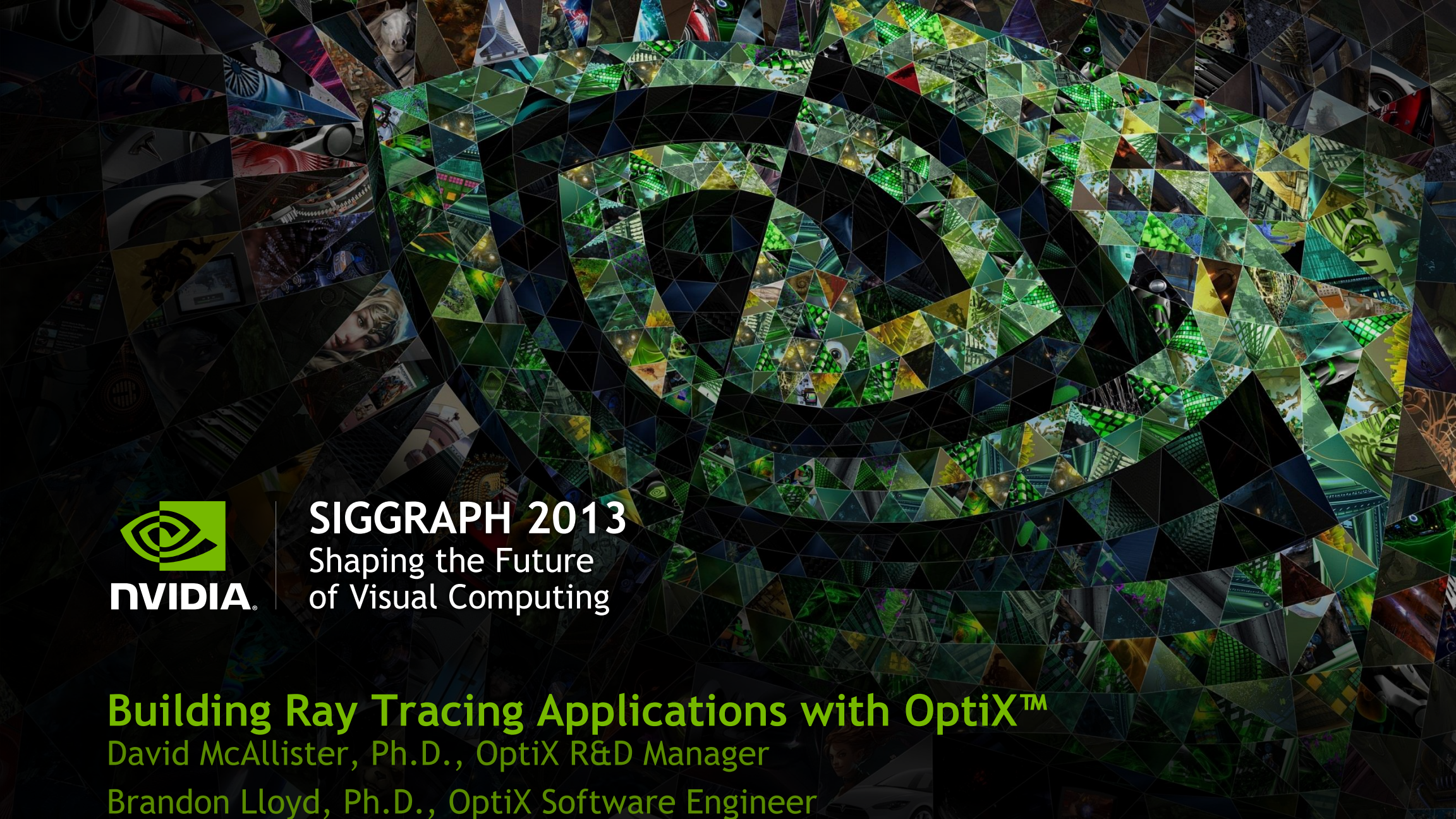


SIGGRAPH 2013
Shaping the Future
of Visual Computing

Building Ray Tracing Applications with OptiX™

David McAllister, Ph.D., OptiX R&D Manager

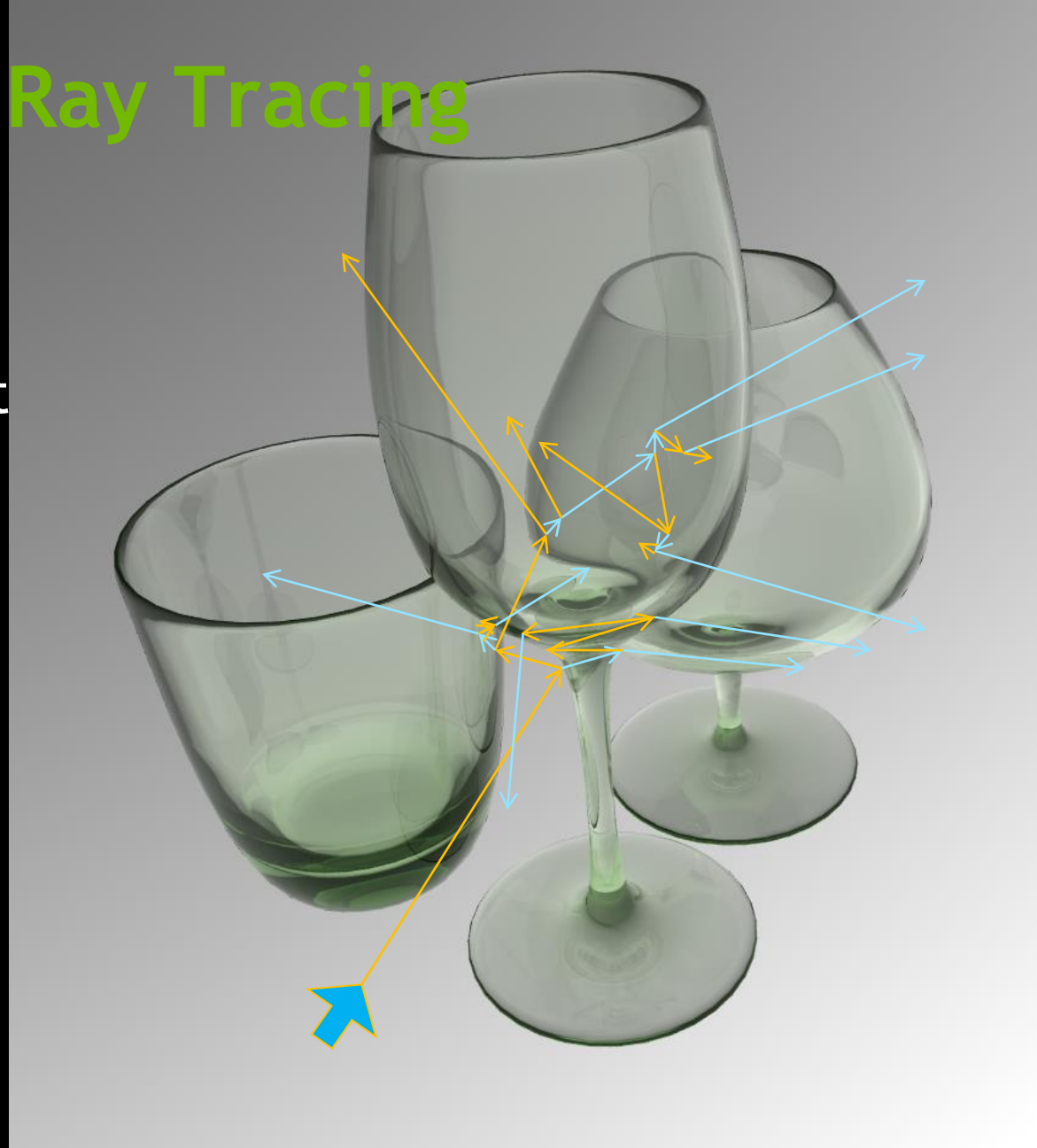
Brandon Lloyd, Ph.D., OptiX Software Engineer

Why ray tracing?

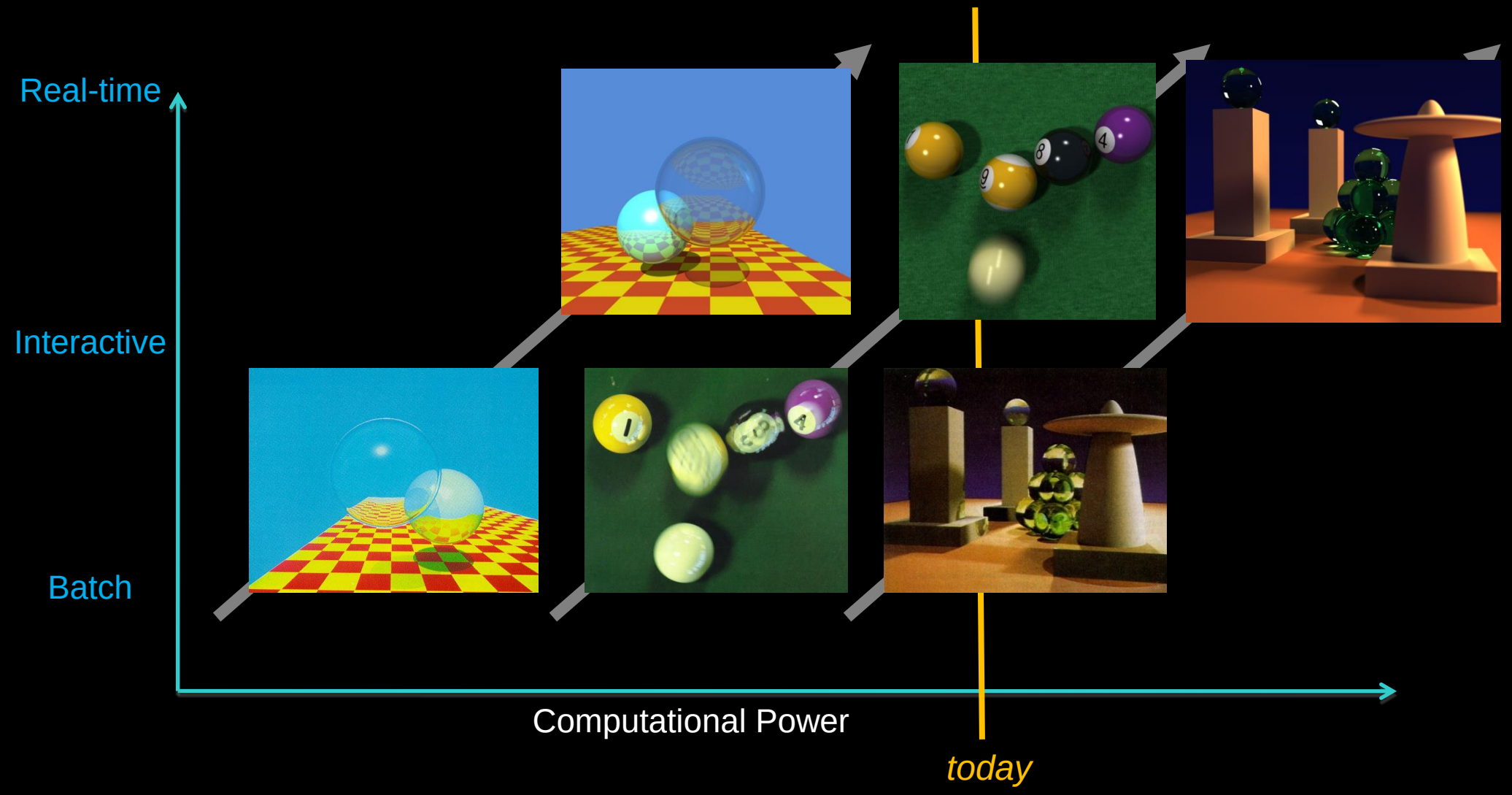
- Ray tracing unifies rendering of visual phenomena
  - fewer algorithms with fewer interactions between algorithms
- Easier to combine advanced visual effects robustly
  - soft shadows
  - indirect illumination
  - transparency
  - reflective & glossy surfaces
  - subsurface scattering
  - depth of field

Recursive Ray Tracing

- Whitted-style recursive
- Reflection and refraction per hit
- Beer's Law attenuation
- Depth cut-off
- Importance cut-off



Ray Tracing Algorithms



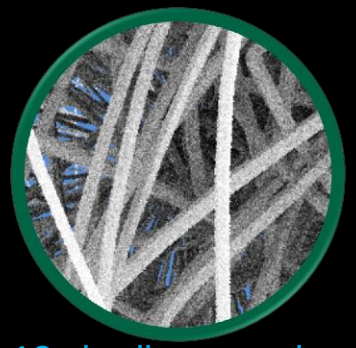
Real Time Path Tracing

- What would it take?
  - 4 rays / sample
  - 50 samples / pixel
  - 2M pixels / frame
  - 30 frames / second
  - **= 12B rays / second**
- GeForce GTX 680: 350M rays / second
  - Need 35X speedup

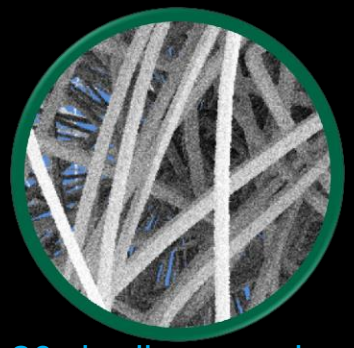
Good enough for games

1 shading sample
1 AA sample

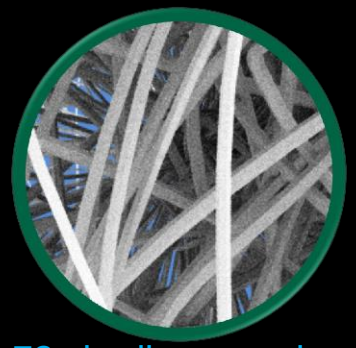
9 shading samples
1 AA sample



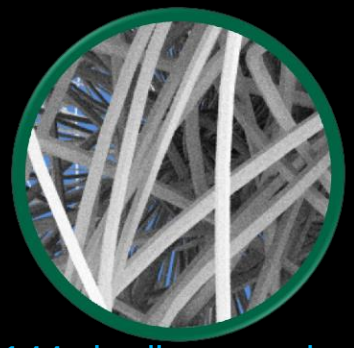
18 shading samples
2 AA samples



36 shading samples
4 AA samples



72 shading samples
8 AA samples



144 shading samples
16 AA samples

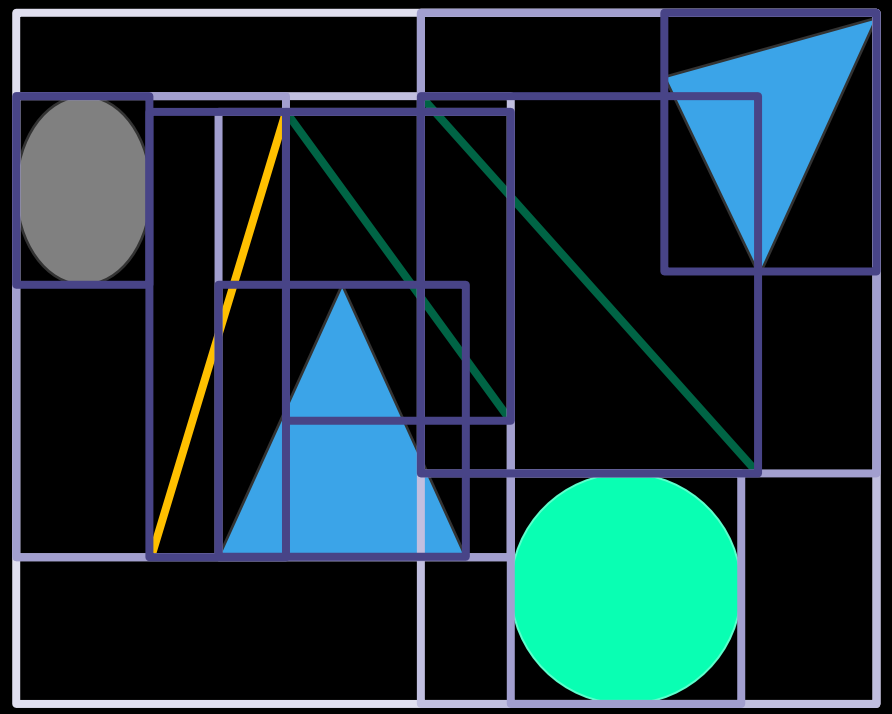
How to optimize ray tracing (or anything)

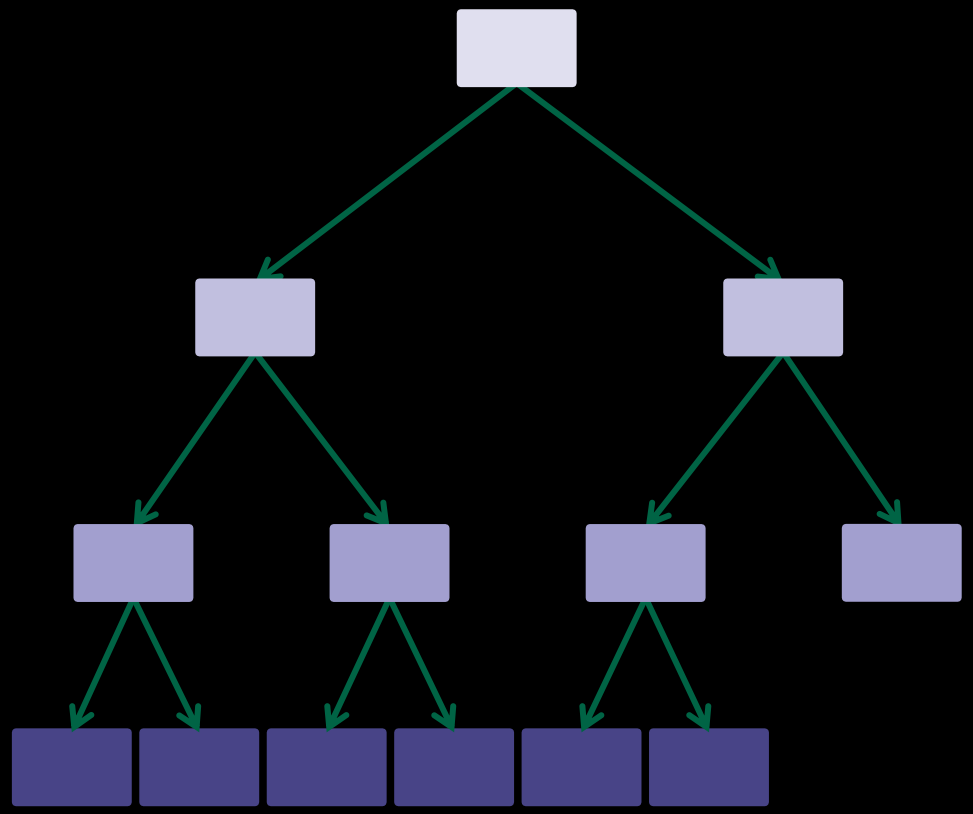
- Better hardware (GPUs)
- Better software (Algorithmic improvement)
- Better middleware (Tune for the architecture)

GPUs – the processor for ray tracing

- Abundant parallelism, massive computational power
- GPUs excel at shading
- Opportunity for hybrid algorithms

Acceleration Structures



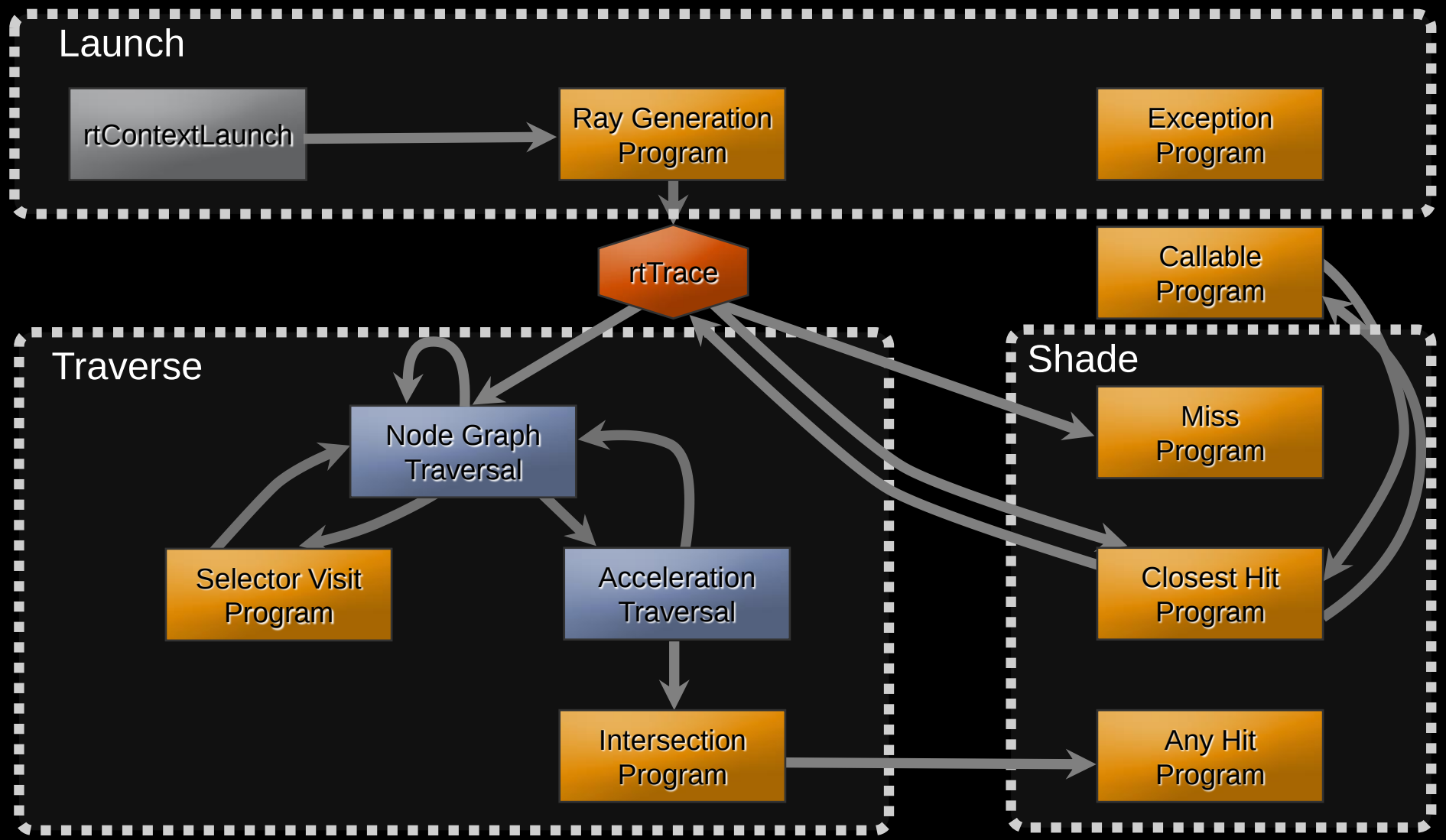



OptiX 3.0


OpenGL®


DIRECTX 11

OptiX Execution Model



C Host API Sample

```
RTresult rtContextCreate (RTcontext* context);
RTresult rtContextDestroy (RTcontext context);
RTresult rtContextDeclareVariable (RTcontext context, const char* name, RTvariable* v);
RTresult rtContextSetRayGenerationProgram (RTcontext context, unsigned int entry_point_index, RTprogr
RTresult rtBufferCreate (RTcontext context, unsigned int bufferdesc, RTbuffer* buffer);
RTresult rtBufferSetFormat (RTbuffer buffer, RTformat format);
RTresult rtBufferMap (RTbuffer buffer, void** user_pointer);
RTresult rtBufferUnmap (RTbuffer buffer);
RTresult rtProgramCreateFromPTXString (RTcontext context, const char* ptx, const char* program_name,
RTresult rtProgramCreateFromPTXFile (RTcontext context, const char* filename, const char* program_nam
RTresult rtContextLaunch2D (RTcontext context, unsigned int entry_point_index, RTsize image_width, RT
```

C++ Host API Sample

```
Context* context = Context::create();
context["max_depth"]->setInt( 5 );
context["scene_epsilon"]->setFloat( 1.e-4f );
// Ray gen program
Program ray_gen_program = context->createProgramFromPTXFile( “myprogram.ptx”,"pinhole_camera" );
context->setRayGenerationProgram( 0, ray_gen_program );

BasicLight lights[] = { ..... };
Buffer light_buffer = context->createBuffer(RT_BUFFER_INPUT);
light_buffer->setFormat(RT_FORMAT_USER);
light_buffer->setElementSize(sizeof(BasicLight));
light_buffer->setSize( sizeof(lights)/sizeof(lights[0]) );
memcpy(light_buffer->map(), lights, sizeof(lights));
light_buffer->unmap();
context["lights"]->set(light_buffer);
```

OptiX 3.0.1

- Bug fix release
- Available this week

OptiX Feature Sneak Peek

- Next release coming in a few months

Compilation Optimization

rtContextCompile()

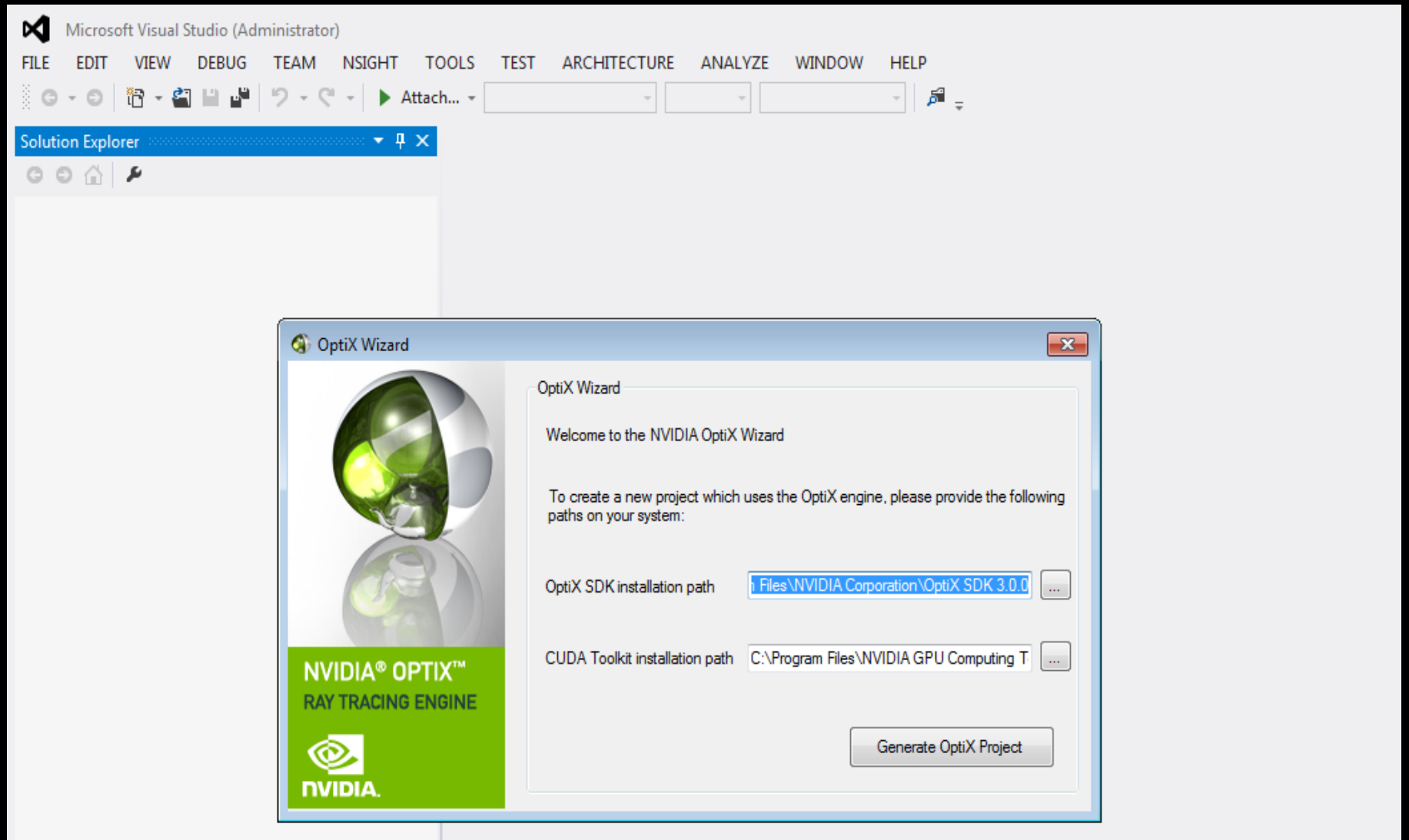
3-7X faster

- *Still, you should avoid recompiles if possible.*

Updated Support

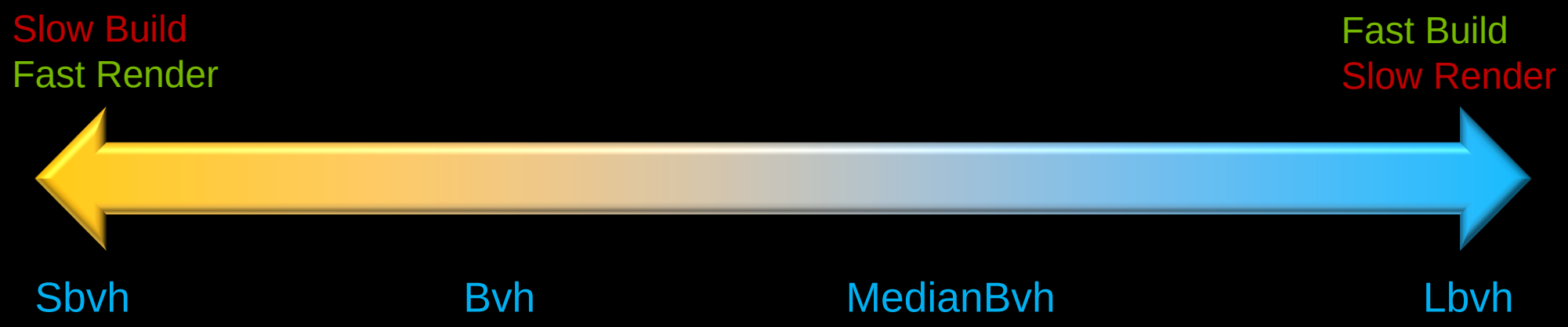
- Visual Studio 2012 support
- CUDA 5.5 support
- Quadro K6000 support

Visual Studio OptiX Wizard

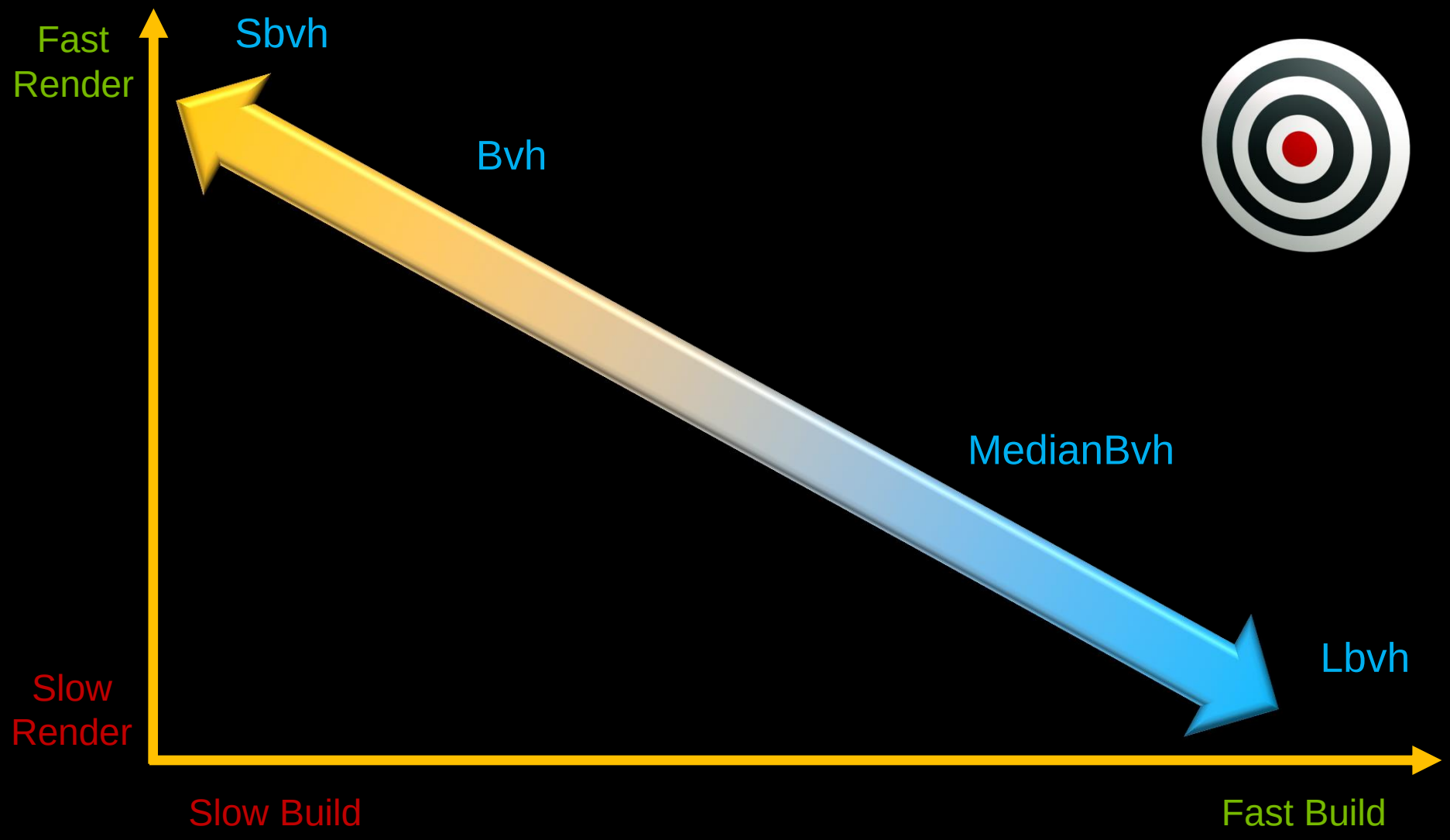


Acceleration Builder Options

- Sbvh has world class ray tracing performance
- Lbvh is extremely fast and works on very large datasets



Acceleration Builder Options



Fast BVH Builds

LBVH + refinement

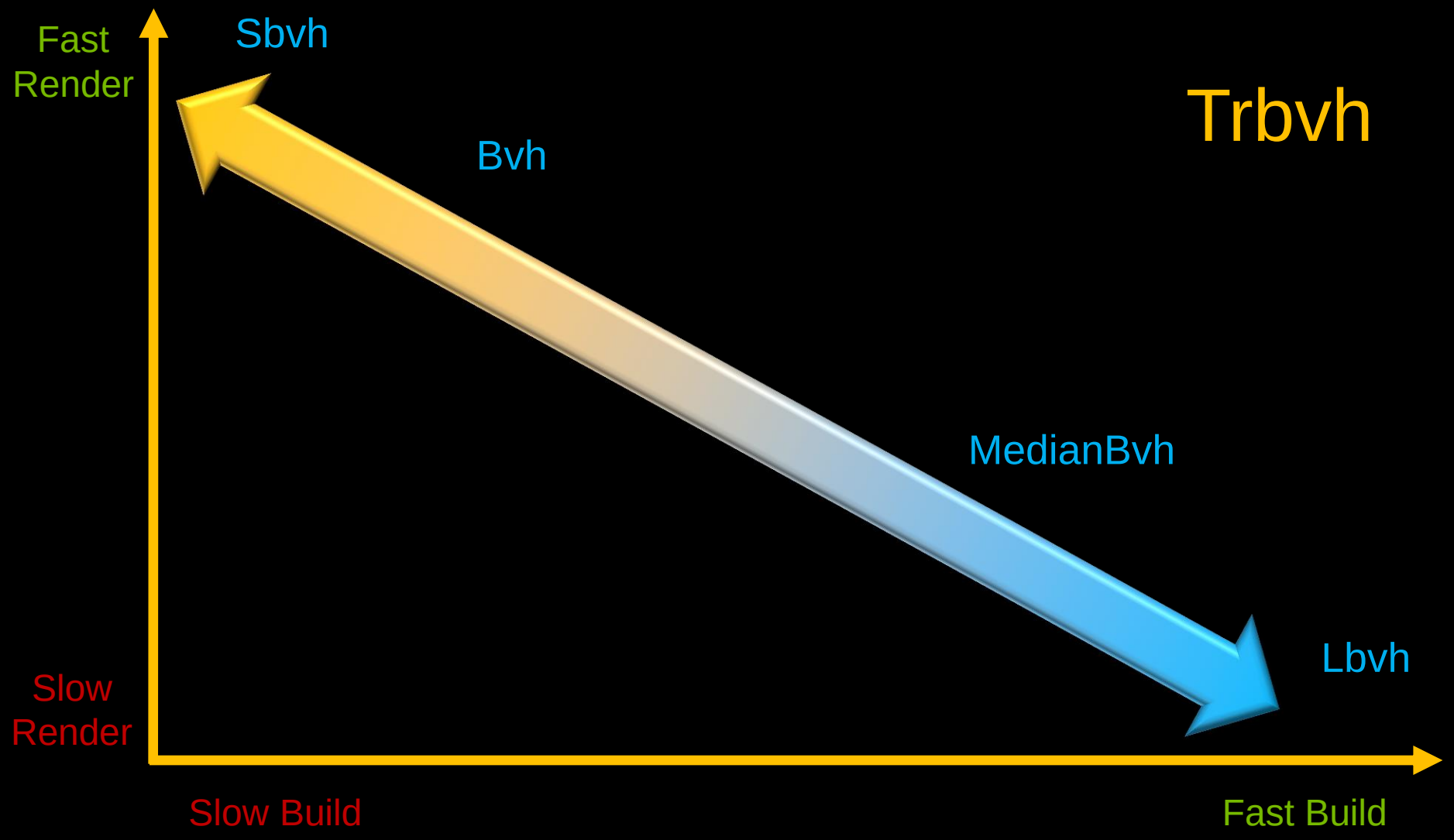
- A new approach introduced in OptiX 3.0
- ☺ Very fast to build
- ☺ Good for animation
- ☹ Quality does not approach optimal

TRBVH

LBVH + $\text{refinement}_{\text{new}}$ + splits

- New work by NVIDIA Research
- ☺ VERY fast to build
  - 40M tris / sec on a GeForce GTX Titan
  - World's fastest high quality BVH builder
- ☺ Quality averages 91% of SBVH
- HPG 2013 paper: https://research.nvidia.com/publication/fast-parallel-construction-high-quality-bounding-volume-hierarchies

Acceleration Builder Options



OptiX Low-Level Library

- Specialized for ray tracing (no shading)
- Replaces rtuTraversal
- Improved performance
  - Uses latest algorithms from NVIDIA Research
    - ray tracing kernels [Aila and Laine 2009; Aila et al. 2012]
    - Treelet Reordering BVH (TRBVH) [Karras 2013]
  - Can use CUDA buffers as input/output
  - Support for asynchronous computation
- Distributed as DLL and static library
- Designed with an eye towards future features

API Overview

- C API with C++ wrappers
- API Objects
  - Context
  - Buffer Descriptor
  - Model
  - Query

Context

- Context tracks other API objects and encapsulates the ray tracing backend
- Creating a context

```
OLLresult
ollContextCreate(OLLcontexttype type, OLLcontext* context)
```

- Context types

```
OLL_CONTEXT_TYPE_CPU
OLL_CONTEXT_TYPE_CUDA
```

- Default for CUDA backend uses all available GPUs
  - Selects “Master GPU” and makes it the current device
  - Master GPU builds acceleration structure

Context

- Selecting devices:

```
ollContextSetCudaDeviceNumbers(
   OLLcontext   context,
   int          deviceCount,
   const int*   deviceNumbers )
```

  - First device is used as the master GPU
- Destroying the context
  - destroys objects created by the context
  - synchronizes the CPU and GPU

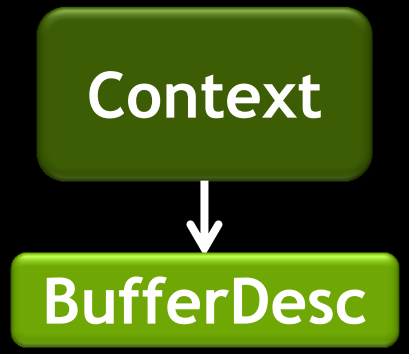
Buffer Descriptor

- Buffers are allocated by the application
- Buffer descriptors encapsulate information about the buffers

```
ollBufferDescCreate(
    OLLcontext      context,
    OLLbufferformat format,
    OLLbuffertype   type,
    void*           buffer,
    OLLbufferdesc*  desc )
```

- Specify region of buffer to use (in elements)

```
ollBufferDescSetRange( OLLbufferdesc desc, int begin, int end )
```



Buffer Descriptor

- Variable stride supported for vertex format

  `ollBufferDescSetStride`

  - Allows for vertex attributes

Buffer Descriptor

- Formats

  OLL_BUFFER_FORMAT_INDICES_INT3

  OLL_BUFFER_FORMAT_VERTEX_FLOAT3,

  OLL_BUFFER_FORMAT_RAY_ORIGIN_DIRECTION,

  OLL_BUFFER_FORMAT_RAY_ORIGIN_TMIN_DIRECTION_TMAX,

  OLL_BUFFER_FORMAT_HIT_T_TRIID_U_V

  OLL_BUFFER_FORMAT_HIT_T_TRIID

  …

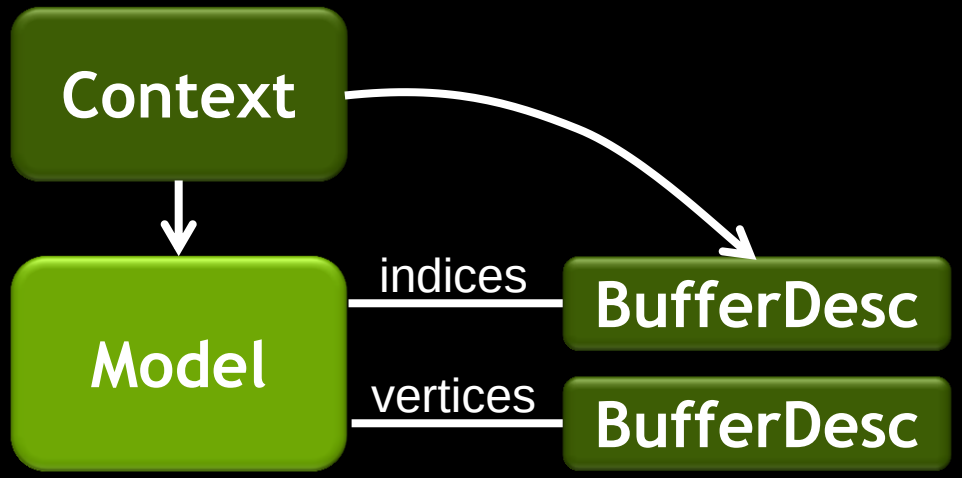
- Types

  OLL_BUFFER_TYPE_HOST

  OLL_BUFFER_TYPE_CUDA_LINEAR

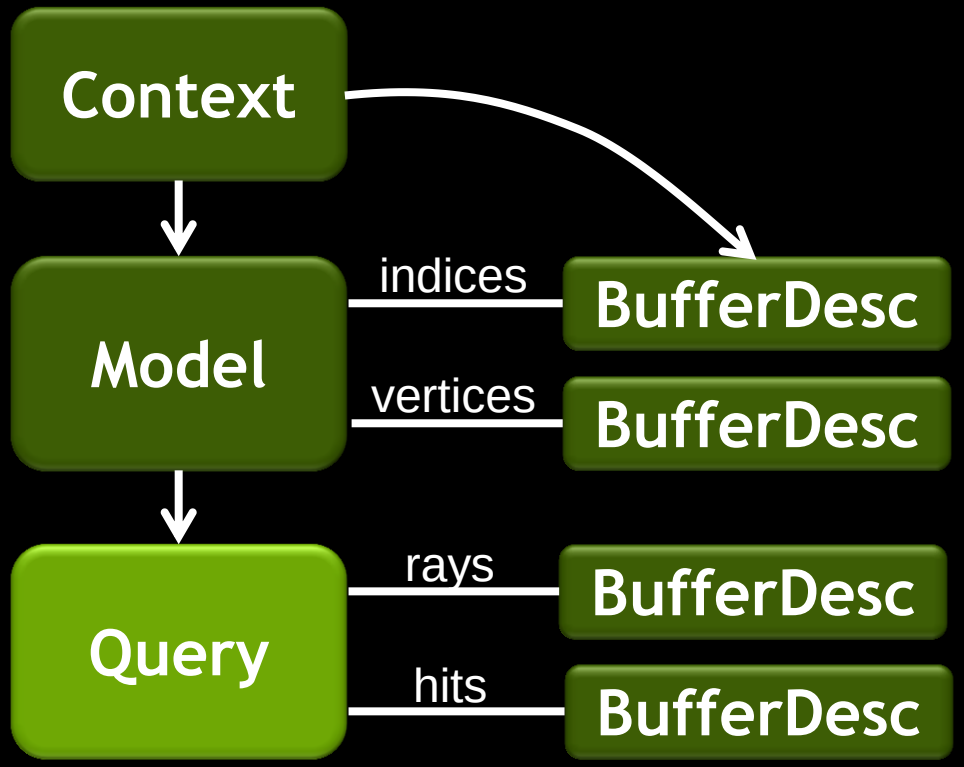
Model

- A model is a set of triangles combined with an acceleration data structure
  - `ollModelCreate`
  - `ollModelSetTriangles`
  - `ollModelFinalize`
- Asynchronous finalize
  - `ollModelFinalizeWait`
  - `ollModelFinalizePoll`

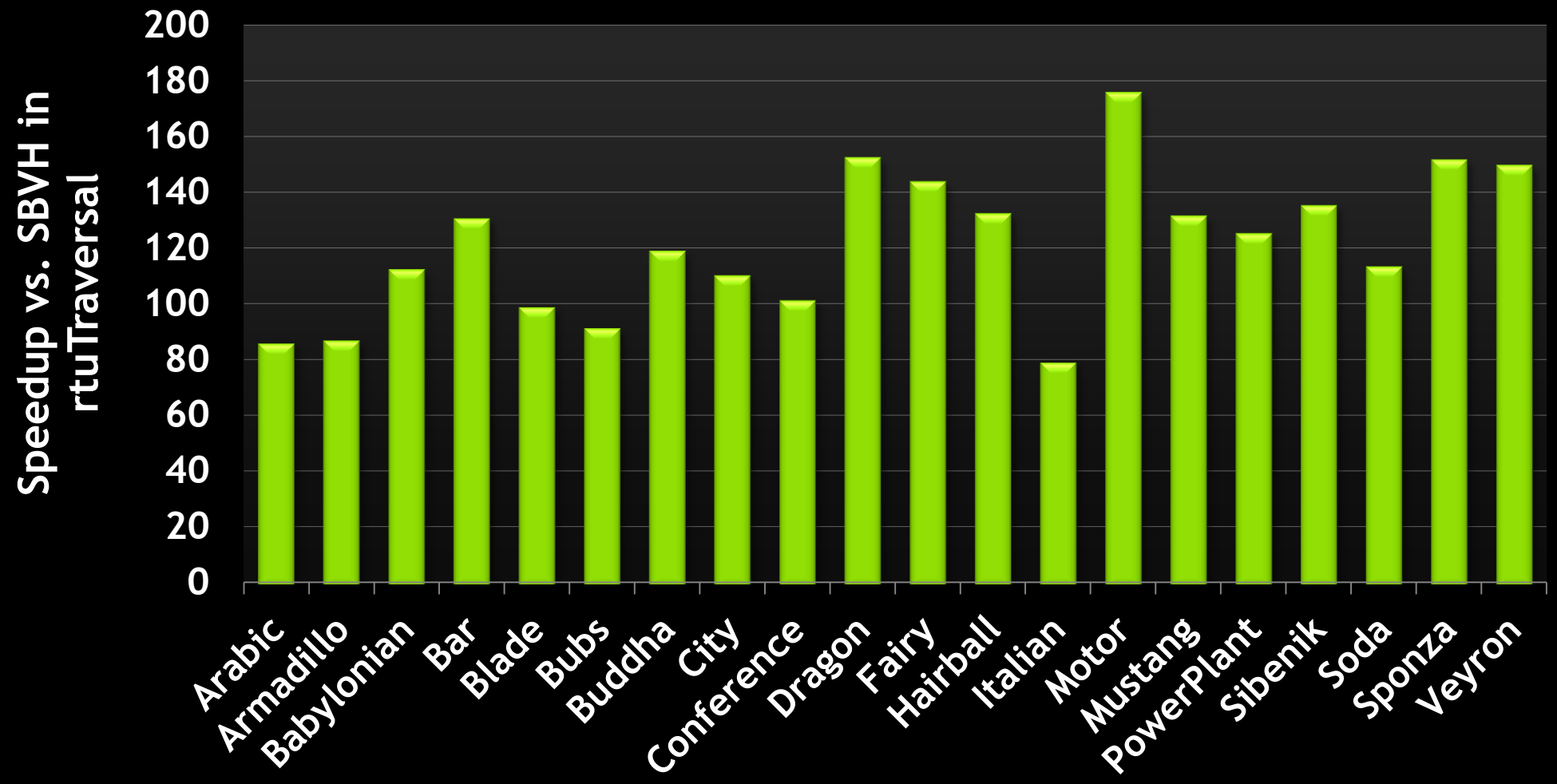


Query

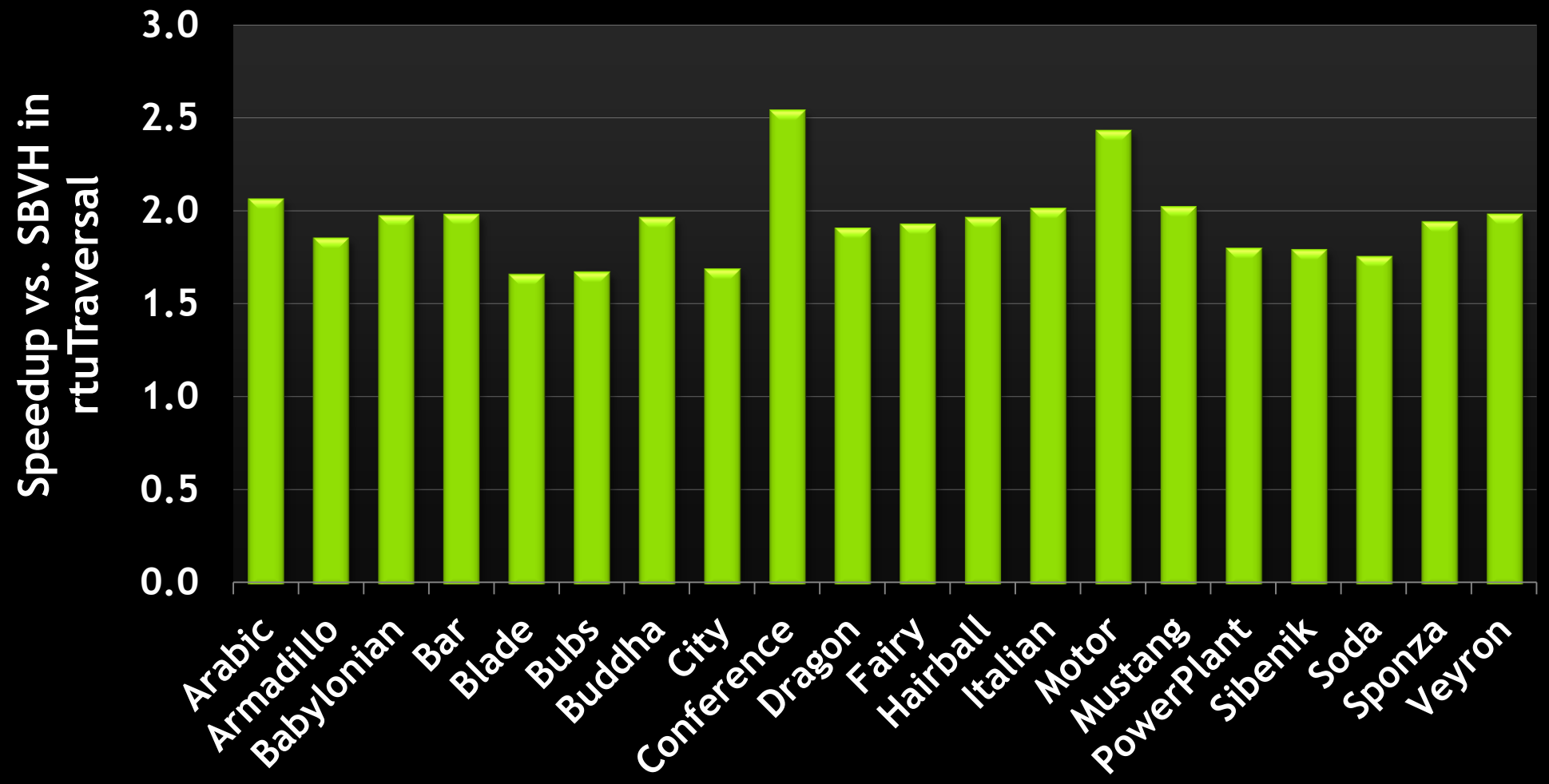
- Queries perform the ray tracing on a model
  - `ollQueryCreate`
  - `ollQuerySetRays`
  - `ollQuerySetHits`
  - `ollQueryExecute`
- Query types
  - `OLL_QUERY_TYPE_ANY`
  - `OLL_QUERY_TYPE_CLOSEST`
- Asynchronous query
  - `ollQueryWait`
  - `ollQueryPoll`



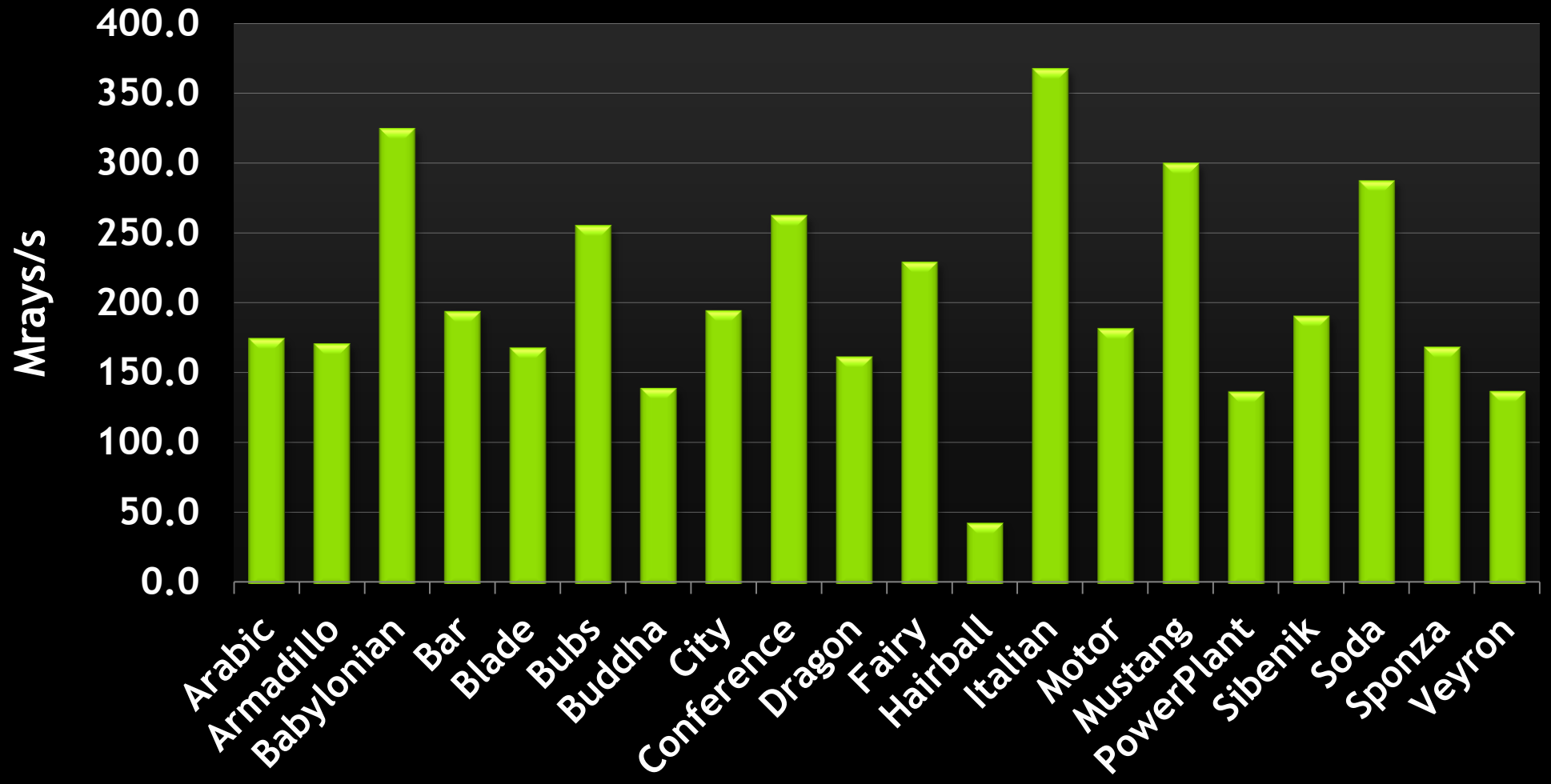
Build Performance



Ray Tracing Performance



Ray Tracing Performance



Future

- Features we want to implement
  - Animation support (refit/refine)
  - Instancing
  - Large-model optimizations

Art And Animation Studio

Vertex Light Baking

- Working with Bungie
  - But will be made available. *Contact us if interested.*
  - Kavan, Bargteil, Sloan, “Least Squares Vertex Baking”, EGSR 2011

“NVIDIA's Optix has been instrumental when baking Ambient Obscurance (AO) over the extremely complex geometry in the worlds of Destiny. The high performance and ability to quickly explore various formulations of AO were invaluable.”

BUNGiE®

Vertex Light Baking

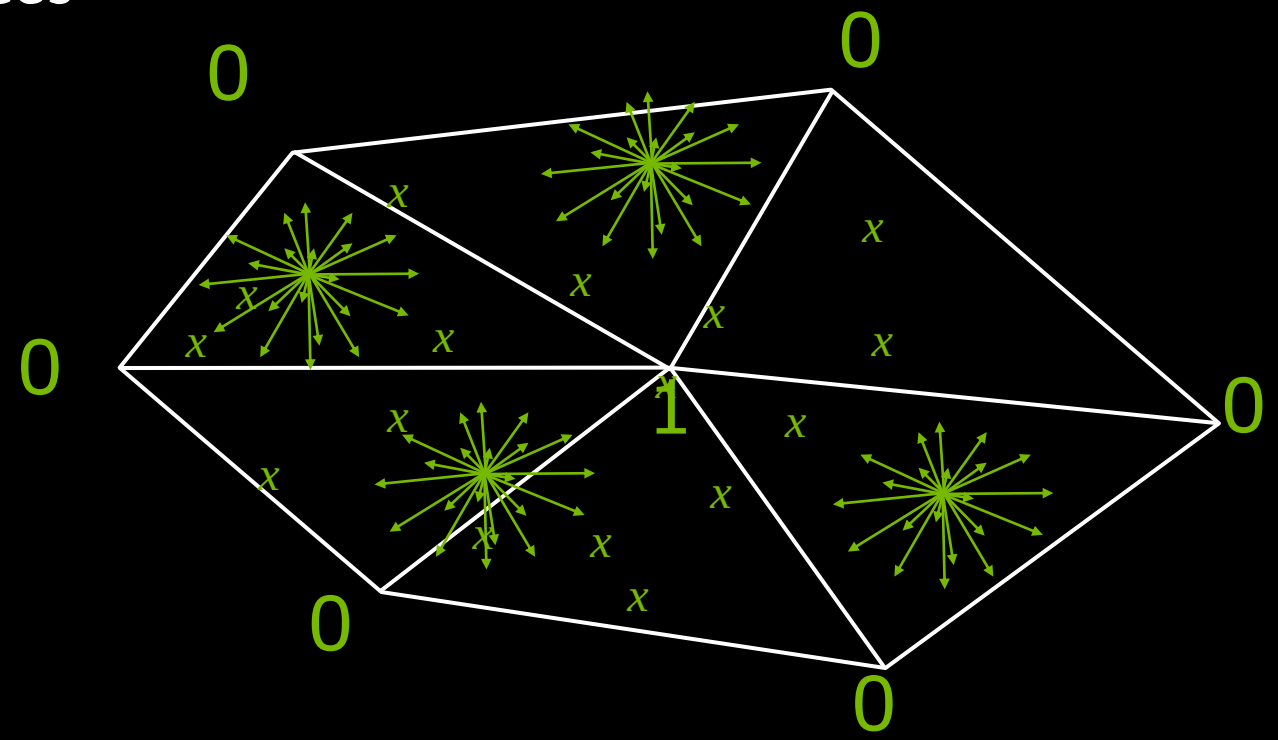
- Working with Bungie
  - But will be made available. *Contact us if interested.*
  - Kavan, Bargteil, Sloan, “Least Squares Vertex Baking”, EGSR 2011
- Compared to textures…
  - Less memory & bandwidth
  - No *u,v* parameterization
  - Good for low-frequency effects

Typical Scene

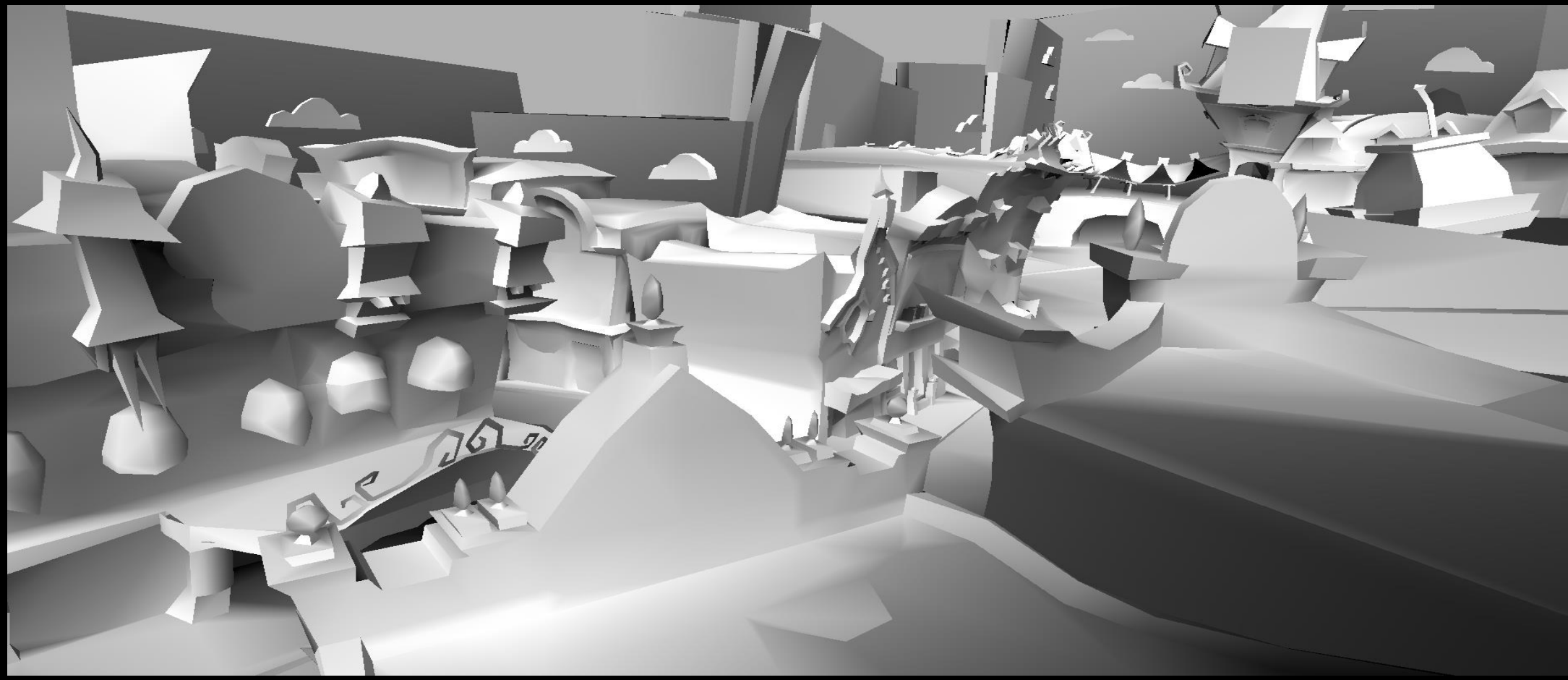
- Linear interpolation
- Static mesh
- Coarse mesh

Piecewise Linear Approximation

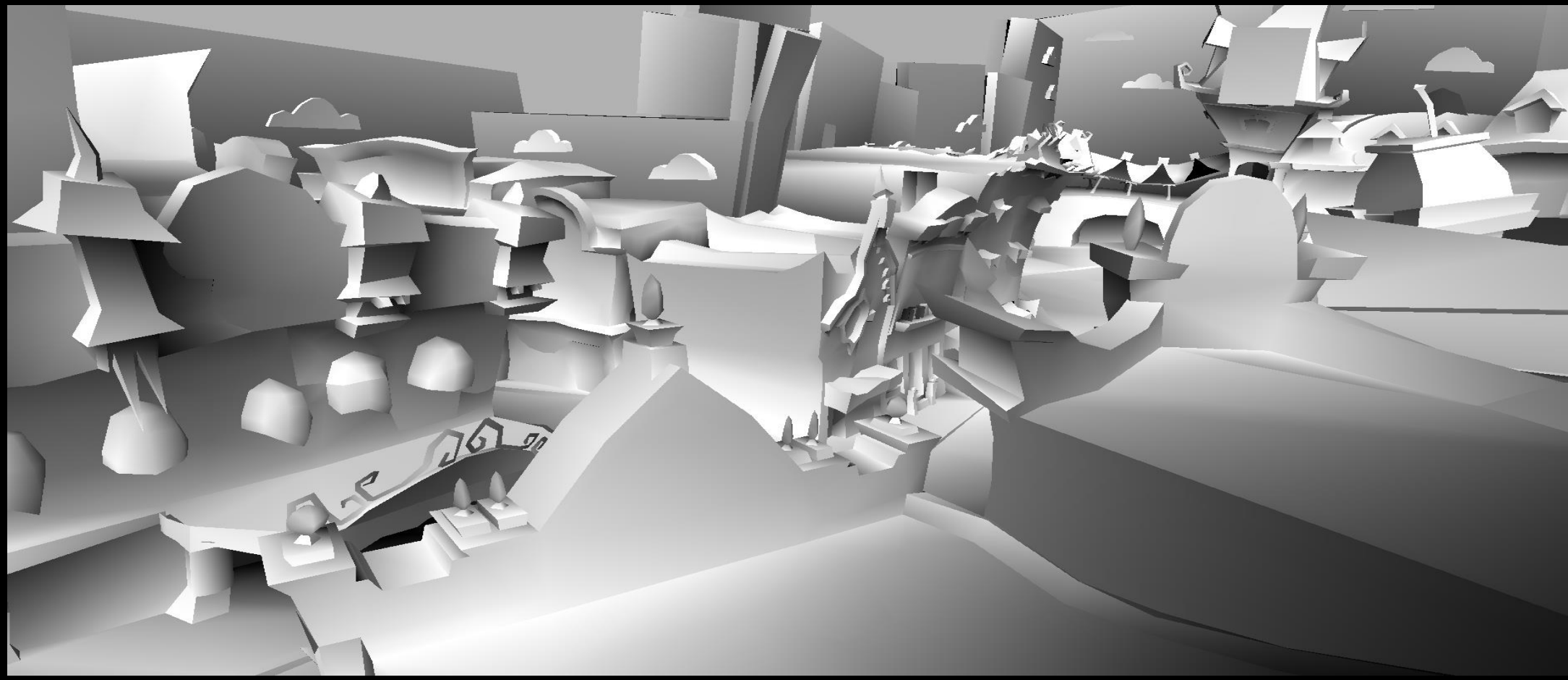
- Sample illumination on surface
- Each sample is a hemisphere of rays
- Reconstruct values at vertices



Weighted Averaging



Least Squares Vertex Baking



Pixar Research & NVIDIA OptiX Interactive Lighting Collaboration

Pixar Research
Interactive Lighting

Jean-Daniel Nahmias

NVIDIA Visual Computing Theater in NVIDIA Booth

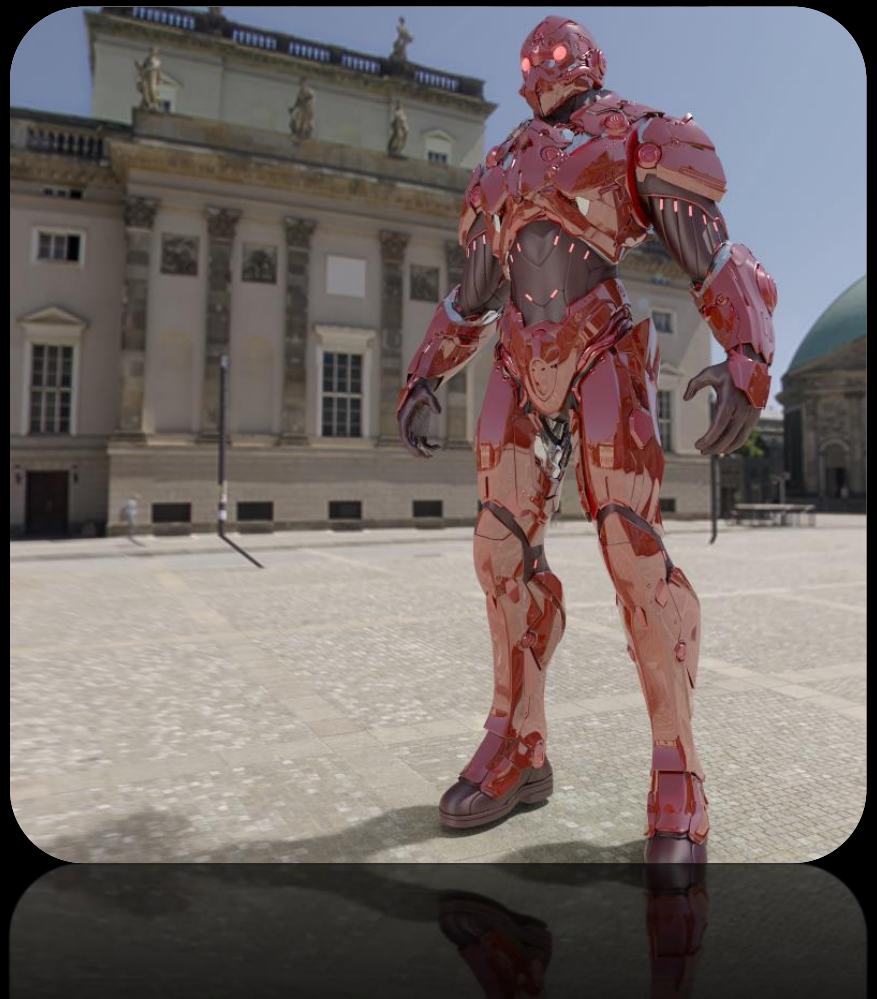
- Tue. 4:00 – 4:30
- Wed. 2:00 – 2:30
- Thur. 10:40 – 11:10

PIXAR
ANIMATION STUDIOS

PIXAR
RESEARCH GROUP

See OptiX at Siggraph

- Pixar Research Interactive Lighting
  - NVIDIA Visual Computing Theater
  - Tue. 4:00, Wed. 2:00, Thur. 10:40
- Mental ray with OptiX-powered Final Gather
  - NVIDIA booth
- Bunkspeed Shot / Move / Drive with IRay
  - Bunkspeed booth, NVIDIA booth
  - NVIDIA Visual Computing Theater: Thur. 1:20
- Course: “Physically Based Shading in Theory and Practice”
  - Thur. 9:00
  - 10:00: “Crafting a Next-Gen Material Pipeline for The Order: 1886”



OptiX Resources

- Slides & Video
  - GTC OptiX Introduction
  - GTC OptiX Optimization
  - http://www.gputechconf.com/gtcnew/on-demand-gtc.php?topic=39
  - This talk: http://www.nvidia.com/object/siggraph2013-tech-talks.html
- Download OptiX
  - Available for free: Windows, Linux, Mac
  - http://developer.nvidia.com
- OptiX forum
  - https://devtalk.nvidia.com/default/board/90