Realizing Multi-Hit Ray Tracing in Embree & OSPRay

Christiaan Gribble
SURVICE Engineering

Intel HPC Developer Conference
12 November 2016
Take-home messages

• Multi-hit ray traversal
  Enables a new class of ray-based rendering & simulation applications
• Embree permits efficient implementation
• OSPRay permits easy integration
Take-home messages

- Multi-hit ray traversal
- Embree permits efficient implementation
  Intersection filter functions enable user-level implementation of state-of-the-art multi-hit ray traversal techniques
- OSPRay permits easy integration
Take-home messages

• Multi-hit ray traversal
• Embree permits efficient implementation
• OSPRay permits easy integration

Supports scalable, high performance visual analysis tools across optical & non-optical domains
Acknowledgements

- SURVICE
  - Joe Rosenthal
  - Mark Butkiewicz
- Intel
  - Jeff Amstutz
  - Ingo Wald
  - Jim Jeffers
Overview
Interval computation

Optical rendering
Interval computation

Optical rendering
Interval computation

Optical rendering

Non-optical rendering

C. Gribble, Realizing Multi-Hit Ray Tracing in Embree & OSPRay
Interval computation

Optical rendering

Non-optical rendering

C. Gribble, Realizing Multi-Hit Ray Tracing in Embree & OSPRay
Interval computation

Non-optical rendering
Interval computation

Non-optical rendering
Interval computation

Non-optical rendering

Interval computation
Interval computation

- Difficult or impossible
  - Epsilon hacks
  - Missed/repeated intersections
- Performance impacts
Interval computation

- Difficult or impossible
- Performance impacts
  - Traversal restart
  - Operational overhead
Interval computation

- Difficult or impossible
- Performance impacts

Is there a better solution?
Multi-hit ray traversal

- Which primitives are intersected?
  - One or more, & possibly all
  - Ordered by $t$-value along ray

- Possible applications
  - GNK14
  - AGGW15
  - Gri16, GWA16
Multi-hit ray traversal

- Which primitives are intersected?
- Possible applications
  - Transparent rendering
  - Alpha textures
- GNK14
- AGGW15
- Gri16, GWA16
Multi-hit ray traversal

• Which primitives are intersected?
• Possible applications
• GNK14
  • Spatial partitioning
  • Two algorithms: naive, buffered
• AGGW15
• Gri16, GWA16
Multi-hit ray traversal

- Which primitives are intersected?
- Possible applications
- GNK14
- AGGW15
  - Object partitioning
  - User-level implementation mechanisms
- Gri16, GWA16
Multi-hit ray traversal

• Which primitives are intersected?
• Possible applications
• GNK14
• AGGW15
• Gri16, GWA16
  • Enable early-exit in BVH
  • Implement in Embree & OSPRay
Implementation
Mechanisms

- Direct implementation
  - Kernels specific to multi-hit
  - Runs counter to our goal
- Intersection callbacks
- Traversal callbacks
- Reference implementation
Mechanisms

• Direct implementation
• Intersection callbacks
  • Invoked on valid ray/primitive intersection
  • User accepts/rejects hit
• Traversal callbacks
• Reference implementation
Mechanisms

- Direct implementation
- Intersection callbacks
- Traversal callbacks
  - Invoked on ray/node interaction
  - Two variants: every-node, leaf-node
- Reference implementation
Mechanisms

- Direct implementation
- Intersection callbacks
- Traversal callbacks

- Reference implementation
  - Supports callback mechanisms
  - Opts for clarity & simplicity

http://www.rtvtk.org/~cgribble/research/mhBVH/
Embree implementation

- Intersection filters
  - Compatible with mainline developments
  - v2.10.0+ – enables node-culling
- Assumptions
- Full source code available
Embree implementation

- Intersection filters
- Assumptions
  - \textit{Nquery} known \textit{a priori}
  - Preallocated hit data buffer
  - Initial value beyond range
- Full source code available
Embree implementation

- Intersection filters
- Assumptions
- Full source code available
  - Apache License, v2.0
  - Public *git* repository

http://www.rtvtk.org/~cgribble/research/ospMultiHit/
static void collectIntersectionsFilter(void* /* unused */, RTCRay& _ray)
{
    // Find index at which to store candidate intersection
    uint idx = Nquery;
    while (idx > 0 && _ray.tfar < hits[idx-1].tval)
    {
        hits[idx] = hits[idx-1];
        --idx;
    }
}
Scalar implementation

// Store intersection, possibly beyond index of the
// N ≤ Nquery closest intersections (i.e., at
// idx = Nquery)
HitData& hit = hits[idx];
hit.geomID = ray.geomID;
hit.primID = ray.primID;
hit.tval = ray.tfar;
hit.Ng = ray.Ng;

// Update number of intersections identified so far
ray.nhits += (ray.nhits < Nquery ? 1 : 0);
Scalar implementation

```c
17     if (ray.nhits < Nquery)
18     {
19       // Reject intersection to continue traversal with
20       // incoming ray interval, as in previous work
21       // [Amstutz et al. 2015]
22       ray.geomID = RTC_INVALID_GEOMETRY_ID;
23       return;
24     }
```
Scalar implementation

```c
// Induce node culling
// Trick: set ray.tfar to farthest value among the
//        N = Nquery intersections identified so far
//        and (implicitly) accept intersection with
//        modified ray interval
ray.tfar = hits[Nquery-1].tval;
```
Results
Performance – tests

• Find-first-intersection
  • First-hit v. multi-hit variants
  • Isolates multi-hit overhead

• Find-all-intersections

• Find-some-intersections
Performance – tests

• Find-first-intersection
• Find-all-intersections
  • Naive v. node culling multi-hit
  • Bounds performance expectations
• Find-some-intersections
Performance – tests

• Find-first-intersection
• Find-all-intersections
• Find-some-intersections
  • Naive v. node culling multi-hit
  • Demonstrates in situ performance
Performance – scenes

- sibe: 80K tris
- fair: 174K tris
- conf: 282K tris
- truck: 426K tris
- tank: 1.0M tris
- hball: 2.8M tris
- sanm: 10.5M tris
- plant: 12.7M tris
Performance – *truck* scene

![Performance Chart](image)

C. Gribble, Realizing Multi-Hit Ray Tracing in Embree & OSPRay
Performance – *truck* scene

C. Gribble, Realizing Multi-Hit Ray Tracing in Embree & OSPRay
Performance – *truck* scene

C. Gribble, Realizing Multi-Hit Ray Tracing in Embree & OSPRay
Limitations

- Number of hits specified *a priori*
  - Too few – incorrect
  - Too many – wasteful
  - C++ templates?
- Ordered BVH traversal [WAB16]
- Spatial partitioning [GNK14]
Limitations

• Number of hits specified *a priori*
• Ordered BVH traversal [WAB16]
  • Enables early-exit
  • No dynamic allocation
  • Requires new traversal kernels
• Spatial partitioning [GNK14]
Limitations

• Number of hits specified \textit{a priori}
• Ordered BVH traversal [WAB16]
• Spatial partitioning [GNK14]
  • Enables early-exit
  • No dynamic allocation
  • Requires user-defined geometry
Demonstrations

Engineering CAD visualization

Vulnerability analysis

C. Gribble, Realizing Multi-Hit Ray Tracing in Embree & OSPRay
Wrap-up
Take-home messages

• Multi-hit ray traversal
  Enables a new class of ray-based rendering & simulation applications
  • Embree permits efficient implementation
  • OSPRay permits easy integration
Take-home messages

• Multi-hit ray traversal
• Embree permits efficient implementation
  Intersection filter functions enable user-level implementation of state-of-the-art multi-hit ray traversal techniques
• OSPRay permits easy integration
Take-home messages

• Multi-hit ray traversal
• Embree permits efficient implementation
• OSPRay permits easy integration

Supports scalable, high performance visual analysis tools across optical & non-optical applications
### Key references

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>
Contact information

Address
- Applied Technology Operation
- SURVICE Engineering
- 4695 Millennium Drive
- Belcamp, MD 21017

E-mail
- christiaan.gribble@survice.com

Web