CUDA SAMPLES

TRM-06704-001_v9.1 | November 2017

Reference Manual

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. RELEASE NOTES

This section describes the release notes for the CUDA Samples only. For the release notes for the whole CUDA Toolkit, please see CUDA Toolkit Release Notes.

1.1. CUDA 9.0

- Added **7_CUDALibraries/nvgraph_SpectralClustering**. Demonstrates Spectral Clustering using NVGRAPH Library.
- Added **6_Advanced/warpAggregatedAtomicsCG**. Demonstrates warp aggregated atomics using Cooperative Groups.
- Added **6_Advanced/reductionMultiBlockCG**. Demonstrates single pass reduction using Multi Block Cooperative Groups.
- Added **6_Advanced/conjugateGradientMultiBlockCG**. Demonstrates a conjugate gradient solver on GPU using Multi Block Cooperative Groups.
- Added Cooperative Groups(CG) support to several samples notable ones to name are **6_Advanced/cdpQuadtree**, **6_Advanced/cdpAdvancedQuicksort**, **6_Advanced/threadFenceReduction**, **3_Imaging/dxtc**, **4_Finance/MonteCarloMultiGPU**, **0_Simple/matrixMul_nvrtc**.
- Added **0_Simple/simpleCooperativeGroups**. Illustrates basic usage of Cooperative Groups within the thread block.
- Added **0_Simple/cudaTensorCoreGemm**. Demonstrates a GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API introduced in CUDA 9, as well as the new Tensor Cores introduced in the Volta chip family.
- Updated **0_Simple/simpleVoteIntrinsics** to use newly added *_sync equivalent of the vote intrinsics _any, _all.
- Updated **6_Advanced/shfl_scan** to use newly added *_sync equivalent of the shfl intrinsics.

1.2. CUDA 8.0

- Added **7_CUDALibraries/FilterBorderControlNPP**. Demonstrates how any border version of an NPP filtering function can be used in the most common mode (with border control enabled), can be used to duplicate the results of the equivalent non-border version of the NPP function, and can be used to enable and disable border control on various source image edges depending on what portion of the source image is being used as input.

- Added **7_CUDALibraries/cannyEdgeDetectorNPP**. Demonstrates the recommended parameters to use with the nppiFilterCannyBorder_8u_C1R Canny Edge Detection image filter function. This function expects a single channel 8-bit grayscale input image. You can generate a grayscale image from a color image by first calling nppiColorToGray() or nppiRGBToGray(). The Canny Edge Detection function combines and improves on the techniques required to produce an edge detection image using multiple steps.
- Added **7_CUDALibraries/cuSolverSp_LowlevelCholesky**. Demonstrates Cholesky factorization using cuSolverSP's low level APIs.
- Added **7_CUDALibraries/cuSolverSp_LowlevelQR**. Demonstrates QR factorization using cuSolverSP's low level APIs.
- Added **7_CUDALibraries/BiCGStab**. Demonstrates Bi-Conjugate Gradient Stabilized (BiCGStab) iterative method for nonsymmetric and symmetric positive definite linear systems using CUSPARSE and CUBLAS
- Added **7_CUDALibraries/nvgraph_Pagerank**. Demonstrates Page Rank computation using nvGRAPH Library.
- Added **7_CUDALibraries/nvgraph_SemiRingSpMV**. Demonstrates Semi-Ring SpMV using nvGRAPH Library.
- Added **7_CUDALibraries/nvgraph_SSSP**. Demonstrates Single Source Shortest Path(SSSP) computation using nvGRAPH Library.
- Added **7_CUDALibraries/simpleCUBLASXT**. Demonstrates simple example to use CUBLAS-XT library.
- Added **6_Advanced/c++11_cuda**. Demonstrates C++11 feature support in CUDA.
- Added **1_Utilities/topologyQuery**. Demonstrates how to query the topology of a system with multiple GPU.
- Added **0_Simple/fp16ScalarProduct**. Demonstrates scalar product calculation of two vectors of FP16 numbers.
- Added **0_Simple/systemWideAtomics**. Demonstrates system wide atomic instructions on migratable memory.
- Removed **0_Simple/template_runtime**. Its purpose is served by **0_Simple/template**.

1.3. CUDA 7.5

- Added **7_CUDALibraries/cuSolverDn_LinearSolver**. Demonstrates how to use the CUSOLVER library for performing dense matrix factorization using cuSolverDN's LU, QR and Cholesky factorization functions.
- Added **7_CUDALibraries/cuSolverRf**. Demonstrates how to use cuSolverRF, a sparse re-factorization package of the CUSOLVER library.
- Added **7_CUDALibraries/cuSolverSp_LinearSolver**. Demonstrates how to use cuSolverSP which provides sparse set of routines for sparse matrix factorization.

- The **2_Graphics/simpleD3D9**, **2_Graphics/simpleD3D9Texture**, **3_Imaging/cudaDecodeD3D9**, and **5_Simulations/fluidsD3D9** samples have been modified to use the Direct3D 9Ex API instead of the Direct3D 9 API.
- The **7_CUDALibraries/grabcutNPP** and **7_CUDALibraries/imageSegmentationNPP** samples have been removed. These samples used the NPP graphcut APIs, which have been deprecated in CUDA 7.5.

1.4. CUDA 7.0

- Removed support for Windows 32-bit builds.
- The Makefile x86_64=1 and ARMv7=1 options have been deprecated. Please use TARGET_ARCH to set the targeted build architecture instead.
- The Makefile GCC option has been deprecated. Please use HOST_COMPILER to set the host compiler instead.
- The CUDA Samples are no longer shipped as prebuilt binaries on Windows. Please use VS Solution files provided to build respective executable.
- Added **0_Simple/clock_nvrtc**. Demonstrates how to compile clock function kernel at runtime using libNVRTC to measure the performance of kernel accurately.
- Added **0_Simple/inlinePTX_nvrtc**. Demonstrates compilation of CUDA kernel having PTX embedded at runtime using libNVRTC.
- Added **0_Simple/matrixMul_nvrtc**. Demonstrates compilation of matrix multiplication CUDA kernel at runtime using libNVRTC.
- Added **0_Simple/simpleAssert_nvrtc**. Demonstrates compilation of CUDA kernel having assert() at runtime using libNVRTC.
- Added **0_Simple/simpleAtomicIntrinsics_nvrtc**. Demonstrates compilation of CUDA kernel performing atomic operations at runtime using libNVRTC.
- Added **0_Simple/simpleTemplates_nvrtc**. Demonstrates compilation of templatized dynamically allocated shared memory arrays CUDA kernel at runtime using libNVRTC.
- Added **0_Simple/simpleVoteIntrinsics_nvrtc**. Demonstrates compilation of CUDA kernel which uses vote intrinsics at runtime using libNVRTC.
- Added **0_Simple/vectorAdd_nvrtc**. Demonstrates compilation of CUDA kernel performing vector addition at runtime using libNVRTC.
- Added **4_Finance/binomialOptions_nvrtc**. Demonstrates runtime compilation using libNVRTC of CUDA kernel which evaluates fair call price for a given set of European options under binomial model.
- Added **4_Finance/BlackScholes_nvrtc**. Demonstrates runtime compilation using libNVRTC of CUDA kernel which evaluates fair call and put prices for a given set of European options by Black-Scholes formula.
- Added **4_Finance/quasirandomGenerator_nvrtc**. Demonstrates runtime compilation using libNVRTC of CUDA kernel which implements Niederreiter Quasirandom Sequence Generator and Inverse Cumulative Normal Distribution functions for the generation of Standard Normal Distributions.

1.5. CUDA 6.5

- Added **7_CUDALibraries/cuHook**. Demonstrates how to build and use an intercept library with CUDA.
- Added **7_CUDALibraries/simpleCUFFT_callback**. Demonstrates how to compute a 1D-convolution of a signal with a filter using a user-supplied CUFFT callback routine, rather than a separate kernel call.
- Added **7_CUDALibraries/simpleCUFFT_MGPU**. Demonstrates how to compute a 1D-convolution of a signal with a filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain on Multiple GPUs.
- Added **7_CUDALibraries/simpleCUFFT_2d_MGPU**. Demonstrates how to compute a 2D-convolution of a signal with a filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain on Multiple GPUs.
- Removed **3_Imaging/cudaEncode**. Support for the CUDA Video Encoder (NVCUVENC) has been removed.
- Removed **4_Finance/ExcelCUDA2007**. The topic will be covered in a blog post at Parallel Forall.
- Removed **4_Finance/ExcelCUDA2010**. The topic will be covered in a blog post at Parallel Forall.
- The **4_Finance/binomialOptions** sample is now restricted to running on GPUs with SM architecture 2.0 or greater.
- The **4_Finance/quasirandomGenerator** sample is now restricted to running on GPUs with SM architecture 2.0 or greater.
- The **7_CUDALibraries/boxFilterNPP** sample now demonstrates how to use the static NPP libraries on Linux and Mac.
- The **7_CUDALibraries/conjugateGradient** sample now demonstrates how to use the static CUBLAS and CUSPARSE libraries on Linux and Mac.
- The **7_CUDALibraries/MersenneTwisterGP11213** sample now demonstrates how to use the static CURAND library on Linux and Mac.

1.6. CUDA 6.0

- New featured samples that support a new CUDA 6.0 feature called UVM-Lite
- Added **0_Simple/UnifiedMemoryStreams** - new CUDA sample that demonstrates the use of OpenMP and CUDA streams with Unified Memory on a single GPU.
- Added **1_Utilities/p2pBandwidthTestLatency** - new CUDA sample that demonstrates how measure latency between pairs of GPUs with P2P enabled and P2P disabled.
- Added **6_Advanced/StreamPriorities** - This sample demonstrates basic use of the new CUDA 6.0 feature stream priorities.

- Added **7_CUDALibraries/ConjugateGradientUM** - This sample implements a conjugate gradient solver on GPU using cuBLAS and cuSPARSE library, using Unified Memory.

1.7. CUDA 5.5

- Linux makefiles have been updated to generate code for the AMRv7 architecture. Only the ARM hard-float floating point ABI is supported. Both native ARMv7 compilation and cross compilation from x86 is supported
- Performance improvements in CUDA toolkit for Kepler GPUs (SM 3.0 and SM 3.5)
- Makefiles projects have been updated to properly find search default paths for OpenGL, CUDA, MPI, and OpenMP libraries for all OS Platforms (Mac, Linux x86, Linux ARM).
- Linux and Mac project Makefiles now invoke NVCC for building and linking projects.
- Added **0_Simple/cppOverload** - new CUDA sample that demonstrates how to use C++ overloading with CUDA.
- Added **6_Advanced/cdpBezierTessellation** - new CUDA sample that demonstrates an advanced method of implementing Bezier Line Tessellation using CUDA Dynamic Parallelism. Requires compute capability 3.5 or higher.
- Added **7_CUDALibrariess/jpegNPP** - new CUDA sample that demonstrates how to use NPP for JPEG compression on the GPU.
- CUDA Samples now have better integration with Nsight Eclipse IDE.
- **6_Advanced/ptxjit** sample now includes a new API to demonstrate PTX linking at the driver level.

1.8. CUDA 5.0

- New directory structure for CUDA samples. Samples are classified accordingly to categories: **0_Simple**, **1_Utilities**, **2_Graphics**, **3_Imaging**, **4_Finance**, **5_Simulations**, **6_Advanced**, and **7_CUDALibraries**
- Added **0_Simple/simpleIPC** - CUDA Runtime API sample is a very basic sample that demonstrates Inter Process Communication with one process per GPU for computation. Requires Compute Capability 2.0 or higher and a Linux Operating System.
- Added **0_Simple/simpleSeparateCompilation** - demonstrates a CUDA 5.0 feature, the ability to create a GPU device static library and use it within another CUDA kernel. This example demonstrates how to pass in a GPU device function (from the GPU device static library) as a function pointer to be called. Requires Compute Capability 2.0 or higher.
- Added **2_Graphics/bindlessTexture** - demonstrates use of **cudaSurfaceObject**, **cudaTextureObject**, and MipMap support in CUDA. Requires Compute Capability 3.0 or higher.

- Added **`3_Imaging/stereoDisparity`** - demonstrates how to compute a stereo disparity map using SIMD SAD (Sum of Absolute Difference) intrinsics. Requires Compute Capability 2.0 or higher.
- Added **`0_Simple/cdpSimpleQuicksort`** - demonstrates a simple quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added **`0_Simple/cdpSimplePrint`** - demonstrates simple printf implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added **`6_Advanced/cdpLUDecomposition`** - demonstrates LU Decomposition implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added **`6_Advanced/cdpAdvancedQuicksort`** - demonstrates an advanced quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added **`6_Advanced/cdpQuadtree`** - demonstrates Quad Trees implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added **`7_CUDALibraries/simpleDevLibCUBLAS`** - implements a simple cuBLAS function calls that call GPU device API library running cuBLAS functions. cuBLAS device code functions take advantage of CUDA Dynamic Parallelism and requires compute capability of 3.5 or higher.

1.9. CUDA 4.2

- Added **`segmentationTreeThrust`** - demonstrates a method to build image segmentation trees using Thrust. This algorithm is based on Boruvka's MST algorithm.

1.10. CUDA 4.1

- Added **`MersenneTwisterGP11213`** - implements Mersenne Twister GP11213, a pseudorandom number generator using the **`cuRAND`** library.
- Added **`HSOpticalFlow`** - When working with image sequences or video it's often useful to have information about objects movement. Optical flow describes apparent motion of objects in image sequence. This sample is a Horn-Schunck method for optical flow written using CUDA.
- Added **`volumeFiltering`** - demonstrates basic volume rendering and filtering using 3D textures.
- Added **`simpleCubeMapTexture`** - demonstrates how to use **`texcubemap`** fetch instruction in a CUDA C program.
- Added **`simpleAssert`** - demonstrates how to use GPU assert in a CUDA C program.
- Added **`grabcutNPP`** - CUDA implementation of Rother et al. GrabCut approach using the 8 neighborhood **`NPP`** Graphcut primitive introduced in CUDA 4.1. (C.

Rother, V. Kolmogorov, A. Blake. *GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. ACM Transactions on Graphics (SIGGRAPH'04)*, 2004).

Chapter 2. GETTING STARTED

The CUDA Samples are an educational resource provided to teach CUDA programming concepts. The CUDA Samples are not meant to be used for performance measurements.

For system requirements and installation instructions, please refer to the Linux Installation Guide, the Windows Installation Guide, and the Mac Installation Guide.

2.1. Getting CUDA Samples

Windows

On Windows, the CUDA Samples are installed using the CUDA Toolkit Windows Installer. By default, the CUDA Samples are installed in:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.1\
```

The installation location can be changed at installation time.

Linux

On Linux, to install the CUDA Samples, the CUDA toolkit must first be installed. See the Linux Installation Guide for more information on how to install the CUDA Toolkit.

Then the CUDA Samples can be installed by running the following command, where <target_path> is the location where to install the samples:

```
$ cuda-install-samples-9.1.sh <target_path>
```

Mac OSX

On Mac OSX, to install the CUDA Samples, the CUDA toolkit must first be installed. See the Mac Installation Guide for more information on how to install the CUDA Toolkit.

Then the CUDA Samples can be installed by running the following command, where <target_path> is the location where to install the samples:

```
$ cuda-install-samples-9.1.sh <target_path>
```

2.2. Building Samples

Windows

The Windows samples are built using the Visual Studio IDE. Solution files (.sln) are provided for each supported version of Visual Studio, using the format:

```
*_vs<version>.sln - for Visual Studio <version>
```

Complete samples solution files exist at:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.1\
```

Each individual sample has its own set of solution files at:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.1\<sample_dir>\
```

To build/examine all the samples at once, the complete solution files should be used. To build/examine a single sample, the individual sample solution files should be used.

> Some samples require that the Microsoft DirectX SDK (June 2010 or newer) be installed and that the VC++ directory paths are properly set up (**Tools** > **Options...**). Check DirectX Dependencies section for details.

Linux

The Linux samples are built using makefiles. To use the makefiles, change the current directory to the sample directory you wish to build, and run **make**:

```
$ cd <sample_dir>
$ make
```

The samples makefiles can take advantage of certain options:

- **TARGET_ARCH=<arch>** - cross-compile targeting a specific architecture. Allowed architectures are x86_64, armv7l, aarch64, and ppc64le.

  By default, TARGET_ARCH is set to HOST_ARCH. On a x86_64 machine, not setting TARGET_ARCH is the equvalent of setting TARGET_ARCH=x86_64.

  ```
  $ make TARGET_ARCH=x86_64
  $ make TARGET_ARCH=armv7l
  $ make TARGET_ARCH=aarch64
  $ make TARGET_ARCH=ppc64le
  ```

  See here for more details.
- **dbg=1** - build with debug symbols

  ```
  $ make dbg=1
  ```
- **SMS="A B ..."** - override the SM architectures for which the sample will be built, where **"A B ..."** is a space-delimited list of SM architectures. For example, to generate SASS for SM 20 and SM 30, use **SMS="20 30"**.

```
$ make SMS="20 30"
```

- **HOST_COMPILER=<host_compiler>** - override the default g++ host compiler. See the Linux Installation Guide for a list of supported host compilers.

```
$ make HOST_COMPILER=g++
```

Mac

The Mac samples are built using makefiles. To use the makefiles, change directory into the sample directory you wish to build, and run **make**:

```
$ cd <sample_dir>
$ make
```

The samples makefiles can take advantage of certain options:

- **dbg=1** - build with debug symbols

```
$ make dbg=1
```

- **SMS="A B ..."** - override the SM architectures for which the sample will be built, where **"A B ..."** is a space-delimited list of SM architectures. For example, to generate SASS for SM 20 and SM 30, use **SMS="20 30"**.

```
$ make SMS="A B ..."
```

- **HOST_COMPILER=<host_compiler>** - override the default clang host compiler. See the Mac Installation Guide for a list of supported host compilers.

```
$ make HOST_COMPILER=clang
```

2.3. CUDA Cross-Platform Samples

This section describes the options used to build cross-platform samples. **TARGET_ARCH=<arch>** and **TARGET_OS=<os>** should be chosen based on the supported targets shown below. **TARGET_FS=<path>** can be used to point nvcc to libraries and headers used by the sample.

Table 1 Supported Target Arch/OS Combinations

		TARGET OS			
		linux	darwin	android	qnx
TARGET ARCH	x86_64	YES	YES	NO	NO
	armv7l	YES	NO	YES	YES
	aarch64	NO	NO	YES	NO
	ppc64le	YES	NO	NO	NO

TARGET_ARCH

The target architecture must be specified when cross-compiling applications. If not specified, it defaults to the host architecture. Allowed architectures are:

- **x86_64** - 64-bit x86 CPU architecture
- **armv7l** - 32-bit ARM CPU architecture, like that found on Jetson TK1
- **aarch64** - 64-bit ARM CPU architecture, found on certain Android systems
- **ppc64le** - 64-bit little-endian IBM POWER8 architecture

TARGET_OS

The target OS must be specified when cross-compiling applications. If not specified, it defaults to the host OS. Allowed OSes are:

- **linux** - for any Linux distributions
- **darwin** - for Mac OS X
- **android** - for any supported device running Android
- **qnx** - for any supported device running QNX

TARGET_FS

The most reliable method to cross-compile the CUDA Samples is to use the TARGET_FS variable. To do so, mount the target's filesystem on the host, say at **/mnt/target**. This is typically done using **exportfs**. In cases where **exportfs** is unavailable, it is sufficient to copy the target's filesystem to **/mnt/target**. To cross-compile a sample, execute:

```
$ make TARGET_ARCH=<arch> TARGET_OS=<os> TARGET_FS=/mnt/target
```

Copying Libraries

If the TARGET_FS option is not available, the libraries used should be copied from the target system to the host system, say at **/opt/target/libs**. If the sample uses GL, the GL headers must also be copied, say at **/opt/target/include**. The linker must then be told where the libraries are with the **-rpath-link** and/or **-L** options. To ignore unresolved symbols from some libraries, use the **--unresolved-symbols** option as shown below. **SAMPLE_ENABLED** should be used to force the sample to build. For example, to cross-compile a sample which uses such libraries, execute:

```
$ make TARGET_ARCH=<arch> TARGET_OS=<os> \
           EXTRA_LDFLAGS="-rpath-link=/opt/target/libs -L/opt/target/libs --
unresolved-symbols=ignore-in-shared-libs" \
           EXTRA_CCFLAGS="-I /opt/target/include" \
           SAMPLE_ENABLED=1
```

2.4. Using CUDA Samples to Create Your Own CUDA Projects

2.4.1. Creating CUDA Projects for Windows

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

(**<category>** refers to one of the following folders: **0_Simple**, **1_Utilities**, **2_Graphics**, **3_Imaging**, **4_Finance**, **5_Simulations**, **6_Advanced**, **7_CUDALibraries**.)

1. Copy the content of:
```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.1\<category>\template
```
   to a directory of your own:
```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.1\<category>\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 or debug configurations using:

   myproject_vs<version>.sln
5. Run **myproject.exe** from the **release** or **debug** directories located in:
```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.1\bin\win[32|64]\[release|
debug]
```
6. Now modify the code to perform the computation you require.
   See the *CUDA Programming Guide* for details of programming in CUDA.

2.4.2. Creating CUDA Projects for Linux

> The default installation folder **<SAMPLES_INSTALL_PATH>** is **NVIDIA_CUDA_9.1_Samples** and <category> is one of the following: **0_Simple**, **1_Utilities**, **2_Graphics**, **3_Imaging**, **4_Finance**, **5_Simulations**, **6_Advanced**, **7_CUDALibraries.**

Creating a new CUDA Program using the NVIDIA CUDA Samples 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 **template** project:
```
cd <SAMPLES_INSTALL_PATH>/<category>
cp -r template <myproject>cd <SAMPLES_INSTALL_PATH>/<category>
```

2. Edit the filenames of the project to suit your needs:

```
mv template.cu myproject.cu
mv template_cpu.cpp myproject_cpu.cpp
```

3. Edit the **Makefile** and source files.

   Just search and replace all occurrences of **template** with **myproject**.

4. Build the project as (release):

```
make
```

   To build the project as (debug), use "make dbg=1":

```
make dbg=1
```

5. Run the program:

```
../../bin/x86_64/linux/release/myproject
```

6. Now modify the code to perform the computation you require.

   See the *CUDA Programming Guide* for details of programming in CUDA.

2.4.3. Creating CUDA Projects for Mac OS X

The default installation folder **<SAMPLES_INSTALL_PATH>** is: **/Developer/NVIDIA/CUDA-9.1/samples**

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

(<category> is one of the following: **0_Simple**, **1_Utilities**, **2_Graphics**, **3_Imaging**, **4_Finance**, **5_Simulations**, **6_Advanced**, **7_CUDALibraries**.)

1. Copy the template project:

```
cd <SAMPLES_INSTALL_PATH>/<category>
cp -r template <myproject>
```

2. Edit the filenames of the project to suit your needs:

```
mv template.cu myproject.cu
mv template_cpu.cpp myproject_cpu.cpp
```

3. Edit the **Makefile** and source files.

   Just search and replace all occurrences of **template** with **myproject**.

4. Build the project as (release):

```
make
```

   Note: To build the project as (debug), use "make dbg=1"

```
make dbg=1
```

5. Run the program:

```
../../bin/x86_64/darwin/release/myproject
```

   (It should print **PASSED**.)

6. Now modify the code to perform the computation you require.

   See the *CUDA Programming Guide* for details of programming in CUDA.

Chapter 3. SAMPLES REFERENCE

This document contains a complete listing of the code samples that are included with the NVIDIA CUDA Toolkit. It describes each code sample, lists the minimum GPU specification, and provides links to the source code and white papers if available.

The code samples are divided into the following categories:

Simple Reference
: Basic CUDA samples for beginners that illustrate key concepts with using CUDA and CUDA runtime APIs.

Utilities Reference
: Utility samples that demonstrate how to query device capabilities and measure GPU/CPU bandwidth.

Graphics Reference
: Graphical samples that demonstrate interoperability between CUDA and OpenGL or DirectX.

Imaging Reference
: Samples that demonstrate image processing, compression, and data analysis.

Finance Reference
: Samples that demonstrate parallel algorithms for financial computing.

Simulations Reference
: Samples that illustrate a number of simulation algorithms implemented with CUDA.

Advanced Reference
: Samples that illustrate advanced algorithms implemented with CUDA.

Cudalibraries Reference
: Samples that illustrate how to use CUDA platform libraries (NPP, cuBLAS, cuFFT, cuSPARSE, and cuRAND).

3.1. Simple Reference

asyncAPI

This sample uses CUDA streams and events to overlap execution on CPU and GPU.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events
Supported OSes	Linux, Windows, OS X

cdpSimplePrint - Simple Print (CUDA Dynamic Parallelism)

This sample demonstrates simple printf implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows, OS X

cdpSimpleQuicksort - Simple Quicksort (CUDA Dynamic Parallelism)

This sample demonstrates simple quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows, OS X

clock - Clock

This example shows how to use the clock function to measure the performance of block of threads of a kernel accurately.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows, OS X

cppIntegration - C++ Integration

This example demonstrates how to integrate CUDA into an existing C++ application, i.e. the CUDA entry point on host side is only a function which is called from C++ code and only the file containing this function is compiled with nvcc. It also demonstrates that vector types can be used from cpp.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Supported OSes	Linux, Windows, OS X

cudaOpenMP

This sample demonstrates how to use OpenMP API to write an application for multiple GPUs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	OpenMP
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Systems Integration, OpenMP, Multithreading
Supported OSes	Linux, Windows

inlinePTX - Using Inline PTX

A simple test application that demonstrates a new CUDA 4.0 ability to embed PTX in a CUDA kernel.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaMemcpy
Key Concepts	Performance Strategies, PTX Assembly, CUDA Driver API
Supported OSes	Linux, Windows, OS X

matrixMul - Matrix Multiplication (CUDA Runtime API Version)

This sample implements matrix multiplication and is exactly the same as Chapter 6 of the programming guide. It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication. To illustrate GPU performance for matrix multiply, this sample also shows how to use the new CUDA 4.0 interface for CUBLAS to demonstrate high-performance performance for matrix multiplication.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Linear Algebra
Supported OSes	Linux, Windows, OS X

matrixMulCUBLAS - Matrix Multiplication (CUBLAS)

This sample implements matrix multiplication from Chapter 3 of the programming guide. To illustrate GPU performance for matrix multiply, this sample also shows how to use the new CUDA 4.0 interface for CUBLAS to demonstrate high-performance performance for matrix multiplication.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMalloc, cudaFree, cudaMemcpy, cublasCreate, cublasSgemm
Key Concepts	CUDA Runtime API, Performance Strategies, Linear Algebra, CUBLAS
Supported OSes	Linux, Windows, OS X

matrixMulDrv - Matrix Multiplication (CUDA Driver API Version)

This sample implements matrix multiplication and uses the new CUDA 4.0 kernel launch Driver API. It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication. CUBLAS provides high-performance matrix multiplication.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, Matrix Multiply
Supported OSes	Linux, Windows, OS X

simpleAssert

This CUDA Runtime API sample is a very basic sample that implements how to use the assert function in the device code. Requires Compute Capability 2.0 .

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaMemcpy
Key Concepts	Assert
Supported OSes	Linux, Windows

simpleAtomicIntrinsics - Simple Atomic Intrinsics

A simple demonstration of global memory atomic instructions. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Atomic Intrinsics
Supported OSes	Linux, Windows, OS X

simpleCallback - Simple CUDA Callbacks

This sample implements multi-threaded heterogeneous computing workloads with the new CPU callbacks for CUDA streams and events introduced with CUDA 5.0.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaStreamCreate, cudaMemcpyAsync, cudaStreamAddCallback, cudaStreamDestroy
Key Concepts	CUDA Streams, Callback Functions, Multithreading
Supported OSes	Linux, Windows, OS X

simpleCubemapTexture - Simple Cubemap Texture

Simple example that demonstrates how to use a new CUDA 4.1 feature to support cubemap Textures in CUDA C.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaMalloc3DArray, cudaMemcpy3D, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Volume Processing
Supported OSes	Linux, Windows, OS X

simpleIPC

This CUDA Runtime API sample is a very basic sample that demonstrates Inter Process Communication with one process per GPU for computation. Requires Compute Capability 2.0 or higher and a Linux Operating System

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	IPC
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaIpcGetEventHandlet, cudaIpcOpenMemHandle, cudaIpcCloseMemHandle, cudaFreeHost, cudaMemcpy
Key Concepts	CUDA Systems Integration, Peer to Peer, InterProcess Communication
Supported OSes	Linux

simpleLayeredTexture - Simple Layered Texture

Simple example that demonstrates how to use a new CUDA 4.0 feature to support layered Textures in CUDA C.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaMalloc3DArray, cudaMemcpy3D, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Volume Processing
Supported OSes	Linux, Windows, OS X

simpleMPI

Simple example demonstrating how to use MPI in combination with CUDA.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	MPI

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMallco, cudaFree, cudaMemcpy
Key Concepts	CUDA Systems Integration, MPI, Multithreading
Supported OSes	Linux, Windows, OS X

simpleMultiCopy - Simple Multi Copy and Compute

Supported in GPUs with Compute Capability 1.1, overlapping compute with one memcopy is possible from the host system. For Quadro and Tesla GPUs with Compute Capability 2.0, a second overlapped copy operation in either direction at full speed is possible (PCI-e is symmetric). This sample illustrates the usage of CUDA streams to achieve overlapping of kernel execution with data copies to and from the device.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	CUDA Streams and Events, Asynchronous Data Transfers, Overlap Compute and Copy, GPU Performance
Supported OSes	Linux, Windows, OS X

simpleMultiGPU - Simple Multi-GPU

This application demonstrates how to use the new CUDA 4.0 API for CUDA context management and multi-threaded access to run CUDA kernels on multiple-GPUs.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events, Multithreading, Multi-GPU
Supported OSes	Linux, Windows, OS X

simpleP2P - Simple Peer-to-Peer Transfers with Multi-GPU

This application demonstrates CUDA APIs that support Peer-To-Peer (P2P) copies, Peer-To-Peer (P2P) addressing, and Unified Virtual Memory Addressing (UVA) between multiple GPUs. In general, P2P is supported between two same GPUs with some exceptions, such as some Tesla and Quadro GPUs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	only-64-bit
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaDeviceCanAccessPeer, cudaDeviceEnablePeerAccess, cudaDeviceDisablePeerAccess, cudaEventCreateWithFlags, cudaEventElapsedTime, cudaMemcpy
Key Concepts	Performance Strategies, Asynchronous Data Transfers, Unified Virtual Address Space, Peer to Peer Data Transfers, Multi-GPU
Supported OSes	Linux, Windows

simplePitchLinearTexture - Pitch Linear Texture

Use of Pitch Linear Textures

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMallocPitch, cudaMallocArray, cudaMemcpy2D, cudaMemcpyToArray, cudaBindTexture2D, cudaBindTextureToArray, cudaCreateChannelDesc, cudaMalloc, cudaFree, cudaFreeArray, cudaUnbindTexture, cudaMemset2D, cudaMemcpy2D
Key Concepts	Texture, Image Processing
Supported OSes	Linux, Windows, OS X

simplePrintf

This CUDA Runtime API sample is a very basic sample that implements how to use the printf function in the device code. Specifically, for devices with compute capability less than 2.0, the function cuPrintf is called; otherwise, printf can be used directly.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaPrintfDisplay, cudaPrintfEnd
Key Concepts	Debugging
Supported OSes	Linux, Windows, OS X

simpleSeparateCompilation - Simple Static GPU Device Library

This sample demonstrates a CUDA 5.0 feature, the ability to create a GPU device static library and use it within another CUDA kernel. This example demonstrates how to pass in a GPU device function (from the GPU device static library) as a function pointer to be called. This sample requires devices with compute capability 2.0 or higher.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Separate Compilation
Supported OSes	Linux, Windows, OS X

simpleStreams

This sample uses CUDA streams to overlap kernel executions with memory copies between the host and a GPU device. This sample uses a new CUDA 4.0 feature that supports pinning of generic host memory. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events
Supported OSes	Linux, Windows, OS X

simpleSurfaceWrite - Simple Surface Write

Simple example that demonstrates the use of 2D surface references (Write-to-Texture)

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaMallocArray, cudaBindSurfaceToArray, cudaBindTextureToArray, cudaCreateChannelDesc, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Surface Writes, Image Processing
Supported OSes	Linux, Windows, OS X

simpleTemplates - Simple Templates

This sample is a templatized version of the template project. It also shows how to correctly templatize dynamically allocated shared memory arrays.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	C++ Templates
Supported OSes	Linux, Windows, OS X

simpleTexture - Simple Texture

Simple example that demonstrates use of Textures in CUDA.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaMallocArray, cudaMemcpyToArray, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	CUDA Runtime API, Texture, Image Processing
Supported OSes	Linux, Windows, OS X

simpleTextureDrv - Simple Texture (Driver Version)

Simple example that demonstrates use of Textures in CUDA. This sample uses the new CUDA 4.0 kernel launch Driver API.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0

CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuLaunchKernel, cuCtxSynchronize, cuMemcpyDtoH, cuMemAlloc, cuMemFree, cuArrayCreate, cuArrayDestroy, cuCtxDetach, cuMemcpy2D, cuModuleGetTexRef, cuTexRefSetArray, cuTexRefSetAddressMode, cuTexRefSetFilterMode, cuTexRefSetFlags, cuTexRefSetFormat, cuParamSetTexRef
Key Concepts	CUDA Driver API, Texture, Image Processing
Supported OSes	Linux, Windows, OS X

simpleVoteIntrinsics - Simple Vote Intrinsics

Simple program which demonstrates how to use the Vote (any, all) intrinsic instruction in a CUDA kernel. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMallco, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Vote Intrinsics
Supported OSes	Linux, Windows, OS X

simpleZeroCopy

This sample illustrates how to use Zero MemCopy, kernels can read and write directly to pinned system memory.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaHostAlloc, cudaHostGetDevicePointer, cudaHostRegister, cudaHostUnregister, cudaFreeHost
Key Concepts	Performance Strategies, Pinned System Paged Memory, Vector Addition
Supported OSes	Linux, Windows, OS X
Whitepaper	CUDA2.2PinnedMemoryAPIs.pdf

template - Template

A trivial template project that can be used as a starting point to create new CUDA projects.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaMalloc, cudaFree, cudaDeviceSynchronize, cudaMemcpy
Key Concepts	Device Memory Allocation
Supported OSes	Linux, Windows, OS X

vectorAdd - Vector Addition

This CUDA Runtime API sample is a very basic sample that implements element by element vector addition. It is the same as the sample illustrating Chapter 3 of the programming guide with some additions like error checking.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Vector Addition
Supported OSes	Linux, Windows, OS X

vectorAddDrv - Vector Addition Driver API

This Vector Addition sample is a basic sample that is implemented element by element. It is the same as the sample illustrating Chapter 3 of the programming guide with some additions like error checking. This sample also uses the new CUDA 4.0 kernel launch Driver API.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, Vector Addition
Supported OSes	Linux, Windows, OS X

3.2. Utilities Reference

bandwidthTest - Bandwidth Test

This is a simple test program to measure the memcopy bandwidth of the GPU and memcpy bandwidth across PCI-e. This test application is capable of measuring device to device copy bandwidth, host to device copy bandwidth for pageable and page-locked memory, and device to host copy bandwidth for pageable and page-locked memory.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaSetDevice, cudaHostAlloc, cudaFree, cudaMallocHost, cudaFreeHost, cudaMemcpy, cudaMemcpyAsync, cudaEventCreate, cudaEventRecord, cudaEventDestroy, cudaDeviceSynchronize, cudaEventElapsedTime
Key Concepts	CUDA Streams and Events, Performance Strategies
Supported OSes	Linux, Windows, OS X

deviceQuery - Device Query

This sample enumerates the properties of the CUDA devices present in the system.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaSetDevice, cudaGetDeviceCount, cudaGetDeviceProperties, cudaDriverGetVersion, cudaRuntimeGetVersion
Key Concepts	CUDA Runtime API, Device Query
Supported OSes	Linux, Windows, OS X

deviceQueryDrv - Device Query Driver API

This sample enumerates the properties of the CUDA devices present using CUDA Driver API calls

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cuInit, cuDeviceGetCount, cuDeviceComputeCapability, cuDriverGetVersion, cuDeviceTotalMem, cuDeviceGetAttribute
Key Concepts	CUDA Driver API, Device Query

Supported OSes	Linux, Windows, OS X

3.3. Graphics Reference

bindlessTexture - Bindless Texture

This example demonstrates use of cudaSurfaceObject, cudaTextureObject, and MipMap support in CUDA. A GPU with Compute Capability SM 3.0 is required to run the sample.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Linux, Windows, OS X

Mandelbrot

This sample uses CUDA to compute and display the Mandelbrot or Julia sets interactively. It also illustrates the use of "double single" arithmetic to improve precision when zooming a long way into the pattern. This sample uses double precision. Thanks to Mark Granger of NewTek who submitted this code sample.!

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0

CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms
Supported OSes	Linux, Windows, OS X

marchingCubes - Marching Cubes Isosurfaces

This sample extracts a geometric isosurface from a volume dataset using the marching cubes algorithm. It uses the scan (prefix sum) function from the Thrust library to perform stream compaction.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	OpenGL Graphics Interop, Vertex Buffers, 3D Graphics, Physically Based Simulation
Supported OSes	Linux, Windows, OS X

simpleD3D10 - Simple Direct3D10 (Vertex Array)

Simple program which demonstrates interoperability between CUDA and Direct3D10. The program generates a vertex array with CUDA and uses Direct3D10 to render the geometry. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, 3D Graphics
Supported OSes	Windows

simpleD3D10RenderTarget - Simple Direct3D10 Render Target

Simple program which demonstrates interop of rendertargets between Direct3D10 and CUDA. The program uses RenderTarget positions with CUDA and generates a histogram with visualization. A Direct3D10 Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows

simpleD3D10Texture - Simple D3D10 Texture

Simple program which demonstrates how to interoperate CUDA with Direct3D10 Texture. The program creates a number of D3D10 Textures (2D, 3D, and CubeMap) which are generated from CUDA kernels. Direct3D then renders the results on the screen. A Direct3D10 Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample

will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows

simpleD3D11Texture - Simple D3D11 Texture

Simple program which demonstrates Direct3D11 Texture interoperability with CUDA. The program creates a number of D3D11 Textures (2D, 3D, and CubeMap) which are written to from CUDA kernels. Direct3D then renders the results on the screen. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D11GetDevice, cudaD3D11SetDirect3DDevice, cudaGraphicsD3D11RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Windows

simpleD3D9 - Simple Direct3D9 (Vertex Arrays)

Simple program which demonstrates interoperability between CUDA and Direct3D9. The program generates a vertex array with CUDA and uses Direct3D9 to render the geometry. A Direct3D capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D9GetDevice, cudaD3D9SetDirect3DDevice, cudaGraphicsD3D9RegisterResource, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop
Supported OSes	Windows

simpleD3D9Texture - Simple D3D9 Texture

Simple program which demonstrates Direct3D9 Texture interoperability with CUDA. The program creates a number of D3D9 Textures (2D, 3D, and CubeMap) which are written to from CUDA kernels. Direct3D then renders the results on the screen. A Direct3D capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D9GetDevice, cudaD3D9SetDirect3DDevice, cudaGraphicsD3D9RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaMemcpy3D, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows

simpleGL - Simple OpenGL

Simple program which demonstrates interoperability between CUDA and OpenGL. The program modifies vertex positions with CUDA and uses OpenGL to render the geometry.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Vertex Buffers, 3D Graphics
Supported OSes	Linux, Windows, OS X

simpleTexture3D - Simple Texture 3D

Simple example that demonstrates use of 3D Textures in CUDA.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures, Surface Writes
Supported OSes	Linux, Windows, OS X

SLID3D10Texture - SLI D3D10 Texture

Simple program which demonstrates SLI with Direct3D10 Texture interoperability with CUDA. The program creates a D3D10 Texture which is written to from a CUDA kernel. Direct3D then renders the results on the screen. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Performance Strategies, Graphics Interop, Image Processing, 2D Textures
Supported OSes	Windows

volumeFiltering - Volumetric Filtering with 3D Textures and Surface Writes

This sample demonstrates 3D Volumetric Filtering using 3D Textures and 3D Surface Writes.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures, Surface Writes
Supported OSes	Linux, Windows, OS X

volumeRender - Volume Rendering with 3D Textures

This sample demonstrates basic volume rendering using 3D Textures.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures
Supported OSes	Linux, Windows, OS X

3.4. Imaging Reference

bicubicTexture - Bicubic B-spline Interoplation

This sample demonstrates how to efficiently implement a Bicubic B-spline interpolation filter with CUDA texture.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows, OS X

bilateralFilter - Bilateral Filter

Bilateral filter is an edge-preserving non-linear smoothing filter that is implemented with CUDA with OpenGL rendering. It can be used in image recovery and denoising. Each pixel is weight by considering both the spatial distance and color distance between its neighbors. Reference:"C. Tomasi, R. Manduchi, Bilateral Filtering for Gray and Color Images, proceeding of the ICCV, 1998, http://users.soe.ucsc.edu/~manduchi/Papers/ICCV98.pdf"

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows, OS X

boxFilter - Box Filter

Fast image box filter using CUDA with OpenGL rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing

Supported OSes	Linux, Windows, OS X

convolutionFFT2D - FFT-Based 2D Convolution

This sample demonstrates how 2D convolutions with very large kernel sizes can be efficiently implemented using FFT transformations.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUFFT
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cufftPlan2d, cufftExecR2C, cufftExecC2R, cufftDestroy
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux, Windows, OS X

convolutionSeparable - CUDA Separable Convolution

This sample implements a separable convolution filter of a 2D signal with a gaussian kernel.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux, Windows, OS X
Whitepaper	convolutionSeparable.pdf

convolutionTexture - Texture-based Separable Convolution

Texture-based implementation of a separable 2D convolution with a gaussian kernel. Used for performance comparison against convolutionSeparable.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Texture, Data Parallel Algorithms

Supported OSes	Linux, Windows, OS X

cudaDecodeD3D9 - CUDA Video Decoder D3D9 API

This sample demonstrates how to efficiently use the CUDA Video Decoder API to decode MPEG-2, VC-1, or H.264 sources. YUV to RGB conversion of video is accomplished with CUDA kernel. The output result is rendered to a D3D9 surface. The decoded video is not displayed on the screen, but with -displayvideo at the command line parameter, the video output can be seen. Requires a Direct3D capable device and Compute Capability 2.0 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cuDeviceGet, cuDeviceGetAttribute, cuDeviceComputeCapability, cuDeviceGetCount, cuDeviceGetName, cuDeviceTotalMem, cuD3D9CtxCreate, cuD3D9GetDevice, cuModuleLoad, cuModuleUnload, cuModuleGetFunction, cuModuleGetGlobal, cuModuleLoadDataEx, cuModuleGetTexRef, cuD3D9MapResources, cuD3D9UnmapResources, cuD3D9RegisterResource, cuD3D9UnregisterResource, cuD3D9ResourceSetMapFlags, cuD3D9ResourceGetMappedPointer, cuD3D9ResourceGetMappedPitch, cuParamSetv, cuParamSeti, cuParamSetSize, cuLaunchGridAsync, cuCtxCreate, cuMemAlloc, cuMemFree, cuMemAllocHost, cuMemFreeHost, cuMemcpyDtoHAsync, cuMemsetD8, cuStreamCreate, cuCtxPushCurrent, cuCtxPopCurrent, cuvidCreateDecoder, cuvidDecodePicture, cuvidMapVideoFrame, cuvidUnmapVideoFrame, cuvidDestroyDecoder, cuvidCtxLockCreate, cuvidCtxLockDestroy, cuCtxDestroy
Key Concepts	Graphics Interop, Image Processing, Video Compression
Supported OSes	Windows
Whitepaper	CUDA_Video_Decoder.pdf

cudaDecodeGL - CUDA Video Decoder GL API

This sample demonstrates how to efficiently use the CUDA Video Decoder API to decode video sources based on MPEG-2, VC-1, and H.264. YUV to RGB conversion of video is accomplished with CUDA kernel. The output result is rendered to a OpenGL surface. The decoded video is black, but can be enabled with -displayvideo added to the command line. Requires Compute Capability 2.0 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, cuvid
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cuDeviceGet, cuDeviceGetAttribute, cuDeviceComputeCapability, cuDeviceGetCount, cuDeviceGetName, cuDeviceTotalMem, cuGLCtxCreate, cuGLGetDevice, cuModuleLoad, cuModuleUnload, cuModuleGetFunction, cuModuleGetGlobal, cuModuleLoadDataEx, cuModuleGetTexRef, cuGLMapResources, cuGLUnmapResources, cuGLRegisterResource, cuGLUnregisterResource, cuGLResourceSetMapFlags, cuGLResourceGetMappedPointer, cuGLResourceGetMappedPitch, cuParamSetv, cuParamSeti, cuParamSetSize, cuLaunchGridAsync, cuCtxCreate, cuMemAlloc, cuMemFree, cuMemAllocHost, cuMemFreeHost, cuMemcpyDtoHAsync, cuMemsetD8, cuStreamCreate, cuCtxPushCurrent, cuCtxPopCurrent, cuvidCreateDecoder, cuvidDecodePicture, cuvidMapVideoFrame, cuvidUnmapVideoFrame, cuvidDestroyDecoder, cuvidCtxLockCreate, cuvidCtxLockDestroy, cuCtxDestroy
Key Concepts	Graphics Interop, Image Processing, Video Compression
Supported OSes	Linux, Windows
Whitepaper	CUDA_Video_Decoder.pdf

dct8x8 - DCT8x8

This sample demonstrates how Discrete Cosine Transform (DCT) for blocks of 8 by 8 pixels can be performed using CUDA: a naive implementation by definition and a more traditional approach used in many libraries. As opposed to implementing DCT in a fragment shader, CUDA allows for an easier and more efficient implementation.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Video Compression
Supported OSes	Linux, Windows, OS X
Whitepaper	dct8x8.pdf

dwtHaar1D - 1D Discrete Haar Wavelet Decomposition

Discrete Haar wavelet decomposition for 1D signals with a length which is a power of 2.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Video Compression
Supported OSes	Linux, Windows, OS X

dxtc - DirectX Texture Compressor (DXTC)

High Quality DXT Compression using CUDA. This example shows how to implement an existing computationally-intensive CPU compression algorithm in parallel on the GPU, and obtain an order of magnitude performance improvement.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Cooperative Groups, Image Processing, Image Compression
Supported OSes	Linux, Windows, OS X
Whitepaper	cuda_dxtc.pdf

histogram - CUDA Histogram

This sample demonstrates efficient implementation of 64-bin and 256-bin histogram.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux, Windows, OS X
Whitepaper	histogram.pdf

HSOpticalFlow - Optical Flow

Variational optical flow estimation example. Uses textures for image operations. Shows how simple PDE solver can be accelerated with CUDA.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Data Parallel Algorithms

Supported OSes	Linux, Windows, OS X
Whitepaper	OpticalFlow.pdf

imageDenoising - Image denoising

This sample demonstrates two adaptive image denoising techniques: KNN and NLM, based on computation of both geometric and color distance between texels. While both techniques are implemented in the DirectX SDK using shaders, massively speeded up variation of the latter technique, taking advantage of shared memory, is implemented in addition to DirectX counterparts.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing
Supported OSes	Linux, Windows, OS X
Whitepaper	imageDenoising.pdf

postProcessGL - Post-Process in OpenGL

This sample shows how to post-process an image rendered in OpenGL using CUDA.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing

Supported OSes	Linux, Windows, OS X

recursiveGaussian - Recursive Gaussian Filter

This sample implements a Gaussian blur using Deriche's recursive method. The advantage of this method is that the execution time is independent of the filter width.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows, OS X

SobelFilter - Sobel Filter

This sample implements the Sobel edge detection filter for 8-bit monochrome images.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows, OS X

stereoDisparity - Stereo Disparity Computation (SAD SIMD Intrinsics)

A CUDA program that demonstrates how to compute a stereo disparity map using SIMD SAD (Sum of Absolute Difference) intrinsics. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Video Intrinsics
Supported OSes	Linux, Windows, OS X

3.5. Finance Reference

binomialOptions - Binomial Option Pricing

This sample evaluates fair call price for a given set of European options under binomial model.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Computational Finance
Supported OSes	Linux, Windows, OS X
Whitepaper	binomialOptions.pdf

BlackScholes - Black-Scholes Option Pricing

This sample evaluates fair call and put prices for a given set of European options by Black-Scholes formula.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Computational Finance
Supported OSes	Linux, Windows, OS X
Whitepaper	BlackScholes.pdf

MonteCarloMultiGPU - Monte Carlo Option Pricing with Multi-GPU support

This sample evaluates fair call price for a given set of European options using the Monte Carlo approach, taking advantage of all CUDA-capable GPUs installed in the system. This sample use double precision hardware if a GTX 200 class GPU is present. The sample also takes advantage of CUDA 4.0 capability to supporting using a single CPU thread to control multiple GPUs

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Supported OSes	Linux, Windows, OS X
Whitepaper	MonteCarlo.pdf

quasirandomGenerator - Niederreiter Quasirandom Sequence Generator

This sample implements Niederreiter Quasirandom Sequence Generator and Inverse Cumulative Normal Distribution functions for the generation of Standard Normal Distributions.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Computational Finance
Supported OSes	Linux, Windows, OS X

SobolQRNG - Sobol Quasirandom Number Generator

This sample implements Sobol Quasirandom Sequence Generator.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Computational Finance
Supported OSes	Linux, Windows, OS X

3.6. Simulations Reference

fluidsD3D9 - Fluids (Direct3D Version)

An example of fluid simulation using CUDA and CUFFT, with Direct3D 9 rendering. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaD3D9SetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Windows

fluidsGL - Fluids (OpenGL Version)

An example of fluid simulation using CUDA and CUFFT, with OpenGL rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, CUFFT
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Linux, Windows, OS X

Whitepaper	fluidsGL.pdf

nbody - CUDA N-Body Simulation

This sample demonstrates efficient all-pairs simulation of a gravitational n-body simulation in CUDA. This sample accompanies the GPU Gems 3 chapter "Fast N-Body Simulation with CUDA". With CUDA 5.5, performance on Tesla K20c has increased to over 1.8TFLOP/s single precision. Double Performance has also improved on all Kepler and Fermi GPU architectures as well. Starting in CUDA 4.0, the nBody sample has been updated to take advantage of new features to easily scale the n-body simulation across multiple GPUs in a single PC. Adding "-numbodies=<bodies>" to the command line will allow users to set # of bodies for simulation. Adding "-numdevices=<N>" to the command line option will cause the sample to use N devices (if available) for simulation. In this mode, the position and velocity data for all bodies are read from system memory using "zero copy" rather than from device memory. For a small number of devices (4 or fewer) and a large enough number of bodies, bandwidth is not a bottleneck so we can achieve strong scaling across these devices.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux, Windows, OS X
Whitepaper	nbody_gems3_ch31.pdf

oceanFFT - CUDA FFT Ocean Simulation

This sample simulates an Ocean height field using CUFFT Library and renders the result using OpenGL.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample

will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, CUFFT
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource, cufftPlan2d, cufftExecR2C, cufftExecC2R, cufftDestroy
Key Concepts	Graphics Interop, Image Processing, CUFFT Library
Supported OSes	Linux, Windows, OS X

particles - Particles

This sample uses CUDA to simulate and visualize a large set of particles and their physical interaction. Adding "-particles=<N>" to the command line will allow users to set # of particles for simulation. This example implements a uniform grid data structure using either atomic operations or a fast radix sort from the Thrust library

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation, Performance Strategies
Supported OSes	Linux, Windows, OS X
Whitepaper	particles.pdf

smokeParticles - Smoke Particles

Smoke simulation with volumetric shadows using half-angle slicing technique. Uses CUDA for procedural simulation, Thrust Library for sorting algorithms, and OpenGL for graphics rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux, Windows, OS X
Whitepaper	smokeParticles.pdf

VFlockingD3D10

The sample models formation of V-shaped flocks by big birds, such as geese and cranes. The algorithms of such flocking are borrowed from the paper "V-like formations in flocks of artificial birds" from Artificial Life, Vol. 14, No. 2, 2008. The sample has CPU- and GPU-based implementations. Press 'g' to toggle between them. The GPU-based simulation works many times faster than the CPU-based one. The printout in the console window reports the simulation time per step. Press 'r' to reset the initial distribution of birds.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0

CUDA API	cudaD3D10SetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation, Performance Strategies
Supported OSes	Windows

3.7. Advanced Reference

alignedTypes - Aligned Types

A simple test, showing huge access speed gap between aligned and misaligned structures.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows, OS X

cdpAdvancedQuicksort - Advanced Quicksort (CUDA Dynamic Parallelism)

This sample demonstrates an advanced quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Cooperative Groups, CUDA Dynamic Parallelism
Supported OSes	Linux, Windows, OS X

cdpBezierTessellation - Bezier Line Tessellation (CUDA Dynamic Parallelism)

This sample demonstrates bezier tessellation of lines implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows, OS X

cdpLUDecomposition - LU Decomposition (CUDA Dynamic Parallelism)

This sample demonstrates LU Decomposition implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP, CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows, OS X

cdpQuadtree - Quad Tree (CUDA Dynamic Parallelism)

This sample demonstrates Quad Trees implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Cooperative Groups, CUDA Dynamic Parallelism
Supported OSes	Linux, Windows, OS X

concurrentKernels - Concurrent Kernels

This sample demonstrates the use of CUDA streams for concurrent execution of several kernels on devices of compute capability 2.0 or higher. Devices of compute capability 1.x will run the kernels sequentially. It also illustrates how to introduce dependencies between CUDA streams with the new cudaStreamWaitEvent function introduced in CUDA 3.2

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows, OS X

eigenvalues - Eigenvalues

The computation of all or a subset of all eigenvalues is an important problem in Linear Algebra, statistics, physics, and many other fields. This sample demonstrates a parallel implementation of a bisection algorithm for the computation of all eigenvalues of a tridiagonal symmetric matrix of arbitrary size with CUDA.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Linear Algebra
Supported OSes	Linux, Windows, OS X
Whitepaper	eigenvalues.pdf

fastWalshTransform - Fast Walsh Transform

Naturally(Hadamard)-ordered Fast Walsh Transform for batching vectors of arbitrary eligible lengths that are power of two in size.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Linear Algebra, Data-Parallel Algorithms, Video Compression
Supported OSes	Linux, Windows, OS X

FDTD3d - CUDA C 3D FDTD

This sample applies a finite differences time domain progression stencil on a 3D surface.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows, OS X

FunctionPointers - Function Pointers

This sample illustrates how to use function pointers and implements the Sobel Edge Detection filter for 8-bit monochrome images.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows, OS X

interval - Interval Computing

Interval arithmetic operators example. Uses various C++ features (templates and recursion). The recursive mode requires Compute SM 2.0 capabilities.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Recursion, Templates
Supported OSes	Linux, Windows, OS X

lineOfSight - Line of Sight

This sample is an implementation of a simple line-of-sight algorithm: Given a height map and a ray originating at some observation point, it computes all the points along the ray that are visible from the observation point. The implementation is based on the Thrust library (http://code.google.com/p/thrust/).

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Supported OSes	Linux, Windows, OS X

mergeSort - Merge Sort

This sample implements a merge sort (also known as Batcher's sort), algorithms belonging to the class of sorting networks. While generally subefficient on large sequences compared to algorithms with better asymptotic algorithmic complexity (i.e. merge sort or radix sort), may be the algorithms of choice for sorting batches of short- to mid-sized (key, value) array pairs. Refer to the excellent tutorial by H. W. Lang http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/indexen.htm

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms
Supported OSes	Linux, Windows, OS X

newdelete - NewDelete

This sample demonstrates dynamic global memory allocation through device C++ new and delete operators and virtual function declarations available with CUDA 4.0.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Supported OSes	Linux, Windows, OS X

ptxjit - PTX Just-in-Time compilation

This sample uses the Driver API to just-in-time compile (JIT) a Kernel from PTX code. Additionally, this sample demonstrates the seamless interoperability capability of the CUDA Runtime and CUDA Driver API calls. For CUDA 5.5, this sample shows how to use cuLink* functions to link PTX assembly using the CUDA driver at runtime.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	CUDA Driver API
Supported OSes	Linux, Windows, OS X

radixSortThrust - CUDA Radix Sort (Thrust Library)

This sample demonstrates a very fast and efficient parallel radix sort uses Thrust library. The included RadixSort class can sort either key-value pairs (with float or unsigned integer keys) or keys only.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows, OS X
Whitepaper	readme.txt

reduction - CUDA Parallel Reduction

A parallel sum reduction that computes the sum of a large arrays of values. This sample demonstrates several important optimization strategies for 1:Data-Parallel Algorithms like reduction.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows, OS X

scalarProd - Scalar Product

This sample calculates scalar products of a given set of input vector pairs.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Linear Algebra
Supported OSes	Linux, Windows, OS X

scan - CUDA Parallel Prefix Sum (Scan)

This example demonstrates an efficient CUDA implementation of parallel prefix sum, also known as "scan". Given an array of numbers, scan computes a new array in which each element is the sum of all the elements before it in the input array.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows, OS X

segmentationTreeThrust - CUDA Segmentation Tree Thrust Library

This sample demonstrates an approach to the image segmentation trees construction. This method is based on Boruvka's MST algorithm.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows, OS X

shfl_scan - CUDA Parallel Prefix Sum with Shuffle Intrinsics (SHFL_Scan)

This example demonstrates how to use the shuffle intrinsic __shfl_up to perform a scan operation across a thread block. A GPU with Compute Capability SM 3.0. is required to run the sample

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows, OS X

simpleHyperQ

This sample demonstrates the use of CUDA streams for concurrent execution of several kernels on devices which provide HyperQ (SM 3.5). Devices without HyperQ (SM 2.0 and SM 3.0) will run a maximum of two kernels concurrently.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	CUDA Systems Integration, Performance Strategies
Supported OSes	Linux, Windows, OS X
Whitepaper	HyperQ.pdf

sortingNetworks - CUDA Sorting Networks

This sample implements bitonic sort and odd-even merge sort (also known as Batcher's sort), algorithms belonging to the class of sorting networks. While generally subefficient, for large sequences compared to algorithms with better asymptotic algorithmic complexity (i.e. merge sort or radix sort), this may be the preferred algorithms of choice for sorting batches of short-sized to mid-sized (key, value) array pairs. Refer to an excellent tutorial by H. W. Lang http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/indexen.htm

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Data-Parallel Algorithms
Supported OSes	Linux, Windows, OS X

threadFenceReduction

This sample shows how to perform a reduction operation on an array of values using the thread Fence intrinsic to produce a single value in a single kernel (as opposed to two or more kernel calls as shown in the "reduction" CUDA Sample). Single-pass reduction requires global atomic instructions (Compute Capability 2.0 or later) and the _threadfence() intrinsic (CUDA 2.2 or later).

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Cooperative Groups, Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows, OS X

threadMigration - CUDA Context Thread Management

Simple program illustrating how to the CUDA Context Management API and uses the new CUDA 4.0 parameter passing and CUDA launch API. CUDA contexts can be created separately and attached independently to different threads.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cuCtxCreate, cuCtxDestroy, cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuLaunchKernel, cuMemcpyDtoH, cuCtxPushCurrent, cuCtxPopCurrent
Key Concepts	CUDA Driver API
Supported OSes	Linux, Windows, OS X

transpose - Matrix Transpose

This sample demonstrates Matrix Transpose. Different performance are shown to achieve high performance.

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Performance Strategies, Linear Algebra
Supported OSes	Linux, Windows, OS X
Whitepaper	MatrixTranspose.pdf

3.8. Cudalibraries Reference

batchCUBLAS

A CUDA Sample that demonstrates how using batched CUBLAS API calls to improve overall performance.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS

Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Linear Algebra, CUBLAS Library
Supported OSes	Linux, Windows, OS X

boxFilterNPP - Box Filter with NPP

A NPP CUDA Sample that demonstrates how to use NPP FilterBox function to perform a Box Filter.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows, OS X

conjugateGradient - ConjugateGradient

This sample implements a conjugate gradient solver on GPU using CUBLAS and CUSPARSE library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux, Windows, OS X

conjugateGradientPrecond - Preconditioned Conjugate Gradient

This sample implements a preconditioned conjugate gradient solver on GPU using CUBLAS and CUSPARSE library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux, Windows, OS X

freeImageInteropNPP - FreeImage and NPP Interopability

A simple CUDA Sample demonstrate how to use FreeImage library with NPP.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows, OS X

histEqualizationNPP - Histogram Equalization with NPP

This CUDA Sample demonstrates how to use NPP for histogram equalization for image data.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample

will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, Performance Strategies, NPP Library
Supported OSes	Linux, Windows, OS X

MC_EstimatePiInlineP - Monte Carlo Estimation of Pi (inline PRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using inline PRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows, OS X

MC_EstimatePiInlineQ - Monte Carlo Estimation of Pi (inline QRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using inline QRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Random Number Generator, Computational Finance, CURAND Library

Supported OSes	Linux, Windows, OS X

MC_EstimatePiP - Monte Carlo Estimation of Pi (batch PRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using batch PRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows, OS X

MC_EstimatePiQ - Monte Carlo Estimation of Pi (batch QRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using batch QRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows, OS X

MC_SingleAsianOptionP - Monte Carlo Single Asian Option

This sample uses Monte Carlo to simulate Single Asian Options using the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows, OS X

MersenneTwisterGP11213

This sample demonstrates the Mersenne Twister random number generator GP11213 in cuRAND.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Computational Finance, CURAND Library
Supported OSes	Linux, Windows, OS X

randomFog - Random Fog

This sample illustrates pseudo- and quasi- random numbers produced by CURAND.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, CURAND
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	3D Graphics, CURAND Library
Supported OSes	Linux, Windows, OS X

simpleCUBLAS - Simple CUBLAS

Example of using CUBLAS using the new CUBLAS API interface available in CUDA 4.0.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, CUBLAS Library
Supported OSes	Linux, Windows, OS X

simpleCUFFT - Simple CUFFT

Example of using CUFFT. In this example, CUFFT is used to compute the 1D-convolution of some signal with some filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain. cuFFT plans are created using simple and advanced API functions.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUFFT
Supported SM Architecture	SM 3.0, SM 3.2, SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux, Windows, OS X

simpleDevLibCUBLAS - simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)

This sample implements a simple CUBLAS function calls that call GPU device API library running CUBLAS functions. This sample requires a SM 3.5 capable device.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP, CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0
CUDA API	cublasCreate, cublasSetVector, cublasSgemm, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Dynamic Parallelism, Linear Algebra
Supported OSes	Linux, Windows, OS X

Chapter 4. DEPENDENCIES

Some CUDA Samples rely on third-party applications and/or libraries, or features provided by the CUDA Toolkit and Driver, to either build or execute. These dependencies are listed below.

If a sample has a dependency that is not available on the system, the sample will not be installed. If a sample has a third-party dependency that is available on the system, but is not installed, the sample will waive itself at build time.

Each sample's dependencies are listed in the Samples Reference section.

Third-Party Dependencies

These third-party dependencies are required by some CUDA samples. If available, these dependencies are either installed on your system automatically, or are installable via your system's package manager (Linux) or a third-party website.

FreeImage

FreeImage is an open source imaging library. FreeImage can usually be installed on Linux using your distribution's package manager system. FreeImage can also be downloaded from the FreeImage website. FreeImage is also redistributed with the CUDA Samples.

Message Passing Interface

MPI (Message Passing Interface) is an API for communicating data between distributed processes. A MPI compiler can be installed using your Linux distribution's package manager system. It is also available on some online resources, such as Open MPI. On Windows, to build and run MPI-CUDA applications one can install MS-MPI SDK.

Only 64-Bit

Some samples can only be run on a 64-bit operating system.

DirectX

DirectX is a collection of APIs designed to allow development of multimedia applications on Microsoft platforms. For Microsoft platforms, NVIDIA's CUDA Driver supports DirectX. Several CUDA Samples for Windows demonstrates CUDA-DirectX Interoperability, for building such samples one needs to install Direct X SDK (June 2010 or newer) , this is required to be installed on Windows 7, Windows 10 and Windows Server 2008, Other Windows OSes do not need to explicitly install the DirectX SDK.

OpenGL

OpenGL is a graphics library used for 2D and 3D rendering. On systems which support OpenGL, NVIDIA's OpenGL implementation is provided with the CUDA Driver.

OpenGL ES

OpenGL ES is an embedded systems graphics library used for 2D and 3D rendering. On systems which support OpenGL ES, NVIDIA's OpenGL ES implementation is provided with the CUDA Driver.

OpenMP

OpenMP is an API for multiprocessing programming. OpenMP can be installed using your Linux distribution's package manager system. It usually comes preinstalled with GCC. It can also be found at the OpenMP website.

Screen

Screen is a windowing system found on the QNX operating system. Screen is usually found as part of the root filesystem.

X11

X11 is a windowing system commonly found on *-nix style operating systems. X11 can be installed using your Linux distribution's package manager, and comes preinstalled on Mac OS X systems.

EGL

EGL is an interface between Khronos rendering APIs (such as OpenGL, OpenGL ES or OpenVG) and the underlying native platform windowing system.

EGLOutput

EGLOutput is a set of EGL extensions which allow EGL to render directly to the display.

CUDA Features

These CUDA features are needed by some CUDA samples. They are provided by either the CUDA Toolkit or CUDA Driver. Some features may not be available on your system.

CUFFT Callback Routines

CUFFT Callback Routines are user-supplied kernel routines that CUFFT will call when loading or storing data. These callback routines are only available on Linux x86_64 and ppc64le systems.

CUDA Dynamic Paralellism

CDP (CUDA Dynamic Paralellism) allows kernels to be launched from threads running on the GPU. CDP is only available on GPUs with SM architecture of 3.5 or above.

Multi-block Cooperative Groups

Multi Block Cooperative Groups(MBCG) extends Cooperative Groups and the CUDA programming model to express inter-thread-block synchronization. MBCG is available on GPUs with Pascal and higher architecture on Linux systems.

CUBLAS

CUBLAS (CUDA Basic Linear Algebra Subroutines) is a GPU-accelerated version of the BLAS library.

CUDA Interprocess Communication

IPC (Interprocess Communication) allows processes to share device pointers. IPC is only available on Linux x86_64 and ppc64le systems.

CUFFT

CUFFT (CUDA Fast Fourier Transform) is a GPU-accelerated FFT library.

CURAND

CURAND (CUDA Random Number Generation) is a GPU-accelerated RNG library.

CUSPARSE

CUSPARSE (CUDA Sparse Matrix) provides linear algebra subroutines used for sparse matrix calculations.

CUSOLVER

CUSOLVER library is a high-level package based on the CUBLAS and CUSPARSE libraries. It combines three separate libraries under a single umbrella, each of which can be used independently or in concert with other toolkit libraries. The intent ofCUSOLVER is to provide useful LAPACK-like features, such as common matrix factorization and triangular solve routines for dense matrices, a sparse least-squares solver and an eigenvalue solver. In addition cuSolver provides a new refactorization library useful for solving sequences of matrices with a shared sparsity pattern.

NPP

NPP (NVIDIA Performance Primitives) provides GPU-accelerated image, video, and signal processing functions.

NVGRAPH

NVGRAPH is a GPU-accelerated graph analytics library..

NVRTC

NVRTC (CUDA RunTime Compilation) is a runtime compilation library for CUDA C++.

NVCUVID

NVCUVID (NVIDIA CUDA Video Decoder) provides GPU-accelerated video decoding capabilities.

Stream Priorities

Stream Priorities allows the creation of streams with specified priorities. Stream Priorities is only available on GPUs with SM architecture of 3.5 or above.

Unified Virtual Memory

UVM (Unified Virtual Memory) enables memory that can be accessed by both the CPU and GPU without explicit copying between the two. UVM is only available on Linux and Windows systems.

16-bit Floating Point

FP16 is a 16-bit floating-point format. One bit is used for the sign, five bits for the exponent, and ten bits for the mantissa. FP16 is only available on specific mobile platforms.

C++11 CUDA

NVCC Support of C++11 features.

Chapter 5. KEY CONCEPTS AND ASSOCIATED SAMPLES

The tables below describe the key concepts of the CUDA Toolkit and lists the samples that illustrate how that concept is used.

Basic Key Concepts

Basic Concepts demonstrates how to make use of CUDA features.

Table 2 Basic Key Concepts and Associated Samples

Basic Key Concept	Description	Samples
3D Graphics	*3D Rendering*	Random Fog, Simple Direct3D10 (Vertex Array), Simple OpenGL
3D Textures	*Volume Textures*	Simple Texture 3D
Assert	*GPU Assert*	simpleAssert
Asynchronous Data Transfers	*Overlapping I/O and Compute*	Simple Multi Copy and Compute, Simple Multi-GPU, Simple Peer-to-Peer Transfers with Multi-GPU, asyncAPI, simpleStreams
Atomic Intrinsics	*Using atomics with GPU kernels*	Simple Atomic Intrinsics
C++ Templates	*Using Templates with GPU kernels*	Simple Templates
CUBLAS	*CUDA BLAS samples*	Matrix Multiplication (CUBLAS)
CUBLAS Library	*CUDA BLAS samples*	Simple CUBLAS, batchCUBLAS

Basic Key Concept	Description	Samples
CUDA Driver API	*Samples that show the CUDA Driver API*	Device Query Driver API, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Using Inline PTX, Vector Addition Driver API
CUDA Dynamic Parallelism	*Dynamic Parallelism with GPU Kernels (SM 3.5)*	Simple Print (CUDA Dynamic Parallelism), simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)
CUDA Runtime API	*Samples that use the Runtime API*	Device Query, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Texture, Vector Addition
CUDA Streams	*Stream API definies a sequence of operations that can be overlapped with I/O*	Simple CUDA Callbacks
CUDA Streams and Events	*Synchronizing Kernels with Event Timers and Streams*	Bandwidth Test, Simple Multi Copy and Compute, Simple Multi-GPU, asyncAPI, simpleStreams
CUDA Systems Integration	*Samples that integrate with Multi Process (OpenMP, IPC, and MPI)*	cudaOpenMP, simpleIPC, simpleMPI
CUFFT Library	*Samples that use the CUDA FFT accelerated library*	Simple CUFFT
CURAND Library	*Samples that use the CUDA random number generator*	MersenneTwisterGP11213, Random Fog
Callback Functions	*Creating Callback functions with GPU kernels*	Simple CUDA Callbacks
Computationa Finance	*Finance Algorithms*	Black-Scholes Option Pricing, MersenneTwisterGP11213
Cooperative Groups	*Cooperative Groups is an extension to the CUDA programming model that allows the CUDA program to express the granularity at which different-sized groups of threads are communicating.*	Advanced Quicksort (CUDA Dynamic Parallelism), DirectX Texture Compressor (DXTC), Quad Tree (CUDA Dynamic Parallelism), threadFenceReduction

Basic Key Concept	Description	Samples
Data Parallel Algorithms	*Samples that show good usage of Data Parallel Algorithms*	CUDA Separable Convolution, Texture-based Separable Convolution
Debugging	*Samples useful for debugging*	simplePrintf
Device Memory Allocation	*Samples that show GPU Device side memory allocation*	Template
Device Query	*Sample showing simple device query of information*	Device Query, Device Query Driver API
GPU Performance	*Samples demonstrating high performance and data I/O*	Simple Multi Copy and Compute
Graphics Interop	*Samples that demonstrate interop between graphics APIs and CUDA*	Bicubic B-spline Interoplation, Bilateral Filter, Box Filter, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target, Simple Direct3D9 (Vertex Arrays), Simple OpenGL, Simple Texture 3D
Image Processing	*Samples that demonstrate image processing algorithms in CUDA*	Bicubic B-spline Interoplation, Bilateral Filter, Box Filter, Box Filter with NPP, CUDA Separable Convolution, FreeImage and NPP Interopability, Histogram Equalization with NPP, Pitch Linear Texture, Simple CUBLAS, Simple CUFFT, Simple D3D11 Texture, Simple Surface Write, Simple Texture, Simple Texture (Driver Version), Simple Texture 3D, Texture-based Separable Convolution
InterProcess Communicatio	*Samples that demonstrate Inter Process Communication between processes*	simpleIPC
Linear Algebra	*Samples demonstrating linear algebra with CUDA*	Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), batchCUBLAS, simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)

Basic Key Concept	Description	Samples
MPI	*Samples demonstrating how to use CUDA with MPI programs*	simpleMPI
Matrix Multiply	*Samples demonstrating matrix multiply CUDA*	Matrix Multiplication (CUDA Driver API Version)
Multi-GPU	*Samples demonstrating how to take advantage of multiple GPUs and CUDA*	Simple Multi-GPU, Simple Peer-to-Peer Transfers with Multi-GPU
Multithreading	*Samples demonstrating how to use multithreading with CUDA*	Simple CUDA Callbacks, Simple Multi-GPU, cudaOpenMP, simpleMPI
NPP Library	*Samples demonstrating how to use NPP (NVIDIA Performance Primitives) for image processing*	Box Filter with NPP, FreeImage and NPP Interopability, Histogram Equalization with NPP
OpenMP	*Samples demonstrating how to use OpenMP*	cudaOpenMP
Overlap Compute and Copy	*Samples demonstrating how to overlap Compute and Data I/O*	Simple Multi Copy and Compute
PTX Assembly	*Samples demonstrating how to use PTX code with CUDA*	Using Inline PTX
Peer to Peer	*Samples demonstrating how to handle P2P data transfers between multiple GPUs*	simpleIPC
Peer to Peer Data Transfers	*Samples demonstrating how to handle P2P data transfers between multiple GPUs*	Simple Peer-to-Peer Transfers with Multi-GPU
Performance Strategies	*Samples demonstrating high performance with CUDA*	Bandwidth Test, Box Filter with NPP, Clock, FreeImage and NPP Interopability, Histogram Equalization with NPP, Matrix Multiplication (CUBLAS), Simple Peer-to-Peer Transfers with Multi-GPU, Using Inline PTX, simpleZeroCopy
Pinned System Paged Memory	*Samples demonstrating how to properly handle data I/O efficiently between the CPU host and GPU video memory*	simpleZeroCopy

Basic Key Concept	Description	Samples
Separate Compilation	*Samples demonstrating how to use CUDA library linking*	Simple Static GPU Device Library
Surface Writes	*Samples demonstrating how to use Surface Writes with GPU kernels*	Simple Surface Write, Simple Texture 3D
Texture	*Samples demonstrating how to use textures GPU kernels*	Pitch Linear Texture, Simple Cubemap Texture, Simple D3D10 Texture, Simple D3D9 Texture, Simple Direct3D10 Render Target, Simple Layered Texture, Simple Surface Write, Simple Texture, Simple Texture (Driver Version), Texture-based Separable Convolution
Unified Virtual Address Space	*Samples demonstrating how to use UVA with CUDA programs*	Simple Peer-to-Peer Transfers with Multi-GPU
Vector Addition	*Samples demonstrating how to use Vector Addition with CUDA programs*	Vector Addition, Vector Addition Driver API, simpleZeroCopy
Vertex Buffers	*Samples demonstrating how to use Vertex Buffers with CUDA kernels*	Simple OpenGL
Volume Processing	*Samples demonstrating how to use 3D Textures for volume rendering*	Simple Cubemap Texture, Simple Layered Texture
Vote Intrinsics	*Samples demonstrating how to use vote intrinsics with CUDA*	Simple Vote Intrinsics

Advanced Key Concepts

Advanced Concepts demonstrate advanced techniques and algorithms implemented with CUDA.

Table 3 Advanced Key Concepts and Associated Samples

Advanced Key Concept	Description	Samples
2D Textures	*Texture Mapping*	SLI D3D10 Texture

Advanced Key Concept	Description	Samples
3D Graphics	*3D Rendering*	Marching Cubes Isosurfaces
3D Textures	*Volume Textures*	Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
CUBLAS Library	*CUDA BLAS samples*	ConjugateGradient, Preconditioned Conjugate Gradient
CUDA Driver API	*Samples that show the CUDA Driver API*	CUDA Context Thread Management, PTX Just-in-Time compilation
CUDA Dynamic Parallelism	*Dynamic Parallelism with GPU Kernels (SM 3.5)*	Advanced Quicksort (CUDA Dynamic Parallelism), Bezier Line Tessellation (CUDA Dynamic Parallelism), LU Decomposition (CUDA Dynamic Parallelism), Quad Tree (CUDA Dynamic Parallelism), Simple Quicksort (CUDA Dynamic Parallelism)
CUDA Systems Integration	*Samples that integrate with Multi Process (OpenMP, IPC, and MPI)*	simpleHyperQ
CUFFT Library	*Samples that use the CUDA FFT accelerated library*	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution, Fluids (Direct3D Version), Fluids (OpenGL Version)
CURAND Library	*Samples that use the CUDA random number generator*	Monte Carlo Estimation of Pi (batch PRNG), Monte Carlo Estimation of Pi (batch QRNG), Monte Carlo Estimation of Pi (inline PRNG), Monte Carlo Estimation of Pi (inline QRNG) , Monte Carlo Single Asian Option
CUSPARSE Library	*Samples that use the cuSPARSE (Sparse Vector Matrix Multiply) functions*	ConjugateGradient, Preconditioned Conjugate Gradient
Computational Finance	*Finance Algorithms*	Binomial Option Pricing, Monte Carlo Estimation of Pi (batch PRNG), Monte Carlo Estimation of Pi (batch QRNG), Monte Carlo Estimation of Pi (inline PRNG), Monte Carlo Estimation of Pi (inline QRNG) , Monte Carlo Single Asian Option, Niederreiter

Advanced Key Concept	Description	Samples
		Quasirandom Sequence Generator, Sobol Quasirandom Number Generator
Data Parallel Algorithms	*Samples that show good usage of Data Parallel Algorithms*	CUDA Histogram, CUDA N-Body Simulation, Mandelbrot, Optical Flow, Particles, Smoke Particles, VFlockingD3D10
Data-Parallel Algorithms	*Samples that show good usage of Data Parallel Algorithms*	CUDA Parallel Prefix Sum (Scan), CUDA Parallel Prefix Sum with Shuffle Intrinsics (SHFL_Scan), CUDA Parallel Reduction, CUDA Radix Sort (Thrust Library), CUDA Segmentation Tree Thrust Library, CUDA Sorting Networks, Fast Walsh Transform, Merge Sort, threadFenceReduction
Graphics Interop	*Samples that demonstrate interop between graphics APIs and CUDA*	Bindless Texture, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Fluids (Direct3D Version), Fluids (OpenGL Version), Function Pointers, Mandelbrot, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, SLI D3D10 Texture, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
Image Compression	*Samples that demonstrate image and video compression*	DirectX Texture Compressor (DXTC)
Image Processing	*Samples that demonstrate image processing algorithms in CUDA*	1D Discrete Haar Wavelet Decomposition, CUDA FFT Ocean Simulation, CUDA Histogram, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, DCT8x8, DirectX Texture Compressor (DXTC), FFT-Based 2D Convolution, Function Pointers, Image denoising, Optical Flow, Post-Process in OpenGL, Recursive Gaussian Filter, SLI D3D10 Texture, Sobel Filter, Stereo Disparity Computation (SAD SIMD Intrinsics), Volume Rendering with 3D

Advanced Key Concept	Description	Samples
		Textures, Volumetric Filtering with 3D Textures and Surface Writes
Linear Algebra	*Samples demonstrating linear algebra with CUDA*	ConjugateGradient, Eigenvalues, Fast Walsh Transform, Matrix Transpose, Preconditioned Conjugate Gradient, Scalar Product
OpenGL Graphics Interop	*Samples demonstrating how to use interoperability CUDA with OpenGL*	Marching Cubes Isosurfaces
Performance Strategies	*Samples demonstrating high performance with CUDA*	Aligned Types, CUDA C 3D FDTD, CUDA Parallel Prefix Sum (Scan), CUDA Parallel Prefix Sum with Shuffle Intrinsics (SHFL_Scan), CUDA Parallel Reduction, CUDA Radix Sort (Thrust Library), CUDA Segmentation Tree Thrust Library, Concurrent Kernels, Matrix Transpose, Particles, SLI D3D10 Texture, VFlockingD3D10, simpleHyperQ, threadFenceReduction
Physically Based Simulation	*Samples demonstrating high performance collisions and/or physocal interactions*	Marching Cubes Isosurfaces
Physically-Based Simulation	*Samples demonstrating high performance collisions and/or physocal interactions*	CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Particles, Smoke Particles, VFlockingD3D10
Random Number Generator	*Samples demonstrating how to use random number generation with CUDA*	Monte Carlo Estimation of Pi (batch PRNG), Monte Carlo Estimation of Pi (batch QRNG), Monte Carlo Estimation of Pi (inline PRNG), Monte Carlo Estimation of Pi (inline QRNG) , Monte Carlo Single Asian Option
Recursion	*Samples demonstrating recursion on CUDA*	Interval Computing
Surface Writes	*Samples demonstrating how to use Surface Writes with GPU kernels*	Volumetric Filtering with 3D Textures and Surface Writes

Advanced Key Concept	Description	Samples
Templates	*Samples demonstrating how to use templates GPU kernels*	Interval Computing
Texture	*Samples demonstrating how to use textures GPU kernels*	Bindless Texture
Vertex Buffers	*Samples demonstrating how to use Vertex Buffers with CUDA kernels*	Marching Cubes Isosurfaces
Video Compression	*Samples demonstrating how to use video compression with CUDA*	1D Discrete Haar Wavelet Decomposition, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, DCT8x8, Fast Walsh Transform
Video Intrinsics	*Samples demonstrating how to use video intrinsics with CUDA*	Stereo Disparity Computation (SAD SIMD Intrinsics)

Chapter 6. CUDA API AND ASSOCIATED SAMPLES

The tables below list the samples associated with each CUDA API.

CUDA Driver API Samples

The table below lists the samples associated with each CUDA Driver API.

Table 4 CUDA Driver API and Associated Samples

CUDA Driver API	Samples
cuArrayCreate	Simple Texture (Driver Version)
cuArrayDestroy	Simple Texture (Driver Version)
cuCtxCreate	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuCtxDestroy	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuCtxDetach	Simple Texture (Driver Version)
cuCtxPopCurrent	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuCtxPushCurrent	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuCtxSynchronize	Simple Texture (Driver Version)
cuD3D9CtxCreate	CUDA Video Decoder D3D9 API
cuD3D9GetDevice	CUDA Video Decoder D3D9 API
cuD3D9MapResources	CUDA Video Decoder D3D9 API

CUDA Driver API	Samples
cuD3D9RegisterResource	CUDA Video Decoder D3D9 API
cuD3D9ResourceGetMappedPitch	CUDA Video Decoder D3D9 API
cuD3D9ResourceGetMappedPointer	CUDA Video Decoder D3D9 API
cuD3D9ResourceSetMapFlags	CUDA Video Decoder D3D9 API
cuD3D9UnmapResources	CUDA Video Decoder D3D9 API
cuD3D9UnregisterResource	CUDA Video Decoder D3D9 API
cuDeviceComputeCapability	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Device Query Driver API
cuDeviceGet	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuDeviceGetAttribute	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Device Query Driver API
cuDeviceGetCount	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Device Query Driver API
cuDeviceGetName	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuDeviceTotalMem	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Device Query Driver API
cuDriverGetVersion	Device Query Driver API
cuGLCtxCreate	CUDA Video Decoder GL API
cuGLGetDevice	CUDA Video Decoder GL API
cuGLMapResources	CUDA Video Decoder GL API
cuGLRegisterResource	CUDA Video Decoder GL API
cuGLResourceGetMappedPitch	CUDA Video Decoder GL API
cuGLResourceGetMappedPointer	CUDA Video Decoder GL API
cuGLResourceSetMapFlags	CUDA Video Decoder GL API
cuGLUnmapResources	CUDA Video Decoder GL API
cuGLUnregisterResource	CUDA Video Decoder GL API
cuInit	Device Query Driver API

CUDA Driver API	Samples
cuLaunchGridAsync	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuLaunchKernel	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuMemAlloc	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuMemAllocHost	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuMemFree	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuMemFreeHost	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuMemcpy2D	Simple Texture (Driver Version)
cuMemcpyDtoH	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuMemcpyDtoHAsync	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuMemcpyHtoD	Matrix Multiplication (CUDA Driver API Version), Vector Addition Driver API
cuMemsetD8	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuModuleGetFunction	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuModuleGetGlobal	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuModuleGetTexRef	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Simple Texture (Driver Version)
cuModuleLoad	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Matrix Multiplication (CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuModuleLoadDataEx	CUDA Context Thread Management, CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API, Matrix Multiplication

CUDA Driver API	Samples
	(CUDA Driver API Version), Simple Texture (Driver Version), Vector Addition Driver API
cuModuleUnload	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuParamSetSize	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuParamSetTexRef	Simple Texture (Driver Version)
cuParamSeti	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuParamSetv	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuStreamCreate	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuTexRefSetAddressMode	Simple Texture (Driver Version)
cuTexRefSetArray	Simple Texture (Driver Version)
cuTexRefSetFilterMode	Simple Texture (Driver Version)
cuTexRefSetFlags	Simple Texture (Driver Version)
cuTexRefSetFormat	Simple Texture (Driver Version)
cuvidCreateDecoder	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuvidCtxLockCreate	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuvidCtxLockDestroy	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuvidDecodePicture	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuvidDestroyDecoder	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuvidMapVideoFrame	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuvidUnmapVideoFrame	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API

CUDA Runtime API Samples

The table below lists the samples associated with each CUDA Runtime API.

Table 5 CUDA Runtime API and Associated Samples

CUDA Runtime API	Samples
cublasCreate	Matrix Multiplication (CUBLAS), simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)
cublasSetVector	simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)
cublasSgemm	Matrix Multiplication (CUBLAS), simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)
cudaBindSurfaceToArray	Simple Surface Write
cudaBindTexture2D	Pitch Linear Texture
cudaBindTextureToArray	Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture
cudaCreateChannelDesc	Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture
cudaD3D10GetDevice	SLI D3D10 Texture, Simple D3D10 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaD3D10SetDirect3DDevice	SLI D3D10 Texture, Simple D3D10 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaD3D10SetGLDevice	VFlockingD3D10
cudaD3D11GetDevice	Simple D3D11 Texture
cudaD3D11SetDirect3DDevice	Simple D3D11 Texture
cudaD3D9GetDevice	Simple D3D9 Texture, Simple Direct3D9 (Vertex Arrays)
cudaD3D9SetDirect3DDevice	Simple D3D9 Texture, Simple Direct3D9 (Vertex Arrays)
cudaD3D9SetGLDevice	Fluids (Direct3D Version)
cudaDeviceCanAccessPeer	Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceDisablePeerAccess	Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceEnablePeerAccess	Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceSynchronize	Bandwidth Test, Template
cudaDriverGetVersion	Device Query

CUDA Runtime API	Samples
cudaEventCreate	Bandwidth Test, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventCreateWithFlags	Simple Peer-to-Peer Transfers with Multi-GPU
cudaEventDestroy	Bandwidth Test, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventElapsedTime	Bandwidth Test, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Simple Peer-to-Peer Transfers with Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventQuery	Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventRecord	Bandwidth Test, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventSynchronize	Matrix Multiplication (CUDA Runtime API Version), Vector Addition
cudaFree	Bandwidth Test, C++ Integration, Clock, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Pitch Linear Texture, Simple Atomic Intrinsics, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture, Simple Vote Intrinsics, Template, Using Inline PTX, Vector Addition, cudaOpenMP, simpleAssert, simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism), simpleMPI
cudaFreeArray	Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture
cudaFreeHost	Bandwidth Test, Simple Atomic Intrinsics, Simple Vote Intrinsics, Using Inline PTX, simpleAssert, simpleIPC, simpleZeroCopy

CUDA Runtime API	Samples
cudaGLSetGLDevice	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGetDeviceCount	Device Query
cudaGetDeviceProperties	Device Query
cudaGraphicsD3D10RegisterResource	SLI D3D10 Texture, Simple D3D10 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaGraphicsD3D11RegisterResource	Simple D3D11 Texture
cudaGraphicsD3D9RegisterResource	Simple D3D9 Texture, Simple Direct3D9 (Vertex Arrays)
cudaGraphicsGLRegisterBuffer	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsMapResources	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsRegisterResource	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter,

CUDA Runtime API	Samples
	VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsResourceGetMappedPointer	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsResourceSetMapFlags	SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaGraphicsSubResourceGetMappedArray	SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaGraphicsUnmapResources	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsUnregisterResource	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, Fluids (Direct3D Version), Fluids (OpenGL Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target, Simple Direct3D9 (Vertex Arrays), Simple OpenGL, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaHostAlloc	Bandwidth Test, simpleZeroCopy
cudaHostGetDevicePointer	simpleZeroCopy

CUDA Runtime API	Samples
cudaHostRegister	simpleZeroCopy
cudaHostUnregister	simpleZeroCopy
cudaIpcCloseMemHandle	simpleIPC
cudaIpcGetEventHandlet	simpleIPC
cudaIpcOpenMemHandle	simpleIPC
cudaMallco	Simple Vote Intrinsics, simpleMPI
cudaMalloc	C++ Integration, Clock, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Pitch Linear Texture, Simple Atomic Intrinsics, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture, Template, Using Inline PTX, Vector Addition, cudaOpenMP, simpleAssert, simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism)
cudaMalloc3DArray	Simple Cubemap Texture, Simple Layered Texture
cudaMallocArray	Pitch Linear Texture, Simple Surface Write, Simple Texture
cudaMallocHost	Bandwidth Test, Using Inline PTX, simpleAssert
cudaMallocPitch	Pitch Linear Texture
cudaMemcpy	Bandwidth Test, C++ Integration, Clock, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Atomic Intrinsics, Simple Cubemap Texture, Simple Layered Texture, Simple Peer-to-Peer Transfers with Multi-GPU, Simple Surface Write, Simple Texture, Simple Vote Intrinsics, Template, Using Inline PTX, Vector Addition, cudaOpenMP, simpleAssert, simpleDevLibCUBLAS GPU Device API Library Functions (CUDA Dynamic Parallelism), simpleIPC, simpleMPI
cudaMemcpy2D	Pitch Linear Texture
cudaMemcpy2DToArray	SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaMemcpy3D	Simple Cubemap Texture, Simple D3D9 Texture, Simple Layered Texture

CUDA Runtime API	Samples
cudaMemcpyAsync	Bandwidth Test, Simple CUDA Callbacks, Simple Multi Copy and Compute, Simple Multi-GPU, asyncAPI, simpleStreams
cudaMemcpyToArray	Pitch Linear Texture, Simple Texture
cudaMemset2D	Pitch Linear Texture
cudaPrintfDisplay	simplePrintf
cudaPrintfEnd	simplePrintf
cudaRuntimeGetVersion	Device Query
cudaSetDevice	Bandwidth Test, Device Query
cudaStreamAddCallback	Simple CUDA Callbacks
cudaStreamCreate	Simple CUDA Callbacks
cudaStreamDestroy	Simple CUDA Callbacks
cudaUnbindTexture	Pitch Linear Texture
cufftDestroy	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution
cufftExecC2R	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution
cufftExecR2C	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution
cufftPlan2d	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution

Chapter 7. FREQUENTLY ASKED QUESTIONS

Answers to frequently asked questions about CUDA can be found at http://developer.nvidia.com/cuda-faq and in the CUDA Toolkit Release Notes.

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2007-2017 NVIDIA Corporation. All rights reserved.