Ray Tracing Visualization Toolkit

Jeremy Fisher  Daniel Eby  Ed Quigley  Gideon Ludwig  Christiaan Gribble*
Department of Computer Science
Grove City College

Figure 1: Ray Tracing Visualization Toolkit. A collection of C++ libraries and an extensible GUI integrate existing renderers via a flexible plug-in architecture to support the visual analysis of ray-based rendering algorithms. Effective visualization of ray tracing operations may promote better understanding—and, ultimately, more efficient implementation—of these algorithms.

1 Introduction
Monte Carlo rendering algorithms typically simulate light transport throughout an environment via ray tracing. Even with recent advances targeting highly parallel platforms [van Antwerpen 2011], high-quality images often require many seconds of computation to converge. The visual analysis of ray tracing operations may promote a better understanding of the way in which computation proceeds and expose opportunities to utilize hardware resources more effectively. This work-in-progress introduces the Ray Tracing Visualization Toolkit (rtVTK), a collection of C++ libraries and an extensible GUI designed to support the visual analysis of ray-based rendering algorithms.

2 Components
rtVTK consists of several components that combine to form a complete ray tracing visualization framework.

Rendering state recording library. An OpenGL-style API called rl captures rendering state in client applications. Existing renderers are instrumented with calls to the rl engine, and per-ray data—including arbitrary client payloads—is recorded throughout rendering. For example, the following pseudocode demonstrates the use of core rl functionality in a basic recursive ray tracer:

```c
// loop over pixels
for (uint y = 0; y < height; ++y)
  for (uint x = 0; x < width; ++x)
    // generate visibility ray and trace
    rlBeginTree(x, y);
    trace(visibilityRay, ...);
    rlEndTree();

// perform ray tracing computations and resource
rlAddRay(r.o, r.d, r.t, ray.type, &my_data, sizeof(MyData));
rlDescendTree();
trace(nextRay, ...);
rlAscendTree();
```

The resulting state is then explored using the rtVTK visualization engine (Figures 1 and 2) or by layering additional components above the existing engine via plug-ins.

rtVTK visualization engine. An interactive GPU path tracing plug-in and OpenGL/rl visualization plug-ins provide core rendering and visualization facilities. The rtVTK plug-in manager enables additional renderers and visualization components to extend the core facilities in a flexible, highly configurable manner. Finally, these elements are integrated with an extensible GUI to provide a common ray tracing visualization process across multiple platforms, including Windows, Linux, and Mac OS X.

3 Discussion
Monte Carlo light transport algorithms must process billions of rays to generate highly realistic images. Visual analysis of the resulting state may lead to insights that allow more efficient implementation of these algorithms. The Ray Tracing Visualization Toolkit is designed to support these goals.

As the rtVTK framework matures, a number of interesting rendering and visualization problems—for example, new methods for real-time Monte Carlo ray tracing and new techniques to meaningfully represent large quantities of ray tracing program state—can be explored both thoroughly and rapidly.

Acknowledgments
This work was funded by grants from the II-VI Foundation and the Grove City College Swezey Scientific Research and Instrumentation Fund. The GPUs used in this research were generously donated by NVIDIA through their Professor Partnership Program.

References