1

Getting Started

WHAT'S IN THIS CHAPTER?

- Learning about the software used in this book
- Downloading the book's SDK
- Understanding the SDK architecture
- Importing projects into your IDE
- Understanding this book's template application
- Learning how to work with the template code structure

In this chapter, you will first start by setting up your development environment to be able to work with this book's tutorials and examples.

You will then receive a quick introduction about this book's SDK and where to download it, and learn about the different directories it contains. Then you will learn how to import this book's existing SDK projects and templates into your favorite IDE, as you will do throughout this book when following the different tutorials.

Moving on to the last section of this chapter, you will learn about this book's cross-platform template project. And finally, this chapter concludes with a quick tutorial that will help you to get familiar with the events of the template, as well as with the tone that will be used for all the tutorials in this book.

SOFTWARE REQUIREMENTS

This book's content is built to run on iOS 5.x+ as well as for Android 2.x+, the latest and most stable versions of these two mobile operating systems at the time this book was written.

For iOS Developers

To use this book for iOS, all you have to do is to grab a copy of the latest iOS SDK available at `http://developer.apple.com`, and install it on your Mac.

Out-of-the-box the iOS SDK provides a simulator with full GLES v2 support, so even if you do not have an iOS device, or do not have an official iOS Developer Certification from Apple, you can still make full use of this book.

For Android Developers

To set up your environment for Android, it is unfortunately not as easy as for iOS. First go to `http://developer.android.com/sdk/installing.html` and follow the instructions to install the Android SDK, Eclipse, and the ADT plug-in. Please note that the Android SDK version used for this book was v2.3.4, but later versions should also work as well.

All the code in this book uses C/C++, which means that you will have to install Android Native Code support. To finalize the installation of your development environment, follow these steps:

1. Grab a copy of the Android NDK at the following address: `http://developer.android.com/sdk/ndk/index.html`. The version used at the time of writing this book was r5c, but all examples and tutorials should work on later versions as well. Download the Android NDK zip package and decompress it on your machine where you have read and write access.
2. In order to compile and debug native code using Eclipse, you will need to install the Sequoyah plug-in. To do this, first enable the repository that is located (from the Eclipse main menu) in: Help ➪ Install New Software ➪ Available Software Sites ➪ Sequoyah Metadata Repository. Then select the entry from the Work With combo box, and once the repository data is loaded, select and install the Sequoyah Android Native Code Support, as shown in Figure 1-1.
3. Once Sequoyah is installed, go to (from the main menu): Eclipse Preferences ➪ Android ➪ Native Development and specify the location where you extracted the Android NDK in step 1, as shown in Figure 1-2.

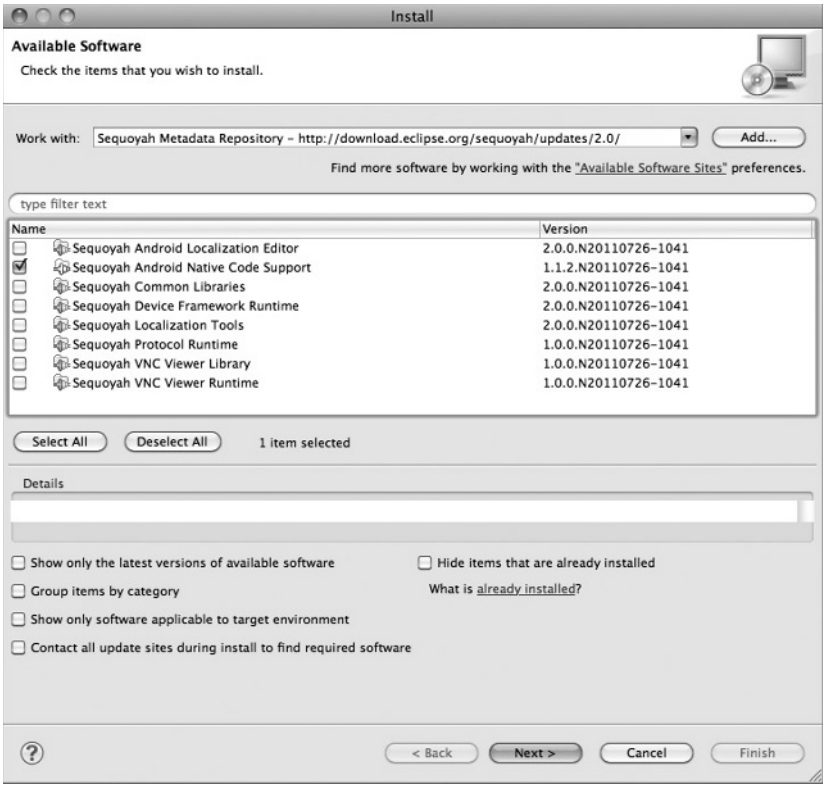

FIGURE 1-1: Sequoyah Native Code Support plug-in

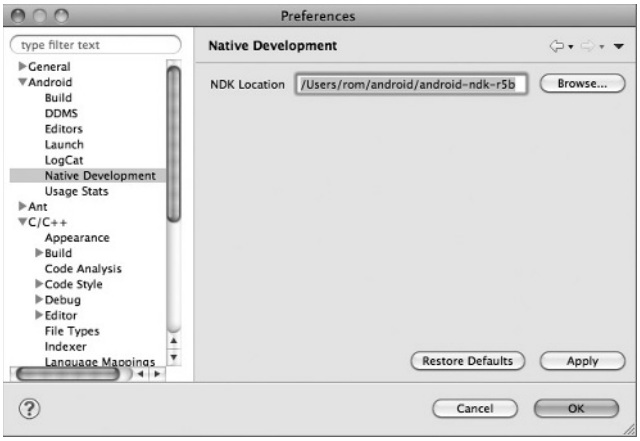

FIGURE 1-2: Specify the location of the Android NDK

Congratulations — your Android development environment is now all set! However, please note that in order to use this book with Android you will need an actual device with OpenGL ES 2.0 support. The emulator provided by the Android SDK supports only OpenGL ES 1.x, not OpenGL ES 2.0. So local deployment on the simulator is not possible on Android; only device deployment is supported when using GLES 2.

DOWNLOADING THE BOOK'S SDK

Once your development environment is set up, you should now grab a copy of this book's SDK. The official SDK is available for download at `http://www.wrox.com`. Alternatively, if you wish to download it through GIT, go to the official GFX 3D engine website, `http://gfx.sio2interactive.com`, where you can find detailed instructions.

If you have downloaded the zip file, simply decompress it in a directory that you have read and write access to. If you have downloaded it using GIT, all the files and the SDK architecture are already available on your drive.

The architecture of this book's SDK is very simple. For more information, please refer to the following directory list:

- `_chapter#-#`: Contains the final result that you should reproduce by reading the tutorials in the book. At any time while reading this book, if you feel that the instructions are not clear, or if you are unsure where to insert some code, or even if you simply want to preview the final result of a tutorial, open this directory. Inside the directory, you can then find at the root the source files used by the tutorial (respectively named `templateApp.cpp` and `templateApp.h`) and two directories that contain the project files for iOS and Android. You can then load the project into your IDE and rebuild it from scratch.

Models and textures are generously provided by David Radford (`http://dmradford.com`).

FIGURE 1-3: An FPS demo using the GFX 3D engine

- `common`: Contains the free and open source GFX 3D engine (the mini game and graphics engine that you will be using in this book) source code of the version that was used to create the templates and tutorials for this book, along with the source of the libraries the engine depends on. The GFX 3D engine is a very small and lightweight graphic engine that is built with bits and pieces of my own professional engine. It is very small, fast, flexible, and scalable; and will allow you to render state-of-the-art graphics on your mobile device, as shown in Figure 1-3.
- `data`: In this directory, you can find all the original assets that were used in each tutorial. These assets are either linked dynamically to the projects (in the case of iOS) or simply duplicated inside the `assets` directory of each Android tutorial. Please note that all the

original project 3D scenes are available as `.blend` (the default file extension of Blender). It is not mandatory, but highly suggested that you download a copy of Blender for your platform, which is available at `http://blender.org`. This will enable you to study the way the scenes are built and how the assets are linked and exported to the Wavefront OBJ (the official 3D model exchange format used in the book).

- `EULA`: In here, you can find all the End User License Agreements for the different libraries that this book's SDK relies on. If you plan to release a commercial application using this book's SDK, make sure that your application complies with all of these licenses.
- `glsloptimizerCL`: Contains the source to a simple yet powerful command line program that you can use to optimize your GLSL code (as demonstrated in Chapter 5, "Optimization").
- `md5_exporter`: A python script for Blender (v2.6x) that allows you to export bone animation sequences created in Blender to the MD5 version 10 file format (script generously provided by Paul Zirkle).
- `template`: The original template project that you will be using when creating a new project from scratch.
- `template_chapter#-#`: In order to speed up and avoid redundancies, you will duplicate these directories by following the tutorials throughout the book. This will give you a head start and save you from having to rebuild everything from scratch using the default template project.

IMPORTING PROJECTS

This book has over 50 tutorials, varying from the demonstration of a single technique to full-fledged games. To be able load and rebuild the projects from this book into your IDE, you will have to import them. To do this, just follow the instructions in the subsection that corresponds to the type of developer you are.

For iOS Developers

As usual for iOS developers, importing files is very easy. All you have to do to import a project into XCode is simply double-click the `.xcodeproj` file. To compile, simply click the `Build & Run` button.

For Android Developers

Things are a little bit more tedious if you're using Eclipse. You need to import this book's projects as instructed in the following procedure. Of course, this procedure assumes that you have properly installed and configured Android SDK, Android NDK, Eclipse Classic, the ADT plug-in, and the Sequoyah Android Native Development plug-in (as described at the beginning of this chapter).

Once you have configured all the necessary prerequisite files, follow these steps to import this book's project files:

1. From the Eclipse main menu, select File ➪ New ➪ Android Project. The New Android Project dialog should appear.
2. In the Project name text box, enter the project name. Example: **chapter2-1**.
3. Select the Create Project From Existing Source option.
4. Click the Browse button, and then select the existing Android directory inside the chapter or template project. Example: `<path_to_sdk>/SDK/_chapter2-1/Android`.
5. Click the Finish button at the bottom of the dialog box.

Figure 1-4 illustrates each of these steps.

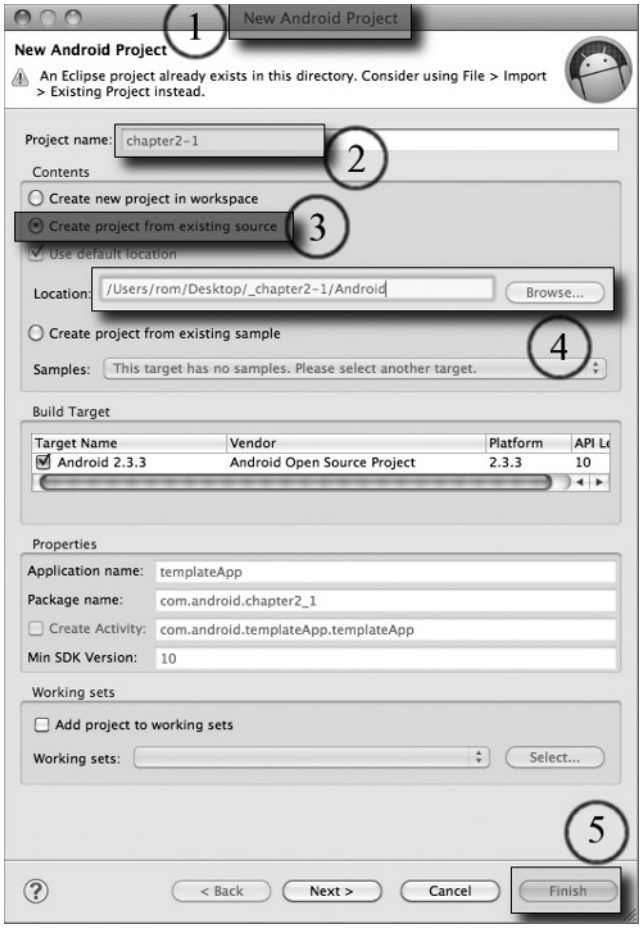

FIGURE 1-4: Importing an Android project into Eclipse

Every time you want to open an existing Android project using Eclipse, you will have to go through this importing procedure.

THE TEMPLATE

As briefly mentioned earlier in this chapter, you will work mostly with the template project that is provided inside this book's SDK. This template is a C/C++ cross-platform project that initializes internally for you a vanilla, ready-to-use OpenGLES 2 context. In addition, the template provides an `init` and `exit` function callback, which you can just plug your creation and destruction code into.

The template also provides you with an easy-to-use callback mechanism that acts as a universal HUB to handle all the platform-specific events for you.

Using this mechanism, all you have to do is to link a function callback for the specific event you want to intercept, and you'll receive updates for this event in real time. This mechanism covers all of the `touche` events such as `ToucheBegan`, `ToucheMoved`, `ToucheEnded`, as well as the accelerometer data. In other words, everything is already set up for you. You can just go ahead and create the code as instructed in this book's tutorials without having to worry about platform-specific issues.

As the title of this chapter says, it's time to get started! In order to get familiar with both the template and the type of tutorials you will be studying throughout this book, follow these instructions:

1. Duplicate the `template` project directory at the root of the SDK and rename it **template_test**.
2. Load the `template_test` project (following the appropriate importing method for your platform as described previously) into your IDE, and then open the `templateApp.cpp` (for iOS developers, it is located under the `templateApp` directory inside the Project Navigator; for Android developers, you can find it under the `jni` directory inside the Project Explorer panel).
3. Read the code comments that explain what each function is doing.
4. Uncomment the following callbacks from the initialization (`TEMPLATEAPP templateApp = {}`: `templateAppToucheBegan`, `templateAppToucheMoved`, and `templateAppToucheEnded`.
5. Move to the `templateAppInit` function and add the following code on the line before the end bracket of the function:

```
    /* Use the built-in GFX cross-platform API to print on the
console (XCode) or LogCat (Eclipse) that the execution pointer
passes the templateAppInit function. */
    console_print(
    "templateAppInit, screen size: %dx%d\n", width, height );
```

6. On the line before the end bracket of the `templateAppDraw` function callback, add the following code block:

```
    /* Specify that you want to use a chili red color to clear the
screen and spice up your app. */
    glClearColor( 1.0f, 0.0f, 0.0f, 1.0f );
    /* Report that the execution pointer was here. */
    console_print( "templateAppDraw\n" );
```

7. Add the following line before the end bracket of the `templateAppToucheBegan` function:

```
   /* Print that the execution pointer enters the touche began
function and print the touche XY value as well as the number of
taps. */
   console_print( "templateAppToucheBegan,"
                  "touche: %f,%f"
                  "tap: %d\n", x, y, tap_count );
```

8. Repeat the same procedure as in step 7 for `templateAppToucheMoved` and `templateAppToucheEnded`, updating the `console_print` text with the appropriate callback function you are dealing with.

9. Move on to the `templateAppExit` function that has already been linked to the `atexit` built-in C function, and add the following line before the end bracket of the function:

```
console_print( "templateAppExit...\n" );
```

10. Build and run the application. While the application is running, observe the console or LogCat (depending on which platform you are developing for). Touch the screen, move your finger around, and monitor in real time on the console how and in which sequence events are triggered internally.

SUMMARY

By stepping through this chapter, you now have your development environment set up. You have this book's SDK resident on your drive and have learned how to find your way around its architecture.

You now know how to import new or existing projects into XCode or Eclipse, and have a good overview of what the default template project can do for you.

You are now ready to embark on a very challenging journey in game and graphics programming. Before moving on to the next chapter, make sure that you fully understand what has been covered inside the different sections of this chapter.