iPhone® for Programmers
An App-Driven Approach

Contains 14 Fully Coded iPhone® Apps

iPhone® Developer Program • SDK 3.x • Xcode® • Objective-C® • Cocoa®
Interface Builder • App Templates • GUI • Views • Tables • Controllers
Multi-Touch™ • Core Audio • Core Animation • Core Data • Core Location
GPS • Compass • iPod® Library Access • Serialization • Audio/Video
Game Kit • Bluetooth® • Web Services • Collections • Submitting Apps
iTunes® Connect • Great App Design • Pricing • Monetization • And More!

PAUL DEITEL • HARVEY DEITEL
ABBIE DEITEL • ERIC KERN • MICHAEL MORGAN
Welcome to the world of iPhone app development with the iPhone Software Development Kit (SDK) 3.x, the Objective-C® programming language, the Cocoa® frameworks and the Xcode® development tools.

This book presents leading-edge computing technologies for professional software developers. At the heart of the book is our “app-driven approach”—we present concepts in the context of 14 completely coded iPhone apps, rather than using code snippets. The introduction and app test drives at the beginning of each chapter show one or more sample executions. The book’s source code is available at www.deitel.com/books/iPhoneFP/.

Sales of the iPhone and app downloads have been growing explosively. The first-generation iPhone sold 6.1 million units in its initial five quarters of availability.1 The second-generation iPhone 3G sold 6.9 million units in its first quarter alone. The iPhone 3GS, launched in June 2009, sold 5.2 million units in its first month! At the time of this writing, there were approximately 75,000 apps in the App Store, and in just one year, over 1.5 billion apps were downloaded.2 The potential for iPhone apps is enormous.

*iPhone for Programmers* was fun to write! We got to know (and love) the iPhone and many of its most popular apps. Then we let our imaginations run wild as we started developing our own iPhone apps. Some of the apps appear in this book, and some we’ll sell through the iTunes App Store. The book’s apps were carefully designed to introduce you to key iPhone features and frameworks (e.g., audio, video, animation, the compass, peer-to-peer connectivity, GPS and much more). You’ll quickly learn everything you’ll need to start building iPhone apps—starting with a test-drive of the Painter app in Chapter 1, then building your first app in Chapter 3. Chapter 2, iPhone App Store and App Business Issues walks you through what makes a great app, the submission process including uploading your apps for consideration by Apple, criteria for approval, what to expect in the process, why Apple rejects apps, deciding whether to sell your apps or offer them for free, and marketing them using the Internet, word-of-mouth, and so on.

**Copyright Notice and Code License**

This book is copyrighted by Pearson. All of the code and iPhone apps in this book are copyrighted by Deitel & Associates, Inc. *As a user of the book, we grant you the nonexclusive right to copy, distribute, display the code, and create derivative apps based on the code for non-commercial purposes only*—so long as you attribute the code to Deitel & Associates, Inc. and reference www.deitel.com/books/iPhoneFP/. If you have any questions, or specifically would like to use our code for commercial purposes, contact deitel@deitel.com.

---

Intended Audience

We assume that you’re comfortable with Mac OS X, as you’ll need to work on a Mac to develop iPhone apps. We also assume that you’re a programmer with significant experience working in a C-based object-oriented language such as Objective-C, C++, Java or C#. If you have not worked in any of these languages, you should still be able to master iPhone app development and object-oriented programming by reading the code and our code walkthroughs, running the apps and observing the results. You’ll quickly learn a great deal about object-oriented iPhone app development in Objective-C and Cocoa. We overview the basics of object-oriented programming in Chapter 1.

Key Features

Here are some of the book’s key features:

App-Driven Approach. You’ll learn the programming technologies in the context of 14 complete working iPhone apps. Each chapter presents one app—we discuss what the app does, show screen shots, test-drive it and overview the technologies and the architecture you’ll use to build it. Then we build the app, present the complete code and do a detailed code walkthrough. As part of the code walkthrough, we discuss the programming concepts and demonstrate the functionality of the iPhone APIs (application programming interfaces). Figure 1 lists the 14 apps in the book and the key technologies we introduce as we present each.

<table>
<thead>
<tr>
<th>iPhone for Programmers apps and the technologies they introduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 3, Welcome App</td>
</tr>
<tr>
<td>Introducing Xcode, Cocoa and Interface Builder</td>
</tr>
<tr>
<td>Chapter 4, Tip Calculator App</td>
</tr>
<tr>
<td>Introducing Objective-C Programming</td>
</tr>
<tr>
<td>Chapter 5, Favorite Twitter® Searches App</td>
</tr>
<tr>
<td>Collections and Cocoa GUI Programming</td>
</tr>
<tr>
<td>Chapter 6, Flag Quiz Game App</td>
</tr>
<tr>
<td>Controllers and the Utility Application Template</td>
</tr>
<tr>
<td>Chapter 7, Spot-On Game App</td>
</tr>
<tr>
<td>Using UIView and Detecting Touches</td>
</tr>
<tr>
<td>Chapter 8, Cannon Game App</td>
</tr>
<tr>
<td>Animation with NSTimer and Handling Drag Events</td>
</tr>
<tr>
<td>Chapter 9, Painter App</td>
</tr>
<tr>
<td>Using Controls with a UIView</td>
</tr>
<tr>
<td>Chapter 10, Address Book App</td>
</tr>
<tr>
<td>Tables and UINavigationController</td>
</tr>
<tr>
<td>Chapter 11, Route Tracker App</td>
</tr>
<tr>
<td>Map Kit and Core Location (GPS and Compass)</td>
</tr>
<tr>
<td>Chapter 12, Slideshow App</td>
</tr>
<tr>
<td>Photos and iPod Library Access</td>
</tr>
<tr>
<td>Chapter 13, Enhanced Slideshow App</td>
</tr>
<tr>
<td>Saving Data and Playing Video</td>
</tr>
<tr>
<td>Chapter 14, Voice Recorder App</td>
</tr>
<tr>
<td>Audio Recording and Playback</td>
</tr>
<tr>
<td>Chapter 15, Enhanced Address Book App</td>
</tr>
<tr>
<td>Managing and Transferring Persistent Data</td>
</tr>
<tr>
<td>Chapter 16, Twitter® Discount Airfares App</td>
</tr>
<tr>
<td>Internet Enabled Applications</td>
</tr>
</tbody>
</table>

Fig. 1 | iPhone for Programmers apps and the technologies they introduce.

Objective-C. This book is not an Objective-C tutorial, but it teaches a good portion of this object-oriented programming language in the context of iPhone app development.
**Cocoa Frameworks.** Cocoa is the set of frameworks and the runtime environment for the iPhone. Throughout the book, we use many of the Cocoa features and frameworks. (Figure 1.9 in Chapter 1 shows the Cocoa frameworks.)

**iPhone SDK 3.x.** We cover many of the new features included in iPhone Software Development Kid (SDK) 3.x—the Game Kit framework for Bluetooth peer-to-peer connectivity, the Map Kit framework for embedding Google Maps, the Media Player framework for accessing the iPod music library, the Core Location framework for accessing the compass and the Core Data framework for managing app data.

**Xcode.** Apple’s Xcode integrated development environment (IDE) and its associated tools for Mac OS, combined with the iPhone SDK, provide everything you need to develop and test iPhone apps.

**Instruments.** The Instruments tool, which is packaged with the SDK, is used to inspect apps while they’re running to check for memory leaks, monitor CPU usage and network activity, and review the objects allocated in memory. We discuss how we used the Instruments tool to fix memory leaks and performance problems in Chapter 6’s *Flag Quiz Game* App and Chapter 8’s *Cannon Game* App, respectively.

**Multimedia.** The apps use a broad range of iPhone multimedia capabilities, including graphics, images, audio, video, speech synthesis and speech recognition.

**iPhone App Design Patterns.** This book adheres to Apple’s app coding standards, including the Model-View-Controller (MVC) design pattern. (Figure 1.8 in Chapter 1 shows many of the design patterns we use directly or indirectly in the book.)

**Web Services.** Web services enable information sharing, e-commerce and other interactions using standard Internet protocols and technologies. Web services allow you to use the web as a library of reusable software components. Chapter 11’s *Route Tracker* app uses built-in Apple APIs to interact with the Google Maps web services. In Chapter 16’s *Twitter® Discount Airfares* app, you’ll work directly with Twitter’s REST-based web services.

**Uploading Apps to the App Store.** In Chapter 2, iPhone App Store and App Business Issues, we walk you through the process of obtaining development certificates, creating provisioning profiles, submitting your apps to the App Store for approval, deciding whether your app should be free or fee based, marketing it and much more.

### Features

**Syntax Shading.** For readability, we syntax shade the code, similar to Xcode’s use of syntax coloring. Our syntax-shading conventions are as follows:

- comments appear in gray
- keywords appear in bold black
- constants and literal values appear in bold gray
- all other code appears in black

3. *Note:* The *Route Tracker* App uses the Map Kit framework which allows you to incorporate Google™ Maps in your app. Before developing any app using the Map Kit, you must agree to the Google Maps Terms of Service for the iPhone (including the related Legal Notices and Privacy Policy) at: code.google.com/apis/maps/iphone/ terms.html.
Code Highlighting. We use gray rectangles to emphasize the key code segments in each program that exercise the new technologies the program presents.

Using Fonts for Emphasis. We place the defining occurrences of key terms in bold italic text for easier reference. We emphasize on-screen components in the bold Helvetica font (e.g., the Project menu) and emphasize Objective-C and Cocoa program text in the Lucida font (e.g., int x = 5;).

In this book you’ll create GUIs using a combination of visual programming (drag and drop) and writing code. We’ll constantly be referring to GUI elements on the screen. We use different fonts when we refer to GUI components. For example, if a button is part of the IDE, we write the word “button” in lowercase and plain text, as in “Build and Go button.” If on the other hand, it’s a button that we create as part of an app, we use the name Button as it appears in the library of controls you can use in an app. When we refer to a Button’s class, we use the class name UIButton.

Source Code. All of the source-code examples are available for download from:

www.deitel.com/books/iPhoneFP/

Documentation. All of the manuals that you’ll need to develop iPhone apps are available free at developer.apple.com/iphone/.

Chapter Objectives. Each chapter begins with a list of objectives.

Figures. Abundant charts, tables, app source code listings and iPhone screen shots are included.

Index. We include an extensive index, which is especially useful when you use the book as a reference. Defining occurrences of key terms are highlighted with a bold page number.

The Deitel Online Resource Centers

Our website www.deitel.com provides more than 100 Resource Centers on various topics including programming languages, software development, Web 2.0, Internet business and open-source projects—see the list of Resource Centers in the first few pages of this book and visit www.deitel.com/ResourceCenters.html. Each week we announce our latest Resource Centers in our newsletter, the Deitel® Buzz Online (www.deitel.com/newsletter/subscribe.html). The Resource Centers evolve out of the research we do to support our publications and business operations. We’ve found many exceptional iPhone and iPhone programming resources online, including tutorials, documentation, software downloads, articles, blogs, podcasts, videos, code samples, books, e-books and more—most of them are free. Check out the growing list of iPhone-related Resource Centers, including:

- iPhone (www.deitel.com/iPhone/)
- Objective-C (www.deitel.com/ObjectiveC/)
- Cocoa (www.deitel.com/Cocoa/)
- iPhone App Development (www.deitel.com/iPhoneAppDev/)
**Deitel® Buzz Online Free E-mail Newsletter**

The Deitel® Buzz Online e-mail newsletter will keep you posted on issues related to this book. It also includes commentary on industry trends and developments, links to free articles and resources from our published books and upcoming publications, product-release schedules, errata, challenges, anecdotes, information on our corporate instructor-led training courses delivered at client locations worldwide and more. To subscribe, visit

www.deitel.com/newsletter/subscribe.html

**Follow Deitel on Twitter® and Facebook®**

To receive updates on Deitel publications, Resource Centers, training courses, partner offers and more, follow us on Twitter®

@deitel

and join the Deitel & Associates group on Facebook®

www.deitel.com/deitelfan/

**Acknowledgments**

We’re fortunate to have worked on this project with the talented and dedicated team of publishing professionals at Prentice Hall/Pearson. We appreciate the extraordinary efforts and mentorship of Mark L. Taub, Editor-in-Chief of Pearson Technology Group. Sandra Schroeder designed the book’s cover. John Fuller managed the book’s production.

**Reviewers**

We wish to acknowledge the efforts of our reviewers. Adhering to a tight time schedule, they scrutinized the manuscript and the programs and provided constructive suggestions for improving the accuracy and completeness of the presentation:

- Marcantonio Magnarapa, Research & Development on Mobile Platforms, Ogilvy Interactive
- Zach Saul, Founder, Retronyms
- Rik Watson, Senior Software Engineer, Lockheed Martin

Well, there you have it! This book will quickly get you comfortable developing iPhone apps. As you read the book, we’d sincerely appreciate your comments, criticisms, corrections and suggestions for improvement. Please address all correspondence to:

deitel@deitel.com

We’ll respond promptly, and post corrections and clarifications on:

www.deitel.com/books/iPhoneFP/
Preface

We hope you enjoy reading *iPhone for Programmers: An App-Driven Approach* as much as we enjoyed writing it!

Paul Deitel  
Harvey Deitel  
Abbey Deitel  
Eric Kern  
Michael Morgano  
October 2009

**About Deitel & Associates, Inc.**

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring, corporate training and software development organization specializing in computer programming languages, object technology, Internet and web software technology, iPhone app development and training, and Internet business development. The company offers instructor-led courses delivered at client sites worldwide on major programming languages and platforms, such as Objective-C and iPhone app development, C, C++, Visual C++®, Java™, Visual C#®, Visual Basic®, XML®, Python®, object technology, Internet and web programming, and a growing list of additional programming and software-development-related courses. The company's clients include many of the world's largest companies, government agencies, branches of the military, and academic institutions. Through its 33-year publishing partnership with Prentice Hall/Pearson, Deitel & Associates, Inc., publishes leading-edge programming professional books, textbooks, *LiveLessons* DVD- and web-based video courses, and e-content for popular course-management systems. Deitel & Associates, Inc., and the authors can be reached via e-mail at:

deitel@deitel.com

To learn more about Deitel's *Dive Into® Series* Corporate Training curriculum, visit:

www.deitel.com/training/

To request a proposal for on-site, instructor-led training at your company or organization, e-mail:

deitel@deitel.com

To learn more about the company and its publications, subscribe to the free *Deitel® Buzz Online* e-mail newsletter at:

www.deitel.com/newsletter/subscribe.html

Individuals wishing to purchase Deitel books and *LiveLessons* DVD- and web-based training courses can do so through www.deitel.com. Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, visit www.prenhall.com/mischtm/support.html#order.
Painter App
Using Controls with a UIView

OBJECTIVES
In this chapter you’ll learn:

■ How to combine custom views with Cocoa GUI components to create a richer app.
■ How to process multiple screen touches.
■ How to detect when touches move and leave the screen.
■ How to detect motion events to clear the screen when the user shakes the iPhone.
■ How to add variables of primitive and struct types to collections.
9.1 Introduction

The Painter app turns the iPhone screen into a virtual canvas (Fig. 9.1). The user paints by dragging one or more fingers across the screen. The line color and thickness can be set by touching the info button in the lower-right corner of the screen. The control panel (Fig. 9.2) includes a slider for line width and red, green and blue sliders for line color. As the Line Width slider is moved from left to right, the width of the line increases. At the bottom of the screen, two buttons allow the user to turn a finger into an eraser or clear the screen entirely. At any point while painting, the user can shake the iPhone to clear the entire drawing from the screen.

Fig. 9.1 | Painter app and its control panel.

9.2 Overview of the Technologies

The Painter app stores painted lines using the custom Squigg1e class. Each Squigg1e contains an array of points, a UIColor object and a numeric line-width value. When the user touches the screen, a new Squigg1e is created, given a unique key and placed in an NSMut-
ableDictionary. New points are added to the Squiggle as the user drags a finger along the screen. When the touch ends, the Squiggle is transferred from the dictionary to an array of finished Squiggles.

The app uses the Utility Application template. The MainView displays the user’s painting—showing all the finished Squiggles and any Squiggles currently in progress. The user sets the line characteristics in the FlipsideView. The color is set using three Sliders, representing the RGB values of the painted line. We display the currently selected color using a UIView’s backgroundColor property that is updated dynamically as the user moves any of the Sliders. When the user flips from the FlipsideView to the MainView, the values for the color and line width are loaded from the Sliders and passed to the MainView.

9.3 Building the App

To begin, open Xcode and create a new project. Choose the Utility Application template and name the project Painter.

Declaring the Squiggle Interface

Create a new file and name it Squiggle. Squiggle.h declares a class named Squiggle, which represents a single stroke of a finger on the iPhone screen. A Squiggle saves each point touched by the user’s finger between where the first touch occurred and where the finger was finally lifted from the screen. It also saves the color and line width at the time of the stroke—representing all of the information needed to draw the stroke to the screen. Let’s take a look at the interface (Fig. 9.2).

```objc
// Squiggle.h
// Class Squiggle represents the points, color and width of one line.
// Implementation in Squiggle.m
#import <UIKit/UIKit.h>
@interface Squiggle : NSObject
{
    NSMutableArray *points; // the points that make up the Squiggle
    UIColor *strokeColor; // the color of this Squiggle
    float lineWidth; // the line width for this Squiggle
} // end instance variable declaration

// declare strokeColor, lineWidth and points as properties
@property (retain) UIColor* strokeColor;
@property (assign) float lineWidth;
@property (nonatomic, readonly) NSMutableArray *points;
- (void)addPoint:(CGPoint)point; // adds a new point to the Squiggle
@end
```

Fig. 9.2 | Class Squiggle represents the points, color and width of one line.

The points are stored in an NSMutableArray (line 8), and the color, line width and points are stored as properties (lines 14–16). The addPoint: method adds a new point to a Squiggle. We declared the points property as readonly so that other classes can modify the points array only by calling the addPoint: method.
Implementing the Squiggle Class

Class Squiggle (Fig. 9.3) contains the information required to display a Squiggle but it does not define how to draw one. Drawing is handled by the view containing a Squiggle.

```objc
// Squiggle.m
// Squiggle class implementation.
#import "Squiggle.h"

@implementation Squiggle

@synthesize strokeColor; // generate set and get methods for strokeColor
@synthesize lineWidth; // generate set and get methods for lineWidth
@synthesize points; // generate set and get methods for points

// initialize the Squiggle object
- (id)init
{
  // if the superclass properly initializes
  if (self = [super init])
  {
    points = [[NSMutableArray alloc] init]; // initialize points
    strokeColor = [[UIColor blackColor] retain]; // set default color
  } // end if

  return self; // return this object
} // end method init

// add a new point to the Squiggle
- (void)addPoint:(CGPoint)point
{
  // encode the point in an NSValue so we can put it in an NSArray
  NSValue *value = [[NSValue valueWithBytes:&point objCType:@encode(CGPoint)]
  [points addObject:value]; // add the encoded point to the NSArray
} // end method addPoint:

// release Squiggle's memory
- (void)dealloc
{
  [strokeColor release]; // release the strokeColor UIColor
  [points release]; // release the points NSMutableArray
  [super dealloc];
} // end method dealloc
@end
```

Fig. 9.3 | Squiggle class implementation.

Lines 7–9 synthesize `get` and `set` methods for the `strokeColor`, `lineWidth` and `points` properties. The compiler generates only a `get` method for `points` because it’s `readonly`. The `init` method (lines 12–22) initializes a `Squiggle` by allocating the `points` array and setting the `strokeColor` to black (line 18), which is the default color for a `Squiggle`.

The `addPoint:` method adds a new point to the `Squiggle` (lines 25–31). This method takes a `CGPoint` as an argument. You cannot add a `CGPoint` directly to an `NSArray` because
9.3 Building the App

CGPoint is a struct not a class. For this reason, we convert the CGPoint to an NSValue object, which is used as a container to store nonobject types, such as ints, floats, structs and pointers. We perform the conversion using NSValue's `valueWithBytes:objCType:` method (lines 28–29), which takes two arguments—a pointer to the value being encoded and its type. We obtain a pointer to the CGPoint using the & (address of) operator, which returns a pointer to the variable (i.e., its location in memory). The @encode compiler directive converts a type's name to the C string representing the type. This technique can be used when you need to store a nonobject type (such as a primitive value or a struct) in a collection. Line 30 adds the NSValue object to the array. When a Squiggle is removed from memory, the dealloc method releases all of the objects initialized in the init method (lines 34–39).

Declaring the MainView Interface

MainView.h (Fig. 9.4) declares class MainView—a UIView subclass that represents the app's canvas. MainView handles touches, draws the Squiggles and stores the painting.

```
1 // MainView.h
2 // View for the frontside of the Painter app.
3 // Implementation in MainView.m
4 #import <UIKit/UIKit.h>
5 #import "Squiggle.h"
6
7 @interface MainView : UIView
8 {
9     NSMutableDictionary *squiggles; // squiggles in progress
10     NSMutableArray *finishedSquiggles; // finished squiggles
11     UIColor *color; // the current drawing color
12     float lineWidth; // the current drawing line width
13 } // end instance variable declaration
14
15 // declare color and lineWidth as properties
16 @property(nonatomic, retain) UIColor *color;
17 @property float lineWidth;
18
19 // draw the given Squiggle into the given graphics context
20 - (void)drawSquiggle:(Squiggle *)squiggle inContext:(CGContextRef)context;
21 - (void)resetView; // clear all squiggles from the view
22 @end // end interface MainView
```

Fig. 9.4 | View for the frontside of the Painter app.

To display the painting, the MainView stores all the Squiggles on the screen in two data structures—one for Squiggles in progress and one for finished Squiggles (lines 9–10). MainView also stores the current drawing color and line width (lines 11–12). The drawSquiggle:inContext: method displays one Squiggle in the given graphics context, and resetView clears the entire painting.

Implementing the MainView Class

MainView.m (Fig. 9.5) contains class MainView's implementation. Lines 7–8 synthesize properties color and lineWidth (lines 7–8). The initWithCoder: method is called when
the `MainView` is created in a nib file. If the superclass is initialized properly (line 14), we initialize the `squiggles` `NSMutableDictionary` and the `finishedSquiggles` `NSMutableArray` (lines 17–18). The drawing color is initially set to black (line 21) and the line width is initially set to 5 pixels (line 22).

```c
1 // MainView.m
2 // View for the frontside of the Painter app.
3 #import "MainView.h"
4
5 @implementation MainView
6
7 @synthesize color; // generate getters and setters for color
8 @synthesize lineWidth; // generate getters and setters for lineWidth
9
10 // method is called when the view is created in a nib file
11 - (id)initWithCoder:(NSCoder*)decoder
12 {
13     // if the superclass initializes properly
14     if (self = [super initWithCoder:decoder])
15     {
16         // initialize squiggles and finishedSquiggles
17         squiggles = [[NSMutableDictionary alloc] init];
18         finishedSquiggles = [[NSMutableArray alloc] init];
19
20         // the starting color is black
21         color = [[UIColor alloc] initWithRed:0 green:0 blue:0 alpha:1];
22         lineWidth = 5; // default line width
23     } // end if
24
25     return self; // return this object
26 } // end method initWithCoder:
27
```

**Fig. 9.5** Method initWithCoder: of class MainView.

*Methods resetView and drawRect: of Class MainView*

The `resetView` method (Fig. 9.6, lines 29–34) clears the painting from the screen by calling the `removeAllObjects` method on both the `squiggles` dictionary and `finishedSquiggles` array. Calling `UIView`'s `setNeedsDisplay` method (line 33) forces the `MainView` to redraw, thus clearing the screen. The `drawRect:` method draws the entire painting using the stored squiggles. Line 40 retrieves the current graphics context to use for drawing. Then we loop through `finishedSquiggles`, passing each `Squiggle` and the graphics context to the `drawSquiggle:inContext:` method (lines 43–44). Finally, we loop through the `squiggles` `NSMutableDictionary` to draw any `Squiggle`s still in progress (lines 47–51).

```c
28 // clears all the drawings
29 - (void)resetView
30 {
31     [squiggles removeAllObjects]; // clear the dictionary of squiggles
```

**Fig. 9.6** Methods resetView and drawRect: of class MainView. (Part 1 of 2.)
9.3 Building the App

Method `drawSquiggle:inContext:` of Class `MainView`

The `drawSquiggle:inContext:` method receives a `Squiggle` and a graphics context, then draws the `Squiggle` into the graphics context using the `Squiggle`'s color and line width.

```swift
[finishedSquiggles removeAllObjects]; // clear the array of squiggles
[self setNeedsDisplay]; // refresh the display
}
}
```

```
- (void)drawRect:(CGRect)rect {
    // get the current graphics context
    CGContextRef context = UIGraphicsGetCurrentContext();
    // draw all the finished squiggles
    for (Squiggle *squiggle in finishedSquiggles) {
        [self drawSquiggle:squiggle inContext:context];
    }
    // draw all the squiggles currently in progress
    for (NSString *key in squiggles) {
        Squiggle *squiggle = [squiggles valueForKey:key]; // get squiggle
        [self drawSquiggle:squiggle inContext:context]; // draw squiggle
    }
}
```

Method `drawSquiggle:inContext:` of Class `MainView`

```
// draws the given squiggle into the given context
- (void)drawSquiggle:(Squiggle*)squiggle inContext:(CGContextRef)context {
    // set the drawing color to the squiggle's color
    UIColor *squiggleColor = squiggle.strokeColor; // get squiggle's color
    CGColorRef colorRef = [squiggleColor CGColor]; // get the CGColor
    CGContextSetStrokeColorWithColor(context, colorRef);
    // set the line width to the squiggle's line width
    CGContextSetLineWidth(context, squiggle.lineWidth);
    // retrieve the NSValue object and store the value in firstPoint
    CGPoint firstPoint; // declare a CGPoint
    [[points objectAtIndex:0] getValue:&firstPoint];
    // move to the point
    CGContextMoveToPoint(context, firstPoint.x, firstPoint.y);
}
```

Fig. 9.6 | Methods `resetView` and `drawRect:` of class `MainView`. (Part 2 of 2.)

Fig. 9.7 | Method `drawSquiggle:` of class `MainView`. (Part 1 of 2.)
First, the color of the Squiggle is retrieved and set as the current stroke color (lines 58–60). Line 63 then gets the Squiggle’s line width and updates the graphics context with it. Next, we draw the Squiggle. Lines 68–69 get the first point in the Squiggle and move to it. Recall that we added each CGPoint to the points array by storing it in an NSValue object. To retrieve the CGPoint from the NSValue, we use the `getValue:` method, which receives a pointer to where the value will be stored.

Once we move to the first point, we add lines to each of the Squiggle’s remaining points in sequence (lines 72–83). We get the next NSValue (line 77), get the CGPoint contained in the NSValue (lines 78–79) and add a line to the CGPoint (line 82). We then call the CGContextStrokePath function (line 85) to draw the Squiggle we just defined.

**Touch-Handling Methods of Class MainView**
The next three methods defined in MainView.m perform touch handling (Fig. 9.8). The method `touchesBegan:withEvent:` is called when the user touches the screen, `touchesMoved:withEvent:` is called when the user drags a finger and `touchesEnded:withEvent:` is called when the user lifts a finger.

```objective-c
// called whenever the user places a finger on the screen
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    NSArray *array = [touches allObjects]; // get all the new touches
    // loop through each new touch
    for (UITouch *touch in array)
    {
        // create and configure a new squiggle
        Squiggle *squiggle = [[Squiggle alloc] init];
        [squiggle setStrokeColor:color]; // set squiggle's stroke color
        [squiggle setLineWidth:lineWidth]; // set squiggle's line width
    }
}
```

---

**Fig. 9.7** | Method `drawSquiggle:` of class MainView. (Part 2 of 2.)

**Fig. 9.8** | Touch-handling methods of class MainView. (Part 1 of 3.)
Building the App

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event

- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event

Fig. 9.8 | Touch-handling methods of class MainView. (Part 2 of 3.)
In touchesBegan:withEvent:, we first get all the new touches by using the allObjects method of NSSet (line 91). This method returns an NSArray containing all the UITouch objects in the NSSet. We then loop through all the new touches (lines 94–111). For each touch, we create a new Squiggle and add it to the dictionary under a unique key. For the entire duration of a touch (from when it begins to when it ends), we are always guaranteed to be passed the same UITouch object in our touch-handling methods. So, we can use the memory address of the UITouch object as the key for the new Squiggle. We create the new Squiggle (line 97), customize it (lines 98–99) and add its first point (line 100). We then create the key (105–106). We use the valueWithPointer: method of NSValue to convert the memory address of the UITouch into an object (line 105). We then convert the NSValue to an NSString (line 106) and store the Squiggle in the dictionary using the NSString as the key (line 109).

In the touchedMoved:withEvent: method (lines 115–148), we add new points to the Squiggles in the squiggles dictionary for each touch that moved. For each moved touch, we get the unique key for that touch (line 123), then get the Squiggle using that key (lines 126–127). We then get the point the touch was moved to (line 130) and add it to the Squiggle (line 132).

Now that the Squiggle is updated, we need to update the view to draw the new line (lines 137–146). We could use the setNeedsDisplay method to redraw the entire view, but this is inefficient because only a portion of the view is changing. Instead, we use the setNeedsDisplayInRect: method (lines 144–146) to tell the view to update the display only in the area defined by the CGRect argument. To determine the CGRect that encloses the line segment, we first calculate the upper-left and bottom-right corners of the CGRect (lines 137–141) using the ?: (conditional) operator, which takes three arguments. The first is a condition. The second is the value for the entire expression if the condition is true, and the third is the value for the entire expression if the condition is false. Once we calculate the points, we use them, along with some padding on either side to account for the line’s thickness, to create the CGRect (lines 144–146).
In the `touchesEnded:withEvent:` method (lines 151–168), we transfer the SquiggleS that correspond to the finished touches from the `NSMutableDictionary` of SquiggleS in progress to the `NSMutableArray` of finished SquiggleS. We loop through each finished touch (lines 154–167), and for each touch we get its corresponding Squiggle, using the touch’s memory address as the key (157–161). We then add this Squiggle to the finishedSquiggles `NSMutableArray` (line 165) and remove it from the squiggles `NSMutableDictionary` (line 166).

Methods `motionEnded:withEvent:`, `alertView:clickedButtonAtIndex:`, `canBecomeFirstResponder` and dealloc of Class `MainView`:

The next three methods in `MainView` (Fig. 9.9) clear the painting when the user shakes the iPhone. The method `motionEnded:withEvent:` is called when the user finishes a motion event, such as a shake. If the ended event was a shake (line 174), we display an alert asking whether the user really wanted to erase the painting (lines 177–182). The `alertView:clickedButtonAtIndex:` method is called when the user touches one of the buttons in the alert. If the user touched the button labeled Clear (line 194), we clear the entire painting (line 195). The `canBecomeFirstResponder` method is called to determine whether an object of this class can become the first responder. Only the first responder receives notifications about motion events, so we need `MainView` to be the first responder. We return `YES` (line 201) to enable this.

```
170 // called when a motion event, such as a shake, ends
171 - (void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event
172 {
173    // if a shake event ended
174    if (event.subtype == UIEventSubtypeMotionShake)
175    {
176        // create an alert prompting the user about clearing the painting
177        NSString *message = @"Are you sure you want to clear the painting?";
178        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:
179            @"Clear painting" message:message delegate:self
180            cancelButtonTitle:@"Cancel" otherButtonTitles:@"Clear", nil];
181        [alert show]; // show the alert
182        [alert release]; // release the alert UIAlertView
183    } // end if
184    // call the superclass's moetionEnded:withEvent: method
185    [super motionEnded:motion withEvent:event];
186 } // end method motionEnded:withEvent:
187
188 // clear the painting if the user touched the "Clear" button
189 - (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
190    (NSInteger)buttonIndex
191 {
192    // if the user touched the Clear button
193    if (buttonIndex == 1)
194        [self resetView]; // clear the screen
195 } // end method alertView:clickedButtonAtIndex:
```

Fig. 9.9 | Methods `motionEnded:withEvent:`, `alertView:clickedButtonAtIndex:`, `canBecomeFirstResponder` and dealloc of class `MainView`. (Part 1 of 2.)
Chapter 9 Painter App

Declaring the MainViewController Interface

MainViewController.h (Fig. 9.10) defines the class MainViewController, a subclass of UIViewController. This class is the controller for the frontside of our app. Its main functions are to show the flipside when the info button is touched and to pass messages from the flipside to MainView. We declare the MainViewController class as a subclass of UIViewController (line 6).

MainViewController also conforms to the FlipsideViewControllerDelegate protocol, which is defined in FlipsideViewController.h. The showInfo: method creates a new FlipsideViewController and displays it when the info button is touched (line 11).

```
// MainViewController.h
// Controller for the front side of the Painter app.
// Implementation in MainViewController.m
#import "FlipsideViewController.h"

@interface MainViewController : UIViewController
<FlipsideViewControllerDelegate>
@end
```

Implementing the MainViewController Class

MainViewController.m (Fig. 9.11) provides the definition of class MainViewController. The viewDidLoad: and didReceiveMemoryWarning methods (lines 9–20) are inherited from UIViewController. They are called when MainView is going to be shown or hidden, respectively. For MainView to receive notifications about motion events, it must be the first responder. These notifications are necessary for the “shake to erase” feature to work. We

```
// determines if this view can become the first responder
-(BOOL)canBecomeFirstResponder
{
    return YES; // this view can be the first responder
}
@end

@end
```

```
-(void)dealloc
{
    [squiggles release]; // release the squiggles NSDictionary
    [finishedSquiggles release]; // release finishedSquiggles
    [color release]; // release the color UIColor
    [super dealloc];
}
```

```
// MainViewController.h
// Controller for the front side of the Painter app.
// Implementation in MainViewController.m
#import "FlipsideViewController.h"

@interface MainViewController : UIViewController
<FlipsideViewControllerDelegate>
@end
```

```
@end
```
don’t want MainView to be the first responder when it’s hidden, so we make it the first responder when it appears by using the becomeFirstResponder method (line 12). We then remove the first-responder status when the MainView disappears by using the resignFirstResponder method (line 19).

```
1 // MainViewController.m
2 // Controller for the front side of the Painter app.
3 #import "MainViewController.h"
4 #import "MainView.h"
5
6 @implementation MainViewController
7
8 // make the main view the first responder
9 - (void)viewDidAppear:(BOOL)animated
10 {
11     [super viewDidAppear:animated]; // pass message to superclass
12     [self.view becomeFirstResponder]; // make main view the first responder
13 } // end method viewDidAppear
14
15 // resign the main view as the first responder
16 - (void)viewDidDisappear:(BOOL)animated
17 {
18     [super viewDidDisappear:animated]; // pass message to superclass
19     [self.view resignFirstResponder]; // resign view as first responder
20 } // end method viewDidDisappear:
21
22 // called when the Done button on the flipside is touched
23 - (void)flipsideViewControllerDidFinish:(FlipsideViewController *)c
24 {
25     // make the app flip back to the main view
26     [self dismissModalViewControllerAnimated:YES];
27 } // end method flipsideViewControllerDidFinish:
28
29 // called when the info button is touched
30 - (IBAction)showInfo
31 {
32     // load a new FlipsideViewController from FlipsideView.xib
33     FlipsideViewController *controller = [[FlipsideViewController alloc]
34     initWithNibName:@"FlipsideView" bundle:nil];
35
36     controller.delegate = self; // set the delegate of controller
37     
38     // set the animation effect and show the flipside
39     controller.modalTransitionStyle = UIAlcoholTransitioaStyleFlipHorizontal;
40     [self presentModalViewController:controller animated:YES];
41
42     // set the sliders on the flipside to the current values in view
43     MainView *view = (MainView *)self.view;
44     [controller setColor:view.color lineWidth:view.lineWidth];
45     [controller release]; // we are done with controller so release it
46 } // end method showInfo
```

**Fig. 9.11** Controller for the front side of the Painter app. (Part 1 of 2.)
The flipsideViewControllerDidFinish: method (lines 23–27) is called when the user touches the “Done” Button on the FlipSideView. The showInfo method (lines 30–46) switches to the FlipsideView when the info button is touched. Lines (33–34) create a new FlipsideViewController, setting the view it controls to FlipsideView.xib. This is accessed via the controller pointer. We then set controller’s delegate property to self—allowing the FlipsideViewController to access MainViewController’s methods and properties. Line 39 sets controller’s modalTransitionStyle property (inherited from UIViewController) to UIModalTransitionStyleFlipHorizontal. This makes it flip horizontally between the MainView and the FlipsideView.

Line 43 gets a pointer to the MainView. Line 44 calls controller’s setColor:lineWidth: method, passing the MainView’s color and lineWidth properties as arguments. This initializes the FlipsideView’s GUI components to match the current painted line’s color and width. Line 45 releases controller, because it’s no longer needed by the MainViewController.

The setColor: method (lines 49–53) takes a UIColor—retrieving the MainView and setting its color property to the given UIColor. The setLineWidth method (lines 56–60) sets MainView’s lineWidth property in a similar manner. The resetView method (lines 63–67) simply calls the MainView’s resetView method.

**Declaring the FlipsideViewController Interface**

FlipsideViewController.h (Fig. 9.12) declares the FlipsideViewController class, which is a UIViewController subclass that controls the flipside of our app. Line 8 declares instance variable delegate (line 8), which is of type id and implements the FlipsideViewControllerDelegate protocol. This is the object that will receive a message when the user touches the “Done” Button. We next declare five outlets that will be connected to GUI components in Interface Builder. Four UISliders represent the Sliders used
to set the color and width of the painted line (lines 9–13). The UIView shows a preview of
the painting color. The clearScreen variable tracks whether the user has touched the
“Clear Screen” Button.

```objective-c
@protocol FlipsideViewControllerDelegate; // declare a new protocol

@interface FlipsideViewController : UIViewController
{
    id <FlipsideViewControllerDelegate> delegate; // this class's delegate
    IBOutlet UISlider *redSlider; // slider for changing amount of red
    IBOutlet UISlider *greenSlider; // slider for changing amount of green
    IBOutlet UISlider *blueSlider; // slider for changing amount of blue
    IBOutlet UISlider *widthSlider; // slider for changing line width
    IBOutlet UIView *colorView; // view that displays the current color
    BOOL clearScreen; // was the Clear Screen button touched?
} // end instance variable declaration

// declare delegate and outlets as properties
@property (nonatomic, assign) id <FlipsideViewControllerDelegate> delegate;
@property (nonatomic, retain) IBOutlet UISlider *redSlider;
@property (nonatomic, retain) IBOutlet UISlider *greenSlider;
@property (nonatomic, retain) IBOutlet UISlider *blueSlider;
@property (nonatomic, retain) IBOutlet UISlider *widthSlider;
@property (nonatomic, retain) IBOutlet UIView *colorView;

- (IBAction)done; // called when the Done button is touched
- (IBAction)updateColor:sender; // called when a color slider is moved
- (IBAction)erase:sender; // called when the Erase button is touched
- (IBAction)clearScreen:sender; // called by Clear Screen button

// sets the color and line width
- (void)setColor:(UIColor *)c lineWidth:(float)width;
@end // end interface FlipsideViewController

// protocol that the delegate implements
@protocol FlipsideViewControllerDelegate
- (void)flipsideViewControllerDidFinish:(FlipsideViewController *)controller;
- (void)setColor:(UIColor *)color; // sets the current drawing color
- (void)setLineWidth:(float)width; // sets the current drawing line width
- (void)resetView; // erases the entire painting
@end // end protocol FlipsideViewControllerDelegate
```

Fig. 9.12 | FlipsideViewController interface.

The FlipsideViewController class has five methods:

- **done** returns the user to the **MainView** when the “Done” Button is touched.
- **updateColor** updates the UIView previewing the chosen color when any of the
color Sliders’ thumbs are moved.
erase sets the color of the painted line to white when the “Eraser” Button is
touched. The Sliders move to the right to reflect the change.

• clearColor:sender: is called when the “Clear Screen” Button is touched and
causes the painting to be erased when the app returns to the MainView.

• setColor:lineWidth: sets the Sliders’ thumb positions to match the current col-
or and width of the painted line.

Implementing the FlipsideViewController Class
FlipsideViewController.m (Fig. 9.13) defines the FlipsideViewController class. The
viewDidLoad method (lines 16–20) initializes FlipsideViewController’s instance vari-
bables when its view loads. We set the view’s backgroundColor property to the default UI-
Color used for flipside views.

```objective-c
// Fig. 9.13: FlipsideViewController.m
// Controller for the flipside of the Painter app.
#import "FlipsideViewController.h"
#import "MainViewController.h"

@implementation FlipsideViewController

@synthesize delegate; // generate getter and setter for delegate
@synthesize redSlider; // generate getter and setter for redSlider
@synthesize greenSlider; // generate getter and setter for greenSlider
@synthesize blueSlider; // generate getter and setter for blueSlider
@synthesize widthSlider; // generate getter and setter for widthSlider
@synthesize colorView; // generate getter and setter for colorView

// called when view finishes loading
- (void)viewDidLoad
{
    // initialize the background color to the default
    self.view.backgroundColor = [UIColor viewFlipsideBackgroundColor];
} // end method viewDidLoad

// called when view is going to be displayed
- (void)viewWillAppear:(BOOL)animated
{
    // reset clearColor
    clearColor = NO;
} // end method viewWillAppear:

// set the values for color and lineWidth
- (void)setColor:(UIColor *)c lineWidth:(float)width
{
    // split the passed color into its RGB components
    const float *colors = CGColorGetComponents(c.CGColor);

    // update the sliders with the new value
    redSlider.value = colors[0]; // set the red slider's value
}
```

Fig. 9.13 | FlipsideViewController class. (Part 1 of 3.)
9.3 Building the App

```objective-c
37 greenSlider.value = colors[1]; // set the green slider's value
38 blueSlider.value = colors[2]; // set the blue slider's value
39
40 // update the color of colorView to reflect the sliders
41 colorView.backgroundColor = c;
42
43 // update the width slider
44 widthSlider.value = width;
45 } // end method setColor:lineWidth:
46
47 // called when any of the color sliders are changed
48 - (IBAction)updateColor:sender
49 {
50 // get the color from the sliders
51 UIColor *color = [UIColor colorWithRed:redSlider.value
52 green:greenSlider.value blue:blueSlider.value alpha:1.0];
53
54 // update colorView to reflect the new slider values
55 [colorView setBackgroundColor:color];
56 } // end method updateColor:
57
58 // called when the Eraser button is touched
59 - (IBAction)erase:sender
60 {
61 // do all the changes in an animation block so all the sliders finish
62 // moving at the same time
63 [UIView beginAnimations:nil context:nil]; // begin animation block
64 [UIView setAnimationDuration:0.5]; // set the animation length
65
66 // set all sliders to their max value so the color is white
67 [redSlider setValue:1.0]; // set the red slider's value to 1
68 [greenSlider setValue:1.0]; // set the green slider's value to 1
69 [blueSlider setValue:1.0]; // set the blue slider's value to 1
70
71 // update colorView to reflect the new slider values
72 [colorView setBackgroundColor:[UIColor whiteColor]];
73 [UIView commitAnimations]; // end animation block
74 } // end method erase
75
76 // called when the Clear Screen button is touched
77 - (IBAction)clearScreen:sender
78 {
79 clearScreen = YES; // set clearScreen to YES
80 } // end method clearScreen:
81
82 // called when the Done button is touched
83 - (IBAction)done
84 {
85 // set the new values for color and line width
86 [self.delegate setColor:colorView.backgroundColor];
87 [self.delegate setLineWidth:widthSlider.value];
88
Fig. 9.13 | FlipsideViewController class. (Part 2 of 3.)
```
The `viewWillAppear` method (lines 23–27) is called when the `FlipsideView` is about to be displayed. The method resets `clearScreen` to `NO`. We call the superclass’s `viewWillAppear:` method (line 25) to ensure that the `UIView` is ready to be displayed.

The `setColor:lineWidth:` method (lines 30–45) is used to update the GUI components on the flipside to match the current appearance of the painted line. Remember, a new `FlipsideViewController` is created every time the user touches the info button, but we want to save the settings through each one. The `CGColorGetComponents` function breaks down a `CGColor` into an array of its RGB values (line 33). Lines 36–38 update each `Slider`’s `value` property to the appropriate colors—moving the thumbs to their proper locations. The `colorView UIView`’s `backgroundColor` is updated to display the current color of the painted line and `widthSlider`’s `value` is updated to the current width (lines 41 and 44).

The `updateColor` method (lines 48–56) is called to update `colorView` each time a `Slider`’s thumbs is moved. We create a new `UIColor` object using the values of the `Slider` (lines 51–52). We then update the background color of `colorView` to reflect the new color.

The `erase` method (lines 59–74) sets each color `Slider`’s `value` property to one—setting the color of the painted line to white. The `Slider`’s thumbs are moved to their new positions using animation. Line 63 begins a new Core Animation block by calling `UIView`’s `beginAnimations:context:` method. The `setAnimationDuration:` method specifies that the animation will last half a second. Lines 67–69 set all of the `Slider`’s values to 1.0 using `UISlider`’s `setValue:` method. The `colorView UIView` is then updated to display the color white. Line 73 calls `UIView`’s `commitAnimations` method to end the animation block and start the animation.

The `clearScreen:` method (lines 77–80) sets `clearScreen` to `YES` when the “Clear Screen” Button is touched. This causes the painting to clear when the user switches back to the `MainView`.
The done method (lines 83–95) is called when the user touches the “Done” Button. We then call the delegate’s `setColor` method—setting the color of the painted line equal to `colorView`’s `backgroundColor` property (line 86). Line 87 sets the painted line width equal to the value of `widthSlider` using the delegate’s `setLineWidth` method. If the “Clear Screen” Button was touched (line 90), we call the delegates’s `resetView` method to erase the current painting. `MainViewController`’s `flipsideViewControllerDidFinish:` method returns the app to the `MainView`.

**Building the Flipside View**

The interface for the flipside view is contained in the file `FlipsideView.xib`. The flipside view contains components used to set the width and the color of the painted line. Begin by changing `Title` to `Painter`, then add a `Slider` for changing the line width and a `Label` to describe it. Select the `Slider` and open the `Inspector`. Change `Minimum` to 1.0, `Maximum` to 20.0 and `Current` to 5.0. Drag three more `Sliders` to set the RGB values of the painted line. In the `Inspector` check the checkbox `Continuous` for each one. This makes the `Slider` send events every time it’s moved, rather than once only when it stops moving. Add a `Button` titled `Clear Screen` to allow the user to erase the canvas, and add a `Button` titled `Eraser` which will turn the painted line into an eraser. The finished interface is shown in Fig. 9.14.

![Fig. 9.14 | The finished flipside interface.](image)

Next, connect the outlets and actions as we discussed in Section 4.6. In the `FlipsideView.xib` window, the `FlipsideViewController` object is represented by `File’s Owner`. Select this object and connect its outlets as labeled in Fig. 9.14. Next, select the three color `Sliders` and connect their `Value Changed` event to the `updateColor:` method of `File’s Owner`. Also connect the “Eraser” Button’s `Touch Up Inside` event to the `erase:` method and the “Clear Screen” Button’s event to the `clearScreen:` method.

### 9.4 Wrap-Up

In the `Painter` app, you learned more about how custom `UIView`s and `UIViewController`s interact. We saw how to handle all three types of touch events, along with motion events generated when the user shakes the iPhone. We also saw how to store primitives and struc-
tures in collections using the NSValue class, and how to selectively redraw a UIView to optimize the app’s performance.

In Chapter 10, we build the Address Book app. We introduce the Table View component to display a list of information. We show the different kinds of Table Views and how to populate them with information. We also introduce Navigation Controllers, which are used to manage a hierarchy of Views and are usually used in conjunction with Table Views. Both of these new classes are used in the context of the Navigation-based Application template.
Symbols
!= operator 80
?: operator 182
(CSR) Certificate Signing Request 26
"string" NSString literal 80
@encode compiler directive 177
@selector 98
@synthesize directive 176
* operator 81
/ operator 81
& (address of) operator 177
# preprocessor operator 72
#import macro 72
% operator 81
- operator 81
+ operator 81
< operator 81
<= operator 81
== operator 80
> operator 81
>= operator 81

Numerics
148Apps app review site 46

A
absoluteString method of class NSURL 297
Abstract Factory design pattern 10
abstract factory design pattern 91
accelerometer 6
acceptConnectionFromPeer:didFailWithError: method of class GKSession 354
access a property with dot (.) notation 79
accessibility 7, 32

Accessibility Programming Guide for iPhone OS 35
accessoryView property of class UITableView 337
action 73
actionSheet:clickedButtonAtIndex: of protocol UIToolbar 272
Ad 25
Ad Hoc distribution 25, 29, 32
addAttachmentData:mimeType:fileName: method of class MFMailComposeViewController 340
addition 81
addObject: method of class NSMutableArray 248, 292
Address Book 12
Address Book app xxx, 15
Address Book UI 11
address of (&) operator 177
addSubview: method of class UIView 233
addTarget:action:forControlEvents: method of a GUI component 98, 100
addTarget:action:forControlEvents: of class UIControl 216
AdMob 37, 47
advertising revenue 37
AdWhirl 37, 47
alertView:clickedButtonAtIndex: of protocol UIAlertViewDelegate 183
allKeys method of class NSDictionary 350
allObjects of class NSSet 182
allowMetering property of class AVAudioRecorder 320
allowsImageEditing property of class UIImageViewPickerController 270, 295
allowsPickingMultipleItems property of class MPMediaPickerController 272
alpha property of class UIImage 260
alpha transparency 20
alphabetical order 84
altitude 224
Amazon Mobile app 37
Android 49
Anecdotes 48
animation 132
manually perform with timer events 156
API 11
apiwiki.twitter.com/ 49
app xxxv
app approval process 24
app delegate 302
app development xxxv
app distribution 29
App ID 25, 27
app review sites
148Apps 46
AppCraver 46
Apple iPhone School 46
Appletell 46
Apptism 46
AppVee 46
Ars Technica 46
Fresh Apps 46
Gizmodo 46
iPhone App reviews 45
iPhone Toolbox 46
iusethis 46
Macworld 46
app review sites (cont.)
  The App Podcast 46
  What's on iPhone 45
App Store xxxi, xxxv, 3, 7, 24, 25, 37, 41, 45
  Books category 7
  Business category 7
  Education category 7
  Entertainment category 7
  Finance category 7
  Games category 7
  Healthcare and Fitness category 7
  Lifestyle category 7
  Medical category 7
  Music category 7
  Navigation category 7
  News category 8
  Photography category 8
  Productivity category 8
  Reference category 8
  Social Networking category 8
  Sports category 8
  Travel category 8
  Utilities category 8
  Weather category 8

App Store 5
App Store distribution 25, 29, 32
AppCraver app review site 46
app-driven approach xxx, 2
Apple developer account xxxv
Apple Inc. 9
Apple iPhone School app review site 46
Apple Macintosh 9
Apple online documentation 2
Apple Push Notification 13, 32
Appletell app review site 46
applicationMusicPlayer method of class MPMusicPlayerController 262
apps
  Amazon Mobile 37
  Bank of America 37
  Comcast Mobile 37
  ESPN ScoreCenter 37
  Nationwide Mobile 37
  Apptism app review site 46
AppVee app review site 46
archivedDataWithRootObject: method of class NSKeyedArchiver 348
archiveRootObject:toFile: method of class NSKeyedArchiver 303
archiving 282
arithmetic operators 81
Ars Technica app review site 46
arstechnica.com/apple/iphone/apps/46
assign keyword 124
association 17
Atom format 366
attribute 16
attribute of an entity (Core Data) 345
audio xxxi
audio book 14
audio messages 9
Audio Toolbox 12
Audio Unit 12
audiobooks 9
autofocus camera 6
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease method of class NSObject 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease method of class NSObject 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autorelease pool 122
autorelease message 122
autorelease method of class NSObject 122
autoreleasepool: method of class NSObject 122
arcViewRootObject:toFile: method of class NSKeyedArchiver 303
AVFoundation 12
AV Foundation framework 314
available property of class GKSession 352
availableMediaTypesForSourceType method of class UIImagePickerController 295
AVAudioPlayer class 132, 133, 158, 314, 334
currentTime property 333
pause method 335
volume property 332
AVAudioRecorder class 315
allowMetering property 320
averagePowerForChannel: method 321
AVAudioRecorder class (cont.)
prepareToRecord method 320
record method 320
updateMeters method 321
AVAudioSession class 314, 334
category property 316
setCategory: method 317, 319
sharedInstance method 316
AVAudioSession class
setCategory: method 334
AVAudioSessionCategory-Playback 334
AVAudioSessionCategory-Playback class 314
AVAudioSessionCategory-Record 319
AVAudioSessionCategory-Record class 314
AVAudioSessionCategory-SoloAmbient 317, 335
averagePowerForChannel: method of class AVAudioRecorder 321
AVFoundation framework 132
awakeFromNib message 79
awakeFromNib method 78
backBarButtonItem property of class UINavigationController 198
backgroundColor of class UIView 190
backgroundColor property of class UIView 191, 226
Bank of America app 37
Bar Button Item 86
becomeFirstResponder method of a GUI component 79
becomeFirstResponder method of class UITextField 286, 324
becomeFirstResponder method of class UIViewController 185
Before You Begin xxxv
beginAnimations:context:method of class UIView 138, 190, 208
behavior 16
binary 40
Bing 49
bitwise OR operator 259, 305
BlackBerry 49
blog.wundrbar.com/ 48
Blogger 44
blogging 44
Bluetooth 7, 13, 14
brand awareness 37
Build and Debug button (Xcode) 18, 55
Build and Go button (Xcode) 18, 55, 61
Build and Run button (Xcode) 18, 55
Bundle Indentifier 27
Bundle Programming Guide 34, 50
Bundle Seed ID 27
Button 18, 86

C
C# xxx
C++ xxx
CALayer class 132, 141, 145
presentationLayermethod 141
removeAllAnimations method 146
Calculator 5
CalDAV 9
Calendar 9
Calendar 5
call a function after a specified delay 132
Camera 5
camera 4, 6
camera, autofocus 6
canBecomeFirstResponder of class UIResponder 183
Cannon Game app xxx, xxxi, 12, 15
category 91, 101, 121, 198, 204, 257, 288
ehance an existing class 91
category (cont.)methods added to a class at runtime 91
category property of class AVAudioSession 316
Certificate Signing Request (CSR) 26
Certificates 26, 27, 29
CFNetwork 13
CGAffineTransformIdentity 237
CGColorGetComponents function 190
CGContext class 168
CGContextAddLineToPoint function of CGContext Reference 229
CGContextDrawImage function of CGContext Reference 169
CGContextMoveToPoint function of CGContext Reference 169, 229, 328
CGContextRestoreCGState function of the CoreGraphics framework 230
CGContextRotateCTM function of the CoreGraphics framework 230
CGContextScaleCTM function 168
CGContextSelectFont function of CGContext Reference 169
CGContextSetLineWidth function of the CoreGraphics framework 227
CGContextSetRGBFillColor function of CGContextReference 169
CGContextSetRGBStrokeColor function of the CoreGraphics framework 230
CGContextShowTextAtPoint function of CGContextReference 169
CGContextStrokePath function 180
CGContextStrokePath function of CGContext Reference 169, 328
CGContextTranslateCTM function of the CoreGraphics framework 230
CGImage class 169
CGImage property of class UIView 169
CGMakeRect function of CGGeometry Reference 169
CGPoint 96, 176
CGPoint class 227, 229
CGPointMake function of CGGeometry Reference 169
CGRect class 96, 169, 257, 263
CGRectMake function of CGContext Reference 169
CGSize class 96, 99, 327
CGSizeMake 99
Chain-of-Responsibility design pattern 10, 141
characteristics of great apps 35
chat 14
choosing photos from the iPhone’s photo library 245
Chrome 49
class 16
interface 72
Class Actions 91
class cluster 91
class declaration 65
class implementation 65
class library 10
Class Outlets 91
Classes
AVAudioPlayer 132, 133, 158, 314, 334
AVAudioRecorder 315
AVAudioSession 314, 334
Classes (cont.)
- AVAudioSessionCategory-Playback 314
- AVAudioSessionCategory-Record 314
- CALayer 132, 141, 145
- CGContext 168
- CGImage 169
- CGPoint 227, 229
- CGRect 96, 169, 257, 263
- CGSize 96, 99, 327
- CLHeading 224, 232
- CLLocation 224, 226, 229, 230
- CLLocationManager 224, 232, 236, 237
- FetchedResultsController 345
- FlipsideViewController 188
- GKPeerPickerController 345, 346, 347
- GKSession 345, 348, 352, 354
- MFMailComposeViewController 314, 340
- MKAnnotationView 236
- MKMapView 224, 226
- MPMediaItemCollection 245, 257, 267, 287
- MPMediaPickerController 245, 267, 272
- MPMoviePlayerController 282
- MPMusicPlayer 262
- MPMusicPlayerController 245, 257, 262
- NSArray 91, 207
- NSAutoreleasePool 122
- NSBundle 112
- NSCoder 282, 284, 288
- NSDatalist 339, 348, 355
- NSDate 224, 232
- NSDictionary 90, 98, 200, 296, 348, 350, 355
- NSEntityDescription 345, 348, 350, 355
- NSError 352
- Classes (cont.)
- NSFetchedResultsController 351, 352, 355, 358
- NSFetchRequest 345
- NSFileManager 94, 198, 298, 332
- NSIndexPath 202, 214, 249, 251, 293, 300, 350, 358
- NSKeyedArchiver 282, 303, 348
- NSKeyedUnarchiver 292
- NSLocale 80
- NSManagedObject 345, 347, 348, 358
- NSManagedObjectContext 345, 352
- NSMutableArray 86, 91, 133, 178, 198, 226, 246, 252, 288, 292, 326, 327
- NSMutableData 373
- NSMutableDictionary 86, 90, 178, 200, 348, 355
- NSNotificationCenter 308
- NSNumber 111, 137, 178, 200, 326, 328
- NSNumberFormatter 80
- NSObject 112
- NSPredicate 314, 325
- NSSet 140
- NSSortDescriptor 362
- NSString 73, 297, 332
- NSTimer 156, 158, 161, 262, 315
- NSURL 100, 297, 298, 372
- NSURLConnection 366, 374
- NSURLRequest 372
- NSUserDefaults 137
- NSError 177
- NSXMLParser 366, 378
- UIActionSheet 245, 267, 299
- UIActivityIndicatorView 368
- UIAlertView 118, 353
- UIApplication 259, 262, 302
- UIBarButtonItem 198, 245, 247, 267, 334, 347, 369
- UIButton 97, 253, 315
- Classes (cont.)
- UIColor 174, 188, 226
- UIImage 132, 169, 249, 258, 288, 294, 296, 305
- UIImagePickerController 245, 267, 270, 282, 295
- UIImageView 53, 59, 132, 257, 259, 262, 305
- UILabel 53, 350
- UINavigationController 245, 247, 248, 251, 259, 293, 359
- UIPickerView 247, 349, 352
- UIScrollView 88
- UISlider 190
- UITableView 196, 201, 202, 214, 247, 248, 249, 293, 296, 332, 350, 351, 355, 366
- UITableViewCell 197, 210, 249, 293, 300, 350, 358, 366
- UITableViewEditingStyle 203
- UITableViewCell 198, 246
- UITextField 216, 286, 322, 324
- UIKeyboard 245, 267
- UITouch 132, 140, 182
- UIView 178, 187, 190, 198, 225, 233, 257, 258, 262, 325
- UIImagePickerController 245, 247, 251, 259, 293, 359
- UILabel 53, 350
- Classes group 55, 71, 89
- classified listings 17
- clearColor method of class UIColor 226
- CLHeading 224, 232
- trueHeading property 237
- ClickPress 46
- client of a class 16
- CLLocation 224, 226, 229, 230
- getDistanceFrom: method 236
CLLocation class (cont.)
  latitude property 230
  longitude property 230
CLLocationManager class 224, 232, 236, 237
  startUpdatingHeading method 235
  startUpdatingLocation method 235
  stopUpdatingHeading method 234
  stopUpdatingLocation method 234, 237
CLLocationManagerDelegate protocol 231, 236, 237
  locationManager:didFailWithError: 237
  locationManager:didUpdateHeading: 237
  locationManager:didUpdateToLocation:fromLocation: 236
Cocoa xxix, xxx, xxxi, 2, 10, 11, 16
  frameworks 10, 11
Cocoa frameworks 55, 56
  Address Book 12
  Address Book UI 11
  Audio Toolbox 12
  Audio Unit 12
  AV Foundation 12
  CFNetwork 13
  Cocoa Touch Layer 11
  Core Audio 12
  Core Data 12
  Core Foundation 12
  Core Graphics 12
  Core Location 12
  Core OS Layer 13
  Core Services Layer 12
  External Accessory 13
  Foundation 12
  Map Kit 11
  Media Layer 12
  Media Player 12
  Message UI 11
Cocoa frameworks (cont.)
  Mobile Core Services 12
  OpenGL ES 12
  Quartz Core 12
  Security 13
  Store Kit 13
  System 13
  System Configuration 13
  UIKit 11
Cocoa Touch 10
Cocoa Touch Class 71
  Cocoa Touch Layer 11
  code examples xxxv
  code highlighting 2
  code license xxix
  code walkthrough 2
  code.google.com 49
  code.google.com/chromium/ 49
Comcast Mobile app 37
Command design pattern 10, 100
commitAnimations method of class UIButton 138, 190, 208
Compass 5
  compass 6
  compass heading 224
  component 15
Components
  Flexible Space Bar Button Item 225
Composite design pattern 10
connection:didFailWithError: of protocol NSURLConnectionDelegate 376
connection:didReceiveData: of protocol NSURLConnectionDelegate 376
connection:didReceiveResponse: of protocol NSURLConnectionDelegate 376
connectionDidFinishLoading: of protocol NSURLConnectionDelegate 376
connectionTypesMask property of class GKPeerPickerController 347
const qualifier 110
constant 89, 158
consumables 42
Contacts 4, 5
contentView property of class UIView 254
continue audio when the screen locks 316
contract information 34
Contracts, Tax & Banking Information 43
controller (in MVC) 71
Controls
  Button 18
  Label 18
  Slider 18
  View 18
convertCoordinate:toPointToView: method of class MKMapView 229
copy and paste 8
copy text 8
copyItemAtPath:newPath:error: of class NSFileManager 298
copyright xxix
Core Animation 260, 282, 309
Core Animation block 138, 190, 306
Core Animation framework 132, 156
Core Animation Layer 132
Core Audio 12
Core Audio File 319
Core Data 12, 351
Core Data data model 345
Core Data framework xxxi, 345
Core Data object 347
Core Foundation 12
Core Graphics 12
Core Graphics framework 157
Core Location 12
Core Location framework xxxi, 224, 232
Core OS Layer 13
Core Services Layer 12
count property 335
count property of class NSMutableArray 335
CPU usage xxxi
Craigslist (www.craigslist.org) 17
create derivative apps xxix
Index

cStringUsingEncoding method of class NSString 169
current system time 319
currentTime property of class AVAudioPlayer 332, 333
cut and paste 8
cut text 8

daily mobile.se/2009/02/11/iphone-humor-cell-phone-reunion/ 49
data model editor 346
dataSource of class UITableView 201
dataWithContentsOfFile: method of class NSData 339
Decktrade 48
decodeIntForKey: method of class NSCoder 285
decodeObjectForKey method of class NSCoder 285, 288
decoding 282
Decorator design pattern 10, 121
Default Apps 4
  App Store 5
  Calculator 5
  Calendar 5
  Camera 5
  Compass 5
  Contacts 5
  iPod 5
  iTunes 5
  Mail 5
  Maps 5
  Messages (SMS/MMS) 5
  Notes 5
  Phone 5
  Photos 5
  Safari 5
  Settings 5
  Stocks 5
  Voice Memos 5
  Weather 5
  YouTube 5
default install location for the SDK xxxvi
defaultCenter of class NSNotificationCenter 308
defaultManager method of class NSFileManager 198, 332
Deitel® Buzz Online Newsletter (www.deitel.com/newsletter/subscribe.html) xxxiii, xxxvi, 17, 386
Deitel® Training (deitel.com/training) 386
delay before calling a function 132
delegate 121
delegate protocol 121
deleteObject: method of class NSManagedObjectContext 360
deleteRowsAtIndexPaths:withRowAnimation: method of class UITableView 203, 338
Delicious (www.delicious.com) 17, 44
denyConnectionFromPeer: method of class GKSession 354
dequeueReusableCellWithIdentifier: method of class UITableView 249, 293
dequeueReusableCellWithIdentifier: method of class UITableView 197, 202, 210, 214, 358
deserialized 282
Design patterns xxxi, 10, 71
  Abstract Factory 10, 91
  Chain of Responsibility 10
  Command 10, 100
  Composite 10
  Decorator 10, 121
  Facade 10
  Memento 11
  Model View Controller 10
  Singleton 11, 94
  Template Method 11, 79
Detail Disclosure Button 86
detect performance problems 169
developer.apple.com documentation/Cocoa/Conceptual/
  CocoaFundamentals/
  CocoaFundamentals.pdf 3
developer.apple.com documentation/Cocoa/Conceptual/
  CodingGuidelines/
  CodingGuidelines.pdf 3
developer.apple.com documentation/Cocoa/Conceptual/ObjC.pdf 3
developer.apple.com documentation/DeveloperTools/Conceptual/
  Xcode_Overview/
  Contents/Resources/en.lproj/
  Xcode_Overview.pdf 3
developer.apple.com documentation/DeveloperTools/Conceptual/
  XcodeDebugging/
  Xcode_Debugging.pdf 3
developer.apple.com documentation/DeveloperTools/Conceptual/ObjectiveC/
  ObjC.pdf 3
developer.apple.com documentation/DeveloperTools/Conceptual/
  UserExperience/
  AppleHIGuidelines/
  XHIGIntro/XHIGIntro.html 3
developer.apple.com/iphone/ downloads xxxv
developer.apple.com/iphone/library/
  documentation/Cocoa/Conceptual/Strings/introStrings.html 80
financial transaction 41
Find My iPhone 6
Finder xxxvi
Finder window 18
Flag Quiz Game app xxx, xxxi, 10
Flexible Space Bar Button Item component 225, 267
flick 4, 15
Flickr 44
flipside view 106, 191
used for settings 106
FlipsideViewController class 188
FlipsideViewControllerDelegate protocol 109, 123
float 227, 327
floatValue method of class NSNumber 328
for...in operator 95
format specifier 80, 100
formatting string objects 80
Foundation 12
frame property 96
frame property of class UIView 257
Frameworks
AV Foundation 314
AVFoundation 132
Core Animation 132, 156
Core Data 345
Core Graphics 157
Core Location 224
Game Kit 345
Map Kit 224, 226
Store Kit 41
UIKit 56
free app 8, 36, 41
Free Applications contract 34
Fresh Apps app review site 46
friend 45
friend in Facebook 45
frontside view 106
function 16

g
Game Kit 32
Game Kit framework xxxi, 14, 345
games 14, 35
generic pointer 80
gesture 4
Gestures
double tap 4
drag 4
flick 4
pinch 4
swipe 4
tap 4
touch and hold 4
getAddressFrom: method of class CLLocation 236
getAddressValue: method of class NSNumber 180
Gizmodo app review site 46
gizmodo.com/5300060/find-my-iphone-saved-my-phone-from-a-thief 49
gizmodo.com/tag/iphone-apps-directory/ 46
GKPeerPickerControllerDelegate protocol 346, 351, 353
GKPeerPickerController class 345, 346, 347
connectionTypesMask property 347
show method 347
GKPeerPickerControllerDelegate protocol 346, 348, 349
peerPickerController:didConnectToPeer:toSession: method 348
peerPickerControllerDidCancel: method 349
GKSendReliable constant 348
GKSession class 345, 348, 352, 354
acceptConnectionFromPeer: error: method 354
available property 352
denyConnectionFromPeer: method 354
displayNameForPeer: method 353
initialize 352
sendDataToAllPeers: withDataMode: method 345, 348
GKSession class (cont.)
setDelegate:withContext: method 352
GKSessionDelegate protocol 346, 351, 353
session:didReceiveConnectionRequestFromPeer: method 353
GKSessionModeServer 352
global variables 109
Google 49
Google Maps xxxi, 13
Google Maps web services 226
Google Mobile Maps Service 14
GPS 224
Graphical User Interface (GUI) 9
graphics xxxi
graphics context 168, 178, 227
greater than 81
greater than or equal to 81
Groups and Files window 55, 71, 108, 109, 122
gesture 15
GUI (Grahical User Interface) 9
GUI Components
Bar Button Item 86
Button 86
Detail Disclosure Button 86
Image View 86
Info Button 106, 108, 125
Label 53, 67, 86, 107, 133
Rounded Rect Button 86, 87
Scroll View 86, 88
Segmented Control 106, 110, 127
Slider 65, 68, 86, 190
Switch 123
Tab Bar 86
Tab Bar Item 86
Text Field 66, 87, 285
Toolbar 86
View 86
GUI design 35

H
hashtag 45
header file 71, 108
heading, compass 224
Index

headset jack 4
hearing impaired 7
Home button 4
Humor 49
i-Newswire 47
IBAction 73
IBOutlet 72
icon 32, 33
icon design firms
icon design 33
IconDrawer 33
Razorianfly Graphic Design 33
The Iconfactory 33
id <ProtocolName> 124
id generic pointer type 80
id type 73
implicit 97
IDE (integrated development environment) xxxi, 14
if...else keyword 80
image picker 245, 296
image property of class
UIImage 132, 141
image transition 245
Image View 53, 59, 61, 86
Image View GUI component 107
imagePickerController:didFinishPickingImage:editingInfo method of protocol
UIImagePickerControllerDelegate 271
images xxxi
implementation file 78
@implementation keyword 78
in-app advertising 36, 37
In App Purchase 13, 32, 41, 42
in-game voice communication 14
indexPathForCell: method of class
UITableView 208
inequality 80
info button 18, 31
Info Button GUI Component 106, 108, 125
information hiding 16
inheritance 16, 65, 72, 111
inherited 72
init method 92
init of class NSMutabileDictionary 94
initialize an NSFetchedResultsController 361
initWithCapacity: method of
NSMutableArray 361
initWithCapacity: of class
NSArray 326
initWithCoder: method of
class NSObject 159
initWithCoder: method of
protocol NSCoder 282
initWithCoder: method of
protocol NSCoder 288
initWithContentsOfFile 94
initWithContentsOfFile: method of class
NSMutableArray 198
initWithCustomView: of class
UIBarButton 369
initWithNibName:bundle: of
class UIViewController 207
initWithSubviews: method of
class NSArray 207
initWithStyle:reuseIdentifier: method of class
UITableViewCell 94
initWithStyle:reuseIdentifier: method of class
UITableViewCell 202, 384
initWithTitle:delegate:cancelButtonTitle:destruc
tiveButtonTitle: otherButtonTitles: method of class
UIActionSheet 299
insertRowsAtIndexPaths method of class
UITableView 355
Inspector window 61, 66, 73, 87, 91, 122
instance 16
instance method 73
instance variable 16, 71, 73, 133, 198
instantiated 16
Instruments tool xxxi, 53, 169
Activity Monitor template 169
checking for memory leaks 118
integrated development environment (IDE) xxxi, 14
Intel-based Mac xxxv
interface 16, 78
Interface Builder 9, 10, 14, 53
interface of a class 72
international App Stores 32
Internet Public Relations
ClickPress 46
i-Newswire 47
InternetNewsBureau.com 47
Markerwire 46
openPR 47
PR Leap 46
Press Release Writing 47
PRLog 47
PRWeb 46
PRX Builder 47
Internet telephony 17
Internet tethering 7
InternetNewsBureau.com 47
invalidate method of class
NSTimer 166
iPhone 3G xxix, 3
iPhone 3GS xxix, 3
iPhone App Reviews 45
iPhone Application Programming Guide 34, 50
iPhone Developer Center 43
iPhone Developer Program 2, 24, 25
iPhone Developer Program Portal 25, 26, 27, 28, 29
iPhone Developer University Program 3
iPhone Development Certificate 26
iPhone Development Guide 32, 50
iPhone Development Team 25
iPhone Distribution Certificate 29

iPhone for Programmers website
www.deitel.com/books/iPhoneFP/ xxix

iPhone Human Interface Guidelines 24, 30, 34, 50

iPhone OS 3 Readiness Checklist 32

iPhone OS 3.0 8

iPhone OS 3.x 41

iPhone OS 3.x compatible 32

iPhone Reference Library 50

iPhone sales 3

iPhone SDK xxxv, xxxvi, 14

iPhone simulator 14, 52

rotate left 245

rotate right 245

iPhone Toolbox app review site 46

iphone.iusethis.com/ 46

iPhoneSDK.mpkg xxxvi

iphonetoolbox.com/category/application/ 46

iPod 3, 9

iPod 4, 5

iPod library access xxxi, 14, 32

iPod music library 267, 281, 298

iPod Touch 2, 8

iterate through the items in a collection 95

iTunes 4, 7, 9, 39, 42

iTunes 5

iTunes Connect 24, 41, 42

iTunes Connect Developer Guide 32, 33, 34, 42

iTunes Connect Modules 43

iTunes Store 9

itunesconnect.apple.com 38, 42

iusethis app review site 46

Java xxx

Jobs, Steve 9

K

kCLLocationAccuracyBest constant 233

keyboard 4

how to display 79

how to set the type 210

layout 9

Keychain Access 26, 27, 29

Keywords 32, 33

for...in 95

if 73

if...else 80

nil 93

self 92

struct 158

super 92

kuTypeImage class 296

L

Label GUI Component 18, 53, 61, 67, 86, 107, 133, 191

landscape keyboard 6, 8

language support 9

lastObject method of class NSMutableView 230

lastPathComponent method of class NSString 112, 297

latitude 224

latitude property of class CLLocation 230

launch image 32, 34, 34

layer property of class UIView 141, 145

leftBarButtonItem property of class UINavigationItem 352

less than 81

less than or equal to 81

Library window 59, 67, 87, 91, 122

LinkedIn 44, 50

litera

พิเศ80

loadView method of class UIView 257

local variable

declared static 79

localization 40

locate your iPhone 6

location (GPS) 224

location-based app 14

locationInView: method of class UIView 141

locationManager:didFailWithError: method of protocol CLLocationManagerDelegate 237

locationManager:didUpdateHeading: method of protocol CLLocationManagerDelegate 237

locationManager:didUpdateToLocation: method of protocol CLLocationManagerDelegate 236

lock the iPhone 4

longitude 224

longitude property of class CLLocation 230

M

Mac xxx

Mac OS X xxx, xxxv, 8, 9

Macintosh 9

Macworld app review site 46

Mail 4, 5

mailComposeController:didFinishWithResult:error: method of protocol MFMailComposeViewControllerDelegate 340

mailComposeDelegate property of class MFMailComposeViewController 340

mainBundle method of class NSBundle 112

Manage Users 43

Manage Your Applications 43

Managed Object Context 345

managed object model 345

managedObjectContext method of class NSFetchedResultsController 355

map 14

Map Kit 11

Map Kit framework xxxi, 14, 224, 226

Maps 5

mapType property of class MKMapView 235
mapView:regionDidChangeAnimated: of protocol MKMapViewDelegate 231
mapView:regionWillChangeAnimated: of protocol MKMapViewDelegate 231
marketing xxxi
Marketwire 46
mashup 13
Media Layer 12
Media Player 12
Media Player framework xxxi, 14
Medialets 47
mediaPicker:didPickMediaItems: of protocol MPMediaPickerControllerDelegate 272
mediaTypes property of class UIImagePickerController 282, 295
Memento design pattern 11
memory leak xxxi
memory limitation 30
memory management 106, 107
developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/MemoryMgmt/MemoryMgmt.html 107
menu name xxxv
Menus
Build 54
Subclass of 71
message 75, 79
Message UI 11
Messages (SMS/MMS) 5
method implementations that enhance an existing class 91
method of a class 16, 71
MFMailComposeViewController class 314, 340
addAttachmentData: migraine: method 340
mailComposeDelegate property 340
MFMailComposeViewController protocol 340
mailComposeController: didFinishWithResult: error: method 340
micro blogging 44, 45
microphone 4, 8
Microsoft Exchange ActiveSync 9
MKAnnotationView class 236
MKCoordinateRegion struct 237
MKCoordinateSpan struct 237
MKCoordinateSpanMake function of MapKit 236
MKMapTypeSatellite map type constant 235
MKMapTypeStandard map type constant 235
MKMapType class 224, 226
transform property 237
convertCoordinate:toPointToView: method 229
mapType property 235
scrollEnabled property 234
zoomEnabled property 234
MKMapViewDelegate protocol 226
mapView:regionDidChangeAnimated: 231
mapView:regionWillChangeAnimated: 231
MKTypeHybrid map type constant 235
MMS (Multimedia Messaging Service) 9
mobile advertising network 37, 47
AdMob 37, 47
AdWhirl 37, 47
Decktrade 48
Medialets 47
Pinch Media 38
PinchMedia 48
Quattro Wireless 47
Tapjoy 37, 48
Mobile Core Services 12
MobileMe 6, 49
modalTransitionStyle property of class UIViewController 121, 186
model (in MVC) 71
Model-View-Controller (MVC) design pattern xxxi, 10, 71, 286, 345
modulus operator 116
monetization 47
monetize apps 37
monetizing apps 24
motionEnded:withEvent: of class UIResponder 183
mount xxxvi
mounted image xxxvi
moveItemAtPath:toPath: method of class NSFileManager 321
movies 9
MPMediaItemCollection class 245, 257, 267, 287
MPMediaPickerController class 245, 267, 272
allowsPickingMultipleItems property 272
MPMediaPickerControllerDelegate protocol
mediaPicker:didPickMediaItems: 272
MPMoviePlayerController class 282
MPMusicPlayer class 262
play method 262
setQueueWithItemCollection: method 262
MPMusicPlayerController class 245, 257, 262
MPMusicShuffleMode=None 262
MPMusicShuffleModeOff 262
msdn.microsoft.com/en-us/windowsmobile/default.aspx 49
MSMArray class
removeAllObjects method 327
multi-touch events 132
Multi-Touch screen 4, 11
multimedia xxxi
Multimedia Messaging Service (MMS) 9
multiplayer game 14
multiplication 81
music 14
music library 9
mutableCopy method of class
NSMutableArray 335
mutableCopy method of class
NSObject 112
mutually exclusive options 106
MVC (Model-view-controller) xxxi
MySpace 17, 44, 50

N
na.blackberry.com/eng/services/appworld/? 49
Nationwide Mobile app 37
navigate between an app’s screens 245
navigation bar 198, 347, 349, 352
Navigation-based Application
template 197, 204, 366
navigationController property of class
UIViewController 197
navigationItem property of class
UIViewController 198, 246
network activity xxxi
networkActivityIndicatorVisible of class
UIApplication 374
New App ID button 27
New Project dialog 53
NeXT 9, 56
NeXT Interface Builder 14
NeXTSTEP operating system 9
NeXTSTEP programming environment 56
nib file 14, 56, 92
Nike + iPod Sensor 14
nil keyword 93
nonatomic keyword 109
non-consumables 42
Notes 5, 9
nouns in a system specification 17

NSArray class 91, 207
initWithObjects: method 207
NSAutoreleasePool class 122
NSBundle class 112
mainBundle method 112
pathForResource:ofType: method 112, 134
NSCoder class 282, 284, 288
decodeInt:forKey: method 285
decodeObject:forKey method 285, 288
encodeInt:forKey: method 285, 289
encodeObject:forKey: method 285, 289
encodeWithCoder: method 282
initWithCoder: method 282
NSCoding protocol 282, 288
initWithCoder: method 288
NSData class 339, 355
dataWithContentsOfFile: method 339
NSData class 348
NSDate class 224, 232
timeIntervalSinceNow method 235
NSDictionary class 90, 98, 198, 200, 296, 348, 350, 355
allKeys method 350
dictionaryWithDictionary: method 208
valueForKey: method 296
writeToFile:atomically: method 98
NSEntityDescription class
345, 348, 350, 355
properties method 350
propertiesByName method 348, 350
NSError class 352
NSURLConnection class 351, 352, 355, 358
initialize 361
managedObjectContext method 355
NSFetchRequest class 345
setSortDescriptors: method 362
NSFileManager class 94, 198, 298, 332
copyItemAtPath:newPath: method 298
defaultManager method 198, 332
directoryContentsAtPath: method 332
moveItemAtPath:toPath: method 321
NSIndexPath class 202, 214, 249, 251, 293, 300, 350, 358
row property 293, 350
NSKeyedArchiver class 282, 303, 348
archivedDataWithRootObject: method 348
archiveRootObjectToFile: method 303
NSKeyedUnarchiver class 292
unarchiveObjectWithFile: method 292
NSLocale class 80
NSLog function of the Foundation framework 352, 355
NSManagedObject class 345, 347, 348, 358
tenInObject: method 348
save method 348, 357
setValue forKey: method 355
NSManagedObjectContext class 345, 352
deleteObject: method 360
NSMutableArray 198
NSMutableArray class 86, 91, 133, 178, 198, 226, 246, 252, 288, 292, 326, 327, 335
addObject: method 248, 292
initWithCapacity: method 111, 326
initWithContentsOfFile: method 198
lastObject method 230
mutableCopy method 335
objectAtIndex: method 328
removeAllObjects method 230
removeObjectAtIndex: method 252, 327
sortUsingSelector: method 99

NSMutableData class 373
NSMutableDictionary class 86, 90, 178, 200, 348, 355
init 94
initWithCapacity: 207
removeAllObjects method 178
setValue:forKey: 97
writeToFile:atomically: 200

NSNotificationCenter class 308
defaultCenter 308
removeObserver: 309

NSNumber class 111, 137, 326, 328
floatValue method 328
numberWithBool: method 111

NSNumberFormatter class 80

NSObject class 112
autorelease 122
initWithCoder: method 159
mutableCopy method 112
release method 107, 112
retain method 107, 112
NSObject class performSelector: withObject: AfterDelay: method 116, 137

NSPredicate class 314, 325
evaluateWithObject: method 325
predicateWithFormat: method 325

NSearchPathForDirectoriesInDomains function 319, 331
NSearchPathForDirectoriesInDomains function of the Foundation framework 198, 291

NSSet class 140
allObjects 182

NSSortDescriptor class 362

NSString class 73, 297, 332
@"string" literal 80
cStringUsingEncoding method 169
initWithString: method 94
lastPathComponent method 297
pathExtension method 332
rangeOfString: method 381
sizeWithFont: 218
stringByAddingPercentEscapesUsingEncoding: method 374
stringByAppendingPathComponent: method 94, 298, 332
stringByDeletingLastPathComponent method 320
stringWithFormat: method 333
substringWithRange: method 381

NSString literal 80

NSString class (cont.)
filePathWithPathComponent: method 297

NSURL class (cont.)
fileURLWithPath: method 298
path method 320
URLWithString: method 338

NSURLConnection class 366, 374
NSURLConnectionDelegate protocol
connection:didFailWithError: 376
connection:didReceiveData: 376
connection:didReceiveResponse: 376
connectionDidFinishLoading: 376

NSURLRequest class 372
NSURL class 100, 297, 298, 372
absoluteString method 297

numeric keyboard, display 79
Object 91

Object 15, 16

object (or instance) 16

object graph 282, 292, 303

object messaging 100

object-oriented design (OOD) 16

object-oriented language 16

object-oriented programming (OOP) 9, 16

object serialization 282, 309

objectIndex: method of class NSMutablesArray 94, 328

objectAtIndexPath: method of class NSFetchedResultsController 358

Objective-C xxix, xxx, 2, 9

Objective-C code xxxv

on-screen component xxxv

OOD (object-oriented design) 16

OOP (object-oriented programming) 9, 16

Open GL ES 2.0 12, 32

openPR 47

OpenStep 11

openURL method of class UIApplication 101

operating system 8

operating system requirements xxxv

Operators

- 81

!= 80

?: 182

* 81

/ 81

% 81

+ 81

< 81

<= 81

== 80

> 81

>= 81

Orkut 44

OS X 9

outlet 72, 109, 124, 186, 232, 264, 315, 328

Paid Applications contract 34

Painter app xxx, 17

Parental Controls 9, 32, 39

parser:didEndElement:namespaceURI:qualifiedName: of protocol NSXMLParserDelegate 379

parser:didStartElement:namespaceURI:qualifiedName:attributes: of protocol NSXMLParserDelegate 379

parser:foundCharacters: of protocol NSXMLParserDelegate 379

paste text 8

path method of class NSURL 320

pathExtension method of class NSString 332

pathsForResourceOfType: method of class NSBundle 112, 134

pause method of class AVAudioPlayer 335

payment 42

peer ID 353

peerPickerController:didConnectPeer:toSession: method of protocol GKPeerPickerControllerDelegate 348

peerPickerControllerDidCancel: method of protocol GKPeerPickerControllerDelegate 349

peer-to-peer connectivity 14

peer-to-peer games 9


performance problems, detect 169

performFetch: method of class NSFetchedResultsController 352

performSelector:withObject:AfterDelay: method of NSObject 116, 137

Phone 4, 5

Photo API 245

photo sharing 17, 44

Photos 5

photos 4

pinch 4, 15

Pinch Media 38, 48

play method of class MPMusicPlayer 262

.plist extension 98

.plist file 198

.plist format 98

podcast 14

pointer
generic 80

to the sender component 80

popViewControllerAnimated: method of class UINavigationController 259

to power the iPhone 4

PR Leap 46

predicateWithValue: method of class NSPredicate 325

prepareToRecord method of class AVAudioRecorder 320

preprocessor 72

presentationLayer method of class CALayer 141

presentModalViewController:animated: method of class UIViewController 200, 248, 317

Press Release Writing 47

price 8, 36

price tier 40

Pricing Matrix 40

primary screenshot 34

privacy 6

PRLog 47

Programatically update user interface 96
programmatically select a component 79
programming languages
Objective-C 10
project 53

Project Structure group 55
properties method of class
NSEntityDescription 350
propertiesByName method of
class NSEntityDescription
348, 350
property 109, 257, 264, 322
access with dot (.) notation
readonly 175
property-list format 98
property of an object 16
protocol 109
delegate 121
similar to an interface in oth-
er programming languages
109

Protocols
CLLocationManagerDelegate
gate 236, 237
EditableCellDelegate 205
GKPeerPickerControllerDelegate 348,
349
GSessionDelegate 353
MFMailComposeViewControllerDelegate 340
NSFetchedResultsController
Info 358
UITableViewDataSource
201, 205, 212, 328
UITableViewDelegate 201,
328
UITextFieldDelegate 215,
322, 325

Provisioning 28
Provisioning Profile 25, 27
PRWeb 46
PRX Builder 47
public relations 46
purchase 41
purchasing interface 42
Push Notification 2, 13

pushViewController:animated:
method of class UINavigationController
202, 248, 251, 293, 359
removeObjectAtIndex: method
of class NSMutableArray
252, 327
removeObserver: of class
NotificationCenter 309
Request Promotional Codes 43
resignFirstResponder method
of class UIAlertView
185
resignFirstResponder method
of class UIResponder 218

Resources
(www.deitel.com/Resour-
cesCenters.html) 17

Resources group 55, 58
respond chain 141
REST xxxi
retain count 106, 112
retain counting 106
retain keyword 109
resources method of class
NSObject 107, 112

reuse 17
reuse UITableViewCell 197
RGB values 87, 191, 229
Rhapsody 11
Ring/Silent switch 4
rotate left (iPhone simulator)
245
rotate right (iPhone simulator)
245

Rounded Rect Button 86, 87
Route Tracker app xxx, 2, 6, 11,
12, 14, 15
row property of class
NSIndexPath 293, 350
run loop 122

Safari 4, 5, 9
Sales/Trend Reports 43
Salesforce 17
save data on the iPhone 84
save method of class
NSManagedObject 355, 357
scheduledTimerWith-
TimeInterval:target:select:
userInfo:repeat:
method of class NSTimer
161
screen size 30
screenshot 32
scroll 4
Scroll View 86, 88
scrollEnabled property of class MKMapView 234
scrollToRowAtIndexPath:atScrollPosition:animated: of class UITableView 208
SDK (Software Development Kit) xxxv
SDK beta xxxv
SDK documentation xxxv
search 9
Second Life 17
sections method of class NSFetchedResultsController 357
Security 13
seed (random number generation) 111
Segmented Control GUI Component 106, 110, 122, 127
dynamically created 115
select a component programmatically 79
selectedSegmentIndex property of class UISegmentedControl 235
selectionStyle property of class UITableViewCell 249, 293
self keyword 92
sendDataToAllPeers:withDataMode: method of class GKSession 345, 348
sender of an event 80
serialized object 282
session
didReceiveConnectionRequestFromPeer: method of protocol GKSessionDelegate 353
set the keyboard type 210
setAnimationCurve: method of class UIView 139, 208
setAnimationDidStopSelector: method of class UIView 139, 144
setAnimationDuration: method of class UIView 139, 190, 208
setBackBarButtonItem: method of class UINavigationItem 247
setCategory: method of class AVAudioSession 317, 319, 334
setContentSize: method of class UIScrollView 99
setDataReceiveHandler:withContext: method of class GKSession 352
setIdleTimerDisabled: of class GKSession 352
setNeedsDisplay method of class UIView 165, 178, 230
setNeedsDisplayInRect: method of class UIView 182
setQueueWithItemCollection: method of class MPMusicPlayer 262
setRegion: method of class UIMapView 237
setRightBarButtonItem: method of class UINavigationItem 347
setSortDescriptors: method of class NSFetchRequest 362
setStatusBarHidden: method of class UIApplication 259
setStatusBarHidden: method of class UIApplication 262
Settings 5
setTitle: method of class UINavigationItem 246, 349
setValue: method of class UISlider 190
setValue: animated: method of class UISlider 333
setValue: forKey: method of class NSManagedObject 355
setValue: forKey: method of class NSUserDefaults 146
setValue: forKey: of class NSUserDefaults 97
Shake to Shuffle 9
sharedApplication method of class UIApplication 101, 259
sharedInstance method of class AVAudioSession 316
sheet 57
shine effect 33
shouldAutorotateToInterfaceOrientation: method of class UIViewController 204, 262
show method of class GKPeerPickerController 347
SIM card tray 4
simulator 32
singleton 262, 302, 314, 316
Singleton design pattern 11, 94, 101, 314
sizeWithFont: of class NSAttributedString 218
Skype 17
Sleep/Awake button 4
Slider 18, 65, 68, 86, 191
Slider GUI component 190
Slideshow app xxx, 6, 12, 14
social bookmarking 17, 44
social media 44
social media sites
  Blogger 44
  Delicious 44
  Digg 44
  Flickr 44
  LinkedIn 44
  Squidoo 44
  StumbleUpon 44
  Tip’d 44
  Wordpress 44
  YouTube 44
social networking 17, 44
social news 44
Software Development Kit (SDK) xxxv
sort an NSMutableArray 91
sortUsingSelector: method of class NSMutableArray 99
sound 132
source code 2
source-code listing 2
sourceType property of class
UIImagePickerController 271, 295
speaker 4
speech recognition xxxi
speech synthesis xxxi
Spotlight 9
Spot-On Game app xxx, 10, 12, 20
Squidoo 44
srandom library method 111
stackoverflow.com/questions/740127/how-was-your-iphone-developer-experience 48
standardUserDefaults method 137
startUpdatingHeading method of class CLLocationManager 235
startUpdatingLocation method of class CLLocationManager 235
static global variable 110
static keyword
local variable 79
Static method 73
statically typed object 73
status bar 259
StepStone 9
Stocks 5
stopUpdatingHeading method of class CLLocationManager 234
stopUpdatingLocation method of class CLLocationManager 234, 237
Store Kit 2, 13, 32
Store Kit framework 13, 41, 42
Store Kit Framework Reference 42
Store Kit Programming Guide 42
string format specifier 169
string formatting 80
string literal that begins with @ 80
stringByAppendingPathComponent: method of class NSString 94, 298, 332
stringByDeletingLastPathComponent method of class NSString 320
stringWithFormat: method of class NSString 333
struct keywords 158
structure 96, 157, 158
structure members 158
structure tag 158
structure type 158
StumbleUpon 44
subscription 42
substringWithRange: method of class NSString 381
subtraction 81
subview 95
subviews property 95
super keyword 92
superview 95
swipe 4, 15
Switch GUI Component 123
Symbian 49
sync 7, 9
syntax shading 2
synthesize a property 111
@synthesize directive 177, 367
@synthesize keyword 111
System 13
System Configuration 13

T
Tab Bar 86
Tab Bar Item 86
tableView:canMoveRowAtIndexPath: method of class UITableView 252
tableView:cellForRowAtIndexPath: method of class UITableView 249, 293, 300
tableView:cellForRowAtIndexPath: method of protocol UITableViewDataSource 202, 214, 275, 350, 371
tableView:commitEditingStyle:forRowAtIndexPath: method of class
UITableViewController 251
tableView:commitEditingStyle:forRowAtIndexPath: method of protocol
UITableViewDataSource 203, 275, 337
tableView:didSelectRowAtIndexPath: method of protocol
UITableViewDelegate 202, 338, 371
tableView:moveAtIndexPath:toIndexPath: method of class
UITableViewController 252
tableView:moveRowAtIndexPath:toIndexPath: of protocol
UITableViewDataSource 275, 302
tableView:numberOfRowsInSection: method of class
UITableView 213
tableView:numberOfRowsInSection: method of class
UITableViewController 248
tableView:numberOfRowsInSection: method of protocol
UITableViewDataSource 201, 274, 350, 370
tableView:numberOfRowsInSection: of class
UITableViewDataSource 210
tableView:titleForHeaderInSection: method of protocol
UITableViewDataSource 210
tap 4, 15
tapCount property of class
UITouch 171
Tapjoy 37, 48
Team Admin 25, 26, 34
Team Agent 25, 29
Team Member 25, 26
Template Method design pattern 11, 79
testing xxxv

Text Field 66, 87, 285
textField:shouldChangeCharactersInRange:replacementString: method of protocol UITextFieldDelegate 325
textFieldDidBeginEditing: of protocol UITextFieldDelegate 215
textFieldDidEndEditing: of protocol UITextFieldDelegate 215
textLabel method of class UITableViewCell 350
The App Podcast app review site 46
theapppodcast.com/ 46
time library function 111
timeIntervalSinceNow method of class NSDate 235
Tip Calculator app xxx, 10, 11, 15
Tip’d 44
tipd.com/ 44
title property of class UIButton 234
Toolbar 86
touch and hold 4, 15
touch handling 132
Touch Up Inside event 92
touchesBegan method of class UIView 132
touchesBegan:withEvent: method of class UIResponder 139
touchesBegan:withEvent: of class UIResponder 182
touchesEnded:withEvent: of class UIResponder 183
transform property of class MKMapView 237
trueHeading property of class CLHeading 237
TV shows 9
tweet 45

Twitter xxxi, xxxiii, 17, 45, 49, 100
@deitel 386
hashtag 45
tweet 45
Twitter app xxx
Twitter Discount Airfares app 13
Twitter search 84
operators 84
typedef keyword 282
typedef specifier 158

UI
UDID (Unique Device Identifier) 27
UIActionSheet class 245, 267, 299
initWithTitle:delegate:cancelButtonTitle:destructiveButtonTitle:otherButtonTitles: method 299
UIActionSheetDelegate protocol
actionSheet:clickedButtonAtIndex: 272
UIActivityIndicatorView class 368
UIAlertView class 118, 353
UIAlertViewDelegate protocol
alertView:clickedButtonAtIndex: 183
UIApplication class 100, 259, 262, 302
networkActivityIndicatorVisible 374
openURL method 101
setIdleTimerDisabled: 234, 235
setStatusBarHidden: method 259
setStatusBarHidden: method 262
sharedApplication method 259
UIApplicationDelegate protocol 302
UITableViewCell class 350
UIBarButtonItem class 198, 245, 247, 267, 334, 347, 369
initWithCustomView: 369
UIBarButtonSystemItemAction 347
UIButton class 97, 253, 315
title property 234
UIColor class 174, 188, 226
clearColor method 226
UIToolbar class
addTarget:action:forControlEvents: 216
UIToolbarTouchUpInside 98
UIImagePNGRepresentation function 291
UIImagePickerControllerEditedImage 296
UIImagePickerControllerMediaURL 297
UIImagePickerControllerSourceTypePhotoLibrary 295
UIImagePNGRepresentation function 291
UIViewController class 53, 59, 132, 257, 259, 262, 305
alpha property 260
image property 132, 141
Index 405

UIImage 2 class (cont.)
removeFromSuperview method 261, 262
UIKit framework 11
UIKit header file 72
UILabel class 35, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
returnBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UILabel class 53, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
returnBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UILabel class 53, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
returnBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UILabel class 53, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
returnBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UILabel class 53, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
backBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UILabel class 53, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
backBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UILabel class 53, 350
UIMapView class
setRegion: method 237
UIModalTransitionStyle-CrossDissolve 321
UIModalTransitionStyle-FlipHorizontalUIKit 121
UINavigationController class (cont.)
pushViewController: animated: method 202, 248, 249, 293, 302
setNavigationBarHidden: animated: method 247
viewDidLoad property 302
UINavigationItem class 247, 349, 352
backBarButtonItem property 198
leftBarButtonItem property 352
setBackBarButtonItem: method 247
setRightBarButtonItem: method 347
setTitle: method 246, 349
UIResponder class
canBecomeFirstResponder 183
motionEnded:withEvent: 183
resignFirstResponder 218
touchesBegan:withEvent: 182
touchesEnded:withEvent: 139, 183
UIScrollView class
UITableView class 196, 197, 201, 202, 247, 248, 249, 293, 296, 332, 350, 351, 355, 366
dataSource 201
deleteRowsAtIndexPaths:withRowAnimation: method 203, 338
dequeueReusableCell- WithIdentifier: method 202, 210, 214, 249, 293, 358
indexPathForCell: method 208, 293
insertRowsAtIndexPaths method 355
reloadData method 247, 355
scrollToRowAtIndexPath:atScrollPosition:animated: method 208
tableView:cellForRowAtIndexPath: method 202, 210, 214, 249, 293, 358
tableView:commitEditingStyle:forRowAtIndexPath: method 203, 293
tableView:numberOfRowsInSection: method 210
tableView:cellForRowAtIndexPath: method 202, 214, 275, 350, 371
tableView:commitEditingStyle:forRowAtIndexPath: method 203, 275, 337
tableView:moveRowAtIndexPath:toIndexPath: method 275, 302
tableView:numberOfRowsInSection: method 201, 210, 274, 350, 370
tableView:titleForHeaderInSection: method 210
UITableViewDelegate class
tableView:didSelectRowAtIndexPath: method 202, 371
UITableViewDelegate class (cont.)
tableView:canMoveRowAtIndexPath: method 252
tableView:commitEditingStyle:forRowAtIndexPath: method 251
tableView:moveRowAtIndexPath:toIndexPath: method 252
tableView:numberOfStringSections: method 248
UITableViewDataSource protocol 201, 205, 212, 238, 246
tableView:numberOfStringSectionsInTableView: 210
tableView:cellForRowAtIndexPath: method 202, 214, 275, 350, 371
tableView:commitEditingStyle:forRowAtIndexPath: method 203, 275, 337
tableView:moveRowAtIndexPath:toIndexPath: method 275, 302
tableView:numberOfStringSectionsInTableView: method 201, 210, 274, 350, 370
tableView:titleForHeaderInSection: method 210
UITableViewDelegate protocol 201, 328
tableView:didSelectRowAtIndexPath: method 202, 371
UITableViewDelegate protocol (cont.)
tableView:didSelectRowAtIndexPath: method 202, 371
UITextField class 216, 286, 322, 324
textFieldDidBeginEditing: 215
textFieldDidEndEditing: 215
UITextFieldDelegate class
textFieldDidBeginEditing: 215
textFieldDidEndEditing: 215
UITextFieldDelegate protocol 215, 322, 325
UIView class (cont.)
tag property 337
touchesBegan method 132
viewDidAppear method 190
viewDidLoad method 134
UIViewController class 111, 198, 200, 264, 292, 296, 315, 322, 346
becomeFirstResponder method 185
dismissModalViewControllerAnimated: method 120, 200, 248, 292, 296
drawRect: method 227
frame property 257
initWithCoder: method 177
layer property 141, 145
loadView method 257
locationInView: method 141
removeFromSuperview method 112, 145
setAnimationCurve: method 139, 208
setAnimationDidStopSelector: method 139, 144
setAnimationDuration: method 139, 190, 208
setNeedsDisplay method 165, 178, 230
setNeedsDisplayInRect: 182
UIWebView class 366, 371
webViewDidFinishLoad: method 373
unarchiveObjectWithFile: method of class NSKeyedUnarchiver 292
unarchiving 282
Unique Device Identifier (UDID) 27
unlock the iPhone 4
updateMeters method of class AVAudioRecorder 321
upload finished apps xxxv
URL encode a string 374
URLWithString: method of class NSURL 338
utilities 35
Utility Application template 107, 109, 120, 123, 175

V
value property of class UISlider 190
valueForKey 97
valueForKey: method of class NSDictionary 296
valueForKey: method of class NSUserDefaults 137
valueWithBytes:objcType: of class NSValue 177
valueWithPointer: of class NSValue 182
video xxxi, 4, 6
video sharing 17, 44
View 18, 86
view (in MVC) 71
viewController 106
viewController property of class UINavigationController 302
viewDidAppear method of class UIView 190
viewDidAppear: of class UIView 190
viewDidAppear: of class UIViewController 184, 269
viewDidDisappear: of class UIViewController 184
viewDidLoad method of class UIView 134
viewDidLoad method of class UIViewController 198, 286, 291, 324, 347
viewWillAppea...
Index 407

virtual world 17
visible peer 346, 351
vision impaired 7
Viximo 41
voice controls 9
Voice Memos 5
Voice Recorder app xxx
VoiceOver 7
Volume buttons 4
volume property of class AVAudioPlayer 332

W
Weather 5
Web 2.0 17
web services xxxi, 13, 226
webOS 49
webViewDidFinishLoad: of protocol UIWebViewDelegate 373
Welcome app xxx, 11, 14, 15
Welcome to Xcode window 53
What's on iPhone app review site 45
Wi-Fi 9
Wikipedia 17
Window-based Application template 54, 56, 66, 86, 225
Windows xxxv
Windows Mobile 49
word-of-mouth marketing 44
Wordpress 44
Wozniak, Steve 9
writeToFile:atomically:
method of class NSDictionary 98
writeToFile:atomically:
method of class NSMutableDictionary 200
WWDR intermediate certificate 27, 29
www.148apps.com/ 46
www.admob.com/ 37
www.adwhirl.com/ 47
www.appcraver.com/ 46
www.apple.com/iphone/apps-for-iphone/ 8
www.apple.com/iphone/iphone-3gs/accessibility.html 7
www.apple.com/iphone/softwareupdate/ 8
www.applephoneschool.com/ 46
www.appletell.com/apple/tag/iphone+app+reviews/ 46
www.appstism.com/ 46
www.appvee.com/ 46
www.bing.com/developers 49
www.bis.doc.gov/licensing/exportingbasics.htm 38
www.blogger.com 44
www.clickpress.com 46
www.craigslist.org 13
www.deitel.com/Cocoa/ (Cocoa Resource Center) xxxii
www.deitel.com/deitelfan/ (Deitel Facebook Page) xxxiii
www.deitel.com/internetpr/ 46
www.deitel.com/IPhone/ (iPhone Resource Center) xxxii, xxxv, xxxvi, 2
www.deitel.com/newsletter/subscribe.htm (Deitel Buzz Online newsletter) xxxii, xxxvi
www.deitel.com/ObjectiveC/ (Objective-C Resource Center) xxxii
www.deitel.com/ResourceCenters.html (Deitel Resource Centers) xxxii
www.deitel.com/training 386
www.delicious.com 44
www.digg.com 44
www.facebook.com 44
www.flickr.com 44
www.freshapps.com/ 46
www.google.com/mobile/#p=android 49
www.housingmaps.com 13
www.i-newswire.com/ 47
www.internetnewsbureau.com/ 47
www.iphoneappreviews.net/ 45
www.iphonebuzz.com/category/apple-iphone-humor 49
www.khronos.org/opengles 54
www.linkedin.com 44
www.linkedin.com/static?key=developers_widgets&trk=hb_ft_widgets 50
www.macworld.com/appguide/index.html 46
www.marketwire.com 46
www.myspace.com 44
www.openpr.com 47
www.orkut.com 44
www.prleap.com/ 46
www.prlog.org/pub/ 47
www.prweb.com 46
www.prxbuilder.com/x2/ 47
www.squidoo.com 44
www.stumbleupon.com 44
www.twitter.com 44
www.whatsoniphone.com/ 45
www.wired.com/gadgets/wireless/magazine/16-02/ff_iphone?currentPage=all 148
X

Xcode xxix, xxxv, 2, 10, 14, 18, 34
  Build and Debug button 18
  Build and Run button 18
Xcode Groups
  Classes 55, 71
  Project Structure 55
  Resources 55, 58
Xcode toolbar 54
Xcode toolset xxxv
Xcode Windows
  Groups and Files 71, 108, 109, 122
  Groups and Files window 55
  Inspector 61, 73, 87, 91, 122
  Library 59, 67, 87, 91, 122
  Welcome to Xcode 53
Xerox PARC (Palo Alto Research Center) 9
.xib 56

Y

Yahoo 49
Yellow Box API 11
YouTube 6, 9, 17, 44
YouTube app 5

Z

zoom 4
zoomEnabled property of class MKMapView 234