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
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
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
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, RTprogram program);
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, RTprogram* program);
RTresult rtProgramCreateFromPTXFile (RTcontext context, const char* filename, const char* program_name, RTprogram* program);
RTresult rtContextLaunch2D (RTcontext context, unsigned int entry_point_index, RTsize image_width, RTsize image_height);
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
Welcome to the NVIDIA OptiX Wizard

To create a new project which uses the OptiX engine, please provide the following paths on your system:

OptiX SDK installation path: \Files\NVIDIA Corporation\OptiX SDK 3.0

CUDA Toolkit installation path: C:\Program Files\NVIDIA GPU Computing T
Acceleration Builder Options

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

Slow Build
Fast Render

Fast Build
Slow Render

Sbvh  Bvh  MedianBvh  Lbvh
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
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

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

```c
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:

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

- Specify region of buffer to use (in elements)
  
  ```c
  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
A model is a set of triangles combined with an acceleration data structure

- `ollModelCreate`
- `ollModelSetTriangles`
- `ollModelFinalize`

Asynchronous finalize

- `ollModelFinalizeWait`
- `ollModelFinalizePoll`
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

Speedup vs. SBVH in rtuTraversal

- Arabic
- Armadillo
- Babylonian
- Bar
- Blade
- Bubs
- Buddha
- City
- Conference
- Dragon
- Fairy
- Hairball
- Italian
- Motor
- Mustang
- PowerPlant
- Sibenik
- Soda
- Sponza
- Veyron
Ray Tracing Performance

Speedup vs. SBVH in rtuTraversal

Arabic  Armadillo  Babylonian  Bar  Blade  Bubs  Buddha  City  Conference  Dragon  Fairy  Hairball  Italian  Motor  Mustang  PowerPlant  Sibenik  Soda  Sponza  Veyron
Future

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

- Working with Bungie
  - But will be made available. *Contact us if interested.*

“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.”
Vertex Light Baking

- Working with Bungie
  - But will be made available. *Contact us if interested.*

- 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
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
  - This talk:

- Download OptiX
  - Available for free: Windows, Linux, Mac

- OptiX forum
  - https://devtalk.nvidia.com/default/board/90