Get started with game apps development
for the Android platform

Beginning Android Games

Mario Zechner

Apress®

Download from Wow! eBook <www.wowebook.com>

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.

Apress®

Contents at a Glance

Contents v
About the Author xii
About the Technical Reviewer xiii
Acknowledgments xiv
Introduction xv
Chapter 1: Android, the New Kid on the Block 1
Chapter 2: First Steps with the Android SDK 25
Chapter 3: Game Development 101 51
Chapter 4: Android for Game Developers 103
Chapter 5: An Android Game Development Framework 185
Chapter 6: Mr. Nom Invades Android 229
Chapter 7: OpenGL ES: A Gentle Introduction 269
Chapter 8: 2D Game Programming Tricks 351
Chapter 9: Super Jumper: A 2D OpenGL ES Game 429
Chapter 10: OpenGL ES: Going 3D 489
Chapter 11: 3D Programming Tricks 525
Chapter 12: Droid Invaders: the Grand Finale 577
Chapter 13: Publishing Your Game 625
Chapter 14: What's Next? 637
Index 641

Introduction

Hi there, and welcome to the world of Android game development. My name is Mario; I'll be your guide for the next fourteen chapters. You came here to learn about game development on Android, and I hope to be the person who enables 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, and a scary thing called OpenGL ES. Based on all this knowledge we'll develop three different games, one even being 3D.

Game programming can be easy if you know what you're doing. Therefore I'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.

A Word About the Target Audience

This book is aimed first and foremost at complete beginners in game programming. You don't need any prior knowledge on the subject matter; I'll walk you through all the basics. However, I need to assume a little knowledge on your end about Java. If you feel rusty on the matter, I'd suggest refreshing your memory by reading the online edition of *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 the intermediate-level game programmer that wants to get her 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 Organized

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, so I'd suggest going through the chapters in sequence. More experienced readers can of course skip certain sections they feel confident with. Just make sure to read through the code listings of sections you skim over a little, so you will understand how the classes and interfaces are used in subsequent, more advanced sections.

Getting the Source 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. Also, the process of copying code from the book's text to Eclipse can introduce errors. Robert (the book's technical reviewer) and I 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, I created a Google Code project that offers you the following:

- The complete source code and assets, licensed under the GPL version 3, available from the project's Subversion repository.
- 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, I 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. I'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. Each project therefore 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/beginning-android-games`.

Chapter 1

Android, the New Kid on the Block

As a kid of the early nineties, I naturally grew up with my trusty Nintendo Game Boy. I spent countless hours helping Mario rescue the princess, getting the highest score in Tetris, and racing my friends in RC Pro-Am via link cable. I took this awesome piece of hardware with me everywhere and every time I could. My passion for games made me want to create my own worlds and share them with my friends. I started programming on the PC but soon found out that I couldn't transfer my little masterpieces to the Game Boy. I continued being an enthusiastic programmer, but over time my interest in actually playing video games faded. Also, my Game Boy broke . . .

Fast forward to 2010. Smartphones are becoming the new mobile gaming platforms of the era, competing with classic dedicated handheld systems such as the Nintendo DS or the Playstation Portable. That caught my interest again, and I started investigating which mobile platforms would be suitable for my development needs. Apple's iOS seemed like a good candidate to start coding games for. However, I quickly realized that the system was not open, that I'd be able to share my work with others only if Apple allowed it, and that I'd need a Mac to develop for the iOS. And then I found Android.

I 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 comprehensive documentation. I can share my games with anyone without having to pay a fee to do so, and if I want to monetize my work, I can easily publish my latest and greatest innovation to a global market with millions of users in a matter of minutes.

The only thing I was left with was actually figuring out how to write games for Android and how to transfer my PC game development knowledge to this new system. In the following chapters, I want to share my experience with you and get you started with Android game development. This is of course a rather selfish plan: I 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 publicly noticed in 2005 when Google acquired a small startup called Android, Inc. This fueled speculation that Google wanted to enter the mobile space. In 2008, the release of version 1.0 of Android put an end to all speculation, and Android became the new challenger on the mobile market. Since then, it's been battling it out with already established platforms such as iOS (then called iPhone OS) and BlackBerry, and its chances of winning look rather good.

Because Android is open source, handset manufacturers have a low barrier of entry when 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 to low-budget 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 collaborate to develop open standards for mobile devices. Although Android's core is developed mainly by Google, all the OHA members contribute to its source in one form or another.

Android itself is a mobile operating system and platform based on the Linux kernel version 2.6 and is freely available for commercial and noncommercial use. Many members of the OHA build custom versions of Android for their devices with modified user interfaces (UIs)—for example, HTC's HTC Sense and Motorola's MOTOBLUR. The open source nature of Android also enables hobbyists to create and distribute their own versions of Android. These are usually called *mods*, *firmwares*, or *ROMs*. The most prominent ROM at the time of this writing was developed by a fellow known as Cyanogen and is aimed at bringing the latest and greatest improvements to all sorts of Android devices.

Since its release in 2008, Android has received seven version updates, all code-named after desserts (with the exception of Android 1.1, which is irrelevant nowadays). Each version has added new functionality to the Android platform that has relevance 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 upmost concern. Version 1.6 (Donut) introduced support for different screen resolutions. We will revisit this fact a couple of times in this book because it has some impact on how we approach writing games for Android. With version 2.0 (Éclair) came support for multi-touch screens, and version 2.2 (Froyo) added just-in-time (JIT) compilation to the Dalvik virtual machine (VM), which powers all the Java applications on Android. The JIT speeds up the execution of Android applications considerably—depending on the scenario, up to a factor of five. At the time of this writing, the latest version is 2.3, called Gingerbread. It adds a new concurrent garbage collector to the Dalvik VM. If you haven't noticed yet: Android applications are written in Java.

A special version of Android, targeted at tablets, is also being released in 2011. It is called Honeycomb and represents version 3.0 of Android. Honeycomb is not meant to

run on phones at this point. However, some features of Honeycomb will be ported to the main line of Android. At the time of this writing, Android 3.0 is not available to the public, and no devices on the market are running it. Android 2.3 can be installed on many devices using custom ROMs. The only handset using Gingerbread is the Nexus S, a developer phone sold by Google directly.

Fragmentation

The great flexibility of Android comes at a price: companies that opt to develop their own user interfaces have to play catch-up with the fast pace at which new versions of Android are released. This can lead to handsets not older than a few months becoming outdated really fast as carriers and handset manufacturers refuse to create updates that incorporate the improvements of new Android versions. The big bogeyman called *fragmentation* is a result of this process.

Fragmentation has many faces. For the end user, it means being unable to install and use certain applications and features because of being stuck on an old Android version. For developers, it means that some care has to be taken when creating applications that should work on all versions of Android. While applications written for earlier versions of Android will usually run fine on newer versions, the reverse is not true. Some features added in newer Android versions are of course not available on older versions, such as multi-touch support. Developers are thus forced to create separate code paths for different versions of Android.

But fear not. Although this sounds terrifying, it turns out that the measures that have to be taken are minimal. Most often, you can even completely forget about the whole 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 about 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, I will cover the relevant fragmentation issues that might get in your way while you develop 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 Android to grow.

The Android Open Source Project

Google's efforts are summarized under the name *Android Open Source Project*. Most of the code is licensed under Apache License 2, a very open and nonrestrictive license 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 claimed to be Android compatible first have to pass the Android Compatibility

Program, a process ensuring baseline compatibility with third-party applications written by developers like us. Compatible systems are allowed to participate in the Android ecosystem, which also includes the Android Market.

The Android Market

The *Android Market* was opened to the public in October 2008 by Google. It's an online software store that enables users to find and install third-party applications. The market is generally accessible only through the market application on a device. This situation will change in the near future, according to Google, which promises the deployment of a desktop-based online store accessible via the browser.

The market allows third-party developers to publish their applications either for free or as paid applications. Paid applications are available for purchase in only about 30 countries. Selling applications as a developer is possible in a slightly smaller number. Table 1–1 shows you the countries in which apps can be bought and sold.

Table 1–1. *Purchase and Selling Options per Country.*

Country	User Can Purchase Apps	Developer Can Sell Apps
Australia	Yes	Yes
Austria	Yes	Yes
Belgium	Yes	Yes
Brazil	Yes	Yes
Canada	Yes	Yes
Czech Republic	Yes	**No**
Denmark	Yes	Yes
Finland	Yes	Yes
France	Yes	Yes
Germany	Yes	Yes
Hong Kong	Yes	Yes
Hungary	Yes	Yes
India	Yes	Yes
Ireland	Yes	Yes

Country	User Can Purchase Apps	Developer Can Sell Apps
Israel	Yes	Yes
Italy	Yes	Yes
Japan	Yes	Yes
Mexico	Yes	Yes
Netherlands	Yes	Yes
New Zealand	Yes	Yes
Norway	Yes	Yes
Pakistan	Yes	**No**
Poland	Yes	**No**
Portugal	Yes	Yes
Russia	Yes	Yes
Singapore	Yes	Yes
South Korea	Yes	Yes
Spain	Yes	Yes
Sweden	Yes	Yes
Switzerland	Yes	Yes
Taiwan	Yes	Yes
United Kingdom	Yes	Yes
United States	Yes	Yes

Users get access to the market after setting up a Google account. Applications can be bought only via credit card at the moment. Buyers can decide to return an application within 15 minutes from the time of purchasing it and will receive a full refund. Previously, the refund time window was 24 hours. The recent change to 15 minutes has not been well received by end users.

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 market. After successful