New Features in OptiX 6.0
David Hart, Ankit Patel, NVIDIA
AGENDA

OptiX Overview
New features in OptiX 6
OptiX Performance tips
OptiX Debugging tips
General OptiX improvements
Summary / Q&A
NVIDIA RTX TECHNOLOGY

Applications

- NVIDIA OptiX for CUDA
- NVIDIA VKRay for Vulkan
- Microsoft DXR for DX12

NVIDIA® RTX™ Technology

NVIDIA GPU
10 years of work in computer graphics algorithms and GPU architectures
OptiX Rendering Applications

MEDIA & ENTERTAINMENT

PRODUCT DESIGN

ARCHITECTURE
BROAD SUPPORT FOR NVIDIA RTX

RTX COMES TO 9M 3D CREATORS IN 2019
OptiX Scientific Applications

ELECTROMAGNETIC SIMULATIONS

FLUID DYNAMICS

SOUND PROPOGATION [VRWORKS AUDIO]
RT Cores

Traversals
- Hardware accelerated with RT Cores
- Includes custom primitives

Triangle intersection
- Only works with triangles
- Requires OptiX triangle API
**Terminology**

**RT Cores:**
- New in Turing
- Hardware for ray traversal & ray-triangle intersection

**RTX Software:**
- Technology stack for ray tracing
- Driver software

![Applications Diagram](image)
OptiX delivery

OptiX SDK package contents
- Headers
- Documentation
- Samples
- Libraries
  - optix.51.dll [~40MB]
  - optix.6.0.0.dll [~200KB]
OptiX delivery

OptiX SDK package contents
- Headers
- Documentation
- Samples
- Libraries
  - optix.51.dll [~40MB]
  - optix.6.0.0.dll [~200KB]

OptiX in NVIDIA driver
- nvoptix.dll
AGENDA

OptiX Overview

New features in OptiX 6
OptiX Performance tips
OptiX Debugging tips
General OptiX improvements
Summary / Q&A
OptiX Programming Model

You provide: A renderer
CUDA programs for: Rays, Geometry, Shading, Miss, Exception

OptiX provides: plumbing
Compilation, scheduling, traversal, memory management, etc.

OptiX does not make assumptions about your
input data, output data, or algorithms
OptiX Programs

Ray Generation

Intersection

Closest Hit

Miss

Bounding Box

Any Hit

* per geometric primitive type
* per entry point
* per ray type
Getting Started with OptiX

developer.nvidia.com/optix

Tutorial ("Introduction to NVIDIA OptiX", GTC 2018)

OptiX 6 Programming Guide

SDK samples

OptiX advanced samples (github)

Forum developer.nvidia.com/optix-forums
OptiX Overview

New features in OptiX 6

OptiX Performance tips

OptiX Debugging tips

General OptiX improvements

Summary / Q&A
New in OptiX 6

=> Support for RT Cores

RTX API: Maxwell+
RT Cores: Turing+
New Triangle API for hardware accelerated meshes
New attribute programs
New API for stack sizes, rtTrace, and more
rtTrace from bindless callable programs
Separate & parallel compilation of shaders
Multi-GPU improvements
Denoiser is Turing accelerated
Triangle API

OptiX interface to RTX (built-in) triangle intersection

No bounds program or intersection program needed

New “attribute program” lets you compute attributes for shading
Attribute Programs

Compute intersection data for shaders

There is a default attribute program that produces U & V

You get barycentrics, then compute your own attributes shading normals, etc...

See:

[host] \textit{rtGeometryTrianglesSetAttributeProgram()}
[device] \textit{rtGetPrimitiveIndex()}
[device] \textit{rtGetTriangleBarycentrics()}
Triangle API Example: OptiXMesh.cpp

```cpp
optix::GeometryTriangles geom_tri = ctx->createGeometryTriangles();
geom_tri->setPrimitiveCount( mesh.num_triangles );
geom_tri->setTriangleIndices( buffers.tri_indices, RT_FORMAT_UNSIGNED_INT3 );
geom_tri->setVertices( mesh.num_vertices, buffers.positions, buffers.positions->getFormat() );

geom_tri->setAttributeProgram( createAttributesProgram( ctx ) );

size_t num_matls = optix_materials.size();
geom_tri->setMaterialCount( num_matls );
geom_tri->setMaterialIndices( buffers.mat_indices, 0, sizeof( unsigned ), RT_FORMAT_UNSIGNED_INT );

optix_mesh.geom_instance = ctx->createGeometryInstance();
optix_mesh.geom_instance->setGeometryTriangles( geom_tri );

// Set the materials
optix_mesh.geom_instance->setMaterialCount( num_matls );
for( size_t idx = 0; idx < num_matls; ++idx )
{
    optix_mesh.geom_instance->setMaterial( idx, optix_materials[idx] );
}
```
Stack sizes

New API calls:

`rtContextSetMaxTraceDepth()`

`rtContextSetMaxCallableProgramDepth()`

Default is 5. Reduce them when you can. Max. is 31

In RTX Mode, `rtContextSetStackSize()` has no effect
rtTrace from bindless callables

You can now call rtTrace() from inside a bindless callable program

They cost a little bit more than inline code

so balance your usage

Details are covered in the OptiX Programming Guide

“Calling rtTrace from a bindless callable program”
Two new optional flags

```c
rtTrace( topNode, ray, prd,
    RTvisibilitymask mask=RT_VISIBILITY_ALL,
    RTrayflags flags=RT_RAY_FLAG_NONE )
```
rtTrace: 8 bit visibility mask

Hide/show geometry -- hardware support for skipping sub-trees

Allows for up to 8 visibility sets, replaces selectors & visit programs

See:

rtGroupSetVisibilityMask()
rtGeometryGroupSetVisibilityMask()

To use:

rTrace(top_object, ray, per_ray_data, mask=0xFF)
rtTrace: ray flags

rtTrace( topNode, ray, prd,
      RTvisibilitymask mask=RT_VISIBILITY_ALL,
      RTrayflags flags=RT_RAY_FLAG_NONE )

RT_RAY_FLAG_DISABLE_ANYHIT
RT_RAY_FLAG_DISABLE_CLOSESTHIT
RT_RAY_FLAG_CULL_BACK_FACING_TRIANGLES
RT_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES
RT_RAY_FLAG_CULL_DISABLED_ANYHIT
RT_RAY_FLAG_CULL_ENABLED_ANYHIT
New geometry flags

RTgeometryflags

rtGeometrySetFlags()

rtGeometryTrianglesSetFlagsPerMaterial()
  RT_GEOMETRY_FLAG_DISABLE_ANYHIT
  RT_GEOMETRY_FLAG_NO_SPLITTING

RTinstanceflags

rtGeometryGroupSetFlags()
  RT_INSTANCE_FLAG_DISABLE_TRIANGLE_CULLING
  RT_INSTANCE_FLAG_FLIP_TRIANGLE_FACING
  RT_INSTANCE_FLAG_DISABLE_ANYHIT
  RT_INSTANCE_FLAG_FORCE_ANYHIT
Triangle API + Motion Blur

Use \texttt{rtGeometryTrianglesSetMotionVertices()} instead of \texttt{rtGeometryTrianglesSetVertices()}

Motion blur currently requires SM traversal in the motion BVH \& parent BVH

You can mix motion BVHs and static BVHs
The static BVHs will use RT Core traversal
Try not to mix in the same Geometry Group
AGENDA

OptiX Overview
New features in OptiX 6
**OptiX Performance tips**
OptiX Debugging tips
General OptiX improvements
Summary / Q&A
Performance with RT Cores

Don’t forget **Amdahl's Law**

\[
S_{\text{latency}}(s) = \frac{1}{(1 - p) + \frac{p}{s}}
\]

For RT Core speedup to be big, you need to be traversal bound

Both obvious and easy to forget

Imagine 50-50 shade vs traverse+intersect
RT Cores

- Workload
- Scene Hierarchy
- Memory Traffic
- Round Trips
Highest Performance Workloads

Coherent rays are better
  Primary rays

Short rays are better
  AO / small t_max

Simple shading
  trivial shading, e.g.,
    diffuse or normal

Big batches
Scene hierarchy

Hardware support for 1 level of instancing
You get 1 for free!
Memory Traffic

RTX 6000 memory bandwidth: 672 GB/sec

672 GB/sec ÷ 10 GRays/sec = 67.2 bytes / ray

Includes traversal

It’s easy to exceed this budget in shaders
SM + RT Core round-trips

Some features run CUDA code mid-traversal

Let’s build a mental model of how the SMs & RT Cores interact
OptiX 5 traversal

“Mega” Kernel
RTX traversal: custom primitives

*NB: conceptual model of execution, not timing.
RTX traversal: hardware triangles
RTX traversal: any-hit

RTX traversal: 1 level instancing (2 lvl scene)
RTX traversal: 2 level instancing (3 lvl scene)
RTX traversal: the ideal

The goal: uninterrupted traversal on RT core
RTX Traversal: summary

Triangle API + RTX => 2x-10x faster -- first step to gigarays

Things that can impact RT Core traversal:
- intersection program
- any-hit program
- 2 or more levels of instancing
- motion blur

Shadow rays tend to be faster. (re-evaluate balance of shadow rays)
Any-hit & early termination

Disable any-hit if you don’t need it!
   Use one of the *_DISABLE_ANYHIT instance or ray flags

For shadow-like rays and built-in triangles, we now recommend
   closest-hit & rtTrace(,,, RT_RAY_FLAG_TERMINATE_ON_FIRST_HIT)

For shadow-like rays and custom intersection
   continue to use any-hit & call rtTerminateRay()
Continuations: callables & rtTrace

The compiler wraps callable program & rtTrace invocations in continuations, which take longer to compile, and consume registers.

Seize the opportunity if you see ways to trim

Calls very early or very late are better than right in the middle

goal: less live state, think about tail-recursion
Achieving High Performance in OptiX

Peak RT Core throughput depends on your workload

Be judicious with:
- traversal depth
- complex shading
- use of any_hit for cut-outs
- motion blur

NB: # of polygons is less relevant!
AGENDA

- OptiX Overview
- New features in OptiX 6
- OptiX Performance tips
- **OptiX Debugging tips**
- General OptiX improvements
- Summary / Q&A
Common pitfalls in OptiX

Crash?

Stack overrun
  Increase stack size

Indexing user data out of bounds
  Enable exceptions & add bounds checks*
Common pitfalls in OptiX

Slow start?

shader compilation
  avoid dynamic program selection

large number of instances / scene graph
  merge meshes, flatten scene
Common pitfalls in OptiX

Slow trace?

deep instancing: flatten / merge / bake, when you can
large ray payload: trim payload
large any-hit program: trim / remove any-hit
   use closest-hit w/ flags
large number of variable updates:
   use a buffer instead
shader re-compilation
shader complexity
move callable programs to inline code*
memory traffic & payload size
OptiX debugging tips

Turn on exceptions

Add an exception program

Print exception details

Turn on usage report
OptiX 6 Known Issues

- Motion blur is not supported with no-accel enabled in RTX Mode

- Selectors not implemented in RTX Mode, use visibility masks instead.

- Nsight can not (yet) profile OptiX in RTX Mode. Coming soon.
AGENDA

OptiX Overview
New features in OptiX 6
OptiX Performance tips
OptiX Debugging tips
General OptiX improvements
Summary / Q&A
General OptiX improvements

Behind the scenes, OptiX 6 improved:

- Separate & parallel shader compilation
- BVH build times
- BVH memory
- Denoiser perf
RTX Acceleration Structures

They are faster, ~80Mtris/sec build
Re-fit is usually >= 10x faster than build
BVH format is set automatically when using RTX Mode

BVH compaction: may save 1.5x-2x on memory
in return for ~+10% build time
To disable: rtAccelerationSetProperty( accel, "compact", "0" );
Multi-GPU

OptiX 6 adds RTX support for multiple GPUs

Geometry & BVH replicate
Textures replicate until memory gets tight,
=> N GPUs are usually ~N x faster, but usually not N x memory

Simplified load balancing
No mixing Turing and pre-Turing devices
For best performance: Use NVLINK & homogeneous GPUs
  Slow GPUs bottleneck fast GPUs
  Heterogeneous GPUs can trigger multiple compiles.
AGENDA

- OptiX Overview
- New features in OptiX 6
- OptiX Performance tips
- OptiX Debugging tips
- General OptiX improvements

Summary / Q&A
Best practices in OptiX

When possible...

• Use built-in triangles, instead of AABB & intersect programs
• Flatten your scene to zero or one level of instancing
• Share bottom level structures / merge meshes to a single bottom level structure
• Use BVH refit instead of rebuilding
• Distribute accel refits over multiple frames / update lazily, only when needed
• Minimize payload size, attribute size, and trace recursion depth
• Use closest-hit instead of any-hit with built-in triangles
• Do use any-hit for cut-outs. (NB any-hit is out of order, may not be closest)
• Pay attention to dependent memory access when reading vertex & material data.
Summary

- RT Core support

- For fast trace, “Feed The RT Cores”
  - Triangle api, one level of instancing
  - Use sparingly: any-hit, deep hierarchy, motion blur

- New API: attribute programs, stack sizes, callable programs

- Improvements to: BVH, compilation, multi-GPU, denoiser

- OptiX runtime in the driver
Questions?