Overview

Rayforce is a high performance ray tracing engine designed for highly 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 rendering and physics-based simulation. Rayforce exposes core ray tracing operations via a programmable interface to enable the implementation of various computer graphics and scientific computing applications.

Graph-based spatial indexing

Rayforce uses a novel graph-based spatial indexing structure to accelerate ray/primitive intersection operations. The graph is:

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

The structure is comprised of two basic elements:

- **sectors**, which bound geometry; and,
- **nodes**, or separation planes used to disambiguate traversal steps.

Construction proceeds by building a graph of sectors and nodes that minimizes a function representing the traversal cost for all rays, of any origin and direction, intersecting the geometry.

Ray origins are resolved to the starting sector 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 first primitive intersected (if any); typically used for visibility computations.
- **any-hit**: indicates whether any primitive is intersected within a specific interval; often used for shadow or ambient occlusion rays.
- **multi-hit**: returns, in order, all intersected primitives; typically used for transparency and operations in non-optical simulation domains.

These algorithms permit the implementation of a number of ray-based simulation algorithms, including traditional image synthesis.

Performance

The data above show initial performance measurements for each traversal algorithm using the tank (red), conference (gray), and san miguel (white) scenes on an NVIDIA GeForce GTX 680 at 1024×768 pixels:

- **vis**: first-hit visibility & simple N-V shading.
- **x-ray**: all multi-hit intersections along the ray & simple alpha-blending.
- **ao**: first-hit visibility & 32 any-hit ambient occlusion rays per hit point.
- **kajiya**: first-hit visibility, any-hit shadows, & two first-hit diffuse bounces.

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.

Rayforce clearly delivers high performance GPU ray tracing for each of the first-hit, any-hit, and multi-hit traversal algorithms required to solve a variety of problems in physically based simulation.

Future work

We are currently exploring additional memory optimizations for our graph acceleration structure, and we 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. Finally, we hope to further reduce the overheads induced by supporting different traversal algorithms.

Source code available via http://rayforce.survice.com