

Build Android smartphone and tablet game apps



Beginning Android Games

SECOND EDITION

Mario Zechner | Robert Green

Apress®

Beginning Android Games

Second Edition



Mario Zechner
Robert Green

Apress®

Beginning Android Games, Second Edition

Copyright © 2012 by Mario Zechner and Robert Green

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN 978-1-4302-4677-0

ISBN 978-1-4302-4678-7 (eBook)

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning

Lead Editor: Steve Anglin

Developmental Editor: Tom Welsh

Technical Reviewer: Chad Darby

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Christine Ricketts

Copy Editor: William McManus

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

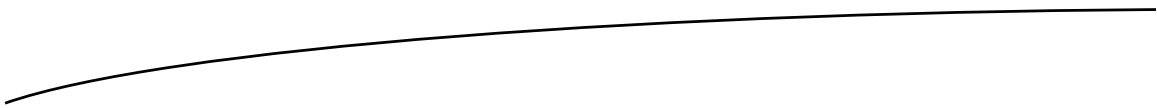
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 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 www.apress.com/source-code.



Dedicated to my idols, Mom and Dad, and my love, Stefanie.

—Mario Zechner

Dedicated to my family and everyone who has helped us along the way.

—Robert Green

Contents at a Glance

About the Authors.....	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
■ Chapter 1: An Android in Every Home	1
■ Chapter 2: First Steps with the Android SDK.....	21
■ Chapter 3: Game Development 101	55
■ Chapter 4: Android for Game Developers	107
■ Chapter 5: An Android Game Development Framework	193
■ Chapter 6: Mr. Nom Invades Android.....	237
■ Chapter 7: OpenGL ES: A Gentle Introduction	275
■ Chapter 8: 2D Game Programming Tricks	355
■ Chapter 9: Super Jumper: A 2D OpenGL ES Game	433
■ Chapter 10: OpenGL ES: Going 3D.....	493
■ Chapter 11: 3D Programming Tricks	529
■ Chapter 12: Android Invaders: The Grand Finale.....	583

■ Chapter 13: Going Native with the NDK	633
■ Chapter 14: Marketing and Monetizing	649
■ Chapter 15: Publishing Your Game	659
■ Chapter 16: What's Next?	675
Index	679



Contents

About the Authors.....	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
 ■ Chapter 1: An Android in Every Home	1
A Brief History of Android	2
Fragmentation	3
The Role of Google	4
The Android Open Source Project.....	4
Google Play	4
Google I/O	5
Android's Features and Architecture	5
The Kernel	7
The Runtime and Dalvik.....	7
System Libraries.....	7
The Application Framework.....	9
The Software Development Kit.....	9
The Developer Community	10

Devices, Devices, Devices!	11
Hardware	11
The Range of Devices	12
Compatibility Across All Devices	17
Mobile Gaming Is Different.....	18
A Gaming Machine in Every Pocket.....	18
Always Connected	19
Casual and Hardcore	19
Big Market, Small Developers	19
Summary.....	20
■ Chapter 2: First Steps with the Android SDK.....	21
Setting Up the Development Environment	21
Setting Up the JDK.....	22
Setting Up the Android SDK	22
Installing Eclipse.....	24
Installing the ADT Eclipse Plug-In.....	25
A Quick Tour of Eclipse	27
Helpful Eclipse Shortcuts.....	29
Creating a New Project in Eclipse and Writing Your Code	30
Creating the Project.....	30
Exploring the Project	31
Making the Application Compatible with All Android Versions.....	33
Writing the Application Code	33
Running the Application on a Device or Emulator	35
Connecting a Device.....	36
Creating an Android Virtual Device	36
Installing Advanced Emulator Features	38
Running an Application.....	42
Debugging and Profiling an Application	46
LogCat and DDMS.....	49
Using ADB	51

Useful Third-Party Tools	53
Summary.....	53
■ Chapter 3: Game Development 101	55
Genres: To Each One's Taste	55
Casual Games.....	56
Puzzle Games	57
Action and Arcade Games.....	59
Tower-Defense Games.....	62
Social Games	63
Beyond the Genres	65
Game Design: The Pen Is Mightier Than the Code.....	66
Core Game Mechanics	66
A Story and an Art Style.....	68
Screens and Transitions	69
Code: The Nitty-Gritty Details	75
Application and Window Management	76
Input	77
File I/O	80
Audio.....	81
Graphics.....	85
The Game Framework	98
Summary.....	105
■ Chapter 4: Android for Game Developers	107
Defining an Android Application: The Manifest File	107
The <manifest> Element.....	109
The <application> Element	109
The <activity> Element	110
The <uses-permission> Element	112
The <uses-feature> Element	114
The <uses-sdk> Element.....	115
Android Game Project Setup in Eight Easy Steps.....	116

Google Play Filters	117
Defining the Icon of Your Game	118
For Those Coming from iOS/Xcode.....	119
Eclipse/ADT vs. Xcode	119
Locating and Configuring Your Targets	120
Other Useful Tidbits	120
Android API Basics	121
Creating a Test Project.....	121
The Activity Life Cycle.....	126
Input Device Handling.....	133
File Handling.....	151
Audio Programming	157
Playing Sound Effects.....	158
Streaming Music	162
Basic Graphics Programming	165
Best Practices	190
Summary.....	191
■ Chapter 5: An Android Game Development Framework	193
Plan of Attack	193
The AndroidFileIO Class.....	194
AndroidAudio, AndroidSound, and AndroidMusic: Crash, Bang, Boom!	195
AndroidInput and AccelerometerHandler	200
AccelerometerHandler: Which Side Is Up?	200
CompassHandler	202
The Pool Class: Because Reuse Is Good for You!	203
KeyboardHandler: Up, Up, Down, Down, Left, Right	205
Touch Handlers.....	209
AndroidInput: The Great Coordinator	217
AndroidGraphics and AndroidPixmap: Double Rainbow	218
Handling Different Screen Sizes and Resolutions	219
AndroidPixmap: Pixels for the People.....	224

AndroidGraphics: Serving Our Drawing Needs	225
AndroidFastRenderView: Loop, Stretch, Loop, Stretch.....	229
AndroidGame: Tying Everything Together	231
Summary.....	236
■ Chapter 6: Mr. Nom Invades Android.....	237
Creating the Assets	237
Setting Up the Project	239
MrNomGame: The Main Activity	240
Assets: A Convenient Asset Store	240
Settings: Keeping Track of User Choices and High Scores	241
LoadingScreen: Fetching the Assets from Disk	244
The Main Menu Screen	245
The HelpScreen Classes.....	249
The High-Scores Screen.....	251
Rendering Numbers: An Excursion	251
Implementing the Screen	253
Abstracting the World of Mr. Nom: Model, View, Controller	255
The Stain Class.....	257
The Snake and SnakePart Classes	258
The World Class	262
The GameScreen Class	267
Summary.....	274
■ Chapter 7: OpenGL ES: A Gentle Introduction	275
What Is OpenGL ES and Why Should I Care?	275
The Programming Model: An Analogy.....	276
Projections.....	278
Normalized Device Space and the Viewport.....	280
Matrices.....	280
The Rendering Pipeline.....	281

Before We Begin	282
GLSurfaceView: Making Things Easy Since 2008.....	283
GLGame: Implementing the Game Interface.....	286
Look Mom, I Got a Red Triangle!.....	293
Defining the Viewport	294
Defining the Projection Matrix	294
Specifying Triangles.....	297
Putting It Together	301
Specifying Per-Vertex Color.....	304
Texture Mapping: Wallpapering Made Easy.....	308
Texture Coordinates.....	308
Uploading Bitmaps	310
Texture Filtering.....	312
Disposing of Textures	313
A Helpful Snippet.....	313
Enabling Texturing	314
Putting It Together	314
A Texture Class	316
Indexed Vertices: Because Re-use Is Good for You	318
Putting It Together	320
A Vertices Class	322
Alpha Blending: I Can See Through You.....	325
More Primitives: Points, Lines, Strips, and Fans.....	328
2D Transformations: Fun with the Model-View Matrix	330
World and Model Space.....	330
Matrices Again.....	331
An Initial Example Using Translation.....	332
More Transformations.....	337
Optimizing for Performance	341
Measuring Frame Rate	341
The Curious Case of the Hero on Android 1.5	343

What's Making My OpenGL ES Rendering So Slow?	343
Removing Unnecessary State Changes	345
Reducing Texture Size Means Fewer Pixels to Be Fetched	347
Reducing Calls to OpenGL ES/JNI Methods	348
The Concept of Binding Vertices	348
In Closing	352
Summary	352
■ Chapter 8: 2D Game Programming Tricks	355
Before We Begin	355
In the Beginning . . . There Was the Vector	356
Working with Vectors	357
A Little Trigonometry	359
Implementing a Vector Class	361
A Simple Usage Example	365
A Little Physics in 2D	369
Newton and Euler, Best Friends Forever	369
Force and Mass	370
Playing Around, Theoretically	371
Playing Around, Practically	372
Collision Detection and Object Representation in 2D	376
Bounding Shapes	377
Constructing Bounding Shapes	378
Game Object Attributes	381
Broad-Phase and Narrow-Phase Collision Detection	382
An Elaborate Example	389
A Camera in 2D	402
The Camera2D Class	405
An Example	407
Texture Atlas: Because Sharing Is Caring	408
An Example	410

Texture Regions, Sprites, and Batches: Hiding OpenGL ES	414
The TextureRegion Class.....	415
The SpriteBatcher Class	416
Sprite Animation.....	425
The Animation Class	426
An Example.....	427
Summary.....	431
■ Chapter 9: Super Jumper: A 2D OpenGL ES Game	433
Core Game Mechanics	433
Developing a Backstory and Choosing an Art Style.....	434
Defining Screens and Transitions.....	435
Defining the Game World.....	436
Creating the Assets	439
The UI Elements.....	439
Handling Text with Bitmap Fonts.....	441
The Game Elements.....	443
Texture Atlas to the Rescue	444
Music and Sound.....	446
Implementing Super Jumper.....	447
The Assets Class.....	447
The Settings Class	450
The Main Activity	451
The Font Class	453
The GLScreen Class.....	454
The Main Menu Screen.....	455
The Help Screens.....	458
The High-Scores Screen.....	460
The Simulation Classes	463
The Game Screen	479
The WorldRenderer Class	486

To Optimize or Not to Optimize.....	490
Summary.....	492
■ Chapter 10: OpenGL ES: Going 3D.....	493
Before We Begin.....	493
Vertices in 3D	494
Vertices3: Storing 3D Positions.....	494
An Example.....	496
Perspective Projection: The Closer, the Bigger.....	500
Z-buffer: Bringing Order to Chaos	502
Fixing the Previous Example.....	503
Blending: There's Nothing Behind You	504
Z-buffer Precision and Z-fighting	507
Defining 3D Meshes	508
A Cube: Hello World in 3D	509
An Example.....	512
Matrices and Transformations, Again	515
The Matrix Stack.....	515
Hierarchical Systems with the Matrix Stack.....	518
A Simple Camera System	524
Summary.....	528
■ Chapter 11: 3D Programming Tricks	529
Before We Begin.....	529
Vectors in 3D	530
Lighting in OpenGL ES.....	534
How Lighting Works.....	535
Light Sources.....	536
Materials.....	537
How OpenGL ES Calculates Lighting: Vertex Normals	538
In Practice.....	539
Some Notes on Lighting in OpenGL ES	553

Mipmapping	553
Simple Cameras	558
The First-Person or Euler Camera	558
An Euler Camera Example	561
A Look-At Camera.....	567
Loading Models	569
The Wavefront OBJ Format.....	570
Implementing an OBJ Loader	571
Using the OBJ Loader	576
Some Notes on Loading Models	576
A Little Physics in 3D.....	576
Collision Detection and Object Representation in 3D	577
Bounding Shapes in 3D	578
Bounding Sphere Overlap Testing.....	578
GameObject3D and DynamicGameObject3D	580
Summary.....	581
■ Chapter 12: Android Invaders: The Grand Finale.....	583
Core Game Mechanics	583
Developing a Backstory and Choosing an Art Style.....	585
Defining Screens and Transitions.....	586
Defining the Game World.....	587
Creating the Assets	589
The UI Assets	589
The Game Assets.....	590
Sound and Music.....	592
Plan of Attack	592
The Assets Class	593
The Settings Class.....	596
The Main Activity	597
The Main Menu Screen	598

The Settings Screen	601
The Simulation Classes	604
The Shield Block Class	604
The Shot Class	605
The Ship Class	605
The Invader Class	607
The World Class	611
The GameScreen Class	617
The WorldRender Class	624
Optimizations	629
Summary.....	631
■ Chapter 13: Going Native with the NDK	633
What Is the Android NDK?	633
The Java Native Interface.....	634
Setting Up the NDK.....	636
Setting Up an NDK Android Project	636
Creating Java Native Methods.....	637
Creating the C/C++ Header and Implementation.....	638
Building the Shared Library.....	640
Putting It All Together	643
Summary.....	648
■ Chapter 14: Marketing and Monetizing	649
Monetizing Your Game.....	649
Advertising.....	650
In-App Products	651
Virtual Currency	652
To Sell or Not to Sell	653
Licensing	653

Getting Your Game Discovered	654
Social Network Integration	654
Discovery Services	654
Blogs and Conventional Web Media	655
Monetizable by Design	655
Discoverable by Design	656
Summary	657
■ Chapter 15: Publishing Your Game	659
A Word on Testing	659
Becoming a Registered Developer	660
Signing Your Game's APK	661
Putting Your Game on Google Play	666
Uploading Assets	668
Product Details	669
Publishing Options	670
APK Files Management	672
Publish!	673
More On The Developer Console	673
Summary	674
■ Chapter 16: What's Next?	675
Location Awareness	675
Multiplayer Functionality	675
OpenGL ES 2.0/3.0 and More	675
Frameworks and Engines	676
Resources on the Web	678
Closing Words	678
Index	679

About the Authors



Mario Zechner has been programming games since age 12. Over the years, he created many prototypes and finished games on various platforms. He's the creator of libgdx, an Android game development framework used by many top-grossing Android games. Before he joined his current employer, Mario was the tech-lead at a mobile gaming startup in San Francisco. Mario's current day job involves working in the fields of information extraction and retrieval, visualization, and machine learning. You can meet Mario at conferences, where he gives talks and workshops on game programming, read his blog at www.badlogicgames.com, or follow him on Twitter @badlogicgames.



Robert Green is the founder of the game studio Battery Powered Games in Portland, Oregon. He has developed over a dozen Android games, including *Deadly Chambers*, *Antigen*, *Wixel*, *Light Racer*, and *Light Racer 3D*. Before diving full time into mobile video game development and publishing, Robert worked for software companies in Minneapolis and Chicago, including IBM Interactive. Robert's current focus is on cross-platform game development and high-performance mobile gaming with BatteryTech SDK. Robert often updates his personal blog with game programming tidbits at www.rbgrn.net and his professional game development software at www.batterypoweredgames.com.

About the Technical Reviewer



Chád Darby is an author, instructor, and speaker in the Java development world. As a recognized authority on Java applications and architectures, he has presented technical sessions at software development conferences worldwide. In his 15 years as a professional software architect, he's had the opportunity to work for Blue Cross/Blue Shield, Merck, Boeing, Northrop Grumman, and a handful of startup companies. Chád is a contributing author to several Java books, including *Professional Java E-Commerce* (Wrox Press), *Beginning Java Networking* (Wrox Press), and *XML and Web Services Unleashed* (Sams Publishing). Chád has Java certifications from Sun Microsystems and IBM. He holds a B.S. in Computer Science from Carnegie Mellon University. You can read Chád's blog at www.luv2code.com and follow him on Twitter @darby1uvs2code.

Acknowledgments

I'd like to thank the excellent team at Apress, who made this book possible. Specifically, I'd like to thank Candace English, Adam Heath, Matthew Moodie, Damon Larson, James Compton, Christine Ricketts, Tom Welsh, Bill McManus, and many more good souls who have worked on this book.

Thanks to Robert Green, for being a great co-author and for living through the hell of early Android versions with me.

Another big thanks to my splendid team of contributors, who carry on the libgdx effort while I'm busy writing this book. I'd especially like to thank my friend Nathan Sweet; the months we spent together in San Francisco and Graz were some of the most memorable times in my life. Ruben Garat and Ariel Coppes from Gemserk, who make sure we stay on target with libgdx. Justin Shapcott, who keeps cleaning up behind my mess in the libgdx repository. Søren Nielsen, one of the most knowledgeable artists I've ever had the pleasure to work with, who's a helping hand in the libgdx IRC channel. And finally, all the folks on the libgdx IRC channel and the forums, whom I consider to be extended family.

Last but not least, thanks to Mom and Dad for being folks to look up to, and to my love, Stefanie, who put up with all the long nights alone in bed, as well as my grumpiness. Lyr!

—Mario Zechner

I'd like to thank Mario for answering so many questions and helping me work through a few tricky bugs over the years. Ryan Foss is an artist and programmer I worked with to build several games and is truly an exceptional person. Scott Lembcke of Howling Moon Software helped me through some difficult programming problems. Zach Wendt, organizer of the Minneapolis IGDA, always hung out late after meetings and talked tech with me. Thanks to Dmitri Salcedo, who's been sticking with me through a tough year and helping to achieve more than we ever have before. Finally, I thank all of the original licensees of BatteryTech SDK, who believed in the product before anyone else. Thank you all for your support and help; I couldn't have made it without you.

—Robert Green



Introduction

Hi there, and welcome to the world of Android game development. You came here to learn about game development on Android, and we hope to be the people who enable you to realize your ideas.

Together we'll cover quite a range of materials and topics: Android basics, audio and graphics programming, a little math and physics, OpenGL ES, an intro to the Android Native Development Kit (NDK), and finally, publishing, marketing, and making money from your game. Based on all this knowledge, we'll develop three different games, one of which is even 3D.

Game programming can be easy if you know what you're doing. Therefore, we've tried to present the material in a way that not only gives you helpful code snippets to reuse, but actually shows you the big picture of game development. Understanding the underlying principles is the key to tackling ever more complex game ideas. You'll not only be able to write games similar to the ones developed over the course of this book, but you'll also be equipped with enough knowledge to go to the Web or the bookstore and take on new areas of game development on your own.

Who This Book Is For

This book is aimed first and foremost at complete beginners in game programming. You don't need any prior knowledge on the subject matter; we'll walk you through all the basics. However, we need to assume a little knowledge on your end about Java. If you feel rusty on the matter, we'd suggest refreshing your memory by reading *Thinking in Java*, by Bruce Eckel (Prentice Hall, 2006), an excellent introductory text on the programming language. Other than that, there are no other requirements. No prior exposure to Android or Eclipse is necessary!

This book is also aimed at intermediate-level game programmers who wants to get their hands dirty with Android. While some of the material may be old news for you, there are still a lot of tips and hints contained that should make reading this book worthwhile. Android is a strange beast at times, and this book should be considered your battle guide.

How This Book Is Structured

This book takes an iterative approach in that we'll slowly but surely work our way from the absolute basics to the esoteric heights of hardware-accelerated game programming goodness. Over the course of the chapters, we'll build up a reusable code base that you can use as the foundation for most types of games.

If you're reading this book purely as a learning exercise, we suggest going through the chapters in sequence starting from Chapter 1. Each chapter builds off of the previous chapter, which makes for a good learning experience.

If you're reading this book with the intent to publish a new game at the end, we highly recommend you skip to Chapter 14 and learn about designing your game to be marketable and make money, then come back to the beginning and begin development.

Of course, more experienced readers can skip certain sections they feel confident with. Just make sure to read through the code listings of sections you skim over, so you will understand how the classes and interfaces are used in subsequent, more advanced sections.

Downloading the Code

This book is fully self-contained; all the code necessary to run the examples and games is included. However, copying the listings from the book to Eclipse is error prone, and games do not consist of code alone, but also have assets that you can't easily copy out of the book. We took great care to ensure that all the listings in this book are error free, but the gremlins are always hard at work.

To make this a smooth ride, we created a Google Code project that offers you the following:

- The complete source code and assets available from the project's Subversion repository. The code is licensed under the Apache License 2.0 and hence is free to use in commercial and noncommercial projects. The assets are licensed under the Creative Commons BY-SA 3.0. You can use and modify them for your commercial projects, but you have to put your assets under the same license!
- A quickstart guide showing you how to import the projects into Eclipse in textual form, and a video demonstration for the same.
- An issue tracker that allows you to report any errors you find, either in the book itself or in the code accompanying the book. Once you file an issue in the issue tracker, we can incorporate any fixes in the Subversion repository. This way, you'll always have an up-to-date, (hopefully) error-free version of this book's code, from which other readers can benefit as well.
- A discussion group that is free for everybody to join and discuss the contents of the book. We'll be on there as well, of course.

For each chapter that contains code, there's an equivalent Eclipse project in the Subversion repository. The projects do not depend on each other, as we'll iteratively improve some of the framework classes over the course of the book. Therefore, each project stands on its own. The code for both Chapters 5 and 6 is contained in the `ch06-mrnom` project.

The Google Code project can be found at <http://code.google.com/p/beginnnginandroidgames2>.

Contacting the Authors

Should you have any questions or comments—or even spot a mistake you think we should know about—you can contact either Mario Zechner, by registering an account and posting at <http://badlogicgames.com/forum/viewforum.php?f=21>, or Robert Green, by visiting www.rbgrn.net/contact.

We prefer being contacted through the forums. That way other readers benefit as well, as they can look up already answered questions or contribute to the discussion!

An Android in Every Home

As kids of the eighties and nineties, we naturally grew up with our trusty Nintendo Game Boys and Sega Game Gears. We spent countless hours helping Mario rescue the princess, getting the highest score in Tetris, and racing our friends in Super RC Pro-Am via Link Cable. We took these awesome pieces of hardware with us everywhere we could. Our passion for games made us want to create our own worlds and share them with our friends. We started programming on the PC, but soon realized that we couldn't transfer our little masterpieces to the available portable game consoles. As we continued being enthusiastic programmers, over time our interest in actually playing video games faded. Besides, our Game Boys eventually broke...

Fast forward to today. Smartphones and tablets have become the new mobile gaming platforms of this era, competing with classic, dedicated handheld systems such as the Nintendo 3DS and the PlayStation Vita. This development renewed our interest, and we started investigating which mobile platforms would be suitable for our development needs. Apple's iOS seemed like a good candidate for our game coding skills. However, we quickly realized that the system was not open, that we'd be able to share our work with others only if Apple allowed it, and that we'd need a Mac in order to develop for the iOS. And then we found Android.

We both immediately fell in love with Android. Its development environment works on all the major platforms—no strings attached. It has a vibrant developer community, happy to help you with any problem you encounter, as well as offering comprehensive documentation. You can share your games with anyone without having to pay a fee to do so, and if you want to monetize your work, you can easily publish your latest and greatest innovation to a global market with millions of users in a matter of minutes.

The only thing left was to figure out how to write games for Android, and how to transfer our PC game development knowledge to this new system. In the following chapters, we want to share our experience with you and get you started with Android game development. Of course, this is partly a selfish plan: we want to have more games to play on the go!

Let's start by getting to know our new friend, Android.

A Brief History of Android

Android was first seen publicly in 2005, when Google acquired a small startup called Android Inc. This fueled speculation that Google was interested in entering the mobile device space. In 2008, the release of version 1.0 of Android put an end to all speculation, and Android went on to become the new challenger on the mobile market. Since then, Android has been battling it out with already-established platforms, such as iOS (then called iPhone OS), BlackBerry OS, and Windows Phone 7. Android's growth has been phenomenal, as it has captured more and more market share every year. While the future of mobile technology is always changing, one thing is certain: Android is here to stay.

Because Android is open source, there is a low barrier of entry for handset manufacturers using the new platform. They can produce devices for all price segments, modifying Android itself to accommodate the processing power of a specific device. Android is therefore not limited to high-end devices, but can also be deployed in low-cost devices, thus reaching a wider audience.

A crucial ingredient for Android's success was the formation of the Open Handset Alliance (OHA) in late 2007. The OHA includes companies such as HTC, Qualcomm, Motorola, and NVIDIA, which all collaborate to develop open standards for mobile devices. Although Android's code is developed primarily by Google, all the OHA members contribute to its source code in one form or another.

Android itself is a mobile operating system and platform based on the Linux kernel versions 2.6 and 3.x, and it is freely available for commercial and noncommercial use. Many members of the OHA build custom versions of Android with modified user interfaces (UIs) for their devices, such as HTC's Sense and Motorola's MOTOBLUR. The open source nature of Android also enables hobbyists to create and distribute their own versions. These are usually called *mods*, *firmware*, or *roms*. The most prominent rom at the time of this writing is developed by Steve Kondik, also known as Cyanogen, and many contributors. It aims to bring the newest and best improvements to all sorts of Android devices and breathe fresh air into otherwise abandoned or old devices.

Since its release in 2008, Android has received many major version updates, all code-named after desserts (with the exception of Android 1.1, which is irrelevant nowadays). Most versions of the Android platform have added new functionality, usually in the form of application programming interfaces (APIs) or new development tools, that is relevant, in one way or another, for game developers:

- *Version 1.5 (Cupcake)*: Added support for including native libraries in Android applications, which were previously restricted to being written in pure Java. Native code can be very beneficial in situations where performance is of utmost concern.
- *Version 1.6 (Donut)*: Introduced support for different screen resolutions. We will revisit that development a couple of times in this book because it has some impact on how we approach writing games for Android.
- *Version 2.0 (Éclair)*: Added support for multitouch screens.
- *Version 2.2 (Froyo)*: Added just-in-time (JIT) compilation to the Dalvik virtual machine (VM), the software that powers all the Java applications on Android. JIT speeds up the execution of Android applications considerably—depending on the scenario, up to a factor of five.

- *Version 2.3 (Gingerbread)*: Added a new concurrent garbage collector to the Dalvik VM.
- *Version 3.0 (Honeycomb)*: Created a tablet version of Android. Introduced in early 2011, Honeycomb contained more significant API changes than any other single Android version released to date. By version 3.1, Honeycomb added extensive support for splitting up and managing a large, high-resolution tablet screen. It added more PC-like features, such as USB host support and support for USB peripherals, including keyboards, mice, and joysticks. The only problem with this release was that it was only targeted at tablets. The small-screen/smartphone version of Android was stuck with 2.3.
- *Android 4.0 (Ice Cream Sandwich [ICS])*: Merged Honeycomb (3.1) and Gingerbread (2.3) into a common set of features that works well on both tablets and phones.
- *Android 4.1 (Jelly Bean)*: Improved the way the UI is composited, and rendering in general. The effort is known as “Project Butter”; the first device to feature Jelly Bean was Google’s own Nexus 7 tablet.

ICS is a huge boost for end users, adding a number of improvements to the Android UI and built-in applications such as the browser, email clients, and photo services. Among other things for developers, ICS merges in Honeycomb UI APIs that bring large-screen features to phones. ICS also merges in Honeycomb’s USB periphery support, which gives manufacturers the option of supporting keyboards and joysticks. As for new APIs, ICS adds a few, such as the Social API, which provides a unified store for contacts, profile data, status updates, and photos. Fortunately for Android game developers, ICS at its core maintains good backward compatibility, ensuring that a properly constructed game will remain well compatible with older versions like Cupcake and Eclair.

Note We are both often asked which new features new versions of Android bring to the table for games. The answer often surprises people: effectively no new game-specific features outside of the native development kit (NDK) have been added to Android since version 2.1. Since that version, Android has included everything you need to build just about any kind of game you want. Most new features are added to the UI API, so just focus on 2.1 and you’ll be good to go.

Fragmentation

The great flexibility of Android comes at a price: companies that opt to develop their own UIs have to play catch-up with the fast pace at which new versions of Android are released. This can lead to handsets no more than a few months old becoming outdated, as carriers and handset manufacturers refuse to create updates that incorporate the improvements of new Android versions. A result of this process is the big bogeyman called *fragmentation*.

Fragmentation has many faces. To the end user, it means being unable to install and use certain applications and features due to being stuck with an old Android version. For developers, it means that some care has to be taken when creating applications that are meant to work on

all versions of Android. While applications written for earlier versions of Android usually run fine on newer ones, the reverse is not true. Some features added to newer Android versions are, of course, not available on older versions, such as multitouch support. Developers are thus forced to create separate code paths for different versions of Android.

In 2011, many prominent Android device manufacturers agreed to support the latest Android OS for a device lifetime of 18 months. This may not seem like a long time, but it's a big step in helping to cut down on fragmentation. It also means that new features of Android, such as the new APIs in Ice Cream Sandwich, become available on more phones, much faster. A year later, this promise hasn't been kept, it seems. A significant portion of the market is still running older Android versions, mostly Gingerbread. If the developers of a game want mass-market acceptance, the game will need to run on no fewer than six different versions of Android, spread across 600+ devices (and counting!).

But fear not. Although this sounds terrifying, it turns out that the measures that have to be taken to accommodate multiple versions of Android are minimal. Most often, you can even forget about the issue and pretend there's only a single version of Android. As game developers, we're less concerned with differences in APIs and more concerned with hardware capabilities. This is a different form of fragmentation, which is also a problem for platforms such as iOS, albeit not as pronounced. Throughout this book, we will cover the relevant fragmentation issues that might get in your way while you're developing your next game for Android.

The Role of Google

Although Android is officially the brainchild of the Open Handset Alliance, Google is the clear leader when it comes to implementing Android itself, as well as providing the necessary ecosystem for it to grow.

The Android Open Source Project

Google's efforts are summarized in the *Android Open Source Project*. Most of the code is licensed under Apache License 2, which is very open and nonrestrictive compared to other open source licenses, such as the GNU General Public License (GPL). Everyone is free to use this source code to build their own systems. However, systems that are proclaimed Android compatible first have to pass the Android Compatibility Program, a process that ensures baseline compatibility with third-party applications written by developers. Compatible systems are allowed to participate in the Android ecosystem, which also includes *Google Play*.

Google Play

Google Play (formerly known as *Android Market*) was opened to the public by Google in October 2008. It's an online store that enables users to purchase music, videos, books and third-party applications, or *apps*, to be consumed on their device. Google Play is primarily available on Android devices, but also has a web front end where users can search, rate, download, and install apps. It isn't required, but the majority of Android devices have the Google Play app installed by default.

Google Play allows third-party developers to publish their programs either for free or as paid applications. Paid applications are available for purchase in many countries, and the integrated purchasing system handles exchange rates using Google Checkout. Google Play also gives the option to price an app manually on a per-country basis.

A user gets access to the store after setting up a Google account. Applications can be purchased via credit card through Google Checkout or by using carrier billing. Buyers can decide to return an application within 15 minutes of the time of purchase for a full refund. Previously, the refund window was 24 hours, but it was shortened to curtail exploitation of the system.

Developers need to register an Android developer account with Google, for a one-time fee of \$25, in order to be able to publish applications on the store. After successful registration, a developer can start publishing new applications in a matter of minutes.

Google Play has no approval process, instead relying on a permission system. Before installing an application, the user is presented with a set of required permissions, which handle access to phone services, networking, Secure Digital (SD) cards, and so on. A user may opt not to install an application because of permissions, but a user doesn't currently have the ability to simply not allow an application to have a particular permission. It is "take it or leave it" as a whole. This approach aims to keep apps honest about what they will do with the device, while giving users the information they need to decide which apps to trust.

In order to sell applications, a developer additionally has to register a Google Checkout merchant account, which is free of charge. All financial transactions are handled through this account. Google also has an in-app purchase system, which is integrated with the Android Market and Google Checkout. A separate API is available for developers to process in-app purchase transactions.

Google I/O

The annual Google I/O conference is an event that every Android developer looks forward to each year. At Google I/O, the latest and greatest Google technologies and projects are revealed, among which Android has gained a special place in recent years. Google I/O usually features multiple sessions on Android-related topics, which are also available as videos on YouTube's Google Developers channel. At Google I/O 2011, Samsung and Google handed out Galaxy Tab 10.1 devices to all regular attendees. This really marked the start of the big push by Google to gain market share on the tablet side.

Android's Features and Architecture

Android is not just another Linux distribution for mobile devices. While developing for Android, you're not all that likely to meet the Linux kernel itself. The developer-facing side of Android is a platform that abstracts away the underlying Linux kernel and is programmed via Java.

From a high-level view, Android possesses several nice features:

- *An application framework* that provides a rich set of APIs for creating various types of applications. It also allows the reuse and replacement of components provided by the platform and third-party applications.

- The *Dalvik virtual machine*, which is responsible for running applications on Android.
- A set of *graphics libraries* for 2D and 3D programming.
- *Media support* for common audio, video, and image formats, such as Ogg Vorbis, MP3, MPEG-4, H.264, and PNG. There's even a specialized API for playing back sound effects, which will come in handy in your game development adventures.
- *APIs for accessing peripherals* such as the camera, Global Positioning System (GPS), compass, accelerometer, touchscreen, trackball, keyboard, controller, and joystick. Note that not all Android devices have all these peripherals—hardware fragmentation in action.

Of course, there's a lot more to Android than the few features just mentioned. But, for your game development needs, these features are the most relevant.

Android's architecture is composed of stacked groups of components, and each layer builds on the components in the layer below it. Figure 1-1 gives an overview of Android's major components.

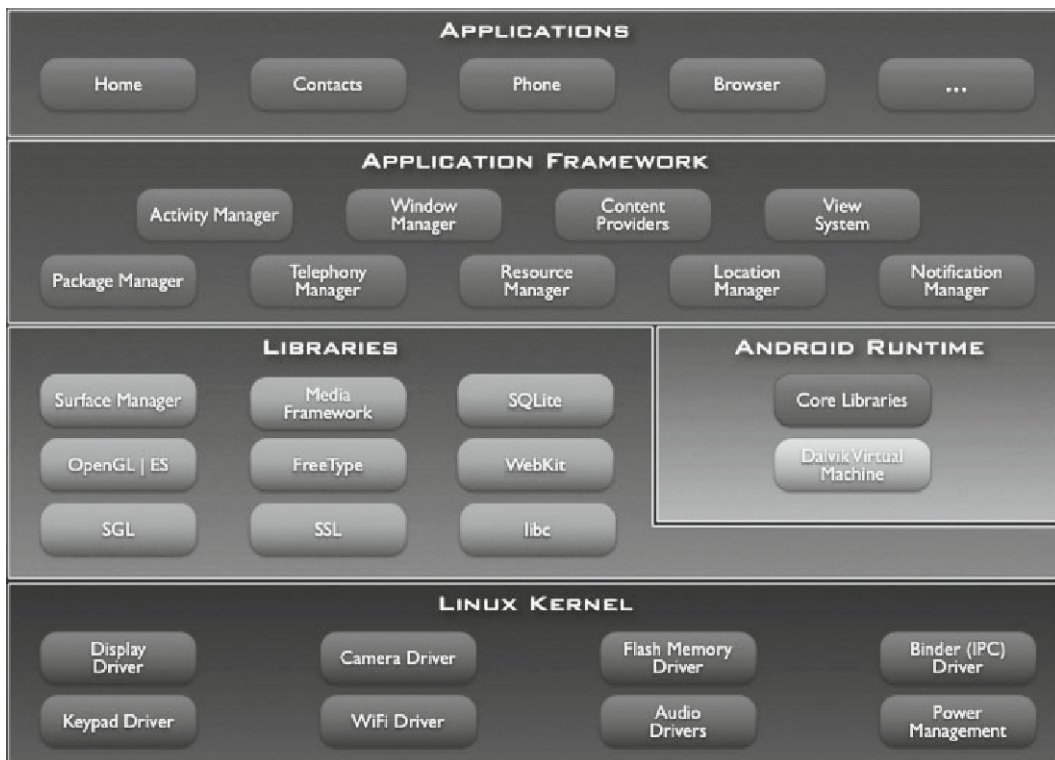


Figure 1-1. Android architecture overview