Introduction

Mac OS X is a great operating system for software developers. Every copy of Mac OS X comes with a set of developer tools, the Xcode Tools. The Xcode Tools contain everything you need to create Mac OS X applications. As powerful as the Xcode Tools are, they can be difficult to learn. Most Mac OS X development books focus on Cocoa programming. They teach enough Xcode and Interface Builder for you to create the projects in the book. But there's more to writing real Mac OS X programs. Real Mac OS X programs must be tested and debugged to make sure they run properly and must be profiled to make sure they run fast enough. A Cocoa programming book isn't going to show you how to profile your program or find memory leaks in it.

Xcode Tools Sensei picks up where other books leave off. It teaches the Xcode Tools, not a particular language or programming framework. By reading this book you can spend more time writing, debugging, and profiling your programs and less time searching and reading documentation.

Contents

26 developer tools. 13 chapters. There's something for every Mac OS X developer in *Xcode Tools Sensei*.

Xcode

The first two chapters of this book cover Xcode, the centerpiece of the Xcode Tools. Xcode is an integrated development environment (IDE) for writing, compiling, and debugging Mac OS X programs. Xcode 2.0 added visual modeling tools for classes and data structures. In Chapter 1 you'll learn the following tasks:

- Creating an Xcode project for your program.
- Adding files to the project.
- Creating project and file templates.
- Editing source code files.
- Using Xcode 2's visual modeling tools.
- Searching developer documentation.
- Configuring compiler settings.
- Compiling programs.
- Creating universal binaries that run on both PowerPC and Intel processors.

Chapter 2 covers Xcode's debugging capabilities. In this chapter you'll learn how to pause your program, execute one line of code at a time, and view your program's variables. Use Xcode's debugger to make sure your program runs the way you expect it to run and to find mistakes in your code.

Interface Builder

Chapter 3 covers Interface Builder, which is the tool to build user interfaces in Mac OS X. In this chapter you'll learn how to create user interfaces for Cocoa and Carbon applications. You'll also learn to perform the following tasks for Cocoa applications:

- Connect items so they can communicate with each other in your program.
- Create Cocoa subclasses.
- Add functions to Cocoa classes.
- Use bindings to manage relationships between user interface elements and data.

Performance Tools

Mac OS X programs must run fast enough to be responsive to the user. Unresponsive applications are at best frustrating and at worst unusable. If your program runs too slow, you must find the slow areas of the program so you can make the program run faster. The next three chapters cover tools that can find the slow spots in your code.

Chapter 4 covers Sampler, which is a tool that examines your program's call stack periodically. By viewing the call stack statistics, you can discover the functions in your program where the program spends the most time.

Chapter 5 covers `gprof`, which is a command-line profiler. You'll learn how to tell Xcode to generate the profiling information `gprof` requires, how to run `gprof`, and how to view `gprof's` results.

Chapter 6 covers the CHUD Tools, which are Mac OS X's most powerful profiling tools. The chapter covers the graphical programs Shark and Saturn as well as the command-line tools `amber`, `simg4`, `simg5`, and `acid`.

Profiling your program with Mac OS X's performance tools is only half the battle. Interpreting the data the tools generate is the other half of the battle. Read Chapters 4–6 to win the whole battle.

Memory Tools

The next two chapters cover tools that help you find memory-related problems in your programs. Chapter 7 covers MallocDebug, which lets you see how much memory you're allocating in each function. It also detects memory leaks and memory overruns, where you write past a block of memory. Chapter 8 covers ObjectAlloc, which is a tool that lets you examine every memory allocation your program makes.

Command-Line Debugging Tools

Chapter 9 covers the Unix command-line debugging tools `fs_usage`, `sc_usage`, `vmmap`, `heap`, `leaks`, and `malloc_history`. If you've never heard of these tools before, don't worry. After reading Chapter 9 you'll become intimately familiar with them.

gcov

Chapter 10 covers `gcov`, which measures how many times your program executes each line of code in your program. By using `gcov` you can ensure that your program executes every line of code in your program.

cvs

Chapter 11 shows you how to work with version control using `cvs`. Version control lets you see the changes you make to source code files and lets you go back to previous versions of files. After reading Chapter 11 you'll know how to perform the following tasks:

- Create a repository to store your program's files under version control.
- Setup a repository so other developers can access it.
- Import Xcode projects into the repository.
- Add files to the repository from Xcode.
- See the changes you've made to files.

Java Tools

Chapter 12 covers the Java tools. The chapter begins with Jar Bundler, a tool that creates a Mac OS X application bundle out of a Java application. Use Jar Bundler to create an application the user can launch from the Finder. The chapter then covers JavaBrowser, a tool to view Java documentation, and ends with Applet Launcher, a tool to test Java applets.

OpenGL Tools

I finish the book by covering the OpenGL developer tools. There are three OpenGL developer tools.

- OpenGL Profiler, which profiles and debugs OpenGL applications.
- OpenGL Driver Monitor, which displays realtime statistics about your graphics card.
- OpenGL Shader Builder, which you use to create OpenGL shaders to give your OpenGL applications more control over drawing a scene.

What About AppleScript Studio?

If you've visited Apple's developer site, you may have read about AppleScript Studio. The site describes AppleScript Studio as a tool to quickly create Mac OS X applications in AppleScript using the Cocoa framework. But AppleScript Studio does not appear in the table of contents. Why didn't I write about such an important tool?

AppleScript Studio is not an application so I can't include it as a developer tool. AppleScript Studio consists of Xcode, Interface Builder, the Cocoa framework, and AppleScript. Read the Xcode and Interface Builder chapters if you want to learn AppleScript Studio.

What the Reader Needs to Know

I assume the reader has some experience writing Mac OS X programs. Xcode has built-in support for five programming languages: AppleScript, C, C++, Java, and Objective C. Apple has the Cocoa and Carbon frameworks for developing Mac OS X applications and supports the Swing framework for developing cross-platform GUI applications in Java. I could not cover all these topics adequately and explain the Xcode Tools in one book.

If you're new to programming, I recommend Dave Mark's *Learn C on the Macintosh*. This electronic book complements the book you're reading right now. *Learn C on the Macintosh* teaches the C programming language using Xcode. After learning C, you'll be ready to tackle a Cocoa programming book and start writing Mac OS X applications.

Some Things to Keep in Mind as You Read This Book

Apple is constantly changing the Xcode Tools. The constant change means Apple is making lots of improvements to the Xcode Tools, which is good. The bad side of constant change is that the way to perform a task can change. I wrote this book for Xcode Tools 2.2. If you have a different version of the Xcode Tools, the screenshots may look different and the way you perform tasks may differ than what appears in the book. Xcode and Shark are the two tools that change the most. You shouldn't have much of a problem with the other tools.

The Xcode Tools have multiple methods to perform many tasks. Rather than detail each possible way to accomplish a task, I usually mention only one of the methods in the chapter text. Just because I mention one way to do something doesn't mean the other methods are worse. You may find it easier to complete a task differently than the approach I mention.

For the latest updates go to this book's official website.

```
http://www.meandmark.com/xcodebook.html
```

If you have any questions or comments about the book, feel free to email me at the address below. I'll do my best to answer any questions.

```
xcodebook@meandmark.com
```