Point-Based Color Bleeding

Per Christensen
Pixar Animation Studios
June 2009
Overview

• What is color bleeding?

• Other computation methods

• Point-based color bleeding
  – generating direct illumination point cloud
  – rendering using point cloud

• Examples of use in movies

• Variations and extensions
Color bleeding

- Soft indirect illumination between matte surfaces
Computation methods

- Faking it: adding extra light sources
  - tedious; labor intensive

- Radiosity (finite elements)
  - requires entire scene geometry in memory

- Ray tracing
  - requires many rays + shader evaluations: slow

- Point-based
  - little memory, no shader evaluations
Computation methods

- **Faking it: adding extra light sources**
  - tedious; labor intensive

- **Radiosity (finite elements)**
  - requires entire scene geometry in memory

- **Ray tracing**
  - requires many rays + shader evaluations: slow

- **Point-based**
  - little memory, no shader evaluations
Point-based color bleeding

- Handles complex geometry (including dense polygon meshes, hair, leaves, displacement), many light sources, complex surface shaders, ...
- Very movie-production friendly
- Part of Pixar’s RenderMan renderer
Point-based color bleeding

- Two steps:
  - Generate point cloud of directly illuminated surface colors (radiosity)
  - Render: compute color bleeding at each shading point
A point cloud

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

- Terminology: “point” or “disk”?
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”
Compute color bleeding at a point

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

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

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

• Multiply all raster pixel colors by reflectance function (BRDF); add

• Result is color bleeding at point
Color bleeding results

direct illum

direct illum + color bleeding
Use in movies

- Pirates of the Carribean 2 & 3, Eragon, Surf’s Up, Spiderman 3, Harry Potter 5 & 6, Chronicles of Narnia, Fred Claus, Beowulf, Spiderwick Chronicles, Ironman, Indiana Jones, 10,000 BC, Batman: Dark Knight, Quantum of Solace, Cloverfield, Doomsday, Hellboy 2, Inkheart, Wall-E, Star Trek, Terminator 4, The Boat that Rocked, Fast & Furious 4, Angels and Demons, Up, ...
Davy Jones

“Pirates of the Caribbean: Dead Man’s Chest”
(Courtesy of Industrial Light & Magic)
“Up” example without color bleeding
“Up” example with color bleeding
“Up” example without color bleeding
“Up” example with color bleeding
“Up” example without color bleeding
“Up” example with color bleeding
Variations and extensions

- Glossy reflection
- Area light sources
- Environment illumination
- Multiple light bounces
- Ambient occlusion, reflection occlusion
- Volumes
Glossy reflection

- Only collect illumination from within a small cone of directions
- Raster cube example:

- Multiply raster pixel colors by glossy reflectance function (BRDF)
Glossy reflection

narrow glossy reflection  wide glossy reflection
Glossy reflection

point cloud

glossy reflection
Area light sources

- 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

Area lights

area light illumination
Environment illumination

- Use environment color for raster pixels not covered by points

HDRI env map

raster cube
Multiple light bounces

- Run the algorithm \( n \) times

- (For efficiency: first \( n-1 \) times can be computed at fewer points)
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)
Ambient occlusion

“Surf’s Up” test (Courtesy of Sony Imageworks)
Special case: reflection occlusion

- As ambient occlusion, but narrow cone of directions (around reflection direction)

narrow reflection

wider reflection
Other result types

• Given the raster cube it is also fast to compute:
  – average unoccluded direction ("bent normal")
  – average illumination direction
Color bleeding in volumes

- Points don’t have normals: spheres, not disks
- Color bleeding from all directions: entire raster cube
- surface $\leftrightarrow$ volume
- volume $\leftrightarrow$ 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

- Color bleeding at each point is independent
- Ideal for parallel execution
- Observed speedups:
  - 4 cores: 3.6
  - 8 cores: 6.6
Summary

• Point-based color bleeding is fast and can handle complex production scenes

• Also works for glossy reflection, area lights, env. map illumination, multiple bounces, ambient occlusion, reflection occlusion, volumes

• In Pixar’s RenderMan

• Is gaining widespread use in production
More information

- “Point-Based Graphics” book by Gross & Pfister

- Pixar technical report #08-01: “Point-based approximate color bleeding”

- Talk this afternoon: Making of “Partly Cloudy” and “Up”
Acknowledgments

- RenderMan team, Jean-Claude Kalache
- Rene Broca, Cedric Guiard, Marine Lamblin
- You for listening

Thanks!
Questions?