Chapter 1

Introducing Xcode Tools for Mac OS X and the iPhone

Since you are reading this, I hope that you are doing so on the way to the checkout at your bookstore! Or perhaps you have just opened a packaging from the mail. Either way, you have a great journey ahead of you, and I hope that this book fulfills the hopes you have for it.

When I learned the arcane rituals of Mac software development I found it difficult, in part because I needed to learn at least three new and difficult paradigms simultaneously. I needed a friendly helper to get me over those barriers. This is the book that I wanted to have at my fingertips, and I hope it becomes your friendly helper.

Software Development Choices for the Mac Programmer

Software development for the professional Mac developer has always had great support from Apple. For new developers and those who are doing it for fun, the situation has been less easy. For many years such developers had some difficult decisions to make if they wanted to build their own Mac software.

Mac OS X offers some great high-level tools for improving productivity and automation, such as Automator, Dashcode, and, of course, the venerable AppleScript. These are very useful in adding features to existing applications and in automating repetitive tasks, but you wouldn’t use them to create real applications.

There have been pseudo-languages like HyperCard and its successors, taking a very modular building-block approach to building software. These let you put together simple
applications quickly and easily, but can be limited in their features. So what if you want to create your killer application for the Mac?

It’s worth mentioning REALbasic. This is a great tool for software development and has a loyal following of active programmers. It has a simple and easy-to-use integrated development environment (IDE) and uses the syntactically simple BASIC as its language. It also has the benefit of creating executables for Mac OS X, Windows, and Linux from essentially a single code base.

However, it has some limitations. REALbasic uses its own libraries, not Cocoa frameworks (at the time of writing the company has announced its intention to support Cocoa), and there is usually an indefinable air that tells you an application has been written in REALbasic. Executable files can be on the large side. Although it supports an object-oriented approach, REALbasic doesn’t enforce it, and this, together with its short learning curve, can lead to products of indifferent quality. You will find that Xcode is quite different. It is what is known as opinionated software: while there is a lot of flexibility, there is a right way to develop in Xcode, and you will find life much easier if you follow it.

REALbasic is not a good grounding or a springboard for a subsequent move to Xcode. The paradigms are completely different, and when I made the move I essentially had to forget everything I knew. REALbasic is a commercial product, of course, in contrast to the free Xcode Tools. Finally, although you are able to target Windows and Linux as platforms, REALbasic cannot be used (at least, at the time of writing) to build iPhone software.

All in all, REALbasic offers a lot of fun and productivity to the developer, and for knocking out simple utility programs it’s unbeatable. But as you develop more of an interest in serious software development, there is a good chance that you will begin to develop some curiosity about Xcode.

**Why You Should Develop Using Xcode Tools**

Given the number of alternative choices, it might seem easy to dismiss Xcode Developer Tools (which you’ll often see referred to as just Xcode Tools) and Cocoa frameworks as a choice for developing software for the Mac. It is fair to say, too, that historically Xcode and its antecedent products have not been very accessible to the new developer. The first sight of the Xcode project interface can be daunting, Objective-C has a quirky syntax, the separation of design imposed by the MVC pattern is a conceptual leap too far for many, and the documentation, though comprehensive, has historically not been aimed at the learner.

The need to come to grips with all of these new concepts was the main thing that prevented me from learning to develop using Cocoa and Xcode. And that is a shame, because once you get past these initial barriers, a major benefit of Apple’s tools and technologies is their ease of use. Really. That observation was the driving force for this book—many would-be developers have no doubt been put off developing using Xcode because of the multiple learning curves that need to be negotiated simultaneously. My