Release Notes for the NVIDIA® OptiX™ ray tracing engine

Version 2.6.0 September 2012

Welcome to the latest release of the NVIDIA OptiX ray tracing engine and SDK, with support for all CUDA-capable GPUs. This package contains the libraries required to experience the latest technology for programmable GPU ray tracing, plus pre-compiled samples (with source code) demonstrating a broad range of ray tracing techniques and highlighting basic functionality.

Support:

The normal path for OptiX support is on NVIDIA's Developer Zone at http://developer.nvidia.com/

As of this writing, this support avenue is not currently available due to these forums being down for maintenance. Until they're restored, please send your issues directly to OptiX-Help@NVIDIA.com and they will be addressed by the development team. You can continue to download OptiX from http://developer.nvidia.com/optix-download/

System Requirements (for running binaries referencing OptiX)

Graphics Hardware:

- CUDA capable devices (G80 or later) are supported on **GeForce**, **Quadro**, or **Tesla** class products. Multiple devices/GPUs are only supported on “GT200”, “Fermi” and “Kepler” class GPUs. Out-of-core ray tracing of large datasets is only supported on Quadro and Tesla GPUs.

Graphics Driver:

- The CUDA R300 or later driver is **required**. The latest drivers available are highly recommended (301.42 or later for Windows, 302.06.03 for Linux and the CUDA 5.0 driver extension for Mac). For the Mac, the driver extension module supplied with CUDA 5.0 or later will need to be installed. Note that the Linux and Mac drivers can only be obtained from the CUDA 5.0 Preview at the moment.
- The TCC driver (used by Tesla products on Windows 7 or Vista) is **not** currently supported. Placing your Tesla hardware into the Windows Display Driver Model (WDDM) mode will make it compatible with OptiX. This restriction should be resolved with the next major OptiX release.
- SLI is not required for OptiX to use multiple GPUs, and it interferes when OptiX uses either D3D or OpenGL interop. Disabling SLI will not degrade OptiX performance and will provide a more stable environment for OptiX applications to run.

Operating System:

- Windows XP/Vista/7 32-bit or 64-bit; Linux RHEL 4.8 - 64-bit only, Ubuntu 10.10 - 64-bit; OSX 10.6+ (universal binary with 32 and 64-bit x86).

Development Environment Requirements (for compiling with OptiX)

All Platforms (Windows, Linux, Mac OSX):

- **CUDA Toolkit 2.3, 3.0, 3.1, 3.2, 4.0, 4.1, 4.2.**
  OptiX 2.6 has been built with CUDA 4.2, but any specified toolkit should work when compiling PTX for OptiX. If an application links against both the OptiX library and the CUDA runtime on Mac and Linux, it is recommended to use CUDA 4.2.
- **C/C++ Compiler**
  Visual Studio 2005, 2008 or 2010 is required on Windows systems. gcc 4.2 and 4.3 have been tested on Linux. The 3.2 Xcode development tools have been tested on Mac OSX 10.7.
- **GLUT**
  Most OptiX samples use the GLUT toolkit. Freeglut ships with the Windows OptiX distribution. GLUT is installed by default on Mac OSX. A GLUT installation is required to build samples on Linux.

Enhancements since OptiX 2.5.1:

- Kepler GPU support.
  OptiX now supports all sm_30 Kepler devices (GK104 and GK107). To support Kepler-based products with existing applications, simply replace the OptiX DLL's with those from the 2.6 version and rerun. In addition, PTX targeting sm_30 can be supplied to OptiX if specific sm_30 instructions are required.
- CUDA LLVM Optimizations
  CUDA 4.1 and 4.2 utilize a newer LLVM-based compiler, which generates PTX code with different performance characteristics than prior versions. OptiX 2.6 now targets this newer output. Performance results may vary from the previous OptiX, but should be within a few percent (plus or minus) of the 2.5.1 version.

Known limitations with version 2.6.0:

- Utilizing 32-bit PTX within a 64-bit application currently works but is discouraged, as this path may be removed in a future OptiX version.
- Specifying “Lbvh” as the builder in a 64-bit host application while using 32-bit PTX will cause the MedianBvh builder to be utilized.
- Support for building host-based acceleration structures in parallel has been disabled on Linux in this version of OptiX.
- OptiX supports running with NVIDIA Parallel Nsight but does not currently support kernel debugging in Nsight. In addition, it is not recommended to compile PTX code using any -g (debug) flags to nvcc.
- Use of OpenGL and DirectX interop causes OptiX to crash when SLI is enabled. As noted previously, SLI is not required to achieve scaling across multiple GPUs and is not recommended when using OptiX.
- All GPUs used by OptiX must be of the same MAJOR compute capability, such as compute capability 1.x or 2.x. OptiX will automatically select the set of GPUs of the highest major compute capability and only use those. For example, in a system with a GeForce GTX 460 (compute 2.1) and a GeForce GTX 480 (compute 2.0), both will be used, but in a system with a Quadro 5800 (compute 1.3) and a Quadro 6000 (compute 2.0) only the compute 2.0 device would be selected. Applications may explicitly choose which GPUs to run, as is done in the progressive photon mapper sample, ppm.cpp, at the start of initScene(), but if the application requests a set of devices of different major compute capability an error will be returned.
- Texture arrays and MIP maps are not yet implemented.
- malloc(), free(), and printf() do not work in device code.
- Applications that use RT_BUFFER_INPUT_OUTPUT or RT_BUFFER_OUTPUT buffers on multi-GPU contexts

must take care to ensure that the stride of memory accesses to that buffer is compatible with the PCIe bus payload size. Using a buffer of type RT_FORMAT_FLOAT3, for example, will cause a massive slowdown; use RT_FORMAT_FLOAT4 instead. Likewise, a group of parallel threads should present a contiguous span of 64 bytes for writing at once on an Intel chipset to avoid massive slowdowns, or 16 bytes on NVIDIA chipsets to avoid moderate slowdowns.

- Linux only: due to a bug in GLUT on many Linux distributions, the SDK samples will not restore the original window size correctly after returning from full-screen mode. A newer version of freeglut may avoid this limitation.
- The CUDA release notes recommend the use of -malign-double with GCC. However, on Mac OSX systems (10.5 with GCC 4.0.1 and 4.2.1 and 10.6 with GCC 4.2.1) this flag can produce miscompiles with std::stream based classes in host code when compiling to 32 bits. If the structs are different sizes between device and host code, consider manually padding the structure rather than using this compiler flag.
- Several OptiX SDK samples that heavily use OpenGL interoperability are intended to run on a single GPU and will perform more slowly when run on multiple GPUs. The samples: Cook, Hybrid Shadows, isgReflections, isgShadows, and SimpleAnimation will only run at full speed on a single GPU. The compiled versions of these samples can be constrained to a single GPU by setting the environment variable CUDA_VISIBLE_DEVICES = N, where "N" is the number of the GPU you wish to utilize. The OptiX sample3 is a convenient means to quickly identify your device ID's.

Performance Notes:

- There is a known performance decrease of 5-10% when running on current drivers (304.00 through 306.00 on Windows, 304.00 through 304.43 on Linux, CUDA library version 5.0.0 through 5.0.28 on Mac). This slow down should be reversed with the released NVIDIA 304 driver early this autumn.
- OptiX performance tracks very closely to a GPU's CUDA core count and core clock speed for a given GPU generation. The performance "value" of a CUDA core varies greatly between GPU generations (e.g., GT200, Fermi, Kepler) but can be compared directly within the same GPU generation.
- Different techniques may find varying performance results on different GPU generations. For example, path tracing benefited more from the Fermi architecture than did simpler techniques as compared to how they performed on GT200.
- OptiX takes advantage of multiple GPUs without using SLI. Multi-GPU scalability will vary with the workload being done, with longer and complex rendering (e.g., path tracing) scaling quite well while simple and fast rendering (e.g. Whitted) may scale far less.
- Mixing board types will reduce the memory size available to OptiX to that of the smallest GPU.
- Performance will be better when the entire scene fits within a single GPU’s memory. Adding additional GPUs increases performance, but does not increase the available memory beyond that of the smallest board. The entire scene must fit within GPU memory when paging is disabled or not available.
- For compute-intensive rendering, performance is currently fairly linear in the number of pixels displayed/rendered. Reducing resolution can make development on entry level boards or laptops more practical.
- Performance on Windows Vista and 7 will be somewhat slower than Windows XP or Linux due to the architecture of the Windows Display Driver Model (WDDM).
- Uninitialized variables can increase register pressure and negatively impact performance.
- Pass arguments by reference instead of value whenever possible when calling local functions for optimal performance.

Other Notes:

- **CMake 2.8.6** (at least 2.6.3; 2.8.6 is the current version and also works.)
  http://www.cmake.org/cmake/resources/software.html
  The executable installer http://www.cmake.org/files/v2.8/cmake-2.8.6-win32-x86.exe is recommended for Windows systems.