Part 4

Final: The Beginning of Disparity

We have been focusing on common mobile topics using counterpart iOS analogies, but there are disparate areas that I tried not to get into for learning objectives. The mapping guidelines and instructions work perfectly for their intended purposes. However, you don't want to create iOS-ish Android apps—and most importantly, you don't want to stop right here.

Some of the topics introduced in this book actually could be too iOS-ish from the perspective of UI or design approaches. As you become more advanced in Android programming, you will want to keep an open eye for Android-specific things. To avoid misleading you in the long run, let me single out the following topics.

Provide an Android Mobile UX for Your Android Users

As you plan or design your app for Android, keep in mind that different platforms may have different rules and conventions. This is particularly true for UI design guidelines. Android apps should not look like iOS apps. On the official Android Developers site, there is a page called "Pure Android" at http://developer.android.com/design/patterns/pure-android.html.

Without getting into the user experience (UX) details, allow me to excerpt some bulleted points from the web page, just to get my points across:

- Don't use bottom tab bars
- Don't use labeled Back buttons on action bars
- Don't use right-pointing carets on list line items

I could not agree more because this pretty much summarizes the UI design mistakes I often made by blindly porting the iOS UI to Android. I find this short document very useful for when I need to switch my UI-thinking mode.

Handle Runtime Changes

By default, Android activities restart when a device configuration changes, such as device orientation. Two things happen:

- The activity/fragment restarts and goes through the regular lifecycles—destroyed and re-created, and so forth.
- The bundle system restarts and loads the appropriate application resources.

This is a unique Android behavior that is different from iOS apps. In this book, we disable this default restart behavior primarily for porting purposes. However, you may encounter situations where you can take advantage of letting the system restart the activity that causes reloading of appropriate alternative application resources. This is a unique but important topic, but I do not go into detail. On the official Android Developers site, there is a page called "Handling Runtime Changes" at http://developer.android.com/guide/topics/resources/runtime-changes.html.

By reading this topic, you will learn when you can take advantage of restarting the bundled system. I also found it is well written especially where it describes the Android runtime system. If you want to have a better understanding of the Android system, then do not miss reading this topic.

Communicate with Other Fragments

To communicate between fragment view controllers, we chose to send and receive parameters directly between the presenting fragment and the presented fragment. Pure Android developers are probably screaming at me right now. There is a good article that specifically addresses the proper solution with sample code on the official Android Developers site at http://developer.android.com/training/basics/fragments/communicating.html.

Based on this web page, all the communication between two fragments should go through their parent activity. In practice, I am not against either way, but you do want to pay attention to the performance overhead when you send serialized data via bundle for communication between fragments directly.

There are no chapters (other than the Appendix) in Part IV of the book. You have gotten this far: you will be the author continuing with your own next chapters.