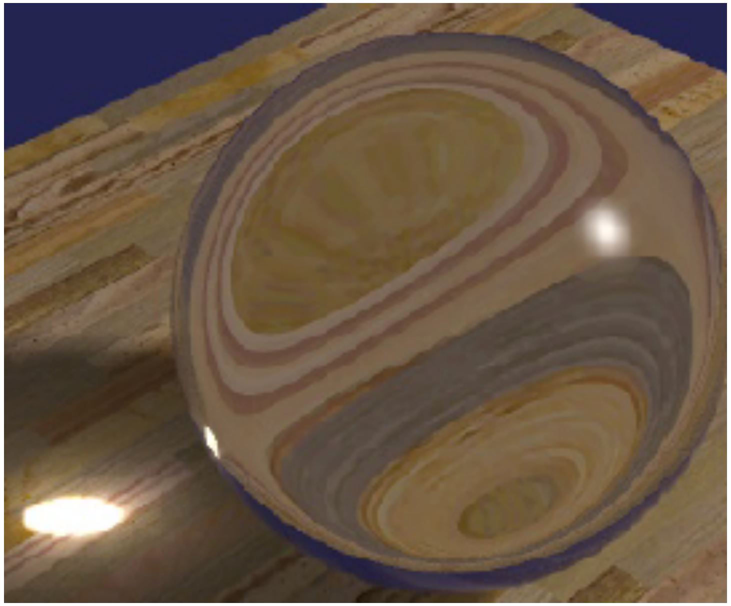
CS 419: Production Rendering

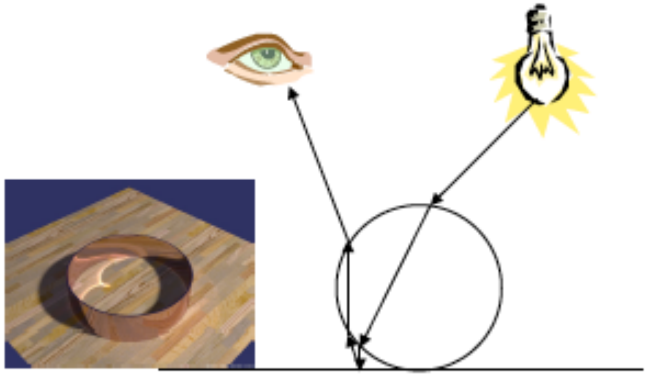
Photon Mapping

Eric Shaffer

Refraction of a Caustic

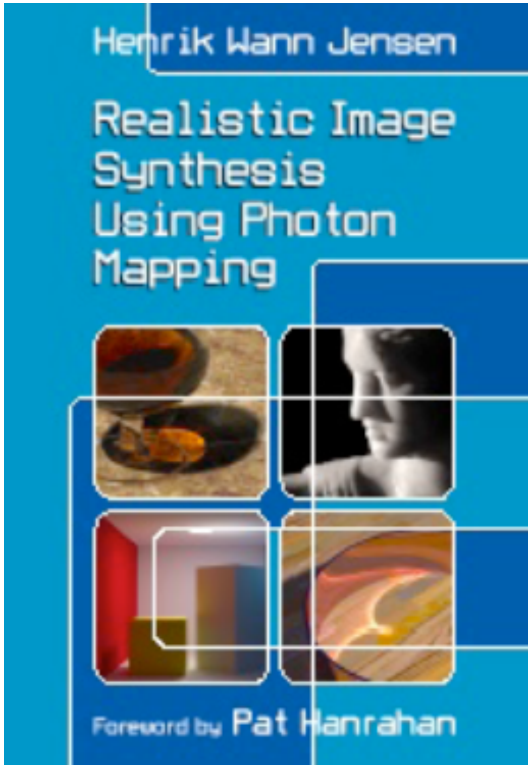
- Monte Carlo ray tracing handles all paths of light L(D|S)*E
  - …but not all equally well
- Has difficulty sampling LS*DS*E paths
- Photon mapping was inspired by this problem
- Also can more easily simulate participating media (eg fog and smoke)





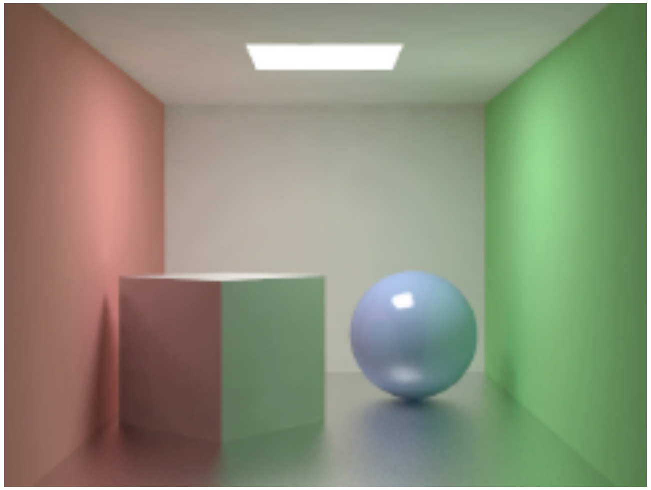
Photon Mapping

- Created by Henrik Wann Jensen in 1996
- Simulates (sort of) the transport of individual photons
- Photons are emitted from light sources
- Bounce off specular surfaces
- Deposited on diffuse surfaces
  - Stored in a k-d tree or similar structure
- Photons are then collected by path tracing

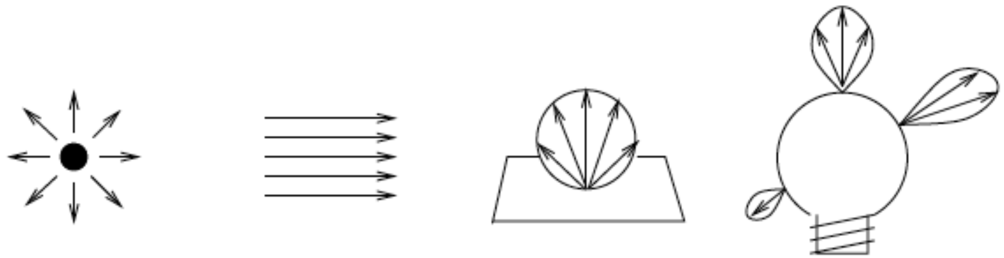


Photon Mapping is Fast(er)

- The flux samples in the photon map allow the use of fewer samples than path tracing to generate a converged solution
- Photon mapping also parallelizes well
- Scene here took 50 minutes with photon mapping
  - 6 hours using Radiance (radiosity+path tracing)



Emission from Light Sources



- Diffuse point lights
  - Emission uniformly in all directions
- Directional lights
  - Emitted in single direction from outside the scene
- Area lights
  - Diffuse square: Emitted in directions sampled from cosine distribution on hemisphere over light
  - General: Sampling directions more complicated but possible

Point Light Emission

- Power of light is distributed among the photons emitted

$$P_{photon} = \frac{P_{light}}{n_e}$$

  - N_e is the number of emitted photons
  - P is "wattage
- Pro-tip: To reduce variation in indirect illumination, the emitted directions should exhibit low variation
  - low-discrepancy quasi-random sampling
  - an you name a quasi-random sequence of numbers?

Point Light Emission

```
emit_photons_from_diffuse_point_light() {
  n_e = 0                     number of emitted photons
  while (not enough photons) {
    do {                      use simple rejection sampling to find diffuse photon direction
      x = random number between -1 and 1
      y = random number between -1 and 1
      z = random number between -1 and 1
    } while ( x^2 + y^2 + z^2 > 1 )

    d = < x, y, z >
    p = light source position
    trace photon from p in direction d
    n_e = n_e + 1
  }
  scale power of stored photons with 1/n_e
}
```

Multiple Lights

- It is possible to use multiple lights
- More powerful lights should emit more photons
  - Keep the power of the photons relatively even
- More lights does not necessarily require more photons
  - Total photons required to illuminate a scene is N
  - N photons divide up among the lights

Projection Map

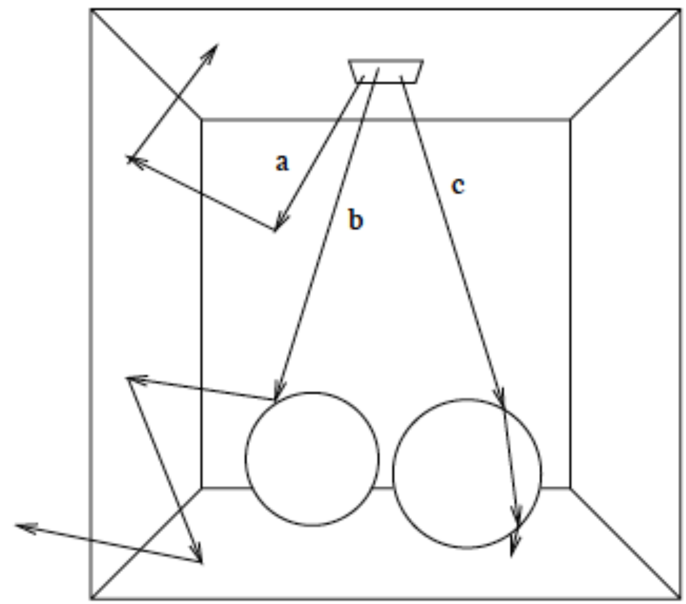
- Emitting into empty space is a waste
- Projection map discretizes the set of directions around a light
  - Like a hemicube
  - Cells are marked as empty space or not
- To emit:
  pick a random non-empty cell and random direction in that cell
- Need to scale energy of the photons (why?)

$$P_{photon} = \frac{P_{light}}{n_e} \frac{\text{cells with objects}}{\text{total number of cells}}$$

Tracing Photons

- Once emitted, we trace photons into the scene like rays
- Photons contact surfaces and can be
  - Reflected
  - Transmitted
  - Absorbed
- Material properties and Russian Roulette
  - Determine photon fate
- Example: Surface with
  - diffuse reflection coefficient d
  - Specular reflection coefficient s
  - d+s ≤ 1
  - Generate random number in [0,1]

$$\begin{aligned} \xi \in [0, d] &\longrightarrow \text{diffuse reflection} \\ \xi \in]d, s+d] &\longrightarrow \text{specular reflection} \\ \xi \in]s+d, 1] &\longrightarrow \text{absorption} \end{aligned}$$



Tracing Photons

- Need to handle at least 3 color bands
- Base decision on max probability of reflection in any band

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

- 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

$$P_s = \frac{s_r + s_g + s_b}{d_r + d_g + d_b + s_r + s_g + s_b} P_r$$

$$\begin{array}{lll} \xi \in [0, P_d] & \longrightarrow & \text{diffuse reflection} \\ \xi \in]P_d, P_s + P_d] & \longrightarrow & \text{specular reflection} \\ \xi \in]P_s + P_d, 1] & \longrightarrow & \text{absorption} \end{array}$$

Tracing Photons

- Power of reflected photons should be adjusted
- For example, with specular reflection

$$P_{refl,r} = P_{inc,r}\, s_r / P_s$$
$$P_{refl,g} = P_{inc,g}\, s_g / P_s$$
$$P_{refl,b} = P_{inc,b}\, s_b / P_s$$

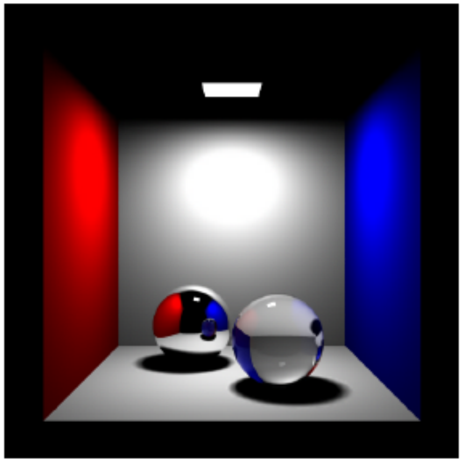
- Can extend the scheme to handle transmission as well
  - Or more colors…
  - Or glossy relfection…

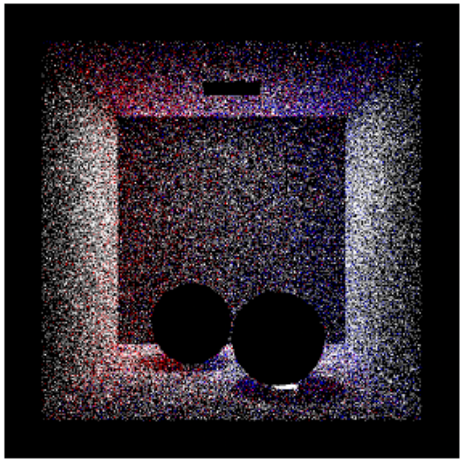
Why use Russian Roulette?

- We can have a photon impact generate multiple new photons
  - e.g. reflection and transmissions
  - e.g. 1 photon can become 256 after 8 bounces
- Russian Roulette will terminate some of these
  - rather than propagating
- What’s the downside to Russian Roulette?

Storing Photons

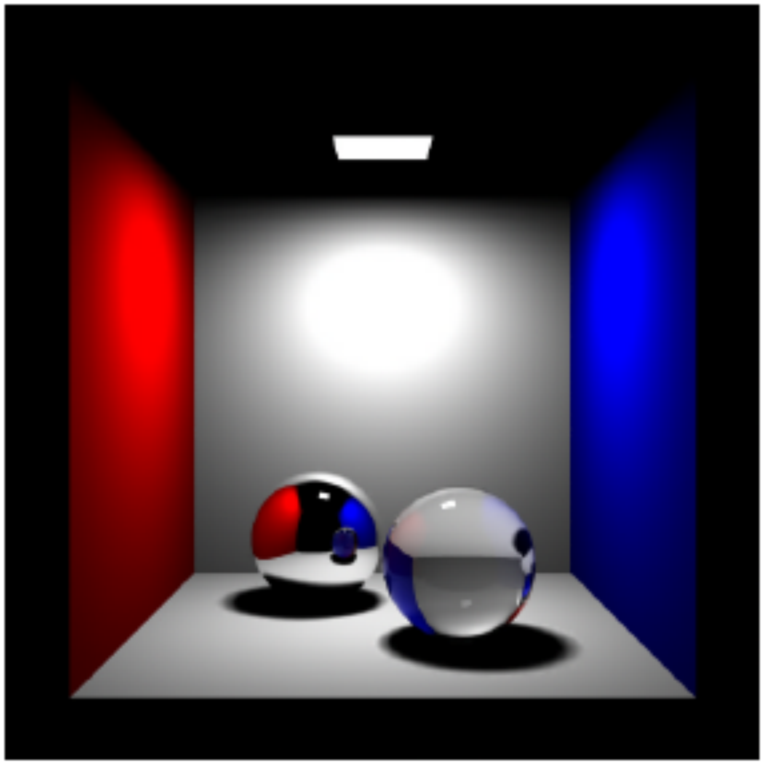
- Photons are not stored on specular surfaces
  - Can you guess why not?
- Are stored on other surfaces



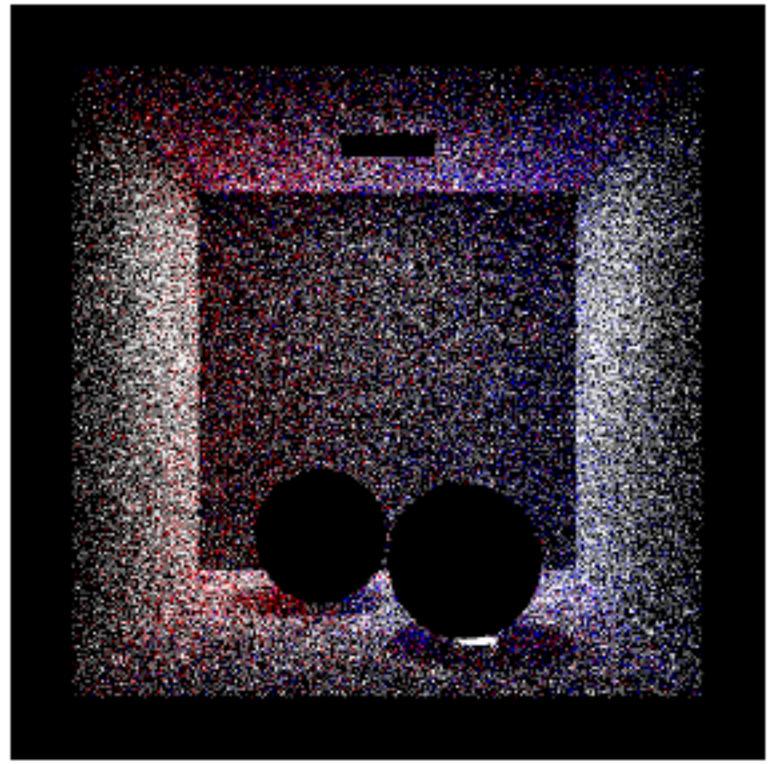


- What’s the bright spot under the sphere in the photon map?

Example



Ray-traced Image



Photon Map

What is a Photon?

```
struct photon {
  float x,y,z;       // position
  char p[4];         // power packed as 4 chars
  char phi, theta;   // compressed incident direction
  short flag;        // flag used in kdtree
}
```

- Power is a float in 3 color channels
  - What format do you think is used to store that information?

Encoding Directions

- Incident direction needed to compute
  - Contribution for non-Lambertian surfaces
  - Front or backfacing arrival on Lambertian surfaces
- Map spherical coordinates of photon to 65536 directions

```
phi = 255 * (atan2(dy,dx)+PI) / (2*PI)
theta = 255 * acos(dx) / PI
```

Multiple Photon Maps

Caustic photon map: contains photons that have been through at least one specular reflection before hitting a diffuse surface: LS^+D.

Global photon map: an approximate representation of the global illumination solution for the scene for all diffuse surfaces: $L\{S|D|V\}^*D$

Volume photon map: indirect illumination of a participating medium: $L\{S|D|V\}^+V$.

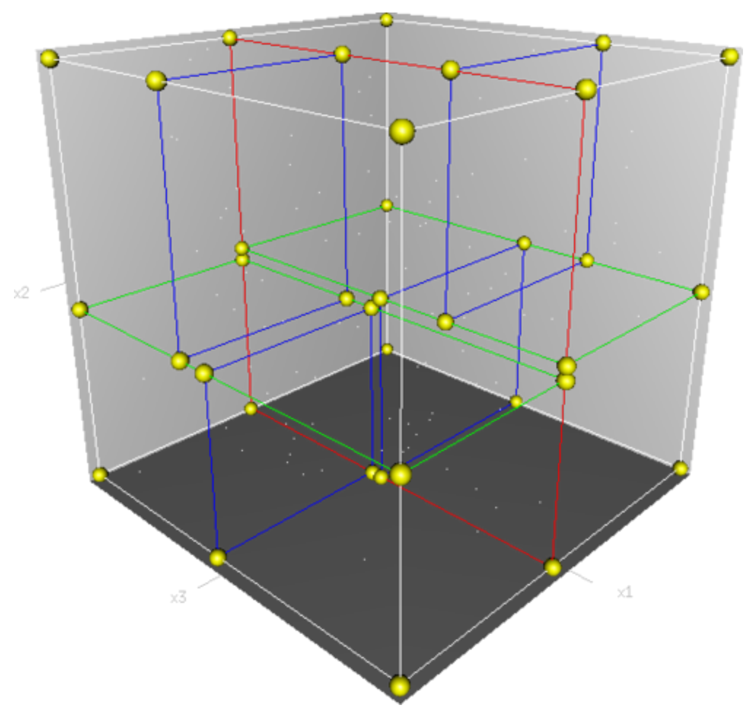
- Usually constructed in 2 passes
  - One for caustic (needs more samples)
  - One for global illumination (including volume)

Storing Photons

- Photons emitted only once during photon tracing
- Stored in a static data structure for the rendering phase
- Structure must support:
  - Query: Closest photon(s) to a point in space
  - Efficiently handle non-uniform distributions of photons
- Jensen used a balanced kd-tree

Balance kd-tree

- Assume N photons in map
- O(log N) to find closest photon
- O(N log N) construction
- Represented by a pointerless heap-like structure
- Photons stored in leaves
- Balanced by median splitting along dimension of greatest coordinate spread



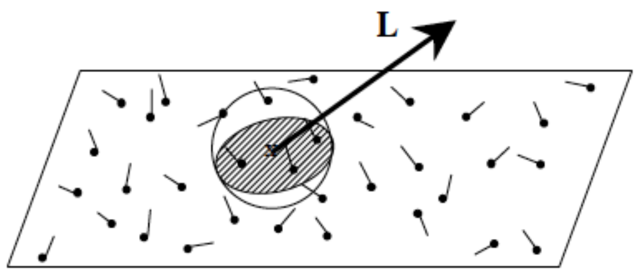
Balanced kd-tree

```
kdtree *balance( points ) {
  Find the cube surrounding the points
  Select dimension dim in which the cube is largest
  Find median of the points in dim
  s1 = all points below median
  s2 = all points above median
  node = median
  node.left = balance( s1 )
  node.right = balance( s2 )
  return node
}
```

Computing Radiance

- To compute radiance we need to integrate incoming flux
- Reflected radiance $$L_r(x,\vec{\omega}) = \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') |\vec{n}_x \cdot \vec{\omega}'| \, d\omega_i'$$
- Relationship between flux radiance $$L_i(x,\vec{\omega}') = \frac{d^2\Phi_i(x,\vec{\omega}')}{\cos\theta_i \, d\omega_i' \, dA_i}$$
- Rewrite the integral

$$\begin{aligned} L_r(x,\vec{\omega}) &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) \frac{d^2\Phi_i(x,\vec{\omega}')}{\cos\theta_i \, d\omega_i' \, dA_i} |\vec{n}_x \cdot \vec{\omega}'| \, d\omega_i' \\ &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) \frac{d^2\Phi_i(x,\vec{\omega}')}{dA_i} . \end{aligned}$$



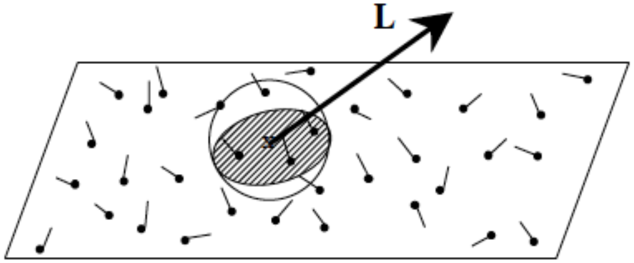
Computing Radiance

- Approximate integral using n photons closest to x

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

- Denominator related to density of photons $\Delta A = \pi r^2$
- Power of a photon is $\Delta\Phi_p(\vec{\omega}_p)$
- Finally we have:

$$L_r(x,\vec{\omega}) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(x,\vec{\omega}_p,\vec{\omega}) \Delta\Phi_p(x,\vec{\omega}_p)$$

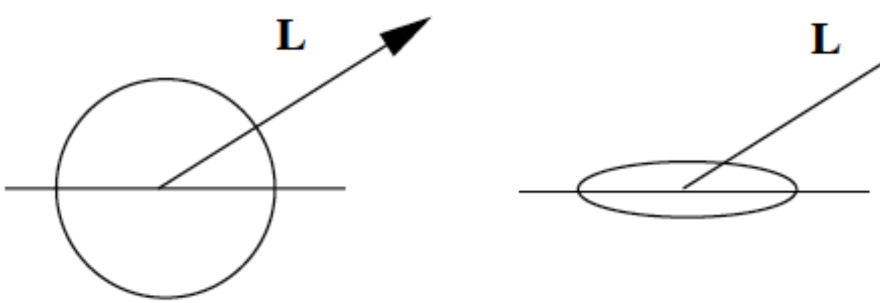


Error in Radiance Estimate

- Wrong photons can be included at corners and sharp edges
  - Result of sampling using a sphere
- Edges and corners also cause the area estimate to be wrong
- Accuracy increases with more photons
- Can show mathematically that arbitrarily good estimates can be obtained using fewer than an infinite number of photons
  - Does not apply to purely specular BRDFs

Alternative Sampling Volumes

- One could use a cube or disc instead of sphere
  - Projected area and distance computations are simple for sphere
- Projected area will change for different volumes
  - Intersection of volume and tangent plane at x
- Disc is attractive alternative
  - What problem would it help solve?



Reducing Error: Filtering

- Visually, error can be seen as blurring at sharp edges
  - eg, lose sharp edge of caustic
- Filtering increases weight of photons closest to x
- Use a radially symmetric 2d filter
  - Cone filter
  - d is distance between x and a photon
  - r is max distance between x and any photon
  - k is a constant <= 1

$$L_r(x,\vec{\omega}) \approx \frac{\sum_{p=1}^{N} f_r(x,\vec{\omega}_p,\vec{\omega})\Delta\Phi_p(x,\vec{\omega}_p)w_{pc}}{(1-\frac{2}{3k})\pi r^2}$$

$$w_{pc} = 1 - \frac{d_p}{k\,r}$$

Gaussian Filter

- r is maximum distance between sample and x
- d is distance between x and a sample
- $\alpha = 0.918$ and $\beta = 1.953$

$$w_{pg} = \alpha \left[1 - \frac{1 - e^{-\beta \frac{d_p^2}{2r^2}}}{1 - e^{-\beta}} \right]$$

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

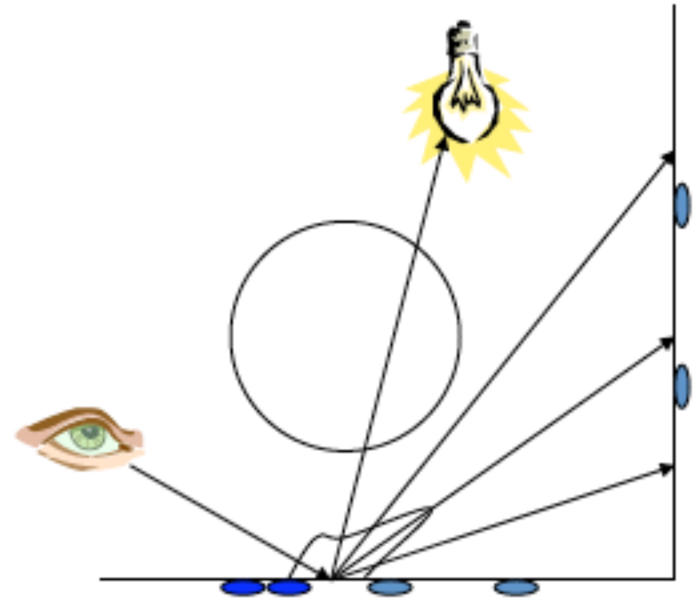
Estimating Radiance in Participating Media

$$
\begin{aligned}
L_i(x,\vec{\omega}) &= \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, L(x,\vec{\omega}')\, d\omega' \\
&= \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, \frac{d^2\Phi(x,\vec{\omega}')}{\sigma(x)\, d\omega'\, dV}\, d\omega' \\
&= \frac{1}{\sigma(x)} \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, \frac{d^2\Phi(x,\vec{\omega}')}{dV} \\
&\approx \frac{1}{\sigma(x)} \sum_{p=1}^{n} f(x,\vec{\omega}'_p,\vec{\omega})\, \frac{\Delta\Phi_p(x,\vec{\omega}'_p)}{\frac{4}{3}\pi r^3}\,,
\end{aligned}
$$

- Scattering coefficient $\sigma(x)$
- f is the phase function

Rendering

- Rendered by glossy-surface distributed ray tracing
- When ray hits first diffuse surface…
  - Compute direct illumination
  - Compute reflected radiance of caustic map photons
  - Ignore global map photons
  - Importance sample BRDF f_r as usual
  - Use global photon map to importance sample incident radiance function L_i
  - Evaluate reflectance integral by casting rays and accumulating radiances from global photon map

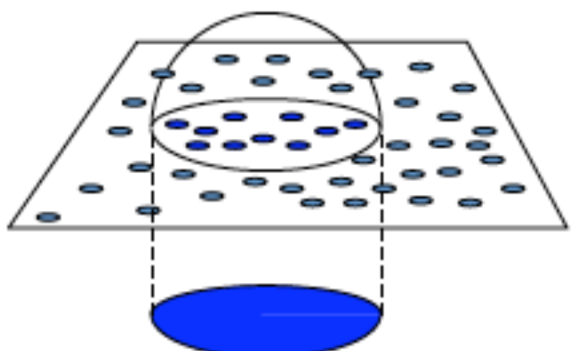


First diffuse intersection. Return radiance of caustic map photons here, but ignore global map photons

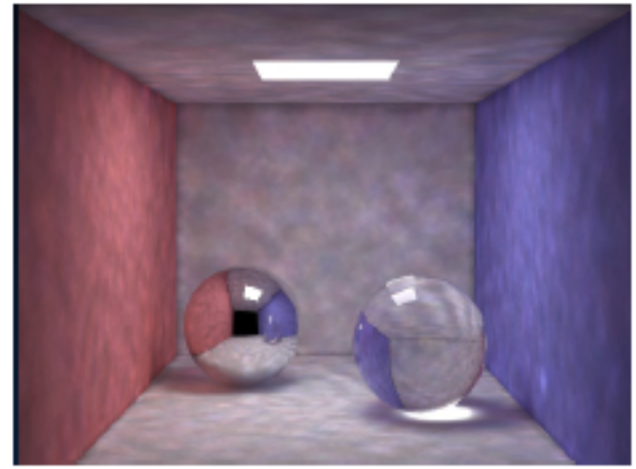
Use global map photons to return radiance when evaluating Li at first diffuse intersection.

How Many Photons?

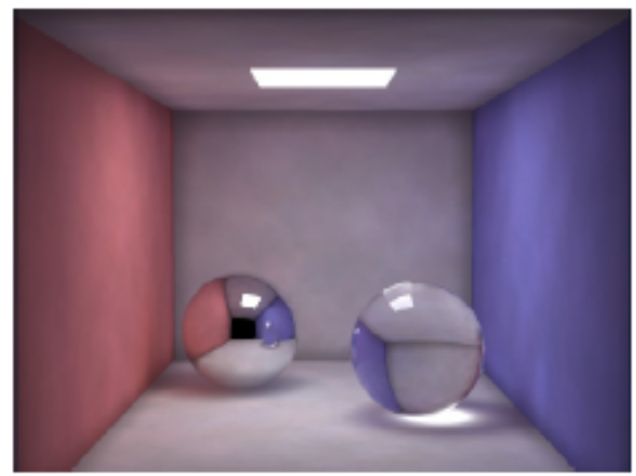
- How big is the disk radius *r*?
- Large enough that the disk surrounds the *n* nearest photons.
- The number of photons used for a radiance estimate *n* is usually between 50 and 500.



$\Delta A = \pi r^2$



Radiance estimate using 50 photons



Radiance estimate using 500 photons