Jonathan Harbour

Sams Teach Yourself

Android Game Programming in 24 Hours

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Jonathan Harbour

Sams Teach Yourself

Android Game Programming in 24 Hours

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself Android Game Programming in 24 Hours

Copyright © 2013 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-33604-1
ISBN-10: 0-672-33604-9

Library of Congress Cataloging-in-Publication Data is on file.

Printed in the United States of America

First Printing November 2012

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearsoned.com

Editor-in-Chief
Greg Wiegand

Acquisitions Editor
Neil Rowe

Development Editor
Mark Renfrow

Managing Editor
Kristy Hart

Project Editor
Elaine Wiley

Copy Editor
Barbara Hacha

Indexer
Joy Lee

Proofreader
Chrissy White,
Language Logistics

Technical Editor
Chris Bossardet

Editorial Assistant
Cindy Teeters

Cover Designer
Mark Shirar

Compositor
Nonie Ratcliff

Contents at a Glance

Introduction xvii

Part I: Introduction

HOUR 1 Introducing Android 4 3
2 Installing the Development Tools 15
3 Configuring NetBeans and Eclipse with the Android SDK 31
4 Creating Your First Android Program 47

Part II: Android Hardware

HOUR 5 Getting Started with Graphics 77
6 Drawing Basic Shapes and Text 93
7 Loading and Drawing Images 111
8 Bringing Your Game to Life with Looping 129
9 Multi-Touch User Input 143
10 Using the Accelerometer 157
11 Using the Linear Acceleration and Proximity Sensors 169
12 Using the Gravity and Pressure Sensors 181
13 Creating Your Own "Tricorder" 191
14 Playing with the Audio System 213

Part III: Android Gameplay

HOUR 15 Building an Android Game Engine 225
16 Creating a Sprite/Actor Class 255
17 Frame Animation Using a Sprite Sheet/Atlas 269
18 Advanced Multi-Animation Techniques 281
19 Manipulating Sprites with Matrix Transforms 299
20 Entity Grouping 321
21 Collision Detection 333

22 Using Linear Velocity for Realistic Movement 349
22 Scrolling the Background 371
22 Ball and Paddle Game 385
Index 397

Table of Contents

Introduction . . . xvii

Part I: Introduction

HOUR 1: Introducing Android 4 . . . 3

Hello, Android 4 . . . 3
About the Android SDK . . . 7
About the Android NDK . . . 8
Android Dev System Requirements . . . 8
History of the Platform . . . 9
Android Hardware Specifications . . . 11
Summary . . . 13
Q&A . . . 13
Workshop . . . 14

HOUR 2: Installing the Development Tools . . . 15

Installing the JDK . . . 16
Downloading the NetBeans Package . . . 17
Installing the Package . . . 17
Installing the Android SDK . . . 19
Downloading the SDK . . . 20
Installing the SDK . . . 20
Running the Android SDK Manager . . . 23
Installing the ADT Plug-in for Eclipse . . . 25
Summary . . . 28
Q&A . . . 29
Workshop . . . 29

HOUR 3: Configuring NetBeans and Eclipse with the Android SDK 31
Creating an Android Emulator Device 31
Plugging Android SDK into NetBeans 35
Adding Android SDK Support to Eclipse 40
Summary 45
Q&A 46
Workshop 46

Hour 4: Creating Your First Android Program 47
Creating a New Android Project 47
Building the New Project 52
Editing the "Hello, Android!" Program 60
Comparing the Emulator to an Android Device 63
Summary 72
Q&A 72
Workshop 72

Part II: Android Hardware

HOUR 5: Getting Started with Graphics 77
Understanding the Activity Class 77
Testing the Activity States 79
World's Simplest Android Graphics Demo 86
Summary 90
Q&A 91
Workshop 91

HOUR 6: Drawing Basic Shapes and Text 93
Drawing Basic Vector Shapes 93
Drawing Text 99
Writing Code for Javadoc 103
Android Screen Densities and Resolutions 104
Summary 109
Q&A 110
Workshop 110

HOUR 7: Loading and Drawing Images **111**
Double-Buffered Drawing 111
Loading a Bitmap File 115
Drawing a Bitmap 120
Summary 126
Q&A 126
Workshop 127

HOUR 8: Bringing Your Game to Life with Looping **129**
Creating a Threaded Game Loop 129
Drawing Without `onDraw()` 132
The Runnable Animation Demo 134
Summary 140
Q&A 141
Workshop 141

HOUR 9: Multi-Touch User Input **143**
Single-Touch Input 143
Multi-Touch Input 148
Summary 155
Q&A 155
Workshop 156

HOUR 10: Using the Accelerometer **157**
Android Sensors 157
Summary 168
Q&A 168
Workshop 168

HOUR 11: Using the Linear Acceleration and Proximity Sensors **169**
Accessing the Linear Acceleration Sensor 169
Accessing the Proximity Sensor 177
Summary 178
Q&A 178
Workshop 178

HOUR 12: Using the Gravity and Pressure Sensors **181**
Using the Gravity Sensor 181
Using the Pressure Sensor 188
Summary 189
Q&A 190
Workshop 190

HOUR 13: Creating Your Own "Tricorder" **191**
Encapsulating the Android Sensors 191
Creating the Tricorder Project 195
Summary 211
Q&A 211
Workshop 211

HOUR 14: Playing with the Audio System **213**
Playing Audio Using `MediaPlayer` 213
Playing Audio Using `SoundPool` 218
Summary 221
Q&A 221
Workshop 221

Part III: Android Gameplay

HOUR 15: Building an Android Game Engine **225**
Designing an Android Game Engine 226
Creating an Android Library Project 229
Writing the Core Engine Classes 234
Engine Test Demo Project 247
Summary 253
Q&A 253
Workshop 254

HOUR 16: Creating a Sprite/Actor Class **255**
Static Sprite as a "Prop" 255
Dynamic Sprite as an "Actor" 257
Encapsulating Basic Sprite Functionality 258

Testing the `Sprite` Class . . . 261
Summary . . . 266
Q&A . . . 266
Workshop . . . 267

HOUR 17: Frame Animation Using a Sprite Sheet/Atlas . . . 269
Animating with a Single Strip . . . 269
Animating with a Sprite Sheet (Texture Atlas) . . . 272
The Animation Demo . . . 273
Summary . . . 278
Q&A . . . 279
Workshop . . . 279

HOUR 18: Advanced Multi-Animation Techniques . . . 281
Creating an Animation System . . . 281
Animation System Demo . . . 293
Summary . . . 297
Q&A . . . 297
Workshop . . . 298

HOUR 19: Manipulating Sprites with Matrix Transforms . . . 299
Matrix Translation . . . 299
Matrix Rotation . . . 305
Matrix Scaling . . . 306
Matrix Transforms Demo . . . 307
Summary . . . 319
Q&A . . . 320
Workshop . . . 320

HOUR 20: Entity Grouping . . . 321
Entity Grouping . . . 321
Summary . . . 332
Q&A . . . 332
Workshop . . . 332

HOUR 21: Collision Detection 333

Collision Detection Techniques 333

Demonstrating Collisions 337

Summary 347

Q&A 348

Workshop 348

HOUR 22: Using Linear Velocity for Realistic Movement 349

Calculating Velocity from a Direction 349

“Pointing” a Sprite in the Direction of Movement 352

Enhancing the Engine 355

Summary 368

Q&A 369

Workshop 369

HOUR 23: Scrolling the Background 371

Background Scrolling Overview 371

The *Shoot-’Em-Up* Game 374

Summary 382

Q&A 382

Workshop 382

HOUR 24: Ball and Paddle Game 385

Creating the Ball and Paddle Game 385

Summary 394

Q&A 395

Workshop 395

Index 397

Foreword

When Jonathan Harbour asked me to write the foreword to this book, I was quite honored. I first met Jon when I started teaching game design at the University of Advancing Technology in Tempe, Arizona. As a novice teacher, I was very grateful to Jon for offering his advice and assistance. Because he taught game programming and I taught game design, it was natural that we would work together.

We also hit it off simply as gamers. We both love strategy games, and we found that we are both huge board wargame fans. We especially enjoyed a WWII battle game called Memoir '44, but our most intense confrontations were in Twilight Struggle, a game covering the entire Cold War period in an innovative card-driven format.

We soon discovered that we also shared similar philosophies about teaching and game development—that game development is hard work, and to prepare our students for careers in the game industry requires that we challenge them and hold them to the highest standards. So when Jon asked me to work with him and a team of students on a small XNA game project, I jumped at the opportunity! We assembled a strong team and spent some time getting to know each other in order to understand our collective skills and strengths.

After a period of brainstorming, research, and concept development we chose to do a 2D side-scrolling platformer, but not just another run-of-the-mill platformer! We really wanted to have some fun, but we also wanted to see if we could find a way to innovate a little. The game we ended up making was Aquaphobia: Mutant Brain Sponge Madness. As the game developed, we found that we were attracting a lot of attention at the school. People were charmed by the main character, the setting, and the overall art style—and the basic gameplay was undeniably fun! UAT honored us with a sponsorship to the Game Developer's Conference (GDC) Austin that summer.

Our follow-up was a more ambitious project. We proposed and received approval to merge Jon's mobile game programming course with my handheld game design course and to have all of the students in both classes work together on a single project. We would make a game for the Nintendo DS, and the concept we pitched was a straightforward translation of the popular board game Memoir '44. The project didn't pan out for a variety of reasons, but as any teacher will assure you, you learn more from your mistakes than you do from your successes! I think our students learned a LOT from that experience, and I know that Jon and I both did!

The bottom line is this: Jonathan Harbour is deeply passionate about making games. He also loves teaching. The book you hold will help you learn to make games, too. Enjoy!

David Wessman
Game Designer

About the Author

Jonathan Harbour is a writer and instructor whose love for computers and video games dates back to the Commodore PET and Atari 2600 era. He has a Master's in Information Systems Management. His portfolio site at www.jharbour.com includes a discussion forum. He also authored *Sams Teach Yourself Windows Phone 7 Game Programming in 24 Hours.* His love of science fiction led to the remake of a beloved classic video game with some friends, resulting in Starflight—The Lost Colony (www.starflightgame.com).

Dedication

This book is dedicated to my friend and colleague, David Wessman. I enjoyed working with David as a fellow instructor at UAT during 2009-2010. Among his many game credits is TIE Fighter (LucasArts).

Acknowledgments

This book was a challenging project because of the quickly evolving Android platform. I am thankful to the production team at Pearson for their patience during the long writing process (including missed deadlines) and hard work to get it into print. Neil Rowe; Mark Renfrow; Barbara Hacha; Elaine Wiley; and technical reviewer, Chris Bossardet.

We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email:	consumer@samspublishing.com
Mail:	Sams Publishing ATTN: Reader Feedback 800 East 96th Street Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

QR Barcodes

You may use these quick reference barcodes with your smartphone scanner to receive links to information about the book!

Publisher's Book Detail Link

Author's Website Link

This page intentionally left blank

Introduction

Since Google acquired Android, Inc., to compete with Apple and Microsoft in the smartphone and tablet markets, competition has heated up in this lucrative market. These are two tough competitors, but Android quickly gained a strong market share in a short time, with Google celebrating its 500 millionth Android OS sale. (Although Android is a license-free OS, devices are still registered with Google—at no cost). Both Apple and Microsoft have invested *billions* to develop and market their proprietary platforms, whereas Google has taken the open standards approach of releasing the source code to Android (which is based on the Linux core). This has allowed smartphone and tablet manufacturers to customize the OS for their devices while maintaining "app" compatibility across the line. Android literally is comparable to Apple's iOS devices in quality and performance, with an equally impressive online shop for purchasing music, books, movies, and apps: Google Play.

Android 4 was an especially important update to the OS, which seems to have been such a big hit that hardware manufacturers are largely leaving it alone—the stock OS—rather than customizing it for their devices. In the past, companies like Toshiba and Samsung have released custom versions that gave their devices a unique look and feel. But that practice is in decline as the OS gained notoriety and branding. An exclusion today is Amazon's Kindle Fire HD, which runs the Android 4 OS with many custom Amazon apps to give the dwevice a uniqueness that leverages the equally impressive Kindle Fire brand.

This book is about writing games for the Android 4 mobile operating system used in smartphones and tablets. The ideal reader for this book is a programmer who knows Java and has already dabbled in game programming before, and who needs a primer for the Android platform. This book is not extremely advanced; the reader level is beginning to intermediate, with absolutely no 3D covered (via OpenGL ES 2.0). An entire book is needed to cover OpenGL ES properly, and our goal with this book is to introduce the most important concepts in developing games for Android 4, not to address high-performance rendering. However, this book *will* take you right up to the point where you will be able to look into OpenGL ES. You will gain a solid understanding of the Android hardware, including the display system, audio system, sensors, and touch screen. A sample game engine is demonstrated in the final hours.

The Android SDK is based on the Java language, so this book's code revolves around Java. The SDK and development tools are free to download and install, and this book explains step by step how to do so, making it suitable for a beginner. The approach taken is that the reader is a knowledgeable person, with some experience at programming already, and is looking for a quick head-start to developing games on the Android platform. The book moves along at a leisurely pace, not getting too technical right away, simply showing the reader how everything works in a step-by-step fashion—in other words, how to get an Android game up and running fairly quickly. The Android SDK is a challenge to set up and use for a complete novice, so we cover every detail on getting started with the tools. Although a reader will greatly benefit from having at least some experience with the Java language, we do not make the assumption and will explain the code for each example. Then, after the basic hurdles are overcome, the latter half of the book delves into some serious gameplay code at a higher level.

In Part I, covering Hours 1–4, you learn how to install and configure the development tools and the Android SDK.

In Part II, covering Hours 5–14, you learn all about the Android OS and how to use the Android devices supported by the SDK, such as the graphics system, touch screen, audio system, and sensors (such as the accelerometer).

In Part III, covering Hours 15–24, you learn how to create a basic game engine for the Android platform with helper classes covering the common gameplay features needed to program most video games, such as sprites and a customizable animation system. The last two hours present game examples to demonstrate the concepts.

To download the source code for this book (as an Eclipse workspace), see the author's website at http://jharbour.com or the publisher's website at http://www.informit.com/store/product.aspx?isbn=0672336049.

HOUR 3

Configuring NetBeans and Eclipse with the Android SDK

What You'll Learn in This Hour:

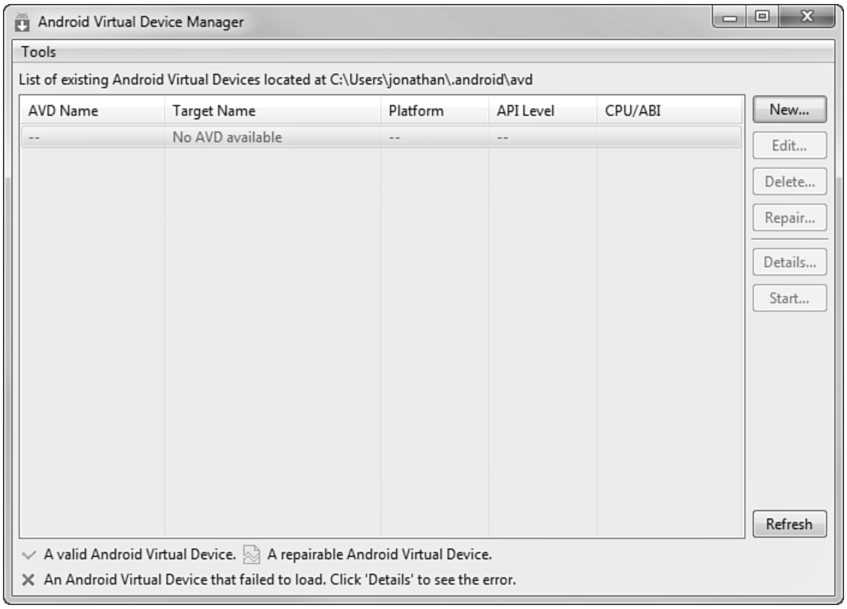
- ▶ Creating an Android emulator device
- ▶ Running the emulator
- ▶ Adding the Android plug-in to NetBeans
- ▶ Adding the Android plug-in to Eclipse

This hour covers additional prerequisites needed to use the Android SDK with an IDE. We're taking this in small steps now with plenty of figure examples to act as a quick reference for your Android programming projects to come. In this hour, you learn how to use the Android Virtual Device Manager to set up the emulator to run your Android programs. Then you learn how to add the Android SDK to NetBeans and Eclipse. The SDK was already installed in Hour 2, "Installing the Development Tools," so if you skipped that step, you will need to go back and install it.

Creating an Android Emulator Device

If you think that there are a lot of steps required just to get up and running with Android, you would be right! But we're on the right track and almost done with all of the prerequisites. Soon we will be writing game code. First, what you need to do is configure an Android emulator. An emulator is called Android Virtual Device, or AVD. You must use the Android Virtual Device Manager, shown in Figure 3.1, to create an emulator device.

The reason for needing an emulation *manager* is because of all the Android OS versions that have come out so quickly, in just the past three years. Also, developers might need to test their programs on more than one version of the Android OS to ensure that they work correctly.



List of existing Android Virtual Devices located at C:\Users\jonathan\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
--	No AVD available	--	--	

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✕ An Android Virtual Device that failed to load. Click 'Details' to see the error.

FIGURE 3.1
The Android Virtual Device Manager is used to set up the Android emulator.

Creating a New Emulator Device

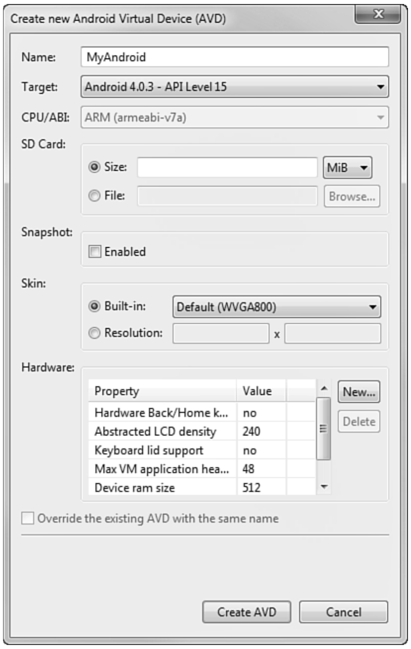
First, we'll create an emulator device. Click the New button on the right side of the AVD Manager. This brings up the dialog shown in Figure 3.2, Create New Android Virtual Device (AVD). If AVD Manager is not running, you can find it in Program Files under Android SDK Tools.

As you can see, a lot of options exist for the emulator! First, we'll focus on the Target field, which has a drop-down list of Android OS targets. This list will be quite small if you installed only 4.0 or 4.1 (using the Android SDK Manager in the previous hour). If multiple SDKs are installed, you will be able to choose the version you want to emulate.

Give your new emulator device a name, such as MyAndroid (or a descriptive name related to the settings chosen).

Choose the target for Android 4. It might say 4.0.3 or 4.1 or some other revision, depending on the specific version you installed on your dev PC.

The CPU/ABI field should be grayed out for Android 4 because devices use a standard CPU. If, for any reason, this field is not grayed out (for instance, if you are targeting API 14 or earlier), be sure to set it to ARM. Again, this shouldn't be necessary if you're using the latest version of the API.


Create new Android Virtual Device (AVD)

Name: MyAndroid
Target: Android 4.0.3 - API Level 15
CPU/ABI: ARM (armeabi-v7a)
SD Card: (●) Size: MiB ○ File:
Snapshot: ☐ Enabled
Skin: (●) Built-in: Default (WVGA800) ○ Resolution: x
Hardware:

Property	Value
Hardware Back/Home k...	no
Abstracted LCD density	240
Keyboard lid support	no
Max VM application hea...	48
Device ram size	512

☐ Override the existing AVD with the same name

FIGURE 3.2
Creating a new emulator—Android Virtual Device.

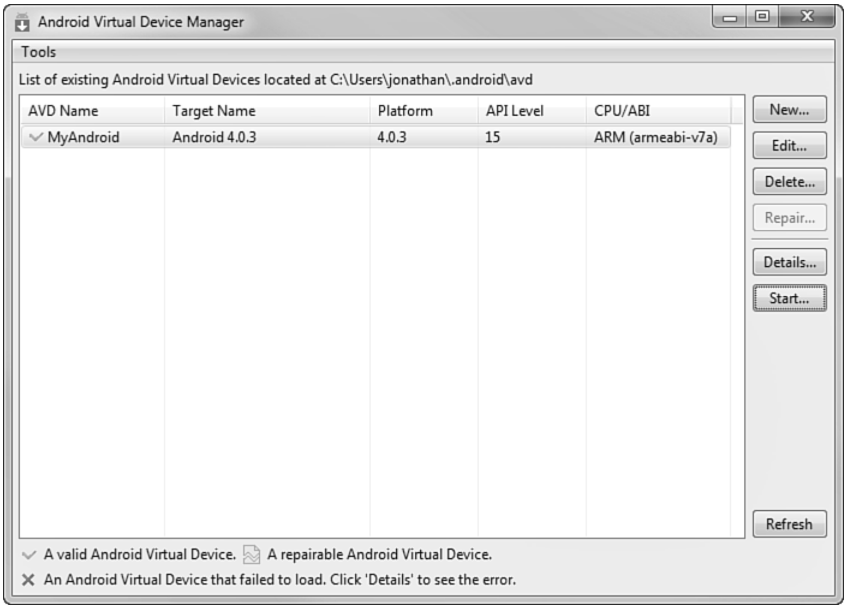
If you want to simulate an SD Card in the emulator, you can specify the size of the SD Card.

The display setting is a challenge because there are so many options. It's probably safe to go with WVGA800, although there are others. This will differ quite significantly depending on the hardware you want to emulate. For instance, if you want to emulate a specific smartphone model, you would look up the screen resolution for that phone. But if you want to emulate a tablet, it will likely have a different screen. This allows you to create more than one emulator device for these various possibilities in the hardware.

Figure 3.3 shows the AVD Manager with the new device added to the list. An emulator device called MyAndroid has been added. If you want to quickly peruse the settings for any device, double-click the device in the list to bring up a mini detail dialog.

Running the Emulator

Choose your emulator device in the list and click the Start button on the right. This brings up the mini launch dialog shown in Figure 3.4. You can tweak a few options if desired and then click the Launch button.



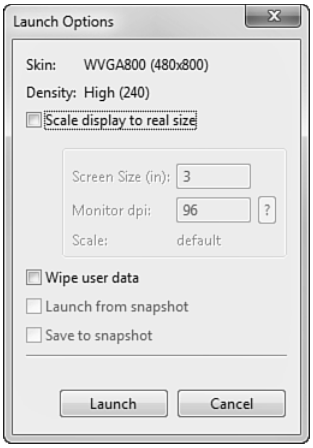
List of existing Android Virtual Devices located at C:\Users\jonathan\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ MyAndroid	Android 4.0.3	4.0.3	15	ARM (armeabi-v7a)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✕ An Android Virtual Device that failed to load. Click 'Details' to see the error.

FIGURE 3.3
A new Android Virtual Device has been added.



Skin: WVGA800 (480x800)
Density: High (240)
☐ Scale display to real size
Screen Size (in): 3
Monitor dpi: 96 ?
Scale: default
☐ Wipe user data
☐ Launch from snapshot
☐ Save to snapshot

FIGURE 3.4
Preparing to launch the emulator.

The emulator device is shown in Figure 3.5, running Android OS 4.0. It may take a few minutes for the emulator to bring up the home screen shown here. The emulator must install the OS and then run it. Because this is rather time consuming, you will want to keep the emulator open while writing Android code so it's available anytime you build and run your code.

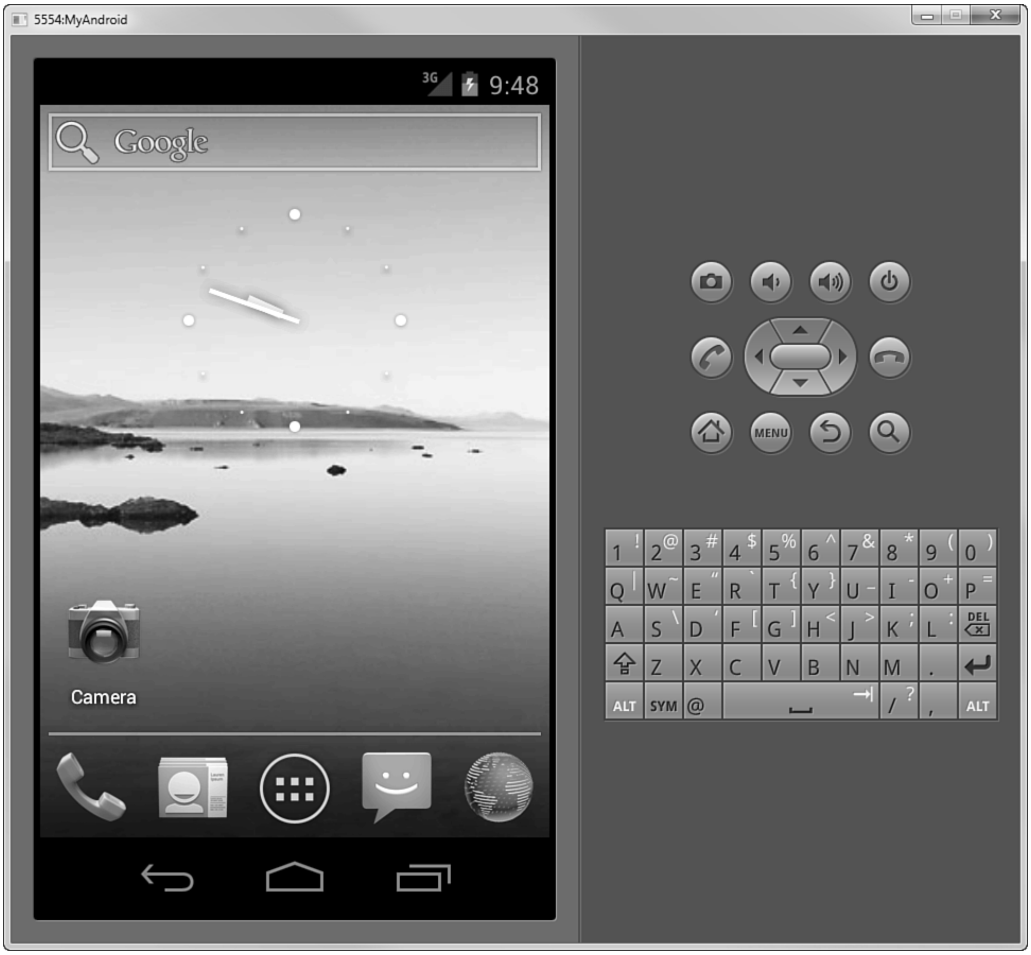


FIGURE 3.5
The Android OS 4.0 emulator is running.

Plugging Android SDK into NetBeans

Although the Android SDK has been installed, NetBeans doesn't automatically know about it, so we have to configure NetBeans to recognize Android projects. This is done with a special plug-in. We'll go over the configuration step by step with plenty of screenshots so you can refer to this hour if needed.

The plug-in has to be downloaded from within NetBeans and is available from a file repository at kenai.com. The plug-in is called NBAndroid, which is short for "NetBeans Android."

First, open the Tools menu in NetBeans, as shown in Figure 3.6, and choose the Plug-ins menu option.

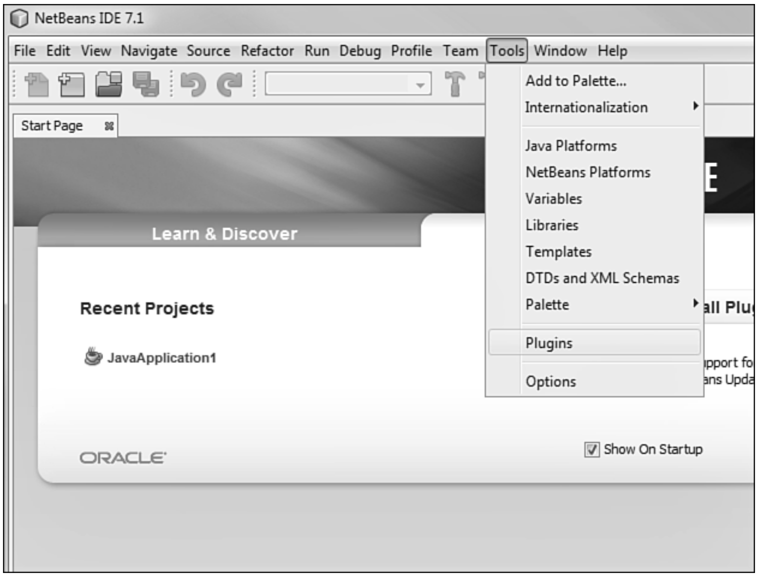


FIGURE 3.6
Invoking the Plug-ins dialog using the Tools menu.

If this is a new install of NetBeans, you likely will not have any additional plug-ins installed yet (as expected). The Plug-ins dialog is shown in Figure 3.7. This first tab shows only updates and is normally empty.

Open the Settings tab, shown in Figure 3.8. Three update centers will be listed (or more, if you are using a more recent version than NetBeans 7.1). The options are not important, but just for reference: Certified Plug-ins, NetBeans Distribution, and Plug-in Portal. We will be adding our own new plug-in source.

On the right side is a button labeled Add. Use this button to bring up the Update Center Customizer dialog (see Figure 3.9). This dialog has two fields where you can specify a new source for plug-ins.

In the Name field, enter **kenai.com**. In the URL field, enter this URL: **http://kenai.com/downloads/nbandroid/updatecenter/updates.xml**.

Click the OK button to proceed.

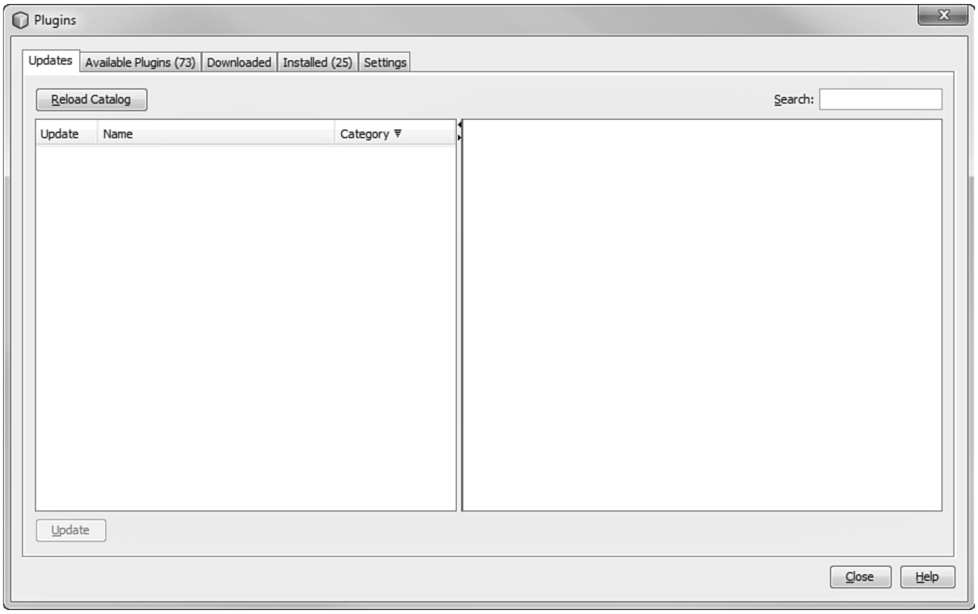
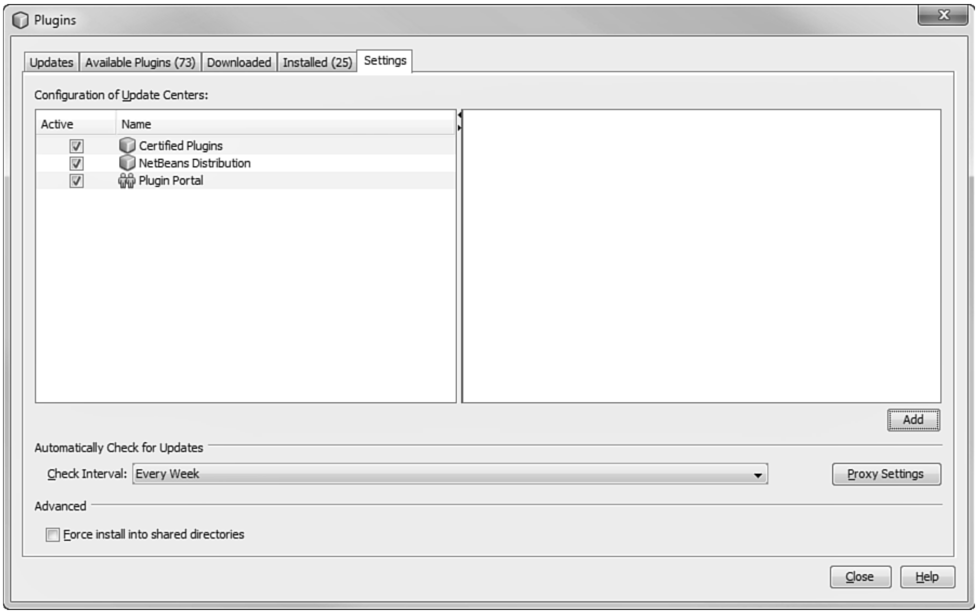


FIGURE 3.7
The Plug-ins dialog has several tabs.



Configuration of Update Centers:

Active	Name
✓	Certified Plugins
✓	NetBeans Distribution
✓	Plugin Portal

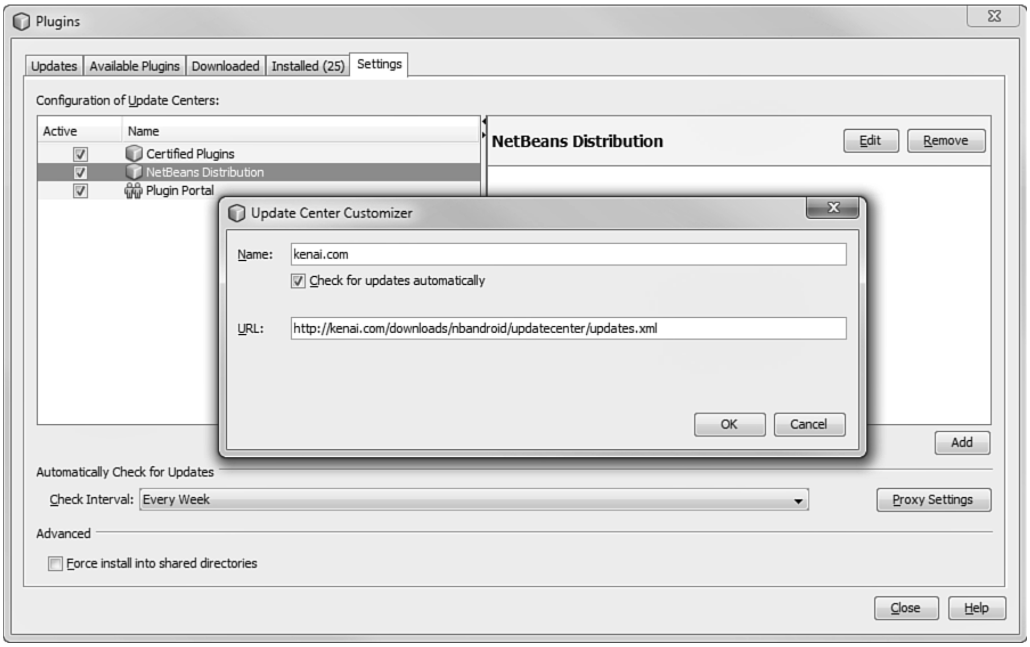
Automatically Check for Updates

Check Interval: Every Week

Advanced

☐ Force install into shared directories

FIGURE 3.8
Viewing the list of plug-in sources.



Configuration of Update Centers:

Active	Name
☑	Certified Plugins
☑	NetBeans Distribution
☑	Plugin Portal

NetBeans Distribution

Update Center Customizer

Name: kenai.com

☑ Check for updates automatically

URL: http://kenai.com/downloads/nbandroid/updatecenter/updates.xml

Automatically Check for Updates

Check Interval: Every Week

Advanced

☐ Force install into shared directories

FIGURE 3.9
Adding a new plug-in source (kenai.com).

BY THE WAY

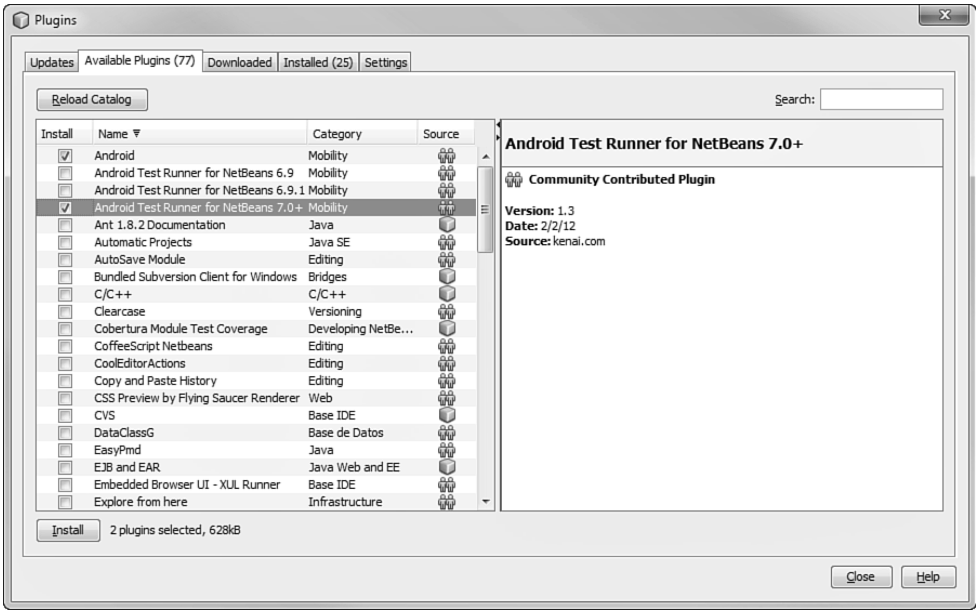
Remember that URLs tend to change without notice! Your best friend is a search engine: Try searching Google for "netbeans android sdk" and you should find the latest tools and plug-ins.

NetBeans then parses the URL specified for any available NetBeans plug-ins. Nothing more will come up—just switch over to the Available Plug-ins tab. The Android plug-ins appear at the top of the list (see Figure 3.10). If the list is not sorted alphabetically, click the Name field heading to sort by Name.

The only plug-in really needed is Android. Two have been selected in Figure 3.11, but the Android Test Runner plug-in is not essential—usually it's for testing larger applications. You may skip it if you like.

Check the Android plug-in and then click the Install button at the bottom left.

A confirmation window will come up showing all the plug-ins you have selected to install. Click Next.



Plugins

Updates | Available Plugins (77) | Downloaded | Installed (25) | Settings

Reload Catalog

Search:

Install	Name	Category	Source
☑	Android	Mobility	
☐	Android Test Runner for NetBeans 6.9	Mobility	
☐	Android Test Runner for NetBeans 6.9.1	Mobility	
☑	Android Test Runner for NetBeans 7.0+	Mobility	
☐	Ant 1.8.2 Documentation	Java	
☐	Automatic Projects	Java SE	
☐	AutoSave Module	Editing	
☐	Bundled Subversion Client for Windows	Bridges	
☐	C/C++	C/C++	
☐	Clearcase	Versioning	
☐	Cobertura Module Test Coverage	Developing NetBe...	
☐	CoffeeScript Netbeans	Editing	
☐	CoolEditorActions	Editing	
☐	Copy and Paste History	Editing	
☐	CSS Preview by Flying Saucer Renderer	Web	
☐	CVS	Base IDE	
☐	DataClassG	Base de Datos	
☐	EasyPmd	Java	
☐	EJB and EAR	Java Web and EE	
☐	Embedded Browser UI - XUL Runner	Base IDE	
☐	Explore from here	Infrastructure	

Android Test Runner for NetBeans 7.0+

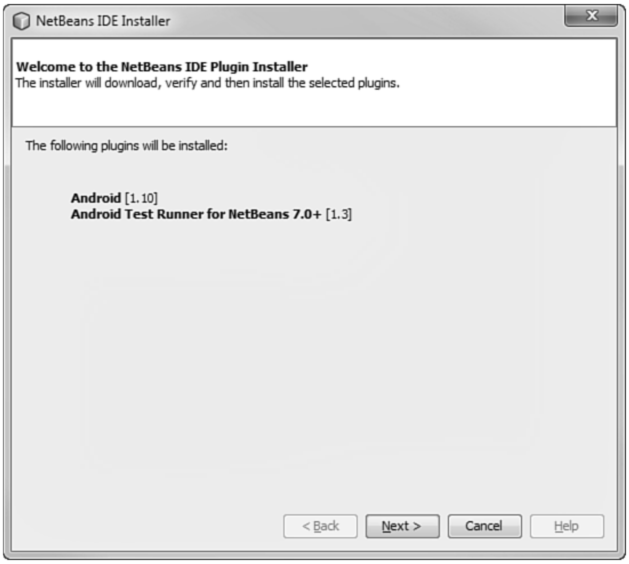
Community Contributed Plugin

Version: 1.3
Date: 2/2/12
Source: kenai.com

Install 2 plugins selected, 628kB

Close Help

FIGURE 3.10
The list of Available Plug-ins (from all sources).



NetBeans IDE Installer

Welcome to the NetBeans IDE Plugin Installer
The installer will download, verify and then install the selected plugins.

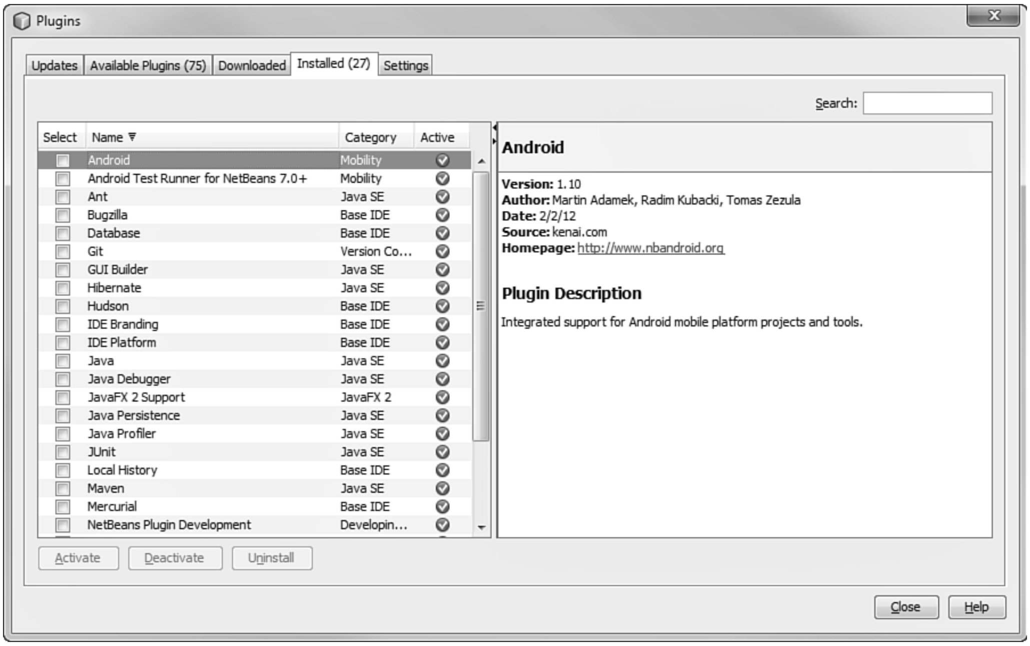
The following plugins will be installed:

Android [1.10]
Android Test Runner for NetBeans 7.0+ [1.3]

< Back Next > Cancel Help

FIGURE 3.11
Preparing to install the Android plug-in for NetBeans.

The new NBAndroid plug-in will be installed. When complete, go to the Installed tab to verify the installation of the new plug-in. See Figure 3.12.



Select	Name	Category	Active
☐	Android	Mobility	✓
☐	Android Test Runner for NetBeans 7.0+	Mobility	✓
☐	Ant	Java SE	✓
☐	Bugzilla	Base IDE	✓
☐	Database	Base IDE	✓
☐	Git	Version Co...	✓
☐	GUI Builder	Java SE	✓
☐	Hibernate	Java SE	✓
☐	Hudson	Base IDE	✓
☐	IDE Branding	Base IDE	✓
☐	IDE Platform	Base IDE	✓
☐	Java	Java SE	✓
☐	Java Debugger	Java SE	✓
☐	JavaFX 2 Support	JavaFX 2	✓
☐	Java Persistence	Java SE	✓
☐	Java Profiler	Java SE	✓
☐	JUnit	Java SE	✓
☐	Local History	Base IDE	✓
☐	Maven	Java SE	✓
☐	Mercurial	Base IDE	✓
☐	NetBeans Plugin Development	Developin...	✓

Android

Version: 1.10
Author: Martin Adamek, Radim Kubacki, Tomas Zezula
Date: 2/2/12
Source: kenai.com
Homepage: http://www.nbandroid.org

Plugin Description

Integrated support for Android mobile platform projects and tools.

FIGURE 3.12
The Android plug-in now appears in the Installed list.

Adding Android SDK Support to Eclipse

The Android SDK plugs into Eclipse a little easier than it does with NetBeans because only one install is required (and no separate plug-in like *NBAndroid* is needed). In the previous hour is a tutorial on installing the Android Development Kit and the Eclipse plug-in, so you may want to refer to Hour 2 if you haven't yet installed these packages. Assuming you have them installed, Eclipse is ready to go. In that case, the title of this section is a misnomer because the Android SDK does not need to be added—it's already good to go. Let's take a look.

Creating a New Android Project in Eclipse

If you finished installing the files in the previous hour, verify the install in Eclipse by opening the Window menu, shown in Figure 3.13. You should see Android SDK Manager and AVD Manager to verify that Eclipse recognizes the new Android packages.

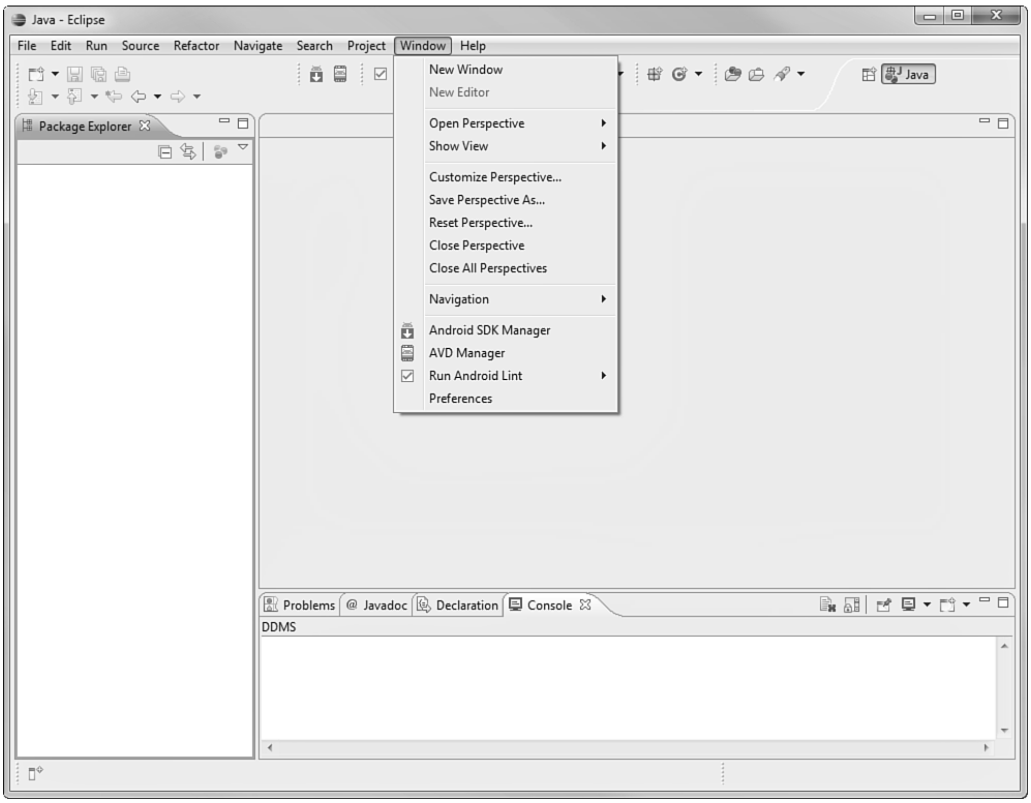


FIGURE 3.13
The Window menu in Eclipse shows the Android SDK tools.

Now, open the File menu and choose New, Project. You should see a new Android group, as shown in Figure 3.14. Choose Android Project from the options shown and click Next.

The New Android Project dialog appears next, as shown in Figure 3.15. Enter a name for the project and choose either the default location or enter a new location for the project files.

The next dialog, shown in Figure 3.16, allows you to choose the Android SDK target (because multiple Android SDK versions may be installed to support various OS release levels). In the example shown, Android 4.0.3 was automatically checked. If you have more than one SDK installed, you may choose from among them.

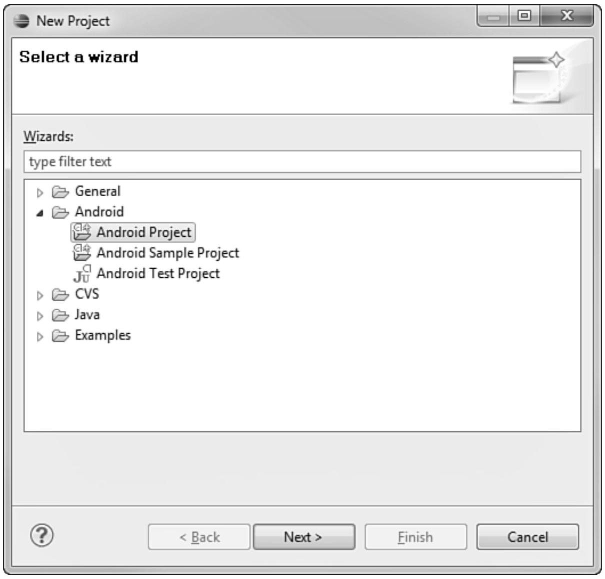
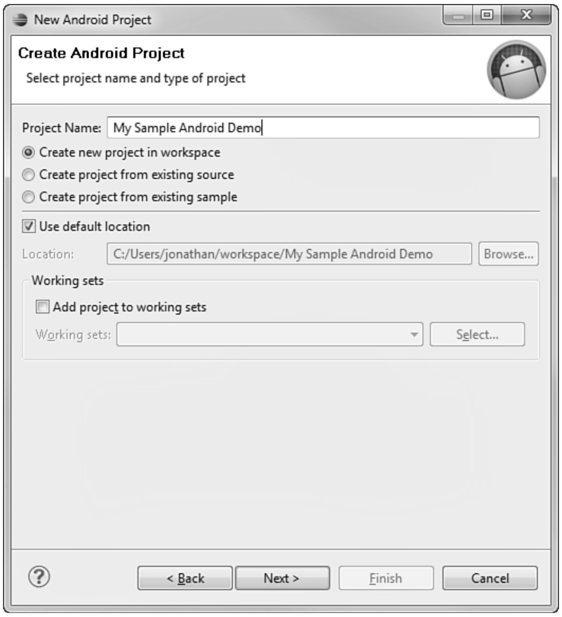
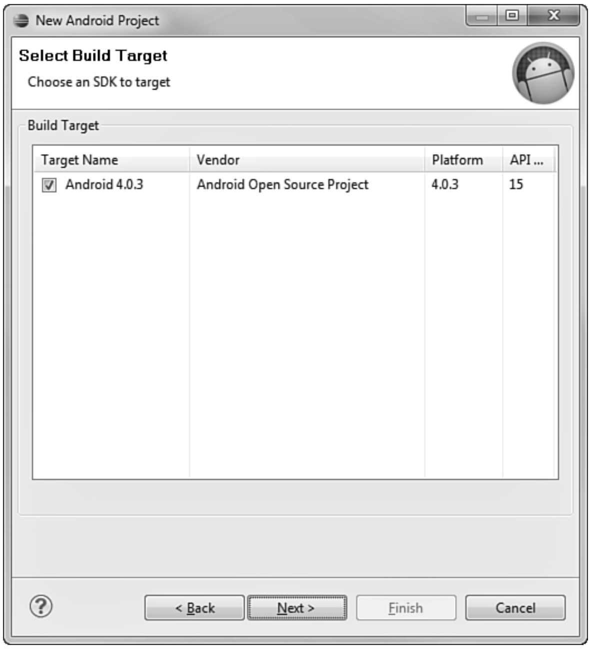


FIGURE 3.14
Creating a new Android project in Eclipse using the New Project dialog.



Project Name: My Sample Android Demo
(●) Create new project in workspace
( ) Create project from existing source
( ) Create project from existing sample
[✓] Use default location
Location: C:/Users/jonathan/workspace/My Sample Android Demo
Working sets
[ ] Add project to working sets
Working sets:

FIGURE 3.15
Entering the new project details.


New Android Project

Select Build Target

Choose an SDK to target

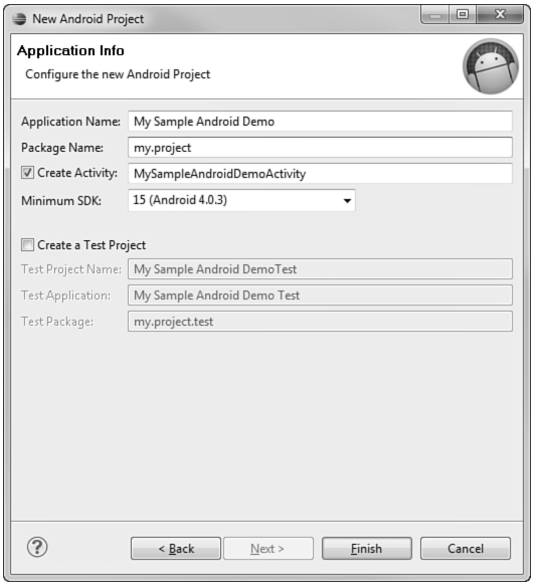
Build Target

Target Name	Vendor	Platform	API ...
☑ Android 4.0.3	Android Open Source Project	4.0.3	15

FIGURE 3.16
Verifying the Build Target for the new project.

The next dialog that comes up in the New Android Project Wizard, shown in Figure 3.17, will look familiar because you dealt with this information earlier in the NetBeans project: the Package Name and Activity. These will make a little more sense in the next hour when you see the names in the source code. For now, you may change the values as needed. Because this is only a configuration tutorial and you aren't writing any real Android code just yet, the values are not that important. But, as was the case with NetBeans, you must enter at least two words separated by a period into the Package Name field.

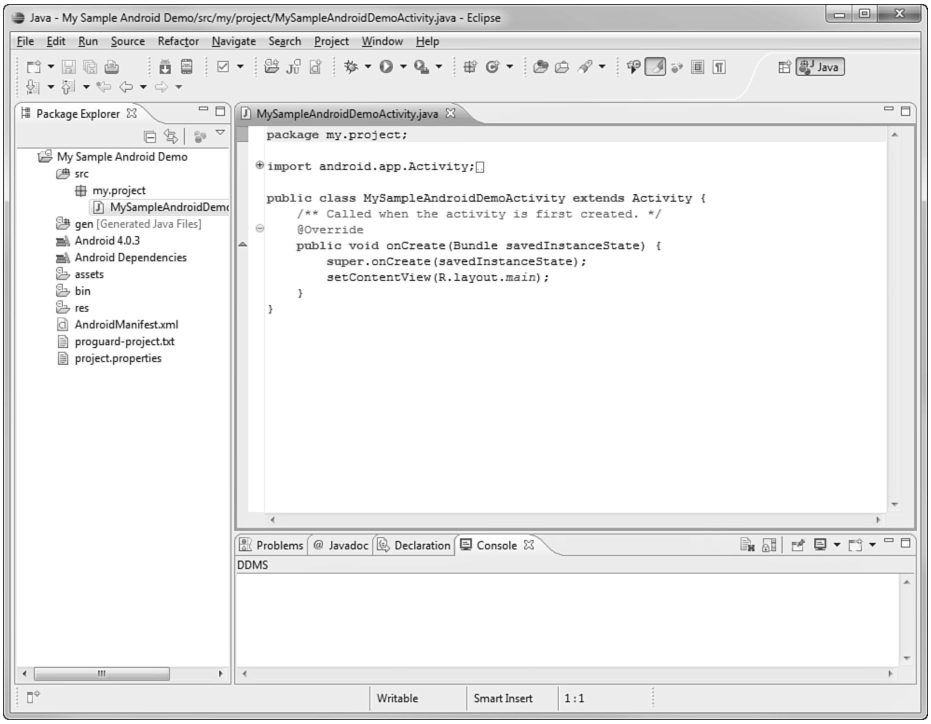
There are a *lot* of files created for a new project. Take a look at Figure 3.18, which shows the newly created project. In Package Explorer (on the left side of the IDE) you will see a folder called `src`, and then `my.project` (the package name), which contains the source code file called `MySampleAndroidDemoActivity.java`. This is similar to the files in the NetBeans project.



Application Name: My Sample Android Demo
Package Name: my.project
[✓] Create Activity: MySampleAndroidDemoActivity
Minimum SDK: 15 (Android 4.0.3)

[ ] Create a Test Project
Test Project Name: My Sample Android DemoTest
Test Application: My Sample Android Demo Test
Test Package: my.project.test

FIGURE 3.17
Entering the Application Info fields.

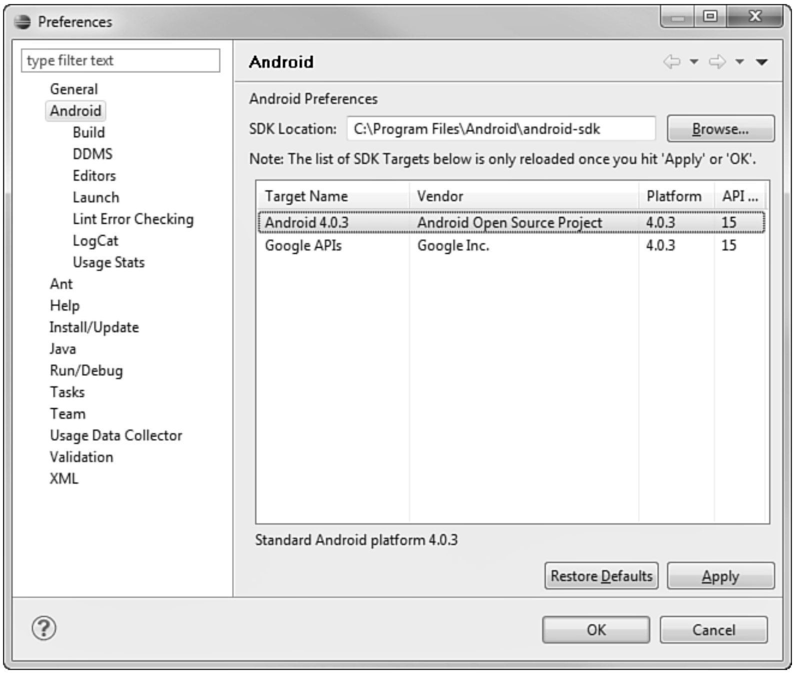

Java - My Sample Android Demo/src/my/project/MySampleAndroidDemoActivity.java - Eclipse
File Edit Run Source Refactor Navigate Search Project Window Help
Java
Package Explorer
My Sample Android Demo
src
my.project
MySampleAndroidDem
gen [Generated Java Files]
Android 4.0.3
Android Dependencies
assets
bin
res
AndroidManifest.xml
proguard-project.txt
project.properties
MySampleAndroidDemoActivity.java

```
package my.project;

import android.app.Activity;

public class MySampleAndroidDemoActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

FIGURE 3.18
The new Android project has been created.

Choosing an Android Build Target

To build and run an Android project in Eclipse, open up the Window menu and choose Preferences. This brings up a dialog called Preferences, shown in Figure 3.19. In the list of preference groups, choose Android to show the Android preferences. Use the Browse button to choose the Android SDK location. This may be in `C:\Program Files\Android`, or it may be in My Documents, or elsewhere—it depends on where you chose to install the SDK according to the steps. Next, choose the target from the list (Android 4.0.3 in this case).



Android

Android Preferences

SDK Location: C:\Program Files\Android\android-sdk

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

Target Name	Vendor	Platform	API ...
Android 4.0.3	Android Open Source Project	4.0.3	15
Google APIs	Google Inc.	4.0.3	15

Standard Android platform 4.0.3

FIGURE 3.19
Setting the Android SDK location and choosing the Android build target.

Summary

This hour covered the additional steps needed to get started programming with the Android SDK using both NetBeans and Eclipse. By now you will have created an emulator device and installed the Android plug-ins for NetBeans and Eclipse and are ready to begin writing code! You write your first real Android project in the next hour.

Q&A

Q. How do you think Java compares to other languages frequently used for game programming, such as C++ and C#?

A. Answers will vary.

Q. If the Android SDK is the library for making apps and games on the Android platform, how does it compare with the DirectX SDK for Windows? You may need to search online for information in order to discuss this topic.

A. Answers will vary.

Workshop

Quiz

1. What is the technical name for the Android emulator?
2. Which version of the Android OS does the emulator support?
3. Which IDE uses the NBAndroid plug-in?

Answers

1. Android Virtual Device (AVD)
2. All versions (that have been installed).
3. NetBeans

Activities

- The Android SDK includes libraries written in Java that interface with a lower-level interface written in C++. It is possible to write C++ code and compile it to run on Android, with Java as a bridge. What is this C++ library called, and how does it work? You may need to do a cursory search online for "android C++ library."

Index

Symbols

/-Enter (Javadoc comments), 103-104**

2D from 3D coordinates, gravity sensors, converting, 183

3D rendering, Android NDK support, 8

3D to 2D coordinates, gravity sensors, converting, 183

A

ABD (Android Debug Bridge)
- installing, 65-68
- running code, 69-71
- *versus* USB device driver, 65-68

AC3 (FFmpeg) audio format, 217

Accelerometer Demo, 164-167

accelerometer sensors, 157-158, 193, 209
- accelMotion variable, 162
- disabling screen orientation changes, 159-160
- initializing, 160-161
- *versus* linear acceleration, 169
- movement of, 161-162

Activity class
- base application class, 78
- methods, 78
- overridable, 78
- setTitle(), 78

addAnimation() method, Sprite class, 287

addToGroup() method, Engine class, 324, 392

adjustAlpha() method, Animation class, 282

adjustPosition() method, Animation class, 282

adjustRotation() method, Animation class, 282

adjustScale() method, Animation class, 282

ADT (Android Development Tools) plug-in
- Android "wizard" dialog, 80
- installing, 25-28

AIFF (Apple) signed 16-bit PCM audio format, 217

AlphaAnimation class, 287

alpha channels, GIMP graphic editor, 121-124

Amazon
- digital media industry, 9
- Kindle Fire
  - Android 2.2 Eclair, 69
  - Android 4.0 Ice Cream Sandwich, 69
  - sensors reported, 164

AMR (narrow band) (FFmpeg) audio format, 217

Android 4/Google. *See also* Android OS/devices
- Apps screen, 6
- based on Linux 3.0, 3
- compatibility of games/apps, 6
- hardware requirements, 11-12
- Home button, 4-5
- home screen, 4-5
- *versus* iPhone, 4
- licensing, 4, 6, 10
- market share, 4, 9, 12
- non proprietary, 4
- Plants vs. Zombies, 7
- programming games, 7
- Return button, 4-5
- Search field with voice recognition, 4
- Tasks button, 4-5
- Unity game engine support, 226

Android Development Tools (ADT), 25-28

android.graphics.Bitmap namespace, 112

android.hardware classes
- methods
  - getSensorList(), 163
  - onAccuracyChanged(), 158-159
  - onCreate(), 158-159, 163
  - onSensorChanged(), 158-159
- Sensor, 160, 162, 165
- SensorEvent, 162, 165

SensorEventListener, 158, 162, 165
SensorManager, 158, 162

Android NDK, C++ for components, 8
pixel buffer access, 8
support for 3D rendering and audio, 8

Android OS/devices. *See also* Android 4/Google
ABD (Android Debug Bridge)
installing, 65-68
running code, 69-71
versus USB device driver, 65-68
adult toy, 7
Android Device Settings, options, 66-67
derivative of iOS/Apple, 9
history, 9-10
Linux kernel numbers, 10
release dates, 10
versions and code names, 10
versus iPhone, 4
licensing, 4, 6, 10
logo, 12
Mac OS X or Linux development, 66
market share, 4, 9, 12
non proprietary, 4
quick-to-market release, 10
rooting, 66
tablets, screen definitions, 107-108
versions to program for, 69
XNA Game Studio, Microsoft, similar to Android, 78

Android SDK (Software Development Kit). *See* SDK

Android Virtual Device. *See* AVD

animation and bitmaps, 269
Runnable Animation Demo, 134-140
walking character, 134-140
Bitmap knight [ ], 135
drawBitmap() method, 135
frames, 135
InputStream object, 135
for loops, 135

Animation class/animation systems, 281
classes
AlphaAnimation, 287
constructors, 287
Animation, 281
CirclingBehavior, 360-361
CircularMovement Animation, 292
FenceBehavior, 326-327, 362, 388-395
FrameAnimation, 288-290
ReboundBehavior, 387-388
SpinAnimation, 290-291
ThrobAnimation, 291-292, 324-325
VelocityBehavior, 359-360
WarpBehavior, 325-326, 363-364
WarpRect, 309-311
methods, 282
destroying/removing sprites from groups, 358-359

Animation Demo
frames
arranging, 270
drawing, from strip images, 270-271
source code, 275-278
sprite sheets
for asteroid animation, 273-274
for zombie animation, 275-274

Animation System Demo, 293-297

Apple products. *See also* iOS/Apple; iPad; iPhone; iPod; iTunes
Apple II, 7
Apple Macintosh, 7

Arkanoid, 386

AssetManager class, 117-118

assets
adding, 115-118
AssetManager class, 117-118
converting from one format to another, 115-116
copying, 121, 271
error handling, 119
InputStream object, 118
istream.close() method, 119
linking, 121

atan() and atan2() methods, Math class, 353

Atari devices
Breakout, 386
history, 7

Audacity audio editor, 214-215
formats supported, 217

audio files, with MediaPlayer, 213
adding to .redraw folder, 214-215
Android NDK support, 8
exporting, 217
formats
converting from one to another, 214-216
exporting, 215
performance issues, 215
supported, 214, 217

initializing, 214
playing, 217
R (resource identifiers), 215
audio files, with SoundPool
asset file extensions, 218
Audio Demo Program, 220-221
initializing, 218
loading resources, 218-219
playing, 219
multiple sounds, 218
R (resource identifiers), 219
AudioManager.STREAM_MUSIC parameter, 218
autorotation on screens, 159-160
AVD (Android Virtual Device) emulator
versus Android devices, 63-64
AVD Manager, 31, 33-34
creating, 32-33
limitations, 64
multi-touch input
receiving basic data, 149
receiving/storing values, 150-155
options
CPU/ABI field, 32
RAM size, 64
SD card field, 33
Skin, 106
Target field, 32, 56
WVGA800 display, 33, 64
running, 33-35, 54-59
single-touch input, 144
axis directions
accelerometer sensor, 161, 164
linear acceleration sensor, 169

B

back buffers, 111, 113-115
background scrolling, seamless texture, 371-374
BaseSensor class, 208-209
Basic Graphics Demo, 95-98, 129
beginDrawing() method, Engine class, 262, 365
Bitmap class, 111
alpha channels for transparencies, 121-124
android.graphics.Bitmap namespace, 112
assets
adding, 115-118
AssetManager class, 117-118
copying, 121, 271
error handling, 119
InputStream object, 118
istream.close() method, 119
linking, 121
back buffer, 111, 113-115
background scrolling, 372-374
Bitmap Loading Demo, 120, 125-126, 129-130
Config.ARGB_8888 parameter, 112, 119
file formats, 115
front buffer, 111
methods
BitmapFactory. decodeStream(), 118-119
createBitmap(), 112, 373-374
drawBitmap(), 120
Texture class, 246
BitmapFactory.decodeStream() method, 118-119
Bitmap knight [ ], 135
Bitmap Loading Demo, 120, 125-126, 129-130
bitmaps and animation, 269
Runnable Animation Demo, 134-140
walking character, 134-140
Bitmap knight [ ], 135
drawBitmap() method, 135
frames, 135
InputStream object, 135
for loops, 135
BMP file format, 115
bounding circles (radial) collision detection, 335-336
bounding rectangles (box) collision detection, 333-336
Box2D physics library, 233
box (bounding rectangles) collision detection, 333-336
boxes, 93-94
Breakout, 386
Buffered Graphics Demo, 113-115
buffers, front and back, 111, 113-115

C

C++ and Android NDK
libraries, 8
supplementing Android SDK, 8
support
for Open GL ES 2.0, 8
for Open SL ES 2.0, 8
Canvas class, 89-90
Basic Graphics Demo, 95-98, 129

Create Canvas Demo, source code, 83-85, 90
drawing
bitmaps, 120
canvas, 112
without onDraw() method, 132-133
game engine core, 227, 235
methods
beginDrawing(), 262
drawBitmap(), 120
drawBox(), 93-94
drawCircle(), 87-89, 93
drawColor(), 89
drawLines(), 93
drawRoundRect(), 94-95
drawText(), 99, 101-102
drawTriangle(), 95
getHolder(), 132
lockCanvas(), 132-133, 262
onDraw(), 88-89, 111, 131
invalidate(), 130
Paint.setColor(), 95, 99, 101-102
Paint.setStyle(), 95
Paint.setTextSize(), 99, 101-102
unlockCanvasAndPost(), 132-133, 262
portrait and landscape mode output, 99
Style.FILL, 95
Style.STROKE, 95
SurfaceHolder variable, 132-133
SurfaceView class, 132
View class, 132
Cartesian coordinate systems, 301-302
circles, 87-89, 93
CircularMovementAnimation class, 292
C# language
similar to Java, 78
XNA Game Studio, 78
.class extension, 8
collisioncheck() method, Engine class, 341-342, 368
CollisionDemo, 343-347
collision detection techniques
bounding circles, 335-336
bounding rectangles, 333-336
collision() method, Engine class, 337, 347, 366, 393
Commodore 64, history, 7
compass sensors, 195, 211
Config.ARGB_8888 parameter, 112, 119
Conley, Ron, 258
coordinate systems, 301-304
Cartesian, 301-302
transforming coordinates, 300, 303-305
cos() method, Math class, 350
createBitmap() method, 112
Create Canvas Demo
output, 85
running on devices, 90
source code, 83-85
Cupcake code name, 10

D

Dalvik Debug Monitor, 70
DDMS Perspective, Eclipse IDE, 70
densities for screens
AVD for each device tested, 107
AVD Skin option, 106
dark over light text display, 107
density-independent pixel (dip), 104
DisplayMetrics class, 106
general resolutions/densities, 106
general sizes/resolutions, 105
Screen Resolution Demo, 107-109
digital media industry, 9
DirectX SDK, 20
Documents app, 10
Donut code name, 10
double buffering, 111
drawBitmap() method, 120, 135
limitations, 308
Matrix class, 299, 305
drawBox() method, Canvas class, 93-94
drawCircle() method, Canvas class, 87-89, 93
drawColor() method
Canvas class, 89
Engine class, 365
draw(delta) method, Engine class, 228
drawing
bitmaps, 120
canvas, 112
without onDraw() method, 132-133
drawLines() method, 93
draw() method
Engine class, 262, 365, 393
Sprite class, 262, 290
drawRoundRect() method, 94-95
drawSheetFrame() method, Sprite or Texture classes, 273

drawStripFrame() method, Sprite class, 271
drawText() method, 99, 101-102
  portrait and landscape mode output, 99
drawTriangle() method, 95
DrawView class
  game engine core, 227
  game engine rendering, 228
  graphics, 88-89
  threaded game loops, 130-131

E

Earth app, 10
Eclair code name, 10
  Amazon Kindle Fire, 69
Eclipse IDE, 3, 8
  .APK files (Android Packages), 233
  Classic version, 25
  DDMS Perspective, 70
  downloading, 25
    versions available, 25
  Helios Service Release, 16, 25
  installing
    ADT plug-in, 25-28
    with JDK, 16
  for Java Developers version, 25
versus NetBeans, 9
  official IDE, 25
  Package Explorer, Assets folder, 116-118, 271
  preferred for Android development, 16
  programming games, 7
  projects
    Android project wizard, 230
    Android "wizard" dialog, 80
    Application Info dialog, 231
    copying/pasting, 269
    Java complier requirements, 140
    Javadoc for self-documented code, 103-104
    new projects, 230
    new project target, 69-70, 231
    Package Name field, 231-232
    perspectives, Java, 149-150
    properties, 233, 263
    references, 263
  versus NetBeans IDE, 80
  and SDK, 40
    build target, 45
    NBAndroid plug-in, configuring, 49
    new projects, 40-43
endDrawing() method, Engine class, 262, 365
Engine class
  conditional test with alive property, 364-365
  methods
    addToGroup() method, 324, 392
    beginDrawing(), 262, 365
    collision(), 337, 347, 366, 393
    collisioncheck(), 341-342, 368
    draw(), 262, 365, 393
    drawColor(), 365
    draw(delta), 228
    endDrawing(), 262, 365
    getGroup(), 368
    load(), 392
    removeFromGroup(), 324
    run(), 338-341, 365-367
    toString(), 368
    update(), 393
    update(delta), 228
entity grouping, 322-324
  LinkedList object, 321
    adding properties, 322
    initializing, 322
Entity Grouping Demo, 327-331

F

Feldman, Ari, 270
FenceBehavior class, 326-327, 362, 388-395
FLAC audio format, 217
Float2 or Float3 classes, 182-183
Float3 p_data variable, 193
for loops, 135
FrameAnimation class, 288-290
frames, 135
front buffers, 111
Froya code name, 10

G

game developers
  compatibility warning, 6
  porting to other platforms, 6

game engines
- components, 227
- design goals, 226-227
- engine core component, 227
  - Engine class, 235-243
  - TextPrinter class, 244-246
  - Texture class, 246-247
  - Timer class, 243-244
- Engine Test Demo Project
  - creating, 247-248
  - logging demo, 252-253
  - source code, 249-251
- Game Engine Library project, 229-233, 288
  - android.engine.VectorMath class, 229
  - .APK file (Android Package), 233
  - creating, 230-232
  - DotProduct() method, 229
  - VectorMath class, 229
- main thread component, 228
- rendering component, 228
- startup component, 227-228
- Unity, 226

game examples
- Ball and Paddle, 385
  - automated ball movement, 386-388
  - automated paddle restriction, 388-390
  - source code, 390-394
- Shoot-'Em-Up
  - output, 375-394
  - source code, 374-395

getBitmap() method, Texture class, 246

getBounds() method, Sprite class, 342-343, 368

getBoundsScaled() method, Sprite class, 342-343

getCanvas() method, Sprite class, 262

getCollidable() method, Sprite class, 342-343

getCollided() method, Sprite class, 342-343

getGroup() method, Engine class, 368

getHolder() method, Canvas class, 132, 228

getIdentifier() method, Sprite class, 342-343

getName() method, Sprite class, 342-343

getOffender() method, Sprite class, 342-343

getSensorList() method, SensorManager class, 163

GIF file format, 115

GIMP graphic editor
- alpha channels for transparencies, 121-124
- converting assets from one format to another, 115-116

Gingerbread code name, 10

Google Drive app, 10

GPS location service versus sensors, 158

Graphics Demo project code, 86-87
- Canvas class, 89-90
  - drawColor() method, 89
  - graphics shapes, 90
  - onDraw() method, 88-89
- DrawView class, 88-89
- MainActivity class, 88-89
- package and import statements, 88
- Paint class, 89
- View class, 90

gravity sensors, 194
- algorithm, 181
- constants/values, 182
- converting 3D to 2D coordinate, 183
- Float2 or Float3 classes, 182-183
- initializing, 181
- onSensorChanged() method, 182
- reading, 182
- testing, 183-188

GSM 6.10 WAV (mobile) audio format, 217

gyroscope sensors, 194-195, 210

H

H15 Game Engine Library, 248

H16 Game Engine Library, 263

H16 Sprite Demo, 262

H17 Game Engine Library, 270

H19 Game Engine Library, 322

H23 Game Engine Library, 367, 375

H23 Velocity Scrolling Demo, 375

hand-held video game systems, 7

Helios Service Release, Eclipse IDE, 16, 25

Honeycomb code name, 10, 69

I

IBM PCs, 7

Ice Cream Sandwich code name, 3-4, 10, 69
- Amazon Kindle Fire, 69

identity matrix, 300

IDEs (integrated development environments), 16
init() method, 252
InputStream object, 118, 135
int change constructor, 287
int maxAlpha constructor, 287
int() method, 390
int minAlpha constructor, 287
invalidate() method, 130
iOS/Apple, Android as derivative of, 9
iOS/Apple versus Android and Windows Phone
- hardware control, 3
- licensing, 4-6
- market share, 4, 9, 12

iPad
- adult toy, 7
- *versus* Android, 9
- development of, 9

iPhone
- adult toy, 7
- *versus* Android 4, 4
- development from iPod, 9
- and Palm Pilot, 9
- Plants vs. Zombies, 7
- and Pocket PC, 9
- release in 2007, 9
- Unity game engine support, 226

iPod
- *versus* Android, 9
- iPhone development, 9

istream.close() method, 119
iTunes, development of, 9

J

JAR (Java Archive) utility, 269
Java
- compiler requirements, 140
- importance of experience, 8
- JAR (Java Archive) utility, 269
- modulus operator, 273
- new classes, 288-289
- programming games, 7
- similar to C# language, 78

Java Development Kit. *See* JDK
Javadoc for self-documented code, 103-104
.java extension, 8
Java Runtime Environment. *See* JRE
JDK (Java Development Kit), 8
- Enterprise Edition, 16
- installing, with Eclipse, 16
- Java Standard Edition 7, 16
- NetBeans
  - downloading, 17
  - installing, 16-20
  - installing, default locations, 19-20
  - license agreement, 18-20
  - plug-in for, 16
- versions available, 17

Jelly Bean code name, 3, 10, 69
JPEG file format, 115
JRE (Java Runtime Environment), 8, 16

K

Katz, Phil, 233
Kindle Fire, Amazon
- Android 2.2 Eclair, 69
- Android 4.0 Ice Cream Sandwich, 69
- sensors reported, 164

L

landscape orientation, 99, 159-160
licensing
- Android OS/devices, 4, 6, 10
- iOS/Apple, 4-6

lifetimes for programs (activities), 79
- foreground, 79
- visible, 79

light detector sensor, 195
Linear Acceleration Demo, 171-177
linear acceleration sensors, 193, 210
- *versus* accelerometer, 169
- initializing sensor, 170-171
- methods
  - onPause(), 171
  - onResume(), 171
  - registerListener(), 171
  - unregisterListener(), 171
- reading sensor, 171
- velocity, 170
- X and Y values, 169

lines, 93
LinkedList object, 321, 392
- adding properties, 322
- initializing, 322

Linux
- and Android development, 8, 66
- basis for Android 4, 3
- versus iOS and Windows Phone OS, licensing, 4

load() method, Engine class, 252, 392
lockCanvas() method, 132-133, 262
Log statement, 252

M

M4A (AAC) (FFmpeg) audio format, 217
Mac OS X
  and Android development, 66
    4.8 or later, 8
  Plants vs. Zombies, 7
magnetic field (compass), 157-158
MainActivity class, 88-89
main() function, MAC OS X, versus Activity class, 77
Maps app, 10
market share
  Android OS devices, 4, 9, 12
  iOS/Apple, 4, 9, 12
  Windows Phone, 4, 9
Math class methods
  atan() and atan2()s, 353
  cos(), 350
  sin(), 350
  toDegrees(), 305, 350
  toRadians(), 305, 350
Matrix class, 300-301
  values stored, 303-304
matrix rotation
  radians versus degrees, 305
  rotation values, 305
    X, Y, and Z components, 306
  transformations with Sprite class, 307
    combined with scaling and translation, 307-308
matrix scaling
  methods, 306-307
  scale values, 307
  transformations with Sprite class, 307
    combined with rotation and translation, 307-308
Matrix Transforms Demo
  getting screen resolution, 309
  rendering frames to scratch bitmaps, 308
  Sprite class
    transforming rotation, scaling, and translation, 307-308
    updated, 311-316
  warping behavior, 309-311
matrix translation
  coordinate systems, 301-304
  identity matrix, 300
  transformations with Sprite class, 307
    combined with rotation and scaling, 307-308
  transforming coordinates, 300, 303-305
  zero matrix, 300
MediaPlayer audio files, 213
  adding to .redraw folder, 214-215
  Android NDK support, 8
  exporting, 217
  formats
    converting from one to another, 214-216
    exporting, 215
    performance issues, 215
    supported, 214, 217
  initializing, 214
  playing, 217
  R (resource identifiers), 215
MediaPlayer class
  audio files, 213
  initializing, 214
method comments, Javadoc, 103-104
Microsoft, XNA Game Studio, similar to Android, 78
MIDI file format, 214
modulus operator, Java, 273
MotionEvent parameter, 151
  multi-touch, 151
    getX() and getY() methods, 149
  single- and multi-touch, get. PointerCount() method, 149
  single-touch, 144-147
MOVE event, 144
MP2 audio format, 217
MP3 audio format, 9, 214, 217
MS-DOS OS, 7
Multi-touch Demo, 150-155
multi-touch input
  methods
    getX() and getY(), 149
    onTouch(), 151
    Point(), 151
  MotionEvent parameter, 151
  Multi-touch Demo, 151-155
  receiving basic data, 149
    and storing values, 150-155
  receiving/storing values, 150-155

N

Napster, 9
NBAndroid plug-in, 36, 38-40
- configuring, 49
- New Android Application dialog, 80-81

NetBeans IDE, 3, 8
- available plug-ins, 38-39
- *versus* Eclipse, 9
- "Hello, Android!" program
  - building, 62
  - editing, 60
  - running, 62
- and JDK
  - downloading, 17
  - installing, 16-20
  - installing, default locations, 19-20
  - installing plug-in, 16
  - license agreement, 18-20
  - plug-in for, 16
- NBAndroid plug-in, 36, 38-40
  - configuring, 49
- new projects, 47-48
  - Output window, 53
  - package naming, 51
  - running project in AVD, 54-59
  - Target Platform table, 51
- Output window, 53, 62
- Package Name field, 51, 82
- programming games, 7
- Project Browser, Source Packages, 82
- projects, 47-48
  - creating, 79-81
  - New Android Application dialog, 80-82
  - running in AVD, 54-59
  - versus Eclipse IDE, 80
- and SDK
  - available plug-ins, 38-39
  - NBAndroid plug-in, 36, 38-40
- Target Platform table, 51, 82
- version 7.1, 16-17

New Android Project Wizard, 43-44
Nintendo Entertainment System (NES), 7
- DSi and Plants vs. Zombies, 7
- Nintendo DS family, 7
- Wii, 7

O

OGG file format, 214
Ogg Vorbis audio format, 217
onAccuracyChanged() method, 158-159
onCreate() method
- Activity class, 78-79, 85
  - game engine startup, 227
- engine test, 252
- SensorManager class, 158-159, 163
- Tricorder class, 197

onDestroy() method, Activity class, 78-79
onDraw() method
- Canvas class, 88-89, 111, 131-132
- Context parameter, 131
- game engine rendering, 228
- invalidate() method, 130

onPause() method
- Activity class, 78, 85-86
- linear acceleration, 171
- Tricorder class, 197

onResume() method
- Activity class, 78, 85-86
- linear acceleration, 171
- Tricorder class, 197

onSensorChanged() method, 158-159
- gravity sensors, 182
- linear acceleration sensors, 171
- pressure sensors, 189

onStart() method, Activity class, 78-79, 85
onStop() method, Activity class, 78-79
OnTouchListener
- game engine core, 227
- single-touch input, 143-148

onTouch() method, 151
- multi-touch, 151
- single-touch, 144-147

Open GL ES 2.0 and Open SL ES 2.0, 8
orientation
- disabling changes, 159-160
- gravity sensors, 182
- landscape and portrait, 159-160
  - drawText() method, 99

OS X (Apple), 7

P

Package Explorer, 43
- bitmap assets, 271
  - File Operation confirmation dialog, 116-117
  - file properties, 117-118

Paint class, 89
Paint.NET graphic editor, 115-116

Paint.setColor() method, 95, 99-102
Paint.setStyle() method, 95
Paint.setTextSize() method, 99-102
Palm Pilot, and iPhone, 9
pause() method, Thread class, 131
PCs, history of use, 7
Picasa app, 10
pixel buffer access, Android NDK, 8
Plants vs. Zombies, 7
PNG file format, 115
Pocket PC, and iPhone, 9
podcasts, 9
Point() points, 151
PopCap Games, Plants vs. Zombies, 7
portrait orientation
Canvas class output, 99
disabling accelerometer changes, 159-160
postRotate() method, Matrix class, 305
postScale() method, Matrix class, 307
postTranslate() method, Matrix class, 304-305
Preferences, Eclipse IDE, Android build target, 45
preRotate() method, Matrix class, 305
preScale() method, Matrix class, 307
pressure sensors, 210
preTranslate() method, Matrix class, 304-305
Prokein, Reiner, free game art, 134
castle images, 116
character sprites
dragon, 257
knight, 257
trees, 255
proximity sensors, 177-178, 193-194, 210
infrared detector, 177
uses, 178

R

radial (bounding circles) collision detection, 335-336
ReboundBehavior class, 387-388
registerListener() method, linear acceleration, 171
removeFromGroup() method, Engine class, 324
resolutions for screens
AVD for each device tested, 107
AVD Skin option, 106
bitmaps, 112
dark over light text display, 107
density-independent pixel (dip), 104
DisplayMetrics class, 106
general screen
resolutions/densities, 106
sizes/resolutions, 105
Screen Resolution Demo, 107-109
resume() method, Thread class, 131
rounded rectangles, 94-95
RTS (real-time strategy) games, 353
run() method
Engine class, 338, 366-367
Runnable class, 130–132
thread updating, 322
Runnable Animation Demo, 134-140
Runnable class, 130
game engine core, 227
run() method, 130-132, 228

S

Samsung Galaxy Nexus, 11
Android hardware, 12
Samsung Galaxy Tab, 11-12
screen autorotation, 159-160
screen densities and resolutions
AVD for each device tested, 107
AVD Skin option, 106
dark over light text display, 107
density-independent pixel (dip), 104
DisplayMetrics class, 106
general resolutions/ densities, 106
general sizes/resolutions, 105
Screen Resolution Demo, 107-109
Screen Resolution Demo, 107-109
SDK (Software Development Kit), 7-8
ADT plug-in, 25-28
Android NDK supplement, 8

Canvas class, 90
downloading, 20-21
Eclipse IDE, 40
build target, 45
NBAndroid plug-in, configuring, 49
new projects, 40-43
history of Android 4, 9
installing, 19-23
default folder, 22
JDK required, 21
NetBeans IDE
available plug-ins, 38-39
NBAndroid plug-in, 36, 38-40
operating systems supported, 8
SDK Manager
additional versions, 24
downloading/installing all packages, 25-26
enabling, 23
Extras, Android Support, 24
running, 23-24
verification dialog, 25
sensors supported, 157-158
system requirements, 8-9

Sega Master System (SMS), 7

self-documented code, 103-104
testing, 104

Sensor class, 160, 162, 165

SensorEvent class, 162, 165, 189
SensorEvent.values array, 191

SensorEventListener class, 162, 165
implements statement, 158, 171
methods
onAccuracyChanged(), 158, 159
onSensorChanged(), 158-159, 171, 189

SensorManager class, 158, 162, 170
gravity constants/values, 182
methods
getSensorList(), 163
onCreate(), 158-159, 163

SensorPanel class, 202-203

sensors
Accelerometer Demo, 164-167
accelerometer sensor, 157-158, 193, 209
accelMotion variable, 162
initializing, 160-161
versus linear acceleration, 169
movement of, 161-162
android.hardware classes, 162
BaseSensor class, 192
compass sensor, 195, 211
GPS location service not sensor, 158
gravity sensor, 194
algorithm, 181
constants/values, 182
converting 3D to 2D coordinate, 183
Float2 or Float3 classes, 182-183
initializing, 181
onSensorChanged() method, 182
reading, 182
testing, 183-188
Gravity Sensor Demo, 183-188
gyroscope sensor, 194-195, 210
light detector sensor, 195
Linear Acceleration Demo, 171-177
linear acceleration sensor, 193, 210
versus accelerometer, 169
initializing sensor, 170-171
onPause() method, 171
onResume() method, 171
reading sensor, 171
registerListener() method, 171
unregisterListener() method, 171
velocity, 170
X and Y values, 169
list of, getting, 163-164
magnetic field (compass), 157-158
pressure sensor, 188, 210
atmospheric pressure levels, 189
initializing, 189
reading, 189
proximity sensor, 177-178, 193-194, 210
infrared detector, 177
uses, 178
screen autorotation, 159-160
Tricorder Demo, 196-211

Sensors class, 205

setBounds() method, Matrix class, 342-343

setCollidable() method, Matrix class, 342-343

setCollided() method, Matrix class, 342-343

setIdentifier() method, Matrix class, 342-343

setName() method, Matrix class, 342-343

setOffender() method, Matrix class, 342-343
setRotate() method, Matrix class, 305, 308
setScale() method, Matrix class, 306, 308
setTitle() method, Activity class, 78
setTranslate() method, Matrix class, 304, 308
single-touch input
  MotionEvent parameter, 144-147
    get.PointerCount() method, 149
  MOVE event, 144
  OnTouchListener, 143-148
  onTouch() method, 144-147
  Single Touch Input Demo, 144-148
  testing on emulator, 144
  UP event, 144
  View class, 143-148
Single Touch Input Demo, 144-148
sin() method, Math class, 350
Sony products, Walkman, 9
Sony PSP family, 7
SoundPool audio files
  asset file extensions, 218
  Audio Demo Program, 220-221
  initializing, 218
  loading resources, 218-219
  playing, 219
    multiple sounds, 218
  R (resource identifiers), 219
SoundPool class
  initializing, 218
  sound effects, 218
SpinAnimation class, 290-291
Sprite class, 258-260
  blueprinting versus evolving, 259
  enhancements, 284-286
  methods
    addAnimation(), 287
    animate(), 283
    draw(), 262
    drawSheetFrame(), 273
    getBounds(), 342-343
    getBoundsScaled(), 342-343
    getCanvas(), 262
    getCollidable(), 342-343
    getCollided(), 342-343
    getIdentifier(), 342-343
    getName(), 342-343
    getOffender(), 342-343
    properties, adding, 342
    setBounds(), 342-343
    setCollidable(), 342-343
    setCollided(), 342-343
    setIdentifier(), 342-343
    setName(), 342-343
    setOffender(), 342-343
  sprites
    character sprites, 257-258
    creating with animation, 287-288
    creating without animation, 283
    drawing requirements, 262
    pointing in direction of movement, 352-354
    prop sprites, 255-256
    transforming rotation, scaling, and translation, 307-308
  subpixel translation support, 355-358
  Texture class, 261
  updated, 311-316
Sprite Demo
  source code, 262-265
  testing Sprite class, 265-266
sprite sheets
  advantages, 270
  animation frames
    creating from sprite sheets, 272-273
    tiling to store as sprite sheets, 270
  for asteroid animation, 273-274
  for zombie animation, 275-274
Sprite Transforms Demo, 316-319
stopwatch() method, Timer class, 243
strokes and fills, changing styles, 95
Style.FILL, 95
Style.STROKE, 95
supported by SDK, 157-158
SurfaceHolder class
  beginDrawing() method, 262
  drawing, 132-133
  game engine rendering, 228
  lockCanvas() method, 262
SurfaceView class
  beginDrawing() method, 262
  game engine core, 227, 235
  threaded game loops, 131-132
System.currentTimeMillis() method, Timer class, 243
system requirements, SDK, 8-9

T

Taito's Arkanoid, 386
Target Platform table, NetBeans IDE, 51
Teach Yourself Windows Phone 7 Game Programming in 24 Hours, 6
TextPrinter class, 244-246
texture atlas. ***See* sprite sheets**
Texture class
- Bitmap object, 246
- core engine classes, 246-247
- drawSheetFrame() method, 273
- getBitmap() method, 246
- Sprite class, 261

TextView widget, 63, 83
threaded game loops
- Context parameter, 131
- methods
  - invalidate(), 130
  - onDraw(), 131-132
- Runnable class, 130
  - run() method, 130-132
- SurfaceView class, 131
- Thread object, 131
  - pause() method, 131
  - resume() method, 131
- Thread object, 131
  - pause() method, 131
  - resume() method, 131

Thread.sleep() method, Timer class, 243
ThrobAnimation class, 291-292, 324-325
Timer class methods, 243-244
- stopwatch(), 243
- System.currentTimeMillis(), 243
- Thread.sleep(), 243

toDegrees() method, Math class, 305, 350
toRadians() method, Math class, 305, 350
Toshiba Thrive 7" tablet, 69
- sensors reported, 163

toString() method, Engine class, 368
touch input
- multi-touch input
  - MotionEvent parameter, 151
  - MotionEvent parameter, getX() and getY() methods, 149
  - Multi-touch Demo, 151-155
  - onTouch() method, 151
  - Point() points, 151
  - receiving basic data, 149
  - receiving/storing values, 150-155
- single-touch input
  - MotionEvent parameter, 144-147
    - get.PointerCount() method, 149
  - MOVE event, 144
  - OnTouchListener, 143-148
  - onTouch() method, 144-147
  - Single Touch Input Demo, 144-148
  - testing on emulator, 144
  - UP event, 144
  - View class, 143-148

transparencies, alpha channels, 121-124
triangles, 95
Tricorder Demo (sensors)
- classes
  - Accelerometer, 209
  - BaseSensor, 208-209
  - CompassSensor, 211
  - GyroscopeSensor, 210
  - LinearAcceleration, 210
  - PressureSensor, 210
  - ProximitySensor, 210
  - SensorPanel, 202-203
  - Sensors, 205
- events, trapping, 205-206
- events, unused accuracy, 207
- helper methods, 201-202
- panels, 199-200
- panels, drawing, 201
- pausing and resuming, 207-208
- printing text lines, 203-204
- updating sensors, 200-201

TYPE sensors
- ACCELEROMETER, 157
- AMBIENT_TEMPERATURE, 157
- GRAVITY, 157, 181, 182
- GYROSCOPE, 157
- LIGHT, 157
- LINEAR_ACCELERATION, 157, 171
- MAGNETIC_FIELD, 157
- PRESSURE, 157
- PROXIMITY, 158, 177
- RELATIVE_HUMIDITY, 158
- ROTATION_VECTOR, 158

U

unlockCanvasAndPost() method, 132-133, 262

unregisterListener() method, linear acceleration, 171

update(delta) method, Engine class, 228

update() method, Engine class, 393

UP event, 144

USB device drivers versus ABD (Android Debug Bridge), 65-68

V

vector shapes, 93
- Basic Graphics Demo, 95-98
- boxes, 93-94
- circles, 87-89, 93
- lines, 93
- rounded rectangles, 94-95
- strokes and fills, changing styles, 95
- triangles, 95

velocity
- angular velocity, 351
- calculating from specific direction, 349
  - radians versus degrees, 350
  - trigonometry functions, 350
- linear acceleration sensors, 170
- pointing sprites to direction of movement, 352-353
  - calculating angles to targets, 353-354
- sine and cosine relationships, 350-351

View class, 90, 132, 143-148

W

WarpBehavior class, 325-326

warping behavior, 309-311

WarpRect class, 309-311

WAV audio format, 214

WAV (Microsoft) signed 16-bit PCM audio format, 217

WEBP file format, 115

Wikipedia, 354

Winamp, 9

Windows Media Player, 9

Windows Mobile OS. *See* Windows Phone

Windows Phone
- adult toy, 7
  - licensing, 4
  - market share, 4, 9

versus Google and Apple
- Plants vs. Zombies, 7

Windows versions, supported by Android SDK
- Vista, 8
- Windows 7, 8
- XP, 8

WinMain() function, Windows versus Activity class, 77

WMA (version 2) (FFmpeg) audio format, 217

WSVGA screen display, 107

WXGA800 screen display, 107

X

Xbox 360, Plants vs. Zombies, 7

XNA Game Studio, Microsoft, 78

Z

zero matrix, 300

ZIP compression algorithm, 233

ZIP library, 269