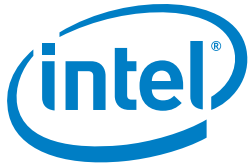
WHITE PAPER
June 2011



Embree: Photo-Realistic Ray Tracing Kernels

Open source ray tracing kernels for fast photo-realistic rendering on Intel® CPUs

ABSTRACT

Embree is a collection of high-performance ray tracing kernels, developed at Intel Labs. The kernels are optimized for photo-realistic rendering on the latest Intel® processors with support for the SSE and AVX instruction sets. In addition to the ray tracing kernels, Embree provides an example photo-realistic rendering engine. Embree is designed for Monte Carlo ray tracing algorithms, where the vast majority of rays are incoherent. The specific single-ray traversal kernels in Embree provide the best performance in this scenario and they are very easy to integrate into existing applications.

This document gives an overview of how photo-realistic rendering with Monte Carlo ray tracing works and how the Embree ray tracing kernels improve the performance of this algorithm.

"The Embree ray tracing kernels accelerate photo-realistic rendering in professional applications by up to 2x"

Manfred Ernst
Intel Labs

Sven Woop
Intel Labs

Who uses photo-realistic rendering?

Photo-realistic rendering is used in a wide range of applications. Designers and engineers use the technology to visualize virtual prototypes. This usage reduces time to market and development cost by reducing the number of physical prototypes required. In recent years, the quality of computer-generated images has reached a level of realism where renderings are indistinguishable from photographs. This made it possible to replace photographs by computer generated pictures for marketing purposes. In the same way, architects use rendering technology to visualize new buildings for their customers and they use similar methods to accurately model the interior lighting. Photo-realistic rendering is also used extensively for visual effects and animated feature films by the movie industry.

Embree is not targeting the end users of rendering technology directly. Instead, the kernels were developed for integration into existing and future rendering applications. By using the open source Embree ray tracing kernels, researchers and developers can achieve the highest level of performance on Intel® CPUs. Users will automatically benefit when software developers make use of Embree in their products.

How does it work?

Photo-realistic rendering is the process of turning 3D models into images that are indistinguishable from a photograph. It requires the accurate simulation of light propagation according to the laws of physics. The best known method for solving this problem is Monte Carlo ray tracing, an algorithm that follows the paths of billions of light rays as they reflect off surfaces in a virtual scene. The two key challenges in Monte Carlo ray tracing are (a) carefully selecting a statistically representative

Figure 1. Progressive rendering of the imperial crown of Austria. A single machine with four Intel® Xeon® processors computes preview images of this 3D model at interactive frame rates (left). The image converges to a better solution within a few seconds (middle). A perfect image (right) only takes about a minute to compute. Model courtesy of Martin Lubich, www.loramel.net.

set of light paths, and (b) determining the intersection points of the path segments with the scene as quickly as possible. The latter, known as the visibility problem, is solved by the ray tracing kernels and it is usually the most compute intensive part of a rendering system.

Embree provides a Monte Carlo ray tracer as an example. This renderer demonstrates how an efficient rendering system is designed and implemented using Embree's key technologies. The renderer is also an excellent framework for evaluating and comparing different ray tracing kernels in a realistic application scenario.

Describing Reality

Monte Carlo ray tracing requires a highly detailed and physically-based scene description as input. The algorithm applies the laws of physics to simulate the propagation of light through the scene, rather than ad hoc approximations of visual phenomena. This type of simulation requires extremely detailed geometric models (engineering models for example are typically accurate to a fraction of a millimeter). Because ray tracing is less performance-sensitive to geometric complexity, all surfaces can be finely tessellated. In addition to high geometric detail, photo-realistic rendering requires that surface appearance is modeled by the physical properties of the material. In contrast to rasterization, where shaders are used to achieve certain visual effects, the materials in a photo-realistic ray tracer describe how light is scattered when striking a surface. This information is represented by a function known as a BRDF (Bidirectional Reflectance Distribution Function). Light sources are also described by their physical emission properties. A very common representation is the high dynamic range (HDR) environment light. It models the lighting conditions of a real location in a single HDR image. This image is then used as a light source in the rendering system. Virtual objects illuminated by this light appear as if they were placed in the real location.

Embree uses triangle meshes to describe the shape of objects. Materials are modeled by a set of BRDF components. Implementations are provided for common materials such as metal, plastic, dielectrics, car paint, cloth and diffuse reflectors. Only basic texturing is supported. Light sources include point lights, triangle lights, directional lights and HDR lights. Both lights and materials are programmable and separated from the integration stage of the renderer. Though Embree's example renderer implements a limited set of scene objects, the underlying technologies are general and can be applied to a broader range of scene representations.

Simulating Reality

The key concept of Monte Carlo ray tracing is to randomly select a large number of light paths for each pixel and average their contributions for the final color value. A light path connects a point on the image plane with a light source, either directly or mediated by one or more surfaces in the scene (Figure 2). The space of valid light paths is defined by the scene description and the laws of physics.

The Monte Carlo ray tracing algorithm computes the final pixel color by averaging a large number of random samples. While the result is statistically correct for a large number of samples, too few samples results in visible noise artifacts. For a high quality result, hundreds or even thousands of light paths are required per pixel. In practice these paths are usually not chosen entirely at random. Instead, sophisticated algorithms have been developed to select the paths which provide the most information about the scene. The part of the rendering engine responsible for choosing the paths and combining their results is known as the integrator, because the most common mathemati-

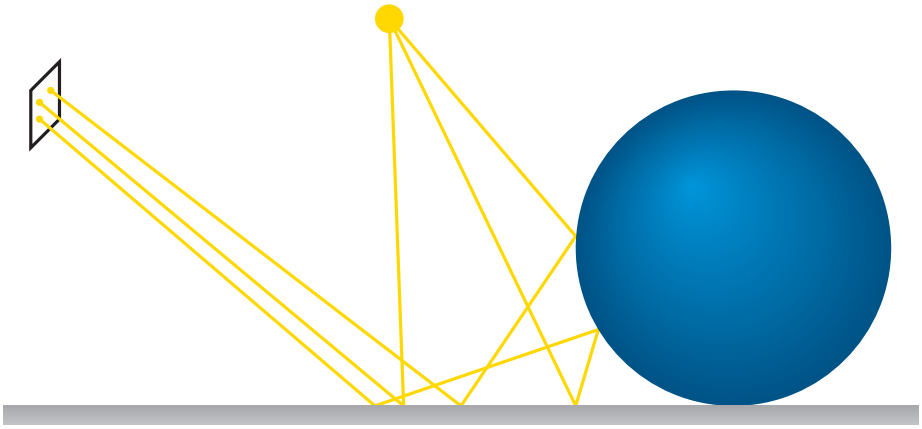


Figure 2. Ray tracing simulates the propagation of light in a scene. The figure shows three possible light paths that connect the light source with a pixel in the image plane. The intersection points of the path segments with the scene are computed by the ray tracing kernel.

cal formulation of the problem takes the form of an integral. Every path consists of multiple segments that each corresponds to the path of a single virtual photon.

There exist a large number of Monte Carlo ray tracing algorithms. They differ in how the light paths are chosen and what effects are efficiently supported. Path tracing[1] is one example. It traces rays backward from the camera towards the light sources. Another example is stochastic progressive photon mapping[2], where paths are traced both from the camera and from the light sources. The paths are then loosely connected at their end points.

Different applications required different rendering algorithms. This is why Embree only provides an example in this space. We have chosen the path tracing[1] algorithm, because it is simple and it works well in many applications. The architecture of the renderer was inspired by the design of PBRT[3].

Incoherent Rays

The performance critical component of a photo-realistic rendering engine is its ray tracing kernel. This component is responsible for determining the intersections between the light paths and the scene's surfaces.

A major challenge in achieving high performance is that the generated rays are geometrically incoherent. That means they do not share a common origin and they propagate in arbitrary directions with no obvious pattern. This differentiates Monte Carlo ray tracing from real-time ray tracing, where high frame rates are achieved by enforcing coherence for all rays (see Figure 3 for a comparison). The requirement of coherency limits the possible visual effects to hard shadows and simple specular reflection and refraction. Advanced effects such as HDR environment lighting, glossy reflections, deep refraction and diffuse global illumination cannot be handled properly. Monte Carlo ray tracing has no such limitations and can potentially simulate every visual effect modeled by classical ray optics. The cost of this flexibility and the resulting photo-realistic image quality is a vast increase in the number of rays required. Even the fastest processors today require several seconds or minutes to compute a noise-free high-resolution image of a complex model, and in 1984 when the first Monte Carlo ray tracing algorithm was introduced[4], even the simplest images took many hours or even days to compute. Consequently, there is persistent demand for faster algorithms and highly optimized implementations. Today it is possible to render preview images at interactive frame rates on a single chip. Compute clusters can even render high-quality pictures interactively.

Embree is specifically optimized for high performance with incoherent rays. As a consequence, it outperforms algorithms designed for coherent rays, such as real-time ray tracing, when used for the incoherent rays in photo-realistic rendering.

Acceleration Structures

The core of a ray tracer is its acceleration structure. Imagine a scene with tens of millions of triangles and billions of rays being traced. The brute force approach of testing every ray against every surface element (typically triangles) for intersection is clearly infeasible. Instead, the triangles are sorted into a spatial data structure that guides the rays to potential intersection candidates. A popular acceleration structure known as a bounding volume hierarchy (BVH) sorts triangles into a hierarchy of boxes, each level containing increasingly smaller subsets of the scene. At each level, the set of triangles is split into two or more sub-sets until the sets are considered small enough. During rendering, a ray only needs to be intersected with triangles that are contained in a box that the ray intersects. Due to the hierarchical nature of the data structure, the majority of boxes and triangles can be quickly discarded, reducing the work per ray to a few dozen ray-box intersection tests and a few ray-triangle intersections.

The acceleration structures are the core contribution of Embree. They take maximum advantage of the latest Intel® CPUs and they are designed for easy integration into other rendering engines. Embree implements a binary BVH as well as a four-wide multi bounding volume hierarchy[5], both with highly optimized single ray traversal kernels. The parallel acceleration structure builders support spatial splits to efficiently handle scenes with problematic geometry such as large diagonal triangles.

Thread Parallelism

Monte Carlo ray tracing is very easy to parallelize with multiple execution threads because all light paths are mutually independent. The image plane is simply subdivided into a set of small tiles. Whenever a thread finishes rendering of its current tile, it picks the next one from the list of unfinished tiles. Scalability on multi-core processors and multi-socket servers is close to linear. On a four-socket server with a total of 40 physical cores, for example, Embree achieves 95% parallel efficiency when rendering.

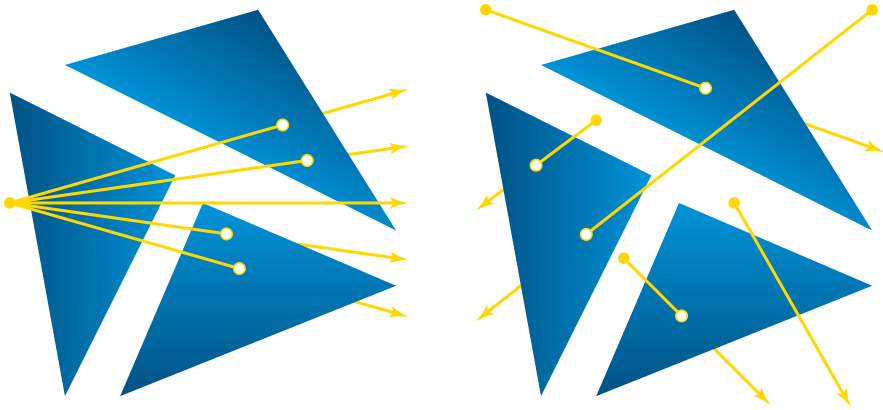


Figure 3. Coherent rays (left) are used for real-time ray tracing. They are handled very efficiently by packet tracing algorithms. Incoherent rays (right), are more difficult to handle, but they are required for photo-realistic rendering.

Data Parallelism

In addition to thread parallelism that maps tasks to the cores of a processor, there also is data parallelism that maps computation within a thread to the SIMD (Single Instruction Multiple Data) units of a CPU. Data parallelism is more difficult to exploit than thread parallelism. It works best when multiple collocated data items are processed by the same instruction stream. For real-time ray tracing this can be achieved by treating a set of similar rays as a packet and tracing them together through the acceleration structure. Because the rays are coherent, they are likely to visit the same boxes and intersect the same triangles. This results in excellent performance, because neither the memory access nor the control flow diverges. This scheme, however, breaks when the rays become incoherent. Each ray may travel through a different part of the scene and they may also want to execute different code sections. One ray, for example, might already have found its closest intersection and now wants to proceed to material evaluation, while another ray is still searching for its hit point. Fortunately, there are other strategies to utilize data parallelism. Instead of grouping rays together, we can also group data elements of the acceleration structure together. Embree uses this approach. All rays are traced independently, which greatly simplifies the development of the renderer.

Embree supports two acceleration structures that use the four-wide data parallel instructions provided by Intel® Streaming SIMD Extensions 4 (Intel® SSE4). The first is a bounding volume hierarchy with a branching factor of four[5]. It packs four boxes together in an SSE friendly layout and computes the intersection of a ray with all four of them in parallel. Triangles are treated similarly. The second acceleration structure is a traditional binary bounding volume hierarchy. It also stores the boxes in a special layout in memory and intersects a ray with the near and far planes of two boxes in parallel. This is possible with the fast shuffling operations provided by Intel® SSE4 and a simple arithmetic trick: min(a,b) = -max(-a,-b). This allows us to execute minimum and maximum computations in the same four-wide register by flipping some of the sign bits before and after the computation.

The acceleration structures are carefully optimized to take maximum advantage of the latest Intel® processors. Optimal instruction scheduling, latency minimization, and cache-coherent memory access patterns were important considerations.

Summary

Embree provides highly optimized ray tracing kernels that speed photo-realistic rendering on Intel® CPUs by up to 2x. Intel® has released these kernels as open source under the Apache 2.0 license.

Download Embree: http://software.intel.com/en-us/articles/embree-photo-realistic-ray-tracing-kernels/

[1] James T. Kajiya: The Rendering Equation. In Proceedings of SIGGRAPH '86, pp. 143–150, (1986).

[2] Toshiya Hachisuka and Henrik Wann Jensen: Stochastic Progressive Photon Mapping. In Proceedings of SIGGRAPH Asia 2009, Article 141, (2009).

[3] Matt Pharr and Greg Humphreys: Physically Based Rendering: From Theory To Implementation. Morgan Kaufmann, 2nd revised edition, (2010).

[4] Robert L. Cook, Thomas Porter, Loren Carpenter: Distributed Ray Tracing. In Proceedings of SIGGRAPH '84, pp. 137–145, (1984).

[5] Manfred Ernst and Günther Greiner: Multi Bounding Volume Hierarchies. In Proceedings of the IEEE/EG Symposium on Interactive Ray Tracing 2008, pp. 35–40, (2008).