Intel® oneAPI Rendering Toolkit

and its application in games

Sven Woop, Jeff Amstutz

Legal Notices and Disclaimers

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].

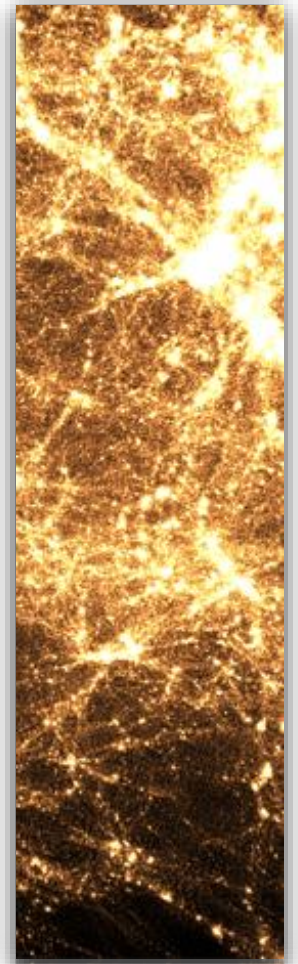
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Open VKL

Open Image Denoise

OSPRay



OpenSWR

Embree

Data courtesy Bentley, Disney* *Other names and brands may be claimed as the property of others

Open VKL

Open Image Denoise

OSPRay

OpenSWR

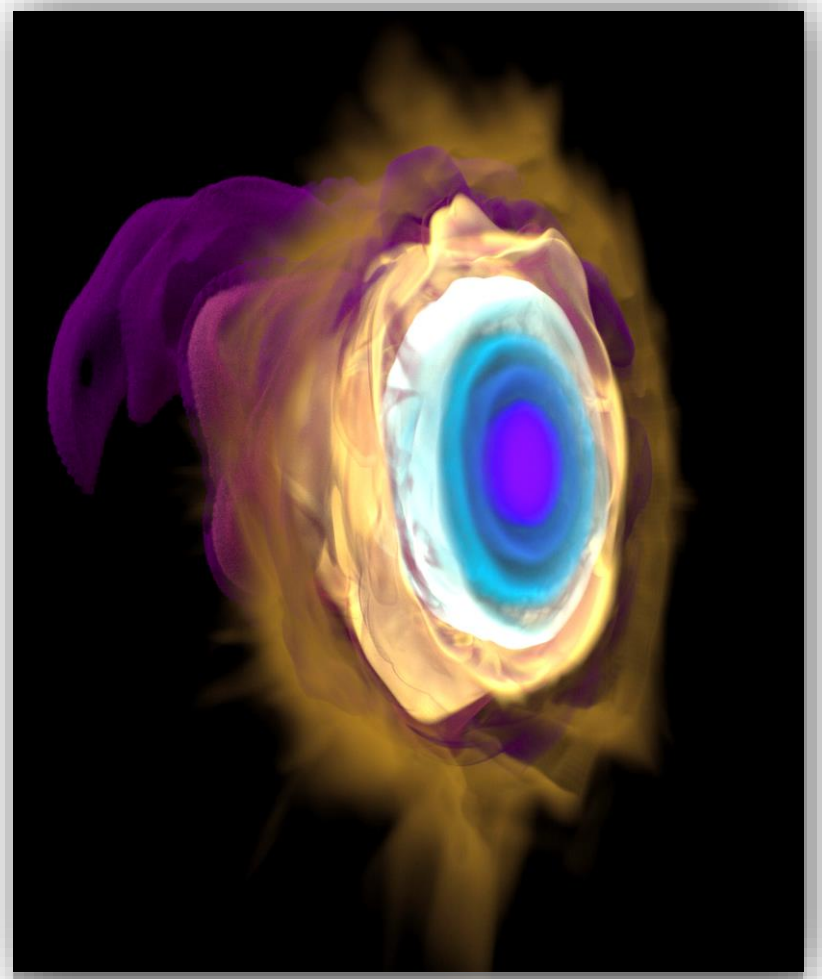
Embree

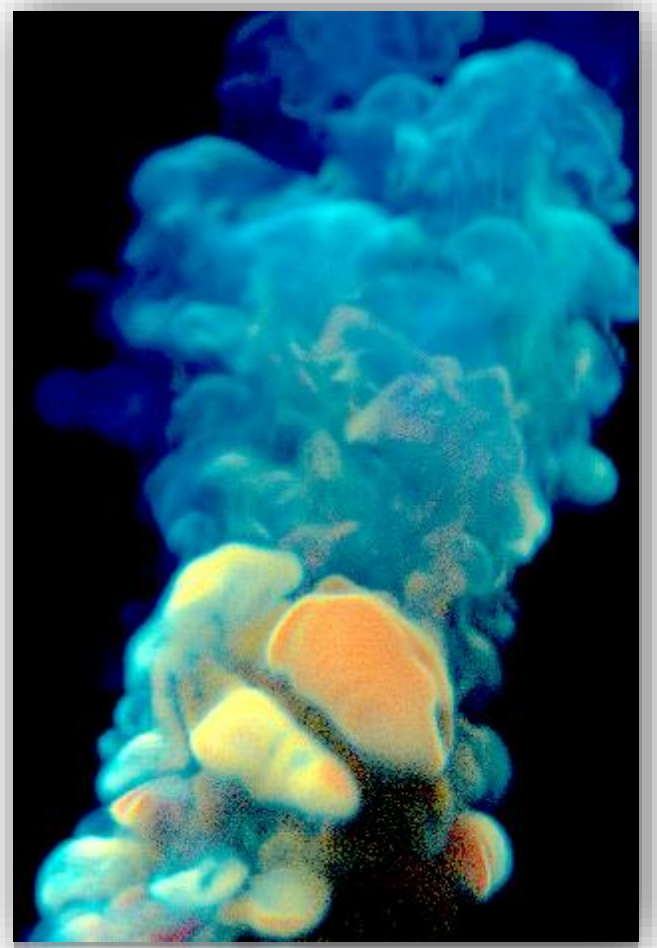
Data courtesy Bentley, Disney* *Other names and brands may be claimed as the property of others

Open **V**olume **K**ernel **L**ibrary

- High performance kernels for sampling and traversing rays in volumetric data (scalar fields)
- APIs for single sampling and packets to aid in vectorization of ray tracing algorithms
- Agnostic to volume structure
  - Implement a common renderer without specializing to specific volume data structures
- Optimized for x86 CPUs
  - GPU support coming!

Open Volume **K**ernel **L**ibrary





Open VKL

Open Image Denoise

OSPRay

OpenSWR

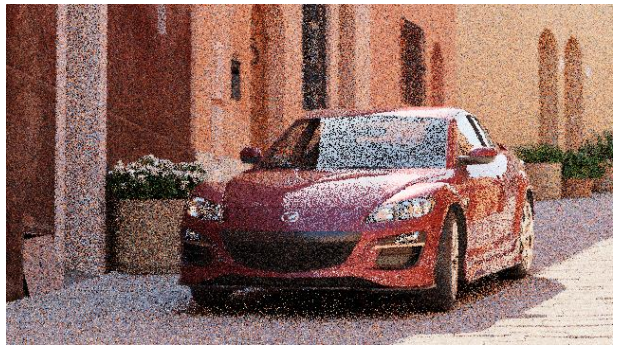
Embree

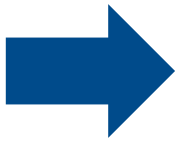
Data courtesy Bentley, Disney* *Other names and brands may be claimed as the property of others

- Denoising library for images rendered with ray tracing
- Provides a high-quality deep learning based denoising filter
- Suitable for both interactive preview and final-frame rendering
- Runs on any modern Intel® Architecture CPU (SSE4.1 → AVX-512)
- GPU support coming!

- Denoising library for images rendered with ray tracing
- Provides a high-quality deep learning based denoising filter
- Suitable for both interactive preview and final-frame rendering
- Runs on any modern Intel® Architecture CPU (SSE4.1 → AVX-512)
- GPU support coming!

- Windows (64-bit), macOS, Linux
- Clean, minimalist C/C++ API and library design
  - Straightforward application integration (in hours)
- Depends only on the Intel® TBB library
- Free and Open Source under Apache 2.0 license
  - http://openimagedenoise.github.com





*Scene by Evermotion.

Crytek Sponza *(HDR, 16 spp)* – Input*

SSIM: 0.1784

*Input buffers: HDR color, albedo, normal. Only the tone mapped color buffer is shown.
Scene courtesy of Frank Meinl, downloaded from Morgan McGuire's Computer Graphics Archive.

Crytek Sponza *(HDR, 16 spp)* – OIDN 1.0.0

SSIM: 0.9756

*Input buffers: HDR color, albedo, normal. Only the tone mapped color buffer is shown.
Scene courtesy of Frank Meinl, downloaded from Morgan McGuire's Computer Graphics Archive.

Crytek Sponza *(HDR, 16 spp)* – Ground Truth

*Input buffers: HDR color, albedo, normal. Only the tone mapped color buffer is shown.
Scene courtesy of Frank Meinl, downloaded from Morgan McGuire's Computer Graphics Archive.

Corona Academy Interior *(HDR, 4 spp)* – Input*

SSIM: 0.3361

*Input buffers: HDR color, albedo, normal. Only the tone mapped color buffer is shown.

Rendered with Corona Renderer. Scene provided by Chaos Czech a.s. www.corona-renderer.com

corona

Corona Academy Interior *(HDR, 4 spp)* – OIDN 1.0.0

SSIM: 0.9468

*Input buffers: HDR color, albedo, normal. Only the tone mapped color buffer is shown.

Rendered with Corona Renderer. Scene provided by Chaos Czech a.s. www.corona-renderer.com

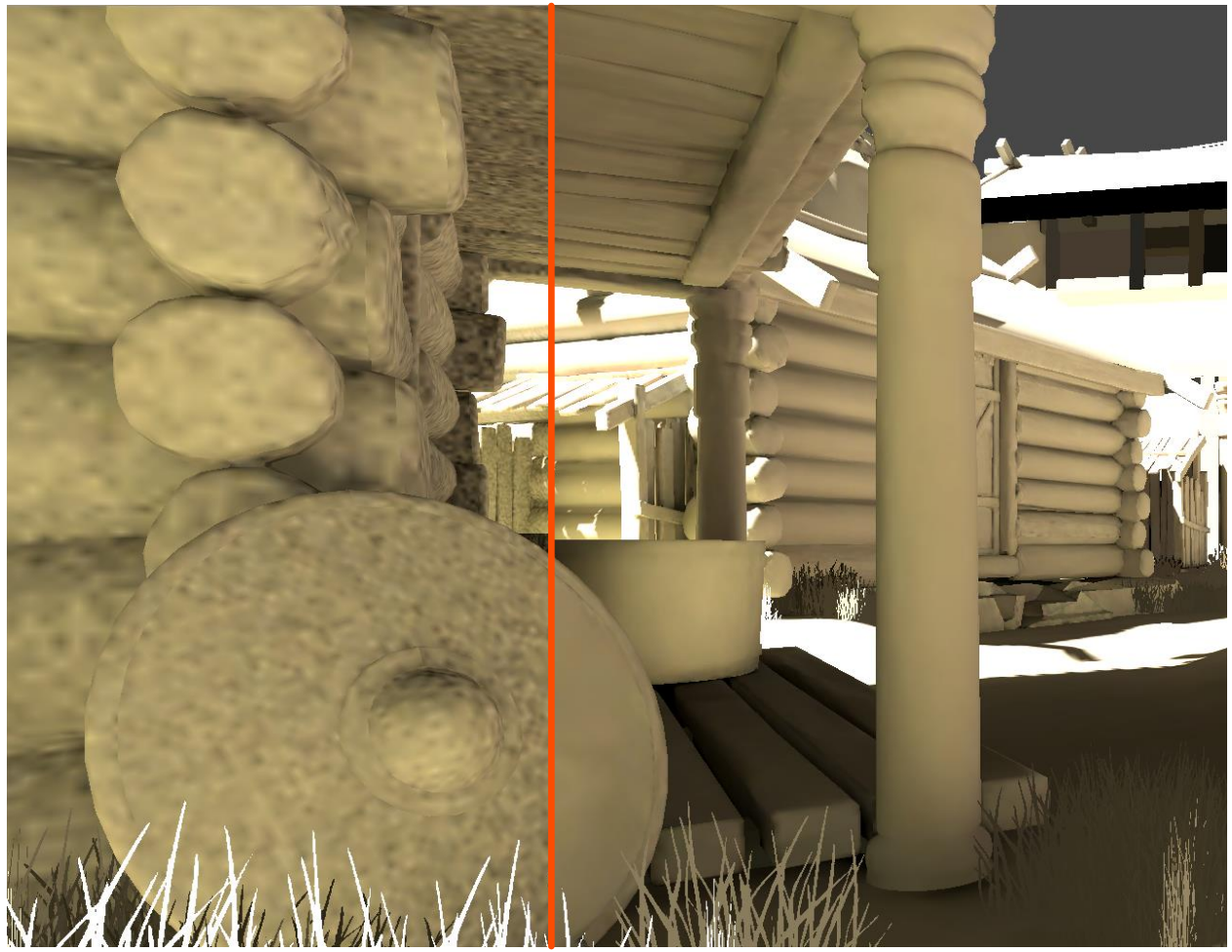
Corona Academy Interior *(HDR, 4 spp)* – Ground Truth

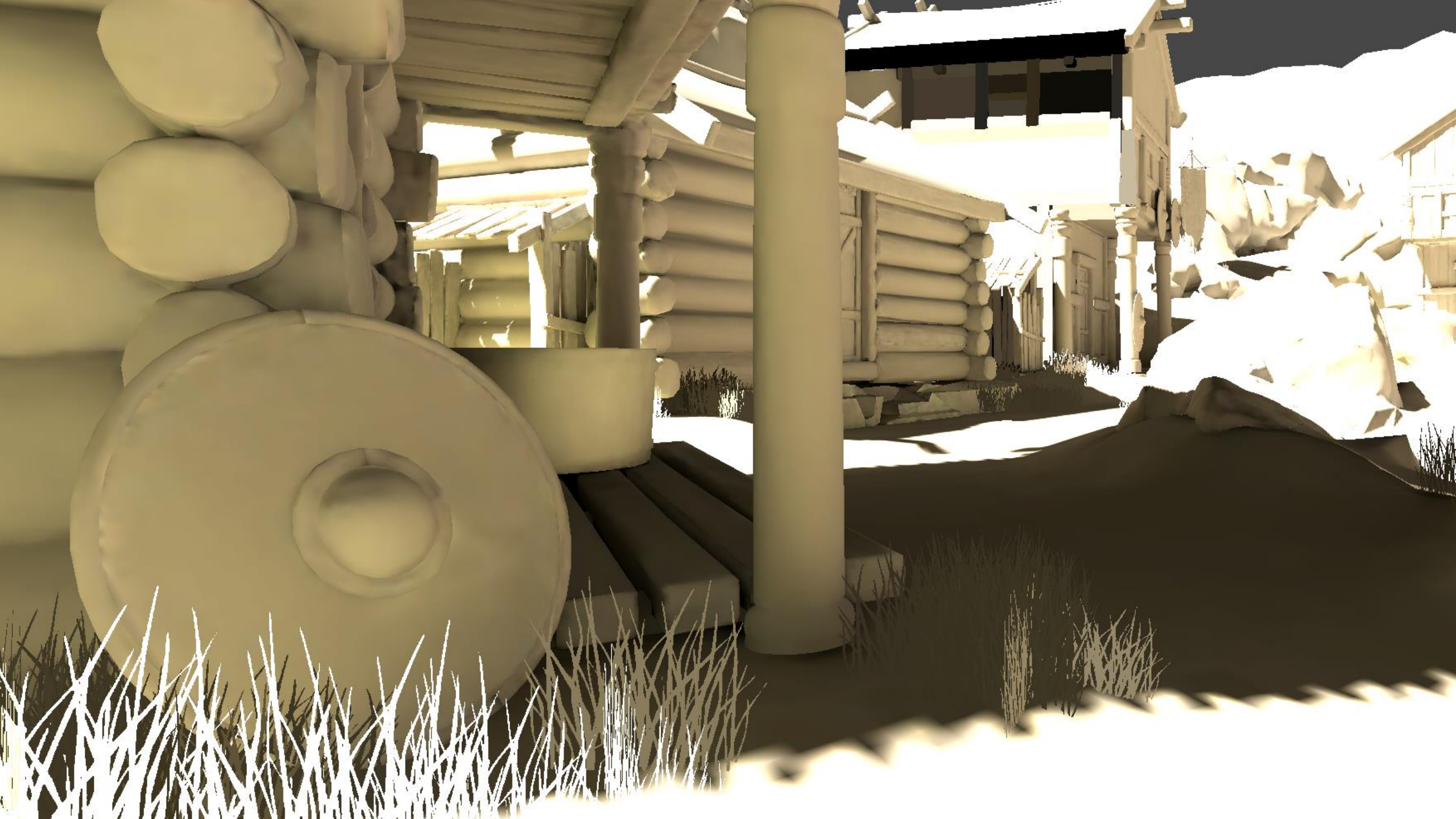
*Input buffers: HDR color, albedo, normal. Only the tone mapped color buffer is shown.
Rendered with Corona Renderer. Scene provided by Chaos Czech a.s. www.corona-renderer.com

corona

Denoising Light Maps in Unity

- Improved image quality over Gaussian filters
- Reduces number of samples required
- Not dependent on specific GPU hardware vendors





Open VKL

Open Image Denoise

OSPRay

OpenSWR

Embree

Data courtesy Bentley, Disney* *Other names and brands may be claimed as the property of others

Open **S**calable **P**ortable **Ray** Tracing Library

- Full rendering library for scalable CPU ray tracing
- Renderers ranging from photoreal path tracing to fast visualization
- Optimized for modern x86 CPUs
- Scalable from laptops up to multi-node supercomputers
  - GPU support coming!


Teradici PCoIP Client
File Panels Camera Help
OSPRay Bentley Motors Demo
BENTLEY
Vehicles:
Flying Spur - Mulliner
Flying Spur - Blackline
Continental GT - Mulliner
Scene Presets:
Default
GT Cockpit
Balcony
BirdsEye
LowDown
Window
Advanced:
Camera
Lighting
RenderSettings

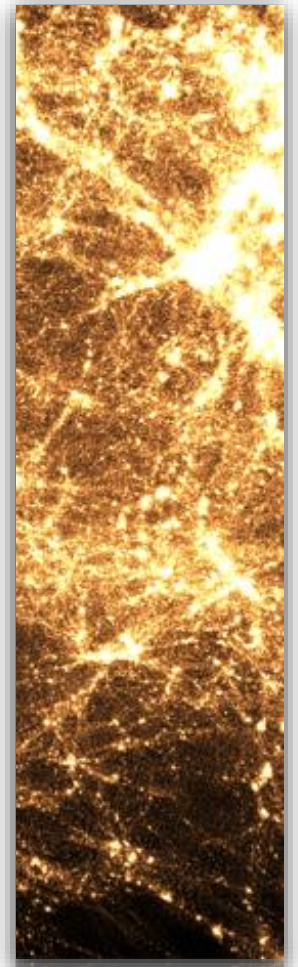

File Panels Camera Help
OSPRay Bentley Motors Demo
BENTLEY
Vehicles:
Flying Spur - Mulliner
Flying Spur - Blackline
Continental GT - Mulliner
Scene Presets:
Default
GT Cockpit
Balcony
BirdsEye
LowDown
Window
Advanced:
Camera
Lighting
RenderSettings
Lights - Advanced
Lights : Node
hdri : HDRILight
dir : 1.000 0.350 -0.140
intensity : 2.000
map : Texture2D
up : 0.000 1.000 0.000
sun : DirectionalLight
angularDiameter : 0.000
color : R:255 G:255 B:255
direction : -0.041 -1.000 -0.297
intensity : 5.000
Camera - Advanced
DoF None Less More
Aperture 0.050
Focus 2.450
FOV 48.000
Rendered with Intel® oneAPI Rendering Toolkit
The Bentley Winged "B" is a registered trademark of
Bentley Motors Ltd. Vehicle model used with permission.
San Miguel Villa © Guillermo M. Leal Llaguno


OSPRay: 0.203 fps | [==========>__]

Open VKL

Open Image Denoise

OSPRay



OpenSWR

Embree

Data courtesy Bentley, Disney* *Other names and brands may be claimed as the property of others

Open **S**oft**w**are **R**asterizer

- Implemented as a part of the MESA driver stack
- Intended for scalable software rendering of very large scenes (billions of triangles)
- Designed specifically for large CPUs found in HPC/Cloud environments
- Currently OpenGL 3.3 (4.0+ coming!)

Data courtesy Bentley, Disney* *Other names and brands may be claimed as the property of others

What is Embree?

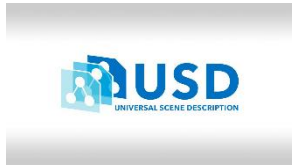
Embree is an open source software library that **solves the fundamental computations** required in ray tracing based rendering:

1. High-performance ray-based visibility queries
2. Building high-quality data structures over primitives

Embree Ray Tracing Kernels

- Mainly targets professional rendering applications
- Highly optimized ray tracing kernels (1.5x – 6x speedup)
- Provides rich functionality and flexibility
- Support for latest CPUs and ISAs (e.g. Intel® AVX-512)
- Windows*, macOS* 10.x, Linux* support
- API for easy integration into applications
- Open Source under Apache* 2.0 license
  www.embree.org

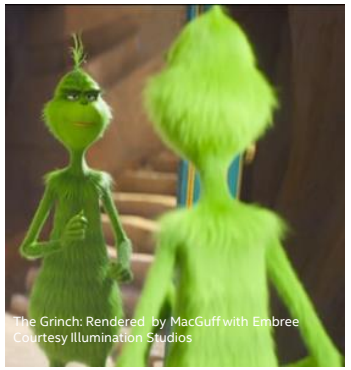
Embree Film Industry Adoption



NOTE: Embree is also widely deployed for HPC Scientific Visualization via inclusion with Intel OSPRay

Embree Film Industry Adoption


Batman: Rendered by Glimpse with Embree
Courtesy Animal Logic

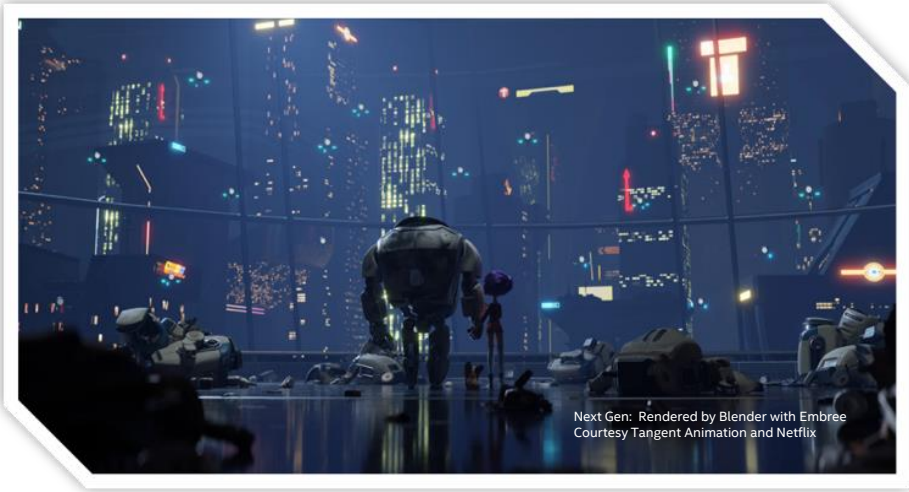

The Grinch: Rendered by MacGuff with Embree
Courtesy Illumination Studios


The Secret Life of Pets 2: Rendered by MacGuff with Embree. Courtesy Illumination Studios


Spider-Man: Far From Home: Rendered by Freak Renderer (with Embree)
Courtesy Framestore

Rendered by CGI Studio with Embree
Courtesy Blue Sky Studios


Abominable: Scene rendered by MoonRay (with Embree):
Property of DreamWorks Animation

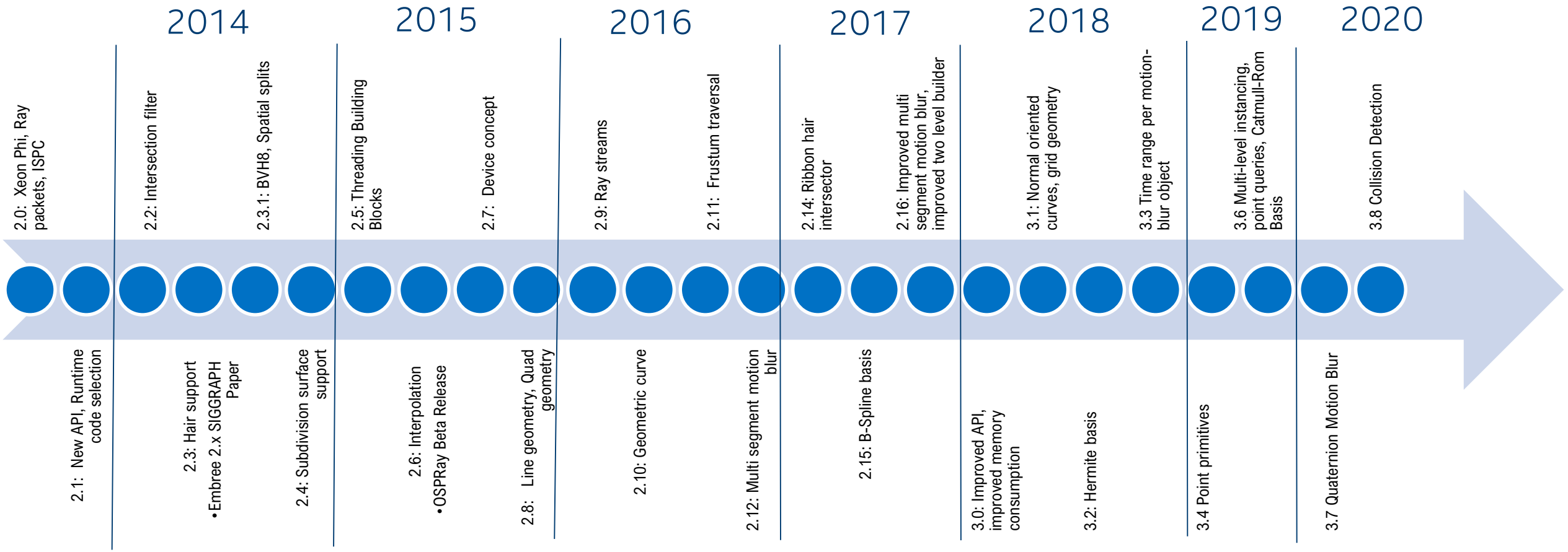

Next Gen: Rendered by Blender with Embree
Courtesy Tangent Animation and Netflix

How Embree achieves its performance

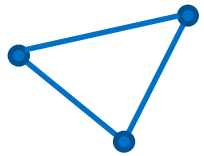
- **Using latest state-of-the-art ray tracing algorithms**
  - High-quality BVH build which is well parallelized using Intel® TBB
  - Wide BVHs, single ray traversal, hybrid ray traversal, …

- **Optimized implementation with respect to hardware**
  - Vectorize where possible to exploit SIMD and other special instructions
  - Reduce instruction dependency chain in inner most loop
  - Implement fast path for common cases
  - Optimize data structures for cache usage, memory access pattern, …

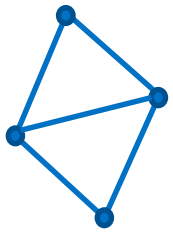
Embree Timeline



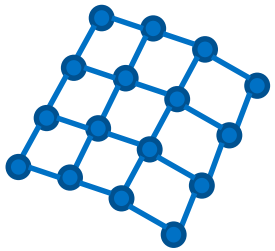
Embree Features



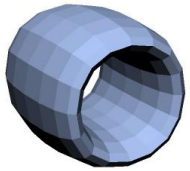
Triangles



Quads



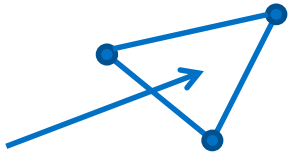
Grids



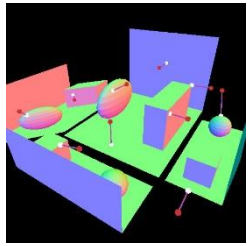
Catmull-Clark
Subdivision Surfaces

Flat, Round and Normal Oriented Curves
Linear, Bezier, B-Spline, Hermite, Catmull-Rom

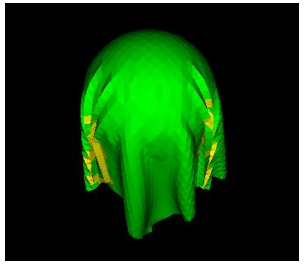
Hair, Fur, Complex Lines



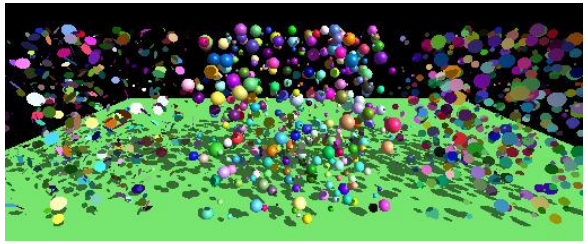
Ray Queries



Nearest Point Queries

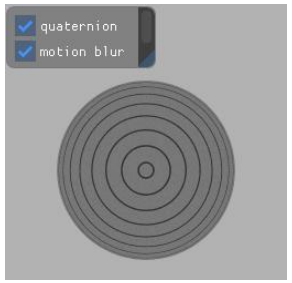


Collision Queries

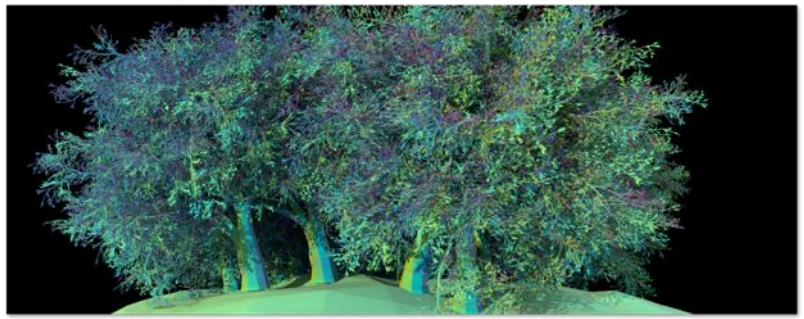


Ray-Oriented Disks, Spheres,
and Normal-Oriented Disks

Multi-Segment Motion Blur



Quaternion Motion Blur for Instances



Multi-Level Instancing for Dramatic Memory Savings

Embree System Overview

Embree API (C99 and ISPC)

Ray Tracing Kernel Selection

Acceleration Structures	Builders	Traversal	Intersection	Subdiv Engine
bvh4.triangle4 bvh8.triangle4 bvh4.quad4v ...	SAH Builder MBlur Builder Spatial Split Builder Morton Builder BVH Refitter	Single Ray Packet/Hybrid Ray Stream Point Query Collision	Möller-Trumbore Plücker Flat Curve Round Curve Oriented Curve Grid	B-Spline Patch Gregory Patch Tessellation Cache Displ. Mapping

Common Vector and SIMD Library

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

Embree API

- Version 3 of the Embree API
- Object-oriented and reference-counted
- Compact and easy to use
- Hides implementation details
- For details visit https://embree.org/api.html

Example: Scene creation

- Device object encapsulates Embree state
- Scene contains a vector of geometries
- Scene geometry changes have to get committed to trigger BVH build

```
// include Embree headers
#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);
}
```

Example: Triangle Mesh creation

- 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 { uint 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

```
// 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);

// detect miss
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;
```

Embree in Games: Light Baking

- Precomute Lighting for static geometry
- Bake lighting into special light texture or vertices
- Typically full path tracing algorithm
  - Ray query performance is important

➔ Embree used in lightmapper of Activision

Embree Dynamic Scenes

- Embree includes highest performance BVH builders to handle dynamic content
  - Morton BVH builder (16 Mprims/core/s*)
  - BVH refit (66 Mprims/core/s*)
  - Partial scene updates at mesh granularity

➔ Sufficient BVH build performance to use library during game play!

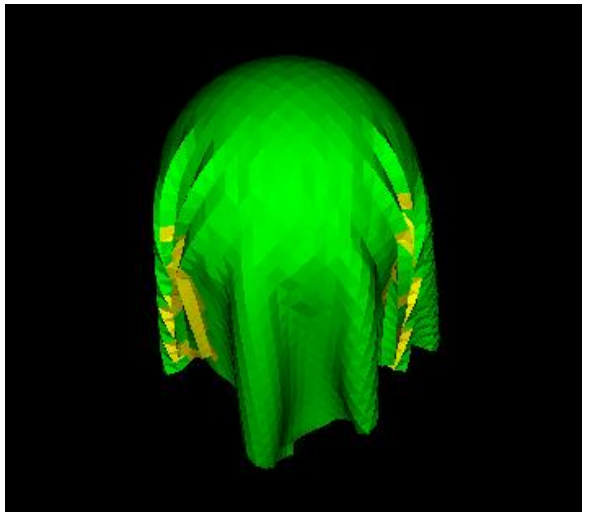
* Measured on a single core of an Intel® Xeon® Platinum 8180 CPU

Embree in Games: Ray-Based Collision Detection

- CPU ray query application in games
  - Gun shots
  - Character placement
  - Simple character collision detection

Embree For Advanced Collision Detection

- Embree provides fast parallel collision detection implementation
- Callback invoked with potentially colliding primitive pairs
- Coarse phase only, narrowing need to be done by application
- Performance of up to 50 M pairs/s/core*



* Measured on a single core of an Intel® Xeon® Platinum 8180 CPU

Embree in Games: BVH Build API

- Supports high quality build (SAH heuristic)
- Supports high performance build (Morton codes)
- Branching factor and heuristic configurable
- Full control over BVH data layout through callbacks
- Effective parallelization using Intel® Threading Building Blocks

Embree BVH Build in World Of Tanks

- Great multi-core performant BVH build on CPU using Intel® Embree
- Software ray tracing on the GPU to calculate high quality pixel-precise shadows of tanks
- Great example of hybrid CPU/GPU use
- Feature will be available in future version of World of Tanks

World of Tanks Shadow Mapping

World of Tanks Raytraced shadows

Embree Integration into SYCL

- SYCL as Programming Language
  - Single source C++11 extension for OpenCL
  - Khronos Open Standard: https://www.khronos.org/sycl/
  - Programming language of Intel® oneAPI
- Embree integration into SYCL coming
  - Multi-platform including CPU and GPU support
  - High performance through fine tuned traversal algorithms


intel
experience
what's inside™