Release Notes for the NVIDIA® OptiX™ ray tracing engine

Version 2.5.0 Release Candidate — January 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:

Please post comments or support questions on the new NVIDIA developer forum that can be found here: http://forums.developer.nvidia.com/devforum/categories/tagged/optix&catid=151 (use the optix tag for all OptiX related posts). Questions that require confidentiality can be e-mailed to OptiX-Help@NVIDIA.com and someone on the development team will respond.

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” or “Fermi” class GPUs. Out-of-core ray tracing of large datasets is only supported on Quadro and Tesla GPUs.

Graphics Driver:

- The CUDA R275 or later driver is **required**. The latest drivers available are highly recommended (285.86 or later for Windows, 290.10 for Linux and the CUDA 4.0 driver extension for Mac). For the Mac, the driver extension module supplied with CUDA 4.0 or later will need to be installed. Driver versions beginning with 285.53 include very large speedup to OptiX compile times.
- The TCC driver (used by Tesla products on Windows 7) is **not** currently supported. Placing your Tesla hardware into WDDM mode will make it compatible with OptiX. This situation should be resolved later this year.

Operating System:

- Windows XP/Vista/7 32-bit or 64-bit; Linux RHEL 4.8 - 64-bit only, Ubuntu 10.10 - 32 and 64-bit 32-bit or 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.**
  OptiX 2.5 has been built with CUDA 4.0, 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 4.0. Note that CUDA 4.1 B2 is unsupported by this release candidate, but CUDA 4.1 should be supported by the final release.

- **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 10.6.

Enhancements from OptiX 2.1.1:

- **Out-of-Core Memory Paging** – scene sizes can now exceed the amount of physical memory on professional GPUs (Quadro or Tesla) to the extent there is host RAM available. This support is automatic, but can be overridden. The resulting performance will vary according to the amount the scene is paging – which is a combination of how much is exceeding GPU memory, how much of the scene is visible to the camera, and the extent of secondary rays in use. Some of the related changes include:
  - New BVH traverser – “BvhCompact” can compress data by up to a factor of four.
  - Added rtuCreateClusteredMesh() and rtuCreateClusteredMeshExt() for laying out data in a paging friendly manner.
  - Whether or not paging has been enabled can be queried with the rtContextGetAttribute() API call and specifying RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE.
  - Paging support can be disabled by calling the rtContextSetAttribute() API function with RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF.
- **Unlimited Textures** – when not using graphics interop textures, the first 127 textures will continue to take advantage of Texture Units, while any additional texture is now automatically stored in Global Memory at a minor performance cost.
- **GPU BVH Builder** – the original Lbvh builder has been replaced with the HLBVH2 algorithm to deliver far faster acceleration structure building than is possible via the CPU. The resulting traversal performance is comparable to CPU builders.
- Further optimizations for Fermi GPUs.
- Improved run time when using 64-bit PTX.
- Visual Studio 2010 support.
- Added RT_TIMEOUT_CALLBACK and rtContextSetTimeoutCallback(). OptiX can now periodically call a user provided function. This function can instruct OptiX to stop and return control to the caller without finishing the call. See programming guide for more information.
- Added new RTcontext attributes that can be queried or set.
  - RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS – for specifying the number of CPU threads OptiX can use for various tasks such as parallel CPU acceleration structure builds.
  - RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY – Get the amount of host memory OptiX is consuming between API calls (note that this isn’t a high water mark).
  - RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE – Indicates if paging has been enabled. Once paging has been enabled it cannot be forced off.
  - RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF – Force paging to be off regardless of whether OptiX attempts to enable it.

Enhancements from OptiX 2.1.1 (continued):

- Errors are now generated during compilation when calling an OptiX function in an illegal location (see table in Programming guide).
- Reduction in compile times for scenes with multiple ray types and programs only used by a single ray type.
- Added ability to throw an exception when rtIntersectChild() and rtReportIntersection() are called with an invalid index.
- Added rtContextSetAttribute().
- Added rtDeviceGetD3D9Device(), rtDeviceGetD3D10Device(), and rtDeviceGetD3D11Device(). These functions return the OptiX device ordinal that corresponds to the given D3D device.
- Added support for VS2010 in RTU's rtuCUDACompileString() and rtuCUDACompileFile().
- For GCC targets, symbol exports are now controlled using visibility attributes. Thus, OptiX now only exports the same set of symbols that the windows version exports.
- Updates to optixu headers
  - Added Matrix3x3 make_matrix3x3(Matrix4x4) function.
  - Fixed variable liveness issues with optix::intersect_triangle() and optix::refract().
  - Added luminanceCIE(float3).
  - Added operator== and operator!= for (uint3,uint3).
  - Added ContextObj::getDeviceName(), ContextObj::getDeviceAttribute() and ContextObj::getUsedHostMemory().
  - Added ContextObj::getCPUNumThreads(), ContextObj::getGPUPagingActive(), ContextObj::getGPUPagingForcedOff(), ContextObj::setCPUNumThreads() and ContextObj::setGPUPagingForcedOff() to match new context properties.
  - OptixPP's destroy methods now set the underlying pointer to zero, so the container can be queried to determine if it is still valid.
- Samples and sample infrastructure
  - Added new sample that illustrates a method of doing displacement surfaces without having to pretessellate the surface. All tessellation happens during intersection.
  - Added sample_phong_lobe(), get_phong_lobe_pdf() and tonemap() to samples/cuda/helpers.h.
  - Refactored much of the code that made use of meshes in the samples into a MeshScene class.
  - The path_tracer sample now comes with a multiple importance sampling mode. Use the -mis flag to try it.
- CMake
  - Look in paths that are installed by CUDA 4.0.
  - Added support for files with the same basename but different paths in the same target.
  - Working directory is now a subdirectory of CMakeFiles instead of the current binary directory.
  - Support for CUDA Toolkit installed in UNC paths.
  - Better support for flags and paths with spaces and quotes.

Known limitations with version 2.5.0:

- Out-of-core dataset paging does not presently work with GeForce cards.
- The Lbvh builder has been completely replaced with the HLBVH2 algorithm. Note that specifying Lbvh as the builder in a 64-bit host application while using 32-bit PTX will cause the MedianBvh builder to be utilized. The internal format for Acceleration Structure data has changed. Previous cached data will not be usable with 2.5 and must be regenerated.
- Support for building host-based acceleration structures in parallel has been disabled on Linux in this version of OptiX.
- OptiX currently does not support running with NVIDIA Parallel 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 previously noted, SLI is not required to achieve scaling across multiple GPUs.
- All GPUs used by OptiX must be of the same compute capability, such as compute capability 1.3 or 2.0. OptiX will automatically select the set of GPUs of the highest 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), the compute 2.1 device would only 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 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.
- Under certain circumstances it is possible that OpenGL buffer / OptiX interop does not work properly on display driver versions prior to 270.00. In particular cases the following sequence of commands will fail:
    - Create an OpenGL buffer and frame buffer object
    - Create an OpenGL texture
    - Attach the OpenGL texture to the frame buffer object (GL_COLOR_ATTACHMENT0)
    - Call glReadPixels to copy from GL_COLOR_ATTACHMENT0 to the OpenGL buffer
    - Create an OptiX buffer from the OpenGL buffer
    - Launching the OptiX kernel might fail now

  When this behavior exhibits try to use the OpenGL texture directly in OptiX and skip the copy operation to the OpenGL buffer via glReadPixels(); e.g. as shown in the isgShadows sample.
- 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.

Performance Notes:

- OptiX performance tracks very closely to a GPU's CUDA core count and core clock speed for a given GPU generation.
- OptiX takes advantage of multiple GPUs without using SLI. It is not recommended to configure GPUs in SLI mode for OptiX applications. Multi-GPU scalability will vary with the workload being done, with longer and complex rendering (e.g., path tracing) scaling quite well while fast and simple rendering (e.g. Whitted or Cook) scaling much 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. If paging is disabled (see below), the entire scene must fit on the GPU.
- 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 practical.
- Performance on Windows Vista and 7 may be somewhat slower than Windows XP due to the architecture of the Windows Display Driver Model.
- 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.