GPU Ray Tracing with Rayforce

Christiaan Gribble* Alexis Naveros
Applied Technology Operation
SURVICE Engineering

(a) *first-hit* (b) *any-hit* (c) *multi-hit*

Figure 1: GPU ray tracing with Rayforce. *The Rayforce ray tracing engine is designed for massively parallel computing architectures and leverages a novel graph-based spatial indexing structure that enables high performance* first-hit*,* any-hit*, and* multi-hit *traversal algorithms required to solve a variety of problems in physics-based simulation domains, including traditional image synthesis.*

1 Overview

Rayforce is a high performance ray tracing engine designed for massively parallel computing architectures, including manycore GPUs. Rayforce leverages a novel graph-based acceleration structure that permits *first-hit*, *any-hit*, and *multi-hit* traversal algorithms required to solve a variety of problems in physics-based simulation domains. Rayforce exposes core ray tracing operations via a programmable interface to enable the implementation of various computer graphics and scientific computing applications.

2 Graph-based Spatial Indexing

A new graph-based spatial indexing structure accelerates ray/primitive intersection operations. The structure is:

- *efficient*: in-memory data layouts are carefully designed to minimize storage, thereby improving cache performance;
- *flexible*: several traversal algorithms can be implemented with low overhead; and,
- *scalable*: complex scenes are handled efficiently, as performance depends only on geometric complexity along a ray.

The structure is comprised of *sectors*, which bound geometry, and *nodes*, or separation planes that disambiguate traversal steps. Construction proceeds by building a graph of sectors and nodes that minimizes a function representing traversal cost.

Ray origins are resolved to starting sectors using displacements from other origins or a 3D bin-based lookup. Rays enter a sector and intersect all bounded primitives, then traverse to linked sectors if necessary. Traversal algorithms include:

- *first-hit*: returns the nearest intersection (if any); used for visibility computations and visual effects such as reflection, refraction, and other forms of indirect illumination.
- *any-hit*: indicates whether any intersection occurs within a specific interval; used for shadow or ambient occlusion rays.
- *multi-hit*: returns one or more, and possibly all, intersections in ray-order (if any); used for transparency or operations in non-optical simulation domains.

The images in Figures 1 and 2 illustrate these traversal algorithms.

3 Performance

The data in Figure 3 show initial performance measurements for several rendering scenarios:

- **vis**: *first-hit* visibility with simple $N \cdot V$ shading.
- **x-ray**: all *multi-hit* intersections with simple alpha-blending.
- **ao**: *first-hit* visibility plus 32 *any-hit* ambient occlusion rays.
- **kajiya**: *first-hit* visibility, *any-hit* shadows, and two *first-hit* diffuse interreflection rays.

*e-mail: christiaan.gribble@survice.com

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGGRAPH 2013, July 21 – 25, 2013, Anaheim, California.
2013 Copyright held by the Owner/Author.
ACM 978-1-4503-2261-4/13/07

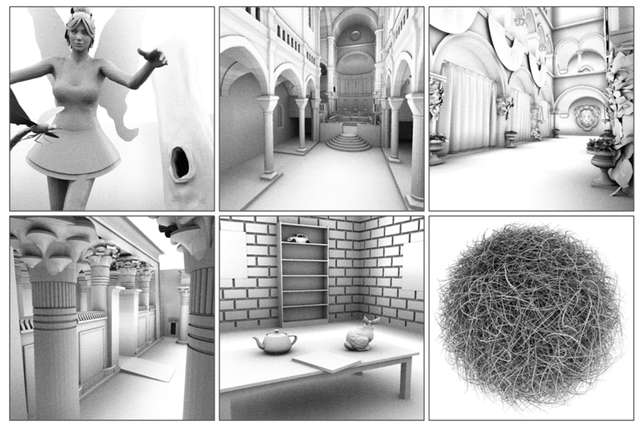

Figure 2: Image synthesis with Rayforce. *Using a new graph-based acceleration structure, Rayforce implements several traversal algorithms. Here,* first-hit *and* any-hit *traversal combine to render scenes common to computer graphics applications with ambient occlusion at highly interactive rates on a single GPU.*

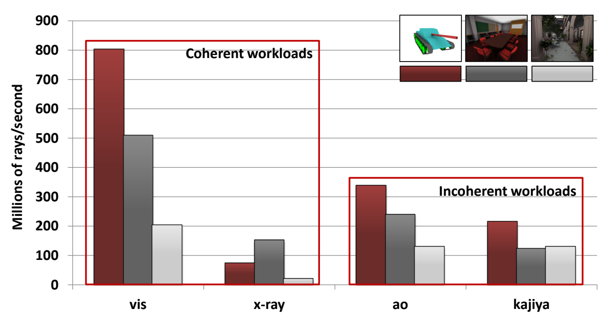

Figure 3: Rendering performance. *Initial measurements using three scenes rendered at* 1024×768 *pixels on an NVIDIA GeForce GTX 690 indicate that Rayforce delivers exceptional performance.*

Except for image display, all per-frame overhead—GPU kernel launch, ray generation/traversal, shading, host/device synchronization, and so forth—is included in these measurements.

4 Future Work

We are exploring additional memory optimizations for our acceleration structure and plan to implement a parallel GPU construction algorithm. We are also considering new methods for exploiting ray coherence to eliminate unnecessary thread divergence and reduce memory bandwidth on SIMT architectures.

Acknowledgments

This work is funded in part by research grants from the US Office of Naval Research and the US Army Research Laboratory.