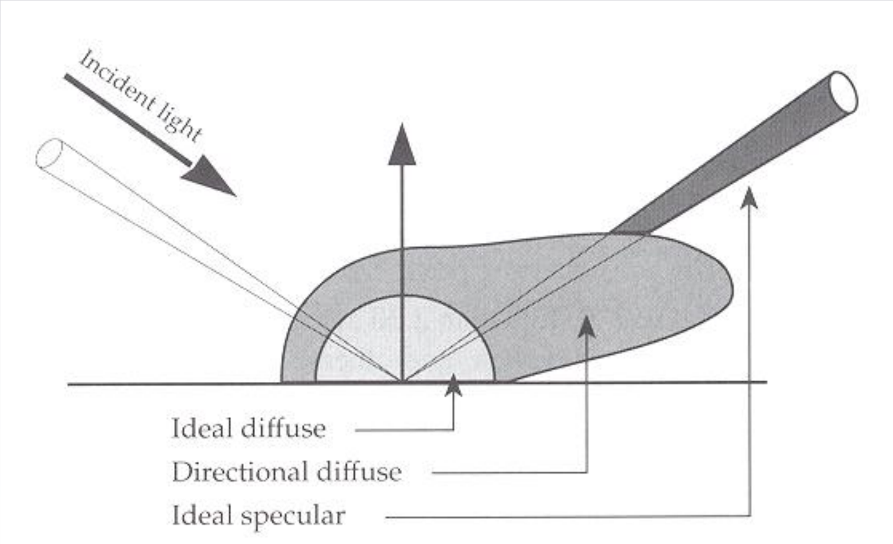
- What is the problem with **LS⁺DE** paths for ray tracing?
- Review: Radiance for a viewing direction given all incoming light:



$$L_o(\mathbf{x}, \theta_o, \phi_o) = \overbrace{L_e(\mathbf{x}, \theta_o, \phi_o)}^{\text{emitted light}} + \underbrace{\int_\Omega f(\theta_o, \phi_o, \theta_i, \phi_i) L_i(\mathbf{x}, \theta_i, \phi_i) \cos\theta_i \, d\omega}_{\text{reflected light}}$$

Review: BRDFs

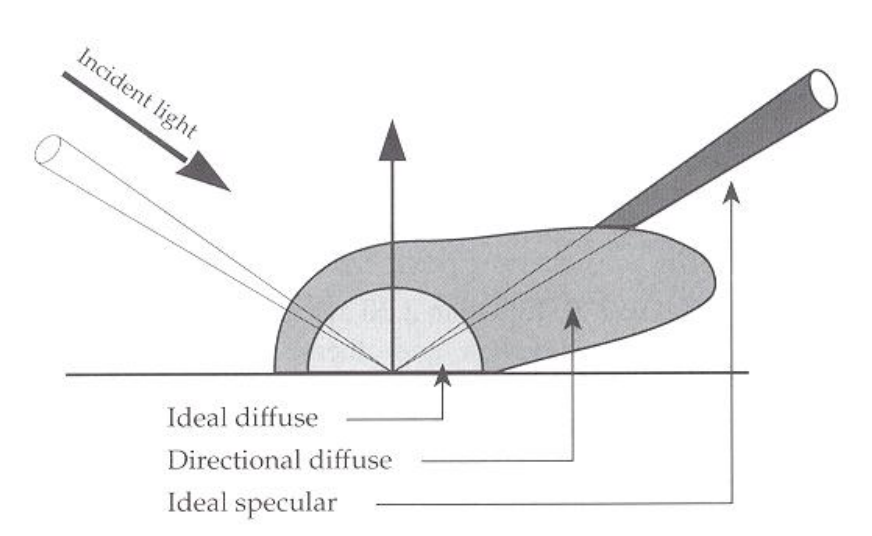
- **Bidirectional Reflectance Distribution Function** (BRDF): Ratio of outgoing radiance in one direction to incident irradiance from another
- Can view BRDF as **probability** that incoming photon will leave in a particular direction (given its incoming direction)



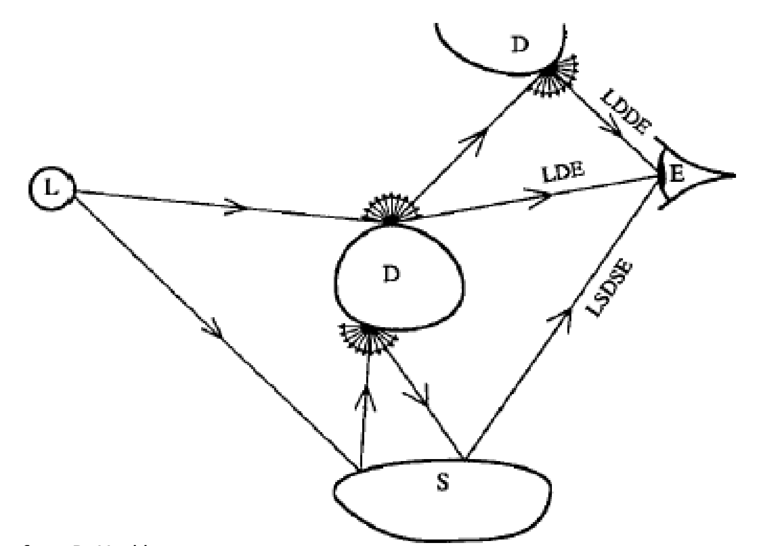
from Sillion & Puech

The Problem with Diffuse Surfaces

- For specular surfaces, when ray tracing we know where the photon "came from", whereas for diffuse surfaces there's much more uncertainty
  - If we're tracing a ray from the eye and we hit a diffuse surface, this uncertainty means that the source of the photon could be anywhere in the hemisphere
  - Conventional ray tracing just looks for lights at this point, but for **LS$^+$DE** paths we need to look for other specular surfaces
    - How to find them?



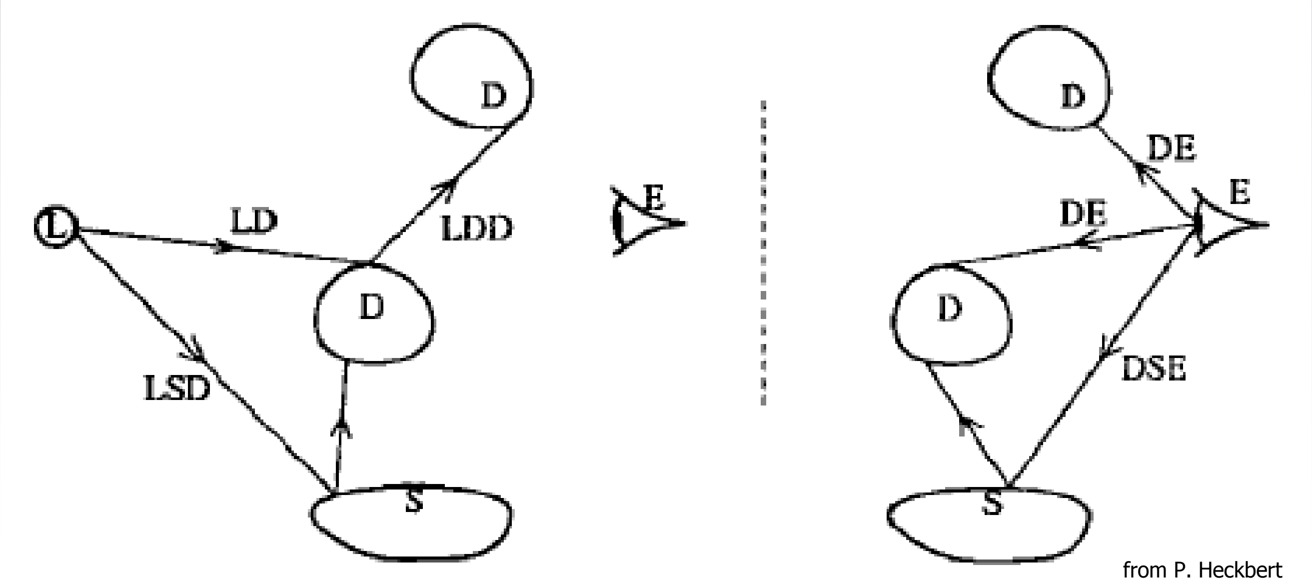
from Sillion & Puech



from P. Heckbert

Bidirectional Ray Tracing (P. Heckbert, 1990)

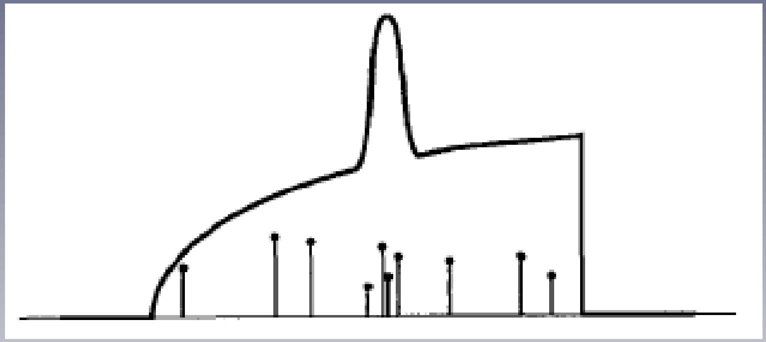
- Idea: Trace forward **light rays** into scene as well as backward **eye rays**
- At diffuse surfaces, light rays additively "deposit" photons in **radiosity textures**, or "rexes", where they are accessed up by eye rays
  - Summation approximates integral term in radiance computation
  - Light rays carry information on specular surface locations—they have no uncertainty



from P. Heckbert

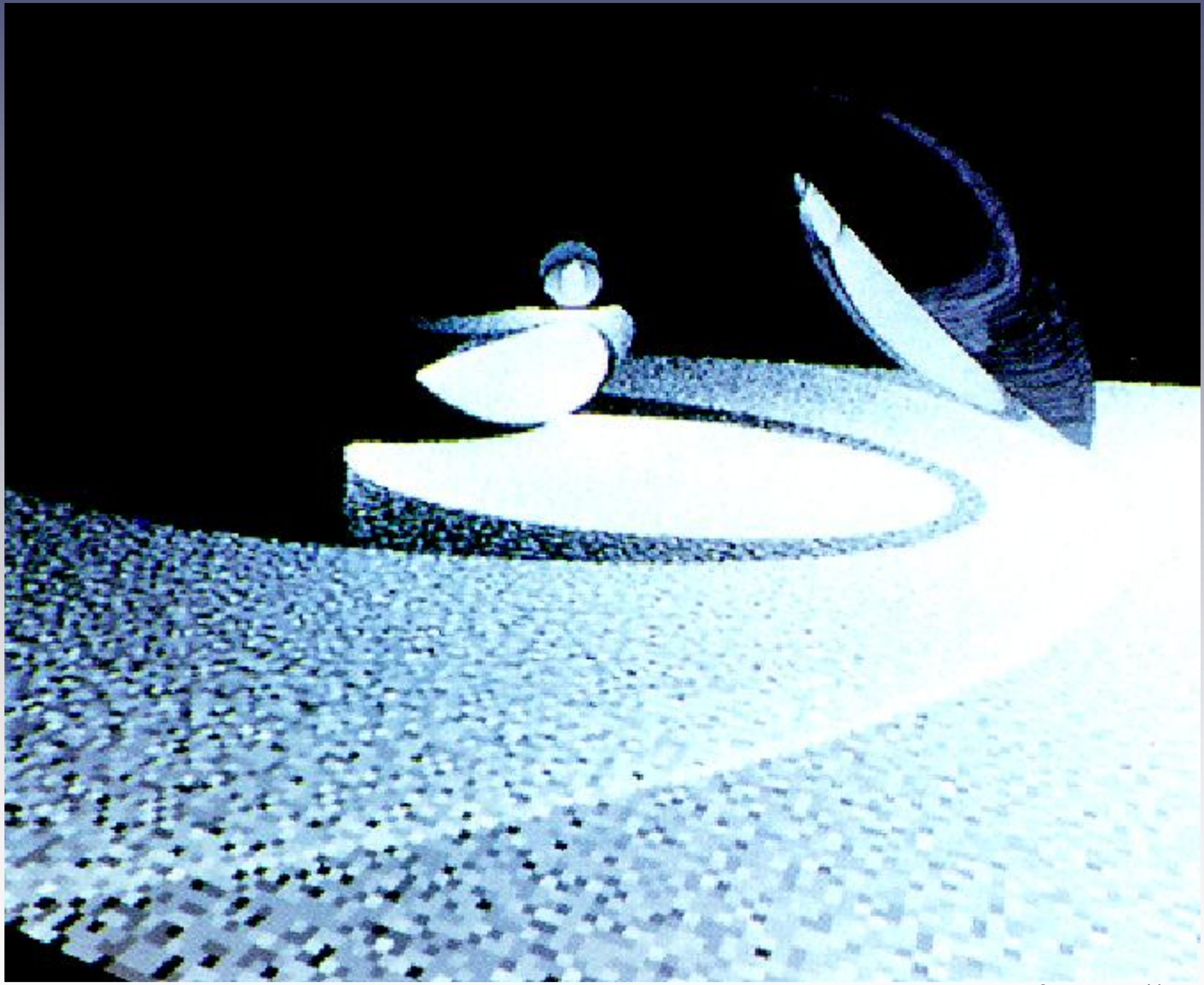
Bidirectional Ray Tracing: Notes

- This kind of bidirectional ray tracing simulates **LS***DS***E** paths
- Photons deposited in rexes are sparse, so they must be interpolated
  - Use density estimation
  - Still have noise issues
- Storage of illumination only on surfaces means that we ignore fog and other volume-based scattering/absorption (aka "participating media")



from P. Heckbert

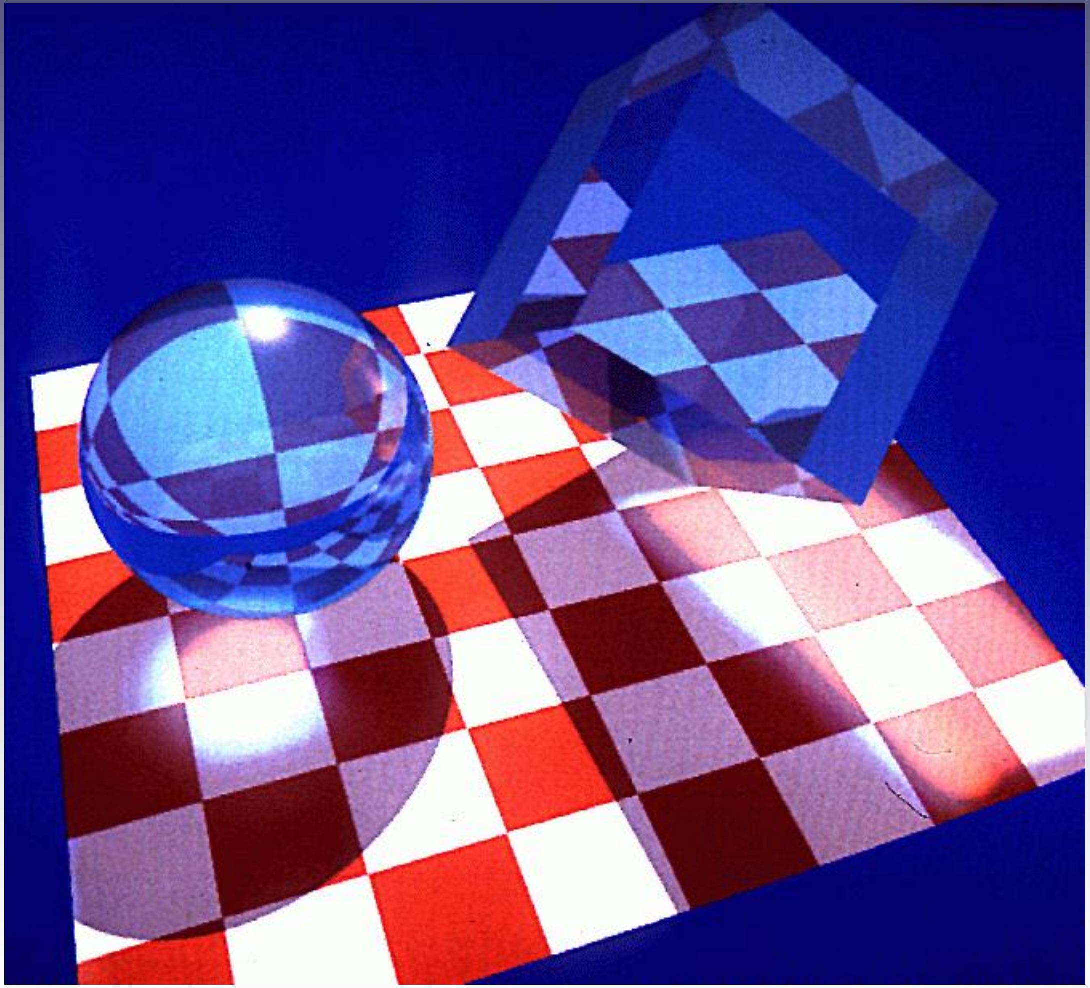
Bidirectional Ray Tracing: Results



from P. Heckbert

Lens, mirrored sphere, and diffuse surface with caustic of focused light

"Backwards" Ray Tracing (1986)

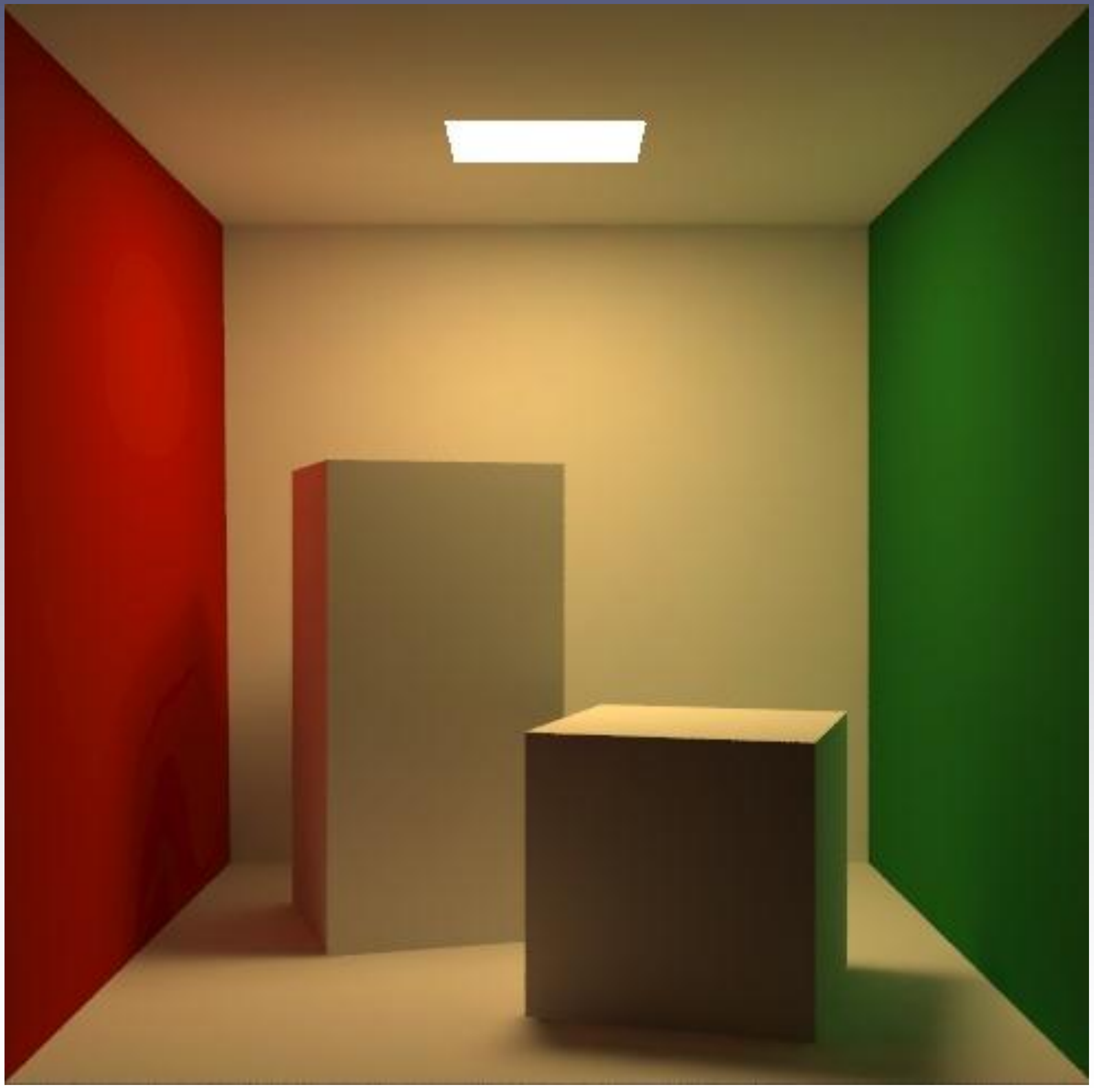


courtesy of J. Arvo

A technique similar to Heckbert's was used to form this image

What's Still Missing?

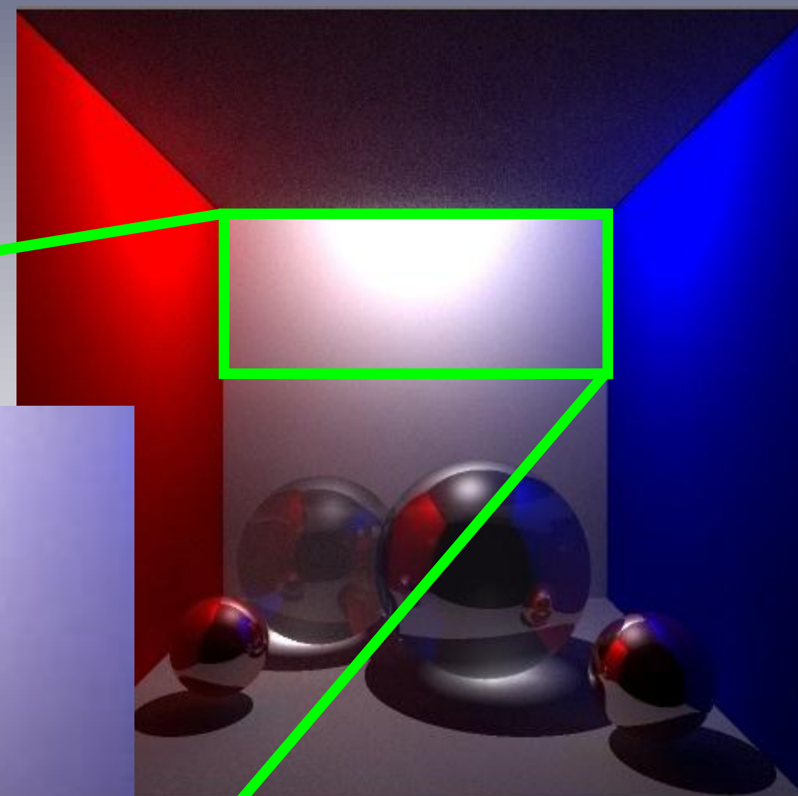
An **LD*E** scene



courtesy of Cornell

Color Bleeding

- Transfer of color between diffuse surfaces via reflection



courtesy of K. Fatahalian & J. Hui

Photon Mapping (H. Jensen, 1996)

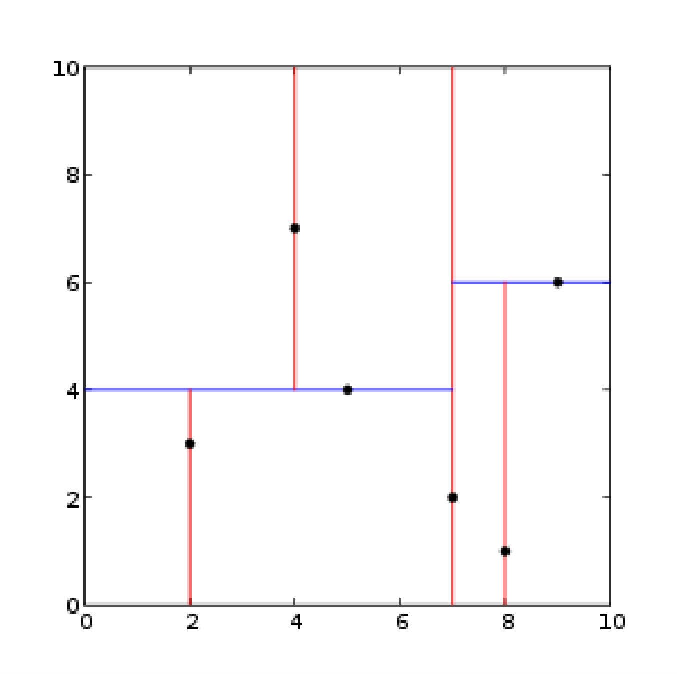
- Two-pass algorithm somewhat like bidirectional ray tracing, but photons stored differently
  - Related to *particle tracing* approach in Shirley 24.1
- 1^{st} pass: Build *photon map* (analog of rexes)
  - – Shoot random rays from light(s) into scene
  - – Each photon carries fraction of light's power
  - – Follow specular bounces, but store photons in map **at each** diffuse surface hit (or scattering event)
- 2^{nd} pass: Render scene
  - – Modified ray tracing: follow eye rays into scene
  - – Use photons near each intersection to compute light

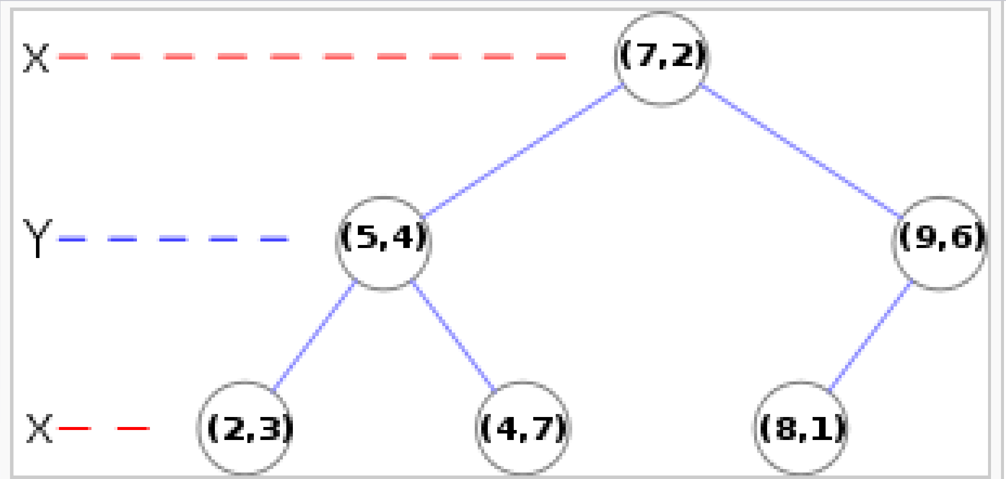
Photon Mapping: 1st pass

- Probabilistically decide on photon reflection, transmission, or absorption based on material properties of object hit
  - Specular surface: Send new photon (with scaled-down power) in reflection/refraction direction just like ray tracing
  - Diffuse surface: If at least one bounce, **store photon** in photon map, send new photon in random direction (usually cosine distribution, see Shirley 14.4.1)
    - So do NOT store photon at specular interactions
  - Arbitrary BRDF: Use BRDF as probability distribution on new photon's direction
- Photon map is **kd-tree**
  - Decoupling from scene geometry allows fewer photons than scene objects/triangles (no texture maps, no meshes)

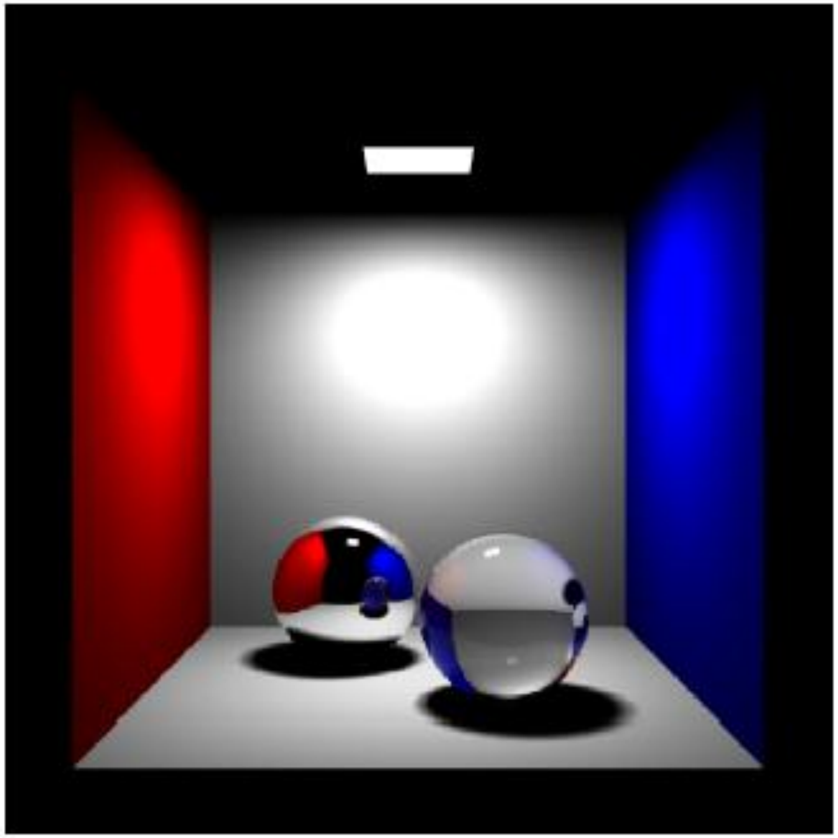
kd-trees

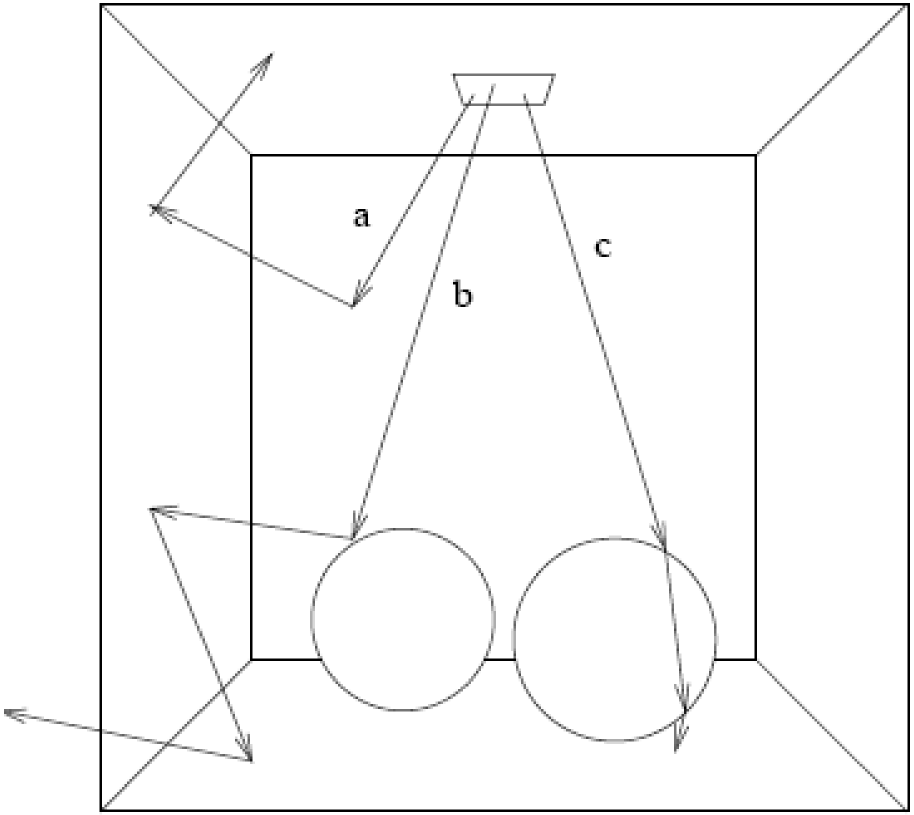
- Related to BSP trees: each point parametrizes axis-aligned splitting plane; rotate which axis is split
- But balance <u>is</u> important to get O(log N) efficiency for nearest-neighbor queries
- Example kd tree for k = 2 and N = 6:





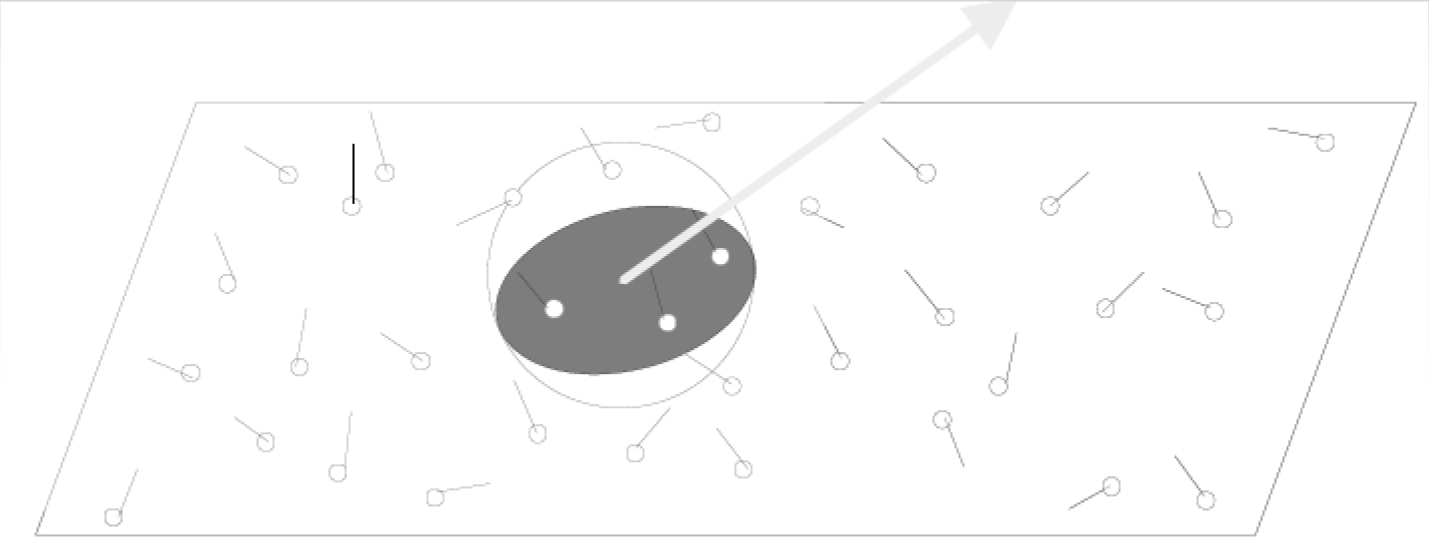
Photon Map: Example





Photon Mapping: 2^{nd} pass

- For each eye ray intersection, estimate irradiance as function of nearby photons
  - Each photon stores position, power, incident direction—can treat like mini-light source
  - Use filtering (cone or Gaussian) to weight nearer photons more
  - Can use discs instead of spheres to only get photons from same planar surface
- Irradiance estimates are combined with standard local illumination calculations in **final gathering**—just like ray tracing adds reflection/refraction components to local color
- As usual, more accurate with more photons → Use **multiple maps for different phenomena**



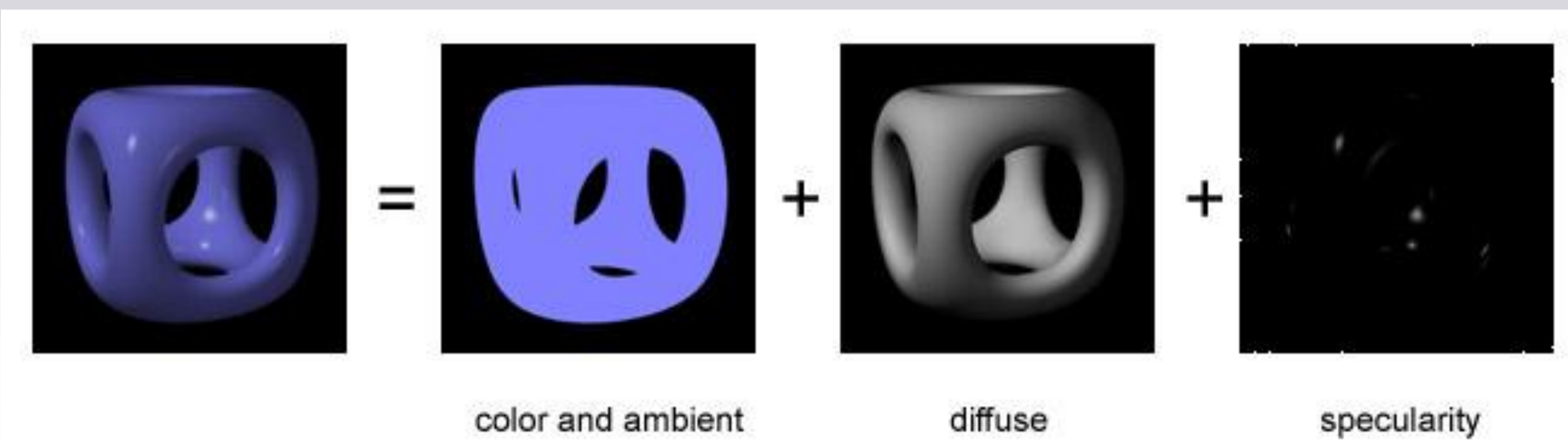
Reflectance equation: Total illumination

- For greater control of appearance, a different light radiance is typically specified in OpenGL for each type of reflectance

$$s_{diff}, s_{spec}, s_{amb}$$

- Actual light at a pixel is combination of three effects:

$$i_{total} = i_{amb} + i_{diff} + i_{spec}$$



from Wikipedia

Lighting Components, Reconsidered

- Break rendering equation into parts:
  $L = L_{direct} + L_{specular} + L_{indirect} + L_{caustic}$
- Can get L_{direct} and $L_{specular}$ using ray-casting, ray-tracing respectively
- $L_{indirect}$ is main reason we're looking at photon mapping—it's our LD*E paths
- $L_{caustic}$ from special "caustic" photon map

Photon Mapping: Diffuse Lighting

courtesy of S. Agarwal

Direct Lighting only

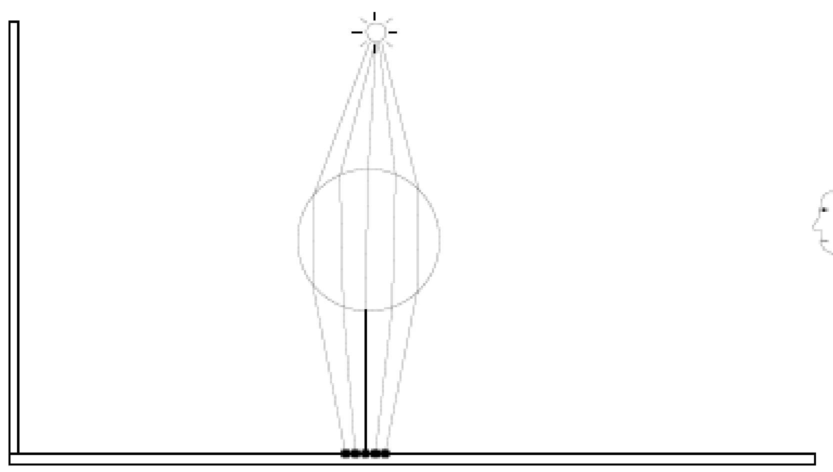
courtesy of S. Agarwal

Indirect Illumination only

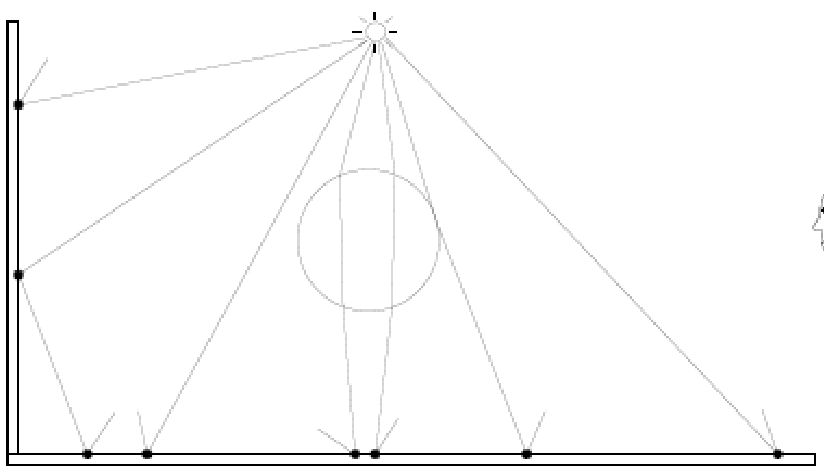
courtesy of S. Aggrwal

Multiple Photon Maps

- Global map: Shoot photons everywhere for diffuse, indirect illumination
- Caustic map: Shoot photons only at specular objects ("aimed" sort of like shadow rays)
- Volume map: Photon interactions with participating media such as fog, smoke



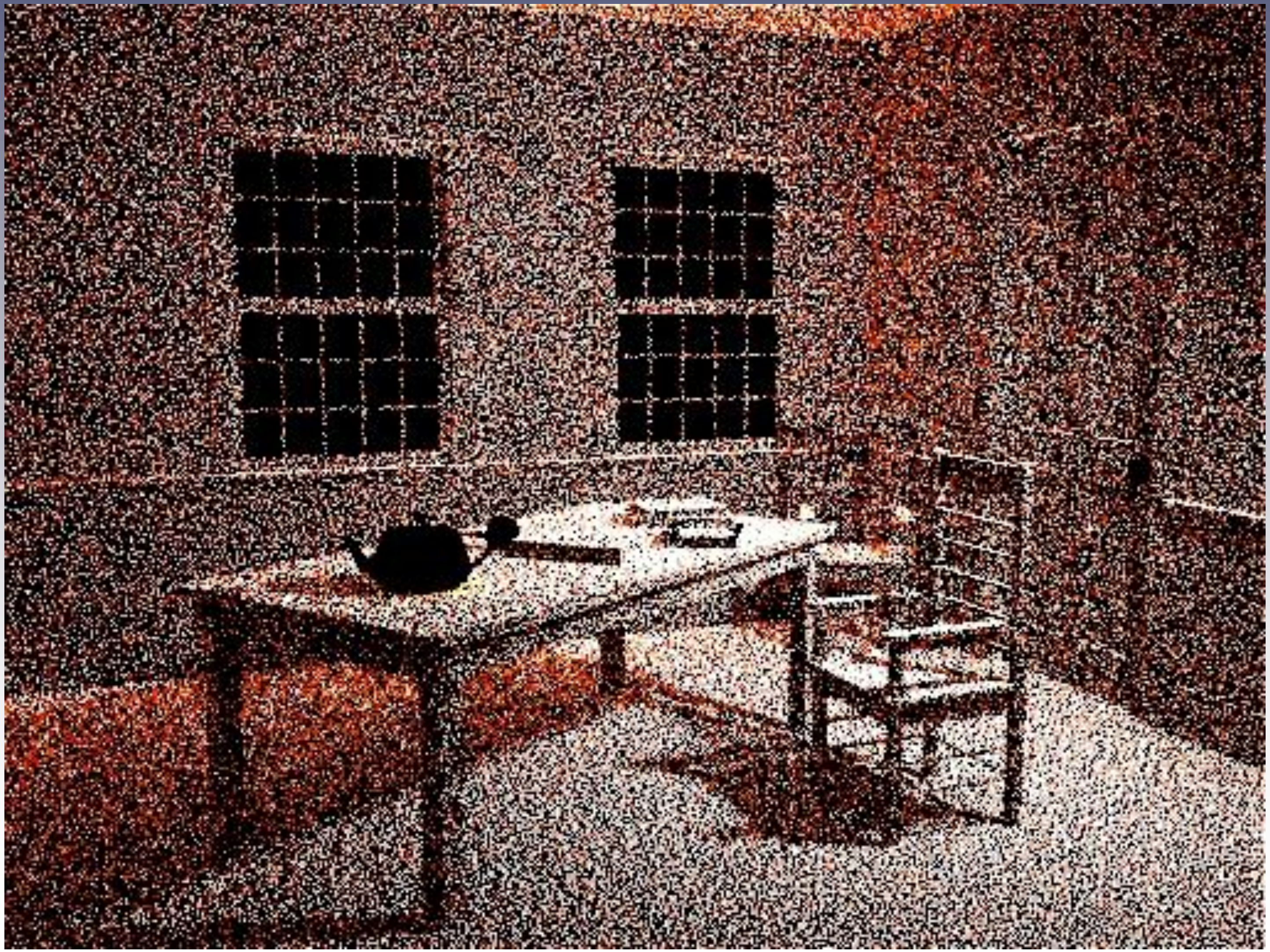
Caustic map



Global map

Raytraced scene (courtesy of P. Christensen)

Photon map of scene (n=500,000) [notice nothing stored at specular surfaces]



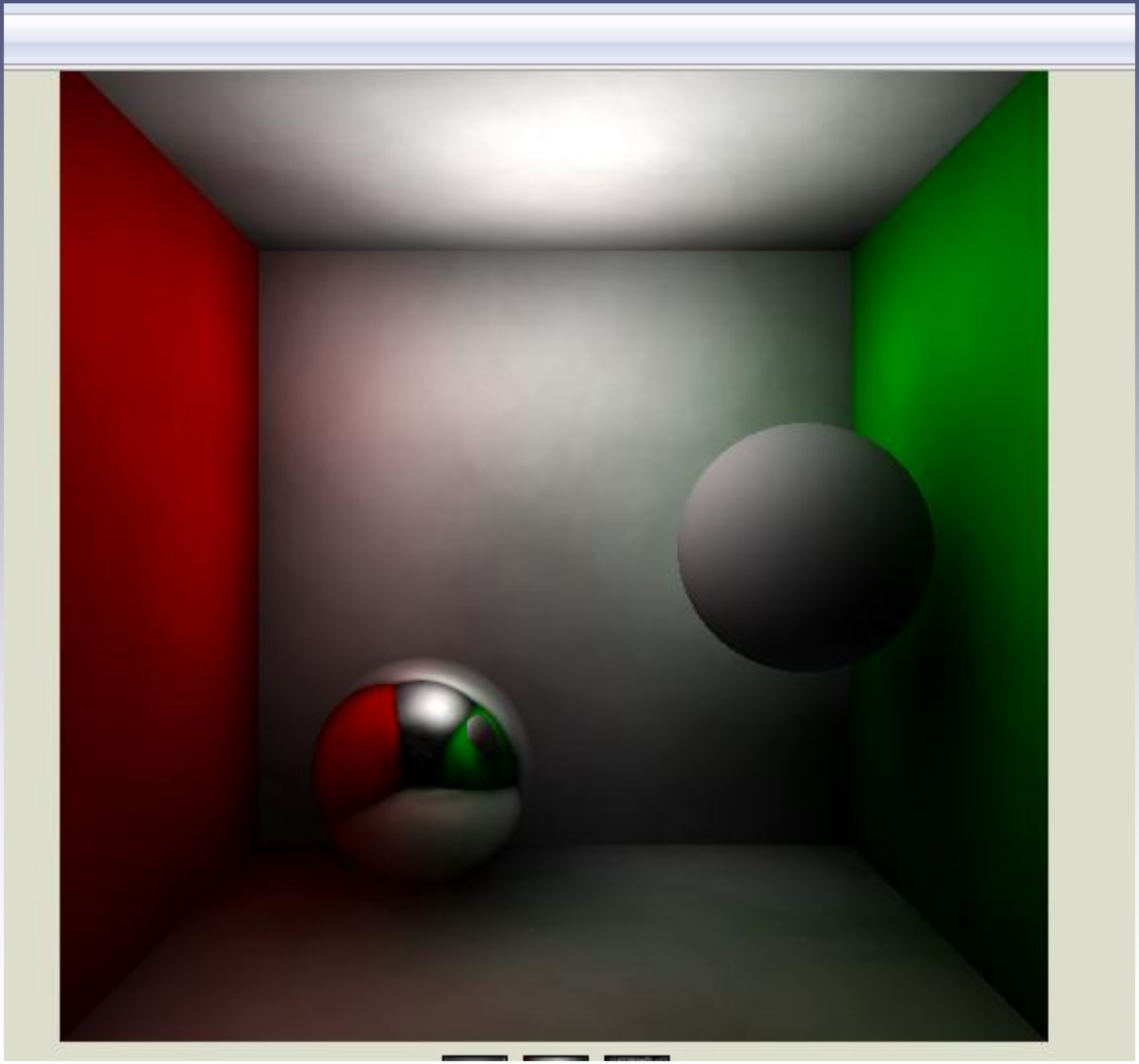
Irradiance estimates based on nearby photons

Previous image combined w/ texture maps & material colors

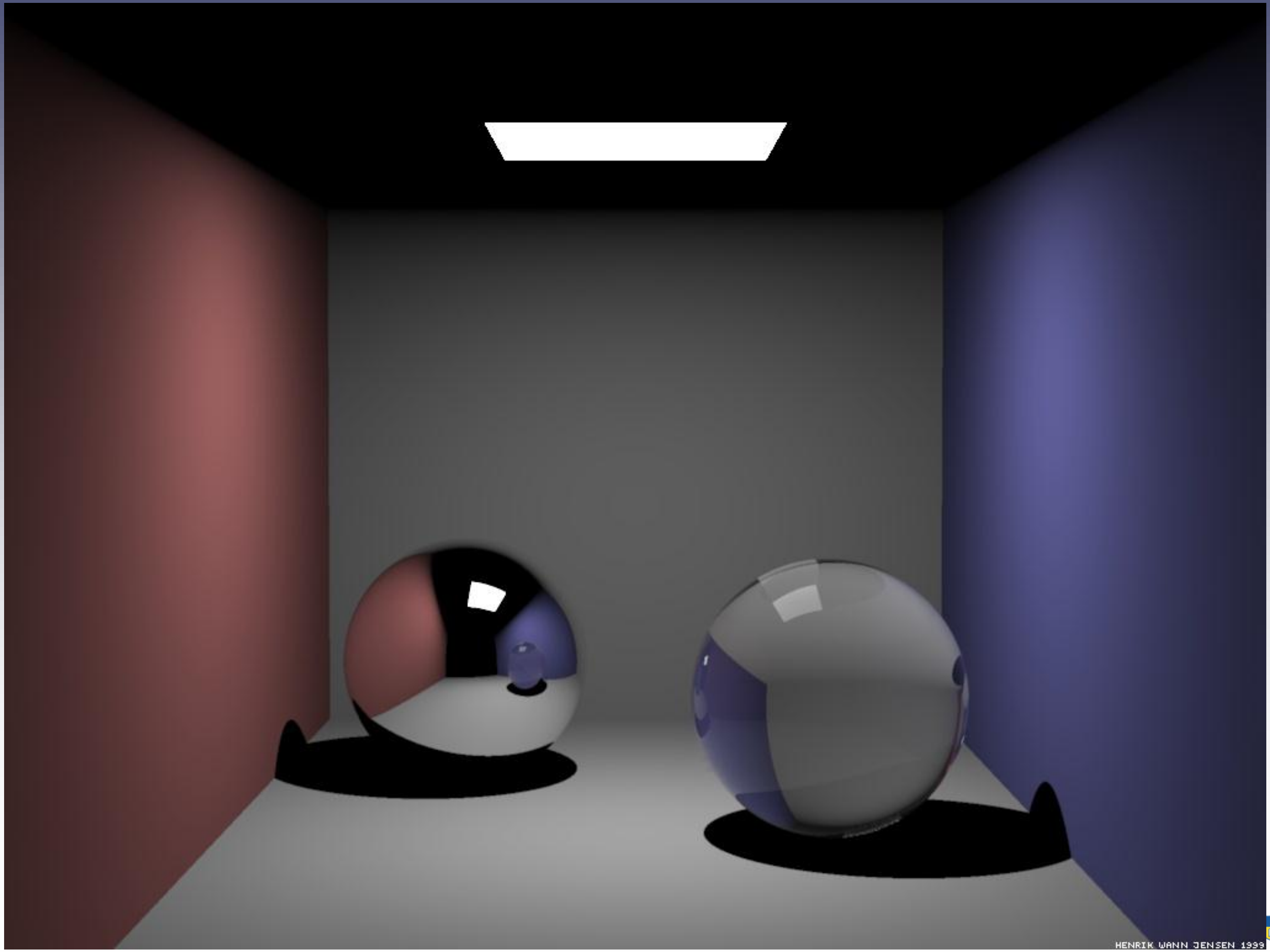
Scene after final gathering

Raytraced scene (courtesy of P. Christensen)

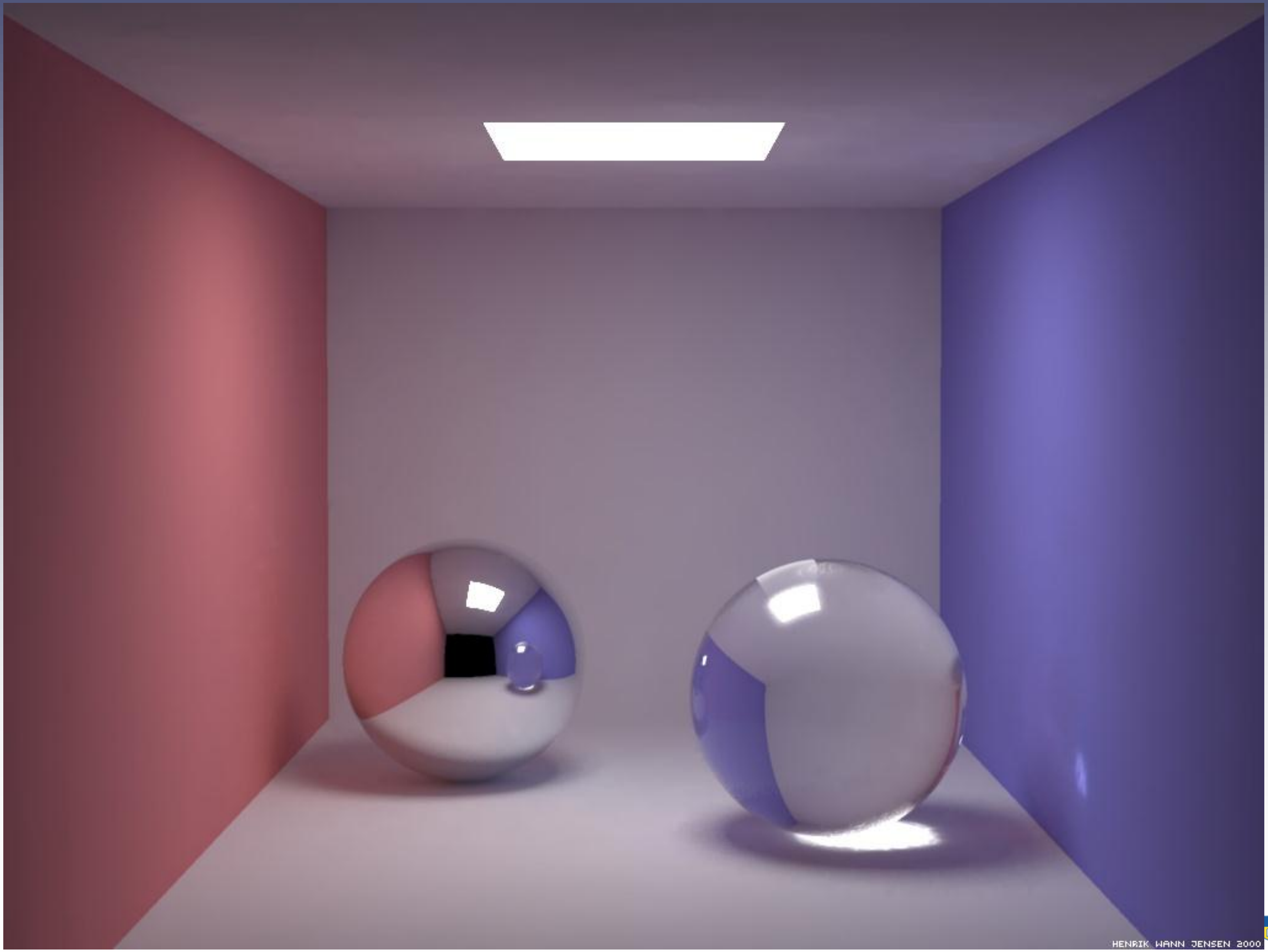
Go to interactive photon mapping demo



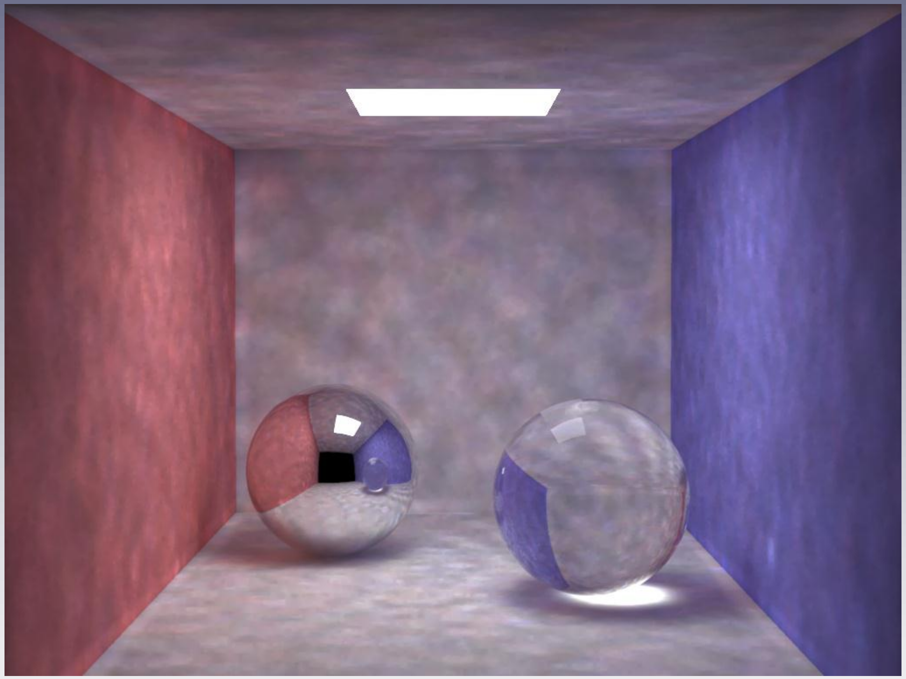
Ray Tracing



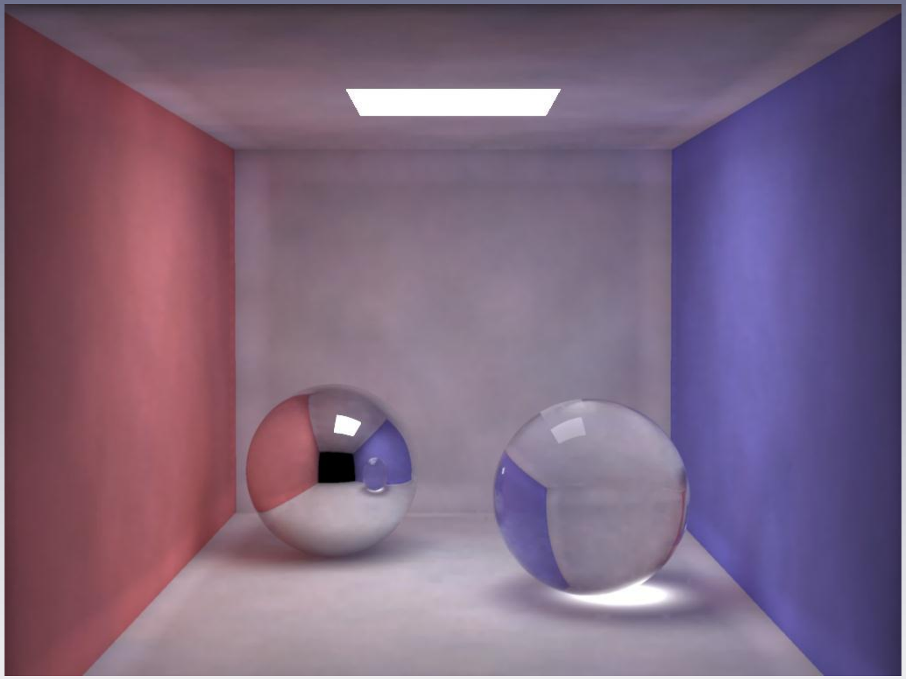
Photon Mapping



Visualization of Radiance Estimates



200,000 photons in global map;
100 nearest photons in each
radiance estimate



500 photons per
radiance estimate
(note incorrect bleeding
near edges/corners)

Example: Water caustics

Example: Smoke (volume map)