Changes for Xcode 4.4

This document details the change I made when updating the book from Xcode 4.2 to 4.4. I've organized the changes by chapter.

Introduction

Apple packages Xcode as a single application instead of creating a `Developer` folder on your startup disk. The change in packaging makes finding the other developer tools more difficult. You can launch other developer tools, such as Instruments, by choosing Xcode > Open Developer Tool.

The Mac OpenGL tools are no longer packaged with Xcode. Choose Xcode > Open Developer Tool > More Developer Tools to go to Apple's developer downloads page. Download the Graphics Tools for Xcode package to install the OpenGL tools. If you want to compile code from the command line, download the Command Line Tools for Xcode package.

Chapter 1

Project Creation

Xcode 4.4 adds an Organization Name text field for most project templates. The name you enter in the text field appears in the copyright notice at the top of new source code files Xcode adds to the project.

Cocoa and iOS applications may see a Class Prefix text field. If you enter a prefix in this field, Xcode adds the prefix to any classes and class files you create in the project. An example of a class prefix is the NS prefix used in many Cocoa classes. If you enter `XYZ` for a Cocoa application project, Xcode adds the files `XYZAppDelegate.h` and `XYZAppDelegate.m` to the project. The name of the class in this case `XYZAppDelegate`.

Project Templates

Xcode 4.4 adds the in-app purchase content project template for both Mac and iOS applications. You can find the project template in the Other group on the left side of the New Project Assistant.

The Sync Schema project template is no longer available in Xcode 4.4.

Search Navigator

Xcode 4.4 adds two styles for searching your project's symbols: symbol definitions and symbol references. When you use symbol definitions, the search navigator generates a listing for each place in your code that defines the symbol. When you use symbol references, the search navigator generates a listing for each place in your code that refers to the symbol. Symbol references generate more listings than symbol definitions.

Suppose you search for a function in your project named `DoSomething`. If you search symbol definitions, the search navigator displays one listing where you defined `DoSomething` in the header file and another listing where you wrote the code for `DoSomething` in the implementation file. If you search symbol references, the search navigator creates the two listings it creates for symbol definitions, plus a listing for each place in your code that calls `DoSomething`.

Xcode 4.4 adds the ability to insert a pattern into the search field. Click on the magnifying glass icon in the search field and choose Insert Pattern. Inserting a pattern allows you to search for things like tabs, white space, hexadecimal digits, email addresses, and URLs. You can add patterns to a search term to fine tune the search results.

File Templates

Apple added the Objective-C class extension file template in Xcode 4.4. A class extension file is a header file that allows you to declare part of a class's interface outside of the class's public header file. The interface is the `@interface` section in Objective-C header files. The main use of a class extension is to place private variables you don't want other developers to see. Place the variables you want other developers to see in the class's public header file, and place the ones you don't want them to see in the class extension file.

The C File template no longer includes a corresponding header file.

The combo box for choosing a superclass for an iOS Objective-C class adds the classes `UIViewController` and `UITableViewController`.

Removing Files from a Project

When you remove a file from the project, the alert that opens has a Move to Trash button instead of a Delete button. Clicking the Move to Trash button removes the file from the project and moves the file to the trash.

Organizer

When you click the Devices button at the top of the Organizer, there is no longer a Developer Profile section on the left side of the Organizer. Use the Provisioning Profiles section to transfer your developer information to a new Mac.

iOS provisioning profiles now last for one year.

Chapter 2

Converting to ARC

The refactoring tool can now convert projects that use garbage collection.

The Precheck button's name is now Check.

Converting to Modern Objective-C Syntax

Xcode 4.4 adds a second project-wide refactoring: converting Objective-C code to modern Objective-C syntax. Modern Objective-C syntax falls into two main categories. The first category is increased support for literals using the @ character. If you have worked with `NSString` objects, you know you can use the @ character to define a string literal.

```
NSString* myString = @"Hello";
```

The increased support adds support for literals to `NSNumber`, `NSArray`, and `NSDictionary` objects, which makes initializing numbers, arrays, and dictionaries easier.

```
NSNumber* myNumber = @10;
NSNumber* myDouble = @3.25;
NSArray* myArray = @[@10, @15, @20];
NSDictionary* myDictionary = {@1: @"red", @2: @"green", @3:
     @"blue", @4: @"alpha"};
```

The second category is support for using C array syntax to access elements in `NSArray` and `NSDictionary` objects. To access element 3 in an `NSArray` with the old syntax, you would use the following code:

```
[myArray objectAtIndex:3];
```

With modern Objective-C syntax you can access element 3 with the following code:

```
myArray[3];
```

Choose Edit > Refactor > Convert to Modern Objective-C Syntax to convert your project. Xcode walks you through the conversion. The first step is to tell you any changes that will be made to your project settings. Click the Next button to move on. The second step is to choose the targets to convert. Click the Next button to generate a preview of what will change. There are two source code views. The left view shows how the code will look after the conversion. The right view shows the original code. Between the two views is a button with a number. If you don't want to make a change, click the button and choose Discard Change. Click the Save button to do the conversion. Click the Cancel button to stop the conversion.

Fix-It

Fix-It now works with C++ code.

Quick Help

Starting with Xcode 4.4 you can view Quick Help from Xcode's code completion list. Select an entry from the completion list. Quick Help for the selected entry appears at the bottom of the completion list. Clicking the More link takes you to the selected entry's documentation in Xcode's documentation window.

Chapter 3

Starting with Interface Builder

New projects now show the hierarchical view of the object list instead of the icon view.

Object Library

Xcode 4.4 adds the page controller. Page controllers control page view animations. If your application displays multiple pages of information or uses swipes to switch pages, use a page controller.

Xcode 4.4 adds the SceneKit group, which contains one item: a scene view. A scene view displays a 3D scene using Apple's Scene Kit framework for 3D graphics. If you are writing a Cocoa application that uses Scene Kit, you should add a scene view to your window.

Auto Layout

New projects now have auto layout enabled.

Chapter 4

Starting with Interface Builder

New projects now show the hierarchical view of the object list instead of the icon view.

Object Library

Apple added the split view controller to Interface Builder in Xcode 4.4. A split view controller manages two view controllers in a split view.

Working with Scenes and Segues

If you add a modal segue, choose a transition from the Transition menu in the attributes inspector. If you add a custom segue, enter the custom segue's class in the Segue Class text field. The class must be a subclass of `UIStoryboardSegue`.

Chapter 5

Synchronizing Data Models

Xcode 4.4 removed support for synchronizing data models from the data model inspector.

Chapter 6

Entitlements

Xcode 4.4 adds a menu to control access to the user's Downloads folder.

Adding an Icon to Your Application

To add an icon to your application, you must add the icon file to your project. After adding the icon file to your project, drag it to the App Icon image well in the project editor.

For iOS applications the icon file is a PNG file. You must supply at least two PNG files, one for retina displays and one for non-retina displays, with the retina display image being twice the width and twice the height of the other image. If you have a universal application, you must supply four PNG files: two for iPhone and two for iPad. Drag the PNG files to the appropriate image well.

Mac applications have two different icon creation methods. The new method is to create a folder with the extension `.iconset` and add a set of PNG files to the folder. The old method is to create an ICNS icon file.

Apple's Icon Composer tool can help you create icon files out of PNG files. Choose Xcode > Open Developer Tool > More Developer Tools to access Apple's developer downloads site. Icon Composer is part of the Graphics Tools for Xcode package. Download the package to install Icon Composer.

In Icon Composer you will see image wells with labels 512, 256, 128, 32, and 16. Drag a PNG image to each image well. If you have a large PNG file (512-by-512 pixels or larger), you can drag it to each image well, and Icon Composer will scale the image down for each resolution. Saving the document in Icon Composer creates an ICNS icon file. Choosing File > Export To Icon Set creates an icon set. The Icon Composer version that comes with older versions of Xcode cannot export to an icon set.

Unfortunately when you create an icon set in Icon Composer, it does not create high-resolution files for retina displays. If you want to support retina displays, you must manually create the high-resolution files. If you look in the .iconset folder Icon Composer creates, you will see the following image files:

```
icon_16x16.png
icon_32x32.png
icon_128x128.png
icon_256x256.png
icon_512x512.png
```

To support retina displays, you must create the following image files and add them to the `.iconset` folder:

```
icon_16x16@2x.png
icon_32x32@2x.png
icon_128x128@2x.png
icon_256x256@2x.png
icon_512x512@2x.png
```

The @2x signifies the icon files are high-resolution. The high-resolution files must be twice the width and twice the height of the other files. The 512-by-512 high-resolution icon must be 1024 pixels wide and 1024 pixels high. After creating the @2x files, add the .iconset folder to the project. Open the project editor and drag the `.iconset` folder to the App Icon image well.

What Should My Deployment Target Be?

If you want all iPhone owners to be able to run your application, set the deployment target to iOS 3.1 because iOS 3.1.3 is the latest version that runs on all devices. Every iPad can run iOS 5. If you're writing an iPad application, iOS 5.0 is a reasonable deployment target.

Localizations

Xcode 4.4 adds a Use Base Localization checkbox for Cocoa and iOS application projects. You must be using the iOS 6 SDK to be able to do anything with the checkbox in an iOS project. Selecting the checkbox creates a new base localization that is used to localize other files. The base localization helps when localizing xib files. By using a base localization Xcode uses one xib file with a strings file for each spoken language your application supports. Separating the localizable data from the xib file simplifies translation.

When you select the Use Base Localization checkbox, a sheet opens with a list of files to localize, which most likely is a list of xib files in your project. If you don't want to localize a file, deselect the checkbox next to it. Click the Finish button to complete the localization.

After clicking the Finish button, you will see a new entry in the Localizations list named Base. If you add a new localization, you will see Base in the Reference Language column. In the File Types column you will see a localization string. If you click the Finish button and look at a xib file in the project navigator, you will see a strings file for the new language.

Build Settings

When entering a value for a build setting, the pop-up editor no longer has a Done button.

New Mac projects are configured to build for 64-bit Intel only.

Xcode 4.4 removed OpenMP build settings, garbage collection build settings, and the Interface Builder Compiler build settings collection.

Code Signing

Mac applications require code signing to submit them to the App Store and to be recognized by the Gatekeeper feature introduced in Mac OS X 10.8. Gatekeeper is a security feature in Mac OS X. In its initial setup Mac OS X will not allow a user to run downloaded applications from unidentified developers. To identify yourself you must join the paid Mac Developer Program, get a developer ID from Apple, and code sign your application. Getting a developer ID from Apple and code signing make it easier for users to launch your application.

Schemes

Mac projects have two possible run destinations: 32-bit and 64-bit, but most projects initially have only the 64-bit run destination. You must set the Architectures build setting to build for both 32-bit and 64-bit Intel to have a choice of run destinations.

Universal Application Build Settings

You don't have to check the Architectures build setting unless you need to support the `armv6` architecture.

Chapter 7

Configuring Your Scheme for Debugging

Xcode 4.4 adds the option to debug your project as yourself or as root. In most cases you should debug as yourself. If you are debugging a root process that requires root access, select the root radio button.

Breakpoint Actions

The pop-up editor has a button named Add Action to add an action to a breakpoint.

Debug Navigator

Xcode 4.4 replaces the By Thread and By Queue buttons at the top of the debug navigator with an icon button with an image of a spool of thread. Click the button to open a menu to group the call stacks by thread or by dispatch queue.

Variables View

The All option in the pop-up cell is now named All Variables, Registers, Globals, and Statics.

Xcode 4.4 has no Enable Data Formatters menu item.

OpenGL ES Debugging

The scheme editor has a OpenGL ES Frame Capture menu. It should be set to Automatically Enabled.

LLDB

Watchpoints

The `watchpoint set` command sets watchpoints. You can set watchpoints on expressions or variables. Setting a watchpoint on an expression takes the following form:

```
watchpoint set expression Expression
```

Setting a watchpoint on a variable takes the following form:

```
watchpoint set variable VariableName
```

Use the `watchpoint list` command to examine your watchpoints. LLDB displays the following information for each watchpoint:

- The watchpoint number.
- The memory address of the watchpoint.
- The size, in bytes.
- State, which is enabled or disabled.
- Type, which can have the following values: `r` for read-only, `w` for write-only, and `rw` for read and write.
- The file and line number for watchpoints set on variables.

Use the `watchpoint disable`, `watchpoint enable`, and `watchpoint delete` commands to disable, enable, and delete watchpoints. Supply a watchpoint number or a range of watchpoint numbers. If you supply no watchpoint numbers, LLDB disables, enables, or deletes all watchpoints.

```
watchpoint disable 3-4
watchpoint enable 3-4
watchpoint delete 5
```

Examinining Variables

The frame variable command has a `-G` option that lets you specify a GDB format specifier string to format the variables' data.

LLDB adds the following memory viewing formats to Table 7.5:

- E displays as an enumeration.
- O displays as an OSType.
- U displays as a 16-bit Unicode character.
- p displays as a pointer.
- a displays as a character array.
- A displays as an address.
- X displays as a hex float.
- i displays as a machine instruction.
- F displays as a C++ complex float.
- I displays as a C++ complex integer.
- intX_t[] displays as an array of X-bit signed integers, where X can be 8, 16, 32, or 64.
- uintX_t[] displays as an array of X-bit unsigned integers, where X can be 8, 16, 32, 64, or 128.
- floatX[] displays as an array of X-bit floating-point numbers, where X can be 32 or 64.

Use the d format to display as a signed decimal integer. Use the f format to display as a floating-point number.

Executing Shell Commands

Use the `platform shell` command to execute a Unix shell command in LLDB. The command takes the following form:

```
platform shell ShellCommand
```

With the `shell` command you can run another command-line program in Xcode's console. The following command runs the `leaks` tool on an application named MyApp:

```
platform shell leaks MyApp
```

Logging

LLDB has the following log channels: `lldb`, `gdb-remote`, `kdp-remote`, and `dwarf`.

Chapter 8

Cloning Repositories

For some git repositories when you enter the location, the Next button's text changes to Clone. Click the Clone button. An Open panel opens. Name the repository, choose a location to save it, and click the Clone button to clone the repository.

Removing Files from a Git Repository

Perform the following steps:

1. Select the files you want to remove from the project navigator.
2. Press the Delete key.
3. An alert opens asking if you want to move the deleted files to the trash or remove references to the files from the project. Click the Move to Trash button.
4. If you clicked the Remove Reference button, move the files you want to delete to the Trash. You won't be able to remove the files from the repository from Xcode unless you move them to the Trash.
5. Choose File > Source Code > Commit to remove the files from the repository.

Committing Changes

Xcode 4.4 adds the ability to cherry pick commits. Cherry picking allows you to commit some changes to a file while leaving other changes uncommitted. If you make five changes to a file, but only want to commit three, cherry pick the commits.

To cherry pick commits start by committing the file. A sheet opens that shows the changes. Between the two versions of the file is a button with a number. There is a button for each change in the file. For each change you don't want to commit, click the button and choose Don't Commit. Click the Commit button to commit the changes.

Discarding Changes

Xcode 4.4 adds the ability to discard changes to a file from the version editor. The version editor has a button for each change in the file. The button is between the two versions of the file and has a number. You won't be able to see the buttons if you're showing the timeline. Click the timeline button to hide the timeline. Click the button and choose Discard Change to discard the change. Using the version editor allows you to discard a single change to a file.

Annotations

Xcode 4.4 does not show the revision number in the blame view.

If you move the mouse cursor over an annotation, a small button appears next to the revision date. Clicking the button opens a pop-up editor that lets you read the commit message. The revision number appears below the person who modified the code. Next to the revision number is a focus button with an arrow. Clicking the button opens that revision in the version editor.

Viewing a File's Revisions

In Xcode 4.4 the SCM log no longer shows the revision number.

If you move the mouse cursor over a log entry, a small Info button appears next to the commit time. Clicking the button opens a pop-up editor that shows the revision number and the files that were modified in the revision. Next to the revision number is a focus button with an arrow. Clicking the button opens that revision in the version editor.

Creating Branches

Xcode 4.4 adds the ability to create a new branch when committing to a git repository. When you commit a file, there is a Commit to Branch button in the lower left corner of the sheet. If you click the Commit to Branch button, a second sheet opens. Click the pop-up cell in the Branch column and choose New Branch to create a branch.

Accessing Your Snapshots File

Above the project's snapshot list in the Organizer is the location of the project's snapshots and an inline button with an arrow. Clicking the button takes you to the snapshot location in the Finder.

Chapter 9

OpenGL ES Driver Template

It now includes the Time Profiler instrument instead of the Sampler instrument.

Source View

Xcode 4.4 adds a fifth button to the source view. To the right of the Designate Source button is a button that shows the source code and disassembly side by side.

Allocations Instrument

The Object Address column is now named Address. The Creation Time column is now named Timestamp.

Time Profiler Instrument

Call Tree

The call tree now has three columns of information: Running Time, Self, and Symbol Name. The Running Time column has two values: an amount of time and a percentage. The time value is the amount of time (in milliseconds) the symbol appears on the call stack. The percentage is the percentage of samples the symbol appears on the call stack.

The Self column tells you the amount of time the function appears at the top of the call stack. When a function appears at the top of the call stack, your application was in that function when Instruments recorded the sample. A function with a high Self value is a prime target for optimization. If you invert the call tree, the Self column matches the amount of time in the Running Time column. If you don't invert the call tree, the Self column tells you the amount of time your application spent in the function.

Strategy Bar

The rightmost menu in the strategy bar applies to the CPU and threads strategies. For the threads strategy the menu lets you choose what appears in the track pane: CPU usage or callstack samples. If you choose Callstack Samples, you can click on a sample in the track pane to view the call stack Instruments recorded for that sample.

Custom Instruments

Instruments no longer provides a Ruby provider. Because the Ruby provider was removed, I changed the custom instrument example to measure the amount of time spent in the Objective-C method `readFromFileWrapper:`.

In the two probes, choose Objective-C from the type menu instead of Ruby. After choosing Objective-C you should see a class text field, a hits text field, and a menu whose initial value is begin.

Enter the name of your class in the class text field. I'm using the class `Book`. Enter the name of the method in the hits text field. To record the `readFromFileWrapper:` method, enter the following text:

```
-readFromFileWrapper?ofType?error?
```

The ? character separates the arguments an Objective-C method takes. Choose begin from the menu for the first probe. Choose return for the second probe.

Chapter 10

fs_usage

`fs_usage` adds two filtering modes for the `-f` option: exec and pathname. Using the exec filtering mode tells `fs_usage` to show only exec and spawn events.

```
fs_usage -f exec AppName
```

Using the pathname filtering mode tells `fs_usage` to show only pathname-related events.

```
fs_usage -f pathname AppName
```

vmmap

`vmmap` adds the `-v` option. The `-v` option tells `vmmap` to generate verbose output. The `-v` option combines the `-w`, `-resident`, `-submap`, `-allSplitLibs`, and `-noCoalesce` options.

```
vmmap -v AppName
```

Chapter 11

OpenGL Performance Detective no longer has an Investigate Further button.