

## Chapter 2

---

# Getting Started

We hope that Chapter 1 has gotten you excited to get started learning CUDA C. Since this book intends to teach you the language through a series of coding examples, you'll need a functioning development environment. Sure, you could stand on the sideline and watch, but we think you'll have more fun and stay interested longer if you jump in and get some practical experience hacking CUDA C code as soon as possible. In this vein, this chapter will walk you through some of the hardware and software components you'll need in order to get started. The good news is that you can obtain all of the software you'll need for free, leaving you more money for whatever tickles your fancy.

## 2.1 Chapter Objectives

Through the course of this chapter, you will accomplish the following:

- You will download all the software components required through this book.
- You will set up an environment in which you can build code written in CUDA C.

## 2.2 Development Environment

Before embarking on this journey, you will need to set up an environment in which you can develop using CUDA C. The prerequisites to developing code in CUDA C are as follows:

- A CUDA-enabled graphics processor
- An NVIDIA device driver
- A CUDA development toolkit
- A standard C compiler

To make this chapter as painless as possible, we'll walk through each of these prerequisites now.

### 2.2.1 CUDA-ENABLED GRAPHICS PROCESSORS

---

Fortunately, it should be easy to find yourself a graphics processor that has been built on the CUDA Architecture because every NVIDIA GPU since the 2006 release of the GeForce 8800 GTX has been CUDA-enabled. Since NVIDIA regularly releases new GPUs based on the CUDA Architecture, the following will undoubtedly be only a partial list of CUDA-enabled GPUs. Nevertheless, the GPUs are all CUDA-capable.

For a complete list, you should consult the NVIDIA website [www.nvidia.com/cuda](http://www.nvidia.com/cuda), although it is safe to assume that all recent GPUs (GPUs from 2007 on) with more than 256MB of graphics memory can be used to develop and run code written with CUDA C.

**Table 2.1** CUDA-enabled GPUs

GeForce GTX 480	GeForce 8300 mGPU	Quadro FX 5600
GeForce GTX 470	GeForce 8200 mGPU	Quadro FX 4800
GeForce GTX 295	GeForce 8100 mGPU	Quadro FX 4800 for Mac
GeForce GTX 285	Tesla S2090	Quadro FX 4700 X2
GeForce GTX 285 for Mac	Tesla M2090	Quadro FX 4600
GeForce GTX 280	Tesla S2070	Quadro FX 3800
GeForce GTX 275	Tesla M2070	Quadro FX 3700
GeForce GTX 260	Tesla C2070	Quadro FX 1800
GeForce GTS 250	Tesla S2050	Quadro FX 1700
GeForce GT 220	Tesla M2050	Quadro FX 580
GeForce G210	Tesla C2050	Quadro FX 570
GeForce GTS 150	Tesla S1070	Quadro FX 470
GeForce GT 130	Tesla C1060	Quadro FX 380
GeForce GT 120	Tesla S870	Quadro FX 370
GeForce G100	Tesla C870	Quadro FX 370 Low Profile
GeForce 9800 GX2	Tesla D870	Quadro CX
GeForce 9800 GTX+	<b>QUADRO MOBILE PRODUCTS</b>	Quadro NVS 450
GeForce 9800 GTX	Quadro FX 3700M	Quadro NVS 420
GeForce 9800 GT	Quadro FX 3600M	Quadro NVS 295
GeForce 9600 GSO	Quadro FX 2700M	Quadro NVS 290
GeForce 9600 GT	Quadro FX 1700M	Quadro Plex 2100 D4
GeForce 9500 GT	Quadro FX 1600M	Quadro Plex 2200 D2
GeForce 9400GT	Quadro FX 770M	Quadro Plex 2100 S4
GeForce 8800 Ultra	Quadro FX 570M	Quadro Plex 1000 Model IV
GeForce 8800 GTX	Quadro FX 370M	<b>GEFORCE MOBILE PRODUCTS</b>
GeForce 8800 GTS	Quadro FX 360M	GeForce GTX 280M
GeForce 8800 GT	Quadro NVS 320M	GeForce GTX 260M
GeForce 8800 GS	Quadro NVS 160M	GeForce GTS 260M
GeForce 8600 GTS	Quadro NVS 150M	GeForce GTS 250M
GeForce 8600 GT	Quadro NVS 140M	GeForce GTS 160M
GeForce 8500 GT	Quadro NVS 135M	GeForce GTS 150M
GeForce 8400 GS	Quadro NVS 130M	GeForce GT 240M
GeForce 9400 mGPU	Quadro FX 5800	GeForce GT 230M
GeForce 9300 mGPU		

*Continued*

**Table 2.1** CUDA-enabled GPUs (Continued)

GeForce GT 130M	GeForce 9700M GTS	GeForce 9200M GS
GeForce G210M	GeForce 9700M GT	GeForce 9100M G
GeForce G110M	GeForce 9650M GS	GeForce 8800M GTS
GeForce G105M	GeForce 9600M GT	GeForce 8700M GT
GeForce G102M	GeForce 9600M GS	GeForce 8600M GT
GeForce 9800M GTX	GeForce 9500M GS	GeForce 8600M GS
GeForce 9800M GT	GeForce 9500M G	GeForce 8400M GT
GeForce 9800M GTS	GeForce 9300M GS	GeForce 8400M GS
GeForce 9800M GS	GeForce 9300M G	


### 2.2.2 NVIDIA DEVICE DRIVER

NVIDIA provides system software that allows your programs to communicate with the CUDA-enabled hardware. If you have installed your NVIDIA GPU properly, you likely already have this software installed on your machine. It never hurts to ensure you have the most recent drivers, so we recommend that you visit [www.nvidia.com/cuda](http://www.nvidia.com/cuda) and click the *Download Drivers* link. Select the options that match the graphics card and operating system on which you plan to do development. After following the installation instructions for the platform of your choice, your system will be up-to-date with the latest NVIDIA system software.

### 2.2.3 CUDA DEVELOPMENT TOOLKIT

If you have a CUDA-enabled GPU and NVIDIA's device driver, you are ready to run compiled CUDA C code. This means that you can download CUDA-powered applications, and they will be able to successfully execute their code on your graphics processor. However, we assume that you want to do more than just run code because, otherwise, this book isn't really necessary. If you want to *develop* code for NVIDIA GPUs using CUDA C, you will need additional software. But as promised earlier, none of it will cost you a penny.

You will learn these details in the next chapter, but since your CUDA C applications are going to be computing on two different processors, you are consequently going to need two compilers. One compiler will compile code for your GPU, and one will compile code for your CPU. NVIDIA provides the compiler for your GPU code. As with the NVIDIA device driver, you can download the *CUDA Toolkit* at <http://developer.nvidia.com/object/gpucomputing.htm>. Click the CUDA Toolkit link to reach the download page shown in Figure 2.1.


**DEVELOPER  
ZONE**

Search Developer Zone

Last Updated: 03 / 30 / 2010

**Quick Links**

- Home
- News
- Developer Newsletter
- Newsletter Sign-Up
- CUDA Newsletter Sign-Up
- Drivers
- Registered Developer Login
- Become a Registered Developer
- Events Calendar

**Parallel Nsight™**

**Graphics**

- DirectX
- OpenGL
- 3D Vision™
- Documentation

**GPU Computing**

- Downloads
- CUDA
- DirectCompute
- OpenCL
- Free GPU Computing Seminars

**Tegra**

**NVIDIA® Application Acceleration Tools**

- ScenIX™
- Complex™
- OptiX™
- PhysX™
- Cg Toolkit

**Forums**

- Developer Forums
- GPU Computing Forums

NOTE: The NVIDIA Developer Forums and the GPU Computing Forums require separate logins. We will fix this in the near future when the two forums are merged. Thank you for your patience!

**Contact**

**Legal Information**

**Site Feedback**

**CUDA 3.0 Downloads**

Click here to view all CUDA Toolkit releases

**Download Quick Links** [Windows] [Linux] [MacOS]

**Release Highlights**

- Support for the new Fermi architecture, with:
  - Native 64-bit GPU support
  - Multiple Copy Engine support
  - ECC reporting
  - Concurrent Kernel Execution
  - Fermi HW debugging support in cuda-gdb
  - Fermi HW profiling support for CUDA C and OpenCL in Visual Profiler
- C++ Class Inheritance and Template Inheritance support for increased programmer productivity
- A new unified interoperability API for Direct3D and OpenGL, with support for:
  - OpenGL texture interop
  - Direct3D 11 interop support
- CUDA Driver / Runtime Buffer Interoperability, which allows applications using the CUDA Driver API to also use libraries implemented using the CUDA C Runtime such as CUFFT and CUBLAS.
- CUBLAS now supports all BLAS1, 2, and 3 routines including those for single and double precision complex numbers
- Up to 100x performance improvement while debugging applications with cuda-gdb
- cuda-gdb hardware debugging support for applications that use the CUDA Driver API
- cuda-gdb support for JIT-compiled kernels
- New CUDA Memory Checker reports misalignment and out of bounds errors, available as a stand-alone utility and debugging mode within cuda-gdb
- CUDA Toolkit libraries are now versioned, enabling applications to require a specific version, support multiple versions explicitly, etc.
- CUDA C/C++ kernels are now compiled to standard ELF format
- Support for device emulation mode has been packaged in a separate version of the CUDA C Runtime (CUDART), and is deprecated in this release. Now that more sophisticated hardware debugging tools are available and more are on the way, NVIDIA will be focusing on supporting these tools instead of the legacy device emulation functionality.
  - On Windows, use the new Parallel Nsight development environment for Visual Studio, with integrated GPU debugging and profiling tools (was code-named "Nexus"). Please see [www.nvidia.com/nsight](http://www.nvidia.com/nsight) for details.
  - On Linux, use cuda-gdb and cuda-memcheck, and check out the solutions from Allinea and TotalView that will be available soon.
- Support for all the OpenCL features in the latest R195 production driver package:
  - Double Precision
  - Graphics Interoperability with OpenCL, Direct3D9, Direct3D10, and Direct3D11 for high performance visualization
  - o Query for Compute Capability, so you can target optimizations for GPU architectures (cl\_nv\_device\_attribute\_query)
  - Ability to control compiler optimization settings via support for pragma unroll in OpenCL kernels and an extension that allows programmers to set compiler flags. (cl\_nv\_compiler\_options)
  - OpenCL Images support, for better/faster image filtering
  - 32-bit global and local atomics for fast, convenient data manipulation
  - Byte Addressable Stores, for faster video/image processing and compression algorithms
  - Support for the latest OpenCL spec revision 1.0.48 and latest official Khronos OpenCL headers as of 2010-02-17

For more information on general purpose computing features of the Fermi architecture, see: [www.nvidia.com/fermi](http://www.nvidia.com/fermi).

Please review the release notes for additional important information about this release.

Note: The developer driver packages below provide baseline support for the widest number of NVIDIA products in the smallest number of installers. More recent production driver packages for end users are available at [www.nvidia.com/drivers](http://www.nvidia.com/drivers).

[Windows] [Linux] [MacOS]

Figure 2.1 The CUDA download page

You will again be asked to select your platform from among 32- and 64-bit versions of Windows XP, Windows Vista, Windows 7, Linux, and Mac OS. From the available downloads, you need to download the CUDA Toolkit in order to build the code examples contained in this book. Additionally, you are encouraged, although not required, to download the GPU Computing SDK code samples, which contains dozens of helpful example programs. The GPU Computing SDK code samples will not be covered in this book, but they nicely complement the material we intend to cover, and as with learning any style of programming, the more examples, the better. You should also take note that although nearly all the code in this book will work on the Linux, Windows, and Mac OS platforms, we have targeted the applications toward Linux and Windows. If you are using Mac OS X, you will be living dangerously and using unsupported code examples.

## 2.2.4 STANDARD C COMPILER

---

As we mentioned, you will need a compiler for GPU code and a compiler for CPU code. If you downloaded and installed the CUDA Toolkit as suggested in the previous section, you have a compiler for GPU code. A compiler for CPU code is the only component that remains on our CUDA checklist, so let's address that issue so we can get to the interesting stuff.

### WINDOWS

On Microsoft Windows platforms, including Windows XP, Windows Vista, Windows Server 2008, and Windows 7, we recommend using the Microsoft Visual Studio C compiler. NVIDIA currently supports both the Visual Studio 2005 and Visual Studio 2008 families of products. As Microsoft releases new versions, NVIDIA will likely add support for newer editions of Visual Studio while dropping support for older versions. Many C and C++ developers already have Visual Studio 2005 or Visual Studio 2008 installed on their machine, so if this applies to you, you can safely skip this subsection.

If you do not have access to a supported version of Visual Studio and aren't ready to invest in a copy, Microsoft does provide free downloads of the Visual Studio 2008 Express edition on its website. Although typically unsuitable for commercial software development, the Visual Studio Express editions are an excellent way to get started developing CUDA C on Windows platforms without investing money in software licenses. So, head on over to [www.microsoft.com/visualstudio](http://www.microsoft.com/visualstudio) if you're in need of Visual Studio 2008!

## LINUX

Most Linux distributions typically ship with a version of the GNU C compiler (`gcc`) installed. As of CUDA 3.0, the following Linux distributions shipped with supported versions of `gcc` installed:

- Red Hat Enterprise Linux 4.8
- Red Hat Enterprise Linux 5.3
- OpenSUSE 11.1
- SUSE Linux Enterprise Desktop 11
- Ubuntu 9.04
- Fedora 10

If you're a die-hard Linux user, you're probably aware that many Linux software packages work on far more than just the "supported" platforms. The CUDA Toolkit is no exception, so even if your favorite distribution is not listed here, it may be worth trying it anyway. The distribution's kernel, `gcc`, and `glibc` versions will in a large part determine whether the distribution is compatible.

## MACINTOSH OS X

If you want to develop on Mac OS X, you will need to ensure that your machine has at least version 10.5.7 of Mac OS X. This includes version 10.6, Mac OS X "Snow Leopard." Furthermore, you will need to install `gcc` by downloading and installing Apple's Xcode. This software is provided free to Apple Developer Connection (ADC) members and can be downloaded from <http://developer.apple.com/tools/Xcode>. The code in this book was developed on Linux and Windows platforms but should work without modification on Mac OS X systems.

## 2.3 Chapter Review

If you have followed the steps in this chapter, you are ready to start developing code in CUDA C. Perhaps you have even played around with some of the NVIDIA GPU Computing SDK code samples you downloaded from NVIDIA's website. If so, we applaud your willingness to tinker! If not, don't worry. Everything you need is right here in this book. Either way, you're probably ready to start writing your first program in CUDA C, so let's get started.