Getting Started with 3D Development for the iPhone

This chapter gives an overview of what you'll need to get started using Blender, SIO2, and Xcode to create interactive 3D content for the iPhone and iPod Touch. It also gives you a heads-up on what you can expect to learn over the course of the rest of the book and tips on where to look for further information on related topics. There's a lot to cover and a few hoops to jump through before you get to the real action, so you'll get right to business by downloading the software you need and setting up your development environment.

- **Getting started**
- **Getting the software**
- **Setting up your SIO2 development environment**

Getting Started

Welcome to the world of interactive 3D graphics programming for the iPhone and iPod Touch using Blender and the SIO2 game engine! I think you'll find that working with these tools is a fun and challenging experience.

Throughout this book, I will assume that you are working on a Mac computer running OS X Leopard (10.5) or later. The official iPhone Software Development Kit (SDK) is designed to run exclusively on Mac. There are some projects underway to create emulators and development environments for doing iPhone development on other platforms, but they are not officially sanctioned by Apple, so if you opt to try to make use of these alternatives, you're on your own. There's no guarantee you're going to be able to install and run your apps on a device or make them available to other iPhone and iPod Touch users.

As mentioned in the introduction, there are a number of areas of background knowledge that will be enormously helpful to you as you work your way through this book. A lot of the necessary information is dealt with in the appendices, but some of it will be up to you to fill in. A good print or online reference for C and C++ syntax will come in handy if you aren't already familiar with these programming languages. As a reference for OpenGL functions, the official *OpenGL Programming Guide (7th Edition)* by Dave Shreiner (Addison Wesley Professional, 2009)—also known as the Red Book—is indispensable.

It is not strictly necessary to have an iPhone or an iPod Touch of your own in order to learn the content of this book. Most (but not all) of the functionality described in this book can be run on your desktop using the iPhone simulator included with the iPhone SDK. However, some functionality, such as the accelerometer, requires the use of an actual device, and if you plan to make your app available to others, it will be necessary to test its performance on an actual device.

Getting the Software

There are a number of software tools you will need to have installed on your computer before you can proceed with this book. Some of what you will need is free and open source, some of it is simply free of charge, and some of it you'll have to pay for (although it won't break the bank). The rest of the chapter will focus on getting what you need and making sure it's working.

The iPhone SDK

If you're thumbing through this book in the shelves of your local bookstore wondering whether it's right for you, the one thing you should know immediately is that this book (like any book on iPhone development) is for Mac users only. The iPhone SDK 3.0 and development tools are available from Apple for Mac OS X 10.5.7 or greater, and it is not possible to develop for the iPhone on any other platform. If you're running an earlier version of Leopard, you should update your system using the Software Update tool in System Preferences.

The other thing you should be aware of, particularly if you are coming from a background of working with open-source software (as many Blender users are), is that there's nothing open about the iPhone development environment. Apple maintains strict control over how you use the iPhone SDK and how you are able to distribute the products you create with it. This isn't necessarily just because the folks at Apple are control freaks. The era of widely programmable mobile phone handsets has just begun, and there are many open questions about which directions the fledgling industry will take. Apple has erred on the side of caution in terms of security, and the success of the iTunes App Store and many of its contributing developers suggests that Apple is doing something right from a commercial standpoint as well. It remains to be seen how other, more open business models will fare in the arena of programmable handsets.

You can download the iPhone SDK at `http://developer.apple.com/iphone/program/sdk/`. The SDK includes Xcode, Apple's flagship integrated development environment (IDE) that includes a powerful editor and code browser, compilers, and a variety of debugging and testing tools. It also includes the Interface Builder, a separate but tightly integrated development application that enables you to create interfaces in a WYSIWYG manner, and the iPhone simulator, which enables you to test your iPhone software on your own computer via a graphical simulation of the iPhone displayed on your screen, as shown in Figure 1.1.

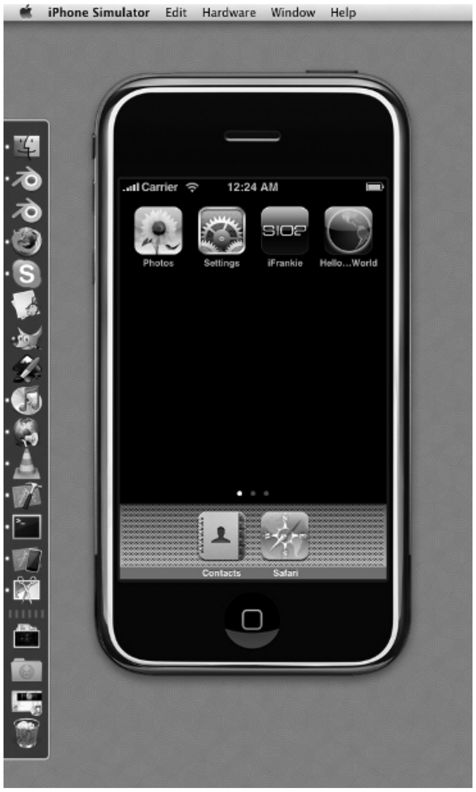


Figure 1.1
The iPhone simulator

To download all these tools, you will need to register with the Apple Developer Connection (ADC). Basic membership is free of charge, but there are restrictions on what you can do with this level of membership. The most serious restriction on the free membership is that the SDK can compile iPhone software *only* to the simulator. If you want to compile your software for use on an actual physical iPhone or iPod Touch device, you will need to join the iPhone Developer Program, which costs $99. This membership also grants you the right to submit your software for possible inclusion in the iTunes App Store. Membership in the iPhone Developer Program is tightly controlled. Although it is open to anybody to join, there is a period of verification before the certification is issued, and any discrepancies in your application can result in annoying delays while your information is further verified. (Don't mistype your billing address on this one!)

Throughout most of this book, I won't assume that you have iPhone Developer Program membership. The majority of examples in this book can be run on the iPhone simulator. A few features of the hardware are not present in the simulator, such as the accelerometer, which recognizes changes in the angle at which the device is being held. Any places in this book that deal with such functionality will be clearly indicated. If you're new to iPhone development, I recommend that you begin with the simulator. It's free to download, and you can get a good sense of what's involved in programming for the iPhone platform. Once you've decided to get serious, you can spring for the full iPhone Developer Program membership.

Installing the iPhone SDK will install Xcode, the iPhone simulator, Interface Builder, and some other tools on your computer. The installation should be straightforward and self-explanatory. There's a ton of documentation available on the Apple Developer Connection website, and you'll definitely want to delve into it.

Getting Blender

A great thing about mature, user-oriented free software like Blender is the relative ease with which you can download and install it. This book was written to correspond with Blender 2.49, which is the Blender version supported by SIO2 version 1.4.

You can download this version of Blender from the official Blender website. Since the latest Blender version may have changed by the time you read this, please download the software for OS X from the 2.49 archive page at `http://download.blender.org/release/Blender2.49a/`.

Blender should run straight "out of the box." Clicking the Blender application's icon should open a session. If you're new to Blender, now might be a good time to run through Appendix A on the basics of working with it. There are tons of tutorials online as well as a growing number of books available covering a variety of specific topics. Obviously, if you plan to create 3D content in Blender, you're going to want to become as skilled as possible in working with the software.

A Python installation is also required, but you shouldn't have to worry about this because Python 2.5 is installed by default in Leopard.

In OS X, Blender-Python output and errors are displayed in the Console. To read any errors or output from Python scripts, you can run Console (`Applications/Utilities/Console`) before starting up Blender.

Getting SIO2

The centerpiece of this book is the SIO2 engine. SIO2 is a set of software tools for exporting 3D assets from Blender and accessing them from within the Xcode development environment for inclusion in iPhone apps. This book was written to correspond to SIO2 version 1.4. The software is regularly updated and the released version changes regularly, but the version that corresponds to this book (as well as the tutorials used in this book) is available for download at the official SIO2 website: `http://sio2interactive.com/DOWNLOAD.html`.

As mentioned previously, SIO2 is available for free and its use is unrestricted except for one thing: If you use SIO2 to make a game or app available, you are asked to include the SIO2 splash screen at the start of the app. To bypass this restriction and use SIO2 without the splash screen, you are asked to purchase an inexpensive per-game Indie Certificate. The SIO2 Indie Certificate also gives you access to email technical support. SIO2 is not proprietary software, but purchasing the Indie Certificate is a big part of what keeps the project going, so I highly recommend doing so for any serious SIO2 projects. For now, though, simply download the ZIP file in the link and unzip it into a convenient location. I'll refer to this location from now on as your *SIO2_SDK directory*.

Setting Up Your Development Environment

Once you've downloaded the software and followed the steps for installing it, you can test your environment to make sure everything is working. In the following sections, you'll get your first look at the development environment that you'll become very familiar with over the course of the rest of the book. Building SIO2 projects in Xcode should be simple and straightforward, but if you're new to Xcode, there are a few things you might miss. If you hit any snags, skip forward to the troubleshooting section at the end of the chapter.

Building the SIO2 Template in Xcode

When you open your SIO2_SDK directory, you'll see a collection of directories. These include the code for the SIO2 engine, documentation of the API and `.sio2` file format, a collection of tutorials in the form of sample projects, supplementary model and texture data for the tutorial projects, and a template for creating new projects. For the purposes of this book, you'll make very heavy use of the template. In fact, the template project will be the starting point for everything you do with SIO2, so it is a good idea to keep a backup copy of the entire directory. Right now you're not going to make any changes to the template—you're only going to build an executable from it to make sure your development environment is properly set up—so it is not necessary to make a copy.

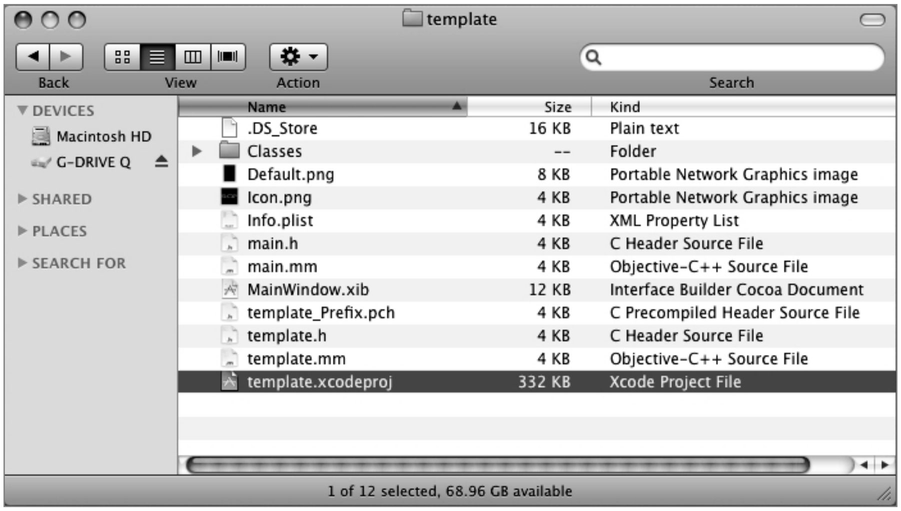
Open the `template` directory and take a look at what's inside. You should see the directory listing shown in Figure 1.2.

Everything that your iPhone app needs resides in this folder. Some of the suffixes are probably familiar to you, but others may not be. Now's not the time to worry about these though. Any code files you need will be dealt with in the Xcode environment. So the only file you really need to bother with here is `template.xcodeproj`. As you might have guessed from the suffix, this file is an Xcode project file. You'll be working a lot with files like these.

Double-click `template.xcodeproj` to open the project in Xcode. The first time you do this, you should see a window something like the one shown in Figure 1.3.

Figure 1.2

The Template project

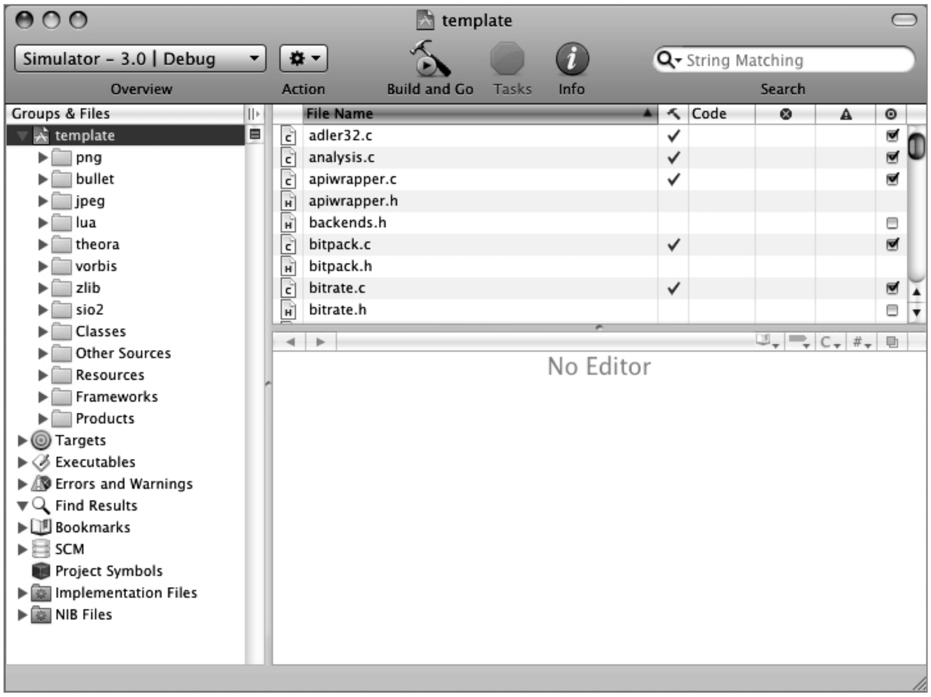


Name	Size	Kind
.DS_Store	16 KB	Plain text
Classes	--	Folder
Default.png	8 KB	Portable Network Graphics image
Icon.png	4 KB	Portable Network Graphics image
Info.plist	4 KB	XML Property List
main.h	4 KB	C Header Source File
main.mm	4 KB	Objective-C++ Source File
MainWindow.xib	12 KB	Interface Builder Cocoa Document
template_Prefix.pch	4 KB	C Precompiled Header Source File
template.h	4 KB	C Header Source File
template.mm	4 KB	Objective-C++ Source File
template.xcodeproj	332 KB	Xcode Project File

1 of 12 selected, 68.96 GB available

Figure 1.3

Opening the Template project in Xcode



File Name		Code			
adler32.c	✓				☑
analysis.c	✓				☑
apiwrapper.c	✓				☑
apiwrapper.h					
backends.h					☐
bitpack.c	✓				☑
bitpack.h					
bitrate.c	✓				☑
bitrate.h					☐

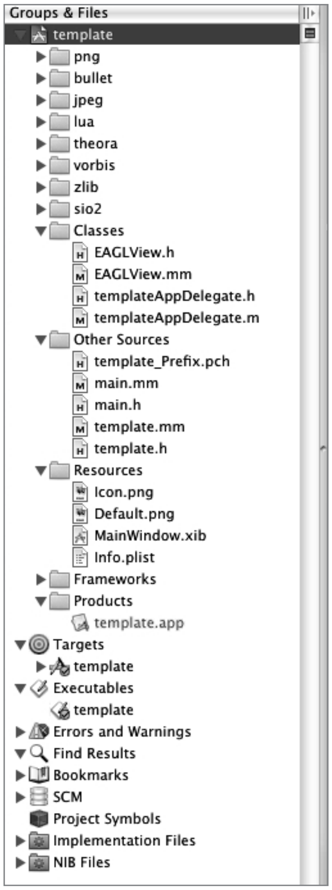
If all has gone smoothly so far, you should now be looking at the Template project in Xcode. This is the integrated development environment (IDE) that you will be working in for all of the coding parts of this book. The main area you see in the lower right (displaying the words *No Editor* in Figure 1.3) is where the code editor will open when a file is selected. Xcode's editor has a lot of powerful features that will help you code more quickly and accurately, and it's worth studying the online documentation available at the ADC website to get fully up to speed with what it has to offer. The window above the editor in the figure gives a listing of files in the project. This window is used for searching and navigating your project quickly.

The drop-down menu in the upper left of the Xcode toolbar is important. This enables you to select the destination for your compiled app. The menu shown in the image is set to Simulator-3.0 | Debug, meaning that the app will be compiled to the iPhone simulator using the 3.0 version of the iPhone OS and with debugging information. Building the application with this setting will automatically start up the iPhone simulator and install and run the app there. If this drop-down menu selection is changed to Device-3.0, Xcode will attempt to install the app on your iPhone or iPod Touch handset. This is possible only if you have registered with the iPhone Developer Program and followed the necessary steps to certify your device.

The tall horizontal pane along the left of the Xcode window is the Groups & Files pane. This gives you a complete overview of everything in your project. Any data or code that your application has access to is listed here. You can click the little triangles to the left of the directory icons to open the directories. Figure 1.4 shows some of the most important files that you'll need to know about as you work your way through this book. Take a close look at those now.

Figure 1.4

The Groups & Files pane



Groups & Files
- template
  - png
  - bullet
  - jpeg
  - lua
  - theora
  - vorbis
  - zlib
  - sio2
  - Classes
    - EAGLView.h
    - EAGLView.mm
    - templateAppDelegate.h
    - templateAppDelegate.m
  - Other Sources
    - template_Prefix.pch
    - main.mm
    - main.h
    - template.mm
    - template.h
  - Resources
    - Icon.png
    - Default.png
    - MainWindow.xib
    - Info.plist
  - Frameworks
  - Products
    - template.app
- Targets
  - template
- Executables
  - template
- Errors and Warnings
- Find Results
- Bookmarks
- SCM
- Project Symbols
- Implementation Files
- NIB Files

The `Classes` directory lists some standard classes that are typically implemented in iPhone apps. These classes will come up again later in the book, but for now you can regard them as boilerplate code that sets up the viewing environment for the app. You should note, however, that the classes come in pairs of files. Each pair of files includes a header file with a `.h` suffix and a code file with either a `.m` suffix or a `.mm` suffix. The `.m` and `.mm` suffixes indicate Objective-C and Objective-C++ code, respectively. Other source code suffixes you will be likely to see when working with SIO2 and its associated libraries include `.c` for plain C code, `.cpp` for C++ code, and `.cc` for code that can compile as both C and C++.

The `Other Sources` directory includes, as its fiendishly straightforward name suggests, other source code files. The `template_Prefix.pch` file is a precompiled header that you will not need to deal with directly. The `main.h` and `main.mm` files contain the code that makes the top-level function calls for the application. You should take a look at this code, but you will not work much with it directly in this book.

By far, the file that you will work with most in the course of this book will be `template.mm`. This contains most (not all) of the SIO2 API code that accesses the 3D assets and implements the interactive behavior of your app. By the end of this book, you will know this file and files like it inside and out.

The `Resources` directory contains non-code data files that the app needs access to. As you can see, there are two PNG image files currently in the `Resources` directory. One of them is the app icon image, and one of them is the loading screen image.

The `MainWindow.xib` file is created by the Interface Builder application. If you go on to do more iPhone development, you will certainly learn about the Interface Builder and XIB files, but you won't need to deal with them directly for the purposes of this book. All of the official SIO2 tutorials and all of the code in this book are built using the default OpenGL ES template from Xcode, and there is no direct support for building OpenGL ES interfaces with the Interface Builder.

The `Info.plist` file is a property list. This is a table of property values for the app. You can change various things here about your app, such as which image is used for the icon.

The `Resources` directory will also be home to the `.sio2` files created when you export 3D assets from Blender. You'll learn about `.sio2` files in Chapter 3. Note that file extensions are case-sensitive in Mac, and the `.sio2` file extension is always formatted in lowercase.

The `Products`, `Targets`, and `Executables` directories hold the elements of your app. Mostly, you will not need to deal with these directly, except to change their names when creating your own project. The Target system in Xcode enables multiple related applications to be created within a single project, which is useful in large-scale development projects such as client-server applications. For iPhone development, however, it is unlikely you will ever need to deal with more than one target per project, so this functionality can be mostly ignored.

All those directories that I haven't mentioned in the upper half of the Groups & Files pane are important too, but you will mostly not need to access them directly. These are the libraries that provide the functions called in the application. The `sio2` directory contains the actual SIO2 code that implements the functionality you'll be using. The `bullet` directory contains the Bullet Physics Library. To learn about the implementation of these libraries, you can browse these directories.

Now that you've got some idea of what's in your project, you can go ahead and advance to the most anticlimactic part of this whole chapter: building the template app. Do this by clicking the Build And Go button at the top of the Xcode window. Wait a few seconds as Xcode builds the app and installs it on your iPhone simulator. The iPhone simulator should open automatically, the SIO2 loading screen should flash for a split second, and then, if everything has gone smoothly. . .nothing! The screen of your iPhone simulator should go completely black.

Of course, the reason nothing happened is that this is, after all, a template. The whole point of this book will be to teach you how to turn this nothing into something interesting. To stop the current app, click the round button at the base of the iPhone simulator, just as you normally would on your iPhone or iPod Touch to stop an app. You'll return to the buttons screen of the iPhone simulator, as shown in Figure 1.5, where you'll see, sure enough, the button for the template app alongside the other apps installed on the simulator.

If this has all gone as described, then you should be pleased. Your development environment is set up and SIO2 is building smoothly. You're ready to move on to creating actual 3D content for your iPhone or iPod Touch in Chapter 2.

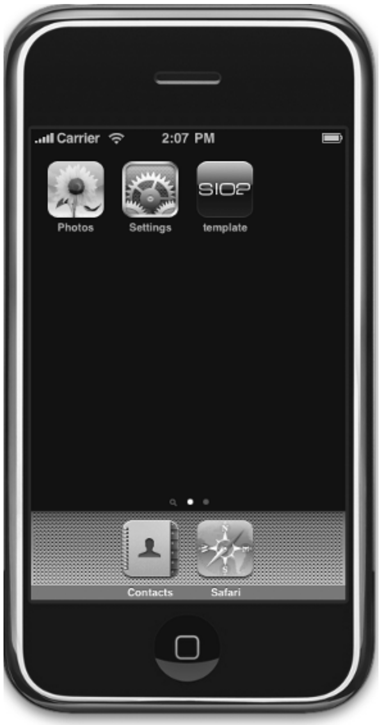


Figure 1.5
The template app installed in your iPhone simulator

Nevertheless, I wouldn't blame you if you felt a little bit gypped after going through a whole chapter without getting to see any actual 3D action on your iPhone simulator. Fortunately, the `SIO2_SDK` directory is packed with ready-made code samples that you can dive into and explore right now. I highly recommend that you take a look at some of them. You can build and run them all in exactly the same way that you did the template, by opening the file with the filename extension `.xcodeproj` in the project's directory and clicking Build And Go. Figure 1.6 shows the results of the `tutorial02` project, featuring every Blender artist's favorite digital monkey. Have fun exploring the other tutorial files. Don't worry if they seem over your head. After you have made your way through this book, you will have the background you need to dive in and pick them apart. Chapter 9 gives an overview of their contents, so you can go straight to the tutorial that has the advanced information you need.

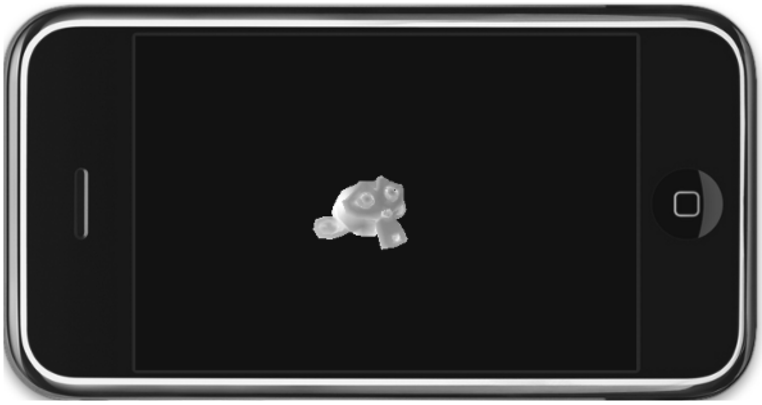


Figure 1.6
Suzanne in your iPhone

Troubleshooting

If Xcode does not open when you double-click a file with the filename extension `.xcodeproj`, it means there is a problem with your Xcode installation. You will need to go back to the instructions at the Apple Developer Connection website and make sure you correctly downloaded and installed the iPhone SDK.

The first time you build an app, it is important to have the build destination and the SDK version set correctly. Make sure the drop-down menu in the upper-left corner of the Xcode window is set to the version of the SDK you are using. Furthermore, make sure the application itself is set to compile using the correct version of the SDK. You can check this in the project settings under Project → Edit Project Settings. If you have trouble, refer to the iPhone Reference Library at `http://developer.apple.com/iphone/library/navigation/index.html` and search for "running applications" for more details about how to set the Project Build settings correctly.

If you are enrolled in the iPhone Developer Program and are compiling to a device, be sure you have read all the relevant documentation on certifying and making your device available to Xcode. If you have done this and have trouble compiling some of the tutorials, it may be due to discrepancies in code signing. If the apps are code-signed to another developer, you will need to change the code-signing value to build them yourself. Code-signing information for a project is found in the Project Settings properties list, and code-signing information for the target application is found in the Active Target properties list, which can be accessed under Project → Edit Active Target *`target_name`* (where *`target_name`* is the name of your target). Code-signing information must be set correctly for both the project and the target. You can find more information about this on the same "Running Applications" iPhone Reference Library page mentioned previously.

You will get a lot of use out of the iPhone Reference Library if you continue with iPhone and iPod Touch programming, so it's a good idea to bookmark it. If you've read this chapter and the pertinent iPhone SDK documents and iPhone Reference Library resources and you're still having problems, go to the SIO2 forum at `http://forum.sio2interactive.com` and run a search on your problem.

In the next chapter, you'll learn more about the fundamentals of OpenGL ES graphics programming in the iPhone.