Parallel Irradiance Caching on the GPU

Nathaniel Jones
18.337/6.338 Parallel Computing
December 2, 2013
Radiance

- Architectural lighting and daylighting simulation
- Backward ray tracing tool
- Simulation engine used by
  - IES<VE>
  - Ecotect
  - OpenStudio
  - DIVA
  - DAYSIM
- Open source
- Written in the 1980’s

http://windows.lbl.gov/comm_perf/nyt_visualizing.html
Radiance

Physically Accurate Ray Paths
Physically Accurate Material Properties
Physically Accurate Geometry
Physiologically Accurate Tone Mapping

Photo-realistic Rendering
Radiance

- Radiance is slow
- Speed affects use
  - Slow simulations occur after design is complete
  - Fast simulations can be repeated as part of the design process
- Low-quality simulations are faster

138,844,405 Rays
1.43 Hours

41,010,721 Rays
3.98 Minutes
Phong Reflection Model

Specular

Diffuse

Combination

Backward Ray Tracing
Replace the Engine

OptiX™

• Free ray tracing engine from NVIDIA®
• Built on top of CUDA™
• Provides
  • Ray traversal using BVH or KD trees
  • User-defined shader programs for ray generation, intersection testing, closest hit, any hit, and miss
  • Interop with OpenGL, Direct3D, and CUDA
• Limitations
  • No `syncthreads()`
  • No `cudamalloc()`
Backward Ray Tracing with OptiX™
Irradiance Caching

Irradiance Caching

• CPU cluster
  • Network file locks [Larson and Shakespeare, 1998]
  • MPI [Koholka et al., 1999][Debattista et al., 2006]
  • Wait-free cache [Dubla et al., 2009]

• GPU
  • Photon mapping [Wang et al., 2009]
  • Splatting [Křivánek and Gautron, 2009]
  • Multi-pass [Frolov et al., 2012]
Irradiance Caching with OptiX™
Irradiance Caching with OptiX™
Results

No caching

Creating cache

Reading cache

OptiX

27 seconds

11 seconds

56 seconds

Radiance

243 seconds

4838 seconds

6 seconds
To Do...

- Adaptive cache sampling density
To Do...

- Adaptive cache sampling density
- Missing radiance problem
  - Reduce ray extinction rate
  - Cache 2\textsuperscript{nd} and 3\textsuperscript{rd} bounces
- Improve timings
  - Code optimization
  - Reduce memory transfer
  - Better hardware
Questions?

Nathaniel Jones <nljones@mit.edu>