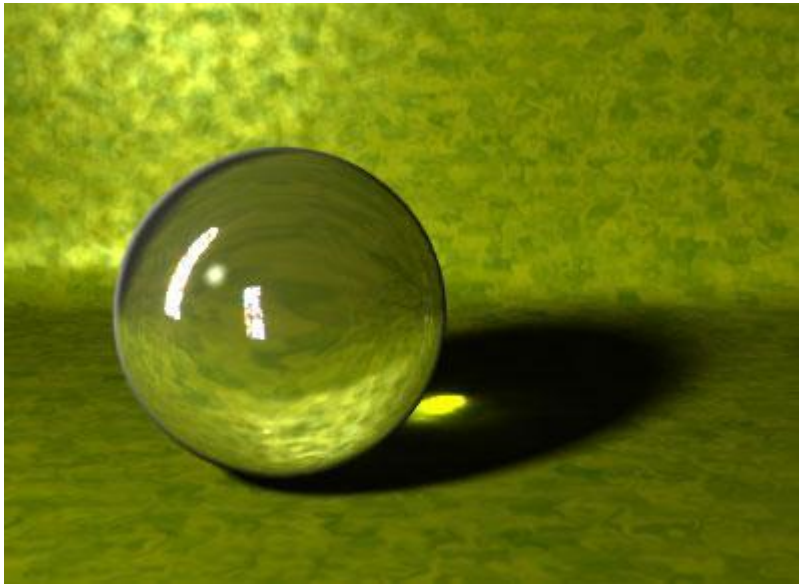
Computer Graphics
Global Illumination (2): Monte-Carlo Ray Tracing and Photon Mapping

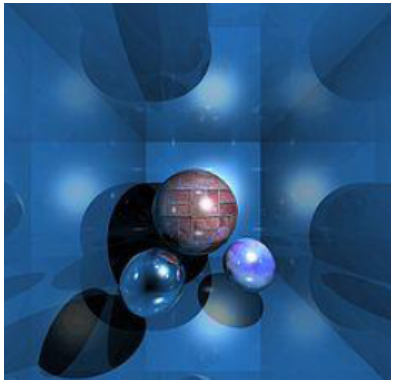


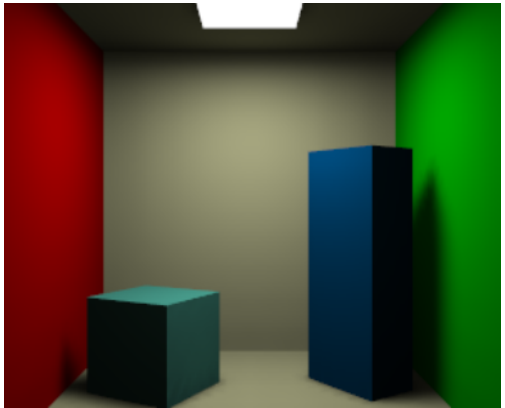
Lecture 15

Taku Komura

In the previous lectures

- 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
- They have to be combined to synthesize photo-realistic images



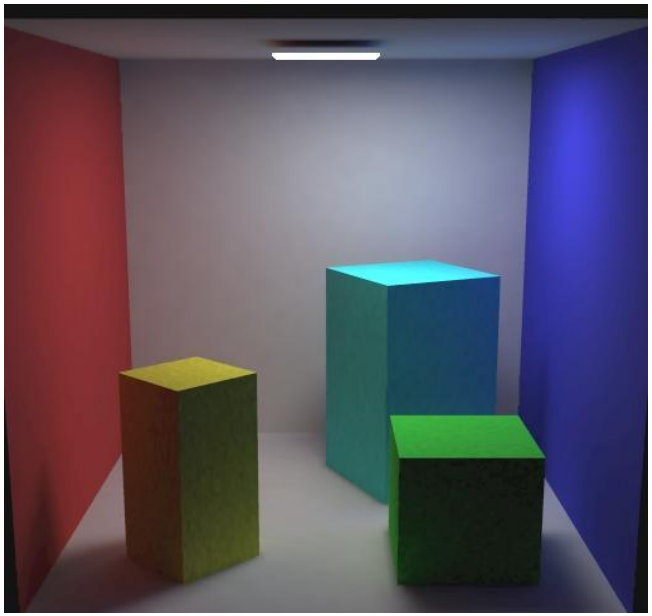


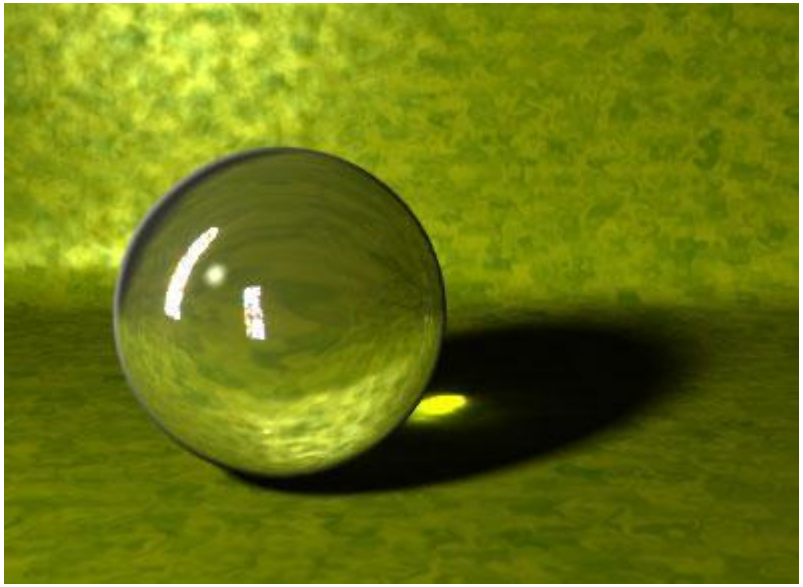
Today

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

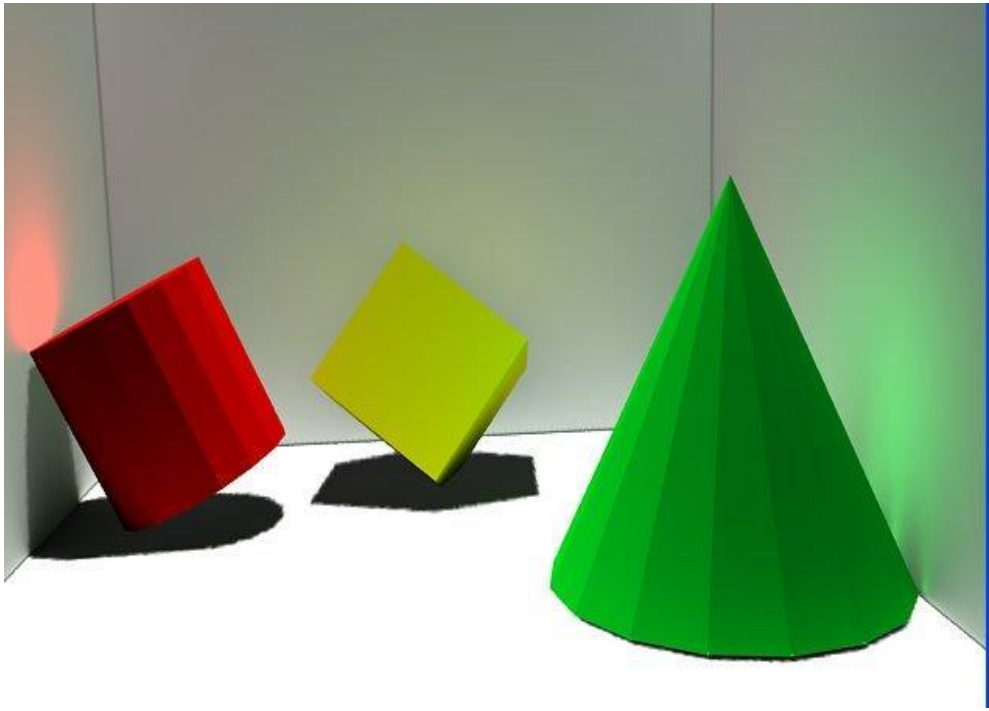
Overview

- **Light Transport Notations**
- Monte-Carlo Ray Tracing
- Photon Mapping



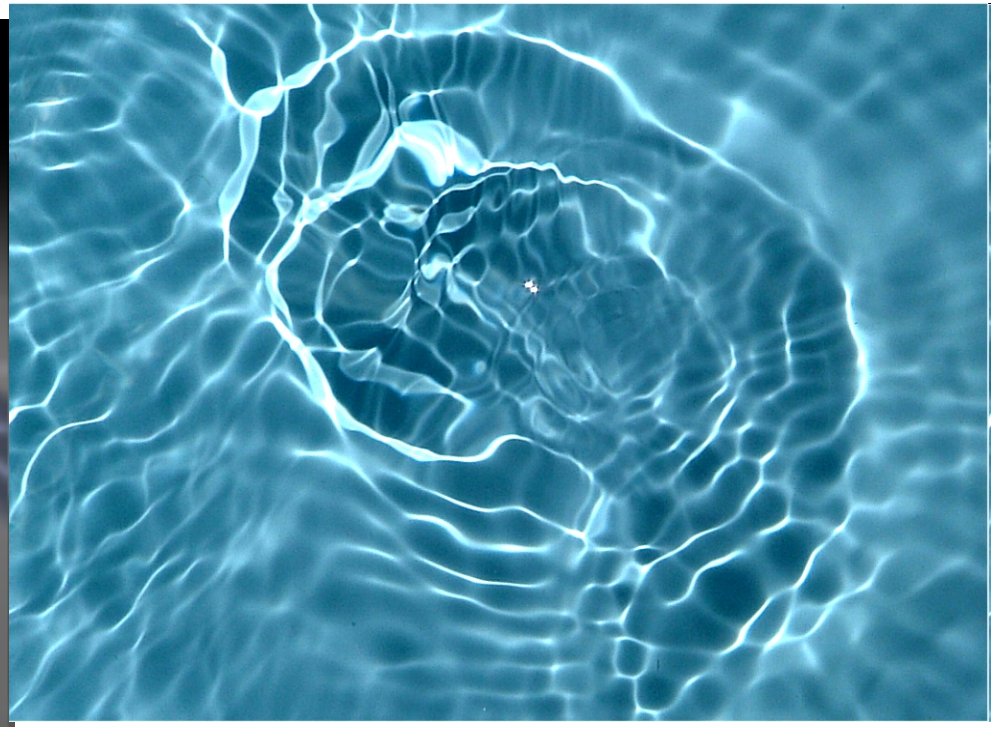


Color Bleeding



*

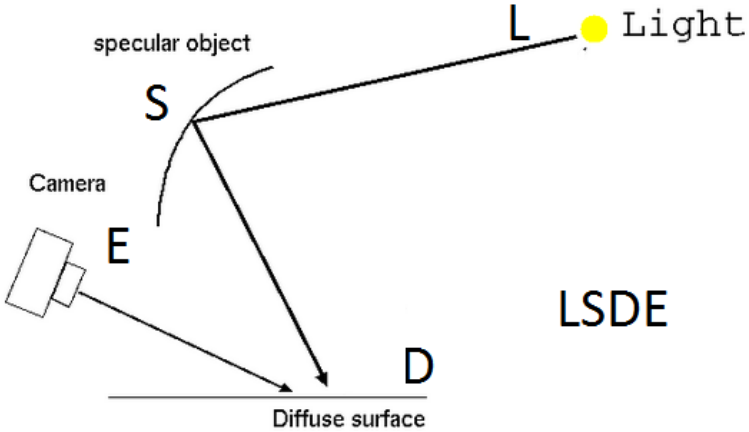
Caustics



Light Transport Notations

When describing a light path, it is sometimes necessary to distinguish the types of reflections

- L: a light source
- E: the eye
- S: a specular reflection or refraction
- D: a diffuse reflection



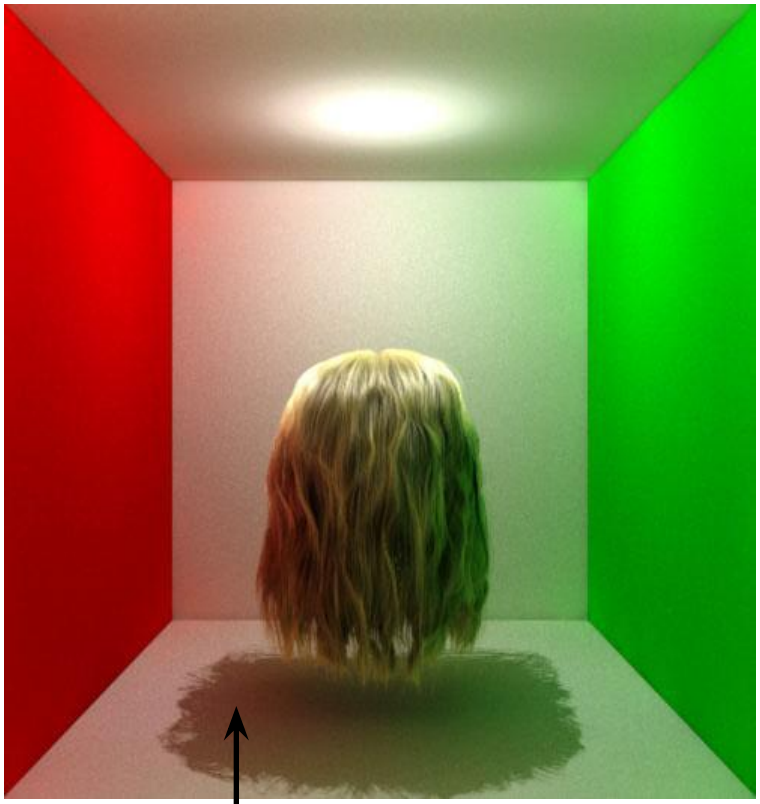
LSDE

Light Transport Notations (2)

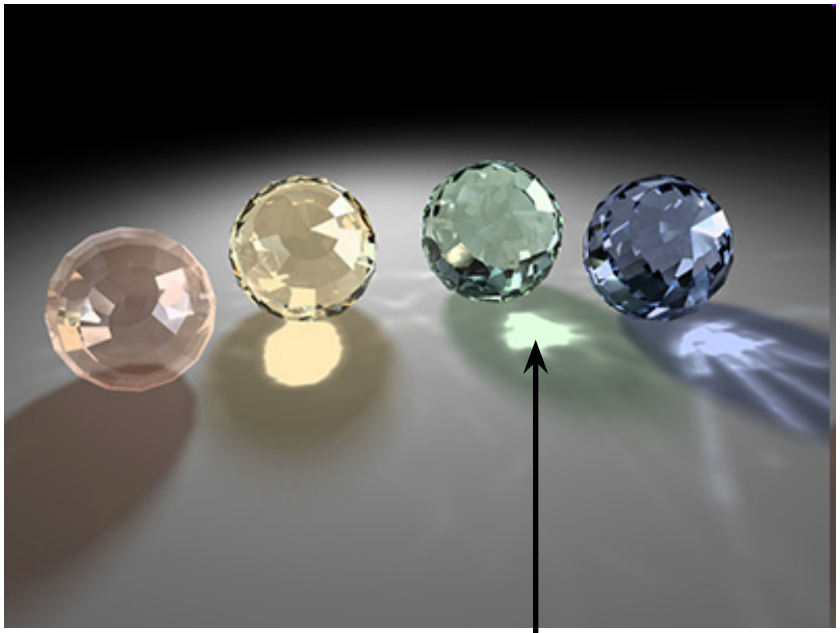
We may also use regular expressions:

- (k)+ : one or more of k events
- (k)* : zero or more of k events
- (k)?: zero or one of k events
- (k|k’) : k or k’

Light Transport Notations



LDDE

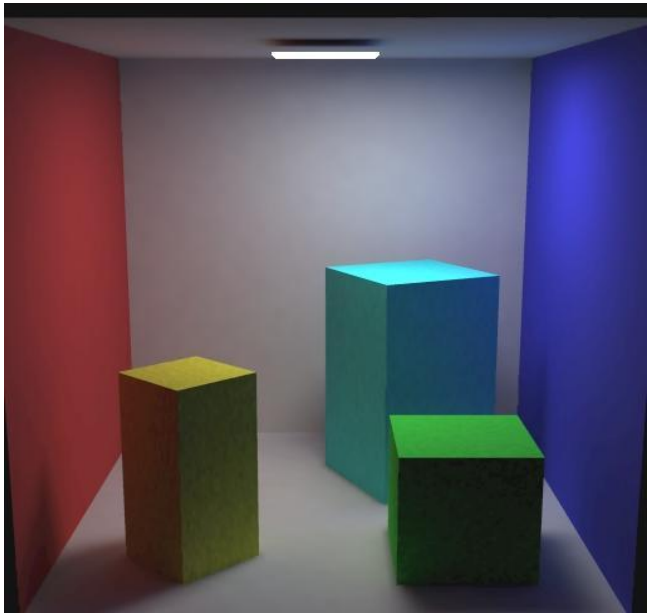


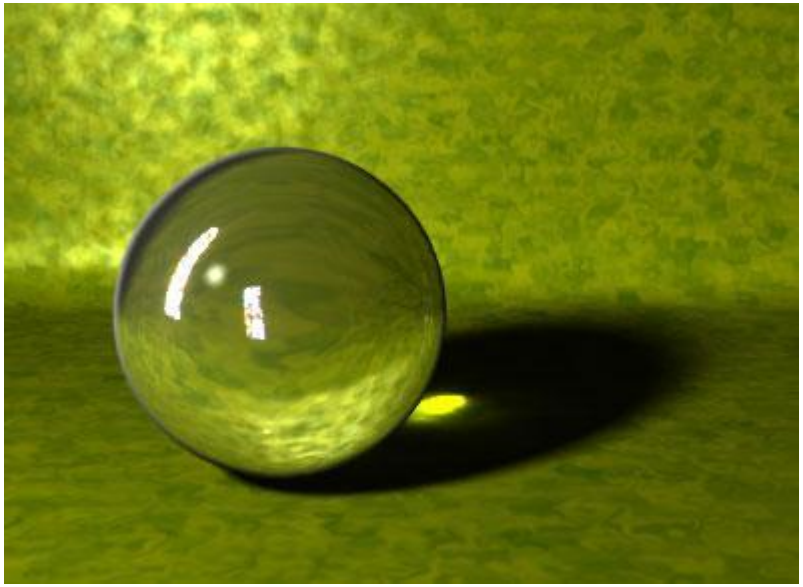
LSDE

*

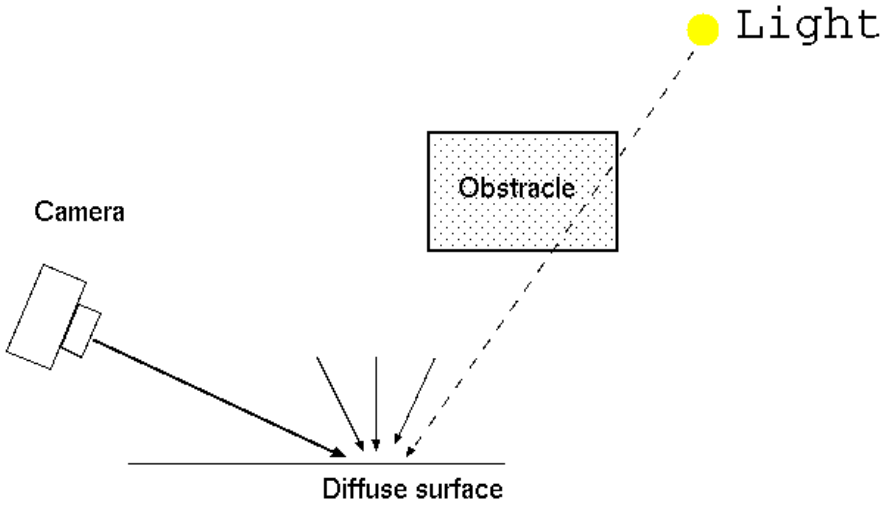
Overview : Global Illumination Methods

- Light Transport Notations
- **Monte-Carlo Ray Tracing**
- Photon Mapping





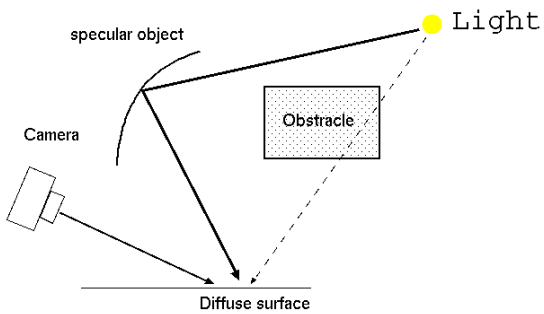
Ray Tracing : review



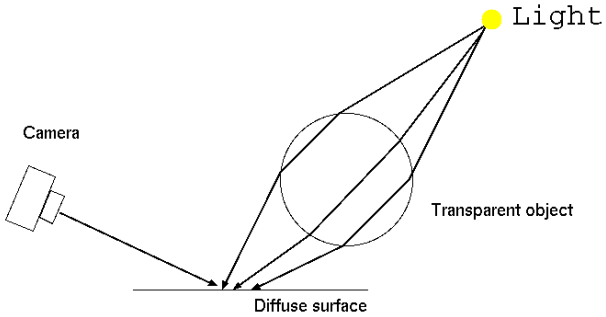
- Shadow ray, reflection ray, etc.
- We simply do a local illumination at diffuse surfaces using the direct light
- We do not know where the indirect light that lit the diffuse surface comes from

Problems simulating indirect lighting by ray-tracing

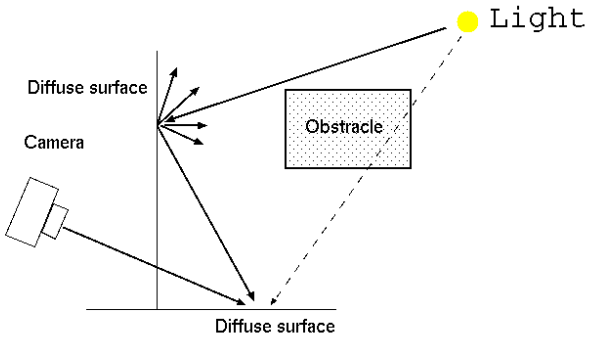
LSDE



LSDE



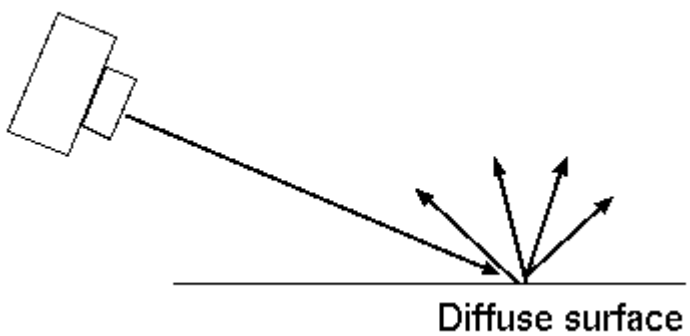
LDDE

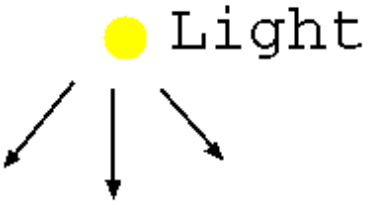


- Caustics and color bleeding are produced by indirect light – how can we simulate such effects in the ray tracing framework?

Two ways to simulate indirect light

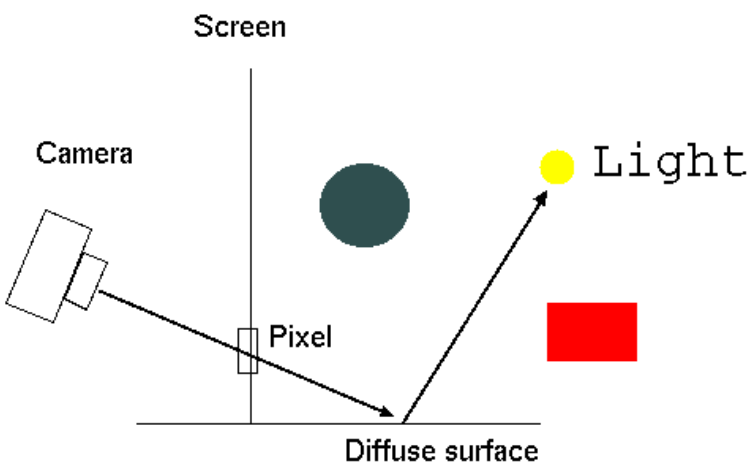
- Launch tracing rays in random directions at diffuse surfaces -> **Path tracing**
- Shoot rays that represent the path of light from the light source-> **Bidirectional Path Tracing, Photon Mapping**

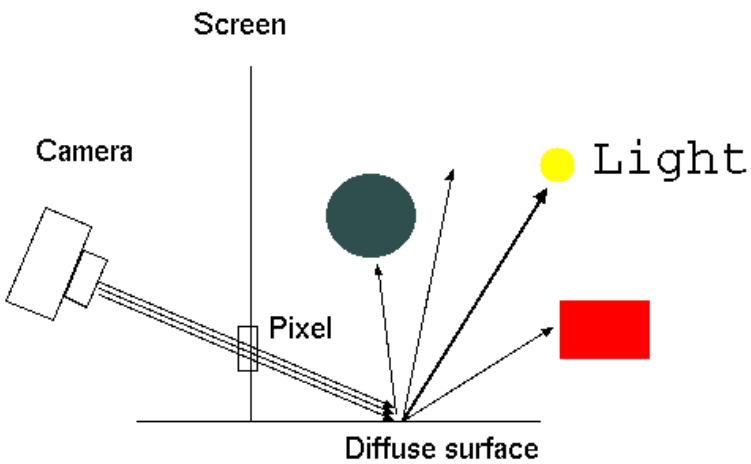




Path Tracing

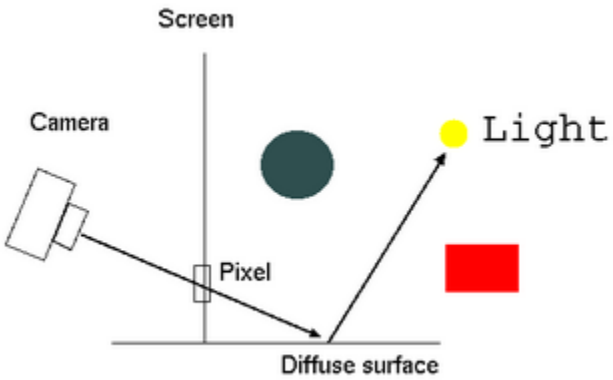
- An enhancement of the ordinary ray-tracing scheme
- But when hitting a diffuse surface, pick one ray at random, and find the colour of the incoming light





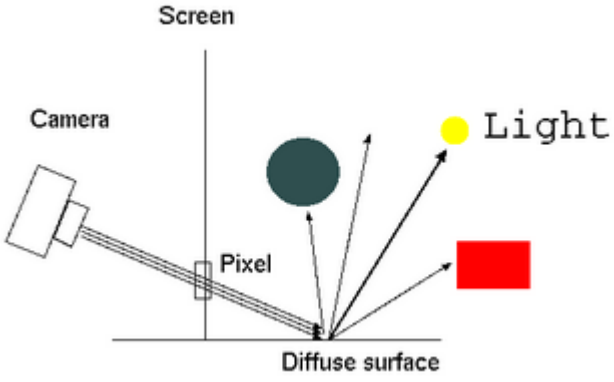
Original Ray Tracing Algorithm

- Trace (ray)
  - Find the intersection of the ray and the scene
  - Compute the shadow ray : Color=Color_ambient
  - Do the local illumination : Color += Color_local (not shadowed)
  - If specular compute the reflection vector R
    - Color += Trace(R)
  - If refractive compute the refractive vector T
    - Color += Trace(T)

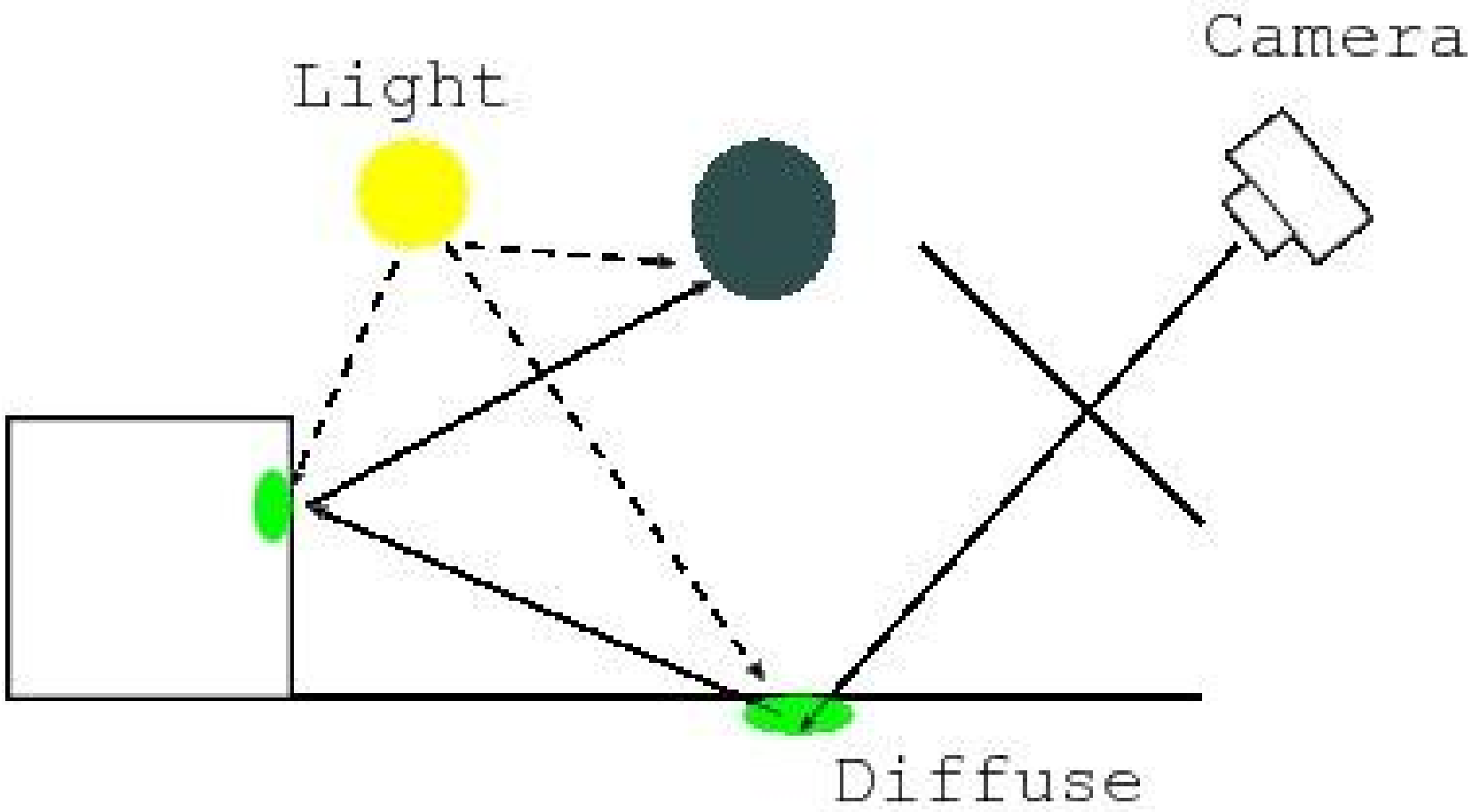


Path Tracing Algorithm

- Trace (ray)
  - –Find the intersection of the ray and the scene
  - –Compute the shadow ray : Color=Color_ambient
  - –Do the local illumination : Color += Color_local (not shadowed)
  - –If specular compute the reflection vector R
    - Color += Trace(R)
  - –If refractive compute the refractive vector T
    - Color += Trace(T)
  - **–Else if diffuse compute a random vector R'**
    - **Color += Trace(R')**



Approach



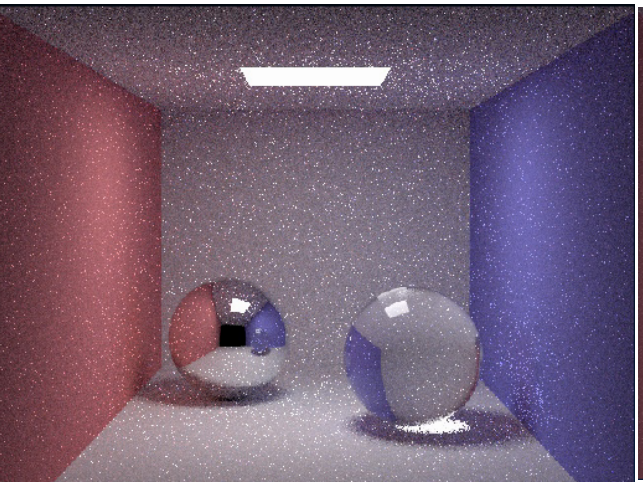
Path Tracing : algorithm

```
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
```

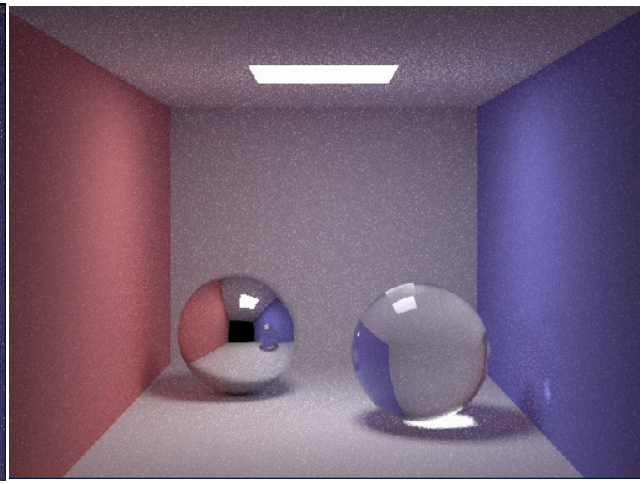
```
Shade ( point, normal )
    color = 0
    for each light source
        test visibility on light source
        if visible
            color=color+direct illumination
    if diffuse surface
        color = color + trace ( a randomly reflected ray)
    else if specular
        color = color + trace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

Path tracing : problems

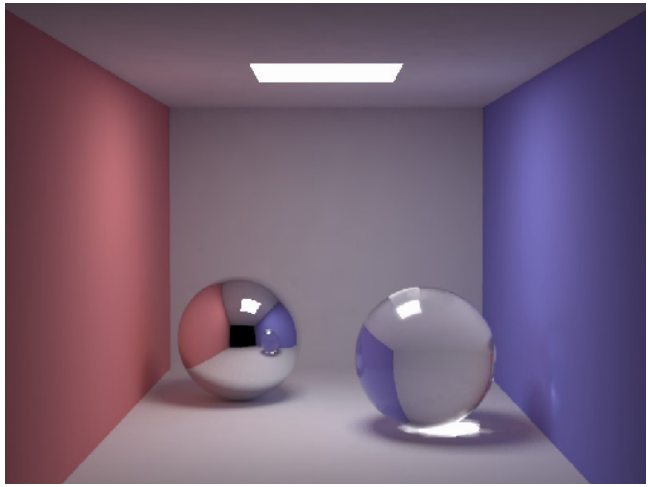
- Variance in the pixel colours, appearing as noise
- Need many samples for precise results
  - Requires 1000 ~ 10000 samples per pixel for good results



10 paths / pixel



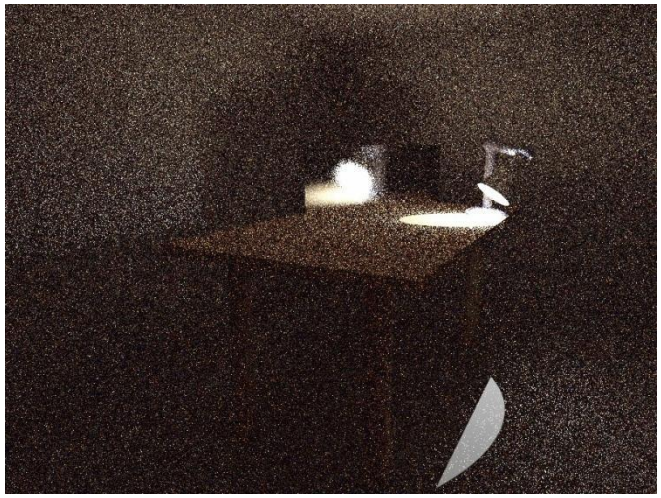
100 paths / pixel

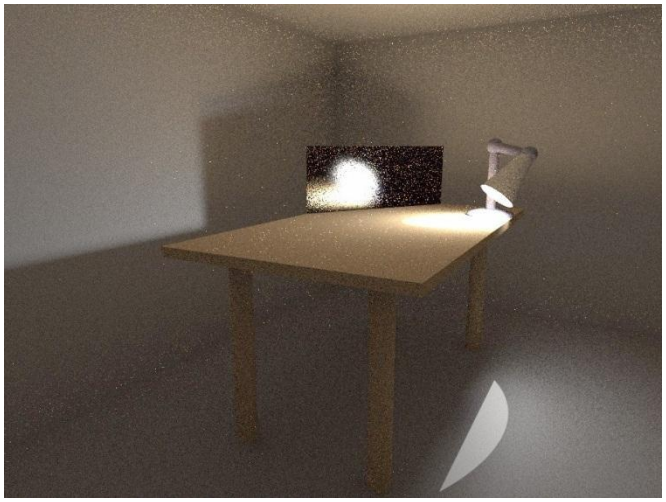


1000 paths / pixel

Path Tracing : Problems (2)

- Some lights are difficult to reach from the camera - such as those produced by spot lights
- For such lights, we cannot simulate indirect light well
- Results in a very dim image with high variance

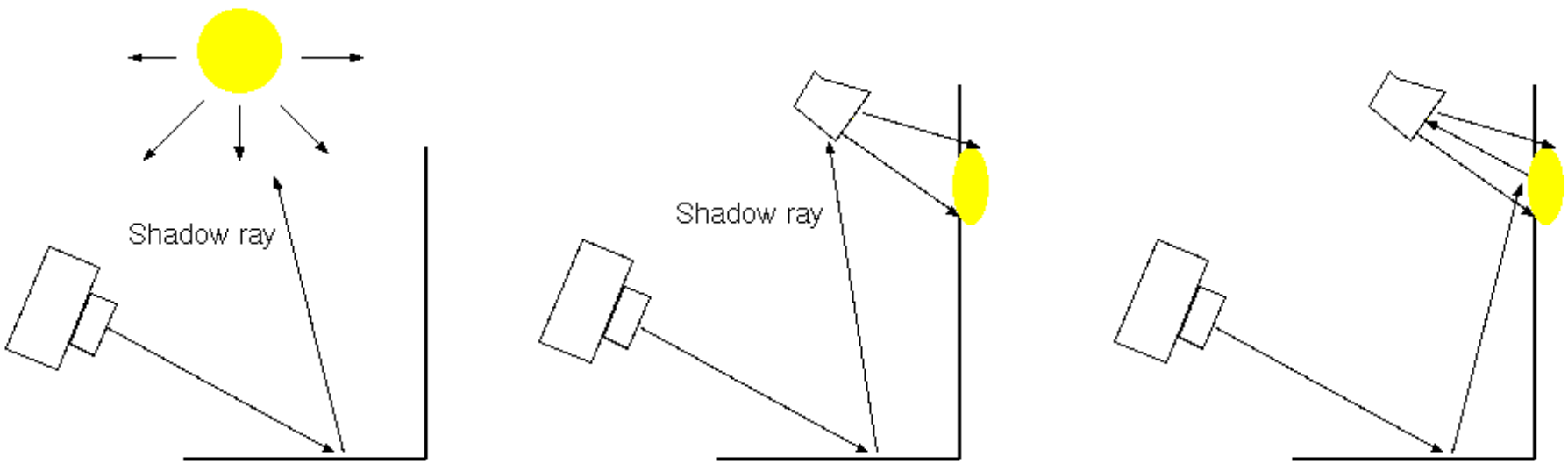




*

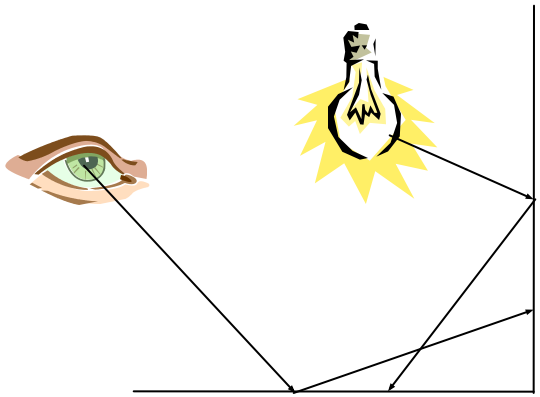
Why? Because the shadow rays are always occluded

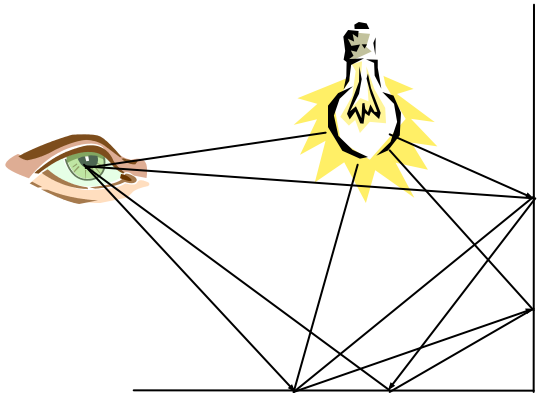
- For the pixel to be lit, the path must be lucky enough to reach the light source



Bidirectional Path Tracing

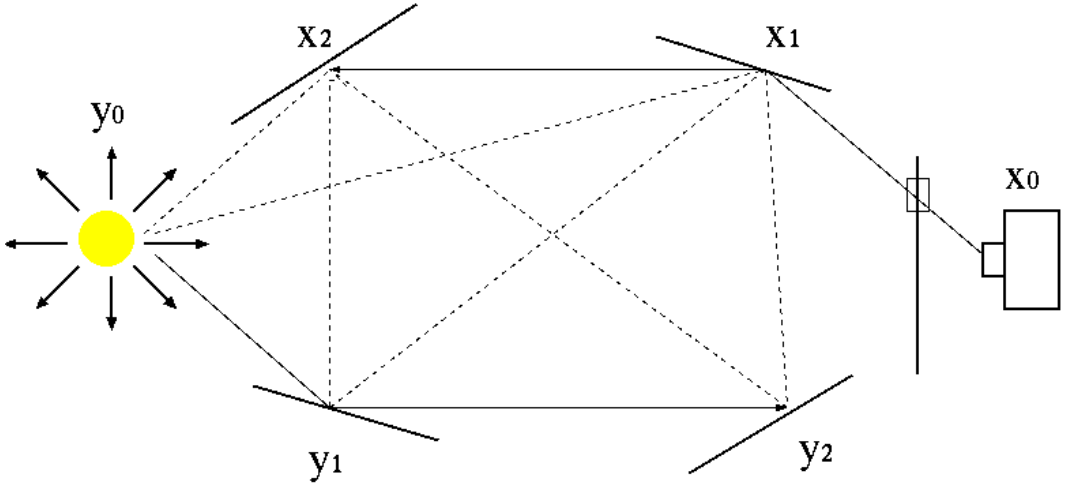
- When the light hits a diffuse surface, the light is reflected in random directions
  - This is like another light source
  - We should use this for local illumination (shadow ray) too





Procedure

- Compute a light path $y_0, y_1, \ldots, y_n$
- Compute an eye path $x_0, x_1, \ldots, x_m$
- The color of the fragment at x_1 is
  - The amount of light reaching x_1 from $y_0, \ldots, y_n$ and reflecting towards x_0 plus
  - The amount of light reaching x_1 from x_2 and reflecting towards x_1

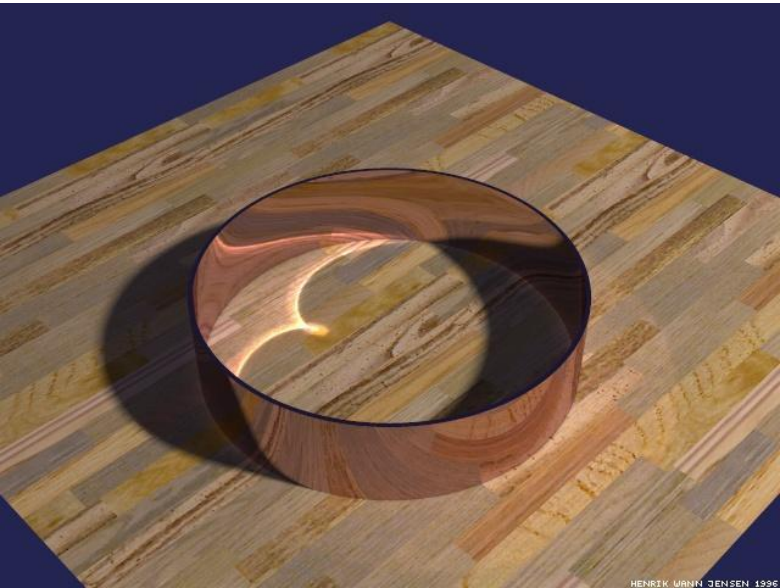


Comparison

(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 bidirectional methods work better than one way path tracing?

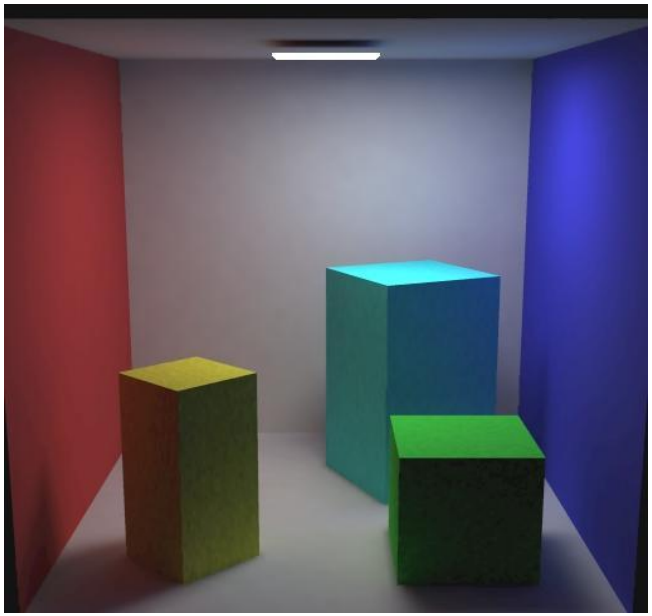
- Caustics
  - Easier to produce by tracing from the light source
- When the light sources are not easy to reach from the eye
  - less shadow rays reach the light source

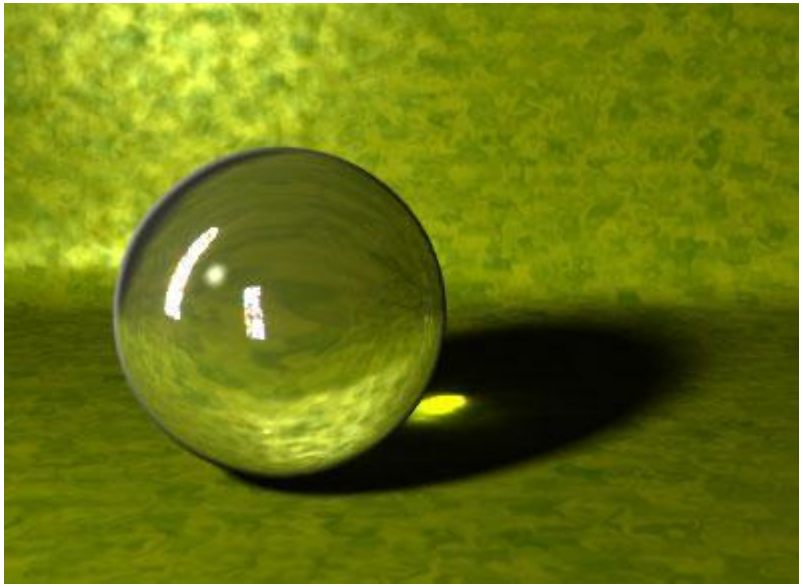
Summary for Monte Carlo Ray Tracing

- An approach that simulates the light reflection at diffuse surfaces
- Can simulate indirect lighting
- Results are subject to variance
  - **Requires a lot of samples per pixel to reduce the noise**
  - Bidirectional methods can reduce the noise

Today : Global Illumination Methods

- Light Transport Notations
- Monte-Carlo Ray Tracing
- Photon Mapping





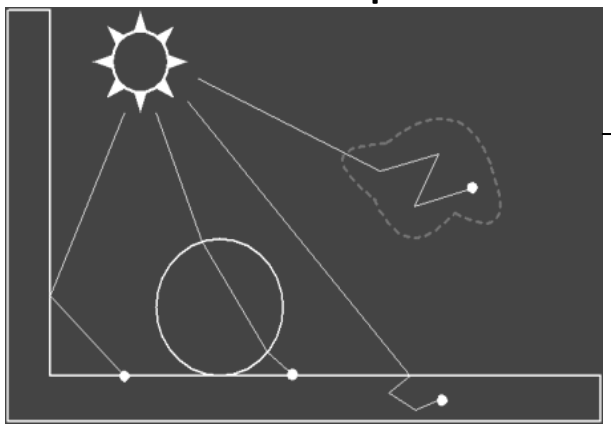
Photon Mapping

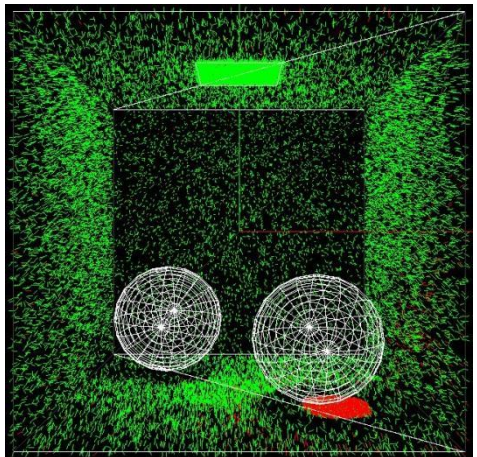
- A fast, global illumination algorithm based on Monte-Carlo method
- A stochastic approach that estimates the radiance from **limited number of samples**

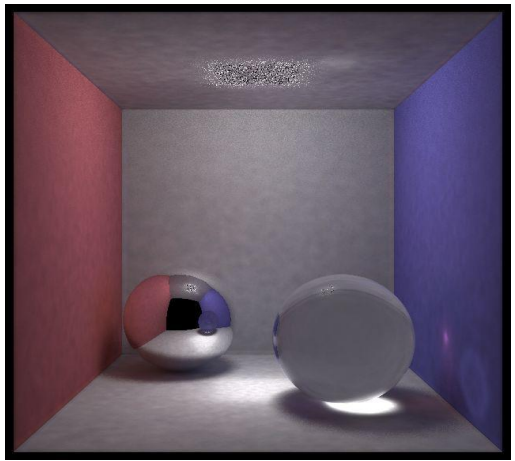
http://www.youtube.com/watch?v=wqWRVcsIcAQ

Photon Mapping

- A two pass global illumination algorithm
  - First Pass - Photon Tracing :
    - Casting photons from the light source, and
    - saving the information of reflection in the “photon map”,
  - Second Pass – Rendering (radiance estimate)
    - the brightness of the pixels are estimated from the photon map

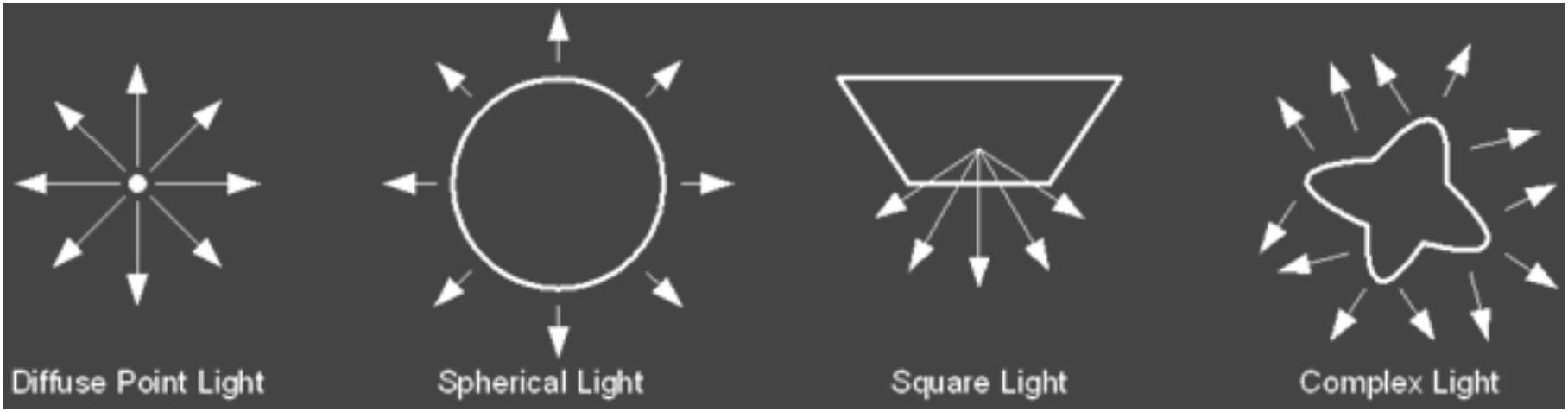






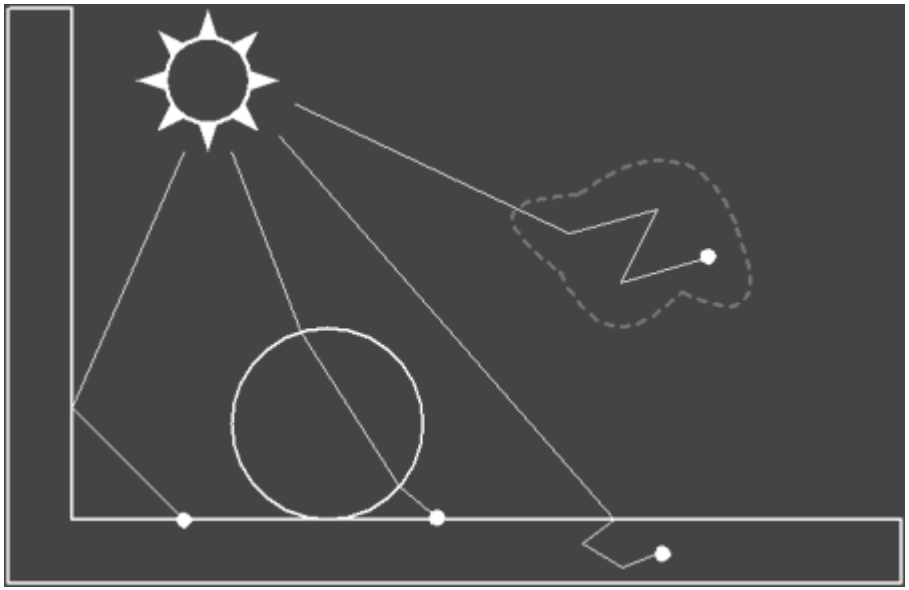
Photon Emission

- 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 happens to photons when they hit surfaces?

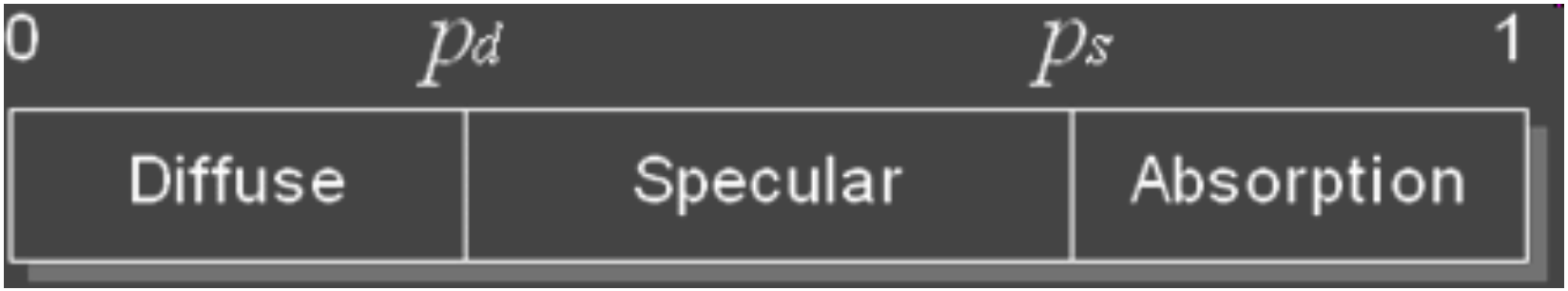
- Reflected or absorbed

There are two ways to determine this:

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

Russian Roulette

- If the surface is diffuse+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



Probability of diffuse and specular reflection, and absorption

- Probability of reflection

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

- Probability of diffuse reflection

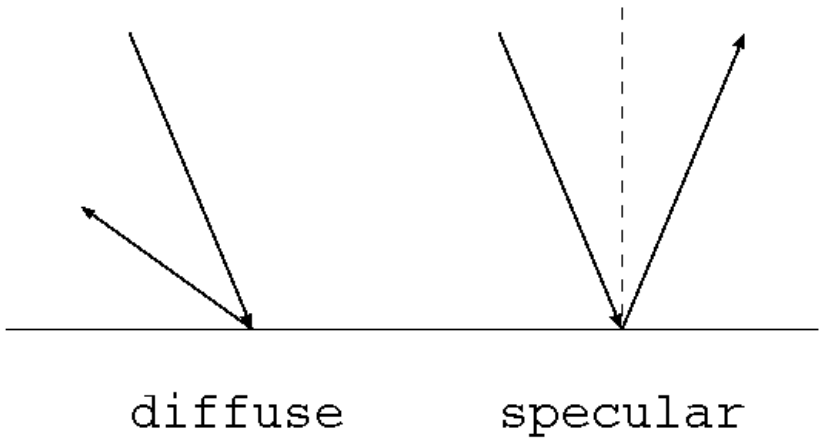
$$P_d = \frac{d_r + d_g + d_b}{d_r + d_g + d_b + s_r + s_g + s_b} P_r \,.$$

- Probability of specular reflection is

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

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



Power Attenuation

- The colour of the light must change after specular / diffuse reflection
- This is essential for producing effects like colour bleeding

Power after reflectance

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

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

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

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

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

Ps is the probability of specular reflection

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

Power after reflectance (diffuse)

- Here is the diffuse version

$$P_{ref,dr} = P_{inc,r} \times d_r / P_d$$

$$P_{ref,dg} = P_{inc,g} \times d_g / P_d$$

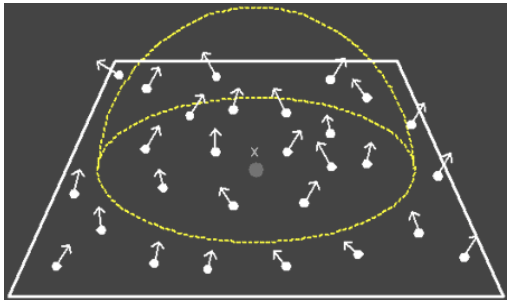
$$P_{ref,db} = P_{inc,b} \times d_b / P_d$$

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

Pd is the probability of diffuse reflection

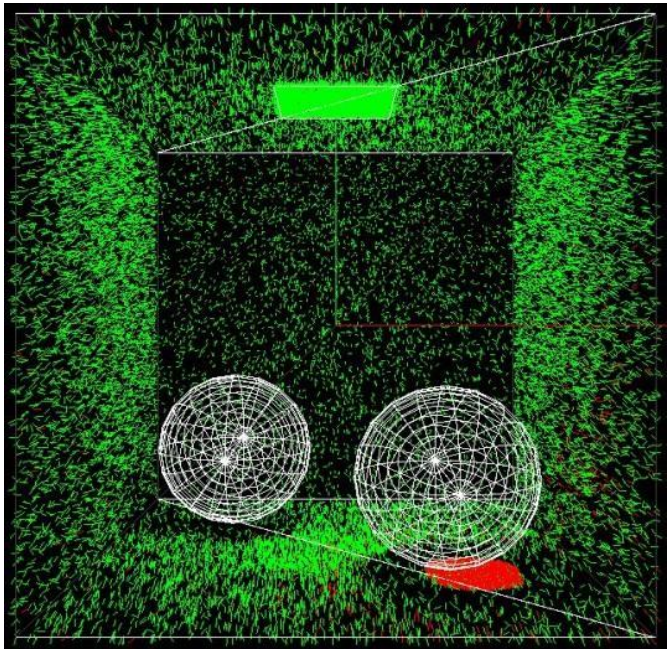
Photon Map

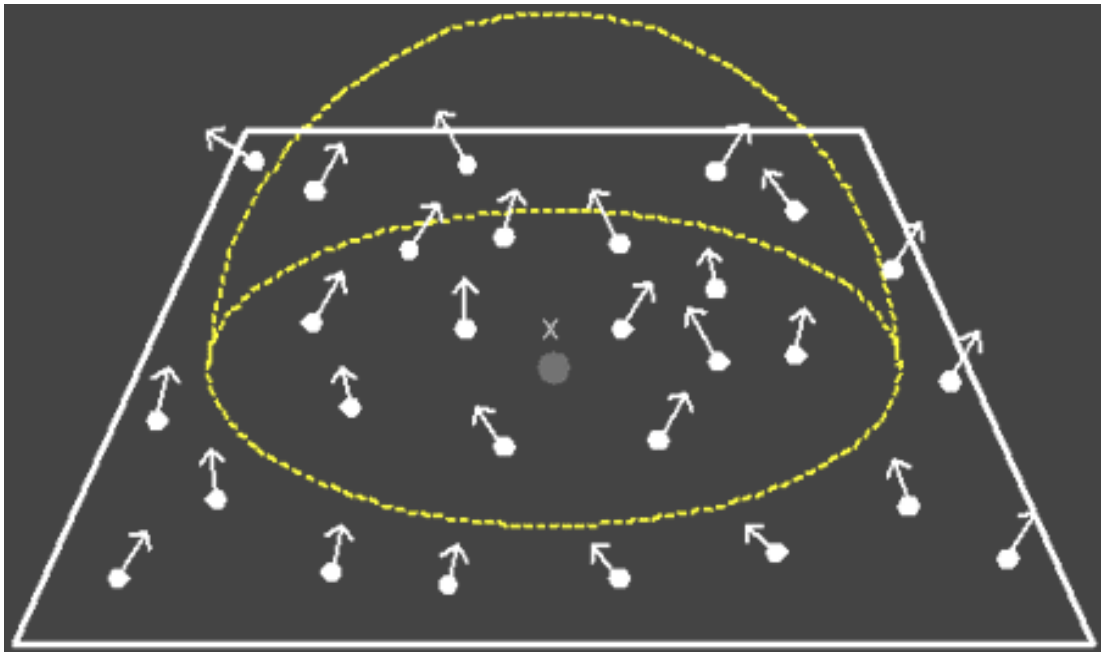
- When a photon makes a diffuse bounce, or is absorbed at the surface, the ray intersection is stored in memory
  - 3D coordinates on the surface
  - Color intensity
  - Incident direction
- The data structure is called Photon Map
- The photon data is not recorded for specular reflections



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

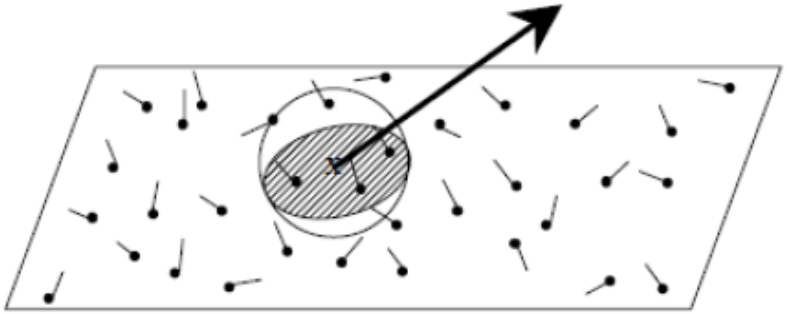
$\overrightarrow{\omega_p}$: incident vector of photon p

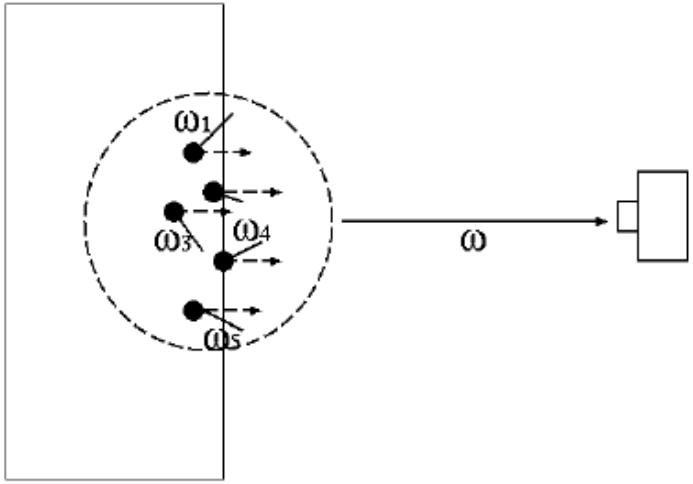
f_r : BRDF

N : the number of photons

$\Delta\Phi_p$: power of photon p

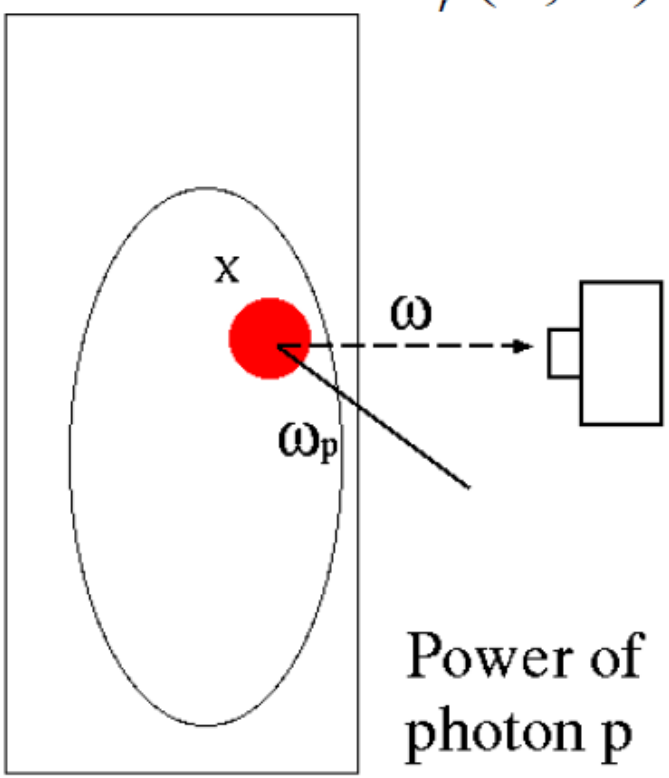
ΔA : Area of the circle πr^2





Radiance Estimation

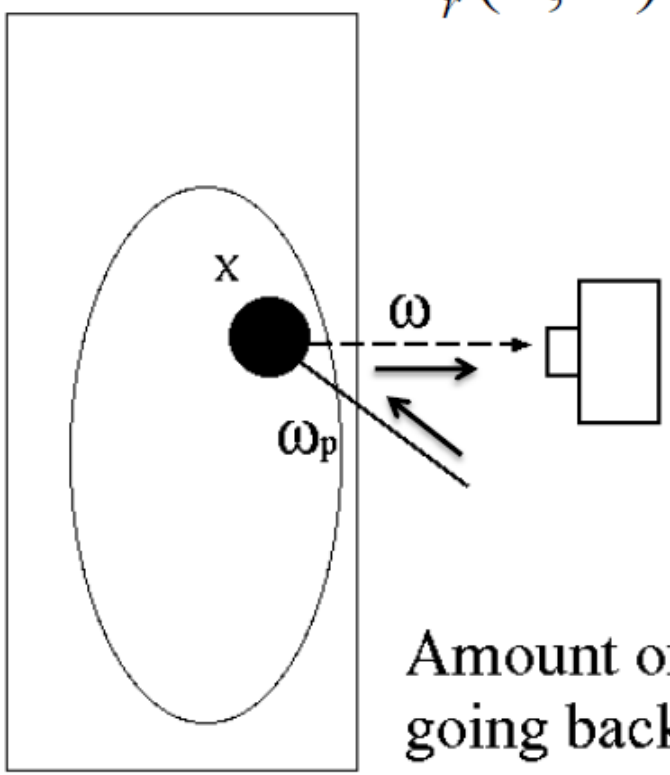
$$L_r(x,\vec{\omega}) = \sum_{p=1}^{N} f_r(x,\vec{\omega_p},\vec{\omega}) \frac{\Delta\Phi_p(x,\vec{\omega_p})}{\Delta A}$$

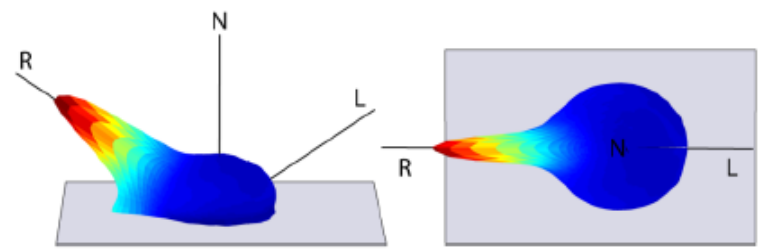


Power of photon p

Radiance Estimation

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

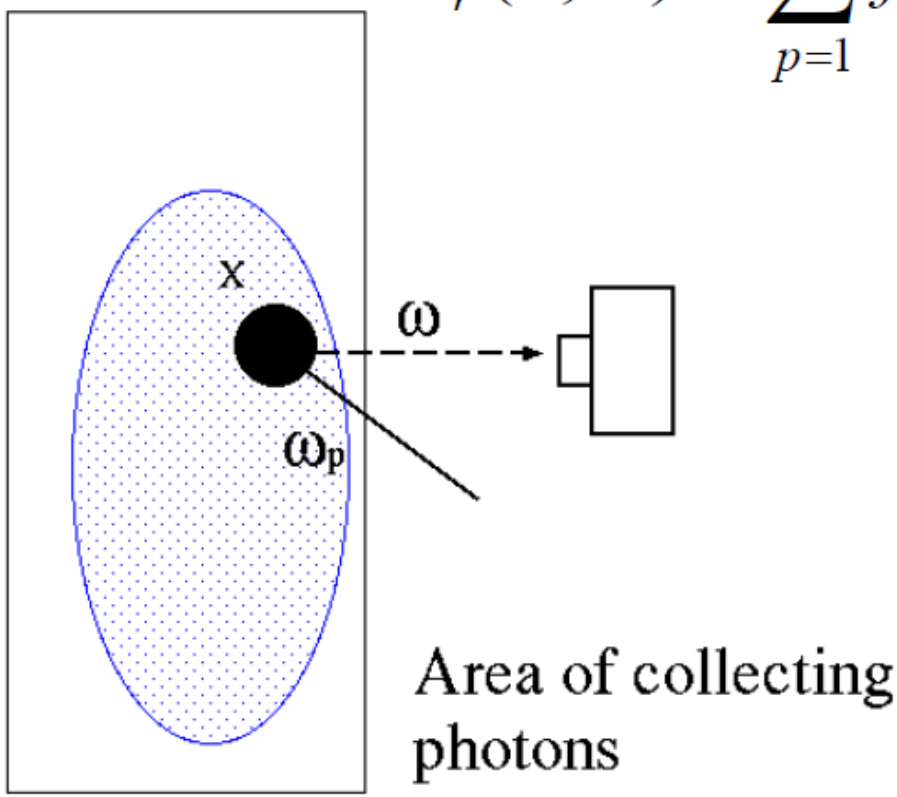




Amount of light coming in from ω_p going back toward ω

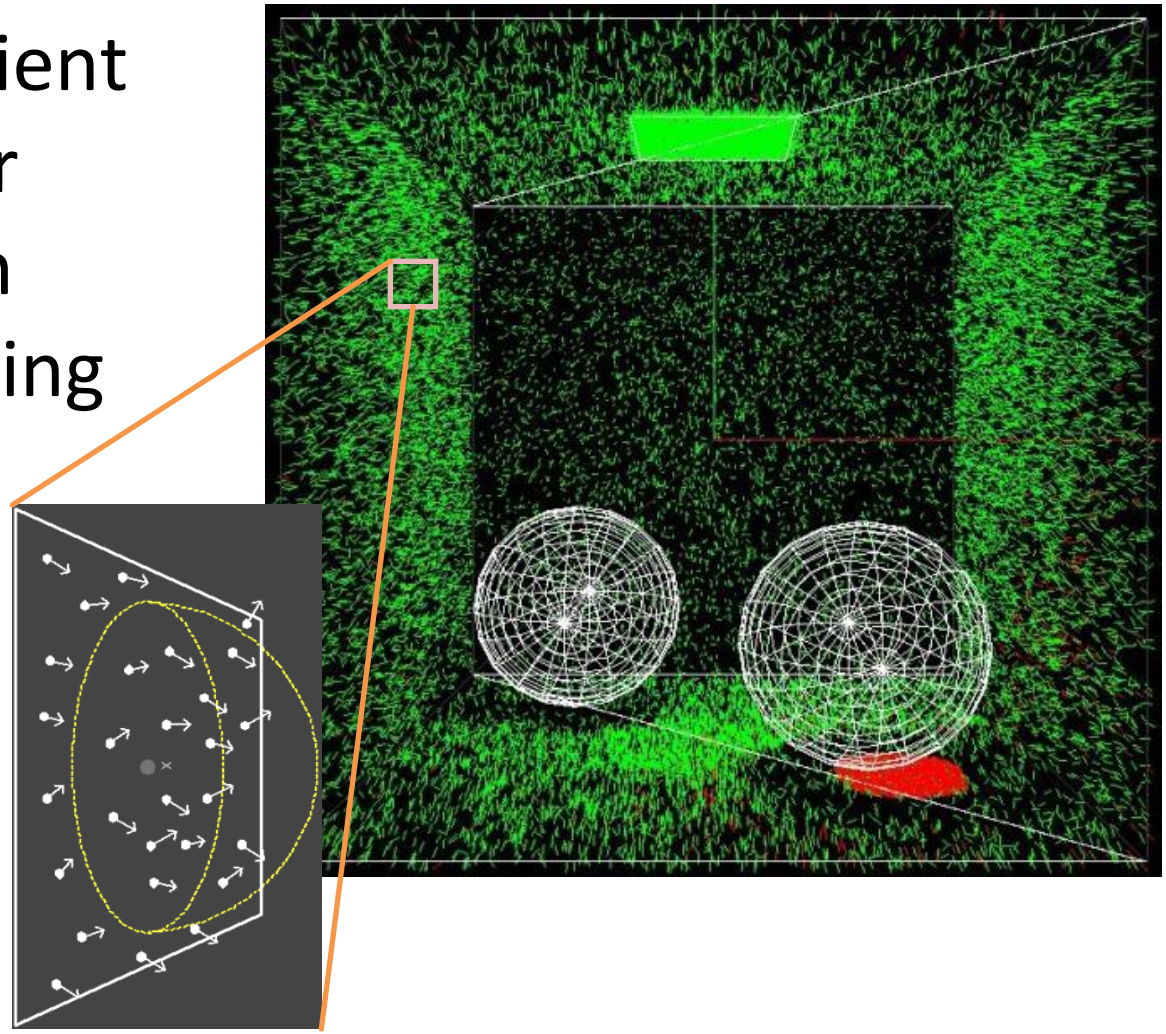
Radiance Estimation

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

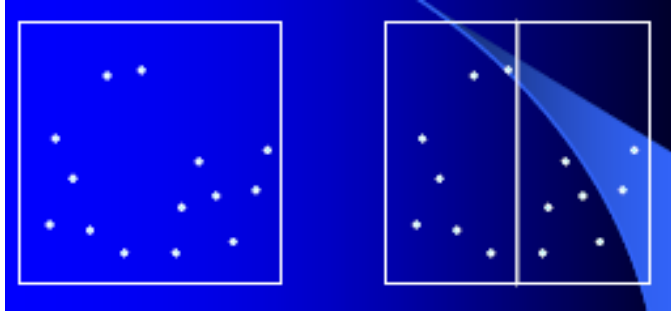


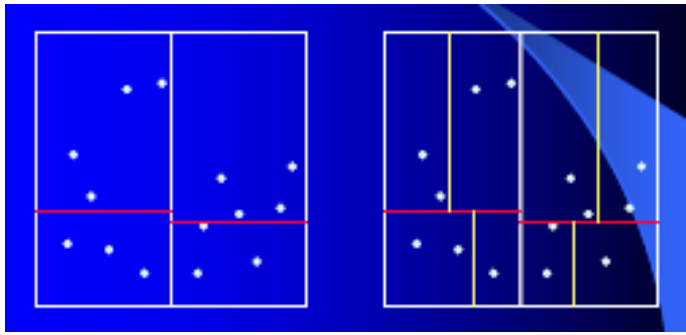
Data Structure for Saving Photon Data

- We need an efficient data structure for retrieving photon maps when coloring the pixels
  - KD-tree
  - Spatial Hash

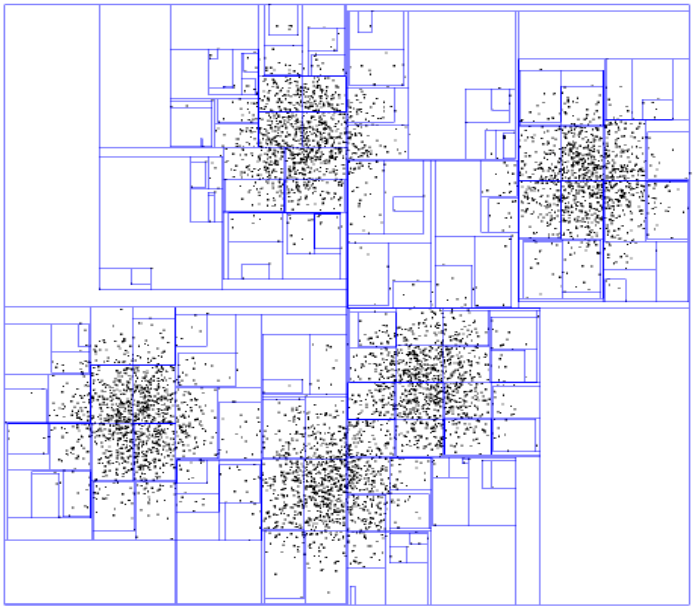


Saving photons: KD tree

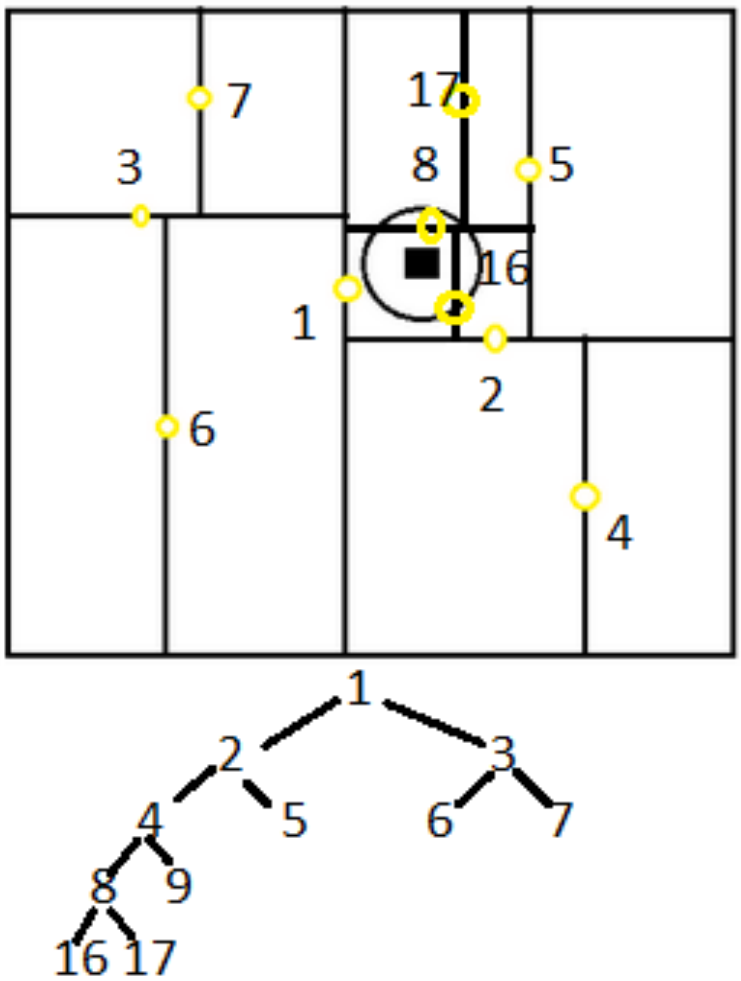




- An efficient hierarchical data structure for saving spatial data
- Procedure to produce it :
  - dividing the samples at the median
  - The median sample becomes the parent node, and the larger data set form a right child tree, the smaller data set form a left child tree
  - Further subdivide the children trees
- Can efficiently find the neighbours when rendering the scene

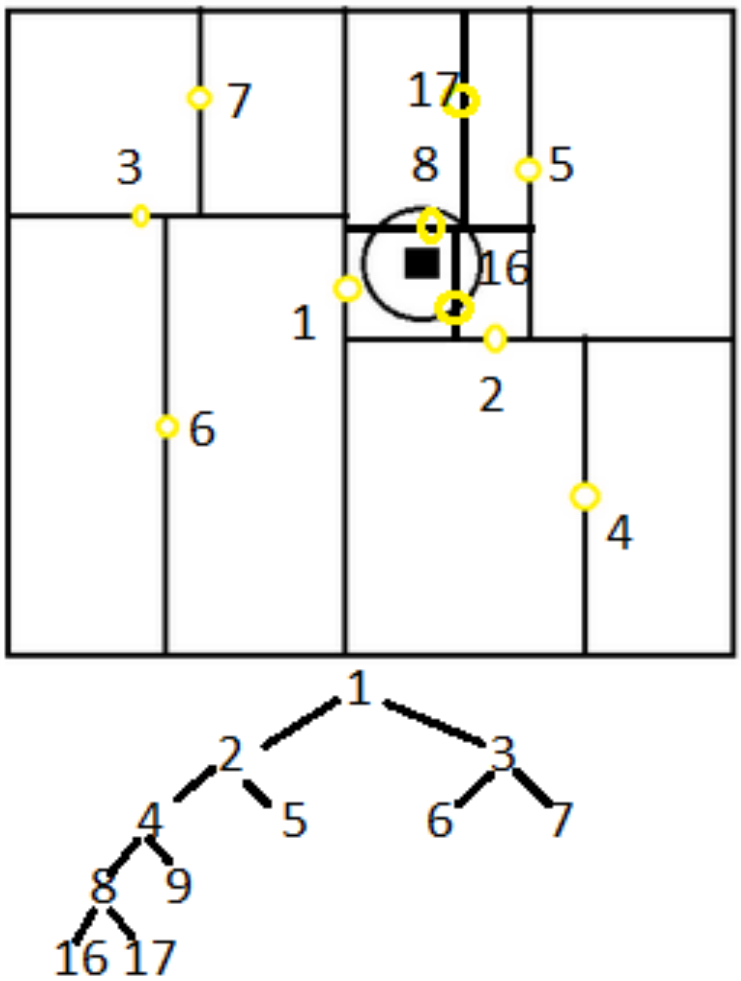


Query for the N-nearest neighbour points



- Given a point X, we traverse the tree to find the nearest N points to X
- Start from the root, you check if the bounding circle is totally within one side or not
- If it is, then you do not have to search the other side at all

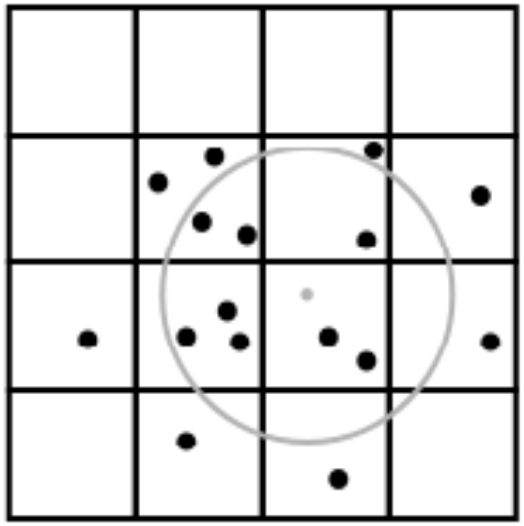
Query for the N-nearest neighbour points (2)



- If the photon is within bounding circle, you add it into the heap
- You further go down to the children nodes.
- The heap is sorted so that the farthest photon is on the top.
- Only the top N photons are kept in the heap.

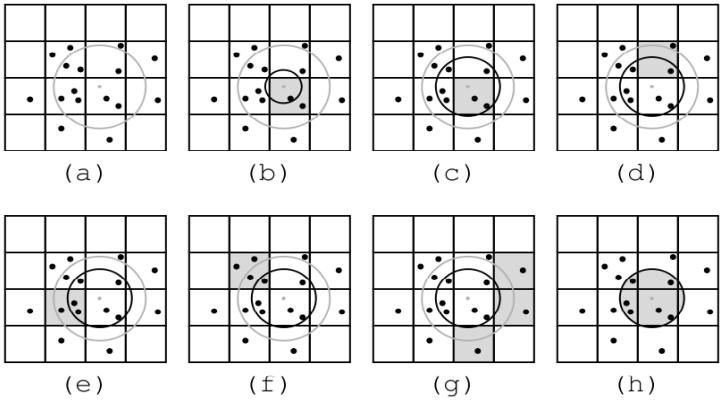
Saving photons: Spatial Hashing

- A uniform 3D grid based hashing system
- 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



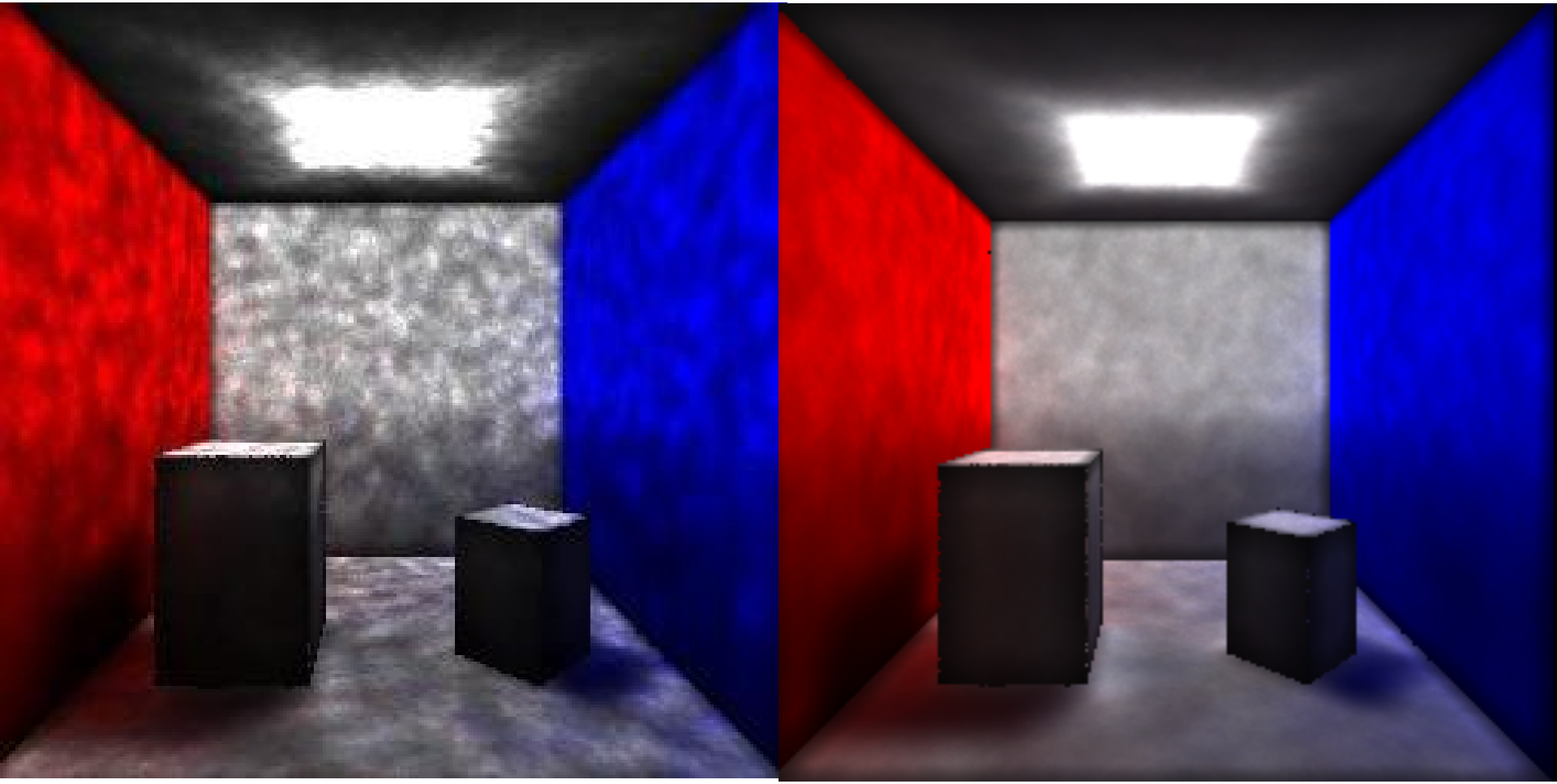
Nearest neighbor-search in the grids

- 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 we reach the photon count
- 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)

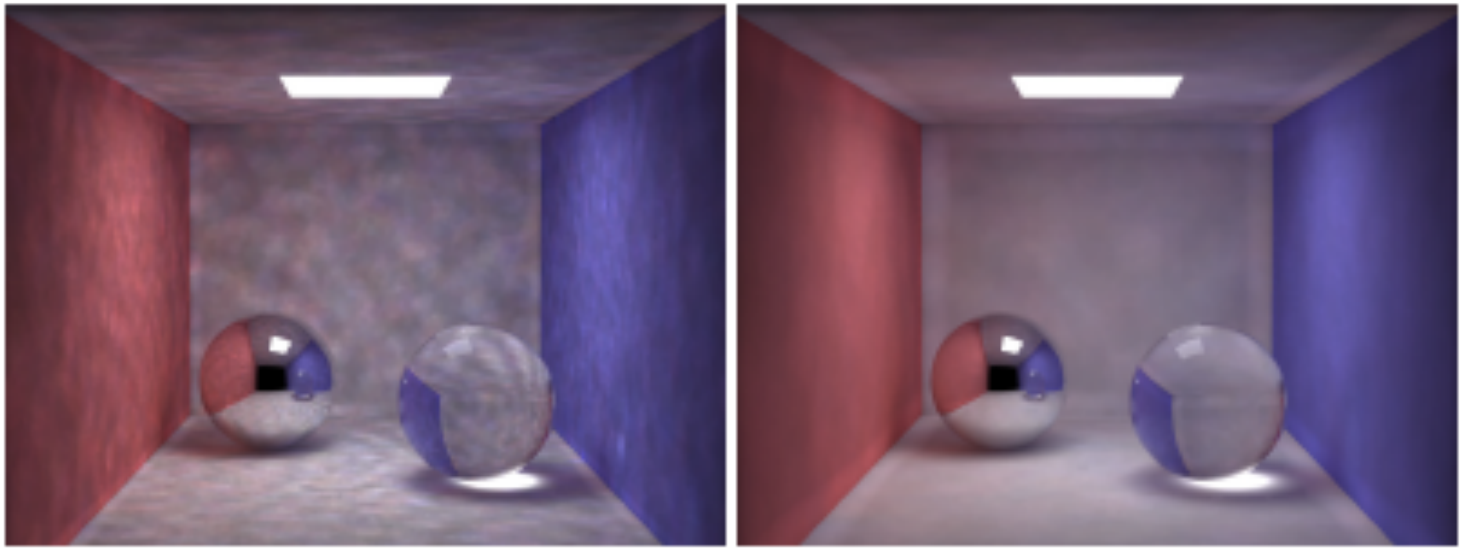


Figure 20: Global photon map radiance estimates visualized directly using 100 photons (left) and 500 photons (right) in the radiance estimate.

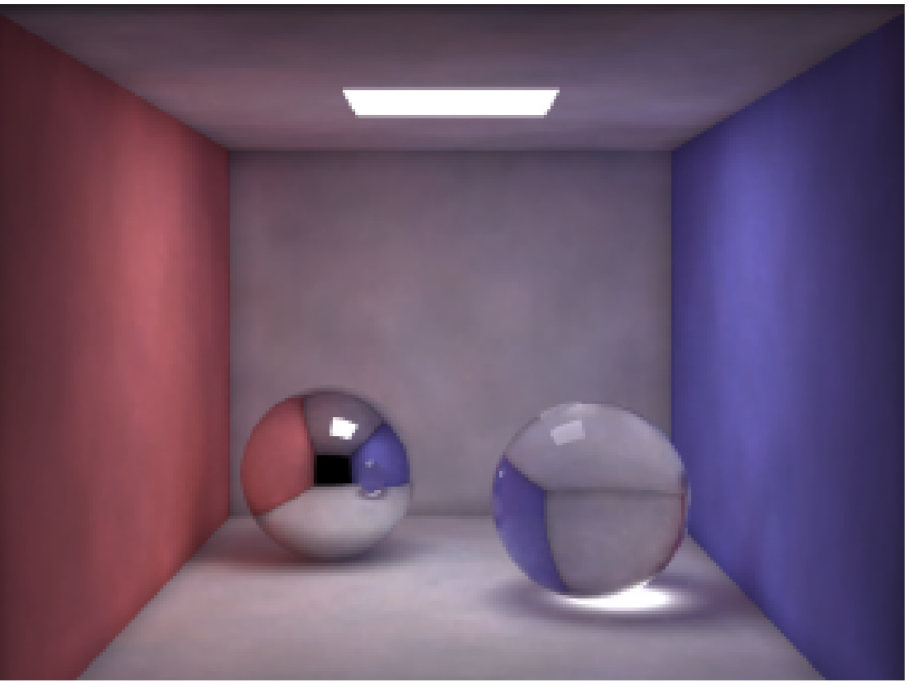


Figure 21: Global photon map radiance estimates visualized directly using 500 photons and a disc to locate the photons. Notice the reduced false color bleeding at the edges.

Summary

- Monte Carlo Ray Tracing
  - Accurate but requires a lot of samples per pixel
  - Suffers from noise which is due to variance
  - Bidirectional method can reduce the variance
- Photon Mapping
  - A stochastic approach that estimates the radiance from a limited number of photons
  - Requires less samples compared 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

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