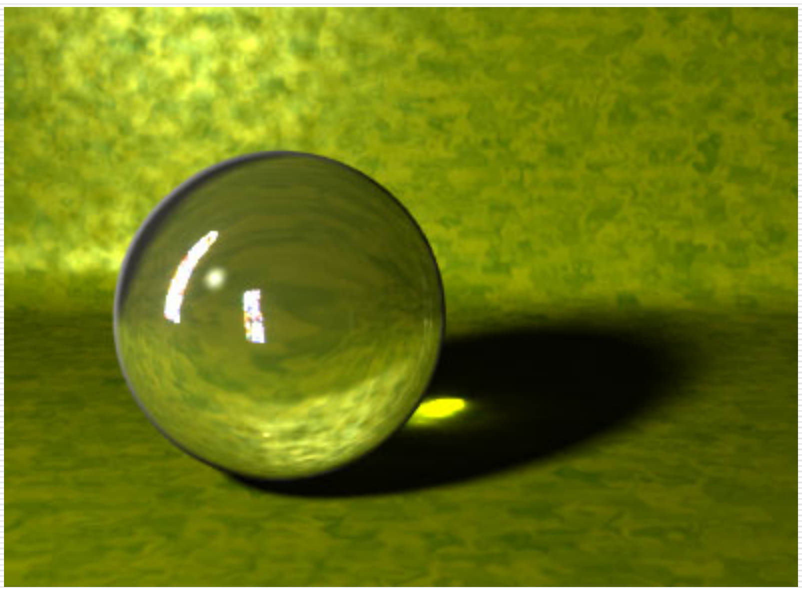
Computer Graphics
Global Illumination: Monte-Carlo Ray Tracing and Photon Mapping



Lecture 14

Taku Komura

In the last lecture

- We did ray tracing and radiosity
- Ray tracing is good to render specular objects but cannot handle indirect diffuse reflections well
- Radiosity can render indirect diffuse reflections but not specular reflections
- Both can be combined to synthesize photo-realistic images

Problems

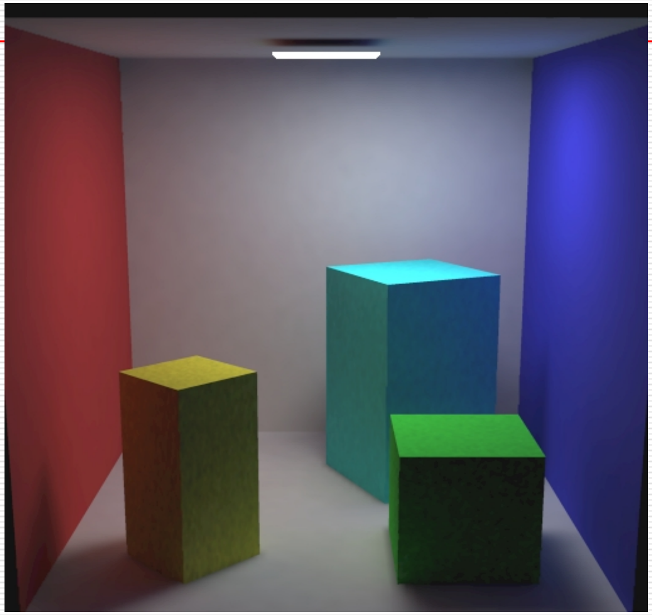
- Radiosity is very slow
  - Calculating the form factor
  - Solving a dense linear system
  - Also problems with parallelisation
- Raytracing cannot handle caustics well

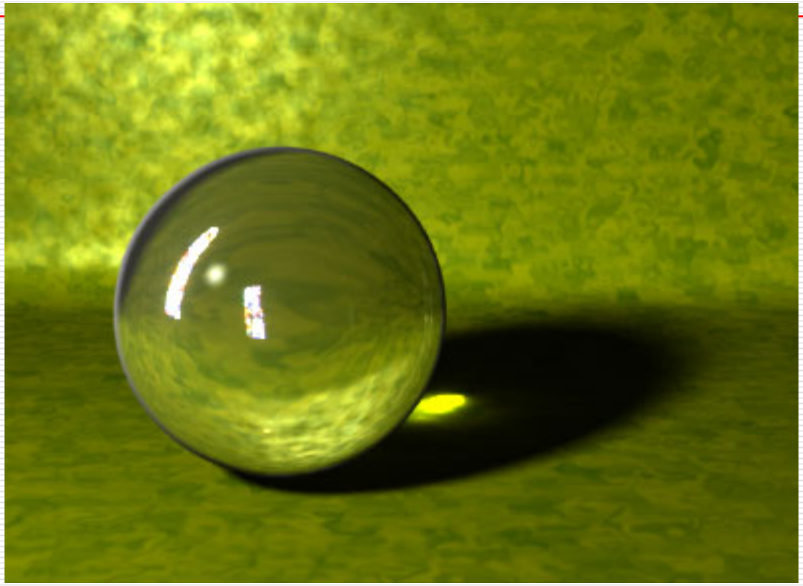
Today

- Other practical methods to synthesize photo-realistic images
- Monte-Carlo Ray Tracing
  - Path Tracing
  - Bidirectional Path Tracing
- Photon Mapping

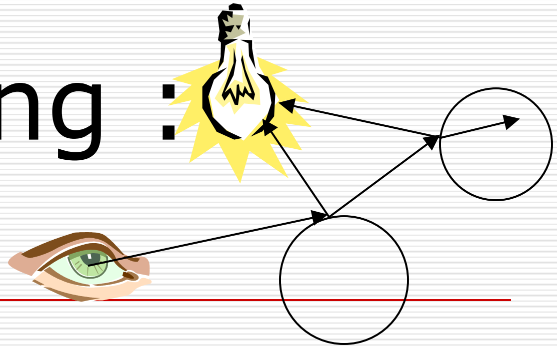
Today : Global Illumination Methods

- Monte-Carlo Ray Tracing
- Photon Mapping





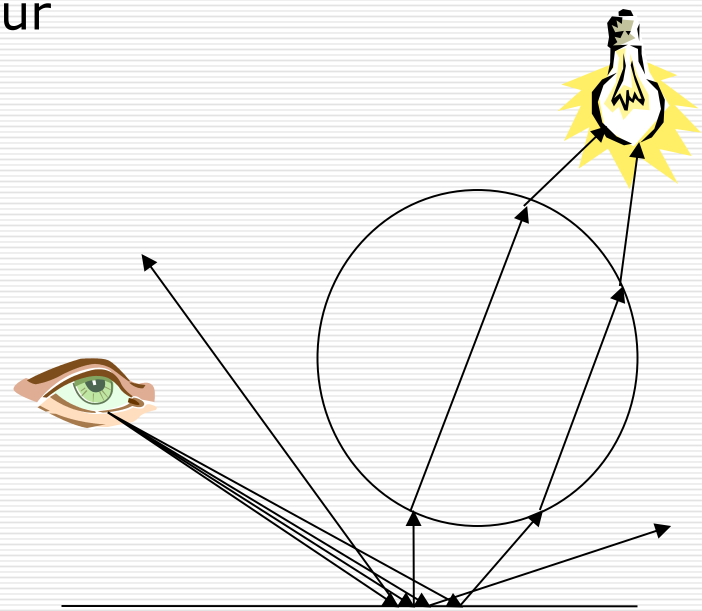
Monte-Carlo Ray Tracing : Path Tracing



- An enhancement of the ordinary ray-tracing scheme
- But when hitting a diffuse surface, pick one ray at random, and find the colour of the incoming light

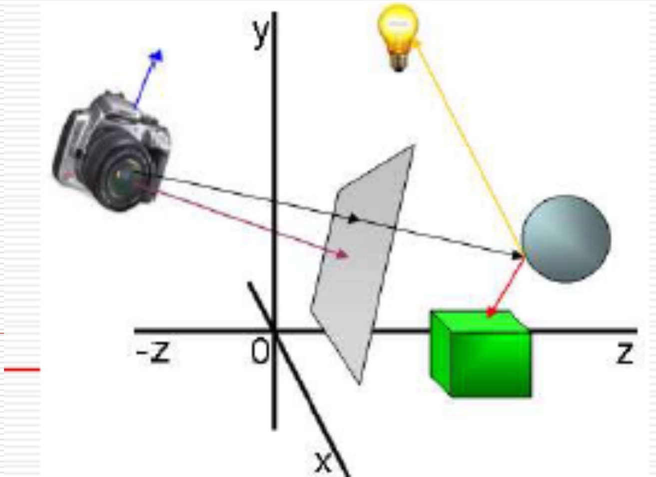
- **Trace many paths per pixel**

  (100-10000 per pixel)
- Only trace one ray per intersection
- by Kajiya, SIGGRAPH 86



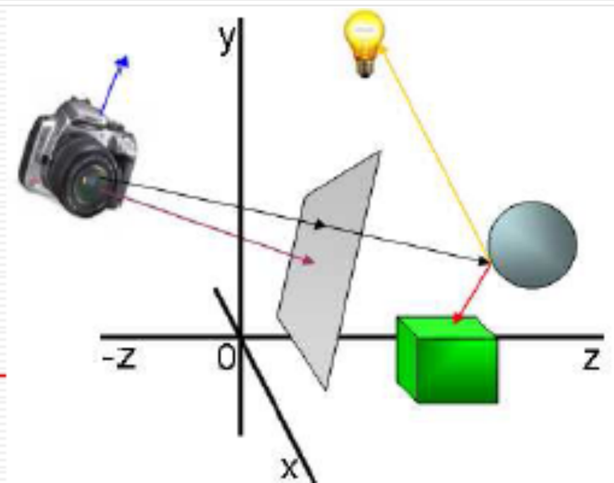
Ray Tracing Algorithm

- Trace (ray)
  - Find the intersection of the ray and the scene
  - Computer the shadow ray : C=Ca
  - Do the local illumination : C+=Cl (not shadowed)
  - If specular compute the reflection vector R
    - C+=Trace(R)
  - If refractive compute the refractive vector T
    - C+=Trace(T)



Path Tracing Algorithm

- Trace (ray)
  - Find the intersection of the ray and the scene
  - Computer the shadow ray : C=Ca
  - Do the local illumination : C+=Cl (not shadowed)
  - If specular compute the reflection vector R
    - C+=Trace(R)
  - If refractive compute the refractive vector T
    - C+=Trace(T)
  - **Else if diffuse compute a random vector R'**
    - **C+=Trace(R')**



Shadow ray towards the light at each vertex

- As we are following the ordinary path tracing scheme, we cast a shadow ray at every intersection of a ray and surface

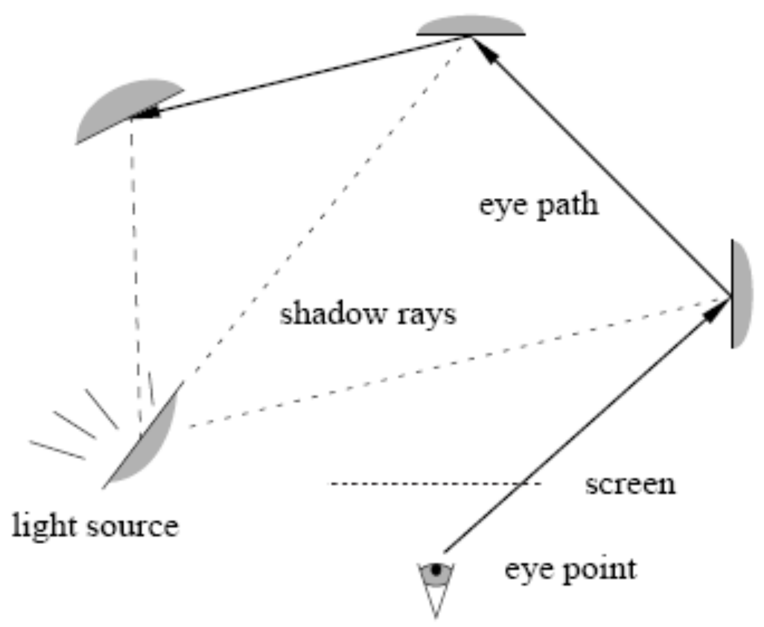


Figure 4.7: Schematic overview of the path tracing algorithm with next event estimation. Direct illumination is now computed explicitly at each point on the random walk by sampling the light sources.

Path Tracing : algorithm

Render image using path tracing

```
for each pixel
    color = 0
    For each sample
        pick ray in pixel
        color = color + trace(ray)
    pixel_color = color/#samples
```

trace(ray)

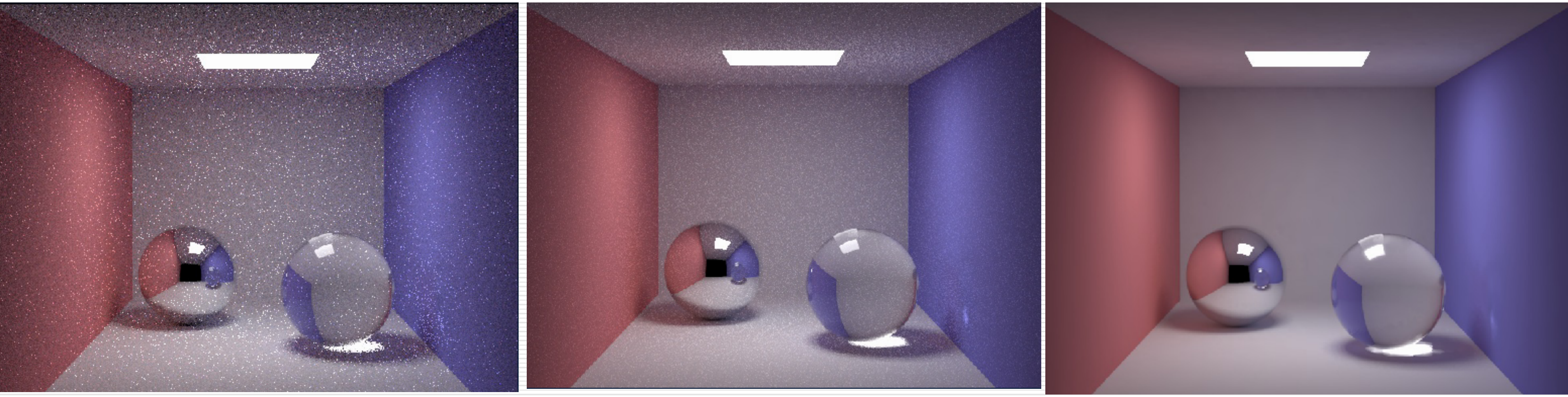
```
find nearest intersection with scene
compute intersection point and normal
color = shade (point, normal)
return color
```

Shade (point, normal)

```
color = 0
for each light source
    test visibility on light source
    if visible
        color=color+direct illumination
if diffuse surface
    color = color + trace ( a randomly reflected ray)
else if specular
    color = color + trace (reflection ray)
else if refraction
    color = color + trace (refraction ray)
return color
```

Path tracing : problems

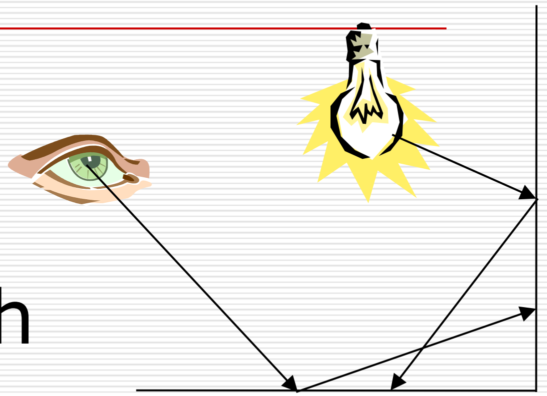
- Very accurate but vulnerable to noise – need many samples
  - Using too few paths per pixel result in noise
  - Requires 1000 ~ 10000 samples per pixel

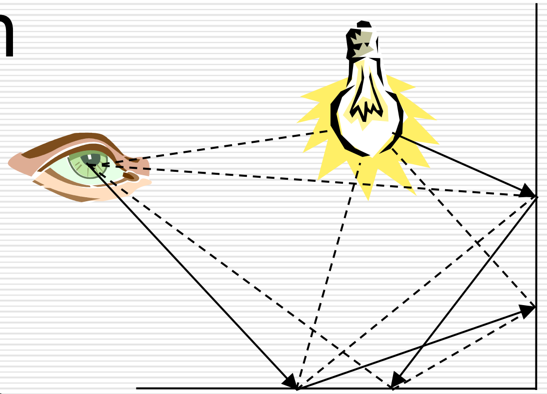


10 paths / pixel   100 paths / pixel   1000 paths / pixel

Bidirectional Path Tracing

- Send paths from light source, record path vertices
- Send paths from eye, record path vertices
- Connect every vertex of eye path with every vertex in light path
- Assume the vertex of the light paths are light sources
- Lafortune & Willems, Compugraphics '93, Veach & Guibas, EGRW 94





Computing the pixel color

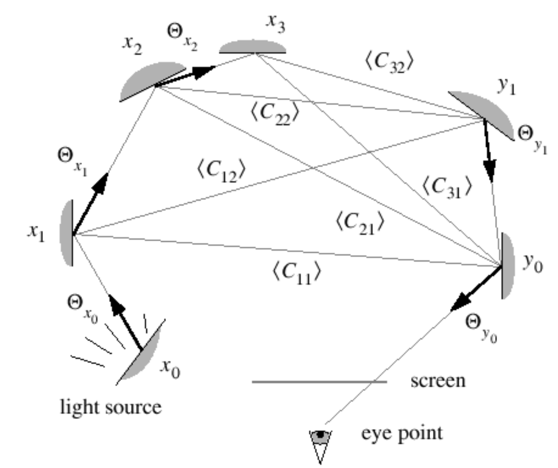


Figure 9: The contribution $\langle C_{ij} \rangle$ is an estimate for the flux that reaches the eye through both the light path and the eye path.

$$L_{i,j}(x_i \to x_{i-1}) =$$

$$f_r(y_j \to x_i \to x_{i-1}) V(x_i, y_j) \frac{\left|(y_j \to x_i) \bullet \vec{n}_{xi}\right|}{\left\|x_i - y_j\right\|^2} I(y_j \to x_i)$$

Pixel colour computed by summing the contributions

from all paths : $L_p = \sum_{i=0}^{Ni} \sum_{j=0}^{Nj} w_{i,j} L_{i,j}$

How to compute the weights?

$$w_{i,j} = \frac{p_{i,j}^2}{\sum_{k=0}^{i+j} p_{k,i+j-k}^2}$$

$p_{i,j}$ is the probability of producing path *0,1,..,i,j,...i + j*

where path from the camera is 0,1,...,i

and path from the light source is j,....,i + j

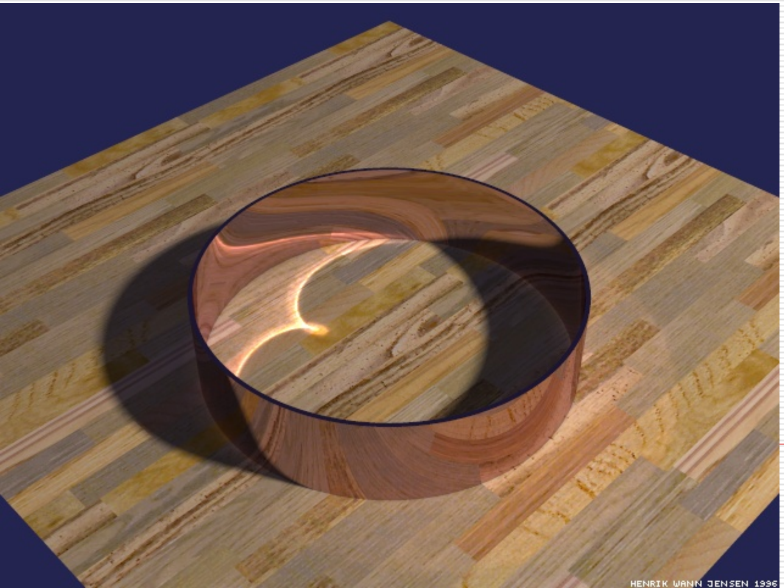
$$p(x_i \to x_{i+1}) = p_{fr}(x_i \to x_{i+1}) \frac{\left|(x_i \to x_{i+1}) \bullet \vec{n}_{xi}\right| \left|(x_i \to x_{i+1}) \bullet \vec{n}_{xi+1}\right|}{\left\| x_i - x_{i+1} \right\|^2}$$

Comparison

(a) Bidirectional path tracing with 25 samples per pixel

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

What about the scenes below?



In what case it works better than path tracing?

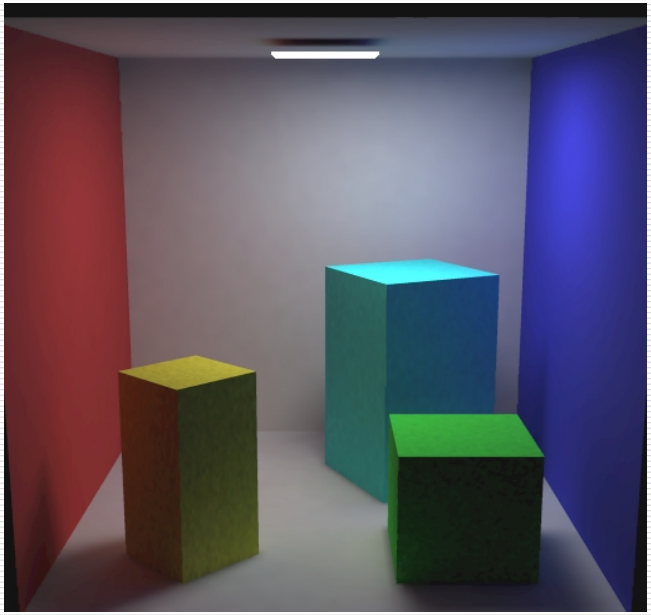
- Caustics
  - Easier to produce by tracing from the light source
- Indoor scenes where indirect lighting is important
  - Bidirectional methods take into account the inter-reflections at diffuse surfaces
- When the light sources are not easy to reach from the eye
  - less shadow rays reach the light source

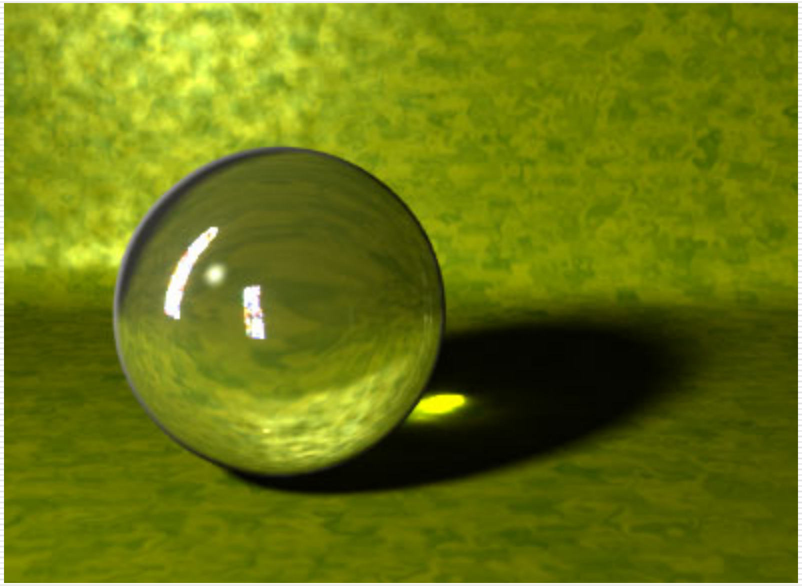
Summary for Monte Carlo Ray tracing

- Can simulate caustics
- Can simulate bleeding
- Results are subject to variance
  - Requires a lot of samples per pixel to reduce the noise
  - The noise decrease only at the order of 1/sqrt(N) where N is the number of samples

Today : Global Illumination Methods

- Monte-Carlo Ray Tracing
- Photon Mapping





Photon Mapping

- A fast, global illumination algorithm based on Monte-Carlo method

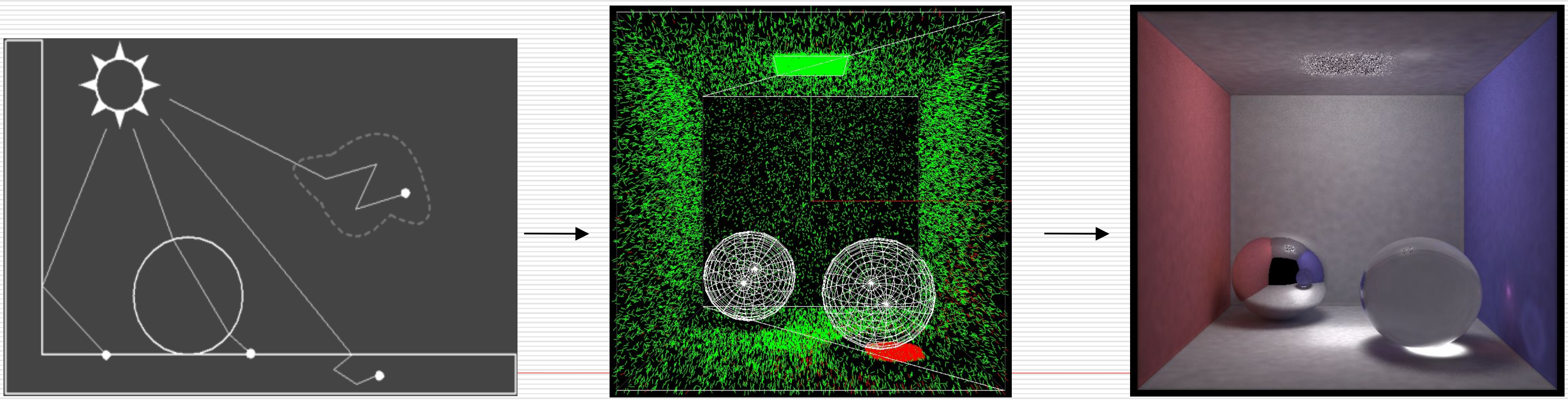
1. Casting photons from the light source, and
2. saving the information of reflection in the "photon map", then
3. render the results

- A stochastic approach that estimates the radiance from limited number of samples

http://www.cc.gatech.edu/~phlosoft/photon/

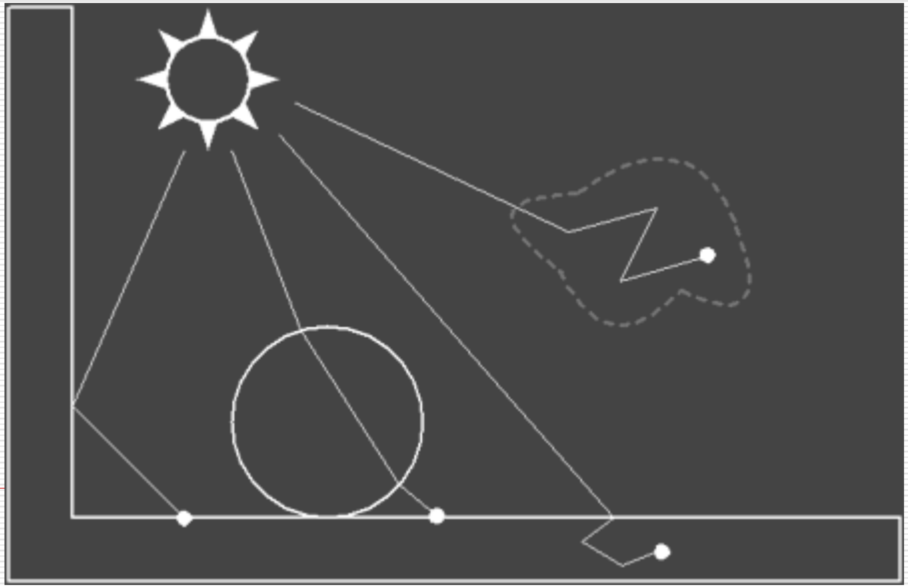
Photon Mapping

- A two pass global illumination algorithm
  - First Pass - Photon Tracing
  - Second Pass - Rendering



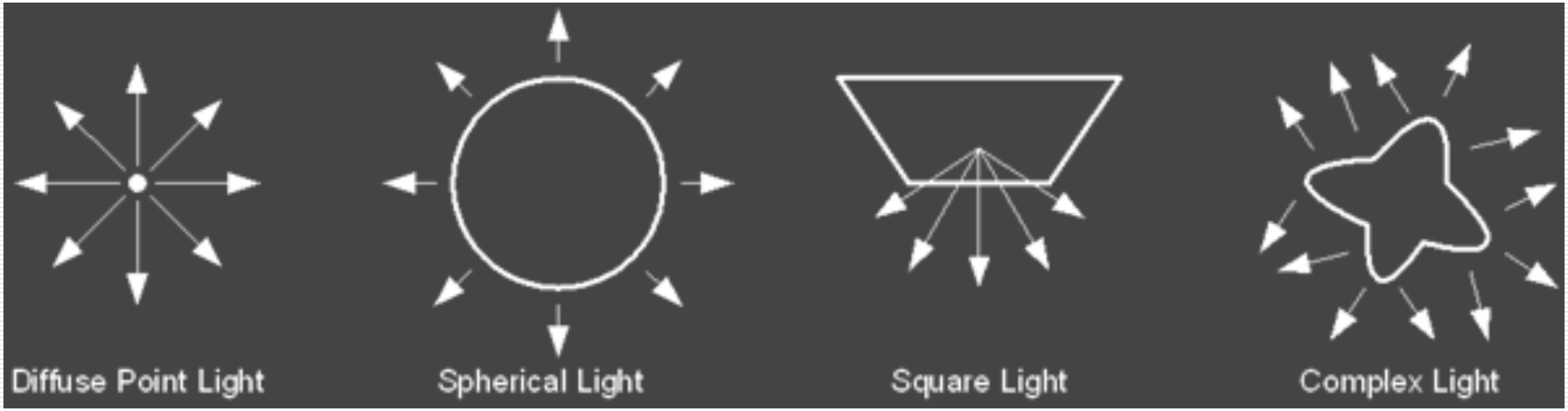
Photon Tracing

- The process of emitting discrete photons from the light sources and
- tracing them through the scene



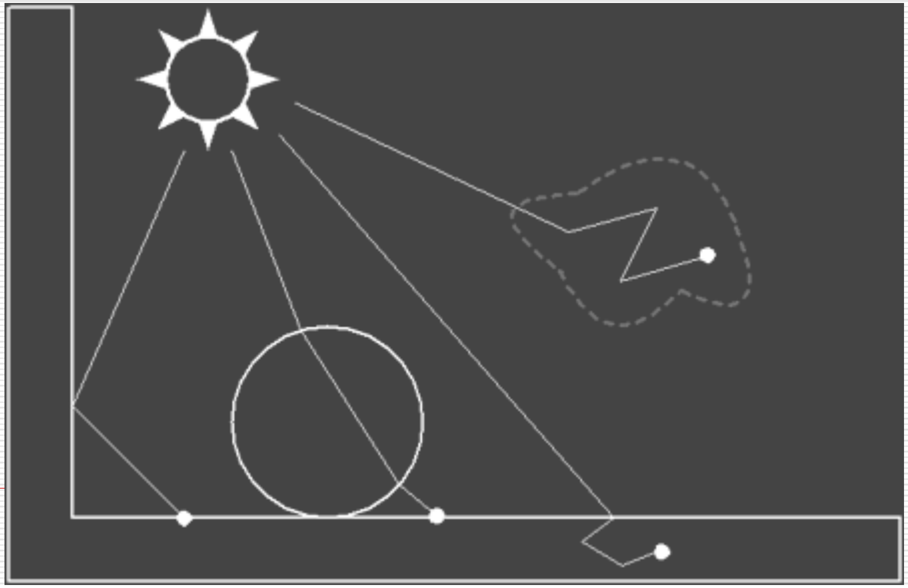
Photon Emission

- A photon's life begins at the light source.
- Different types of light sources
- Brighter lights emit more photons



Photon Scattering

- Emitted photons are scattered through a scene and are eventually absorbed or lost
- When a photon hits a surface we can decide how much of its energy is absorbed, reflected and refracted based on the surface’s material properties



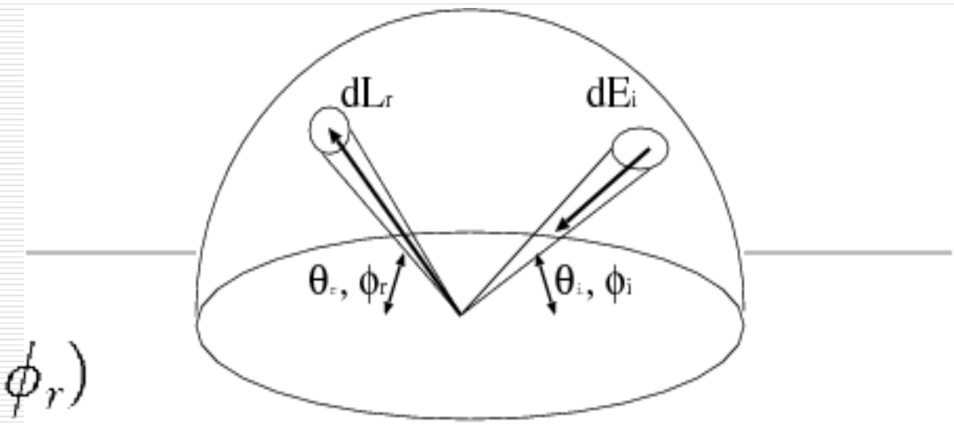
What to do when the photons hit surfaces

- Attenuate the power and reflect the photon
  - For arbitrary BRDFs
- Use Russian Roulette techniques
  - Decide whether the photon is reflected or not based on the probability

Bidirectional Reflectance Distribution Function (BRDF)

- The reflectance of an object can be represented by a function of the incident and reflected angles

- This function is called the Bidirectional Reflectance Distribution Function (BRDF)

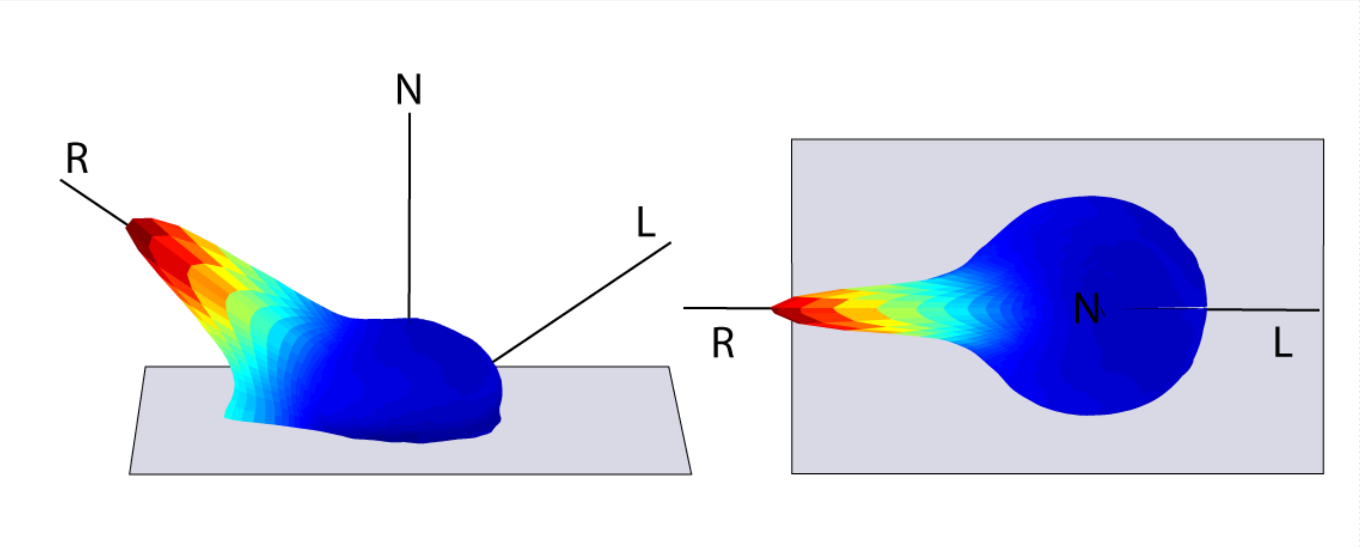


$$\rho(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{dL_r(\theta_r, \phi_r)}{dE_i(\theta_i, \phi_i)},$$

- where *E* is the incoming **irradiance** and *L* is the reflected **radiance**

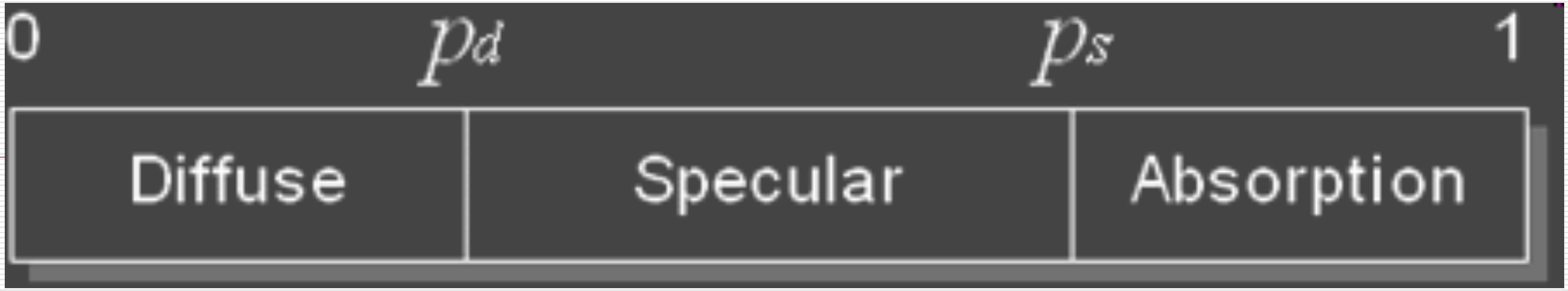
Arbitrary BRDF reflection

- Can randomly calculate a direction and scale the power by the BRDF



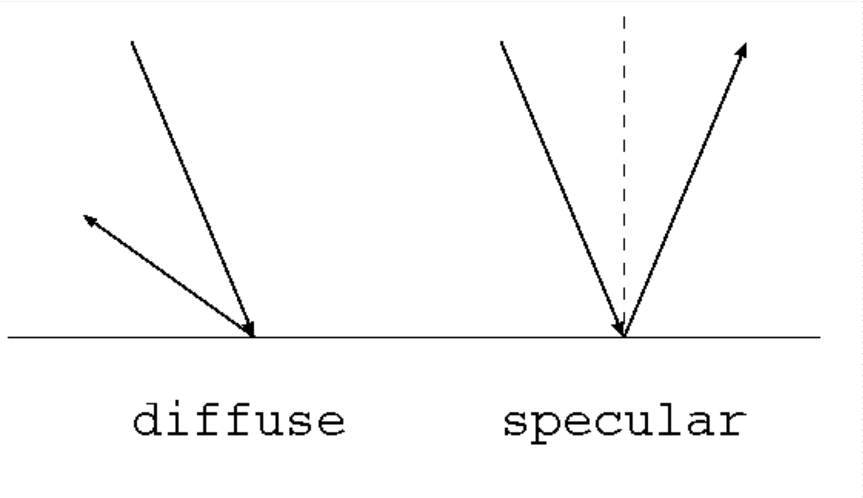
Russian Roulette

- If the surface is diffusive+specular, a Monte Carlo technique called Russian Roulette is used to probabilistically decide whether photons are reflected, refracted or absorbed.
- Produce a random number between 0 and 1
- Determine whether to transmit, absorb or reflect in a specular or diffusive manner, according to the value



Diffuse and specular reflection

- If the photon is to make a diffuse reflection, randomly determine the direction
- If the photon is to make a specular reflection, reflect in the mirror direction



Probability of diffuse and specular reflection, and absorption

- Probability of reflection can be the maximum energy in any colour band

$$P_r = max(d_r + s_r, d_g + s_g, d_b + s_b)$$

- The probability of diffuse reflection is

$$P_d = \frac{d_r + d_g + d_b}{d_r + d_g + d_b + s_r + s_g + s_b} P_r \, .$$

- Similarly, the probability of specular reflection is

$$P_s = \frac{s_r + s_g + s_b}{d_r + d_g + d_b + s_r + s_g + s_b} P_r = P_r - P_d \, .$$

Power Attenuation

- The colour of the light must change after specular / diffuse reflection
- Colour bleeding

Power after reflectance

- The power P_{ref} of the reflected photon is:

$$P_{ref,sr} = P_{inc,r}\ s_r / P_s$$

$$P_{ref,sg} = P_{inc,g}\ s_g / P_s$$

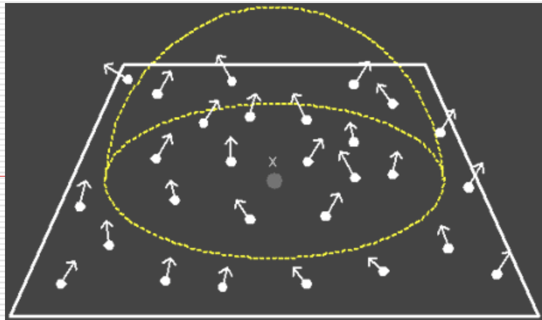
$$P_{ref,sb} = P_{inc,b}\ s_b / P_s$$

where P_{inc} is the power of the incident photon.

The above equation is for specular reflection, but so the same for diffusive reflection

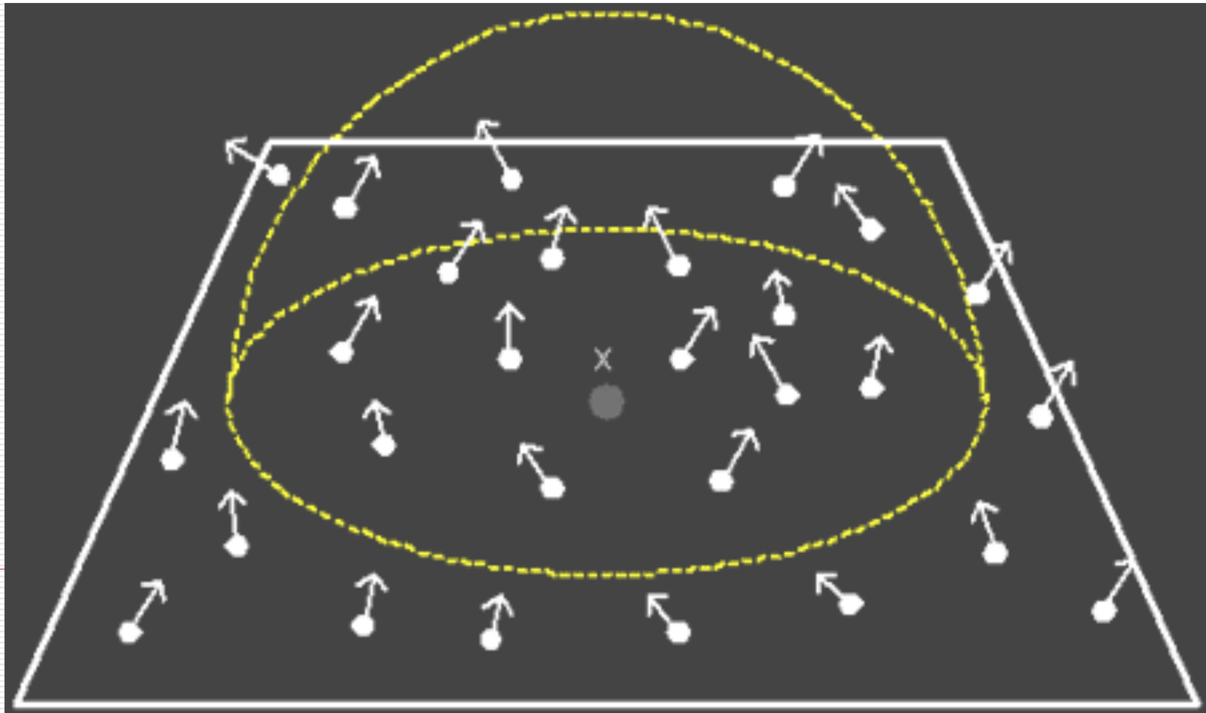
Photon Map

- When a photon makes a diffuse bounce, the ray intersection is stored in memory
  - 3D coordinates on the surface
  - Colour intensity
  - Incident direction
- The data structure of all the photons is called Photon Map
- The photon data is not recorded for specular reflections



Second Pass – Rendering

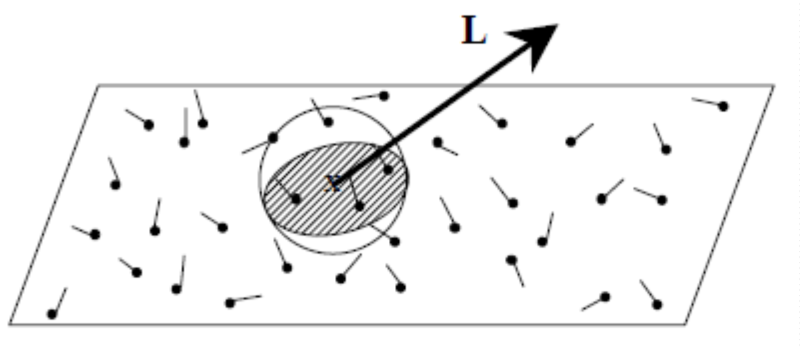
- Finally, a traditional ray tracing procedure is performed by shooting rays from the camera
- At the location the ray hits the scene, a sphere is created and enlarged until it includes N photons



Radiance Estimation

- The radiance estimate can be written by the following equation

$$L_r(x,\vec{\omega}) = \sum_{p=1}^{N} f_r(x,\overrightarrow{\omega_p},\vec{\omega}) \frac{\Delta\Phi_p(x,\overrightarrow{\omega_p})}{\Delta A}$$



x: location the ray hits the scene

$\vec{\omega}$: direction towards the camera

$\overrightarrow{\omega_p}$: incident vector of photon p

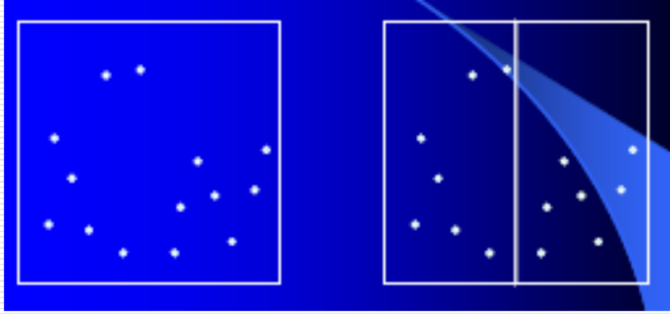
f_r: BRDF

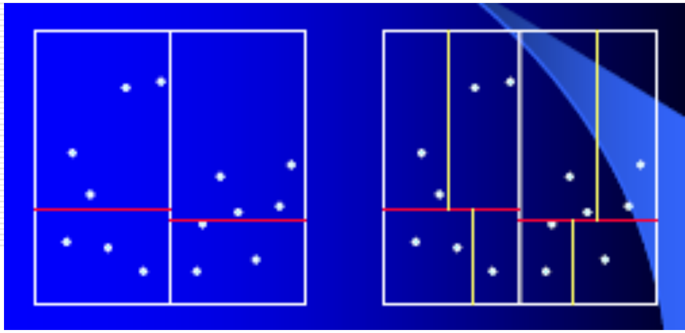
N: the number of photons

$\Delta\Phi_p$: power of photon p

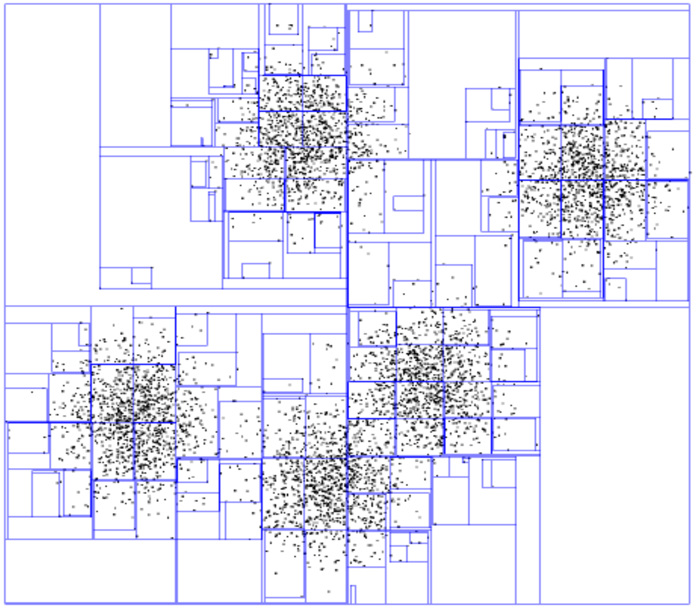
ΔA: Area of the circle πr^2

Saving photons: KD tree



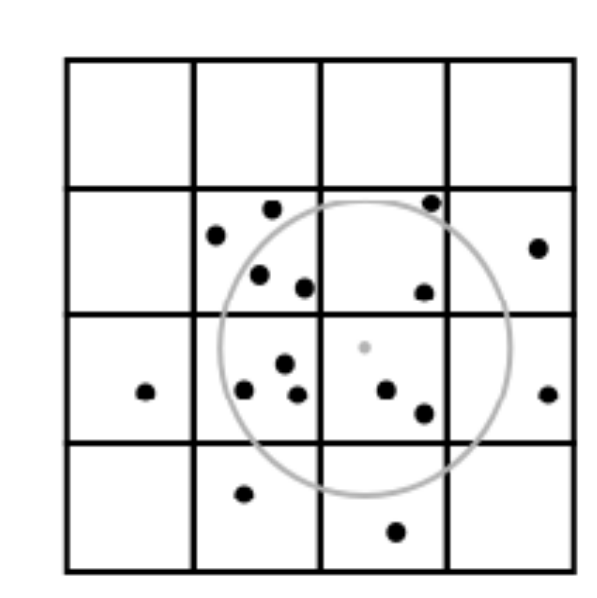


- The photon maps are classified and saved in a KD-tree
- KD-tree :
  - dividing the samples at the median
  - The median sample becomes the parent node, and the larger data set form a right child tree, the smaller data set form a left child tree
  - Further subdivide the children trees
- Can efficiently find the neighbours when rendering the scene



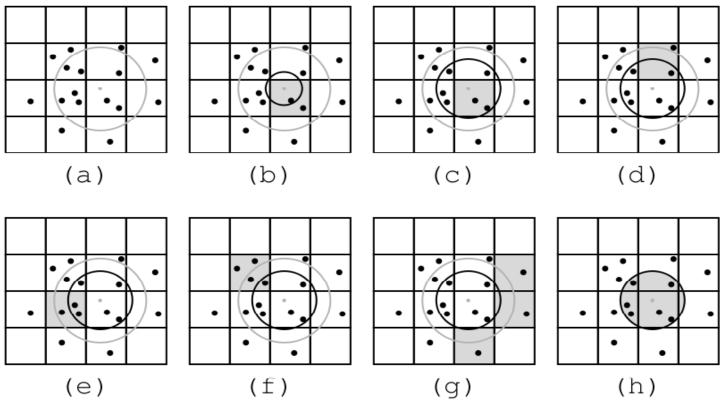
Saving photons: Spatial Hashing

- Produce a 3D grid
- Create a hash function that maps each grid to a list that saves the photons
- Scan the photons in the list to find those close to the sample point



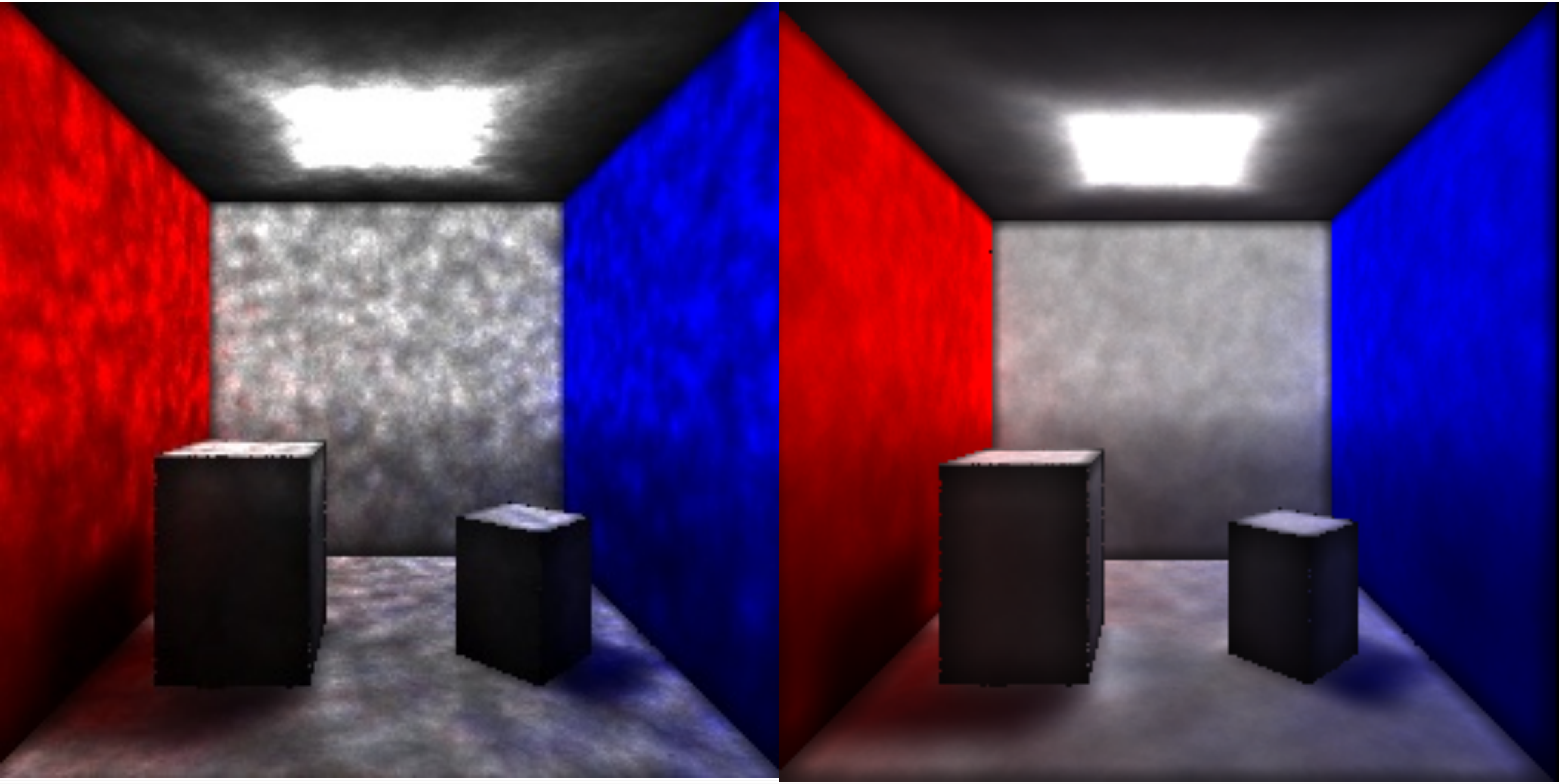
NN-search in the grids

- Decide the maximum radius of search
- Examine the distance between the sample point and the photons in the grid
- Gradually increase the radius, search in all the reachable grids until all the photons are found
- Suitable for hardware implementation
- "Photon Mapping on Programmable Graphics Hardware", Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware, pp. 41-50, 2003



Precision

- The precision of the final results depends on
  - the number of photons emitted
  - the number of photons counted for calculating the radiance



- By 10000 photons and 50 samples(left), and 500000 photons and 500 samples (right)

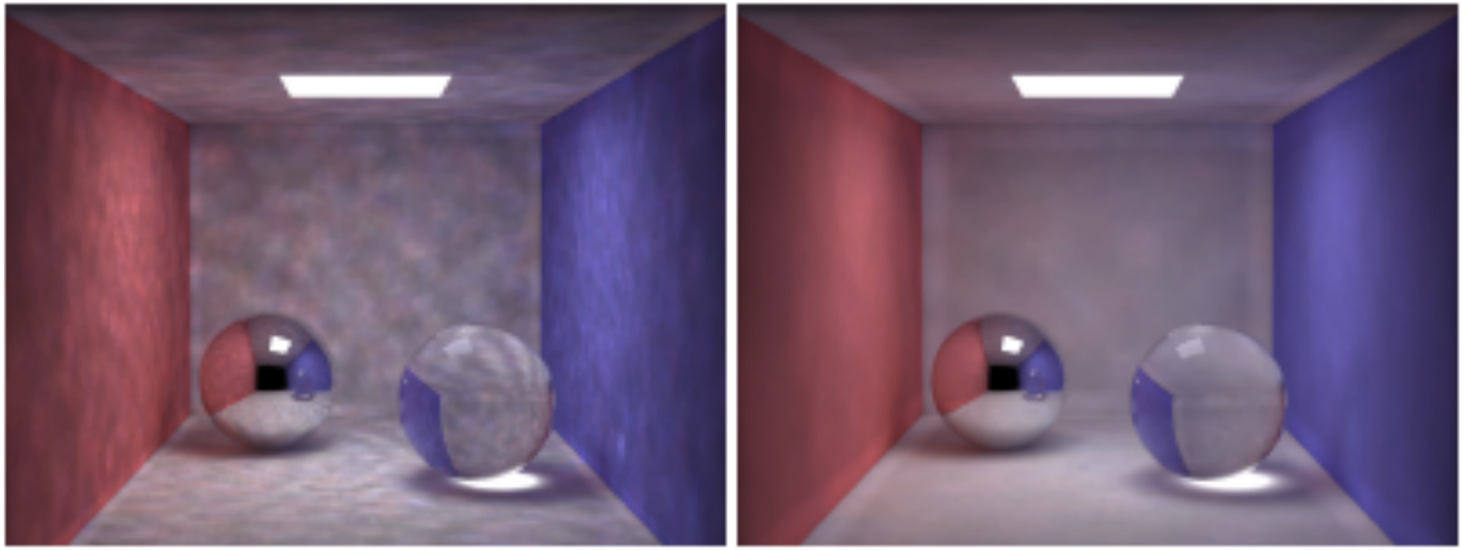
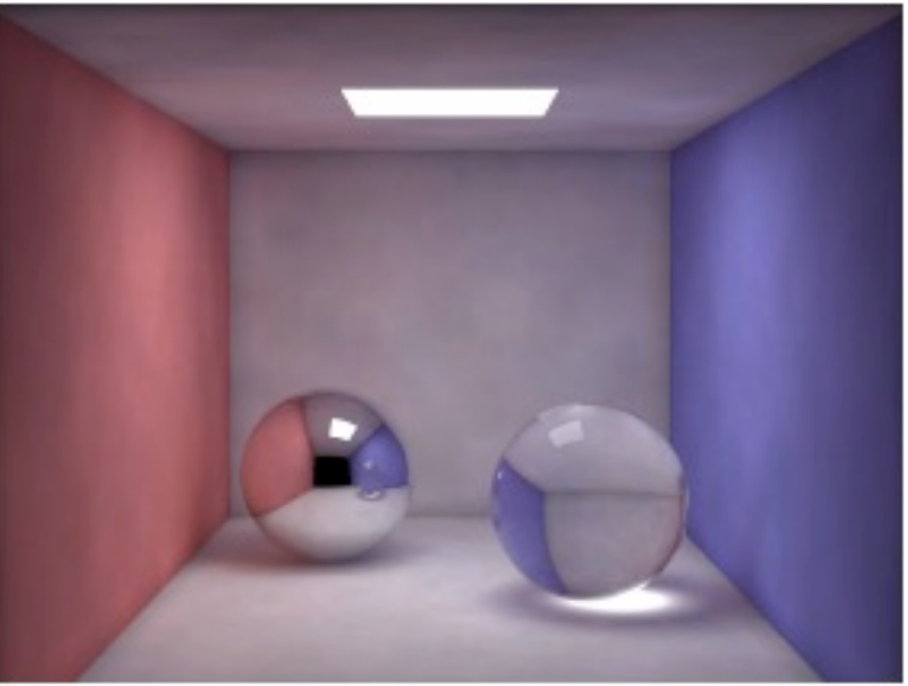


Figure 20: Global photon map radiance estimates visualized directly using 100 photons (left) and 500 photons (right) in the radiance estimate.



http://www.youtube.com/watch?v=wqWRVcsIcAQ

Figure 21: Global photon map radiance estimates visualized directly using 500 photons and a disc to locate the photons. Notice the reduced false color bleeding at the edges.

Summary

- Photon Mapping
  - A stochastic approach that estimates the radiance from a limited number of photons
  - Requires less computation comparing to path tracing

Readings

- Realistic Image Synthesis Using Photon Mapping by Henrik Wann Jensen, AK Peters
- Global Illumination using Photon Maps (EGRW '96) Henrik Wann Jensen
- Caustics Generation by using Photon Mapping, Presentation by Michael Kaiser and Christian Finger
- A Practical Guide to Global Illumination using Photon Maps
  - Siggraph 2000 Course 8
  - http://graphics.stanford.edu/courses/cs348b-01/course8.pdf