Photon Mapping
Photon mapping

A two-pass method

Pass 1: Build the photon map (photon tracing)

Pass 2: Render the image using the photon map
Building the Photon Map

Photon Tracing

CSE272: Advanced Image Synthesis (Spring 2010) Henrik Wann Jensen
Rendering using the Photon Map
Photon Tracing

- Photon emission
- Projection maps
- Photon scattering
- Russian Roulette
- The photon map data structure
- Balancing the photon map
What is a photon?

- Flux (power) - not radiance!
- Collection of physical photons
  - A fraction of the light source power
  - Several wavelengths combined into one entity
Photon emission

Given $\Phi$ Watt lightbulb.
Emit $N$ photons.
Each photon has the power $\frac{\Phi}{N}$ Watt.

- Photon power depends on the number of emitted photons. Not on the number of photons in the photon map.
Diffuse point light

Generate random direction
Emit photon in that direction

// Find random direction
do {
    x = 2.0*random()-1.0;
    y = 2.0*random()-1.0;
    z = 2.0*random()-1.0;
} while ( (x*x + y*y + z*z) > 1.0 );
Example: Diffuse square light

- Generate random position $p$ on square
- Generate diffuse direction $d$
- Emit photon from $p$ in direction $d$

// Generate diffuse direction
$u = \text{random}();$
$v = 2\pi \text{random}();$
$d = \text{vector}( \cos(v)\sqrt{u}, \sin(v)\sqrt{u}, \sqrt{1-u} );$
Projection maps
Projection maps
Surface interactions

The photon is

- Stored (at diffuse surfaces) and
- Absorbed ($A$) or
- Reflected ($R$) or
- Transmitted ($T$)

$$A + R + T = 1.0$$
struct photon {
    float x,y,z;       // position
    char p[4];        // power packed as 4 bytes
    char phi,theta;   // incident direction
    short flag;       // flag used for kd-tree
}

Memory overhead: 20 bytes/photon.
Photon scattering

The simple way:

Given incoming photon with power $\Phi_p$

Reflect photon with the power $R \times \Phi_p$

Transmit photon with the power $T \times \Phi_p$
The simple way:

Given incoming photon with power $\Phi_p$

Reflect photon with the power $R \ast \Phi_p$

Transmit photon with the power $T \ast \Phi_p$

- Risk: Too many low-powered photons - wasteful!

- When do we stop (systematic bias)?

- Photons with similar power is a good thing.
Russian Roulette

- Statistical technique
- Known from Monte Carlo particle physics
- Introduced to graphics by Arvo and Kirk in 1990
Russian Roulette

Probability of termination: $p$
Russian Roulette

Probability of termination: $p$

$E\{X\}$
Russian Roulette

Probability of termination: $p$

$$E\{X\} = p \cdot 0$$
Russian Roulette

Probability of termination: \( p \)

\[
E\{X\} = p \cdot 0 + (1 - p)
\]
Russian Roulette

Probability of termination: \( p \)

\[
E\{X\} = p \cdot 0 + (1 - p) \cdot \frac{E\{X\}}{1 - p}
\]
Russian Roulette

Probability of termination: $p$

$$E\{X\} = p \cdot 0 + (1 - p) \cdot \frac{E\{X\}}{1 - p} = E\{X\}$$
Russian Roulette

Probability of termination: $p$

$$E\{X\} = p \cdot 0 + (1 - p) \cdot \frac{E\{X\}}{1 - p} = E\{X\}$$

Terminate un-important photons and still get the correct result.
Russian Roulette Example

Surface reflectance: $R = 0.5$
Incoming photon: $\Phi_p = 2 \, W$

$$r = \text{random}();$$
$$\text{if ( } r < 0.5 \text{ )}$$
  $$\quad \text{reflect photon with power } 2 \, W$$
$$\text{else}$$
  $$\quad \text{photon is absorbed}$$
Russian Roulette Intuition

Surface reflectance: $R = 0.5$
200 incoming photons with power: $\Phi_p = 2$ Watt

Reflect 100 photons with power 2 Watt instead of 200 photons with power 1 Watt.
Surface reflectance: $R = 0.2$
Surface transmittance: $T = 0.3$
Incoming photon: $\Phi_p = 2$ W

```c
r = random();
if ( r < 0.2 )
    reflect photon with power 2 W
else if ( r < 0.5 )
    transmit photon with power 2 W
else
    photon is absorbed
```
Russian Roulette

- Very important!
- Use to eliminate un-important photons
- Gives photons with similar power :)
Sampling a BRDF

\[ f_r(x, \vec{\omega}_i, \vec{\omega}_o) = w_1 f_{r,1}(x, \vec{\omega}_i, \vec{\omega}_o) + w_2 f_{r,2}(x, \vec{\omega}_i, \vec{\omega}_o) \]
Sampling a BRDF

\[ f_r(x, \vec{ω}_i, \vec{ω}_o) = w_1 \cdot f_{r,d} + w_2 \cdot f_{r,s} \]

\[ r = \text{random}() \cdot (w_1 + w_2); \]
\[ \text{if ( } r < w_1 \) } \]
\[ \quad \text{reflect diffuse photon} \]
\[ \text{else} \]
\[ \quad \text{reflect specular} \]
Specular Reflection

\[ \vec{d}_r = \vec{d}_i - 2\vec{n}(\vec{n} \cdot \vec{d}_i) \]
The photon map datastructure

Requirements:

- Compact (we want many photons)
- Fast insertion of photons
- Fast nearest neighbor search
The photon map datastructure

Requirements:

- Compact (we want many photons)
- Fast insertion of photons
- Fast nearest neighbor search

A left-balanced kd-tree
The kd-tree

- Introduced by Bentley in 1975
- A multidimensional "binary" tree
A left-balanced kd-tree

Post-process:

- No pointers (save 40% memory)
- Faster search

Recursively split photons along the median of the "largest" dimension.
Photon tracing

Overview:

While (we want more photons) {
    Emit a photon
    while (photon hits a surface) {
        Store photon
        Use Russian Roulette to scatter photon
    }
}

Build balanced kd-tree
Rendering

We want a Radiance value, $L$, per pixel.

The photon map stores flux/power.
We want a Radiance value, $L$, per pixel.

The photon map stores flux/power.

Radiance is the differential flux per differential solid angle per differential cross-sectional area:

$$L(x, \bar{\omega}) = \frac{d\Phi^2(x, \bar{\omega})}{d\omega \cos \theta \, dA}$$
We want a Radiance value, $L$, per pixel.

The photon map stores flux/power.

Radiance is the differential flux per differential solid angle per differential cross-sectional area:

$$L(x, \bar{\omega}) = \frac{d\Phi^2(x, \bar{\omega})}{d\omega \cos \theta \, dA}$$

How do we get a radiance estimate from the photon map?
Radiance estimate

\[ L(x, \vec{\omega}) = \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' \, d\omega' \]
Radiance estimate

\[ L(x, \bar{\omega}) = \int_{\Omega} f_r(x, \bar{\omega}', \bar{\omega}) L'(x, \bar{\omega}') \cos \theta' \, d\omega' \]

\[ = \int_{\Omega} f_r(x, \bar{\omega}', \bar{\omega}) \frac{d\Phi'^2(x, \bar{\omega}')}{d\omega'} \cos \theta' \, dA \, \cos \theta' \, d\omega' \]
Radiance estimate

\[ L(x, \bar{\omega}) = \int_{\Omega} f_r(x, \bar{\omega}', \bar{\omega}) L'(x, \bar{\omega}') \cos \theta' \, d\omega' \]

\[ = \int_{\Omega} f_r(x, \bar{\omega}', \bar{\omega}) \frac{d\Phi'^2(x, \bar{\omega}')}{d\omega'} \cos \theta' \, dA \cos \theta' \, d\omega' \]

\[ = \int_{\Omega} f_r(x, \bar{\omega}', \bar{\omega}) \frac{d\Phi'^2(x, \bar{\omega}')}{dA} \]
\( L(x, \vec{\omega}) = \int_\Omega f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' \, d\omega' \)

\( = \int_\Omega f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi'^2(x, \vec{\omega}')}{d\omega'} \cos \theta' \, dA \cos \theta' \, d\omega' \)

\( = \int_\Omega f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi'^2(x, \vec{\omega}')}{dA} \, dA \)

\( \approx \sum_{p=1}^n f_r(x, \vec{\omega}'_p, \vec{\omega}) \frac{\Delta \Phi_p(x, \vec{\omega}'_p)}{\Delta A} \)
Radiance estimate
The radiance estimate

Locate the $n$ nearest photons using kd-tree

\[
L(x, \vec{\omega}) \approx \sum_{p=1}^{n} f_r(x, \vec{\omega}_p, \vec{\omega}) \frac{\Delta \Phi_p(x, \vec{\omega}_p)}{\pi r^2}
\]
The radiance estimate

Locate the $n$ nearest photons using kd-tree

$$L(x, \bar{\omega}) \approx \sum_{p=1}^{n} f_r(x, \bar{\omega}'_p, \bar{\omega}) \frac{\Delta \Phi_p(x, \bar{\omega}'_p)}{\pi r^2}$$

Where the surface is locally flat this is a consistent estimator:
Converges to the correct result :)
Disc filtering
Disc filtering
Filtering

Weight nearby photons higher.
Filtering

Weight nearby photons higher.

Cone filtering:

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

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

Also Gaussian filtering
Caustic from a glass sphere

30000 photons / 50 photons in radiance estimate
Metalring caustic
Caustic on a glossy surface

\[340000 \text{ photons} / \approx 100 \text{ photons in radiance estimate}\]
Are we done?
Are we done?

Works great for caustics :)
Are we done?

Works great for caustics :)

But what about other global illumination effects such as color bleeding?
Direct visualization of the radiance estimation with a significant disparity in photon counts. 200,000 photons vs 50 photons in the radiance estimate.
Direct visualization of the radiance estimate

200000 photons / 500 photons in radiance estimate
A practical two-pass method

Observations:

- Caustics are difficult to sample (from the eye)
- Indirect illumination requires many photons
Two photon maps

global photon map

caustics photon map
The Rendering Equation

\[ L_r(x, \bar{\omega}) = \int_{\Omega_x} f_r(x, \bar{\omega}', \bar{\omega}) L_i(x, \bar{\omega}') \cos \theta' \, d\omega' \]
The Rendering Equation

\[ L_r(x, \omega) = \int_{\Omega_x} f_r(x, \omega', \bar{\omega}) L_i(x, \bar{\omega}') \cos \theta' \, d\omega' \]

Split incoming radiance:

\[ L_i = \underbrace{L_{i,l}}_{\text{direct}} + \underbrace{L_{i,c}}_{\text{caustics}} + \underbrace{L_{i,d}}_{\text{soft indirect}} \]
The Rendering Equation

\[ L_r(x, \vec{\omega}) = \int_{\Omega_x} f_r(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') \cos \theta' \, d\omega' \]

Split incoming radiance:

\[ L_i = L_{i,l} + L_{i,c} + L_{i,d} \]

direct \hspace{1cm} caustics \hspace{1cm} soft indirect

Split the BRDF:

\[ f_r = f_{r,d} + f_{r,s} \]

diffuse \hspace{1cm} specular
The Rendering Equation

\[ L_r = \int_{\Omega_x} f_r L_i \cos \theta' \, d\omega' \]
The Rendering Equation

\[ L_r = \int_{\Omega_x} f_r L_i \cos \theta' \, d\omega' \]

\[ = \int_{\Omega_x} f_r L_l \cos \theta' \, d\omega' + \text{direct} \]
The Rendering Equation

\[ L_r = \int_{\Omega_x} f_r \, L_i \, \cos \theta' \, d\omega' \]

\[ = \int_{\Omega_x} f_r \, L_l \, \cos \theta' \, d\omega' + \text{direct} \]

\[ + \int_{\Omega_x} f_{r,s} \, (L_{i,c} + L_d) \, \cos \theta' \, d\omega' + \text{specular} \]
The Rendering Equation

\[ L_r = \int_{\Omega_x} f_r L_i \cos \theta' \, d\omega' = \int_{\Omega_x} f_r L_l \cos \theta' \, d\omega' + \text{direct} \]
\[ \int_{\Omega_x} f_r, s \left( L_{i,c} + L_d \right) \cos \theta' \, d\omega' + \text{specular} \]
\[ \int_{\Omega_x} f_r, d L_c \cos \theta' \, d\omega' + \text{caustics} \]
The Rendering Equation

\[ L_r = \int_{\Omega_x} f_r L_i \cos \theta' \, d\omega' \]

\[ = \int_{\Omega_x} f_r L_l \cos \theta' \, d\omega' + \text{direct} \]

\[ + \int_{\Omega_x} f_{r,s} (L_{i,c} + L_d) \cos \theta' \, d\omega' + \text{specular} \]

\[ + \int_{\Omega_x} f_{r,d} L_c \cos \theta' \, d\omega' + \text{caustics} \]

\[ + \int_{\Omega_x} f_{r,d} L_d \cos \theta' \, d\omega' + \text{soft indirect} \]
Rendering
Rendering: direct illumination
Rendering: specular reflection
Rendering: caustics
Rendering: indirect illumination
No Importance Sampling
Importance Sampling

(Using the 50 nearest photons)
Lightning