

iOS 4编程 (影印版)

iPhone、iPad和iPod Touch开发基础

涵盖 iOS 4.3
和 Xcode 4



Programming

iOS 4

O'REILLY®

东南大学出版社

Matt Neuburg 著

iOS 4编程 (影印版)

Programming iOS 4



O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目 (CIP) 数据

iOS 4 编程: 英文 / (美) 诺伊贝格 (Neuburg, M.)
著. —影印本. —南京: 东南大学出版社, 2011.10
ISBN 978-7-5641-2941-5

I. ① i… II. ① 诺… III. ① 移动电话机—应用程序—
程序设计—英文 IV. ① TN929.53

中国版本图书馆 CIP 数据核字 (2011) 第 173915 号

江苏省版权局著作权合同登记

图字: 10-2010-416 号

©2011 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2011. Authorized reprint of the original English edition, 2011 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2011。

英文影印版由东南大学出版社出版 2011。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式复制。

iOS 4 编程

出版发行: 东南大学出版社

地 址: 南京四牌楼 2 号 邮编: 210096

出 版 人: 江建中

网 址: <http://www.seupress.com>

电子邮件: press@seupress.com

印 刷: 扬中市印刷有限公司

开 本: 787 毫米 × 960 毫米 16 开本

印 张: 52.25

字 数: 1023 千字

版 次: 2011 年 10 月第 1 版

印 次: 2011 年 10 月第 1 次印刷

书 号: ISBN 978-7-5641-2941-5

定 价: 98.00 元 (册)

本社图书若有印装质量问题, 请直接与读者服务部联系。电话 (传真): 025-83792328

Preface

Aut lego vel scribo; doceo scrutorve sophian.

—Sedulius Scottus

With the advent of version 2 of the iPhone system, Apple proved they could do a remarkable thing — adapt their existing Cocoa computer application programming framework to make applications for a touch-based device with limited memory and speed and a dauntingly tiny display. The resulting Cocoa Touch framework, in fact, turned out to be in many ways better than the original Cocoa.

A programming framework has a kind of personality, an overall flavor that provides an insight into the goals and mindset of those who created it. When I first encountered Cocoa Touch, my assessment of its personality was: “Wow, the people who wrote this are really clever!” On the one hand, the number of built-in interface widgets was severely and deliberately limited; on the other hand, the power and flexibility of some of those widgets, especially such things as UITableView, was greatly enhanced over their Mac OS X counterparts. Even more important, Apple created a particularly brilliant way (UIViewController) to help the programmer make entire blocks of interface come and go and supplant one another in a controlled, hierarchical manner, thus allowing that tiny iPhone display to unfold virtually into multiple interface worlds within a single app without the user becoming lost or confused.

Even more impressive, Apple took the opportunity to recreate and rationalize Cocoa from the ground up as Cocoa Touch. Cocoa itself is very old, having begun life as NeXTStep before Mac OS X even existed. It has grown by accretion and with a certain conservatism in order to maintain something like backward compatibility. With Cocoa Touch, on the other hand, Apple had the opportunity to throw out the baby with the bath water, and they seized this opportunity with both hands.

So, although Cocoa Touch is conceptually based on Mac OS X Cocoa, it is very clearly *not* Mac OS X Cocoa, nor is it limited or defined by Mac OS X Cocoa. It’s an independent creature, a leaner, meaner, smarter Cocoa. I could praise Cocoa Touch’s deliberate use of systematization (and its healthy respect for Occam’s Razor) through numerous examples. Where Mac OS X’s animation layers are glommed onto views as a kind of afterthought, a Cocoa Touch view always has an animation layer counterpart.

Memory management policies, such as how top-level objects are managed when a nib loads, are simplified and clarified. And so on.

At the same time, Cocoa Touch is still a form of Cocoa. It still requires a knowledge of Objective-C. It is not a scripting language; it is certainly not aimed at nonprogrammers, like HyperCard's HyperTalk or Apple's AppleScript. It is still huge and complicated. In fact, it's rather difficult.

Meanwhile, Cocoa Touch itself evolves and changes. The iPhone System 2 matured into the iPhone System 3. Then there was a sudden sally in a new direction when the iPad introduced a larger screen and iPhone System 3.2. The iPhone 4 and its double-resolution Retina display also ran on a major system increment, now dubbed iOS 4. Every one of these changes has brought new complexities for the programmer to deal with. To give just one simple example, users rightly complained that switching between apps on the iPhone meant quitting one app and launching another. So Apple gave the iPhone 4 the power of multitasking; the user can switch away from an app and then return to it later to find it still running and in the state it was left previously. All well and good, but now programmers must scurry to make their apps compatible with multitasking, which is not at all trivial.

The popularity of the iPhone, with its largely free or very inexpensive apps, and the subsequent popularity of the iPad, have brought and will continue to bring into the fold many new programmers who see programming for these devices as worthwhile and doable, even though they may not have felt the same way about Mac OS X. Apple's own annual WWDC developer conventions have reflected this trend, with their emphasis shifted from Mac OS X to iOS instruction.

The widespread eagerness to program iOS, however, though delightful on the one hand, has also fostered a certain tendency to try to run without first learning to walk. iOS gives the programmer mighty powers that can seem as limitless as imagination itself, but it also has fundamentals. I often see questions online from programmers who are evidently deep into the creation of some interesting app, but who are stymied in a way that reveals quite clearly that they are unfamiliar with the basics of the very world in which they are so happily cavorting.

It is this state of affairs that has motivated me to write this book, which is intended to ground the reader in the fundamentals of iOS. I love Cocoa and have long wished to write about it, but it is iOS and its popularity that has given me a proximate excuse to do so. Indeed, my working title was "Fundamentals of Cocoa Touch Programming." Here I have attempted to marshal and expound, in what I hope is a pedagogically helpful and instructive yet ruthlessly Euclidean and logical order, the principles on which sound iOS programming rests, including a good basic knowledge of Objective-C (starting with C itself) and the nature of object-oriented programming, advice on the use of the tools, the full story on how Cocoa objects are instantiated, referred to, put in communication with one another, and managed over their lifetimes, and a survey of the primary interface widgets and other common tasks. My hope, as with my previous

books, is that you will both read this book cover to cover (learning something new often enough to keep you turning the pages) and keep it by you as a handy reference.

This book is not intended to disparage Apple's own documentation and example projects. They are wonderful resources and have become more wonderful as time goes on. I have depended heavily on them in the preparation of this book. But I also find that they don't fulfill the same function as a reasoned, ordered presentation of the facts. The online documentation must make assumptions as to how much you already know; it can't guarantee that you'll approach it in a given order. And online documentation is more suitable to reference than to instruction. A fully written example, no matter how well commented, is difficult to follow; it demonstrates, but it does not teach.

A book, on the other hand, has numbered chapters and sequential pages; I can assume you know C before you know Objective-C for the simple reason that Chapter 1 precedes Chapter 2. And along with facts, I also bring to the table a degree of experience, which I try to communicate to you. Throughout this book you'll see me referring to "common beginner mistakes"; in most cases, these are mistakes that I have made myself, in addition to seeing others make them. I try to tell you what the pitfalls are because I assume that, in the course of things, you will otherwise fall into them just as naturally as I did as I was learning. You'll also see me construct many examples piece by piece or extract and explain just one tiny portion of a larger app. It is not a massive finished program that teaches programming, but an exposition of the thought process that developed that program. It is this thought process, more than anything else, that I hope you will gain from reading this book.

iOS is huge, massive, immense. It's far too big to be encompassed in a book even of this size. And in any case, that would be inappropriate and unnecessary. There are entire areas of Cocoa Touch that I have ruthlessly avoided discussing. Some of them would require an entire book of their own. Others you can pick up well enough, when the time comes, from the documentation. This book is only a beginning — the fundamentals. But I hope that it will be the firm foundation that will make it easier for you to tackle whatever lies beyond, in your own fun and rewarding iOS programming future.

In closing, some version numbers, so that you know what assumptions I am making. At the time I started writing this book, system versions 3.1.3 (on the iPhone) and 3.2 (on the iPad) were most recent. As I was working on the book, iOS 4 and the iPhone 4 came into being, but it didn't yet run on the iPad. Subsequently iOS 4.2 emerged: the first system able to run on both the iPhone and the iPad. At the same time, Xcode was improved up to 3.2.5.

Then, just in time for my final revisions, Xcode 3.2.6 and iOS 4.3 were released, along with the first public version of the long-awaited Xcode 4. Xcode 4 is a thorough overhaul of the IDE: menus, windows, and preferences are quite different from Xcode 3.2.x. At the same time, both Xcode 4 and Xcode 3.2.x can coexist on the same machine and can be used to work on the same project; moreover, Xcode 3.2.x has some specialized

capabilities that Xcode 4 lacks, so some long-standing developers may well continue to use it. This situation presents a dilemma for an author describing the development process. However, for iOS programming, I recommend adoption of Xcode 4, and this book assumes that you have adopted it.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

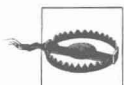
Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*Programming iOS 4* by Matt Neuburg (O’Reilly). Copyright 2011 Matt Neuburg, 978-1-449-38843-0.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Books Online



Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O’Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O’Reilly and other publishers, sign up for free at <http://my.safaribooksonline.com>.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O’Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

<http://oreilly.com/catalog/0636920010258/>

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

Acknowledgments

It's a poor craftsman who blames his tools. No blame attaches to the really great tools by which I have been assisted in the writing of this book. I am particularly grateful to the Unicomp Model M keyboard (<http://pckeyboard.com>), without which I could not have produced so large a book so painlessly. I was also aided by wonderful software, including TextMate (<http://macromates.com>) and AsciiDoc (<http://www.methods.co.nz/asciidoc>). BBEdit (<http://www.barebones.com>) helped with its diff display. Screenshots were created with Snapz Pro X (<http://www.ambrosiasw.com>) and GraphicConverter (<http://www.lemkesoft.com>); diagrams were drawn with OmniGraffle (<http://www.omnigroup.com>).

The splendid O'Reilly production process converted my AsciiDoc text files into PDF while I worked, allowing me to proofread in simulated book format. Were it not for this, and the Early Release program that permitted me to provide my readers with periodic updates of the book as it grew, I would never have agreed to undertake this project in the first place. I would like particularly to thank Tools maven Abby Fox for her constant assistance.

I have taken advice from two tech reviewers, Dave Smith and David Rowland, and have been assisted materially and spiritually by many readers who submitted errata and encouragement. I was particularly fortunate in having Brian Jepson as editor; he provided enthusiasm for the O'Reilly tools and the electronic book formats, a watchful eye, and a trusting attitude; he also endured the role of communications pipeline when I needed to prod various parts of the O'Reilly machine. I have never written an O'Reilly book without the help of Nancy Kotary, and I didn't intend to start now; her sharp eye has smoothed the bristles of my punctuation-laden style. For errors that remain, I take responsibility, of course.

Table of Contents

Preface	xvii
----------------------	-------------

Part I. Language

1. Just Enough C	3
Compilation, Statements, and Comments	4
Variable Declaration, Initialization, and Data Types	6
Structs	8
Pointers	10
Arrays	11
Operators	13
Flow Control and Conditions	15
Functions	19
Pointer Parameters and the Address Operator	22
Files	24
The Standard Library	27
More Preprocessor Directives	27
Data Type Qualifiers	28
2. Object-Based Programming	31
Objects	31
Messages and Methods	32
Classes and Instances	33
Class Methods	36
Instance Variables	37
The Object-Based Philosophy	39
3. Objective-C Objects and Messages	43
An Instance Reference Is a Pointer	43
Instance References, Initialization, and nil	44

Instance References and Assignment	47
Instance References and Memory Management	48
Messages and Methods	49
Sending a Message	50
Declaring a Method	51
Nesting Method Calls	52
No Overloading	52
Parameter Lists	53
Unrecognized Selectors	53
Typecasting and the id Type	55
Messages as Data Type	58
C Functions and Struct Pointers	59
Blocks	61
 4. Objective-C Classes	65
Class and Superclass	65
Interface and Implementation	66
Header File and Implementation File	68
Class Methods	71
The Secret Life of Classes	71
 5. Objective-C Instances	73
How Instances Are Created	73
Ready-Made Instances	73
Instantiation from Scratch	74
Nib-Based Instantiation	77
Polymorphism	78
The Keyword self	79
The Keyword super	82
Instance Variables and Accessors	84
Key-Value Coding	86
Properties	87
How to Write an_INITIALIZER	89
<hr/>	
Part II. IDE	
 6. Anatomy of an Xcode Project	95
New Project	96
The Project Window	97
The Navigator Pane	99
The Utilities Pane	103
The Editor	104

The Project File and Its Dependents	106
The Target	109
Build Phases	109
Build Settings	110
Configurations	111
Schemes and Destinations	112
From Project to App	115
Build Settings	117
Property List Settings	117
Nib Files	118
Other Resources	118
Code	120
Frameworks and SDKs	121
7. Nib Management	125
A Tour of the Nib-Editing Interface	125
The Dock	127
Canvas	128
Inspectors and Libraries	130
Nib Loading and File's Owner	132
Default Instances in the Main Nib File	133
Making and Loading a Nib	134
Outlet Connections	135
More Ways to Create Outlets	139
More About Outlets	141
Action Connections	142
Additional Initialization of Nib-Based Instances	146
8. Documentation	149
The Documentation Window	150
Class Documentation Pages	152
Sample Code	155
Other Resources	156
Quick Help	156
Symbols	157
Header Files	157
Internet Resources	158
9. Life Cycle of a Project	159
Choosing a Device Architecture	159
Localization	162
Editing Your Code	163
Autocompletion	164

Snippets	165
Live Syntax Checking	166
Navigating Your Code	166
Debugging	169
Caveman Debugging	169
The Xcode Debugger	171
Static Analyzer	176
Clean	177
Running in the Simulator	177
Running on a Device	178
Device Management	181
Version Control	181
Instruments	184
Distribution	184
Ad Hoc Distribution	186
Final App Preparations	187
Icons in the App	188
Other Icons	189
Launch Images	189
Screenshots	190
Property List Settings	191
Submission to the App Store	192

Part III. Cocoa

10. Cocoa Classes	197
Subclassing	197
Categories	200
Splitting a Class	201
Private Method Declarations	201
Protocols	202
Optional Methods	206
Some Foundation Classes	208
Useful Structs and Constants	208
NSString and Friends	208
NSDate and Friends	210
NSNumber	211
NSNumber	211
NSData	212
Equality and Comparison	212
NSIndexSet	213
NSArray and NSMutableArray	213

NSSet and Friends	215
NSDictionary and NSMutableDictionary	215
NSNull	217
Immutable and Mutable	217
Property Lists	218
The Secret Life of NSObject	218
11. Cocoa Events	223
Reasons for Events	224
Subclassing	224
Notifications	226
Receiving a Built-In Notification	226
Unregistering	228
NSTimer	228
Delegation	229
Data Sources	232
Actions	233
The Responder Chain	237
Deferring Responsibility	238
Nil-Targeted Actions	238
Application Lifetime Events	239
Swamped by Events	243
12. Accessors and Memory Management	249
Accessors	249
Key–Value Coding	251
Memory Management	254
The Golden Rules of Memory Management	255
How Cocoa Objects Manage Memory	257
Memory Management of Instance Variables	260
Instance Variable Memory Management Policies	263
Autorelease	264
Nib Loading and Memory Management	266
Memory Management Comments on Earlier Examples	267
Memory Management of Pointer-to-Void Context Info	269
Memory Management of C Struct Pointers	270
Properties	271
13. Data Communication	277
Model–View–Controller	277
Instance Visibility	279
Visibility by Instantiation	280
Visibility by Relationship	281

Global Visibility	281
Notifications	282
Key-Value Observing	284

Part IV. Views

14. Views	293
The Window	293
Subview and Superview	295
Frame	298
Bounds and Center	299
Layout	302
Transform	305
Visibility and Opacity	308
15. Drawing	311
UIImage and UIImageView	311
UIImage and Graphics Contexts	313
CGImage	315
Drawing a UIView	318
Graphics Context State	320
Paths	321
Clipping	325
Gradients	326
Colors and Patterns	328
Graphics Context Transforms	330
Shadows	332
Points and Pixels	332
Content Mode	333
16. Layers	335
View and Layer	336
Layers and Sublayers	337
Manipulating the Layer Hierarchy	339
Positioning a Sublayer	339
CASScrollLayer	340
Layout of Sublayers	341
Drawing in a Layer	341
Contents Image	341
Contents on Demand	342
Contents Resizing and Positioning	343
Layers that Draw Themselves	345

Transforms	346
Depth	350
Transforms and Key–Value Coding	352
Shadows, Borders, and More	353
Layers and Key–Value Coding	354
17. Animation	357
Drawing, Animation, and Threading	358
UIImageView Animation	361
View Animation	362
Animation Blocks	362
Modifying an Animation Block	363
Transition Animations	366
Block-Based View Animation	368
Implicit Layer Animation	371
Animation Transactions	372
Media Timing Functions	373
Core Animation	374
CABasicAnimation and Its Inheritance	375
Using a CABasicAnimation	376
Keyframe Animation	379
Making a Property Animatable	380
Grouped Animations	381
Transitions	385
The Animations List	386
Actions	389
What an Action Is	389
The Action Search	390
Hooking Into the Action Search	391
Nonproperty Actions	394
18. Touches	397
Touch Events and Views	398
Receiving Touches	400
Restricting Touches	401
Interpreting Touches	402
Gesture Recognizers	408
Distinguishing Gestures Manually	408
Gesture Recognizer Classes	412
Multiple Gesture Recognizers	416
Subclassing Gesture Recognizers	418
Gesture Recognizer Delegate	419
Touch Delivery	422

Hit-Testing	423
Initial Touch Event Delivery	427
Gesture Recognizer and View	427
Touch Exclusion Logic	429
Recognition	430
Touches and the Responder Chain	431

Part V. Interface

19. View Controllers	435
Creating a View Controller	437
Manual View Controller, Manual View	438
Manual View Controller, Nib View	441
Nib-Instantiated View Controller	443
No View	445
Up-Shifted Root View	446
Rotation	447
Initial Orientation	448
Rotation Events	452
Modal Views	453
Modal View Configuration	454
Modal View Presentation	456
Modal View Dismissal	457
Modal Views and Rotation	459
Tab Bar Controllers	461
Tab Bar Item Images	462
Configuring a Tab Bar Controller	463
Navigation Controllers	464
Bar Button Items	466
Configuring a Navigation Interface	468
Navigation Interface Rotation	474
View Controller Lifetime Events	476
View Controller Memory Management	477
 20. Scroll Views	 481
Creating a Scroll View	482
Scrolling	484
Paging	487
Tiling	488
Zooming	491
Zooming Programmatically	493
Zooming with Detail	493