An Evaluation of Existing BVH Traversal Algorithms for Efficient Multi-Hit Ray Tracing

Jefferson Amstutz (SURVICE)
Johannes Guenther (Intel)
Ingo Wald (Intel)
Christiaan Gribble (SURVICE)
The Multi-Hit Problem

• Find more than one (and possibly all) hits along the ray [Gribble et al. 2014]
  – Ray t-distance sorted order
  – No ray re-launch
Overlapping Geometry
Overlapping Geometry
Bounding Volume Hierarchies

• BVHs have been one of the premier structures for accelerating rays
  – Fast builds
  – Fast traversal

• Node overlaps create issues for doing multi-hit ray tracing
Spatial vs. Object Partitioning
Existing BVH Traversal solutions

• Many state-of-the-art implementations for both CPUs and GPUs
  – Embree (Intel x86, MIC)
  – Optix (NVIDIA GPUs)

• Intersection callbacks
  – Client application injects logic to “accept” or “reject” individual hit points
Traversal Techniques

• Must be all-hit ray traversal
  – Cannot *guarantee* first N hits with first/any hit trav.

• Sorting hit points
  – Sort during ray traversal (progressive)
  – Sort after hit points gathered (post traversal)

• Hit point data layout
  – Array of Structures (AoS)
    • Scalar sorting
  – Structure of Arrays (SoA)
    • Vectorized sorting
Results - SIMD Sorting Efficiency Improvement

- sibe
- fair
- conf
- truck
- tank
- hball
- sanm
- pplant

4-wide (SSE)  8-wide (AVX)  16-wide (Xeon Phi)
Results - hball
Results - plant
Results - truck
Future Work

• Memory allocation strategies for variable hit point buffers
  – Caching strategies for performance
  – Reduce/prevent hit point memory waste

• Multi-hit traversal algorithms for BVHs