Chapter 14

What's Next?

We talked about a ton of stuff in this book, and still there's so much more to learn. If you feel comfortable with all the material in here, you'll probably want to dig deeper. Here are a couple of ideas and directions for your journey.

Getting Social

One of the biggest trends in gaming in the last couple of years has been the integration with social media and services. Twitter, Facebook, and Reddit have become a part of many people's lives. And those people want to have their friends and family with them in their games as well. What's cooler than having your Dad beat you in the latest iteration of Zombie Shooter 13, right?

Both Twitter and Facebook provide APIs that let you interact with their services. Want to give the user a way to tweet their latest high score in your game? No problem—integrate the Twitter API and away you go.

On a related note, there are two big services in the mobile space that make it their goal to connect gamers and let them discover new games easily: Scoreloop and OpenFeint. Both provide an API for Android that lets players easily keep online high scores, compare their achievements, and so on. Both APIs are pretty straightforward and come with good examples and documentation. I prefer Scoreloop, for what it's worth.

Location Awareness

We only briefly touched on this in Chapters 1 and 4 and didn't exploit it in any of our games. All Android devices come with some type of sensor that lets you determine a user's location. That in itself is interesting already, but using it in a game can make for some innovative and never-before-seen game mechanics. It's still a hardly used feature in most Android games. Can you think of a fun way to use the GPS sensor?

M. Zechner, *Beginning Android Games*
© Mario Zechner 2011

Multiplayer Functionality

This being a beginner's book, we haven't talked about how to create multiplayer games. Suffice it to say that Android provides you with the APIs to do just that. Depending on the type of game, the difficulty of implementing multiplayer functionality varies. Turn-based games, such as chess or card games, are pretty simple to implement. Fast-paced action games or real-time strategy games are a different matter altogether. In both cases, you need to have an understanding of network programming, a topic for which a lot of materials exist on the Web.

OpenGL ES 2.0 and More

So far, you've seen only half of OpenGL ES, so to speak. We used OpenGL ES 1.0 exclusively, because it is the most widely supported version on Android at this point. Its fixed-function nature lends itself well to getting into 3D graphics programming. However, there's a newer, shinier version of OpenGL ES that enables you to directly code on the GPU. It's very different from what you have seen in this book, in that you are responsible for all the nitty-gritty details such as fetching a single texel from a texture or manually transforming the coordinates of a vertex, all directly on the GPU.

OpenGL ES 2.0 has a so-called shader-based, or programmable, pipeline as opposed to the fixed-function pipeline of OpenGL ES 1.0 and 1.1. For many 3D (and 2D) games, OpenGL ES 1.*x* is more than sufficient. If you want to get fancy, you might want to consider checking out OpenGL ES 2.0, though! Don't be afraid—all the concepts you learned in this book will be easily transferable to the programmable pipeline.

We also haven't touched on topics such as animated 3D models and some more-advanced OpenGL ES 1.*x* concepts such as vertex buffer objects. As with OpenGL ES 2.0, you can find many resources on the Web as well as in book form. You know the basics. Now it's time to learn even more!

Frameworks and Engines

If you bought this book with a little prior game development knowledge, you may have wondered why I didn't choose to use one of the many frameworks available for Android game development. Reinventing the wheel is bad, right? I want you to firmly understand the principles. Although this may be tedious at times, it will pay off in the end. With the knowledge you gained here, it will be so much easier to pick up any precanned solution out there, and it is my hope that you'll recognize the advantage that gives you.

For Android, a couple of commercial and noncommercial, open source frameworks and engines exist. What's the difference between a framework and an engine?

A framework will give you control over every aspect of your game development environment. This comes at the price of having to figure out your own way of doing things (for example, how you organize your game world, how you handle screens and

transitions, and so on). In this book, we developed a (very simple) framework upon which we build our games.

An engine, on the other hand, is more streamlined for specific tasks. It dictates how you should do things, giving you easy-to-use modules for common tasks and a general architecture for your game. The downside is that your game might not fit the precanned solutions the engine offers you. Often times, you'll have to modify the engine itself to achieve your goals, which might or might not be possible, depending on whether the source is available. Engines can greatly speed up initial development time but might slow it to a grinding halt if you encounter a problem that the engine was not made for.

In the end, it's a matter of personal taste, budget, and goals. As an independent developer, I prefer frameworks because those are usually easier for me to understand and because they let me do things in the exact way I want them to be done.

With that being said, choose your poison. Here's a list of frameworks and engines that can speed up your development process:

- *Unreal Development Kit* (`www.udk.com`): A commercial game engine running on a multitude of platforms and developed by Epic Games. Those guys made games such as Unreal Tournament, so this engine is quality stuff. Uses its own scripting language.
- *Unity* (`www.unity3d.com`): Another commercial game engine with great tools and functionality. It too works on a multitude of platforms, including iOS and Android, or in the browser and is easy to learn. Allows a couple of languages for coding the game logic; Java is not among them.
- *jPCT-AE* (`www.jpct.net/jpct-ae/`): A port of the Java-based jPCT engine for Android, this has some great features with regard to 3D programming. Works on the desktop and on Android. Closed source.
- *Ardor3D* (`www.ardor3d.com`): A very powerful Java-based 3D engine. Works on Android and on the desktop, and is open source with great documentation.
- *libgdx* (`code.google.com/p/libgdx/`): An open source Java-based game development framework by yours truly, for 2D and 3D games. Works on Windows, Linux, Mac OS X, and of course Android without any code modifications. You can develop and test on the desktop without the need for an attached device or the slow emulator (or waiting a minute until your APK file is uploaded to the device). You'll probably feel right at home after having read this book—my evil plan. Did you notice that this bullet-point is only slightly bigger than the rest?
- *Slick-AE* (`http://slick.cokeandcode.com`): A port of the Java-based Slick framework to Android, built on top of libgdx. Tons of functionality and easy-to-use API for 2D game development. Cross-platform and open source, of course.

- *AndEngine* (www.andengine.org): A nice, Java-based Android-only 2D engine, partially based on libgdx code (open source for the win). Similar in concept to the famous cocos2d game development engine for iOS.

I suggest giving those options a try at some point. They can help you speed up your game development quite a bit.

Resources on the Web

The Web is full of game development resources. In general, Google will be your best friend, but there are some special places that you should check out, including these:

- www.gamedev.net: One of the oldest game development sites on the Web, with a huge treasure chest of articles on all sorts of game development topics.
- www.gamasutra.com: Another old Goliath of game development. More industry orientated, with lots of postmortems and insight into the professional game development world.
- http://wiki.gamedev.net: A big wiki on game development, chock full of articles on game programming for different platforms, languages, and so on.
- www.flipcode.com/archives/: The archives of the now defunct flipcode site. Some pearls can be found here. Although slightly outdated at times, it is still an incredibly good resource.
- www.java-gaming.org: The -number one go-to place for Java game developers. People such as Markus Persson of Minecraft fame frequent this place (well, Markus doesn't that much anymore, given his current lifestyle).

Closing Words

It's 2:20 a.m. and I'm sitting at my kitchen table with my trusty netbook in front of me. That's been my default pose for the last couple of months at night. I hope I can change my sleeping pattern back to normal.

Writing this book was a joy (the mornings not so much), and I hope I gave you what you came here for. There's so much more to discover, so many more techniques, algorithms, and ideas to explore. This is just the beginning for you. There's more ahead to learn.

I feel confident that with the material we dissected and discussed, you have a solid foundation to build upon that will enable you to grasp new ideas and concepts faster. There's no need to fall into the trap of copying and pasting code anymore. Even better, almost all the things we discussed will translate well to any other platform (give or take some language or API differences). I hope you can see the big picture and that it will enable you to start building the games of your dreams.