RckT: Scalable Physically Accurate Spectral Rendering in OSPRay

Christiaan Gribble
SURVICE Engineering

Intel HPC Developer Conference
11 November 2017
Innovative Rendering for Simulation

High-performance ray tracing will enable the next-generation of physics-based simulation applications for advanced defense-related R&D

Physics-Based Rendering via the Light Transport Equation

$L(x, \omega_i) = L_e(x, \omega_i) + \int_{\Omega} f_s(x, \omega_i, \omega_o) L_i(x, \omega_o) \, d\omega_o$

Advanced high-performance ray tracing techniques combine with current and future computing architectures to render physically accurate images (front) of complex 3D scenes (back) at interactive frame rates.
Take-Home Messages

• High-performance ray tracing
  Poised to enable the next-generation of physics-based simulation applications for advanced defense-related R&D

• RckT designed for advanced R&D applications

• OSPRay permits easy integration
Take-Home Messages

• High-performance ray tracing

• RckT designed for advanced R&D applications
  Supports scalable physically accurate spectral rendering for ray-based simulation, rendering, & visualization

• OSPRay permits easy integration
Take-Home Messages

- High-performance ray tracing
- RckT designed for advanced R&D applications
- OSPRay permits easy integration
  - Supports scalable, high-performance simulation & visual analysis tools across optical & non-optical domains
Acknowledgments

Jefferson Amstutz
Software Engineer

Ingo Wald
Tech Lead

Jim Jeffers
Director & Principal Engineer
With a staff of approximately 350, SURVICE employees have backgrounds in engineering, physics, mathematics, chemistry, computer science, acquisition, technical writing, training, and other technical and administrative fields. And many of our personnel have DoD or operational military experience. Using a matrix approach, we draw on expertise from across our workforce to build customized work teams for each project and customer. The result is a successful, skilled workforce and many satisfied customers.
Overview
R&D Goals

Our primary objective for the R&D process is to develop and commercialize a high-performance, scalable physics-based rendering system for advanced defense-related simulation applications.

Phase I
Demonstrate efficacy of ray-based rendering

Phase II
Fully develop scalable physics-based rendering system

Phase III
Package HW + SW for turnkey solution supporting advanced simulation & rendering applications
System Concept

Physics-Based Rendering via the Light Transport Equation

Modern ray tracing APIs

**multi-node**

**Phase I** – focus on single-node ray tracing process to demonstrate efficacy of ray-based rendering for simulation applications
Enabling Technology: Monte Carlo Path Tracing

Physics-Based Rendering via the Light Transport Equation

\[ L(x, \omega_i) = \int_{\Omega} f_s(x, \omega_s, \omega_i) L_r(x, \omega_s) (\omega_s \cdot \hat{n}) d\omega_s \]

Advanced high-performance ray tracing techniques combine with current and future computing architectures to render physically accurate images (front) of complex 3D scenes (back) at interactive frame rates.

Elegant, comprehensive, & extensible – satisfies both fidelity requirements and performance requirements
Enabling Technology: Massively Parallel Computing

Scalable, parallel rendering – leverage both fine- and coarse-grained parallelism to scale rendering across full range of modern massively parallel computing resources.
Enabling Technology: Modern Ray Tracing APIs

Accelerated SW development – employ modern, widely available, and extensible ray tracing APIs to rapidly develop phenomenological effects necessary to support simulation scenarios
Design & implementation
OSPRay

Interactive CPU Rendering

MPI Distributed

Open Source
ispc
– Intel SPMD Program Compiler

Implement a Single Program Multiple Data (SPMD) - on SIMD programming model

http://ispc.github.io/

prompt> ispc foo.ispc -o foo.obj

```c
float getResponse(const uniform SpectralResponse* varying self, const varying float lambda)
{
    int nr = self->n_ranges;
    const float* lo = self->lo;
    const float* hi = self->hi;
    const int* ns = self->ns;
    const int* off = self->off;
    const float* bw = self->bw;
    const float* iw = self->iw;
    const float* spd = self->spd;
    if (i < 0)
        return 0.f;
    int j = bin(lambda, lo[i], hi[i], nr);
    const int i = range(lambda, lo[i], hi[i], nr);
    const int o = off[i];

    #if USE_NEAREST_NEIGHBOR
    return spd[o];
    #else
    const int k = clamp(j-1, zero, ns[i]-1);
    const float w = (lambda - (lo[i] + j*bw[i]))*iw[i];
    return lerp(w, spd[i], spd[o+k]);
    #endif // USE_NEAREST_NEIGHBOR
```
Components

- rckt engine
  - Parallel SIMD ray tracing
  - Physically based spectral rendering
  - Module for OSPRay framework
- rcktViewer
- rcktBench
Components

- rckt engine
- rcktViewer
  - Prototype interactive GUI
  - Asynchronous rendering engine
  - Explore simulation environment
- rcktBench
Components

- rckt engine
- rcktViewer
- rcktBench
  - Light-weight command line utility
  - Synchronous rendering engine
  - Measure performance characteristics
rckt Engine

• Spectral components
  • Provide basis for spectral rendering
  • Response, Range, Framebuffer, ...

• BSDFs
• Materials
• Emitters
• Renderer

(Image source: http://www.chromacademy.com)
rckt Engine

- Spectral components
- BSDFs
  - Characterizes interaction of light & matter
  - Lambert, Dielectric, ...
- Materials
- Emitters
- Renderer
rckt Engine

• Spectral components
• BSDFs
• Materials
  • Encapsulate one or more BSDFs to simulate material properties
  • Matte, Glass, …
• Emitters
• Renderer
**rckt Engine**

- Spectral components
- BSDFs
- Materials
- Emitters
  - Inject electromagnetic energy into scene
  - Ambient, Spherical, ...
- Renderer

(Image source: https://wtop.com)
rckt Engine

• Spectral components
• BSDFs
• Materials
• Emitters
• Renderer
  • Implements Monte Carlo path tracing for full global illumination
  • Leverages OSPRay framework for all other rendering components
Current Status

• Demonstrated
  • Physically accurate spectral rendering
  • Highly complex scenes
  • Interactive performance
  • Multi-threaded, multi-node scalability

• Anticipated
Current Status

• Demonstrated

• Anticipated
  • Latest algorithmic improvements
  • Additional low-level optimizations
  • Continued hardware progress
Current Status

• Demonstrated
• Anticipated

**Phase II & beyond** – expecting significant improvements, both in feature set and in rendering performance, to support advanced defense-related R&D
Results
Performance – Scenes

Italian
Bradley
Cougar6x6

Lower
Geometric Complexity
Higher
Performance – Test Platforms

• Dell Precision 7810 workstation
  • Two Intel Xeon E5-2699v3 CPUs
  • 32 GB RAM
  • Ubuntu 16.04

• Intel SDVis Appliance
Performance – Test Platforms

• Dell Precision 7810 workstation

• Intel SDVis Appliance
  • Head: One CX-87723 Intel Xeon node
  • Compute: Eight CX-87719 Intel Xeon Phi nodes
  • CentOS 7.3
Single-node – Summary

![Graph showing normalized performance vs. thread count. The performance increases with the thread count, reaching a peak at 72 threads.]
Multi-node – Summary

The graph shows the normalized performance of the system as a function of node count. As the number of nodes increases, the normalized performance also increases. The data points are color-coded to differentiate between different scenarios or conditions.
Wrap-up
Innovative Rendering for Simulation

High-performance ray tracing will enable the next-generation of physics-based simulation applications for advanced defense-related R&D

Physics-Based Rendering via the Light Transport Equation

$$L(x, \omega_i) = L(x, \omega_i) + \int_{\Omega} f_i(x, \omega_i, \omega_r) L(x, \omega_r)$$

Advanced high-performance ray tracing techniques combine with current and future computing architectures to render physically accurate images (front) of complex 3D scenes (back) at interactive frame rates.
Take-Home Messages

• High-performance ray tracing
  Poised to enable the next-generation of physics-based simulation applications for advanced defense-related R&D

• RckT designed for advanced R&D applications

• OSPRay permits easy integration
Take-Home Messages

• High-performance ray tracing

• RckT designed for advanced R&D applications
  Supports scalable physically accurate spectral rendering for ray-based simulation, rendering, & visualization

• OSPRay permits easy integration
Take-Home Messages

• High-performance ray tracing
• RckT designed for advanced R&D applications
• OSPRay permits easy integration
  Supports scalable, high-performance simulation & visual analysis tools across optical & non-optical domains
Future Work

• Algorithmic optimization
  • Machine learning
  • Advanced ray traversal & rendering
  • Real-time filtering
• Heterogeneous computing
• Enhanced rendering features
Future Work

• Algorithmic optimization
• Heterogeneous computing
  • Leverage all computing resources available on target platform
  • Explore workload distribution schemes
• Enhanced rendering features
Future Work

• Algorithmic optimization
• Heterogeneous computing
• Enhanced rendering features
  • Dynamic geometry
  • Additional material models
  • Advanced modeling primitives

(Image source: https://en.wikipedia.org)
Contact Information

Address
Applied Technology Operation
SURVICE Engineering
4603A Compass Point Road
Belcamp, MD 21017

E-mail
christiaan.gribble@survice.com

Web
http://www.rtvtk.org/~cgribble/
Components

• **rckt engine**
  • Parallel SIMD ray tracing
  • Physically based spectral rendering
  • Module for OSPRay framework

• **rcktViewer**
  • Prototype interactive GUI
  • Asynchronous rendering engine
  • Explore simulation environment

• **rcktBench**
  • Light-weight command line utility
  • Synchronous rendering engine
  • Measure performance characteristics
Current Status

• Demonstrated
  • Physically accurate spectral rendering
  • Highly complex scenes
  • Interactive performance
  • Multi-threaded, multi-node scalability

• Anticipated
  • Latest algorithmic improvements
  • Additional low-level optimizations
  • Continued hardware progress

Phase II & beyond – expecting significant improvements, both in feature set and in rendering performance, to support advanced defense-related R&D
Performance – Test Platforms

Dell Precision 7810 workstation
• Two Intel Xeon E5-2699v3 CPUs
• 32 GB RAM
• Ubuntu 16.04

Intel SDVis Appliance
• Head: One CX-87723 Intel Xeon node
• Compute: Eight CX-87719 Intel Xeon Phi nodes
• CentOS 7.3
Single-Node – italian

[Graph showing performance (millions of rays per second) vs. thread count]
Single-Node – bradley

![Graph showing performance in millions of rays per second vs. thread count for Single-Node bradley. The x-axis represents the thread count ranging from 4 to 72, and the y-axis represents performance ranging from 0.00 to 20.00. The graph includes data points at thread counts of 4, 8, 16, 32, 64, and 72, with corresponding performance values of 0.97, 1.94, 3.87, 7.75, 15.50, and 17.44 millions of rays per second, respectively.](image-url)
Single-Node – cougar6x6
Multi-Node – italian
Multi-Node – bradley

Performance (millions of rays per second) vs. Node Count