Now Playing:

Thunderhorse
Dethklok
from *Dethalbum*
Released September 25, 2007

The Light of Mies van der Rohe

Available online at:
http://www.youtube.com/watch?v=ACbdCrGblSo

Beyond Raytracing: Monte Carlo Methods

Adapted from a slide set
created by Anselmo Lastra

Rick Skarbez, Instructor
COMP 575
November 15, 2007

Announcements

- Programming Assignment 4 (Ray tracer) is out, due Tuesday 11/20 by 11:59pm
  - Any questions?

Distributed Ray Tracing

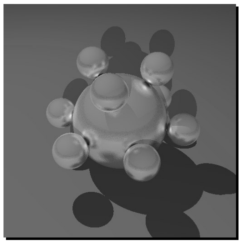
Cook et al., 1984

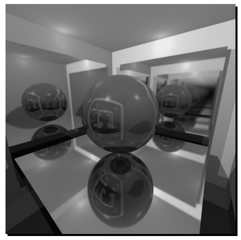
- So what are some of the effects we can generate?
  - Antialiasing
    - Distribute rays across each pixel
  - Glossy reflections
    - Distribute multiple reflection rays instead of just one

Stochastic Ray Tracing

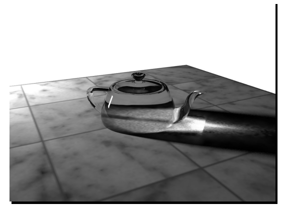
- So what are some of the effects we can expect this way? (cont'd)
  - Soft shadows
    - Distribute multiple rays to an area light source
  - Depth of field
    - Distribute rays across a lens
  - Motion blur
    - Distribute rays over time

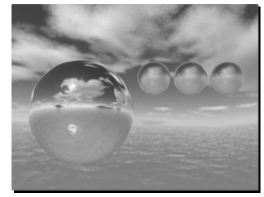
Glossy Reflection Examples



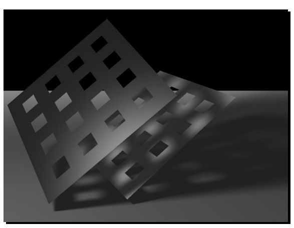


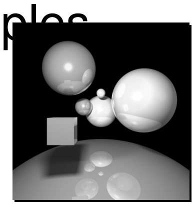
Translucency Examples

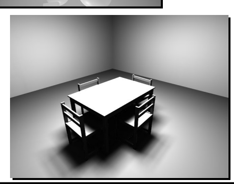




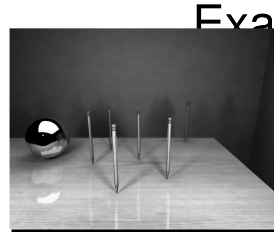
Soft Shadow Examples

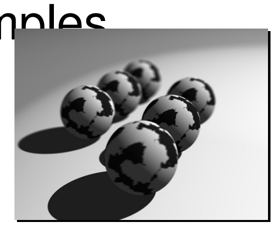




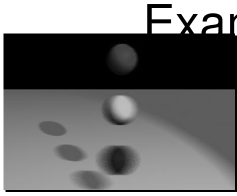


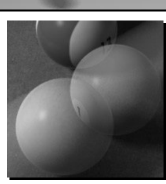
Depth of Field Examples

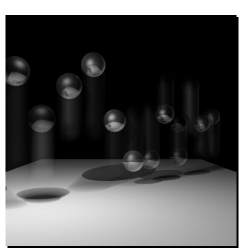




Motion Blur Examples







Distributed Ray Tracing Review

- We introduced the concept of distributed ray tracing
  - NOTE: Don’t confuse this with the way the word “distributed” is commonly used in CS
- Showed some examples of how it can be used to generate more realistic images
- Basic idea: Replace a single ray with many

Done with (Standard) Raytracing

- So that's all we have to say about standard (one-way) ray tracing
- Basic technique: Shoot rays from the eye, trace them back to the lights
- Gives us shadows, reflection, refraction
- Distributed ray tracing gives us even more
  - Gloss, translucency, soft shadows, lens effects

So, what else is there?

Classifying Light Transport Paths

Heckbert, SIGGRAPH 90

- Paul Heckbert proposed a way of classifying light transport paths
  - And thereby stating which cases a renderer can (or can't) handle

Heckbert's Notation

- L : a light
- E: the eye
- S: a specular surface
- D: a diffuse surface
- G: a glossy surface
  - Not always included
- An example: the path from a light, to a diffuse surface, to the eye can be written LDE

Radiosity

- Radiosity is an alternative lighting solution
  - It is nearly the opposite of raytracing, in terms of what effects each method is good at
    - Radiosity yields "global illumination", that is to say, diffuse-diffuse interactions
      - But not reflection or refraction
- Radiosity for lighting grew out of a similar technique used for simulating heat transfer

Classifying Renderers

- Radiosity
  - LD*E
  - Can handle arbitrarily many diffuse-diffuse interactions
  - No reflections
    - Note that this makes the radiosity solution for a scene view independent

Radiosity Assumptions

- Essentially, radiosity treats all surfaces in a scene as emitters (or potential emitters)
  - All surfaces are opaque
  - All surfaces are diffuse
  - Objects are in a vacuum (a pretty fair assumption)

Radiosity Benefits

- Our first real "global illumination" solution
  - Now we can handle diffuse-diffuse interactions
  - Don't have to do "ambient light" hacks anymore
- Solved in object space
  - Totally view independent
  - Can precompute radiosity and "bake it in" to a texture

The Radiosity Equation

- For convenience, move the (1 / π) term into *G*
- Bring back the emissive term, and we have

$$B(\mathbf{x}) = E(\mathbf{x}) + \rho(\mathbf{x})\int_S B(\mathbf{x}')G(\mathbf{x},\mathbf{x}')dA'$$

- Now we have radiosity at each point expressed only in terms of radiosity at each other point

Radiosity Method

1. Subdivide the model into elements.
2. Select locations (nodes) on elements at which to solve for radiosity.
3. Select basis functions to approximate radiosity across the element, based on values at nodes. Most common is to assume constant value of radiosity across the element, so a single node is placed in the middle.
4. Select finite error metric. This will result in a set of linear equations.

Radiosity Method

1. Compute coefficients of linear system. These are based on the geometric relationships between elements, called the form factors.
2. Solve the system of linear equations.
3. Reconstruct the radiosity function. Used to just assign radiosity values to vertices. Now textures common.
4. Render – often Gouraud interpolation of radiosity values at vertices.

In Short

- Build a really big linear system
  - Radiosity for each patch is one variable
- Solve the whole gosh-darn thing

Radiosity Review Over

- Any questions?

The Missing Link

- So now we can handle reflection, refraction, shadows, lens effects, etc.
  - In our raytracer
- And we've seen one way to do diffuse-diffuse interactions
  - In a radiosity solution
- But the rendering holy grail is to get it all at once

Topics

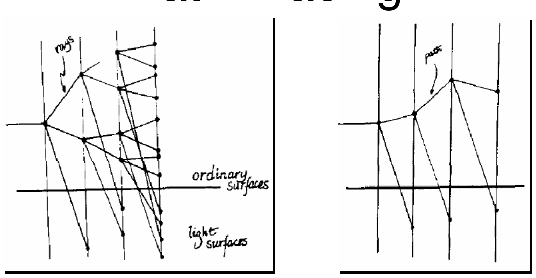
- Path Tracing
- Bi-Directional Path Tracing
- Metropolis Light Transport
- Photon Mapping

Path Tracing

Kajiya, SIGGRAPH 86 (again)

- How do we get both (ray tracing and global illumination effects) at once?
  - That is, how do we approximate the solution to the full rendering equation
- Path tracing extends ray tracing to approximate diffuse interactions

Path Tracing



Trace one ray in one (random) direction, and one ray to a light

Path Tracing

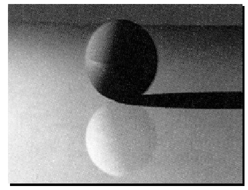
- Remember why we didn't want to model diffuse interactions with our raytracer?
  - Need many many rays to model diffuse reflection
- Path tracing randomly samples one ray direction each time
  - Need many rays per pixel
    - But get diffuse effects in the limit

Path Tracing Method

- At each ray intersection
  - Generate one ray based on diffuse / specular / transmissive coefficients
    - Not random; proportional to distribution
  - Also, generate one random ray per light
- Need a lot of rays per pixel
  - Kajiya used 40

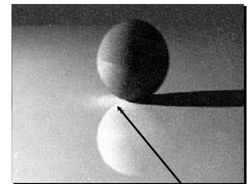
Path Tracing Results

Ray Traced Image



401 minutes

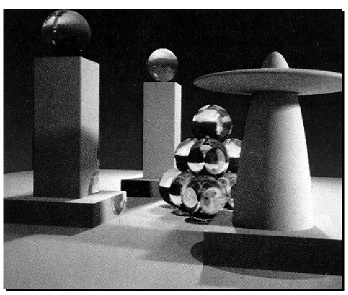
Path Traced Image



533 minutes

Note the light scattered off the diffuse sphere

Path Tracing Results

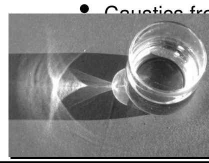


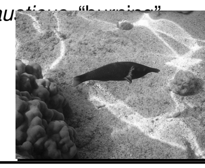
The spheres and base are the only colored objects.

Note the color bleeding and caustics.

Caustics

- <u>Caustics</u> occur when many light rays are reflected/refracted onto a single point
  - Anyone ever burn anything using a magnifying glass?
    - Caustics from Latin *cau*





Caustics

- <u>The</u> feature that clearly distinguishes real global illumination solutions
  - Need to trace an envelope of rays from a diffuse surface through/off a curved reflective/refractive surface
    - Ray tracers can't do it, because they can't follow all the rays from a diffuse surface
    - Radiosity can't do it, because it doesn't have reflection/refraction

Path Tracing Review

- Probabilistically follow <u>just one</u> path through the scene for each ray
  - Works well because the majority of the contribution is due to the first ray, anyway
- If you shoot enough rays, you get a reasonable approximation of global illumination

The New Hotness

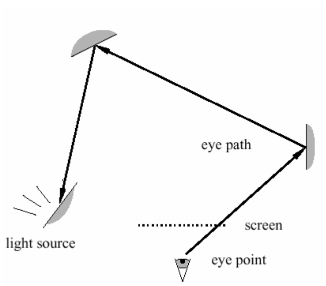
- Path tracing was, to my knowledge, the first reasonable real global illumination solution
- Some newer methods include
  - Bi-directional path tracing
  - Metropolis light transport
  - Photon mapping

Bi-Directional Path Tracing

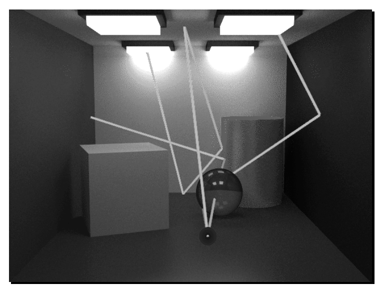
LaFortune and Veach

- Doesn't just trace paths from the eye
- Also traces paths from the light sources
  - Light tracing
- Can combine these paths if appropriate

Pure Path Tracing



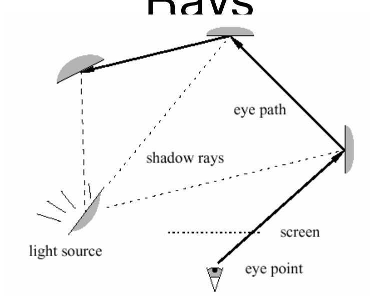
Path Tracing Results



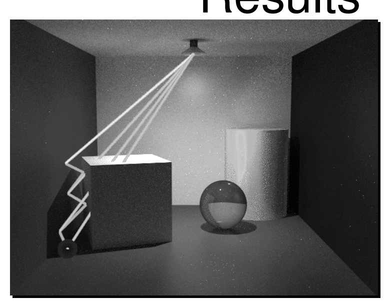
Best for big luminaires.

If lights small, few hits and large variance.

Path Tracing + Shadow Rays



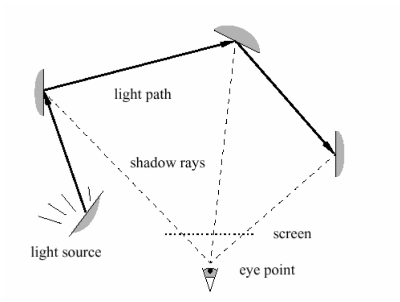
PT + Shadow Rays Results



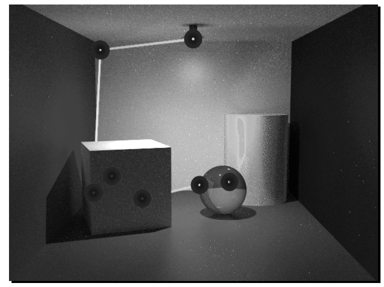
Small lights OK.

Best for specular surfaces.

Light Tracing



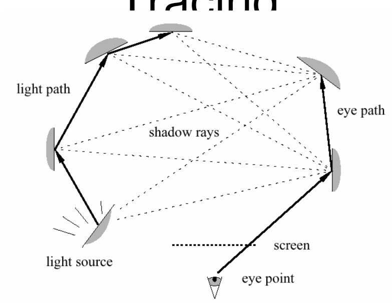
Light Tracing Results



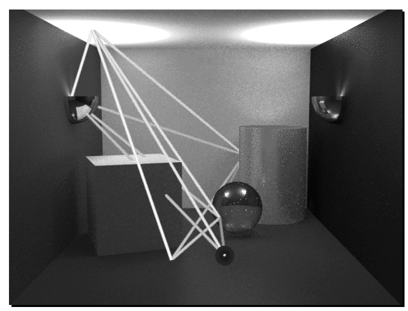
Small lights OK.

Best for caustics.

Bi-Directional Path Tracing



BDPT Results



BDPT Results

(a) Bidirectional path tracing with 25 samples per pixel

(b) Standard path tracing with 56 samples per pixel (the same computation time as (a))

Metropolis Light Transport

- Metropolis is a method for importance sampling paths
  - Instead of sampling paths randomly, identify a “good” path, and then sample paths that are slight perturbations from that path

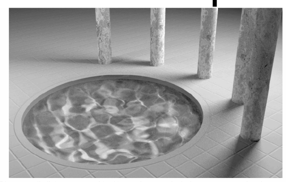
Metropolis Results

(b) Metropolis light transport with 250 mutations per pixel [the same computation time as (a)].

Only light in this scene comes through the crack in the doorway

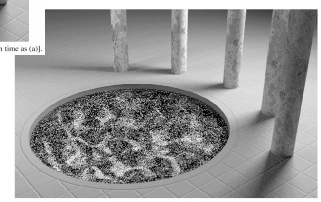
(a) Bidirectional path tracing with 40 samples per pixel.

Metropolis Results



(b) Metropolis light transport with 100 mutations per pixel [the same computation time as (a)].

There are specific mutations to capture caustics.



(a) Path tracing with 210 samples per pixel.

Metropolis Advantages

- Works well for “difficult” lighting conditions
  - Such as small lights, or lights which are difficult to reach
    - Think about the door example
  - Paths stay in the “important” area
- There is only a small amount of work required to generate a mutated path

Photon Mapping

Henrik Wann Jensen

- This is a two pass algorithm:
  - The photon mapping pass traces “photons” along rays from the light, and distributes them in the environment
    - The illumination data is stored in a photon map
  - The rendering pass traces rays from the eye, and reads back the illumination from the photon map to create the image

Photon Map

- A spatial data structure that stores illumination data (how many “photons” landed here) at points
  - A 3D kd-tree
- Each point stores location, power, incident direction
- Structure is filled during photon mapping pass
- Jensen uses global { L(S | D)*D }, caustic {LS+D}, and volume photon maps

Photon Mapping Pass

- Each “photon” represents a fraction of the power of a light
- These get traced through the scene from the lights
  - Just as in raytracing
- When a photon hits an object, it is probabilistically reflected, transmitted, or absorbed
- When a photon hits a diffuse surface, it is stored in the map, or it can be reflected

Using the Photon Map

- *k* nearest photons are filtered together to get an estimate of radiance at a point
  - Uses a sphere or disk to find the nearest photons
    - Note that this can result in “false” photons across the edges of a surface

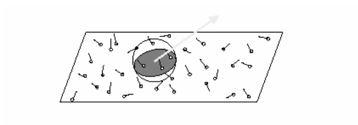


Figure 2.8: Radiance is estimated using the nearest photons in the photon map.

Rendering Pass

- The contributions to each pixel are divided into 4 components
  - Direct lighting
  - Specular and glossy reflections
  - Caustics
  - Diffuse interactions
- Uses approximate solutions after several bounces

Direct Lighting

- Trace rays to lights
  - Just like in distributed ray tracing
- Also check shadows with “shadow photons”
  - Photons with “negative” p…

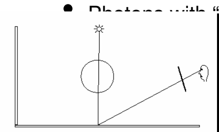


Figure 2.12: Accurate evaluation of the direct illumination.

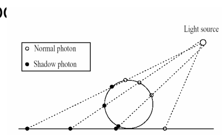


Figure 1: Creating shadow photons

Specular and Glossy

- Standard recursive ray tracing

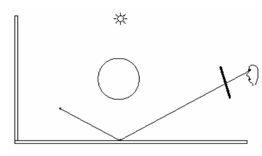


Figure 2.13: Rendering specular and glossy reflections.

Caustics

- Look up in the caustics photon map

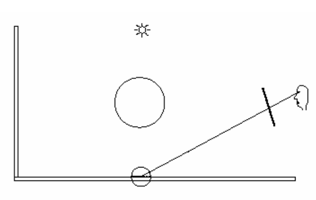


Figure 2.14: Rendering caustics.

Diffuse Interactions

- Each ray takes one bounce and estimates irradiance

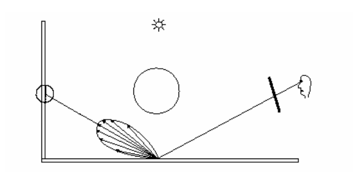


Figure 2.15: Computing indirect diffuse illumination with importance sampling.

- Can also use irradiance cache

Photon Mapping Results

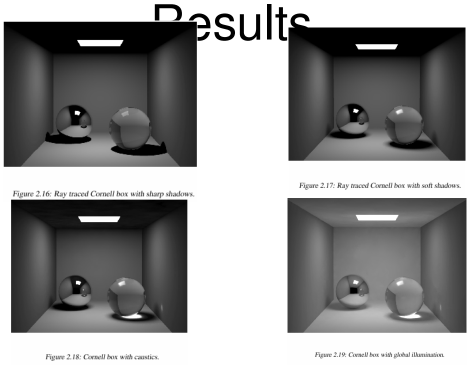


Figure 2.16: Ray traced Cornell box with sharp shadows.

Figure 2.17: Ray traced Cornell box with soft shadows.

Figure 2.18: Cornell box with caustics.

Figure 2.19: Cornell box with global illumination.

Photon Mapping Results