Point-Based Global Illumination for Movie Production

Per Christensen
Pixar Animation Studios

SIGGRAPH 2010 Course
Overview

• Point-based global illumination
  – generating direct illumination point cloud
  – rendering GI using point cloud

• Examples of use in movies

• Variations and extensions

• What’s next?
Related work

- Method is inspired by Bunnell’s point-based GPU method
- Related to clustering radiosity and point-based subsurface scattering
Point-based global illumination

- Fast, low memory, no noise
- Handles complex geometry (including dense polygon meshes, hair, leaves, displacement), many light sources, complex surface shaders, ...
- Movie-production friendly
- Part of Pixar’s RenderMan renderer
Point-based global illumination

• Three steps:

• Generate point cloud of directly illuminated surface colors (radiosity)

• Organize points into octree; larger points and spherical harmonics

• Render: compute diffuse/glossy global illumination at each shading point
A point cloud

- Each point: position, normal, radius, color
  = a colored disk

- Terminology: “point” or “disk” or “surfel”?
Generate point cloud

- Render direct illumination image
- Generate point cloud file at same time

rendered image  point cloud, 560K points (various views)
Generate point cloud

- Point cloud files from “Up”
Organize points into octree

- Organize points into octree
- Each cluster of points is represented by a larger point or a spherical harmonic representation of directional light distribution
Compute global illum at a point

- Basic idea: add up color from all other points!
Compute global illum at a point

- For efficiency: use cluster of points for distant points
- For higher accuracy: ray trace close points
Compute global illum at a point

- Problem: if all points are added up, even points “hidden” behind other points will contribute
Compute global illum at a point

- Solution: rasterize colors contributing to a point -- world “as seen” by that point

- Raster cube examples:
  
  - point on ceiling
  - point on teapot lid
Compute global illum at a point

- Multiply all raster pixel colors by reflectance function (BRDF); add
- Result is diffuse / glossy reflection at point
Global illumination result

direct illum (9 sec)

direct illum + diffuse GI + glossy GI (21 sec)
Use in movies

- Implemented in Pixar’s RenderMan
- Integrated into lighting pipeline at ILM, Pixar, Disney, DNeg, MPC, ...
Use in movies

Sony: “Surf’s Up” ambient occlusion

“Surf’s Up” test (Courtesy of Rene Limberger, Sony)
ILM: Davy Jones

“Pirates of the Caribbean: Dead Man’s Chest”
(Courtesy of Industrial Light & Magic)
Disney: special effects on "Bolt"

(Courtesy of Dale Mayeda, Disney)
“Up” example without global illum
“Up” example with global illum
“Up” example without global illum
“Up” example with global illum
“Toy Story 3” examples
“Toy Story 3” examples
“Toy Story 3” examples
“Toy Story 3” examples
“Toy Story 3” examples
“Toy Story 3” examples
Variations and extensions

• Area light sources
• Environment illumination
• Multiple light bounces
• Final gather for photon maps
• Ambient/directional/reflection occlusion
• Volumes
Area light sources + soft shadows

- Treat area light sources the same as surfaces: generate point cloud with color data
- Light sources can have arbitrary shape and colors
- Also write (black) points for shadow-casting objects
Area light sources + soft shadows

area lights

area light illumination
Environment illumination -- IBL

• Use environment color for raster pixels not covered by points
Multiple light bounces

- Run the algorithm $n$ times
- (For efficiency: first $n-1$ times can be computed at fewer points)
Final gather for photon mapping

- Final gather step is usually done with ray tracing; slowest part of photon mapping
- Use point-based method instead
Final gather for photon mapping

direct illum

photon map

radiance est

pt-based GI
Special case: Ambient occlusion

- Fraction of hemisphere above a point that’s covered
- Similar to shadows on overcast day
- Values between 0 and 1
Ambient occlusion

- Generate point cloud with only position, normal, radius (no colors)
Ambient occlusion
Ambient occlusion (and reflections)
NEW: Image-based relighting

- In addition to ambient occlusion, also compute directional visibility: spherical harmonic coeffs. at each point
- Compute SH coeffs for environment map
- (Re-)rendering is just multiplying SH coefficients -- 9 or 25 mults/point. Fast!
NEW: Image-based relighting
Special case: reflection occlusion

- As ambient occlusion, but narrow cone of directions (around reflection direction)
Global illumination in volumes

- Points don’t have normals: spheres, not disks
- Illumination from all directions: entire raster cube
- Surface ↔ volume
- Volume ↔ volume
Global illumination in volumes

surface to volume

volume to volume
Optimization: interpolation

- If the color bleeding varies only a little in an area (<2%), we simply interpolate it.
- Technique known from ray tracing ("irradiance cache")
Optimization: interpolation

- Compute color bleeding at the 4 corners of surface patch
- Is the difference between 4 values small?
  - yes: interpolate on patch
  - no: split patch in 2; recurse
Parallel computation

- Global illumination at each point is independent
- Ideal for parallel execution
- Observed speedups:
  - 4 cores: \( \sim 3.6 \)
  - 8 cores: \( \sim 6.6 \)
More information

- M. Bunnell, “Dynamic ambient occlusion and indirect lighting”, GPU Gems 2
- P. Christensen, “Point-based approximate color bleeding”, Pixar tech memo #08-01
Summary

- Point-based diffuse and glossy global illumination is fast and can handle complex production scenes
- Also works for area lights, env. map illumination, multiple bounces, ambient occlusion, reflection occlusion, volumes
- In Pixar’s RenderMan
- Widely used in production
What’s next?

• “Up” and “Toy Story 3”: 1-bounce PBGI was used in addition to all the traditional lights

• Next:
  – reduce number of traditional lights?
  – multiple bounces?
What’s next?

- Implementation improvements:
  - improved accuracy in rasterization?
  - baking micropolygon grids?
  - GPU implementation?
Acknowledgments

- RenderMan team: Dana Batali, ...
- Mike Bunnell, Rene Limberger, Christophe Hery
- Pixar: Max P, P Sumo, JC, Stefan, Guido, ...
- Dale Mayeda (Disney), Philippe Leprince (DNeg), Anders Langlands (MPC), ...

Thanks!
Questions?