CUDA SAMPLES

TRM-06704-001_v6.5 | August 2014

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 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.

1.2. 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.

1.3. 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/cdpBezierTesselation** - new CUDA sample that demonstrates how to use NPP for JPEG compression on the GPU
- 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.

1.4. 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/cdpBezierTessellation** - demonstrates an advanced method of implenting Bezier Line Tessellation 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.

1.5. CUDA 4.2

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

1.6. 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

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

2.1. 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\v6.5\
```

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

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5\<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, such as the **simpleD3D9** sample, require that the Microsoft DirectX SDK (June 2010 or newer) be installed and that the VC++ directory paths are properly set up (**Tools** > **Options...**).

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:

- **x86_64=1** - target the x86_64 architecture

```
$ make x86_64=1
```

- **ARMv7=1** - target the ARMv7 architecture (see here for more details)

```
$ make ARMv7=1
```

- **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"
```

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:

- **x86_64=1** - target the x86_64 architecture

```
$ make x86_64=1
```

- **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 ..."
```

2.2. Cross Samples

When cross-compiling an ARM CUDA application, nvcc must be able to find any libraries used, or be told to ignore missing symbols. One of the following methods should be chosen when cross-compiling the CUDA Samples. Regardless of which option is chosen, **ARMv7=1** should always be used.

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 ARMv7=1 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. For samples which use GL, HEADER_SEARCH_PATH must be set. For example, to cross-compile a sample which uses GL, execute:

```
$ make ARMv7=1 \
       EXTRA_LDFLAGS="-rpath-link=/opt/target/libs -L/opt/target/libs" \
       GLPATH=/opt/target/libs \
       HEADER_SEARCH_PATH=/opt/target/include
```

Ignore Symbol Detection

If neither of the above options are available, the linker can be told to ignore unresolved symbols. The samples should be forced to build using SAMPLE_ENABLED, and any library inclusion (**-lfoo**) should be removed from the Makefiles. To perform such a build, execute:

```
$ make ARMv7=1 \
       EXTRA_LDFLAGS="--unresolved-symbols=ignore-in-object-files" \
       SAMPLE_ENABLED=1
```

2.3. Using CUDA Samples to Create Your Own CUDA Projects

2.3.1. Creating CUDA Projects for Windows

Creating a new CUDA Program using the CUDA Samples infrastructure is easy. We have provided a **template** and **template_runtime** 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\v6.5\<category>\template
   ```

   or

   ```
   C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5\<category>
   \template_runtime
   ```

   to a directory of your own:

   ```
   C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5\<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** or **template_runtime** 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\v6.5\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.

2.3.2. Creating CUDA Projects for Linux

The default installation folder **<SAMPLES_INSTALL_PATH>** is **NVIDIA_CUDA_6.5_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** or **template_runtime** project that you can copy and modify to suit your needs. Just follow these steps:

1. Copy the **template** or **template_runtime** project:

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

   or (using **template_runtime**):

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

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

```
mv template.cu myproject.cu
mv template_kernel.cu myproject_kernel.cu
mv template_gold.cpp myproject_gold.cpp
```

   or (using **template_runtime**):

```
mv main.cu  myproject.cu
```

3. Edit the **Makefile** and source files.
   Just search and replace all occurrences of **template** or **template_runtime** 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.

2.3.3. Creating CUDA Projects for Mac OS X

The default installation folder **<SAMPLES_INSTALL_PATH>** is: **/Developer/NVIDIA/CUDA-6.5/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_kernel.cu myproject_kernel.cu
mv template_gold.cpp myproject_gold.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.

Minimum Required GPU SM 1.1

CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	KEPLER SM 3.5
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	KEPLER SM 3.5
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Clock

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

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

cppOverload

This sample demonstrates how to use C++ function overloading on the GPU.

Minimum Required GPU	SM 2.0
CUDA API	cudaFuncSetCacheConfig, cudaFuncGetAttributes
Key Concepts	C++ Function Overloading, CUDA Streams and Events
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

cudaOpenMP

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

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Systems Integration, OpenMP, Multithreading
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Using Inline PTX

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

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaMemcpy
Key Concepts	Performance Strategies, PTX Assembly, CUDA Driver API
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Linear Algebra
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
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 (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, Matrix Multiply
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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 .

Minimum Required GPU	SM 2.0
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaMemcpy
Key Concepts	Assert
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Atomic Intrinsics

A simple demonstration of global memory atomic instructions. Requires Compute Capability 1.1 or higher.

Minimum Required GPU	SM 1.1
CUDA API	cudaMallco, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Atomic Intrinsics
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaStreamCreate, cudaMemcpyAsync, cudaStreamAddCallback, cudaStreamDestroy
Key Concepts	CUDA Streams, Callback Functions, Multithreading
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Cubemap Texture

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

Minimum Required GPU	SM 2.0
CUDA API	cudaMalloc, cudaMalloc3DArray, cudaMemcpy3D, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy

Key Concepts	Texture, Volume Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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

Minimum Required GPU	SM 2.0
CUDA API	cudaIpcGetEventHandlet, cudaIpcOpenMemHandle, cudaIpcCloseMemHandle, cudaFreeHost, cudaMemcpy
Key Concepts	CUDA Systems Integration, Peer to Peer, InterProcess Communication
Supported OSes	Linux (tar.gz)

Simple Layered Texture

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

Minimum Required GPU	SM 2.0
CUDA API	cudaMalloc, cudaMalloc3DArray, cudaMemcpy3D, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Volume Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

simpleMPI

Simple example demonstrating how to use MPI in combination with CUDA. This executable is not pre-built with the SDK installer.

Minimum Required GPU	SM 1.1
CUDA API	cudaMallco, cudaFree, cudaMemcpy
Key Concepts	CUDA Systems Integration, MPI, Multithreading
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
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 (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events, Multithreading, Multi-GPU
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

simpleOccupancy

This sample demonstrates the basic usage of the CUDA occupancy calculator and occupancy-based launch configurator APIs by launching a kernel with the launch configurator, and measures the utilization difference against a manually configured launch.

Minimum Required GPU	SM 1.1
Key Concepts	Occupancy Calculator
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Peer-to-Peer Transfers with Multi-GPU

This application demonstrates the new CUDA 4.0 APIs that support Peer-To-Peer (P2P) copies, Peer-To-Peer (P2P) addressing, and UVA (Unified Virtual Memory Addressing) between multiple Tesla GPUs.

Minimum Required GPU	SM 2.0
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 (tar.gz), Windows (zip), OS X (tar.gz)

Pitch Linear Texture

Use of Pitch Linear Textures

Minimum Required GPU	SM 1.1
CUDA API	cudaMallocPitch, cudaMallocArray, cudaMemcpy2D, cudaMemcpyToArray, cudaBindTexture2D, cudaBindTextureToArray, cudaCreateChannelDesc, cudaMalloc, cudaFree, cudaFreeArray, cudaUnbindTexture, cudaMemset2D, cudaMemcpy2D
Key Concepts	Texture, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaPrintfDisplay, cudaPrintfEnd
Key Concepts	Debugging
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 2.0
Key Concepts	Separate Compilation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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 1.1 or higher.

Minimum Required GPU	SM 1.1
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Surface Write

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

Minimum Required GPU	SM 2.0
CUDA API	cudaMalloc, cudaMallocArray, cudaBindSurfaceToArray, cudaBindTextureToArray, cudaCreateChannelDesc, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Surface Writes, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Templates

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

Minimum Required GPU	SM 1.1

Key Concepts	C++ Templates
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Texture

Simple example that demonstrates use of Textures in CUDA.

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaMallocArray, cudaMemcpyToArray, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	CUDA Runtime API, Texture, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
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 (tar.gz), Windows (zip), OS X (tar.gz)

Simple Vote Intrinsics

Simple program which demonstrates how to use the Vote (any, all) intrinsic instruction in a CUDA kernel. Requires Compute Capability 1.2 or higher.

Minimum Required GPU	SM 1.2
CUDA API	cudaMallco, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Vote Intrinsics
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

simpleZeroCopy

This sample illustrates how to use Zero MemCopy, kernels can read and write directly to pinned system memory. This sample requires GPUs that support this feature (MCP79 and GT200).

Minimum Required GPU	SM 1.2
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 (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	CUDA2.2PinnedMemoryAPIs.pdf

Template

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

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaFree, cudaDeviceSynchronize, cudaMemcpy
Key Concepts	Device Memory Allocation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Template using CUDA Runtime

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

Minimum Required GPU	SM 1.1
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaDeviceSynchronize, cudaMemcpy
Key Concepts	CUDA Data Transfers, Device Memory Allocation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Unified Memory Streams

This sample demonstrates the use of OpenMP and streams with Unified Memory on a single GPU.

Minimum Required GPU	SM 3.0
CUDA API	cudaMallocManaged, cudaStreamAttachManagedMem
Key Concepts	CUDA Systems Integration, OpenMP, CUBLAS, Multithreading, Unified Memory, CUDA Streams and Events
Supported OSes	Linux (tar.gz), Windows (zip)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Vector Addition
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, Vector Addition
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

3.2. Utilities Reference

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.

Minimum Required GPU	SM 1.1
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 (tar.gz), Windows (zip), OS X (tar.gz)

Device Query

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

Minimum Required GPU	SM 1.1
CUDA API	cudaSetDevice, cudaGetDeviceCount, cudaGetDeviceProperties, cudaDriverGetVersion, cudaRuntimeGetVersion
Key Concepts	CUDA Runtime API, Device Query
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Device Query Driver API

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

Minimum Required GPU	SM 1.1
CUDA API	cuInit, cuDeviceGetCount, cuDeviceComputeCapability, cuDriverGetVersion, cuDeviceTotalMem, cuDeviceGetAttribute
Key Concepts	CUDA Driver API, Device Query
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Peer-to-Peer Bandwidth Latency Test with Multi-GPUs

This application demonstrates the CUDA Peer-To-Peer (P2P) data transfers between pairs of GPUs and computes latency and bandwidth. Tests on GPU pairs using P2P and without P2P are tested.

Minimum Required GPU	SM 2.0

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 (tar.gz), Windows (zip), OS X (tar.gz)

3.3. Graphics Reference

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.

Minimum Required GPU	KEPLER SM 3.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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 use double precision hardware if a GT200 class GPU is present. Thanks to Mark Granger of NewTek who submitted this code sample.!

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	OpenGL Graphics Interop, Vertex Buffers, 3D Graphics, Physically Based Simulation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, 3D Graphics
Supported OSes	Windows (zip)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource

Key Concepts	Graphics Interop, Texture
Supported OSes	Windows (zip)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows (zip)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D11GetDevice, cudaD3D11SetDirect3DDevice, cudaGraphicsD3D11RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Windows (zip)

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.

Minimum Required GPU	SM 1.1

CUDA API	cudaD3D9GetDevice, cudaD3D9SetDirect3DDevice, cudaGraphicsD3D9RegisterResource, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop
Supported OSes	Windows (zip)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D9GetDevice, cudaD3D9SetDirect3DDevice, cudaGraphicsD3D9RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaMemcpy3D, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows (zip)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Vertex Buffers, 3D Graphics
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple Texture 3D

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

Minimum Required GPU	SM 1.1

CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures, Surface Writes
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Performance Strategies, Graphics Interop, Image Processing, 2D Textures
Supported OSes	Windows (zip)

Volumetric Filtering with 3D Textures and Surface Writes

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

Minimum Required GPU	SM 2.0
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures, Surface Writes
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Volume Rendering with 3D Textures

This sample demonstrates basic volume rendering using 3D Textures.

Minimum Required GPU	SM 1.1

CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

3.4. Imaging Reference

Bicubic B-spline Interoplation

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

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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"

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Box Filter

Fast image box filter using CUDA with OpenGL rendering.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

FFT-Based 2D Convolution

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

Minimum Required GPU	SM 1.1
CUDA API	cufftPlan2d, cufftExecR2C, cufftExecC2R, cufftDestroy
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

CUDA Separable Convolution

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	convolutionSeparable.pdf

Texture-based Separable Convolution

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Texture, Data Parallel Algorithms

Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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 1.1 or higher.

Minimum Required GPU	SM 1.1
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 (zip)
Whitepaper	nvcuvid.pdf

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 1.1 or higher.

Minimum Required GPU	SM 1.1
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 (tar.gz), Windows (zip)
Whitepaper	nvcuvid.pdf

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.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Video Compression
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	dct8x8.pdf

1D Discrete Haar Wavelet Decomposition

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Video Compression
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Image Compression
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	cuda_dxtc.pdf

CUDA Histogram

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	histogram.pdf

Optical Flow

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	OpticalFlow.pdf

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.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	imageDenoising.pdf

Post-Process in OpenGL

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

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

CUDA and OpenGL Interop of Images

This sample shows how to copy CUDA image back to OpenGL using the most efficient methods.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Sobel Filter

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

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 2.0
Key Concepts	Image Processing, Video Intrinsics
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

3.5. Finance Reference

Binomial Option Pricing

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

Minimum Required GPU	SM 2.0
Key Concepts	Computational Finance
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	binomialOptions.pdf

Black-Scholes Option Pricing

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

Minimum Required GPU	SM 1.1
Key Concepts	Computational Finance
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	BlackScholes.pdf

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

Minimum Required GPU	SM 1.1
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	MonteCarlo.pdf

Niederreiter Quasirandom Sequence Generator

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

Minimum Required GPU	SM 2.0
Key Concepts	Computational Finance
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Sobol Quasirandom Number Generator

This sample implements Sobol Quasirandom Sequence Generator.

Minimum Required GPU	SM 1.1
Key Concepts	Computational Finance
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

3.6. Simulations Reference

Fluids (Direct3D Version)

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

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D9SetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Windows (zip)

Fluids (OpenGL Version)

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

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	fluidsGL.pdf

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	nbody_gems3_ch31.pdf

CUDA FFT Ocean Simulation

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

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource, cufftPlan2d, cufftExecR2C, cufftExecC2R, cufftDestroy
Key Concepts	Graphics Interop, Image Processing, CUFFT Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource

Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	particles.pdf

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.

Minimum Required GPU	SM 1.1
CUDA API	cudaGLSetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
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.

Minimum Required GPU	SM 1.1
CUDA API	cudaD3D10SetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation, Performance Strategies
Supported OSes	Windows (zip)

3.7. Advanced Reference

Aligned Types

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

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	KEPLER SM 3.5
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Bezier Line Tesselation (CUDA Dynamic Parallelism)

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

Minimum Required GPU	KEPLER SM 3.5
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	KEPLER SM 3.5
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	KEPLER SM 3.5
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Linear Algebra
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	eigenvalues.pdf

Fast Walsh Transform

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

Minimum Required GPU	SM 1.1
Key Concepts	Linear Algebra, Data-Parallel Algorithms, Video Compression

Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

CUDA C 3D FDTD

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

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Function Pointers

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

Minimum Required GPU	SM 2.0
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Interval Computing

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

Minimum Required GPU	SM 1.3
Key Concepts	Recursion, Templates
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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/).

Minimum Required GPU	SM 1.1
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version)

This sample revisits matrix multiplication using the CUDA driver API. It demonstrates how to link to CUDA driver at runtime and how to use JIT (just-in-time) compilation from PTX code. 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.

Minimum Required GPU	SM 1.1
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, CUDA Dynamically Linked Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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

Minimum Required GPU	SM 1.1
Key Concepts	Data-Parallel Algorithms
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

NewDelete

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

Minimum Required GPU	SM 2.0
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 2.0
Key Concepts	CUDA Driver API
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

CUDA Radix Sort (Thrust Library)

This sample demonstrates a very fast and efficient parallel radix sort uses Thrust library (http://code.google.com/p/thrust/). The included RadixSort class can sort either key-value pairs (with float or unsigned integer keys) or keys only. The optimized code in this sample (and also in reduction and scan) uses a technique known as warp-synchronous programming, which relies on the fact that within a warp of threads running on a CUDA GPU, all threads execute instructions synchronously. The code uses this to avoid __syncthreads() when threads within a warp are sharing data via __shared__ memory. It is important to note that for this to work correctly without race conditions on all GPUs, the shared memory used in these warp-synchronous expressions must be declared volatile. If it is not declared volatile, then in the absence of __syncthreads(), the compiler is free to delay stores to __shared__ memory and keep the data in registers (an optimization technique), which will result in incorrect execution. So please heed the use of volatile in these samples and use it in the same way in any code you derive from them.

Minimum Required GPU	SM 1.1
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	readme.txt

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.

Minimum Required GPU	SM 1.1

Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Scalar Product

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

Minimum Required GPU	SM 1.1
Key Concepts	Linear Algebra
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.3
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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

Minimum Required GPU	KEPLER SM 3.0
Key Concepts	Data-Parallel Algorithms, Performance Strategies

Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.3
Key Concepts	CUDA Systems Integration, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	HyperQ.pdf

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

Minimum Required GPU	SM 1.1
Key Concepts	Data-Parallel Algorithms
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Stream Priorities

This sample demonstrates basic use of stream priorities.

Minimum Required GPU	SM 3.5
Key Concepts	CUDA Streams and Events
Supported OSes	Linux (tar.gz)

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 1.1 or later) and the _threadfence() intrinsic (CUDA 2.2 or later).

Minimum Required GPU	SM 1.1
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

CUDA Context Thread Management

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

Minimum Required GPU	SM 1.1
CUDA API	cuCtxCreate, cuCtxDestroy, cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuLaunchKernel, cuMemcpyDtoH, cuCtxPushCurrent, cuCtxPopCurrent
Key Concepts	CUDA Driver API
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Matrix Transpose

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

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies, Linear Algebra
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)
Whitepaper	MatrixTranspose.pdf

3.8. Cudalibraries Reference

batchCUBLAS

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

Minimum Required GPU	SM 1.1
Key Concepts	Linear Algebra, CUBLAS Library

Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Box Filter with NPP

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

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

ConjugateGradient

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

Minimum Required GPU	SM 1.1
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Preconditioned Conjugate Gradient

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

Minimum Required GPU	SM 1.1
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

ConjugateGradientUM

This sample implements a conjugate gradient solver on GPU using CUBLAS and CUSPARSE library, using Unified Memory

Minimum Required GPU	SM 3.0
Key Concepts	Unified Memory, Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux (tar.gz), Windows (zip)

CUDA Interception Library

This sample demonstrates how to build and use an intercept library with CUDA.

Minimum Required GPU	SM 2.0
Supported OSes	Linux (tar.gz)

FreeImage and NPP Interopability

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

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

GrabCut with NPP

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)

Minimum Required GPU	SM 1.1
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Histogram Equalization with NPP

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Performance Strategies, NPP Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Image Segmentation using Graphcuts with NPP

This sample that demonstrates how to perform image segmentation using the NPP GraphCut function.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, Performance Strategies, NPP Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

JPEG encode/decode and resize with NPP

This sample demonstrates a simple image processing pipline. First, a JPEG file is huffman decoded and inverse DCT transformed and dequantized. Then the different plances are resized. Finally, the resized image is quantized, forward DCT transformed and huffman encoded.

Minimum Required GPU	SM 2.0
CUDA API	nppGetGpuComputeCapability, nppiDCTInitAlloc, nppiDecodeHuffmanScanHost_JPEG_8u16s_P3R, nppiDCTQuantInv8x8LS_JPEG_16s8u_C1R_NEW, nppiResizeSqrPixel_8u_C1R, nppiEncodeHuffmanGetSize, nppiDCTFree
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Monte Carlo Single Asian Option

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

Minimum Required GPU	SM 1.1
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

MersenneTwisterGP11213

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

Minimum Required GPU	SM 1.1
Key Concepts	Computational Finance, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Random Fog

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

Minimum Required GPU	SM 1.1
Key Concepts	3D Graphics, CURAND Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple CUBLAS

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

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, CUBLAS Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

SimpleCUFFT_2d_MGPU

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 on Multiple GPU.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Simple CUFFT Callbacks

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. The difference between this example and the Simple CUFFT example is that the

multiplication step is done by the CUFFT kernel with a user-supplied CUFFT callback routine, rather than by a separate kernel call.

Minimum Required GPU	SM 2.0
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux (tar.gz)

Simple CUFFT_MGPU

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 on Multiple GPU.

Minimum Required GPU	SM 1.1
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

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.

Minimum Required GPU	KEPLER SM 3.5
CUDA API	cublasCreate, cublasSetVector, cublasSgemm, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Dynamic Parallelism, Linear Algebra
Supported OSes	Linux (tar.gz), Windows (zip), OS X (tar.gz)

Chapter 4.
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 1 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*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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++ Function Overloading	*Use C++ overloading with GPU kernels*	cppOverload
C++ Templates	*Using Templates with GPU kernels*	Simple Templates

Basic Key Concept	Description	Samples
CUBLAS	*CUDA BLAS samples*	Matrix Multiplication (CUBLAS), Unified Memory Streams
CUBLAS Library	*CUDA BLAS samples*	Simple CUBLAS, batchCUBLAS
CUDA Data Transfers	*CUDA Data I/O*	Template using CUDA Runtime
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, Unified Memory Streams, asyncAPI, cppOverload, simpleStreams
CUDA Systems Integration	*Samples that integrate with Multi Process (OpenMP, IPC, and MPI)*	Unified Memory Streams, cudaOpenMP, simpleIPC, simpleMPI
CUFFT Library	*Samples that use the CUDA FFT accelerated library*	Simple CUFFT, Simple CUFFT Callbacks, Simple CUFFT_MGPU, SimpleCUFFT_2d_MGPU
CURAND Library	*Samples that use the CUDA random number generator*	MersenneTwisterGP11213, Random Fog
Callback Functions	*Creating Callback functions with GPU kernels*	Simple CUDA Callbacks

Basic Key Concept	Description	Samples
Computational Finance	*Finance Algorithms*	Black-Scholes Option Pricing, MersenneTwisterGP11213
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, Template using CUDA Runtime
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, CUDA and OpenGL Interop of Images, 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, CUDA and OpenGL Interop of Images, FreeImage and NPP Interopability, GrabCut with NPP, Histogram Equalization with NPP, Image Segmentation using Graphcuts with NPP, Pitch Linear Texture, Simple CUBLAS, Simple CUFFT, Simple CUFFT Callbacks, Simple CUFFT_MGPU, Simple D3D11 Texture, Simple Surface Write, Simple Texture, Simple Texture (Driver Version), Simple Texture 3D, SimpleCUFFT_2d_MGPU, Texture-based Separable Convolution
InterProcess Communication	*Samples that demonstrate Inter Process Communication between processes*	simpleIPC

Basic Key Concept	Description	Samples
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)
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*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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, Unified Memory Streams, cudaOpenMP, simpleMPI
NPP Library	*Samples demonstrating how to use NPP (NVIDIA Performance Primitives) for image processing*	Box Filter with NPP, FreeImage and NPP Interopability, GrabCut with NPP, Histogram Equalization with NPP, Image Segmentation using Graphcuts with NPP
Occupancy Calculator		simpleOccupancy
OpenMP	*Samples demonstrating how to use OpenMP*	Unified Memory Streams, 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*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU

Basic Key Concept	Description	Samples
Performance Strategies	*Samples demonstrating high performance with CUDA*	Bandwidth Test, Box Filter with NPP, CUDA and OpenGL Interop of Images, Clock, FreeImage and NPP Interopability, GrabCut with NPP, Histogram Equalization with NPP, Image Segmentation using Graphcuts with NPP, Matrix Multiplication (CUBLAS), Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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
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 Memory	*Samples demonstrating how to use Unified Memory*	ConjugateGradientUM, Unified Memory Streams
Unified Virtual Address Space	*Samples demonstrating how to use UVA with CUDA programs*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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

Basic Key Concept	Description	Samples
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 2 Advanced Key Concepts and Associated Samples

Advanced Key Concept	Description	Samples
2D Textures	*Texture Mapping*	SLI D3D10 Texture
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, ConjugateGradientUM, Preconditioned Conjugate Gradient
CUDA Driver API	*Samples that show the CUDA Driver API*	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), PTX Just-in-Time compilation
CUDA Dynamic Parallelism	*Dynamic Parallelism with GPU Kernels (SM 3.5)*	Advanced Quicksort (CUDA Dynamic Parallelism), Bezier Line Tesselation (CUDA Dynamic Parallelism), LU Decomposition (CUDA Dynamic Parallelism), Quad Tree (CUDA Dynamic Parallelism), Simple Quicksort (CUDA Dynamic Parallelism)
CUDA Dynamically Linked Library	*Dynamic loading of the CUDA DLL using CUDA Driver API*	Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version)

Advanced Key Concept	Description	Samples
CUDA Streams and Events	*Synchronizing Kernels with Event Timers and Streams*	Stream Priorities
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, ConjugateGradientUM, 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 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

Advanced Key Concept	Description	Samples
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 Textures, Volumetric Filtering with 3D Textures and Surface Writes
Linear Algebra	*Samples demonstrating linear algebra with CUDA*	ConjugateGradient, ConjugateGradientUM, 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

Advanced Key Concept	Description	Samples
		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
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 5.
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 3 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Simple Texture (Driver Version), Vector Addition Driver API
cuModuleGetGlobal	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API

CUDA Driver API	Samples
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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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 Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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

CUDA Driver API	Samples
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 4 CUDA Runtime API and Associated Samples

CUDA Runtime 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
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

CUDA Runtime API	Samples
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
cuLaunchGridAsync	CUDA Video Decoder D3D9 API, CUDA Video Decoder GL API
cuLaunchKernel	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking

CUDA Runtime API	Samples
	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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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), Matrix Multiplication (CUDA Driver API version with Dynamic Linking 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 Version), Matrix Multiplication (CUDA Driver

CUDA Runtime API	Samples
	API version with Dynamic Linking 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)
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

CUDA Runtime API	Samples
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	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceDisablePeerAccess	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceEnablePeerAccess	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceSynchronize	Bandwidth Test, Template, Template using CUDA Runtime
cudaDriverGetVersion	Device Query
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	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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), Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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,

CUDA Runtime API	Samples
	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, Template using CUDA Runtime, 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, Template using CUDA Runtime, Using Inline PTX, simpleAssert, simpleIPC, simpleZeroCopy
cudaFuncGetAttributes	cppOverload
cudaFuncSetCacheConfig	cppOverload
cudaGLSetGLDevice	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA and OpenGL Interop of Images, 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

CUDA Runtime API	Samples
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, CUDA and OpenGL Interop of Images, 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, CUDA and OpenGL Interop of Images, 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, CUDA and OpenGL Interop of Images, 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
cudaGraphicsResourceGetMappedPointer	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA and OpenGL Interop of Images, 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

CUDA Runtime API	Samples
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, CUDA and OpenGL Interop of Images, 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, CUDA and OpenGL Interop of Images, 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
cudaHostRegister	simpleZeroCopy
cudaHostUnregister	simpleZeroCopy
cudaIpcCloseMemHandle	simpleIPC
cudaIpcGetEventHandlet	simpleIPC
cudaIpcOpenMemHandle	simpleIPC
cudaMallco	Simple Atomic Intrinsics, Simple Vote Intrinsics, simpleMPI

CUDA Runtime API	Samples
cudaMalloc	C++ Integration, Clock, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture, Template, Template using CUDA Runtime, 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, Template using CUDA Runtime, Using Inline PTX, simpleAssert
cudaMallocManaged	Unified Memory Streams
cudaMallocPitch	Pitch Linear Texture
cudaMemcpy	Bandwidth Test, C++ Integration, Clock, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, 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, Template using CUDA Runtime, 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
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

CUDA Runtime API	Samples
cudaPrintfEnd	simplePrintf
cudaRuntimeGetVersion	Device Query
cudaSetDevice	Bandwidth Test, Device Query
cudaStreamAddCallback	Simple CUDA Callbacks
cudaStreamAttachManagedMem	Unified Memory Streams
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
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
nppGetGpuComputeCapability	JPEG encode/decode and resize with NPP
nppiDCTFree	JPEG encode/decode and resize with NPP
nppiDCTInitAlloc	JPEG encode/decode and resize with NPP
nppiDCTQuantInv8x8LS_JPEG_16s8u_C1R_NEW	JPEG encode/decode and resize with NPP
nppiDecodeHuffmanScanHost_JPEG_8u16s_P3R	JPEG encode/decode and resize with NPP
nppiEncodeHuffmanGetSize	JPEG encode/decode and resize with NPP

CUDA Runtime API	Samples
nppiResizeSqrPixel_8u_C1R	JPEG encode/decode and resize with NPP

Chapter 6.
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.