Photon Mapping

• Photon mapping is a two pass global illumination algorithm developed by Henrik Jensen
• First pass: photon tracing
  • Emitting discrete photons from the light sources and tracing them through the scene
  • The photons are represented in a spatial data structure called the photon map
• Second pass: photon rendering
  • Render rays from the eye
  • Use photon map to calculate the effects of indirect lighting and caustics

Photon

• Photon is the elementary particle responsible for electromagnetic phenomena
• Has zero invariant mass and travels at the constant speed
• A light ray is a stream of photons
• Our eyes act as a photon detector
• When a group of N photons of wavelength $\lambda$ hit the retina, we observe a light ray:
  • Brightness is determined by $N$
    $b = \log N$
  • Color is controlled by wavelength $\lambda$

Colored Photon

• Tracking photons with different wavelength can be computationally expensive
• The reflection & transmission behaviors of photons are often independent of wavelength
  • Not true for a glass prism
• Encode a group of photons as a single “colored” photon can save computational time
  • Position : $P$
  • Incident direction : $D$
  • Color (power at different wavelength) : $C$

Photon Emission

• Photons are generated at light sources in the scene
• For each light source we create a set of photons and divide the overall power of the light source amongst them
  • Brighter lights emit more photons than dimmer lights
Photon Tracing
- Emitted photons are scattered through the scene
- Photons are eventually absorbed or lost
- Ray-object intersection calculation is used to determine where a given photon hits
  - If there is no intersection, then the photon is lost

Photon at Intersection
- When a photon hits a surface, the percentages of its energy being reflected, transmitted & absorbed are determined by the surface's material properties:
  - $k_r$ being reflected;
  - $k_t$ being transmitted;
  - The rest, $(1-k_r-k_t)$, being absorbed
- The distribution of the energy at intersection can be modeled by breaking up the photon into smaller ones
  - Generates many new photons, which are costly for computation & storage
  - Makes power of different photons varies dramatically

Russian Roulette
- In photon mapping, a Monte Carlo sampling technique called Russian Roulette is used instead
  - Does not break the photon or generate new ones
  - Use random numbers to probabilistically decide how to handle the photon
  - Generate a random number $R$, which is uniformly distributed within (0,1]
    - $R \leq k_r \rightarrow$ reflected
    - $k_r < R \leq k_r + k_t \rightarrow$ transmitted
    - $R > k_r + k_t \rightarrow$ absorbed

Photon Storing
- Photons are only stored where they hit diffuse (nonspecular) surfaces
  - Storing photons on specular surfaces does not give any useful information
  - For all other photon-surface interactions, data is stored in the photon map
    - Each emitted photon can be stored several times along its path
    - The position, incoming photon power, & incident direction are stored

Photon Rendering
- Ray tracing approach is used to handle specular or glossy surfaces & direct lighting
  - Effects of indirect lighting and caustics are achieved using the photons stored
    - First locate the nearest $n$ photons in the photon map
    - Then use the $n$ photos to estimate radiance
Balanced Kd-Tree

- To efficiently search nearest photons, the balanced kd-tree is used to represent the photon map.
- One of the few data structures that are ideal for handling non-uniform distributions of photons.
- The worst-case complexity of locating photons is $O(\log n)$.

Radiance Estimation

- The photon map can be seen as a representation of the incoming flux $\Phi$.
- The reflected radiance along a given direction $\omega_0$ can be calculated based on incoming flux using:
  \[
  L(x, \omega_0) = \int f(x, \omega_0 \rightarrow \omega_1) E(x, \omega_1) \cos \theta d\omega_1
  \]
  \[
  L(x, \omega_1) = \frac{d^2 \Phi(x, \omega_1)}{\cos \theta d\omega_1 dA}
  \]
  \[
  L(x, \omega_0) = \int f(x, \omega_0 \rightarrow \omega_1) \frac{d^2 \Phi(x, \omega_1)}{dA}
  \]

Pseudocode for Kd-Tree Construction

- Node createKdTree(points) {
  - Find the cube enclose all points;
  - Select dimension $d$, in which the cube is largest;
  - Find median of the points in dimension $d$;
  - $s_1$ = all points below median;
  - $s_2$ = all points above median;
  - $n$ = new Node(median);
  - $n$.left = createKdTree(s1);
  - $n$.right = createKdTree(s2);
  - return $n$;
}

Radiance Estimation (Cont’d)

- Assuming the nearest $n$ photons found are all intersect the surface at $x$, the integration can be approximated using:
  \[
  L(x, \omega_0) = \sum_{i=1}^{n} f(x, \omega_p \rightarrow \omega_1) \frac{\Delta \Phi(x, \omega_p)}{\Delta A}
  \]
- Assuming the surface is locally flat, $\Delta A$ can be approximated using the area of the projection circle of the sphere that encloses all $n$ photons:
  \[
  \Delta A = \pi \Delta r \Rightarrow L(x, \omega_0) = \frac{1}{\Delta A} \sum_{i=1}^{n} f(x, \omega_p \rightarrow \omega_1) \Delta \Phi(x, \omega_p)
  \]

Photon Mapping Result I

Photon Mapping Result II