Advanced Android 4 Games

Vladimir Silva

Apress®

Advanced Android 4 Games

ISBN-13 (pbk): 978-1-4302-4059-4

ISBN 978-1-4302-4060-0 (eBook)

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/bulk-sales`.

Any source code or other supplementary materials referenced by the author in this text is available to readers at `www.apress.com`. For detailed information about how to locate your book’s source code, go to `http://www.apress.com/source-code/`.

To all Android developers out there. The future of the platform is in your hands.

Contents at a Glance

Contents

About the Author

Vladimir Silva was born in Quito, Ecuador. He received a system's analyst degree from Ecuador's Army Polytechnic Institute in 1994. The same year, he came to the United States as an exchange student pursuing a master's degree in computer science at Middle Tennessee State University. After graduation, he joined the IBM WebAhead technology think tank. His interests include grid computing, neural networks, and artificial intelligence. He also holds numerous IT certifications, including Oracle Certified Professional (OCP), Microsoft Certified Solution Developer (MCSD), and Microsoft Certified Professional (MCP). He has written many technical articles on security and grid computing for IBM developerWorks.

About the Technical Reviewers

Vikram Goyal is the author of several Apress books and works from home on several web sites. He currently lives in Brisbane, Australia.

James Graham was born in Alabama and grew up in Fort Walton Beach, Florida, and San Antonio, Texas. He received a BS in electronics with a specialty in telecommunications from Texas A&M University in 1988. He has been an associate network engineer for the Network Design group at Amoco Corporation and an intelligence systems analyst for the US Air Force Special Operations Command. In addition to being an Android developer, he is the author of the highly-rated freeware hurricane tracking program, JStrack.

Acknowledgment

We would like to thank the people at Apress for all their help in making this book possible.

Introduction

Welcome to *Advanced Android 4 Games*. This book will help you create great games for the Android platform. There are plenty of books out there that tackle this subject, but only this book gives you a unique perspective by showing you how easy it is to bring native PC games to the platform with minimum effort. This is done using real-world examples and source code in each chapter. Keep in mind that before you dig into this book, you will need a solid foundation in Java and ANSI C. I have tried to clearly and simply explain the most complicated concepts with a combination of graphics and sample code. The source code provided for each chapter will help you understand the concepts in detail and make the most of your time as a mobile game developer.

The Green Robot Has Taken Off

It is hard to believe that is has been just two years since Android came onto the smartphone scene; and it has taken off with a vengeance. Take a look at the US smartphone platform market share, shown in Figure 1, according to a survey by Nielsen.[1] In May 2011, Android commanded 36 percent of the smartphone market in the United States—not too shabby for a two-year-old OS. And the stats just keep getting better and better. Distimo, an analytics company specializing in app stores, forecasted that Android Market would surpass Apple's App Store in size by August 2011.[2] This opens a new frontier for developers looking to capitalize from the rocketing smartphone segment. *Advanced Android 4 Games* is just what you need to get running quickly in building cutting-edge games for the platform.

[1] "Android Leads in U.S. Smartphone Market Share and Data Usage," Nielsen Wire, `http://blog.nielsen.com/nielsenwire/?p=27793`.

[2] "Android to Surpass Apple's App Store In Size By August 2011," a report by Distimo available at `http://techcrunch.com/2011/05/05/android-to-surpass-apples-app-store-in-size-in-august-2011-report-exclusive/`.

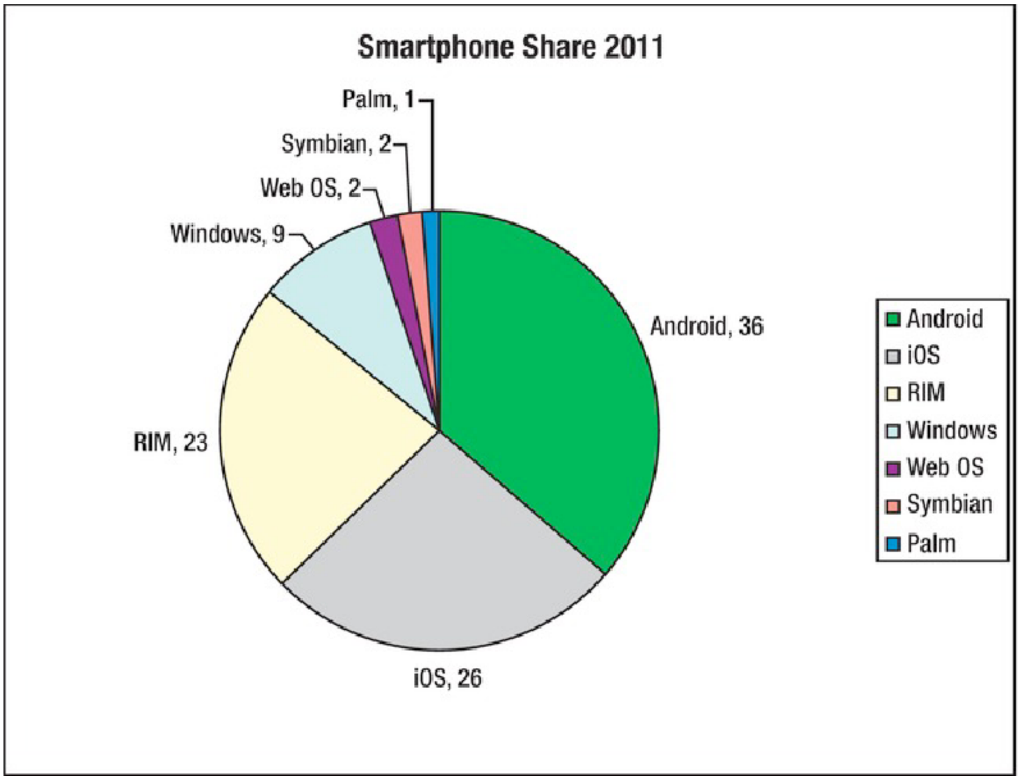


Figure F–1. Smartphone market share, April 2011, Nielsen

Who's the Target Audience?

This book targets seasoned game developers, not only in Java, but also in C. Performance is critical in game development. Other audiences include:

- *Business apps developers*. If you work on native applications, this book can be a valuable tool.
- *Scientific developers*. In the science world, raw performance matters. The chapters dealing with JNI and OpenGL can help you achieve your goals.
- *Computer science students* learning new mobile platforms. Android is open and fairly portable, thus this book can help students in many platforms, including iPhone, Blackberry, and Meego.
- *Anybody interested in Android development.* Android has taken over the mobile market space at a furious pace. You've got to expand your skill set to include games and graphics, or you may be left behind.

Skills Needed to Make the Most of This Book

The required skill set for Pro Android games includes C/C++ and Java, plus some basic LINUX shell scripting. Java provides elegant object-oriented capabilities, but only C gives you the power boost that game development requires. All in all, you must have the skill set described in the following sections.

A Solid Foundation of Android

This book assumes that you already know the basics of Android development; for example, you need to know what activities, views, and layouts are. If you understand what the following fragment does just by looking at it, then you are in good shape.

```
public class MainActivity extends Activity
{
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

This fragment defines the main activity or class that controls the life cycle of the application. The `onCreate` method will be called once when the application starts, and its job is to set the content layout or GUI for the application.

You should also have a basic understanding of how GUIs are created using XML. Take a look at the next fragment. Can you tell what it does?

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<ImageView android:id="@+id/doom_iv"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/doom"
    android:focusableInTouchMode="true" android:focusable="true"/>

 <ImageButton android:id="@+id/btn_upleft"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:src="@drawable/img1" />
</RelativeLayout>
```

This code defines a relative layout. In a relative layout, widgets are placed relative to each other (sometimes overlapping). In this case, there is an image view that fills the entire screen. This image will display as the background the file called `doom.png`, stored in the `res/drawable` folder of the project, and receive key and touch events. In the lower left of the screen, overlapping the image view, an image button with the ID `btn_upleft` will be displayed.

NEED AN ANDROID TUTORIAL?

There are a lot of concepts related to Android development and it is impossible to remember every detail about activities, views, and layouts. A handy place to access this information quickly is the Android tutorial:

http://developer.android.com/

The ultimate guide for Android developers—the latest releases, downloads, SDK Quick Start, version notes, native development tools, and previous releases—can be found at:

`http://developer.android.com/sdk/1.6_r1/index.html`

Throughout this book (especially in the chapters dealing with native code), I make extensive use of the Android Software Development Kit (SDK) command tools (for system administrator tasks). Thus, you should have a clear understanding of these tools, especially the Android Debug Bridge (adb). You should know how to do the following:

- *Create an Android Virtual Device* (AVD). An AVD encapsulates settings for a specific device configuration, such as firmware version and SD card path. Creating an AVD is really simple and can be done from the integrated development environment (IDE) by using the AVD Manager (accessed by clicking the black phone icon in the toolbar).
- *Create an SD card file*. Some of the games in later chapters have big files (5 MB or more). To save space, the code stores all game files in the device SD card, and you should know how to create one. For example, to create a 100 MB SD card file called `sdcard.iso` in your home directory, use the following command:

```
$ mksdcard 100M $HOME/sdcard.iso
```

- *Connect to the emulator*. You need to do this for miscellaneous system administration, such as library extraction. To open a shell to the device, use the following command:

```
$ adb shell
```

- *Upload and pull files from the emulator*. These tasks are helpful for storing and extracting game files to and from the device. Use the following commands:

```
$ adb push <LOCAL_FILE> <DEVICE_FILE>
$ adb pull <DEVICE_FILE> <LOCAL_FILE>
```

NOTE: Make sure the `SDK_HOME/tools` directory is added to your system `PATH` variable before running the commands to create an SD card file, connect to the emulator, or upload and pull files.

A Basic Knowledge of Linux and Shell Scripting

For the chapters dealing with the hybrid games, you will do the work within Ubuntu Linux, so dust off all those old Unix skills.

You should know the basic shell commands, such as those for listing files, installing software components (this can be tricky, depending on your Linux distribution), and basic system administration.

There are a few very simple shell scripts in this book. A basic knowledge of the bash shell is always helpful.

TIP: If you need a refresher on your Linux and shell scripting, check out the following tutorial by Ashley J.S Mills:

`http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/unixscripting/unixscripting.html`.

What Hardware/Software Will You Need?

To make the most of this book, you will need the tools mentioned in this section.

A Windows or Linux PC with a Java SDK Properly Installed

I guess this is kind of obvious, as most development for Android is done in Java. Note that I mentioned a Java SDK, not JRE. The SDK is required because of the JNI header files and command line tools used throughout the latter chapters.

Eclipse IDE and Android SDK Properly Installed

Eclipse is the de facto IDE for Android development. I have used Eclipse Galileo to create the workspace for the book; nevertheless, Eclipse Ganymede should work as well.

NEED A DEVELOPMENT IDE?

Even though Eclipse Helios has been used to create the code workspace, you can use your favorite IDE; of course, that will require a bit of extra setup. You can get Eclipse from `www.eclipse.org/`.

For instructions on how to set up the Android SDK with other IDEs, such as IntelliJ or a basic editor, see `http://developer.android.com/guide/developing/other-ide.html`.

To have the Android SDK properly installed you need to do the following:

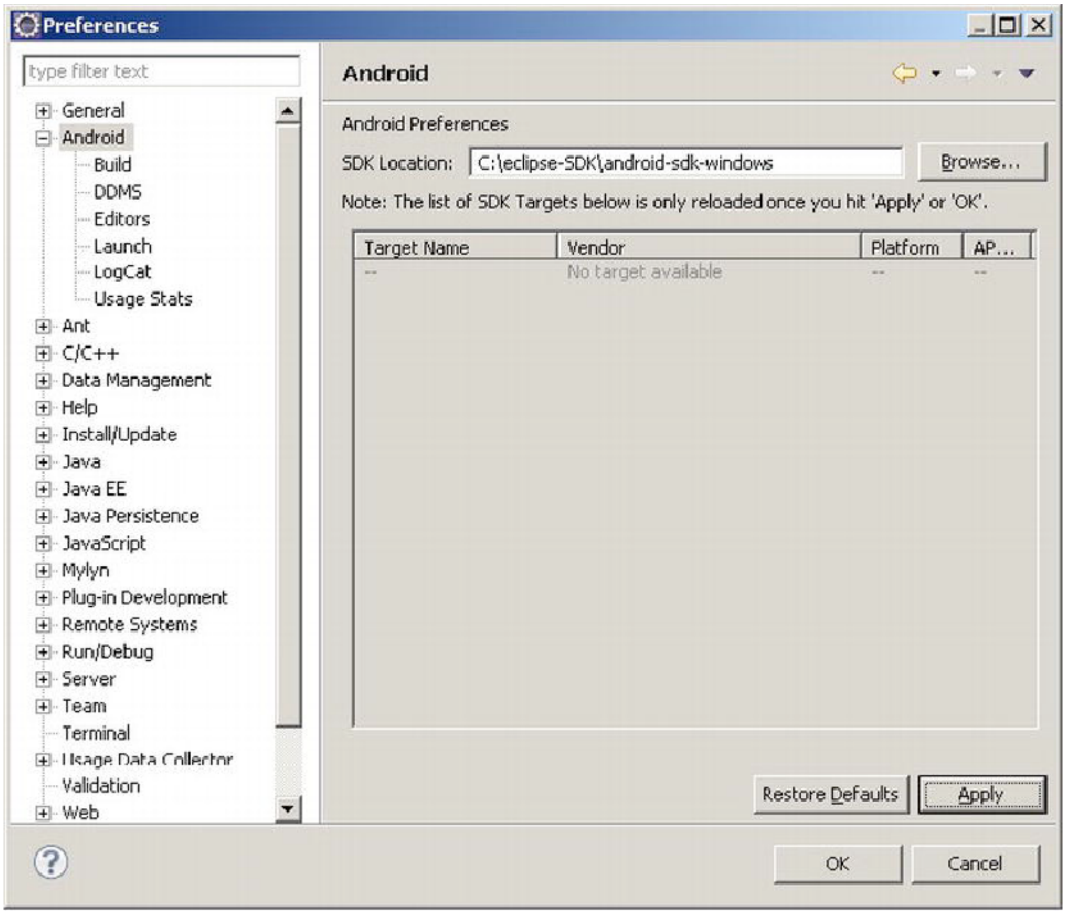
1. Install the Android SDK plug-ins for Eclipse:
    - From the IDE main menu, click **Help ➤ Install New Software**.
    - Click the Add button to add a new Site and enter:

      A name: Android SDK

A location: `https://dl-ssl.google.com/android/eclipse/`. Click OK.

- Select the Android SK from the Available Software dialog and follow the easy installation instructions from the wizard.

2. Install the Android SDK. It can be downloaded from the Android site mentioned earlier. Keep in mind that Eclipse must be told about the location of the Android SDK. From the main IDE menu, click **Window ➤ Preferences**. On the left navigation menu, select Android and enter the SDK location (see Figure 2). I used SDK 3.1 because that was the latest available at the time of this writing. Nevertheless, the code in this book has been tested with SDK 2.3 and 3.1 (see the SDK compatibility section for details).



Android Preferences

SDK Location: C:\eclipse-SDK\android-sdk-windows

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	AP...
--	No target available	--	--

Figure F-2. Android SDK configuration dialog in Eclipse Galileo

Native Development Kit (NDK)

The NDK is the essential tool for any serious game developer out there. It provides the compiler chain, header files, and documentation required to bring your cutting-edge games to the mobile landscape. By using the NDK, developers can escape the shackles of the Java memory heap and unleash their creativity in building the most powerful C/C++ engines, limited only by what the hardware can provide. In this book you will use the NDK extensively, thus a solid foundation of C programming is required to fully understand the concepts presented in each chapter.

Chapter Source

This is an optional tool, but it will help you greatly to understand the concepts as you move along. I have made my best effort to describe each chapter as simply as possible; nevertheless, some of the games (especially *Wolf 3D* and *Doom*) have very large core engines written in C (100 K lines for *Doom*), which are poorly commented and very hard to understand. All in all, you will see how easy these great languages (Java and C) can be combined with minimal effort. Get the companion source for the book at `www.apress.com`. It was built using the latest Eclipse SDK.

What Makes This Book Unique?

I think it is important for the reader to understand my goal with this manuscript and what I believe sets this book apart. Even though Java is the primary development language for Android, Google has realized the need for hybrid Java/C development if Android is to succeed as a gaming platform; so much so that they released the Native Development Kit (NDK). I think that Google has been wise to support C development; otherwise, it would be left behind by the overwhelming number of native games written for other mobile platforms, like the iPhone.
PC games have been around for decades (mostly written in C), and by using a simple ARM C compiler, you could potentially bring thousands of PC games to the Android platform. This is what makes this book unique. Why translate 100 K lines of painfully complicated code from C to Java if you can just combine both languages in an elegant manner—and save yourself lots of time and money in the process? This is my goal and what makes this book stand out. Although, the book does include chapters with pure Java games, presented in a well-balanced layout to satisfy both the Java purist and the C lover in you.

What's Changed Since the Last Edition?

With the relentless pace of Android updates, many things have changed since the last iteration of this book, *Pro Android Games*. These changes include the following:

- *Updates to the latest versions* of the Android SDK, the Native Development Kit (NKD), and the Eclipse IDE.
- *Greater focus on tablets*. People are hungry for bigger screens and higher resolutions. Tablets are growing, my friends, and we must watch out for ever-changing device resolutions and hardware specs.
- *Greater focus on the native side*. I think is fair to say that Java has fallen from grace with the 3D game developers, especially the powerful ones. Java's memory constraints and lack of performance are the main culprits. Therefore, Pro Android Games puts greater emphasis on native game development and hardware-accelerated graphics.
- *Bigger and better real-world engines*. My goal is not to simply offer you some tricks to develop games, but to provide you real, powerful, bigger-than-life samples. This book will show you how powerful PC-caliber game engines such as *Quake I* and *II* can be brought to your mobile device with almost no changes whatsoever. It will also include Doom, an oldie from the previous edition.

Android SDK Compatibility

As a developer, you may ask yourself about the SDK compatibility of the code in this book. This is an important question as new versions of the Android SDK come out frequently. At the time of this writing, Google released Android SDK version 3.2. The code in this chapter has been fully tested with Android SDK versions 4.0 and 3.1.
The bottom line is that the code in this book will run in any version of the SDK and that was my intention all along.
This book has a well-balanced layout of very powerful hybrid games, divided by chapter.

Chapter 1

This chapter provides the first step to set up a Linux system for hybrid game compilation, including fetching the Android source, extracting device system libraries, setting up a custom compilation tool chain, custom compilation scripts, and details on setting up the Eclipse IDE for use throughout the rest of the book.

Chapter 2

In Chapter 2 you will learn how to combine Java and C code in an elegant manner by building a simple Java application on top of a native library. You will learn exciting concepts about the Java Native Interface (JNI) and the API used to combine Java and C in a single unit, including how to load native libraries, how to use the native keyword, how to generate the JNI headers, as well as method signatures, Java arrays vs. C arrays, invoking Java methods, compiling and packing the product, and more.

Chapter 3

This chapter deals with 3D graphics with OpenGL. It presents a neat trick that allows for mixing OpenGL API calls in both Java and C. This concept is illustrated by using the 3D cubes sample provided by Google to demonstrate OpenGL in pure Java and hybrid modes. This trick could open a new frontier of 3D development for Android, with the potential to bring a large number of 3D PC games to the platform with enormous savings in development costs and time.

Chapter 4

Chapter 4 tackles efficient graphics with OpenGL ES 2.0. It starts with a brief description of the most important features that OpenGL ES 2 can offer, including Shaders, GLSL, and how they affect the Android platform. Then, it takes a deeper look into GLSL by creating a neat Android project to render an icosahedron using OpenGL ES 2.0. As a bonus, it will show you how you can use single and multi-touch functionality to alter the rotation speed of the icosahedron, plus pinch for zooming in or out.

Chapter 5

Chapter 5 takes things to the next level with *Doom*, the ground-breaking game for the PC. *Doom* is arguably the greatest 3D game ever created, opening new frontiers in 3D graphics. The ultimate goal of this chapter is not to describe the game itself, but to show you how easy it is to bring a complex PC game like *Doom* to the Android platform. The proof? *Doom* has more than 100 K lines of C code—and is brought to Android with less than 200 lines of extra JNI API calls, plus the Java code required to build the mobile UI. This chapter shows that you don't have to translate 100 K lines of C into Java, but simply marry these two powerful languages in an elegant application. Consider the potential savings in development time and costs! This chapter is a must read.

Chapter 6

This is where things start to get really exiting. Chapter 6 brings you a first person shooter (FPS) gem: *Quake*. You will learn how a powerful PC engine of this caliber can be brought to the Android platform with minimum effort. So much so that 95 percent of the original C code is kept intact, with an extra 500–1,000 lines of new, very simple Java wrapper code. Start playing *Quake* in all its glory on your smartphone now!

Chapter 7

This chapter builds upon Chapter 6 to deliver the *Quake II* engine to your fingertips. You will be introduced to a wonderful tool called NanoGL, which allows developers to translate the complexity of the OpenGL immediate mode drawing into OpenGL ES, transparently keeping your original code intact. You will also learn how to make the *Quake II* engine behave properly in Android by creating custom audio and video handlers, which also demonstrates the great reusability features of the Java language. All in all, 99 percent of the original *Quake II* C code will be kept intact, plus the thin Java wrappers from Chapter 6 will be reused without change. Chapter 7 will show you how a simple combo of very powerful tools can tame the mighty *Quake II* OpenGL renderer. Check it out!

Chapter 1

Welcome to the World of the Little Green Robot

This chapter kicks things off by explaining how to set up your environment to compile hybrid (C/Java) games. This includes the latest versions of the development IDE (Eclipse) and the Android SDK, plus the Native Devlopment Kit (NDK)which are the tools required to build powerful Android games. This information is critical if you wish to learn how to combine the elegant object-oriented features of Java with the raw power of C for maximum performance, and it is required when to build all the native game engines featured in later chapters.

The following software is assumed to be already installed on your desktop:

- *Eclipse:* This is the development IDE used to create your projects. At the time of this writing Eclipse Indigo version 3.7 is the latest. However Version 3.6 (Helios) or 3.5 (Galileo) will work as well.
- *Android SDK, properly configured*: At the time of this writing, the latest version of the SDK is 4.0.
- *Java JDK* : This is required to run Eclipse and the Android SDK itself. Any version of Java after 5.0 should work just fine.

In the next section we'll go through the process of setting up your machine step by step.

Setting Up Your Machine

There are a few steps to be completed before we can get to the nitty-gritty stuff of building games for Android. I summarize the steps as follows:

1. The very first and most basic thing we need is a current Java SDK/JRE (5.0 or later will do). Make sure you have the proper version installed before proceeding. The steps here assume that you already do.

2. Download and Install the Android SDK: The SDK contains the core resources to develop for Android.
3. Configure Your Eclipse: You need to install the Eclipse plugin for Android before you can build anything at all.
4. Install the Native Development Kit (NDK) if you don't have it: This is a critical component for any kind of game that uses native APIS such as OpenGL. By the time of this writing the latest version is r6b. All in all, keep in mind that Android 4 is binary compatible with older NDK versions. This means that if you have an old NDK it will work just fine. Nevertheless, it is always good to use the latest version, as it may provide a performance boost to your native code.
5. Create an Emulator: This is an optional step that will help you with testing your games in many API versions and screen sizes.
6. Configure a Real device: I prefer to work in a real device because it so much faster than using an emulator and is the best way to work if you use OpenGL.

Download and Install the SDK

Download the Android SDK Starter Package (r14) for windows from `http://developer.android.com/sdk/index.html` and unzip it to a working folder such as `C:\eclipse-SDK`.

TIP: Try to keep the SDK, NDK, and Eclipse in the same work folder such as C:\eclipse-SDK. I find this helpful when working on multiple projects at the same time. Thus my development folder C:\eclipse-SDK contains the subfolders: android-sdk-windows (for the SDK), android-ndk-r6b (for the NDK), and eclipse (for Eclipse 3.7). Now let's configure our Eclipse environment.

Configure your Eclipse

You are ready to get your IDE up and running with the Android development kit. Let's go through the installation of the Android 4 SDK (available from `http://developer.android.com/sdk/index.html`) over Eclipse 3.7 (Indigo, available from `http://www.eclipse.org`).

1. Start Eclipse and select **Help ➤ Check for Updates**, as shown in Figure 1–1.

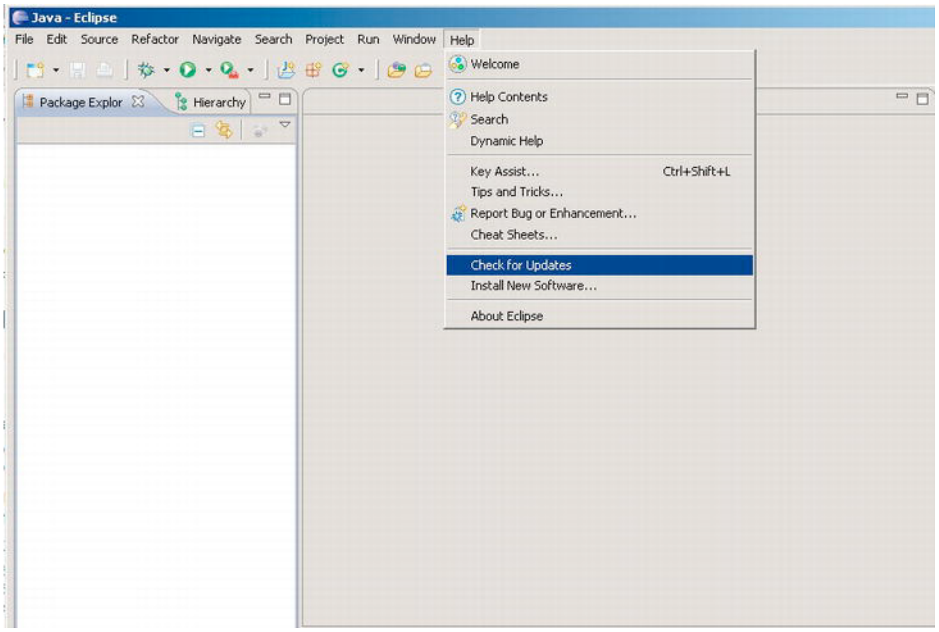
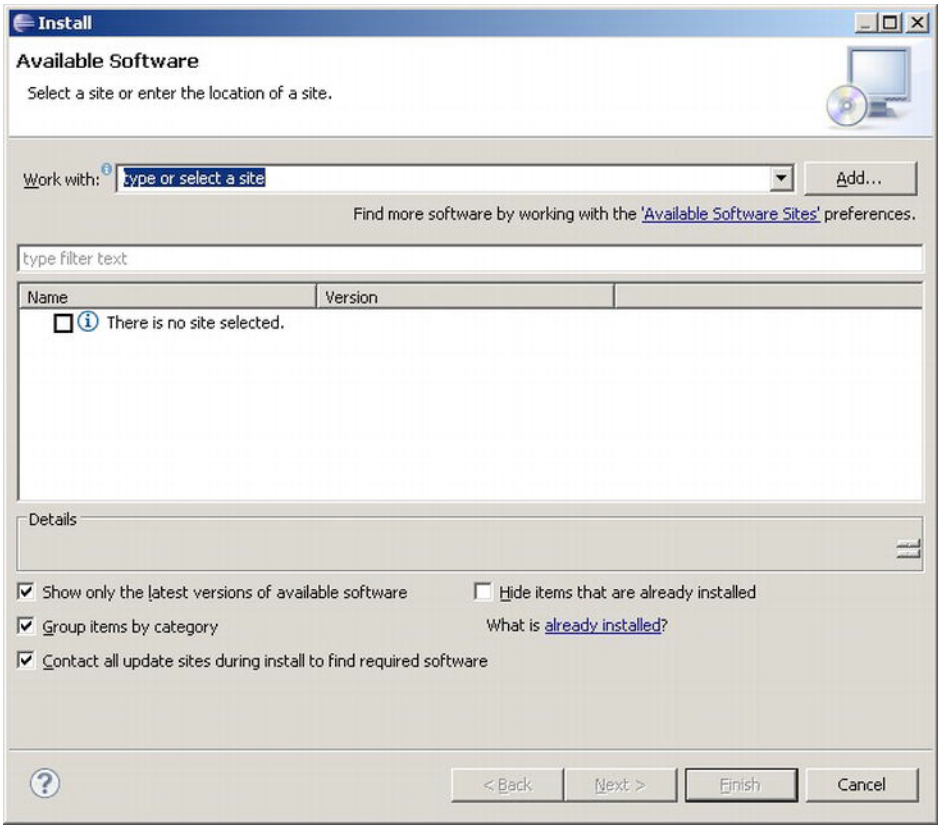


Figure 1–1. *Choosing Check for Updates from the Eclipse 3.7 workbench's Help menu*

2. In the Available Software window, shown in Figure 1–2, click the Add button to install new software.


Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the 'Available Software Sites' preferences.

type filter text

Name	Version	
☐ There is no site selected.		

Details

☑ Show only the latest versions of available software

☐ Hide items that are already installed

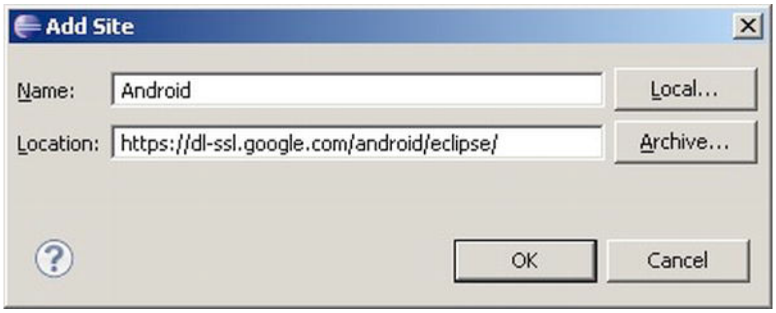
☑ Group items by category

What is already installed?

☑ Contact all update sites during install to find required software

Figure 1–2. *Choosing to add software*

3. In the Add Site dialog box, enter **Android** for the name and `https://dl-ssl.google.com/android/eclipse` for the location, as shown in Figure 1–3.

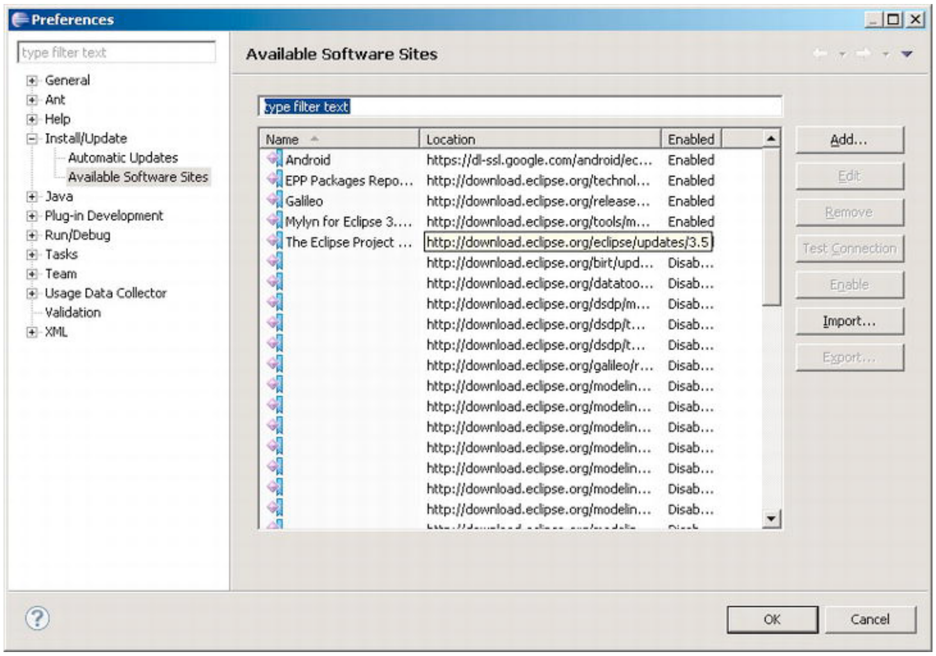

Add Site

Name: Android

Location: https://dl-ssl.google.com/android/eclipse/

Figure 1–3. *Adding the Android site*

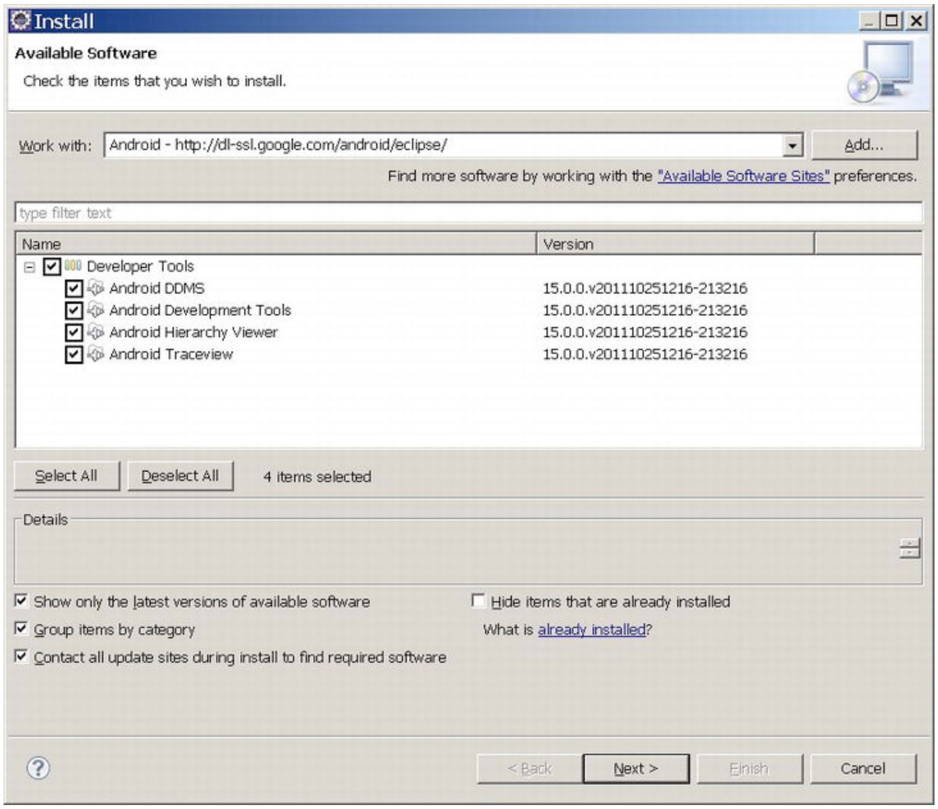
4. From the Available Software window (Figure 1–2), select the Android site you just added from the Work with combo box. If the name is not shown in the list, click the Available Software Sites preferences link, and then click Add to insert the site into the list (see Figure 1–4).



Name	Location	Enabled
Android	https://dl-ssl.google.com/android/ec...	Enabled
EPP Packages Repo...	http://download.eclipse.org/technol...	Enabled
Galileo	http://download.eclipse.org/release...	Enabled
Mylyn for Eclipse 3....	http://download.eclipse.org/tools/m...	Enabled
The Eclipse Project ...	http://download.eclipse.org/eclipse/updates/3.5	
	http://download.eclipse.org/birt/upd...	Disab...
	http://download.eclipse.org/datatoo...	Disab...
	http://download.eclipse.org/dsdp/m...	Disab...
	http://download.eclipse.org/dsdp/t...	Disab...
	http://download.eclipse.org/dsdp/t...	Disab...
	http://download.eclipse.org/galileo/r...	Disab...
	http://download.eclipse.org/modelin...	Disab...
	http://download.eclipse.org/modelin...	Disab...
	http://download.eclipse.org/modelin...	Disab...
	http://download.eclipse.org/modelin...	Disab...
	http://download.eclipse.org/modelin...	Disab...
	http://download.eclipse.org/modelin...	Disab...
	http://download.eclipse.org/modelin...	Disab...

Figure 1–4. *The Available Software Sites Preferences window shows the recently added Android site.*

5. Check the Developer Tools check box in the details section, as shown in Figure 1–5, and then click Next.



Work with: Android - http://dl-ssl.google.com/android/eclipse/

Find more software by working with the "Available Software Sites" preferences.

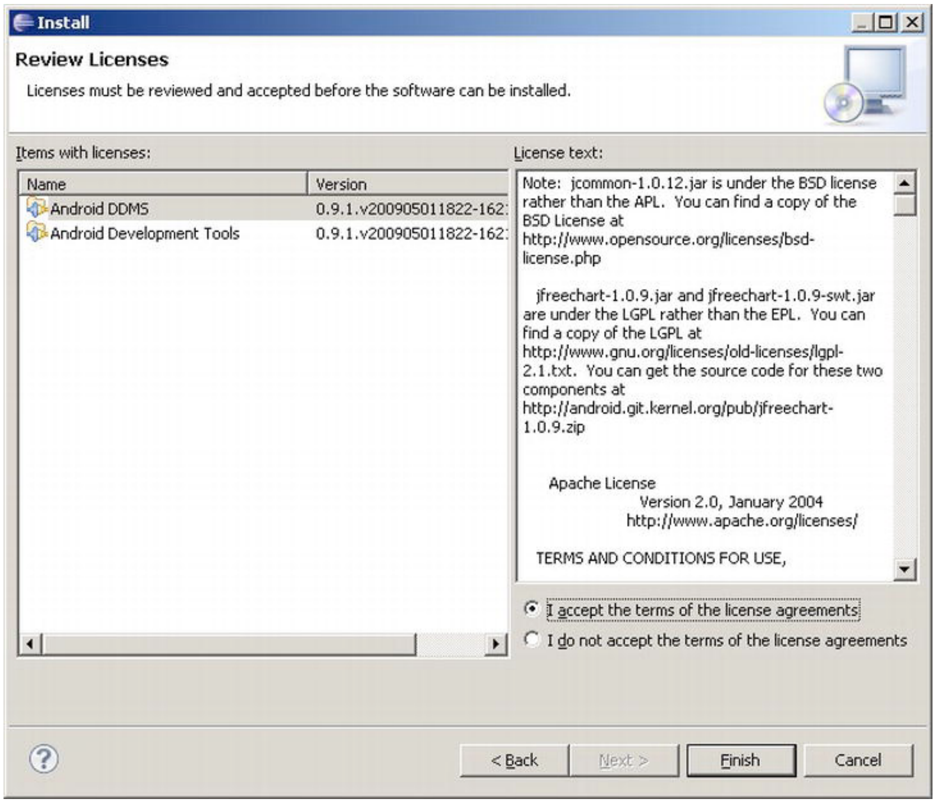
Name	Version
☑ Developer Tools	
☑ Android DDMS	15.0.0.v201110251216-213216
☑ Android Development Tools	15.0.0.v201110251216-213216
☑ Android Hierarchy Viewer	15.0.0.v201110251216-213216
☑ Android Traceview	15.0.0.v201110251216-213216

4 items selected

☑ Show only the latest versions of available software
☑ Group items by category
☑ Contact all update sites during install to find required software
☐ Hide items that are already installed
What is already installed?

Figure 1–5. *Available Software window with the Android plug-in selected*

6. Follow the wizard installation instructions, accept the license agreement, as shown in Figure 1–6, and then complete the installation. At the end, the workbench should ask for a restart.



Licenses must be reviewed and accepted before the software can be installed.

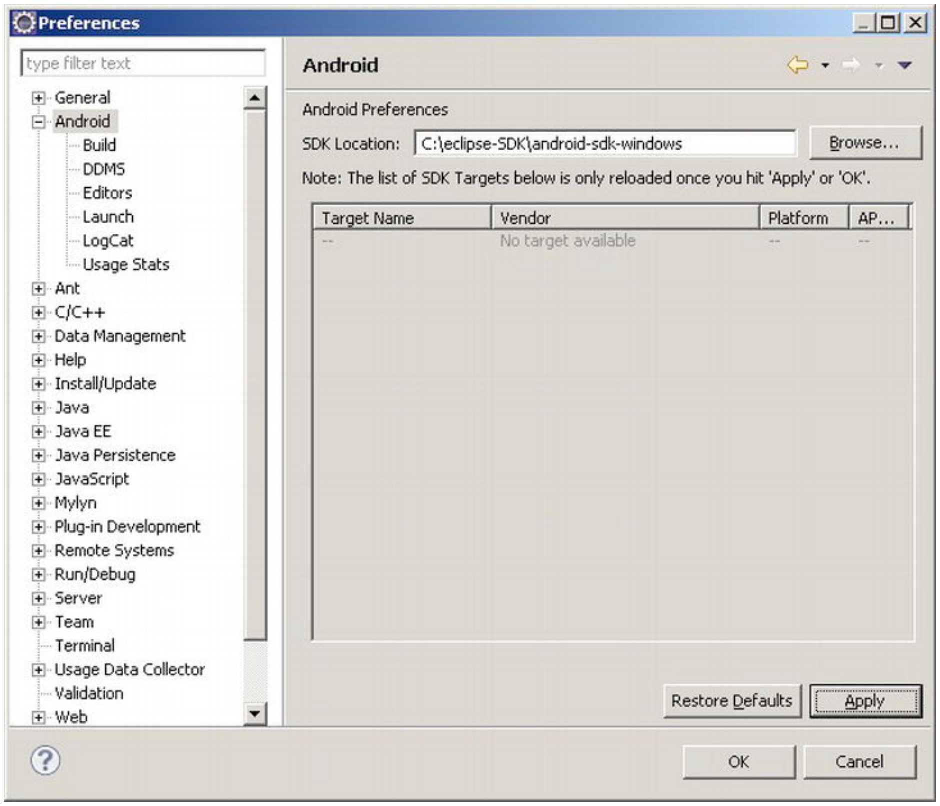
Name	Version
Android DDMS	0.9.1.v200905011822-162
Android Development Tools	0.9.1.v200905011822-162

```
Note:  jcommon-1.0.12.jar is under the BSD license
rather than the APL.  You can find a copy of the
BSD License at
http://www.opensource.org/licenses/bsd-
license.php

  jfreechart-1.0.9.jar and jfreechart-1.0.9-swt.jar
are under the LGPL rather than the EPL.  You can
find a copy of the LGPL at
http://www.gnu.org/licenses/old-licenses/lgpl-
2.1.txt.  You can get the source code for these two
components at
http://android.git.kernel.org/pub/jfreechart-
1.0.9.zip


    Apache License
                Version 2.0, January 2004
             http://www.apache.org/licenses/

  TERMS AND CONDITIONS FOR USE,
```

[✓ I accept the terms of the license agreements] I do not accept the terms of the license agreements

Figure 1–6. *Software license agreement from the installation wizard*

7. After the workbench restarts, select **Window ➤ Preferences** to open the workbench Preferences window, and select the Android option from the left navigation tree. In the Preferences section, set the location of your SDK, as shown in Figure 1–7. Make sure all the build targets are shown. Then click Apply.



Android

Android Preferences

SDK Location: C:\eclipse-SDK\android-sdk-windows

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	AP...
--	No target available	--	--

Figure 1–7. *Workbench Preferences window showing Android options*

8. Click OK, and then open the New Project wizard to make sure the Android plug-in has been successfully installed. If so, you should see a folder to create an Android project, as shown in Figure 1–8.

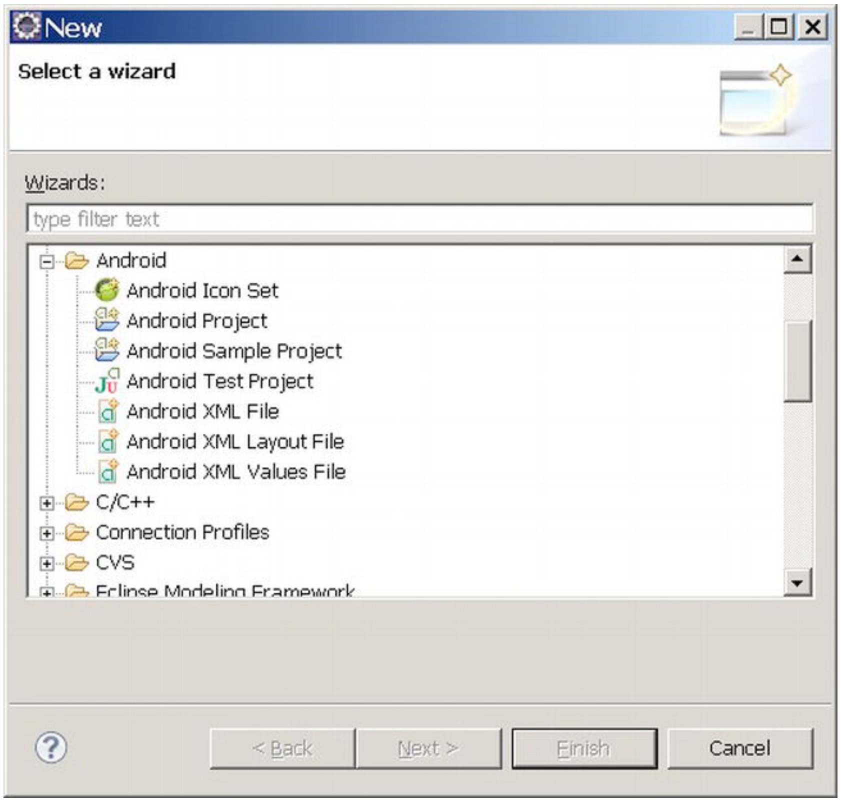


Figure 1–8. *New Project wizard showing the Android options after final configuration*

Our Eclipse is ready for use. Now we must install the NDK.

Installing the Native Development Kit

The NDK is the critical component to create OpenGL games. It provides all the tools (compilers, libraries, and header files) to build apps that access the device natively.

NOTE: The NDK site is a very helpful resource to find step-by-step instructions, API descriptions, changes, and all things related to native development. It is a must for all C/C++ developers.
`http://developer.android.com/sdk/ndk/index.html`

The NDK installation requires two simple steps: downloading the NDK and installing Cygwin.

NDK Install

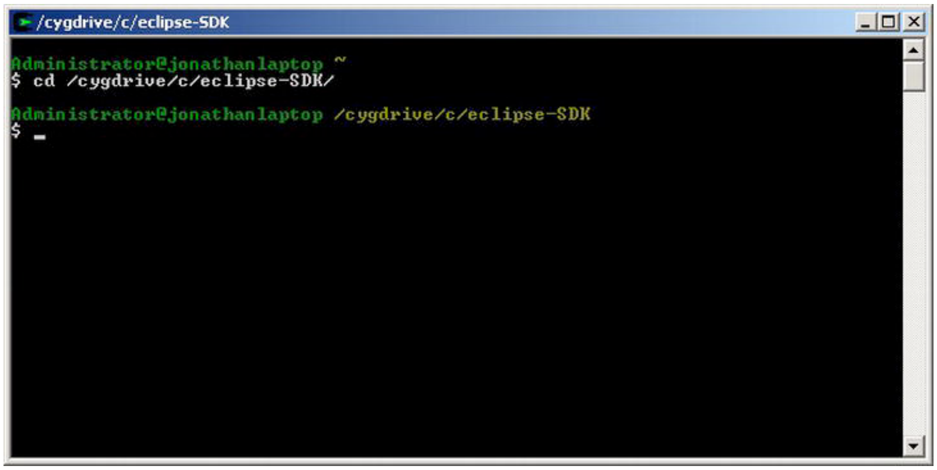
Download and unzip the latest NDK from `http://developer.android.com/sdk/ndk/index.html` into your work folder (in my case C:\eclipse-SDK).

Install Cygwin

Android is built on top of Linux which mixes with windows just like oil and water. Cygwin (version 1.7.9-1) is a tool that provides a Linux look and feel environment for Windows. It is necessary to run the NDK compilation scripts, and it is required if you are doing any type of native development in Windows.

NOTE: Cygwin is not required for native development in Linux.

To Install Cywin, download and run the installer (setup.exe) from the Cywin site available at `http://www.cygwin.com/`. Follow the wizard instructions. After the installer completes you should see the Cygwin icon in your desktop. Double click it and test by changing to your work folder (type cd /cygdrive/c/eclipse-SDK, see Figure 1–9).



```
Administrator@jonathanlaptop ~
$ cd /cygdrive/c/eclipse-SDK/

Administrator@jonathanlaptop /cygdrive/c/eclipse-SDK
$ _
```

Figure 1–9. *Cygwin console*

Creating an Android Emulator

The first step in creating our native app is to create an Android Virtual Device (AVD) we can use to test it. However, if you have a real device such as a phone or tablet you can skip this section and jump to Configure a Real Device.