

NVIDIA CUDA Software Development Kit (CUDA SDK)
Release Notes
Version 2.0 for 32-bit or 64-bit Windows Vista or XP

Please, also refer to the release notes of version 2.0 of CUDA, installed by the CUDA Toolkit installer.

TABLE OF CONTENTS

- I. Installation Instructions
 - II. Creating Your Own CUDA Program
 - III. Known Issues
 - IV. Frequently Asked Questions
 - V. Change Log
-

I. Installation Instructions

- 0. CUDA 2.0 requires at least version 177.35 of the Windows Vista or Windows XP NVIDIA Display Driver. See the NVIDIA CUDA Toolkit 2.0 release notes for more information.

Please make sure to read the Driver Installation Hints Document before you install the driver:
http://www.nvidia.com/object/driver_installation_hints.html

- 1. Uninstall any previous versions of the NVIDIA CUDA Toolkit and NVIDIA CUDA SDK.
You can uninstall the NVIDIA CUDA Toolkit through the Start menu:
Start menu->All Programs->NVIDIA Corporation->CUDA Toolkit->Uninstall CUDA
You can uninstall the NVIDIA CUDA SDK through the Start menu:
Start menu->All Programs->NVIDIA Corporation
->NVIDIA CUDA SDK->Uninstall NVIDIA CUDA SDK
- 2. Install version 2.0 of the NVIDIA CUDA Toolkit by running
NVIDIA_CUDA_Toolkit_2.0_[Win|Vista][32|64].exe corresponding to your operating system.
- 3. Install version 2.0 of the NVIDIA CUDA SDK by running
NVIDIA_CUDA_SDK_2.0_[Win|Vista][32|64].exe corresponding to your operating system.
- 4. Build the 32-bit and/or 64-bit, release, debug, emurelease, and/or emudebug configurations of the SDK project examples using the provided *.sln solution files for Microsoft Visual Studio Version 8 or *_vc7.sln solution files for Microsoft Visual Studio Version 7.
You can:
 - either use the solution files located in each of the examples' directories in "NVIDIA CUDA SDK\projects",

- or use the global solution files `release.sln` or `release_vc7.sln` located in `"NVIDIA CUDA SDK\projects"`.

Notes:

- The `simpleD3D` example requires to have a Direct3D SDK installed and the VC++ directory paths (located in `Tools->Options...`) properly setup.
- Most samples link to a utility library called "cutil" whose source code is in `"NVIDIA CUDA SDK\common"`. The release and emurelease versions of these samples link to `cutil[32|64].lib` and dynamically load `cutil[32|64].dll`. The debug and emudebug versions of these samples link to `cutil[32D|64D].lib` and dynamically load `cutil[32D|64D].dll`.
To build the 32-bit and/or 64-bit, release and/or debug configurations of the cutil library, use the solution files located in `"NVIDIA CUDA SDK\common"`. The output of the compilation goes to `"NVIDIA CUDA SDK\common\lib"`:
 - `cutil[32|64].lib` and `cutil[32D|64D].lib` are the release and debug import libraries,
 - `cutil[32|64].dll` and `cutil[32D|64D].dll` are the release and debug dynamic-link libraries, which get also copied to `"NVIDIA CUDA SDK\common\bin\win[32|64]\[release|emurelease]"` and `"NVIDIA CUDA SDK\common\bin\win[32|64]\[debug|emudebug]"` respectively;

5. Run the examples from the `release`, `debug`, `emurelease`, or `emudebug` directories located in `"NVIDIA CUDA SDK\bin\win[32|64]\[release|debug|emurelease|emudebug]"`.

Notes:

- The release and debug configurations require a CUDA-capable GPU to run properly (see Appendix A.1 of the CUDA Programming Guide for a complete list of CUDA-capable GPUs).
- The emurelease and emudebug configurations run in device emulation mode, and therefore do not require a CUDA-capable GPU to run properly.

II. Creating Your Own CUDA Program

Creating a new CUDA Program using the NVIDIA CUDA SDK infrastructure is easy. We have provided a "template" project that you can copy and modify to suit your needs. Just follow these steps:

1. Copy the content of `"NVIDIA CUDA SDK\projects\template"` to a directory of your own `"NVIDIA CUDA SDK\projects\myproject"`
2. Edit the filenames of the project to suit your needs.
3. Edit the `*.sln`, `*.vcproj` and source files. Just search and replace all occurrences of "template" with "myproject".
4. Build the 32-bit and/or 64-bit, release, debug, emurelease, and/or emudebug configurations using `myproject.sln` or `myproject_vc7.sln`.

5. Run `myproject.exe` from the `release`, `debug`, `emurelease`, or `emudebug` directories located in `"NVIDIA CUDA SDK\bin\win[32|64]\[release|debug|emurelease|emudebug]"`.

(It should print "Test PASSED".)

6. Now modify the code to perform the computation you require. See the CUDA Programming Guide for details of programming in CUDA.

III. Known Issues

Note: Please see the CUDA Toolkit release notes for additional issues.

1. In code sample `alignedTypes`, the following aligned type does not provide maximum throughput because of a compiler bug:

```
typedef struct __align__(16) {
    unsigned int r, g, b;
} RGB32;
```

The workaround is to use the following type instead:

```
typedef struct __align__(16) {
    unsigned int r, g, b, a;
} RGBA32;
```

as illustrated in the sample.

IV. Frequently Asked Questions

The Official CUDA FAQ is available online on the NVIDIA CUDA Forums:
<http://forums.nvidia.com/index.php?showtopic=36286>

Note: Please also see the CUDA Toolkit release notes for additional Frequently Asked Questions.

V. Change Log

Release 2.0 Beta2

- * 2 new code samples:
 `cudaVideoDecode` and `simpleVoteIntrinsics`

Release 2.0 Beta

- * Updated to the 2.0 CUDA Toolkit
- * `CUT_DEVICE_INIT` macro modified to take command line arguments. All samples now support specifying the CUDA device to run on from the command line ("`-device=n`").
- * `deviceQuery` sample: Updated to query number of multiprocessors and overlap flag.
- * `fluidsD3D` sample: Renamed to `fluidsD3D9` and updated to the new Direct3D interoperability API.
- * `multiGPU` sample: Renamed to `simpleMultiGPU`.
- * `reduction`, `MonteCarlo`, and `binomialOptions` samples: updated with optional double precision support for upcoming hardware.
- * `simpleAtomics` sample: Renamed to `simpleAtomicIntrinsics`.

- * simpleD3D sample: Renamed to simpleD3D9 and updated to the new Direct3D interoperability API.
- * 7 new code samples:
dct8x8, quasirandomGenerator, recursiveGaussian, simpleD3D9Texture, simpleTexture3D, threadMigration, and volumeRender

Release 1.1

- * Updated to the 1.1 CUDA Toolkit
- * Removed isInteropSupported() from cutil: graphics interoperability now works on multi-GPU systems
- * MonteCarlo sample: Improved performance. Previously it was very fast for large numbers of paths and options, now it is also very fast for small- and medium-sized runs.
- * Transpose sample: updated kernel to use a 2D shared memory array for clarity, and optimized bank conflicts.
- * 15 new code samples:
asyncAPI, cudaOpenMP, eigenvalues, fastWalshTransform, histogram256, lineOfSight, Mandelbrot, marchingCubes, MonteCarloMultiGPU, nbody, oceanFFT, particles, reduction, simpleAtomics, and simpleStreams

Release 1.0

- * Updated to the 1.0 CUDA Toolkit.
- * Added 4 new code samples: convolutionTexture, convolutionFFT2D, histogram64, and SobelFilter.
- * All graphics interop samples now call the cutil library function isInteropSupported(), which returns false on machines with multiple CUDA GPUs, currently (see above).
- * When compiling in DEBUG mode, CU_SAFE_CALL() now calls cuCtxSynchronize() and CUDA_SAFE_CALL() and CUDA_CHECK_ERROR() now call cudaThreadSynchronize() in order to return meaningful errors. This means that performance might suffer in DEBUG mode.

Release 0.9

- * Updated to the 0.9 CUDA Toolkit.
- * Added 6 new code samples: MersenneTwister, MonteCarlo, imageDenoising, simpleTemplates, deviceQuery, alignedTypes, and convolutionSeparable.
- * Removed 3 old code samples:
 - vectorLoads and loadUByte replaced by alignedTypes;
 - convolution replaced by convolutionSeparable.

Release 0.8.1 beta

- * Standardized project and file naming conventions. Several project names changed as a result.
- * cppIntegration output now matches the other samples ("Test PASSED").
- * Modified transpose16 sample to transpose arbitrary matrices efficiently, and renamed it to transpose.
- * Added 11 new code samples: bandwidthTest, binomialOptions, BlackScholes, boxFilter, convolution, dxTC, fluidsGL, multiGPU, postProcessGL, simpleTextureDrv, and vectorLoads.

Release 0.8 beta

- * First public release.