Raytracing with Embree

Florian Reibold
Sebastian Herholz
Johannes Meng

Graphics Software Engineers

Content / Agenda

- Raytracing and Embree basics
- Demo1: Ray-Triangle intersection
- Embree’s geometry types
- Demo2: User geometries
- Advanced Embree Features
- Demo3: Point queries
- Q&A

Raytracing with Embree

- Library for raytracing
- Mainly targets professional rendering applications
- High performance (1.5x – 6x speedup)
  - Multithreading using Intel® TBB
  - Heavily utilizes SIMD instructions (e.g. Intel® AVX-512)
- Easy to use API
- Open source (Apache* 2.0 license)
- Cross platform (Windows*, Linux*, macOS*)

Raytracing with Embree

AUTODESK

DREAMWORKS

V-ray

3DEXCITE

blender

corona

FluidRay Physically Based Renderer RT

UBISOFT MOTION PICTURES

EasternGraphics visualize your business

pCon PLANNER

4 TERRAGEN

VISIT

ParaView

vtk

Simlab Soft Bringing Art closer

SURVICE ENGINEERING COMPANY

STINGRAY

VMD Visual Molecular Dynamics

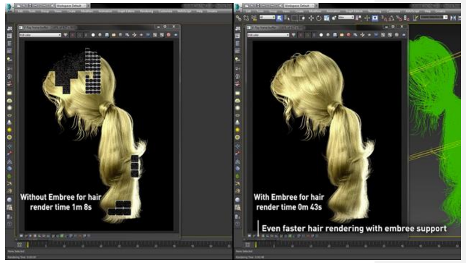
VALVE

REDWAY3D

Cinema4D

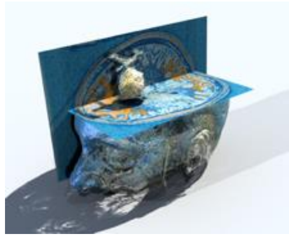
THEA RENDER

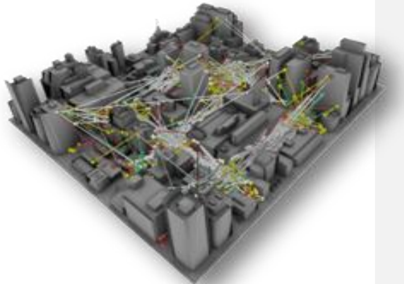
DWA How To Train Your Dragon 2

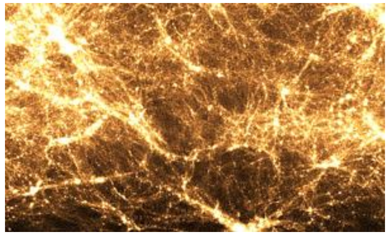

Without Embree for hair render time 1m 8s
With Embree for hair render time 0m 43s
Even faster hair rendering with embree support
V-Ray Embree Hair Primitives

CPU/Embree Only Corona Renderer

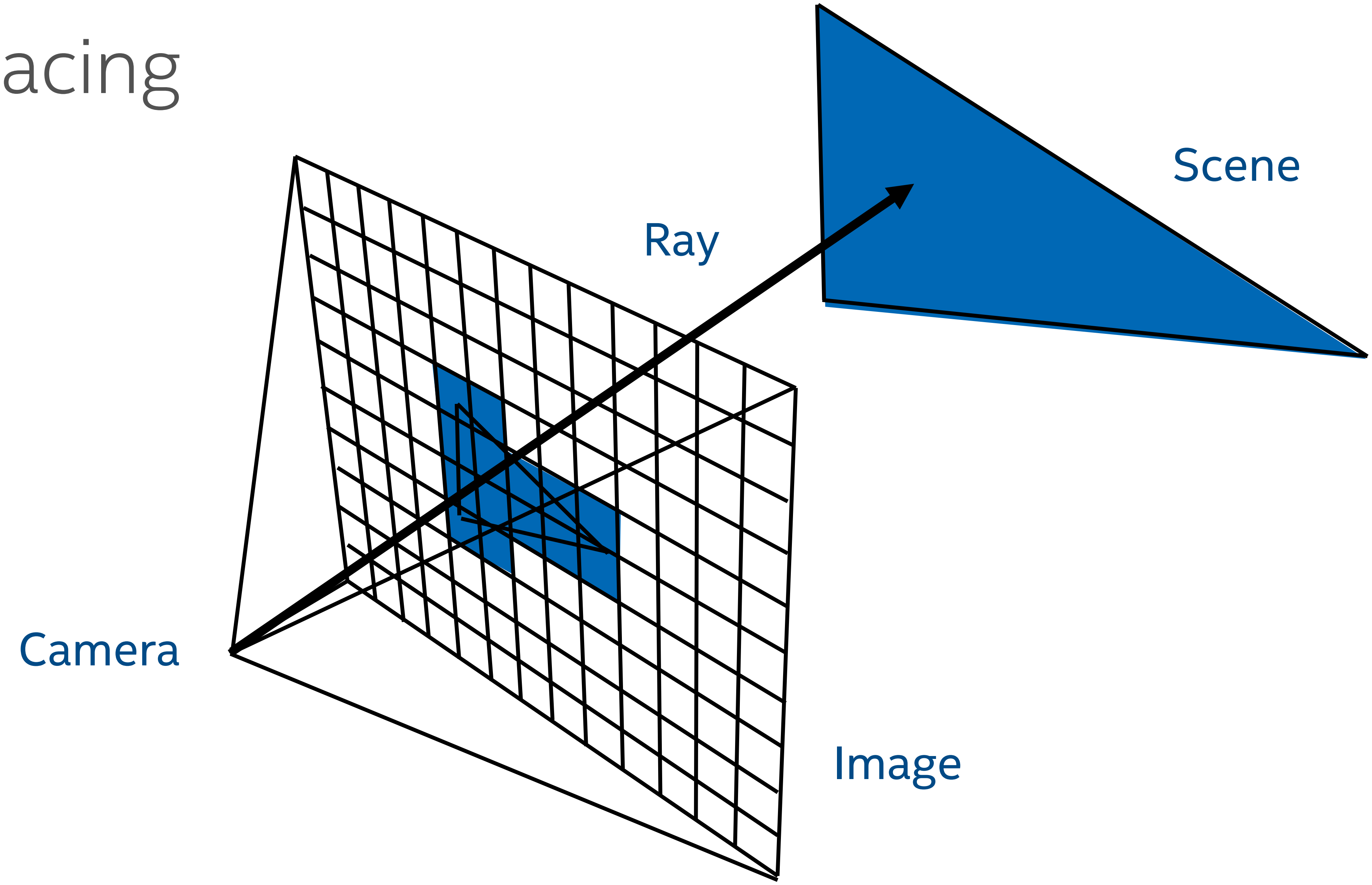
ADSK 360 Cloud – >50M Renders


ParaView with OSPRay


SURVICE StingRay

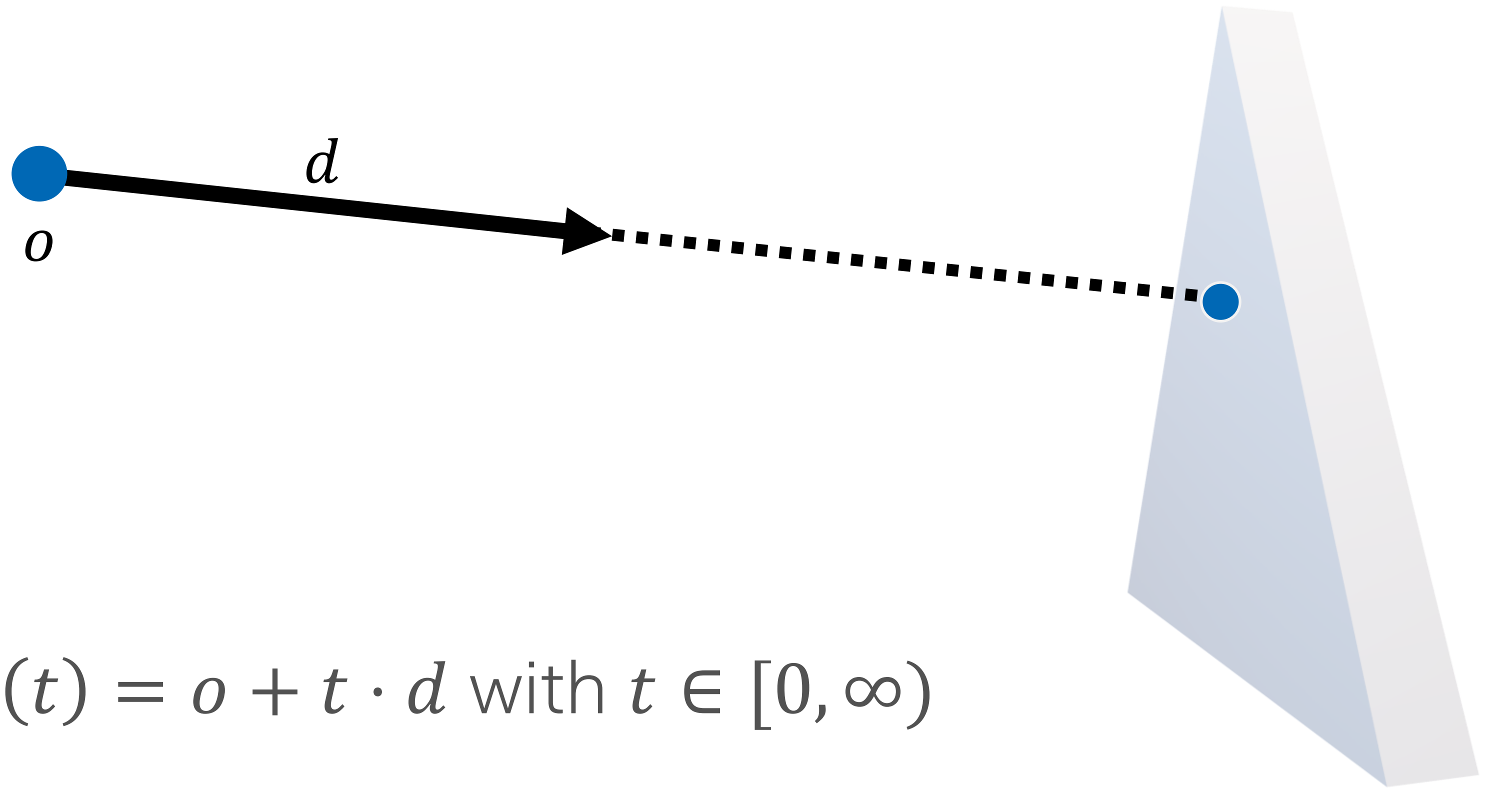

ANL VL3 Dark Matter - OpenSWR

Raytracing



Raytracing

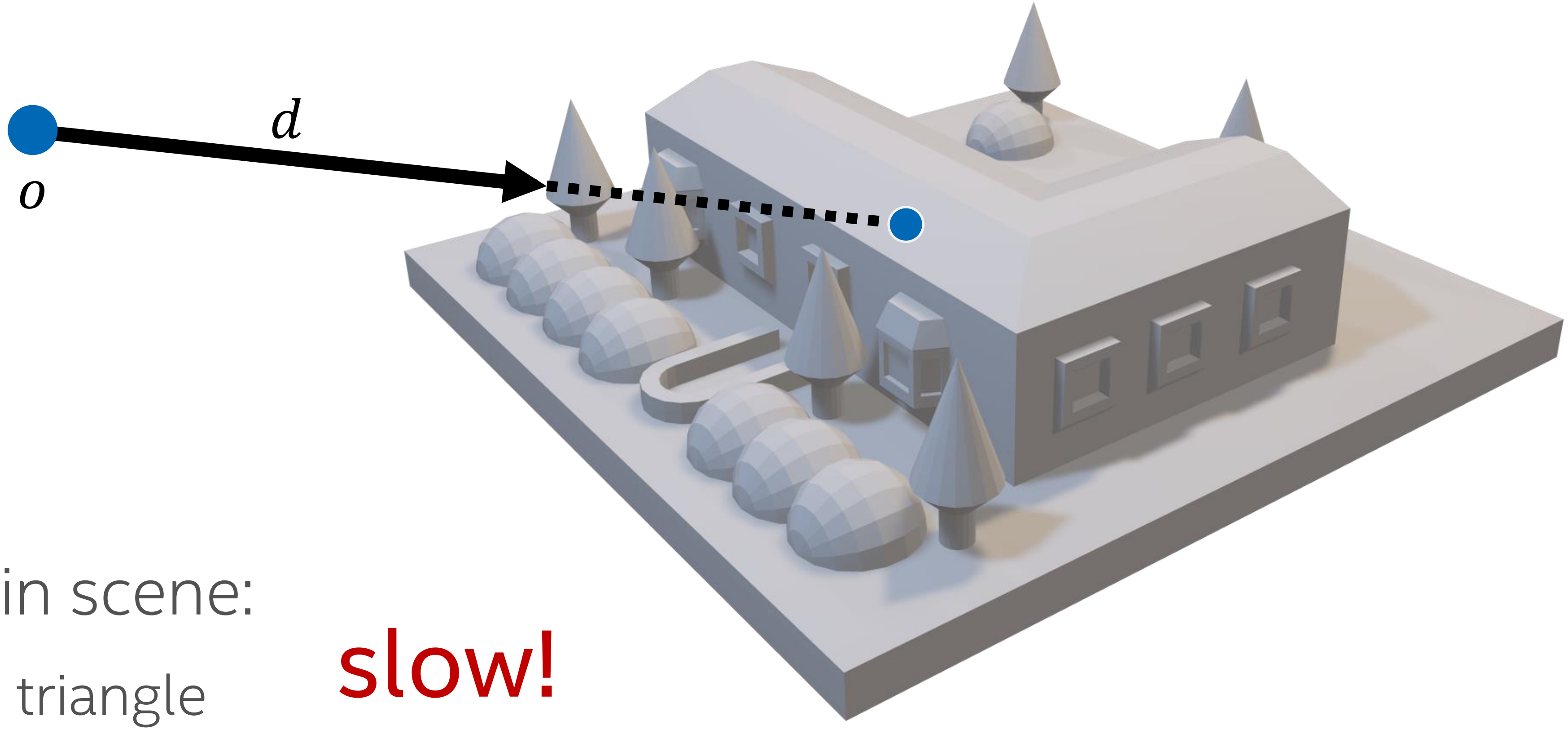
- Find intersection of a ray with the virtual scene



- Ray: All points $r(t) = o + t \cdot d$ with $t \in [0, \infty)$

Raytracing

- Find intersection of a ray with the virtual scene

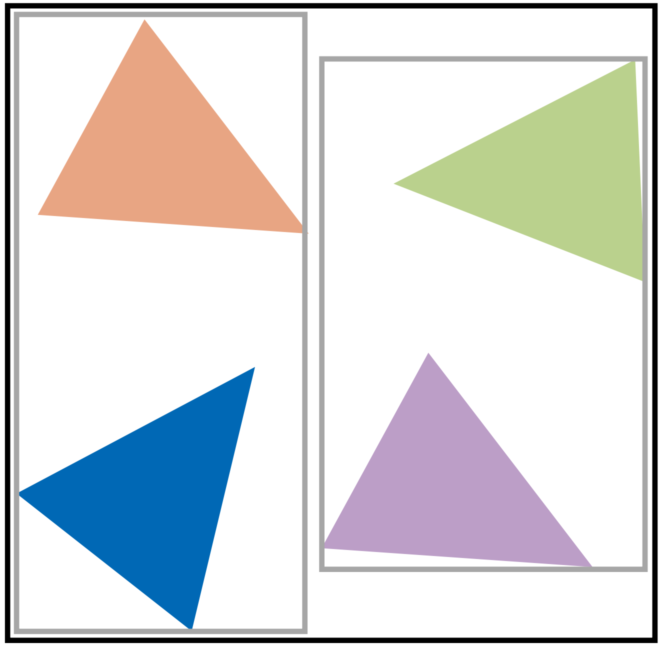


- For all triangles in scene:
  - Intersect ray with triangle

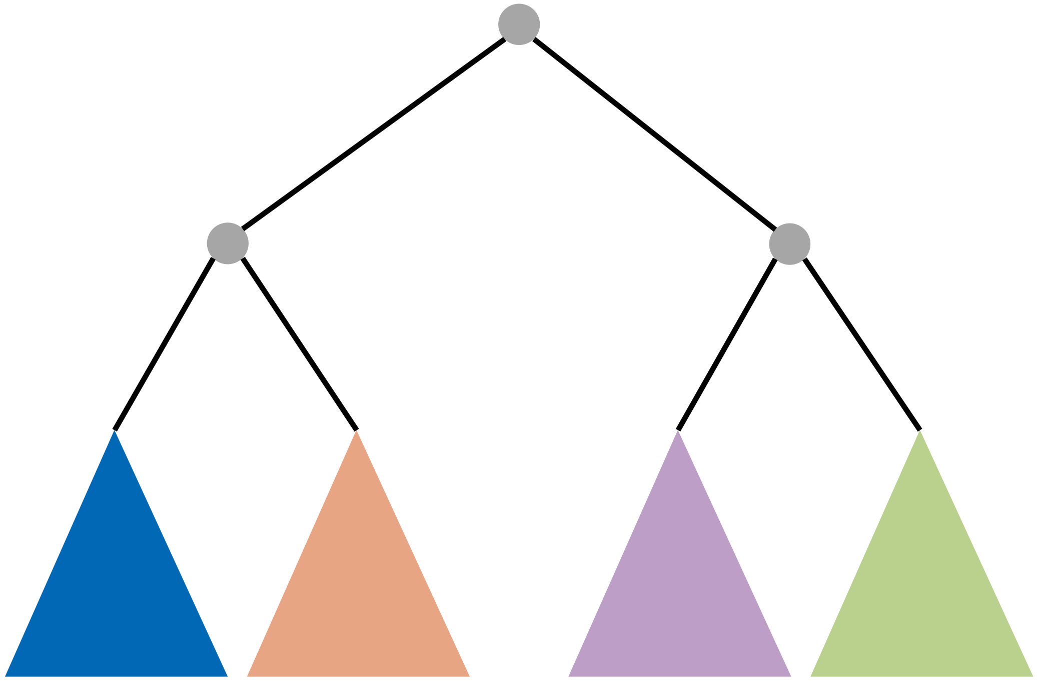
slow!

Bounding Volume Hierarchies (BVH)

- Hierarchical “clustering” of close-by geometry in boxes



scene

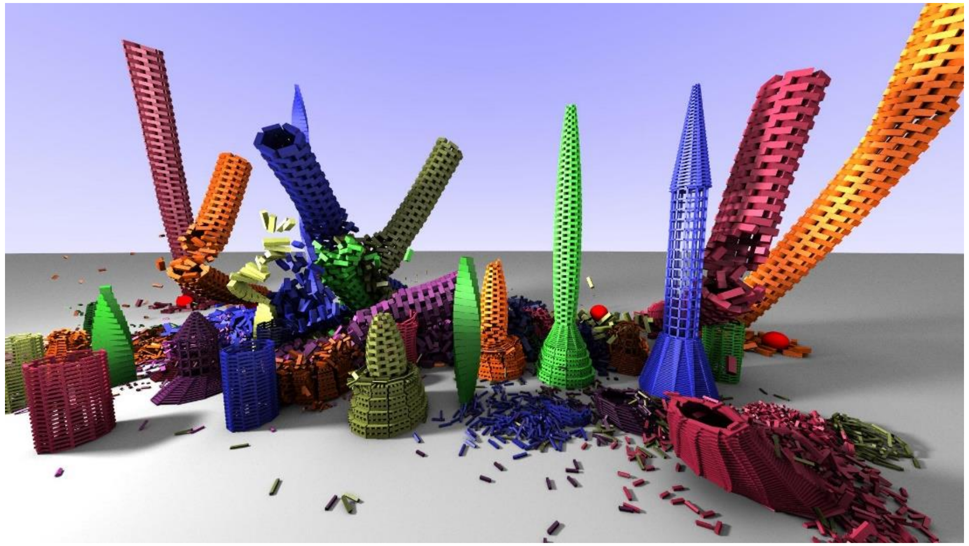


tree structure

Bounding Volume Hierarchies (BVH)

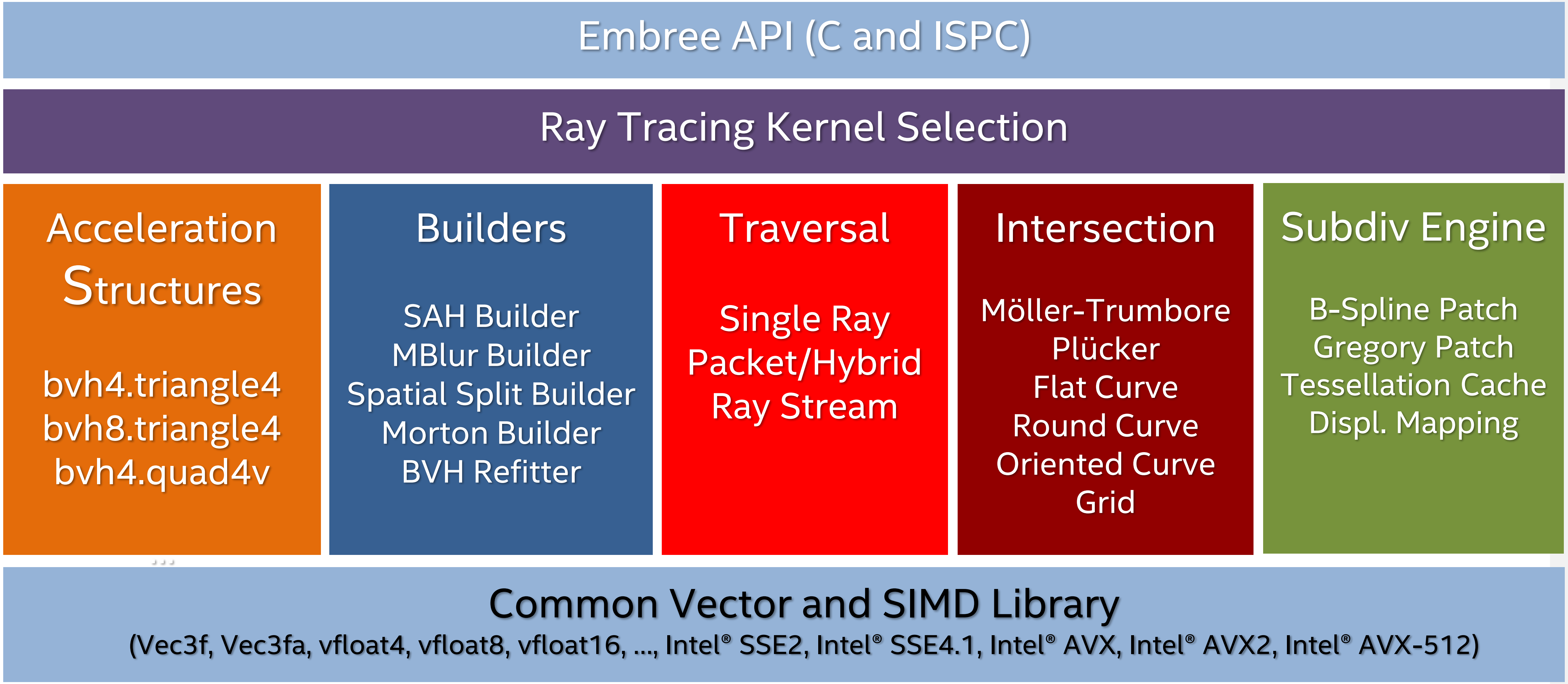
Challenges

- Multi-threaded build
- Vectorization
- Numerical robustness
- Cross-platform availability



Embree solves all of that for you, and more!

Embree System Overview



Embree API (C and ISPC)

Ray Tracing Kernel Selection

Acceleration Structures

bvh4.triangle4
bvh8.triangle4
bvh4.quad4v
...

Builders

SAH Builder
MBlur Builder
Spatial Split Builder
Morton Builder
BVH Refitter

Traversal

Single Ray
Packet/Hybrid
Ray Stream

Intersection

Möller-Trumbore
Plücker
Flat Curve
Round Curve
Oriented Curve
Grid

Subdiv Engine

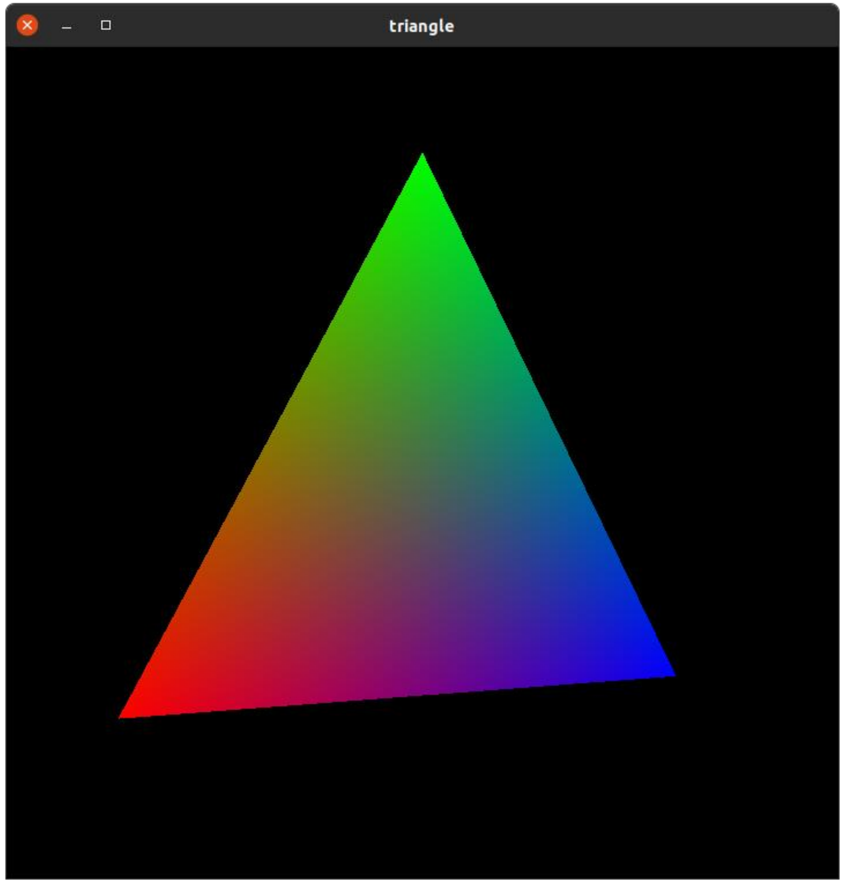
B-Spline Patch
Gregory Patch
Tessellation Cache
Displ. Mapping

Common Vector and SIMD Library

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

Demo 1

Ray-Triangle intersection with Embree



A Ray in Embree

- Input parameters:
  - Origin (org) and direction (dir)
  - Ray interval (tnear, tfar)
  - Time (for motion blur, advanced)

```
struct RTCRay
{
  Vec3f org;
  Vec3f dir;
  float tnear;
  float tfar;
  float time;

  Vec3f Ng;
  float u;
  float v;
  int geomID;
  int primID;
  int instID;
}
```

A Ray in Embree

- Output parameters:
  - Hit distance (tfar)
  - Normal (Ng)
  - Local hit coordinates (u, v)
  - Geometry identifier (geomID)
  - Index of primitive of geometry (primID)
  - Geometry identifier of hit instance (instID, advanced)

```
struct RTCRay
{
  Vec3f org;
  Vec3f dir;
  float tnear;
  float tfar;
  float time;

  Vec3f Ng;
  float u;
  float v;
  int geomID;
  int primID;
  int instID;
}
```

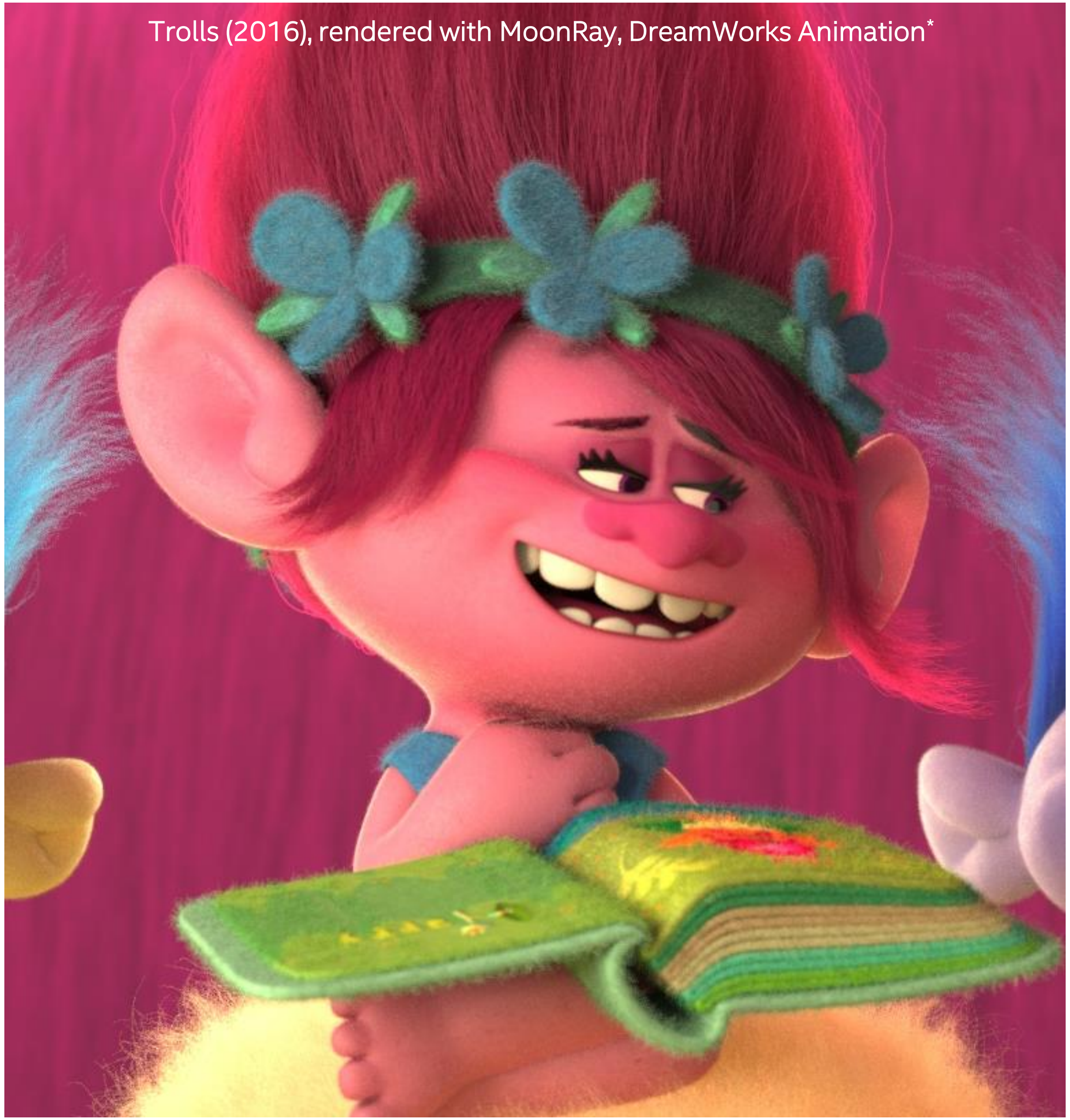
Prerequisites

- Embree device encapsulates
  - ISA configuration,
  - number of threads,
  - logging verbosity, …
- Embree scene
  - Collection of geometries that can be intersected
- Create/Release pairs

```
// 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 ...

  // commit changes
  rtcCommitScene(scene);

  // trace rays ...

  // release objects
  rtcReleaseScene(scene);
  rtcReleaseDevice(device);
}
```

Geometry Types

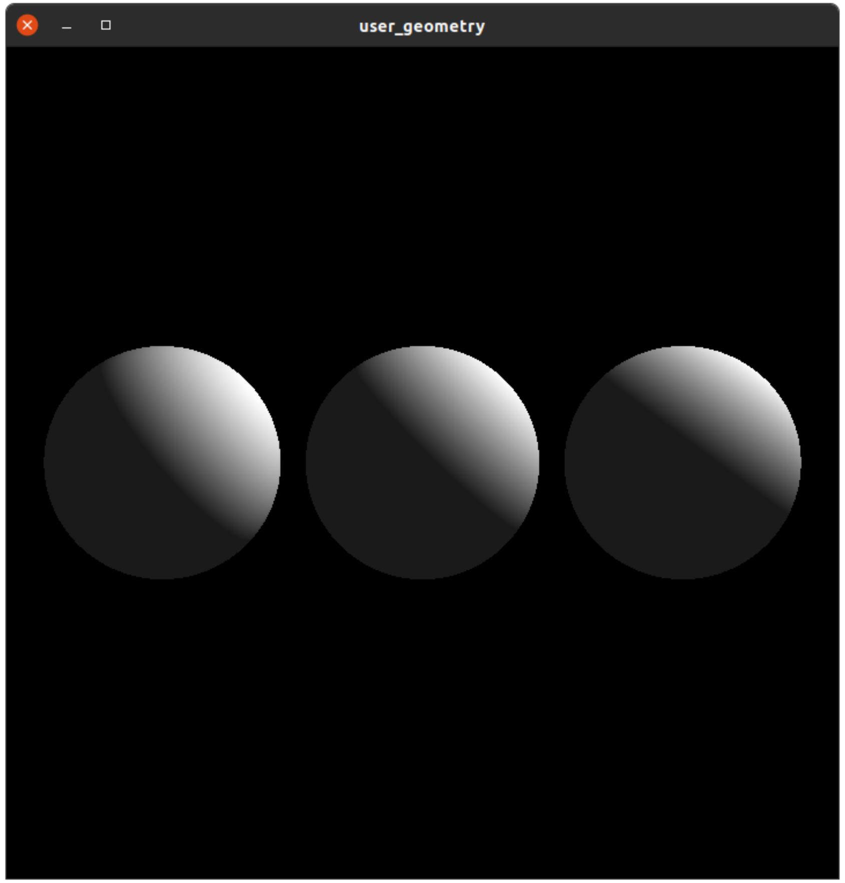
- Triangle meshes
- Quad meshes
- Grid meshes
- Subdivision meshes
- Curves
- User-defined → extensible



Trolls (2016), rendered with MoonRay, DreamWorks Animation*

Demo 2

User-Defined Geometries



User-Defined Geometries

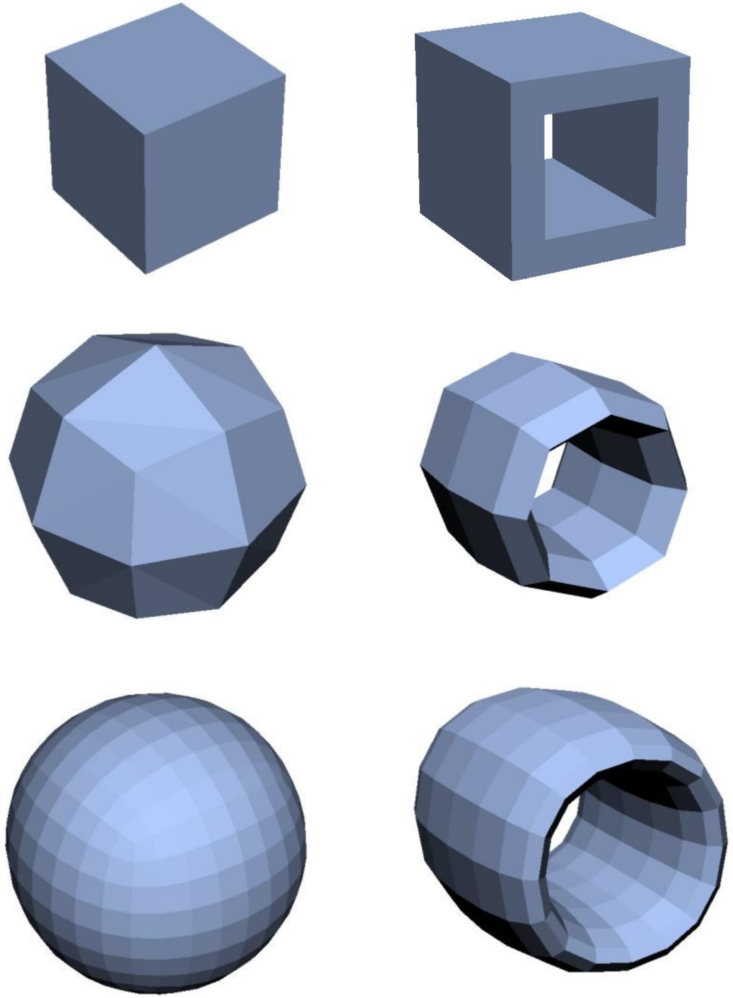
- Enables implementing custom primitives and features
- User provides
  - Bounding function
  - Intersect and occluded functions

- Example: Implementing analytical spheres
  - Sphere: all points where $(x - c)^2 - r^2 = 0, x \in \mathbb{R}^3$
  - Intersections: solve $(o + t \cdot d - c)^2 - r^2 = 0$ for t

Selected Advanced Features

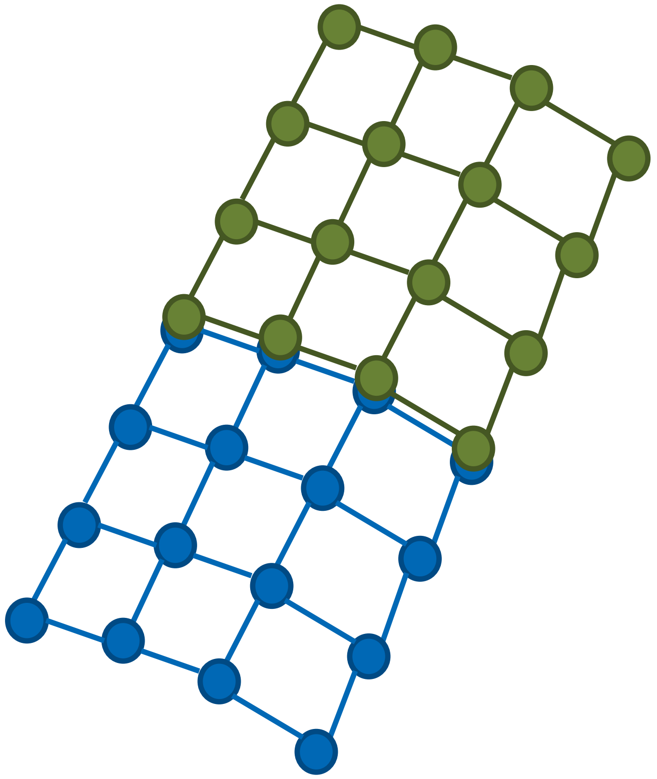
Catmull-Clark Subdivision Surfaces

- Converts coarse mesh into smooth surface (subdivision)
- Established as standard in movie production
- Embree implementation compatible with OpenSubdiv 3.0
- Evaluation of surface supported
- Walking mesh topology supported



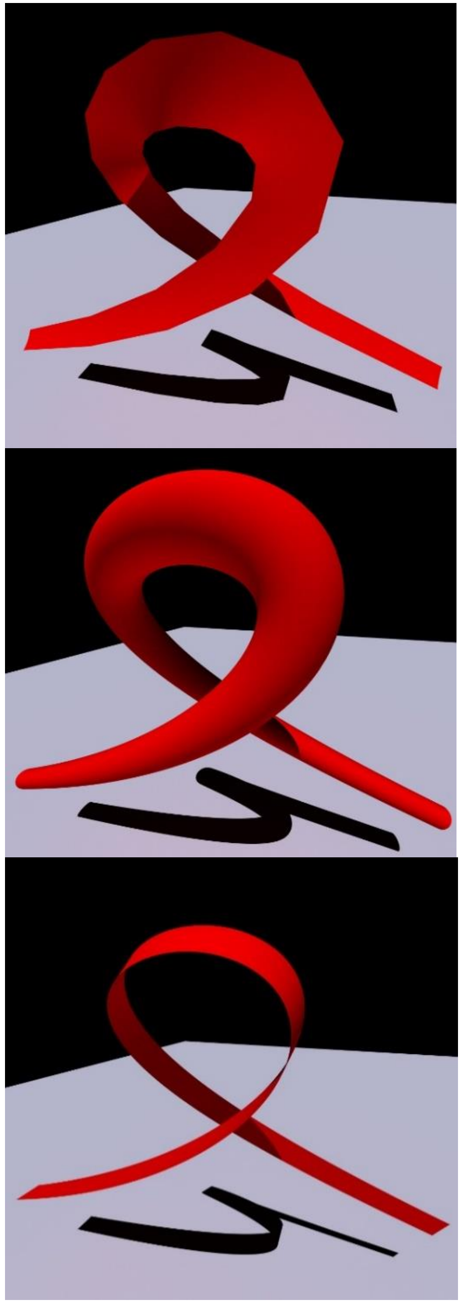
Grid Meshes

- 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
- Can use grid with displacement function instead of subdiv mesh



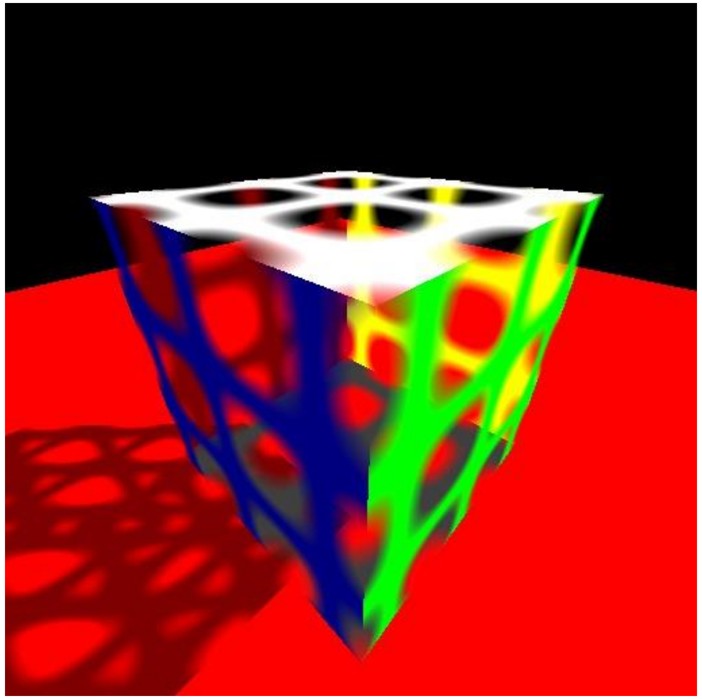
Curve Geometries

- 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)



Intersection Filter Functions

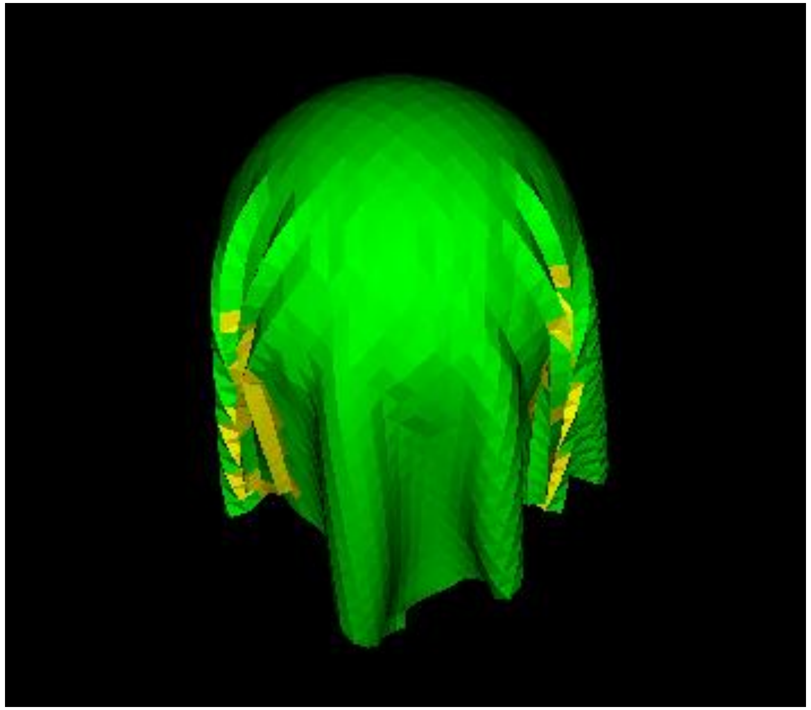
- Per-geometry callback
  - Called during traversal for each primitive intersection
- Callback can **accept** or **reject** hit
- Can be used for:
  - Trimming curves (e.g. modeling tree leaves)
  - Transparent shadows (reject and accumulate)
  - Find all hits (reject and collect)
  - Advanced self-intersection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



Collision Detection

- Fast parallel collision detection implementation
- Callback invoked with potentially colliding primitive pairs
- Coarse phase only, narrowing needs 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

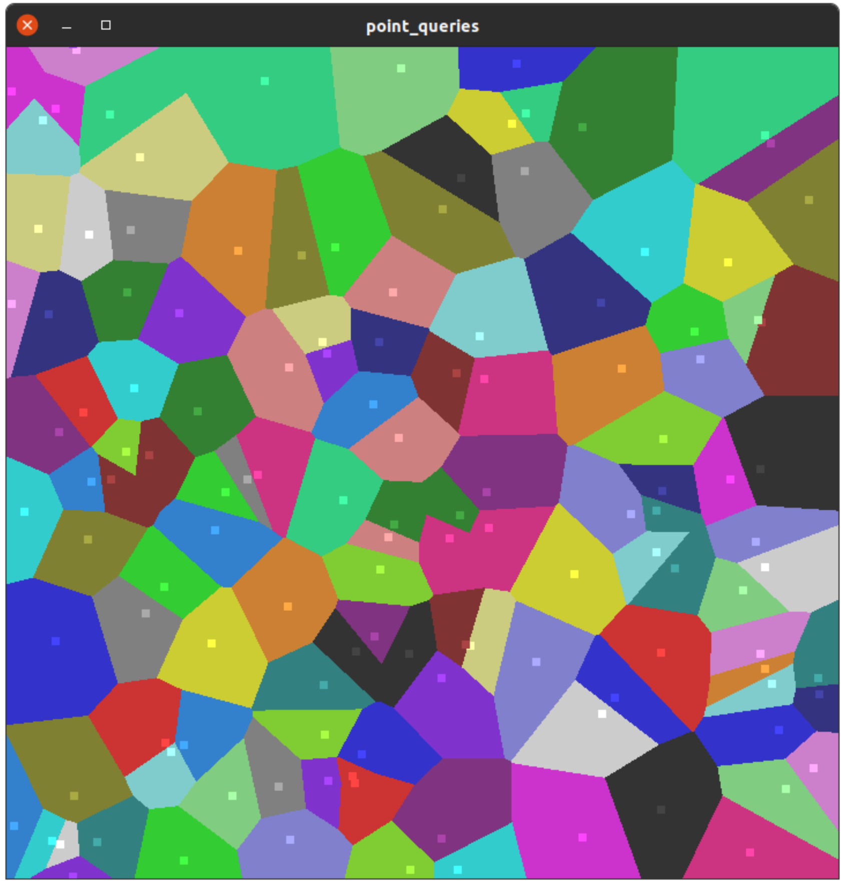


Point Queries

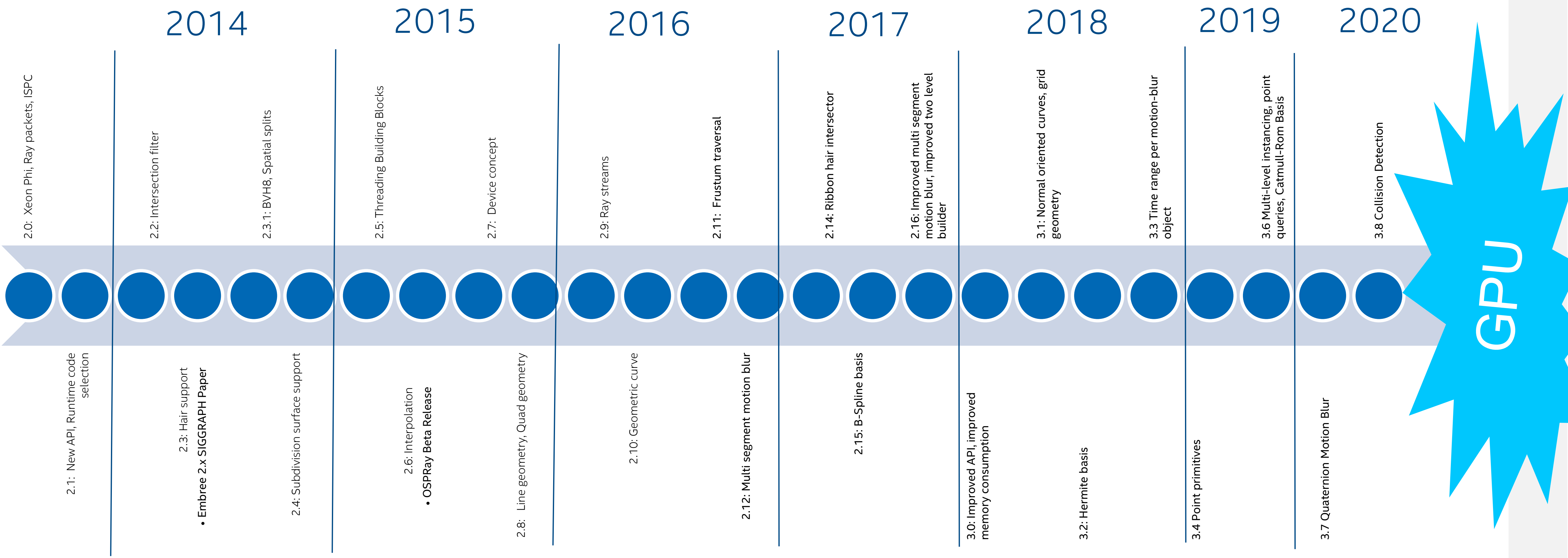
- Allows the traversal of the BVH with a point instead of a ray
  - Can be used for nearest neighbor lookups in point clouds or
  - Find the closest point on the geometry with respect to the query position
- User provides Primitive-Point distance computation in callback function

Demo 3

Point Queries



Embree Timeline and Outlook



2014

2015

2016

2017

2018

2019

2020

2.0: Xeon Phi, Ray packets, ISPC

2.1: New API, Runtime code selection

2.2: Intersection filter

2.3: Hair support

- Embree 2.x SIGGRAPH Paper

2.3.1: BVH8, Spatial splits

2.4: Subdivision surface support

2.5: Threading Building Blocks

2.6: Interpolation

- OSPRay Beta Release

2.7: Device concept

2.8: Line geometry, Quad geometry

2.9: Ray streams

2.10: Geometric curve

2.11: Frustum traversal

2.12: Multi segment motion blur

2.14: Ribbon hair intersector

2.15: B-Spline basis

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

3.3 Time range per motion-blur object

3.4 Point primitives

3.6 Multi-level instancing, point queries, Catmull-Rom Basis

3.7 Quaternion Motion Blur

3.8 Collision Detection

GPU

Q&A

Raytracing with Embree

intel®

Notices and Disclaimers

- 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.
- You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.
- 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 complete information visit www.intel.com/benchmarks.
- 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.
- Intel, Core and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others
- © Intel Corporation.