Embree Ray Tracing Kernels for CPUs and the Xeon Phi Architecture

Sven Woop[*] Intel Labs
Louis Feng[†] Intel Corporation
Ingo Wald[‡] Intel Labs
Carsten Benthin[§] Intel Labs

1 Introduction

Modern CPUs achieve high computational throughput by implementing increasingly wide SIMD vector units (such as 8-wide AVX or 16-wide SIMD for the Xeon Phi instructions). Achieving optimal performance on these architectures requires leveraging these wide SIMD vector units effectively. We present Embree [Ernst and Woop 2011], an open source ray tracing library developed to show performance-focused graphics programmers how to take full advantage of multiple cores and wide SIMD units in the context of ray tracing. Embree features spatial acceleration structures and traversal algorithms that are optimized for CPUs and the Intel Xeon Phi architecture. In particular, Embree supports hybrid ray packet/single ray traversal algorithms—optimized for both CPUs and Xeon Phi—that are designed to handle both coherent and incoherent workloads efficiently [Benthin et al. 2012]. While a first version of Embree originally focused only on single ray traversal on SSE- or AVX-enabled CPUs, this talk specifically covers the upcoming Embree 2.0 release that explicitly also supports the Xeon Phi architecture, adds support for packet tracing, two level hierarchies, partial scene updates, dynamic content, and virtual intersectors for user defined primitives.

Xeon Phi is a powerful platform for rendering because of its high computational capabilities, fast memory system including large caches, and flexible programming model that supports complex algorithms. The Xeon Phi coprocessor provides 60 cores, each core with a 512-bit wide vector unit and four concurrent hyperthreads. Embree contains highly optimized data structure builders and single-ray kernels that achieve best performance on this architecture. Porting an existing renderer to Xeon Phi can be as easy as replacing the ray tracing core with these kernels and recompiling the system. While this approach of porting to Xeon Phi is straightforward and gives good results, it would not fully leverage computational capabilities of Xeon Phi because the non-traversal rendering code might not make good use of the vector units.

To better leverage the wide vector units of Xeon Phi throughout the entire renderer, Embree 2.0 additionally provides a tight integration with the Intel SPMD Program Compiler (**`ispc`**) [Pharr and Mark 2012]. **`ispc`** is a *Single Program, Multiple Data (SPMD)* vectorizing language similar to OpenCL. In addition to SPMD, it also supports scalar data and control flow and allows for a tight integration with existing C++ code. Using **`ispc`** makes it possible to have a single implementation of a rendering system, partly written in C++, that can transparently leverage the SSE, AVX, and Xeon Phi instruction sets throughout the entire rendering system without having to write any IA-specific code at all. While most of the renderer is written in high-level **`ispc`** code, we exploit **`ispc`**'s tight coupling to C++ code to switch over to special high-performance traversal kernels (implemented in C++ and intrinsics) whenever the renderer casts a ray; Embree automatically uses different traversal codes specialized for each instruction set, without the user having to care about this.

Embree provides only a minimal API to the acceleration structure builders and traversal algorithms, which gives the user all the flexibilities for implementing their own rendering system. Embree also provides an example renderer for CPUs written in C++ and a second implementation written in **`ispc`** that performs best on Xeon Phi. We have seen many vendors and users successfully integrate the Embree kernels into their own projects. For example, DreamWorks Animation has showcased a lighting tool prototype utilizing Embree 2.0 at Super Computing 2012.

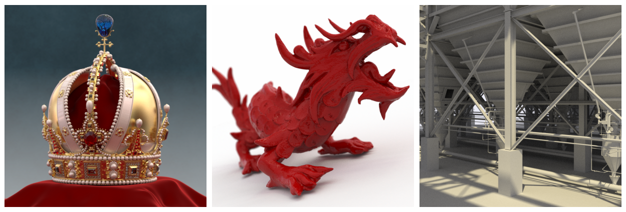

Figure 1: *Models rendered with the Embree ray tracing kernels using the example path tracer. The crown model is provided by Martin Lubich, http://www.loramel.net.*

2 Presentation

In this presentation, we will discuss the design challenges, new features, and our experiences with the development of Embree 2.0. For potential users of Embree, we will show how to use the Embree API from a C++ or **`ispc`** application. For attendees interested in implementation details, we will describe the SIMD friendly ray tracing algorithms for CPUs and the Xeon Phi hardware architecture including performance numbers for non-trivial path tracing workloads for both architectures. In particular, we will present the single ray traversal kernels which achieve best performance on CPUs and our hybrid ray packet/single ray approach that gives optimal performance on Xeon Phi. We will also talk about our impressions on the potential and challenges of using Xeon Phi for ray tracing.

References

BENTHIN, C., WALD, I., WOOP, S., ERNST, M., AND MARK, W. R. 2012. Combining single and packet-ray tracing for arbitrary ray distributions on the intel mic architecture. *IEEE Transactions on Visualization and Computer Graphics 18*, 9, 1438–1448.

ERNST, M., AND WOOP, S., 2011. Embree: Photo-Realistic Ray Tracing Kernels. http://software.intel.com/en-us/articles/embree-photo-realistic-ray-tracing-kernels, June.

PHARR, M., AND MARK, W. 2012. ISPC: A SPMD Compiler for high-performance CPU Programming. In *Innovative Parallel Computing (InPar), 2012*, 1 –13.

[*]e-mail:sven.woop@intel.com
[†]e-mail:louis.feng@intel.com
[‡]e-mail:ingo.wald@intel.com
[§]e-mail:carsten.benthin@intel.com