EMBREE RAY TRACING KERNELS 3.X: OVERVIEW AND NEW FEATURES

Carsten Benthin, Principal Engineer

Intel Corporation
No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade. This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps. The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/benchmarks.

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system. Intel, Xeon and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.
© Intel Corporation.
ACRONYM LIST

• Application Programming Interface (API)
• Bounding Volume Hierarchy (BVH)
• Independent Software Vendor (ISV)
• Instruction Set Architecture (ISA)
• Intel® Advanced Vector Extensions (Intel® AVX)
• Intel® Advanced Vector Extensions 2 (Intel® AVX2)
• Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
• Intel® SPMD Program Compiler (Intel® SPC)
• Intel® Streaming SIMD Extensions (Intel® SSE)
• Intel® Threading Building Blocks (Intel® TBB)
• Non-Uniform Rational Basis Spline (NURBS)
• Single Instruction, Multiple Data (SIMD)
• Single Program, Multiple Data (SPMD)
• Surface Area Heuristic (SAH)
EMBREE OVERVIEW
EMBREE API
SELECTED ADVANCED FEATURES
EMBREE PERFORMANCE
SUMMARY & OUTLOOK
EMBREE OVERVIEW

EMBREE API
SELECTED ADVANCED FEATURES
EMBREE PERFORMANCE
SUMMARY & OUTLOOK
Movie industry intensively uses ray tracing today (better image quality, faster feedback).

High-quality rendering for commercials, prints, etc.

Provides higher fidelity for virtual design (automotive industry, architectural design, etc.)

Various kinds of simulations (lighting, sound, particles, collision detection, etc.)

Prebaked lighting in games, starting to go real-time for ray traced lighting and sound effects.
FAST RAY TRACING CHALLENGES

• Need to multi-thread
  Easy for rendering but difficult for hierarchy construction

• Need to vectorize
  Efficient use of SIMD & ISAs (Intel® SSE, Intel® AVX, Intel® AVX2, Intel® AVX-512)

• Need to support different CPUs
  Different ISAs/CPUs favor different data structures, data layouts, and algorithms

• Need deep domain knowledge
  Many different data structures and algorithms to choose from

• Different usage scenarios
  Large model visualization favors memory conservative algorithms
EMBREE RAY TRACING KERNELS

• Targets professional rendering applications
• Provides highly optimized ray tracing kernels
  • 1.5–6× speedup reported by users
• Provides rich functionality and flexibility
• Support for latest CPUs and ISAs (e.g. Intel® AVX-512)
  • Windows* (64 and 32 bit), macOS* 10.x, Linux*
• API for easy integration into applications
• Open Source under Apache* 2.0 license:
  • http://embree.github.com

*Other names and brands may be claimed as the property of others.
EMBREE BROAD ADOPTION – 70+ APPS

- AUTODESK
- DREAMWORKS
- CPU/Embree
- Corona
- 3D EXCITI
- blender
- UBISOFT
- Hair Primitives
- FluidRay
- Embree
- Broad Adoption – 70+ Apps
- DWA How To Train Your Dragon 2
- CPU/Embree Only Corona Renderer
- ParaView
- Visit
- Embree
- V-Ray
- DWA How To Train Your Dragon 2
- ParasView with OSPRay
- ANL VL3 Dark Matter - OpenSWR
- Survice StingRay
- Rendered with FluidRay RT

*Other names and brands may be claimed as the property of others.
EMBREE TIMELINE

2014

2.0: Xeon Phi, Ray packets, SPCC

2015

2.1: New API, Runtime code selection

2.2: Intersection filter

2.3: Hair support

2.3.1: BVH8, Spatial splits

2.5: Threading Building Blocks

2.7: Device concept

2.8: Line geometry, Quad geometry

2.9: Ray streams

2016

2.10: Geometric curve

2.11: Frustum traversal

2.12: Multi segment motion blur

2.14: Ribbon hair intersector

2.16: Improved multi segment motion blur, improved two level builder

3.0: Improved API, improved memory consumption

3.1: Normal oriented curves, grid geometry

3.2: Hermite basis

2017

2018
GEOMETRY TYPES

- Triangle meshes
- Quad meshes
- Grid meshes (NEW)
- Subdivision meshes
- Flat curves
- Round curves
- Normal-oriented curves (NEW)
- Instances
- User-defined ➔ extensible
EMBREE FEATURES

- Find closest hit (rtcIntersect), find any hit (rtcOccluded)
- Single rays, ray packets (4, 8, 16), ray streams (N)
- High-quality and high-performance parallel BVH builders
  - Exploit nested parallelism through Intel® Threading Building Blocks (TBB)
  - Multi-segment motion blur, instancing, static/dynamic objects, callback funcs., ...
- API support for applications written in:
  - C/C++ and Intel® SPMD Program Compiler (ISPC)
  - No dependence on other graphics APIs like DirectX*, OpenGL*, ...

*Other names and brands may be claimed as the property of others.
## EMBREE SYSTEM OVERVIEW

<table>
<thead>
<tr>
<th>Acceleration Structures</th>
<th>Builders</th>
<th>Traversal</th>
<th>Intersection</th>
<th>Subdiv Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>bvh4.triangle4</td>
<td>SAH Builder</td>
<td>Single Ray</td>
<td>Möller-Trumbore</td>
<td>B-Spline Patch</td>
</tr>
<tr>
<td>bvh8.triangle4</td>
<td>MBlur Builder</td>
<td>Packet/Hybrid</td>
<td>Plücker</td>
<td>Gregory Patch</td>
</tr>
<tr>
<td>bvh4.quad4v</td>
<td>Spatial Split Builder</td>
<td>Ray Stream</td>
<td>Flat Curve</td>
<td>Tessellation Cache</td>
</tr>
<tr>
<td></td>
<td>Morton Builder</td>
<td></td>
<td>Round Curve</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BVH Refitter</td>
<td></td>
<td>Oriented Curve</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grid</td>
<td>Displ. Mapping</td>
</tr>
</tbody>
</table>

### Common Vector and SIMD Library

(Vec3f, Vec3fa, vfloat4, vfloat8, vfloat16, ..., Intel® SSE2, Intel® SSE4.1, Intel® AVX, Intel® AVX2, Intel® AVX-512)
**EMBREE API OVERVIEW**

- Version 3 of the Embree API
- Object-oriented
- Reference-counted
- Device concept
- Compact and easy to use
- Hides implementation details (e.g. ISA and acceleration structure selection)
- For details visit [https://embree.github.io/api.html](https://embree.github.io/api.html)
ADVANTAGES AND NEW FEATURES OF 3.X API

• Cleanup of previous API
• Improved flexibility
• Easier to use + API bug fixes
• New primitives, e.g. normal oriented curves, grids, ...
• Support for > 4 billion primitives
• More robust intersection computations
• Reduced memory consumption for instances and higher performance
• Conversion script makes adoption easy (included in Embree)
EXAMPLE: SCENE CREATION

- Scene contains a vector of geometries
- Scene geometry changes have to get committed (rtcCommitScene), which triggers BVH build

```c
#include <embree3/rtcore.h>

int main()
{
  // create Embree device at application startup
  RTCDevice device = rtcNewDevice();

  // create scene
  RTCScene scene = rtcNewScene(device);

  // attach geometries
  ... later slide ...

  // commit changes
  rtcCommitScene(scene);

  // trace rays
  ... later slide ...

  // release objects
  rtcReleaseScene(scene);
  rtcReleaseDevice(device);
}
```
• Triangle mesh contains vertex and index buffers

• Shared buffers of flexible layout (offset + stride) supported

// application vertex and index layout
struct Vertex { float x, y, z, s, t; };
struct Triangle { int materialID, v0, v1, v2; };

// create triangle mesh
RTCGeometry geom = rtcNewGeometry(device,
RTC_GEOMETRY_TYPE_TRIANGLE);

// share data buffers
rtcSetSharedGeometryBuffer(geom, RTC_BUFFER_TYPE_VERTEX, 0,
RTC_FORMAT_FLOAT3, vertexPtr, 0, sizeof(Vertex));
rtcSetSharedGeometryBuffer(geom, RTC_BUFFER_TYPE_INDEX, 0,
RTC_FORMAT_UINT3, indexPtr, 4, sizeof(Triangle));

// commit geometry
rtcCommitGeometry(geom);

// attach geometry to scene
rtcAttachGeometryByID(scene, geom, user_provided_geomID);

// commit changes
rtcCommitScene(scene);
EXAMPLE: TRACING SINGLE RAYS

- Context passed to potential callbacks
- Use RTCRayHit for normal rays
- Use RTCRay for occlusion rays
- Hit data and ray.tfar set in case of hit

```c
// create intersection context
RTCIntersectContext context;
rtcInitIntersectContext(&context);

// create ray
RTCRayHit query;
query.ray.org_x = 0.0f;
query.ray.org_y = 0.0f;
query.ray.org_z = 0.0f;
query.ray.dir_x = 1.0f;
query.ray.dir_y = 0.0f;
query.ray.dir_z = 0.0f;
query.ray.tnear = eps;
query.ray.tfar = inf;
query.ray.time = 0.0f;
query.hit.geomID = RTC_INVALID_GEOMETRY_ID;
query.hit.primID = RTC_INVALID_GEOMETRY_ID;

// trace ray
rtcIntersect1(scene, &context, query);

// hit data filled on hit
if (query.hit.geomID == RTC_INVALID_GEOMETRY_ID) return;

// hit data filled on hit
float u = query.hit.u;
float v = query.hit.v;
float t = query.ray.tfar;
```
INTEL® SPMD PROGRAM COMPILER (ISPC)

• C99-based language plus vector extensions
• Simplifies writing vectorized renderer
• Scalar looking code that gets vectorized automatically
• Guaranteed vectorization
• Compilation to different ISAs (Intel® SSE, Intel® AVX, Intel® AVX2, Intel® AVX-512)
• Used for written application/rendering/shading code
• Available as Open Source from http://ispc.github.com
Example: Rendering using Intel® ISPC

// loop over all screen pixels
foreach (y=0 ... screenHeight-1, x=0 ... screenWidth-1) {

// create and trace primary ray
RTCRayHit primary = make_RayHit(p, normalize(x*vx + y*vy + vz), eps, inf);
rtcIntersectV(scene, &context, ray);

// environment shading
if (primary.hit.geomID == RTC_INVALID_GEOMETRY_ID) {
    pixels[y*screenWidth+x] = make_Vec3f(0.0f); continue;
}

// calculate hard shadows
RTCRay shadow = make_Ray(primary.ray.hitPoint(), neg(lightDir), eps, inf);
rtcOccludedV(scene, &context, shadow);

if (shadow.tfar <= 0.0f)
    pixels[y*width+x] = colors[ray.primID]*0.5f;
else
    pixels[y*width+x] = colors[ray.primID]*(0.5f + clamp(-dot(lightDir,normalize(primary.hit.Ng)),0.0f,1.0f));
}
SELECTED ADVANCED FEATURES
Quad meshes

- Quad rendered as pairs of triangles \((v0, v1, v3 \text{ and } v2, v3, v1)\)
- Mixed triangle/quad mesh supported \((v0, v1, v3, v3)\)
- Most 3D modeling packages produce quad meshes
- Lower memory consumption
- Faster BVH building
- Ray tracing performance slightly lower than for triangles
• Primitives are grids of vertices with regular triangulation
• For displaced surfaces with higher tessellation levels
  • Use quad meshes for low tessellation levels
• Extremely low memory consumption
  • Down to 4 bytes per triangle
• Use instead of subdiv mesh \textbf{with} displacement function
CATMULL-CLARK SUBDIVISION SURFACES

- Converts coarse mesh into smooth surface (subdivision)
- Support for arbitrary topology
- Established as standard in movie production
- Embree implementation compatible with OpenSubdiv 3.0 (creases, boundary modes, etc.)
- Evaluation of surface supported
- Walking mesh topology supported
CURVE GEOMETRIES

• Curve bases
  • Linear (for very distant curves)
  • Cubic Bézier (widely used representation)
  • Cubic B-spline (most compact)
  • Cubic Hermite (compact and interpolating)

• Curve types
  • Flat curves (for distant geometry)
  • Round curves for close-ups (swept circle)
  • Normal-oriented curves (for grass)
CURVE GEOMETRIES

• Supports varying radius along the curve
• High performance through use of oriented bounding boxes [Woop et al. 2014]
• Low memory consumption through direct ray/curve intersection (new algorithm)
USER-DEFINED GEOMETRIES

• Enables implementing custom primitives and features
  • Sphere, disk, multi level instancing, rotation motion blur, etc.

• User provides:
  • Bounding function
  • Intersect and occluded functions
INTERSECTION FILTER FUNCTIONS

• Per-geometry callback
  • Called during traversal for each primitive intersection
  • Callback can accept or reject hit
• Can be used for:
  • Trimming curves (e.g. modeling tree leaves)
  • Transparent shadows (reject and accumulate)
  • Find all hits (reject and collect)
  • Advanced self-intersection avoidance
MULTI-SEGMENT MOTION BLUR

• Important to render fast curved motion (e.g. rotating wheels, fight scenes, spinning dancers, etc.)

• Sequence of time steps to be piecewise-linearly interpolated

• Typically equidistant time steps and often different number of time steps per geometry

• 4D-BVH which stores linear spatial and temporal bounds
  • BVH can spatially separate geometries
  • BVH can reduce time ranges where required
EMBREE PERFORMANCE

SUMMARY & OUTLOOK
BENCHMARK OVERVIEW

• Path tracer with different material types, different light types, ~2k lines of code
• Similar implementation for CPU (ISPC + Embree) and GPU (CUDA* + OptiX*)
• Highest quality BVH build settings for all platforms
• Evaluation on typical Intel® Xeon® rendering workstation†
  • Dual-socket Intel® Xeon® Platinum 8180 Processor (2x28 cores @ 2.5 GHz)
• Compare against state-of-the-art GPU methods
  • OptiX 5.1.0 and CUDA 9.2.88
  • NVIDIA Tesla* V100 Coprocessor (5120 CUDA cores @ 1.37 GHz, Volta)

*Other names and brands may be claimed as the property of others.
**PERFORMANCE: EMBREE VS. NVIDIA OPTIX**

Frames Per Second (Higher is Better), 1024x1024 image resolution

- **Bentley** (2.3M Tris)
- **Crown** (4.8M Tris)
- **Dragon** (7.4M Tris)
- **Karst Fluid Flow** (8.4M Tris)
- **Power Plant** (12.8M Tris)

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to [http://www.intel.com/performance](http://www.intel.com/performance).

*Other names and brands may be claimed as the property of others.*
SUMMARY & OUTLOOK
SUMMARY

• Embree provides optimized and scalable ray tracing kernels for the CPU
• Latest state-of-the-art feature set
  • Lots of ray tracing research goes directly into Embree 😊
• Actively developed and completely open-source
• Easy to integrate into existing applications
• Lots of ISVs using it as their core ray tracing engine
OUTLOOK

• Denoising
• Quaternion interpolation for transformation motion blur
• Non-uniform motion blur
• New primitive types (disk, sphere, bilinear patch)
• Improve ray/geometry masking and instancing performance
• Point projection onto geometry (robust manifold next event estimation)
• Partial double support
QUESTIONS?

Check out the Embree/OSPRay demos at booth #1300 West Hall

https://embree.github.io
embree_support@intel.com
embree@googlegroups.com