Rendering Algorithms

Spring 2009
Matthias Zwicker
Universität Bern

Overview
• Irradiance caching
• Bidirectional path tracing

Indirect illumination
• Path tracing: slow
• Final gathering using photon maps: slow (though not quite as slow as path tracing)
• Radiance estimation using photon maps: inaccurate for reasonable number of photons
• Other ideas?

Indirect illumination

Global illumination
Indirect irradiance

Indirect illumination

Indirect Irradiation

Observation
• Indirect irradiance (irradiance due to indirect illumination) changes very smoothly
  - Except for caustics
Irradiance caching

- “A ray tracing solution for diffuse interreflection”, Ward, Rubinstein, Clear, SIGGRAPH 88
- One of the most popular strategies to accelerate global illumination
- Works well in practice, though artifacts may appear because of heuristics involved

Irradiance caching

Idea
- Assume diffuse surfaces
- Cache irradiance samples
  - Compute and store irradiance samples at sparse locations (using photon mapping or path tracing)
  - Interpolate cached samples to each pixel
  - Greedily compute new samples only if interpolation fails (use heuristics)
- Note: photons store “incident radiance“ instead of irradiance

Irradiance caching algorithm

Three components
- Irradiance computation
- Irradiance caching (sample point distribution)
- Irradiance extrapolation

- Similar to photon mapping, but all steps performed in main rendering pass
- Limitation: works only with diffuse surfaces (or diffuse component) since irradiance is stored

Irradiance computation

\[ E(x) = \int_{\Omega} L_i(x, \omega_i) \cos \theta_i d\omega_i \]
Irradiance computation

\[
E(x) = \int_{0}^{\pi} L(x, \omega_i) \cos \theta_i d\omega_i \\
= \int_{t=1}^{T \cdot p} \int_{\phi}^{\psi} L(x, \theta_i, \phi) \cos \theta_i \sin \theta_i d\theta_i d\phi_i \\
\approx \pi \frac{1}{TP} \sum_{t=1}^{T} \sum_{p=1}^{P} f_{i}(\theta_i, \phi_i)
\]

- Stratified sampling of the hemisphere
- Subdivision \( T, P, t=1..P, p=1..P \)
- Uniform random random variables \( \xi, \psi \)
  \[
  \theta_i = \sin^{-1} \left( \sqrt{1 - \xi} \right), \quad \phi_i = 2\pi \frac{p-1}{P}
  \]

Irradiance computation

- Compute incident radiance \( L_i(x, \omega_i) \) using path tracing or photon gathering at first hit point of primary ray
- For good quality expect tracing 200 to 5000 paths
- Costly, but will do this only at few locations in image

Irradiance caching

I.e., irradiance sample distribution
- Assign a range for each sample, within which it can be used for interpolation
- When computing irradiance for primary ray
  - Look up if there is a cached sample within valid range, if so use it to extrapolate
  - If not, compute new sample

Irradiance caching

- Where irradiance changes quickly, valid range for cached samples should be small
- Where irradiance changes slowly, range should be large
- Rate of change of irradiance depends on distance to closest visible surfaces as seen from sample point

Irradiance caching

Harmonic mean heuristics
- The range is given by a radius
  \[
r_j = \frac{N}{\Sigma_{i=1}^{N} 1/d_i}
  \]
  where \( N \) is the number of paths, \( d_i \) is the distance to the first intersection along the path
- Average would weight infinite distances too heavily

Irradiance caching

Harmonic mean heuristics

Harmonic mean heuristics

Harmonic mean heuristics

Harmonic mean heuristics

Harmonic mean heuristics

Harmonic mean heuristics

Harmonic mean heuristics

Irradiance sample

```
struct IrradianceSample {
    vector3 E // Irradiance
    vector3 n // normal
    vector3 p // position
    float r // range
}
```
Irradiance extrapolation

- Need fast access to nearby samples similar as in photon mapping (radiance estimation)
- Store samples in spatial data structure
  - Octree
- Add sample to each cell that it overlaps
- Adaptively subdivide octree such that each cell has limited number of samples

Octree example

- Adaptive subdivision such that each cell contains <3 samples

Irradiance extrapolation

- Need to determine
  - Which samples should be used for extrapolation
  - Not all samples within valid range should be used
  - Weight for used samples

Error estimate

- Given a point on the surface and its normal and a cached location, estimate the difference of their irradiances
- Use error estimate to compute weight

Error estimate

\[ \epsilon_j(x) \leq \frac{\|x - x_j\|}{r_j} + \sqrt{2 - 2n(x) \cdot n(x_j)} \]

Algorithm

\[
W = 0 \\
\text{for ( all irradiance samples } j \text{ in octree cell overlapping with } x \text{ ) } \{ \\
\quad \text{compute weight } w_j \\
\quad \text{if ( <sample is valid> ) } \{ \\
\quad\quad W += w_j; \text{ WE } += w_j \cdot E[j] \\
\quad \} \\
\} \\
\text{if ( } W > 0 \text{ ) } \{ \\
\quad \text{return } \text{WE}/W \\
\} \text{ else } \{ \\
\quad \text{return}( \text{ compute new irradiance sample } ) \\
\} 
\]
Algorithm

\[
\text{<sample is valid> = dist( x - i ) < r[j] \ (within range) \ \&\ & w[j] > 1/a \ (sufficient weight) \ \&\ & \text{dot( x[j] - x, n(x) ) < 0 \ (x[j] is behind x)}
\]

Photon mapping and irradiance caching

- Final gathering often used to compute irradiance samples
  - Advanced: use photon map for importance sampling when tracing gather rays
- Caustics break assumptions of irradiance caching
  - Exclude caustic paths from irradiance sample computation
  - Use photon map to render caustics

Non-diffuse surfaces

Approximation

\[
I_{\omega}(x, \omega_i) = \int_{\omega_o} f(x, \omega_o, \omega_i) I_{\omega_o}(x, \omega_i) \cos(\theta) du_i
\]

\[
\approx \left( \int_{\omega_o} f(x, \omega_o, \omega_i) du_i \right) \left( \int_{\omega_o} I_{\omega_o}(x, \omega_i) \cos(\theta) du_i \right)
\]

\[
= \frac{1}{2} \rho_{\text{diff}}(\omega_i) E(x)
\]

Hemispherical directional reflectance \(\rho_{\text{diff}}(\omega_i)\)

Examples

- 1000 sample rays

Examples

- 5000 sample rays

Examples

- Same computation time

Irradiance caching

Path tracing
Overview

- Irradiance caching
- Bidirectional path tracing

Review: global illumination

\[ \int_{\text{Length 3}} + \int_{\text{Length 4}} + \ldots + \int_{\text{Length 7}} \]


Path sampling strategies

- Path tracing
  - Construct paths starting from the eye
- Photon mapping
  - Construct paths from the lights
  - Cache contributions using photons
  - Construct paths from the eye
  - Look up contributions of light paths in photon cache (photon map)

Bidirectional path tracing

- Construct paths from eye and light simultaneously
- Connect paths by joining pairs of vertices from eye and light paths
- How to weight the contribution of each connected path?
- Reference
  Eric Veach PhD thesis, chapter 10

Implementation

- Construct paths from eye and light segment by segment
  - Choose random direction for each new segment
- Store \( \alpha \) value at each path vertex
  - For each new vertex, multiply previous \( \alpha \) value with BRDF, cosine of new direction to normal, and \( 1/\text{pdf} \) for sampled direction
- Initial \( \alpha \) values
  - Light path: first vertex has \( \alpha = L / (\text{area of light}) \) (assuming isotropic light with radiance \( L \), uniform sampling)
  - Eye path: \( \alpha = 1 \)

Implementation

- When connecting, multiply alphas with BRDFs at connection points and with geometry term of connection points
- For each path, divide contribution by number of paths of this length that were sampled
  - Called „average“ or „uniform“ weighting
Implementation

- Use Russian roulette to terminate both eye and light paths
  - Modify path probabilities accordingly, as in regular path tracing
- If light sources are regular surfaces in scene that can be intersected
  - Eye path may hit light source
  - Include paths with s=0 (zero vertices on light path)
  - Combined paths with s>0: treat potential eye vertex on the light source as regular surface with some BRDF

Multiple importance sampling

- Instead of uniformly weighting paths, try to come up with optimal weighting scheme
- See “Optimally combining sampling techniques for Monte Carlo rendering”, Veach et al.
  - Balance heuristics: weight of path = (pdf of path)/(sum of pdf’s of paths of same length)