
Appendix

Completing the Lunar Lander Application

As you've gone through the book, I've slowly been contributing toward your Lunar Lander game where it has made sense. I haven't completed it, though, and this is deliberate. Where would be the fun in providing you with the finished game?! The knowledge you've gained through the book in iOS, its SDK, and Xcode will allow you to complete the game and publish it—or you may want to start a completely different game.

However, it wouldn't be fair to leave you hanging, so in this appendix I discuss some of the elements you may want to consider adding to the game. The source code associated with the book includes code fragments that complement this chapter to help you complete your game.

Implementing the Game Physics

To ensure that your lander observes the laws of physics, you need to implement a simple physics engine for it. You can find a reference describing the physics required at www.physicsclassroom.com/class/newtlaws/u2l4a.cfm. Don't worry; I take you through the basics next.

Gravity

This simulation is quite simple. First the forces of gravity are applied:

```
Y = Y + Gravity * TotaleSecondsOfThrust;
```

Here the Y value that controls the lander's height is adjusted by multiplying the Gravity figure (a double value of 9.8) by the number of seconds that thrust is applied, calculated by detecting the timestamp when the thrust key is held down, and the seconds elapsed from when it is released. You can use the `NSTimeInterval()` method to achieve this.

Thrust

You then need to apply the force of the thrust according to thrust and rotation. You need to use the `Math.h` include for this, but you have to implement your own angle-normalization macro, because the equivalent of `NormalizeAngle()` in .NET doesn't exist in Objective-C. No matter, it's easily done. First define this macro:

```
#define NormalizeAngle(x) ((return x % 360)+(x < 0 ? 360 : 0))
```

Then adjust your `x` and `y` axis according to the force of thrust:

```
Y = Y - Math.sin(NormalizeAngle(rotation + M_PI / 2)) * totalseconds * thrustspeed;
X = X - Math.cos(NormalizeAngle(rotation + M_PI / 2)) * totalseconds * thrustspeed;
```

The variable `rotation` obviously represents the angle of rotation for the lander, as affected by the user pressing the left and right keys and the `totalseconds` values (the number of seconds the thrust key is held down). The `thrustspeed` value is a constant value of 20 indicating how powerful the thrust is.

Finally, don't forget to reduce your fuel figure as the engine is thrusting.

Rotation

You won't cover rotating both left and right because they mirror each other. Let's look at what you need to do to rotate left. This method does of course assume the existence of a fuel tank, initialized with an initial value (here the value 1000) and that you pass the total number of seconds the rotation key was held down. Your method looks as follows:

```
// rotateLeft - rotate the lander left
- (void)rotateLeft:(float) totalseconds
{
    // You can't thrust if you've no fuel
    if (fuel < 0)
        return;

    // Rotate left
    rotationMomentum -= M_PI / 2.0f * totalseconds;

    // Burn fuel
    fuel -= 0.5 // fuel burn every time you thrust
}

```

The obvious difference for rotating right is that you increase the `rotationMomentum` value rather than decrease it.

Enabling User Interaction

The user is obviously required to interact with the game, although how you do this is up to you. Some options are to detect keypresses, or use onscreen images to detect a tap, or even use the accelerometer to detect the device being tipped left or right. In any case, you need to interpret the actions, and the following keys are used to interact with

the lander. Doing nothing simply lets the lander drop to the ground as gravity is applied. For your lander, you do the following.

- If the up arrow is pressed, onscreen or on the keyboard, the thruster is being fired.
- If the left arrow is pressed, onscreen or on the keyboard, the lander is rotating left.
- If the right arrow is pressed, onscreen or on the keyboard, the lander is rotating right.

Catching Game Events

You need to monitor some additional game events: for example, detecting when you hit the ground and the speed at which you hit. This is fairly straightforward using a combination of lunar lander y position and ground height and using the speed of the lander as an indicator of whether you're moving too fast.

The method only works when the ground is flat. If you decide to implement a Moon surface that has different gradients, you need to detect this in your game—maybe looking at the pixel values surrounding the lander.

Handling the Graphics

Your Lunar Lander implementation has already made provision to draw the graphics for the game by implementing the `drawRect` method, although the default implementation simply draws the same graphic (the nonthrusting lander) at the x and y location.

You need to adjust this code to be representative of the other states, including when the game is running, looking for when it should be thrusting or just dropping, and drawing at the angle appropriate for the lander. The angle is obviously in relation to the rotate-left or -right keypresses.

To rotate your image, you could use the `rotateAt()` method in Microsoft .NET, but such a method doesn't exist in Objective-C. Instead, you can use a simple macro to convert between degrees and radians, the measure used by the `UIImage` rotate equivalent called `transform`. First define a macro something like the following:

```
#define degreesToRadians(x) (M_PI * (x) / 180.0)
```

Then add the following line, assuming the `deg` variable represents the degrees you wish to rotate by:

```
myView.transform = CGAffineTransformMakeRotation(degreesToRadians(deg) );
```

If you're interested in knowing more, the post at www.platinumball.net/blog/2010/01/31/iphone-uiimage-rotation-and-scaling/ has a lot of detail about how to do image rotation and scaling.

Displaying a High Score

This should be straightforward because you covered how to create high-score code in Chapter 8, and you also looked at different mechanisms for displaying a dynamic user interface in Chapter 6. Combining these with the high-score mechanics of your game allows you to create a high-score chart that can be displayed when the game is at the main menu.

Resources

In addition to the information provided in this appendix to help you complete the Lunar Lander game, and the additional code supplied with the book, the following resources may help you complete and/or customize your game:

- *PhET Lunar Lander*: A number of resources including an online version of the game, provided as part of a teaching aid.
<http://phet.colorado.edu/en/simulation/lunar-lander>.
- *LunarView.java*: A Java-based implementation of the Lunar Lander game for the Android mobile device.
<http://developer.android.com/resources/samples/LunarLander/src/com/example/android/lunarlander/LunarView.html>.
- *Code Project*: A .NET-based implementation of the Lunar Lander game, written in C#. <http://www.codeproject.com/KB/game/lunarlander.aspx>.
- *History of Lunar Lander*: Wikipedia's history of the Lunar Lander game. [http://en.wikipedia.org/wiki/Lunar_Lander_\(video_game\)](http://en.wikipedia.org/wiki/Lunar_Lander_(video_game)).

Index

■ A

- Accelerate Framework
 - (Accelerate.framework), 96
- Accelerometer class, 273
- accelerometers, 271–273
- action sheets, 185
- Active Server Pages (ASP), 92
- activity indicator, and progress indicator, 174–175
- ADC (Apple Developer Connection), 4, 11
- add() method, 226
- Add Repository option, 32
- addHighScoreEntry, 199–203
- addObject method, 190–191, 199–200
- Address Book UI
 - (AddressBookUI.framework), 93
- AddressBook
 - (AddressBook.framework), 94
- Adhoc mechanism, publishing via, 253–254
- Adhoc method, 260
- Adobe Flash Professional Creative Studio 5 platform, 21–22
- alert sheet, 185
- alerts, 184–185
- alloc() method, 36–37, 40, 46
- API limits, 5
- App Design Strategies, 30
- App Store
 - platform, 22–25
 - selling apps at, 23–24
 - submitting apps to, 24–25
 - publishing via, 254–259
 - preparing for submission, 254–256
 - uploading application binary, 256–259
- Appcelerator Titanium Mobile platform, 18–19, 69–75
 - Hello, World application using, 70–75
 - installing, 69–70
- Apple
 - components, application development using, 10–13
 - platforms and technologies, 7–13
 - application development using Apple components, 10–13
 - iOS, 9–10
 - terminology and concepts used by, 7–9
 - Apple Developer Agreement, 209, 218–219, 221
 - Apple Developer Connection (ADC), 4, 11
 - Apple Developers, registering as, 2–4
 - Apple Operating System. *See* iOS
 - Apple resources, on UIs, 185–186
 - application home directory, 89
 - Application Loader, 256–257
 - Application object, 82
 - application sandbox, 89
 - application types, and associated view controllers, 154–157
 - navigation-based applications, 156–157
 - tab bar-based applications, 155–156
 - utility-based applications, 154–155
 - applicationDidBecomeActive method, 86–87

- applicationDidEnterBackground
 - method, 87
 - ApplicationID, 89
 - applications, 81–83, 223–260
 - .NET Framework comparison, 93–94
 - behaviors in, 88–89
 - and application sandbox, 89
 - multitasking, 89
 - orientation changes, 88
 - debugging
 - capturing diagnostics with NSLog command, 232
 - profiling applications, 233–237
 - with simulator, 237–239
 - Xcode 4 debugger, 229–231
 - deploying, 240–252
 - creating certificate to sign application, 241–242
 - Provisioning Portal feature, 244–250
 - registering device, 243–244
 - design considerations, 82
 - design patterns, 82–83
 - developing, resources on, 259–260
 - development of
 - native applications, 6–7
 - using Apple components, 10–13
 - web applications, 6
 - initializing, 45–47
 - life cycle of, 85–86
 - managing data within, 189–191
 - navigation-based, and associated view controllers, 156–157
 - profiling, 233–237
 - publishing, 253–259
 - resources on, 259–260
 - via Adhoc mechanism, 253–254
 - via App Store, 254–259
 - selling at App Store platform, 23–24
 - states of, 86–87
 - submitting to App Store platform, 24–25
 - tab bar-based
 - and associated view controllers, 155–156
 - implementing, 157–163
 - testing, 223–229
 - on devices, 239–240
 - resources on, 259–260
 - unit, 224–229
 - uploading binary, 256–259
 - utility-based, and associated view controllers, 154–155
 - Applications folder, 63, 65, 69, 77
 - applicationWillEnterForeground
 - method, 87
 - applicationWillResignActive method, 87
 - ARC (automatic reference counting), 40, 54–56
 - ASP (Active Server Pages), 92
 - assemblies, in .NET framework, 217–218
 - Attribute Inspector, 108
 - automatic reference counting (ARC), 40, 54–56
 - AV Foundation
 - (AVFoundation.framework), 94
- B**
- bbitem IBOUTLET, 168
 - behaviors, 88–89
 - and application sandbox, 89
 - multitasking, 89
 - orientation changes, 88
 - bespoke methods, 139, 141, 143
 - binary, applications, 256–259
 - Block objects, 83
 - build folder, 73
 - Build Phases tab, 193
 - Builder file, 44, 268, 276
 - Button class, 177
- C**
- C# class, 100
 - C# interface, 100
 - Calculator class, 226
 - CalculatorTest.m file, 229
 - cameras, 266–267
 - basics of, 266–267
 - example application for, 267–271

- updated features in iOS 5, 279–280
- cameraViewController, 269
- Cascading Style Sheets (CSS), 10
- certificates, creating to sign application, 241–242
- CFNetwork (CFNetwork.framework), 94
- CGRect, 135, 140
- CIL (Common Intermediate Language), 15
- Class class, 136
- class keyword, 100
- classes, in Objective-C, 38–39, 97
- CLLocation class, 262–265
- CLLocation parameter, 262
- CLLocationManagerDelegate, 262–263
- Close() method, 128
- CLR (Common Language Runtime), 40
- Cocoa Touch, 7–8, 10, 12, 14–15
- code completion, in IDE workspace, 107
- code snippets, in XCode 4, 111
- collection classes, 190–191
- comments, in Objective-C, 104
- common controls, 178
- Common Intermediate Language (CIL), 15
- Common Language Runtime (CLR), 40
- Connections Inspector, 108
- constants, self-documenting code, 137
- constraints, for third-party tools, 57–58
- content views, 180–183
 - table, 180–181
 - text, 181–182
 - web, 182–183
- controllers, specific to iPad, 163–174
 - Popover view, 163–171
 - split-view, 171–174
- controls, 174–185
 - action sheets, 185
 - activity and progress indicators, 174–175
 - alerts, 184–185
 - common, 178
 - content views, 180–183
 - table, 180–181
 - text, 181–182
 - web, 182–183

- date and time and general pickers, 175–176
- detail disclosure button, 176
- info button, 176–177
- navigation and information bars, 179–180
 - navigation, 180
 - status, 179
 - toolbar, 179
- page indicator, 177
- search bar, 177
- segmented, 178
- switch, 177
- Core Data (CoreData.framework), 95
- Core Graphics
 - (CoreGraphics.framework), 94
- Core Mono component, 14–15
- Core OS, 10
- Core Services, 7, 10, 13, 94
- Core Telephony
 - (CoreTelephony.framework), 95
- Core Text (CoreText.framework), 94
- CoreData.framework (Core Data), 95
- CoreGraphic.framework, 94, 214
- CoreTelephony.framework (Core Telephony), 95
- CoreText.framework (Core Text), 94
- createTabGroup() method, 73
- CS (Creative Studio), 21–22
- CSS (Cascading Style Sheets), 10

D

- databases
 - connecting to, 197
 - iOS-embedded, 192–197
 - creating or opening database, 194
 - creating table in database, 195
 - reading data from database, 196–197
 - SDK options for, 193–194
 - writing data to database, 195–196
- DataView control, 181

- date and time picker, and general pickers, 175–176
 - DateTimePicker class, 175
 - dealloc() method, 52, 54–55, 122, 125
 - deallocate memory, 270
 - Debug area view, 110
 - Debug-iphoneros folder, 213
 - Debug-iphonesimulator, 213
 - debugging
 - capturing diagnostics with NSLog command, 232
 - profiling applications, 233–237
 - with simulator, 237–239
 - changing device, 238
 - changing iOS version, 238
 - Home feature, 239
 - Lock feature, 239
 - Simulate Hardware Keyboard feature, 239
 - simulating movement, 238
 - Toggle In-Call Status Bar feature, 239
 - triggering low memory, 238
 - TV Out feature, 239
 - Xcode 4 debugger, 229–231
 - Debug.WriteLine, 232
 - Declare class, 129
 - delegation, 83, 96, 103–104
 - deploying, 240–252
 - creating certificate to sign application, 241–242
 - Provisioning Portal feature, 244–250
 - registering device, 243–244
 - design, application
 - considerations, 82
 - patterns, 5, 82–83
 - desiredAccuracy property, 263
 - detail disclosure button, 176
 - developing, resources on, 259–260
 - device compatibility, 5
 - devices
 - changing in simulator, 238
 - form factor, example applications using, 147–149
 - orientation of, supporting, 150–154
 - platforms and, constraints for, 146–154
 - registering, 243–244
 - targeting multiple with code, 276–277
 - testing on, 239–240
 - diagnostics, capturing with NSLog command, 232
 - didAccelerate method, 273
 - didFailWithError, 262–263
 - didFinishLaunchingWithOptions() method, 48, 86, 122
 - didFinishPickingMediaWithInfo method, 269
 - DidLoad event, 273
 - DidLoad method, 266
 - didReceivedMemoryWarning event, 238
 - didRotateFromInterfaceOrientation, 150
 - didUpdateToLocation, 262–263
 - dismissModalViewControllerAnimated, 128, 133
 - dismissPopoverAnimated, 169–170
 - displays, size and resolution of, 146–150
 - example applications using device form factor, 147–149
 - points vs. pixels, 149
 - screen size, 149–150
 - DragonFire SDK, 2, 16–18
 - drawRect method, 135, 140, 283
 - dynamic libraries, 208–209
- ## E
- Editor area, 48
 - enabling, ARC, 55
 - Engine class, 37
 - Engine example, 39
 - Engine object, 37
 - enumerated types, self-documenting code, 138
 - Event Kit (EventKit.framework), 95
 - exception handling, 39–40, 96
 - External Accessory (ExternalAccessory.framework), 96

F

File Inspector, 108
 file structure, of projects, 44–45
 file system-based storage, using
 sandbox, 188–189
 FirstView, 161–162
 form factor, of devices, example
 applications using, 147–149
 Foundation.framework, 95, 214
 Frameworks, 159
 fromInterfaceOrientation, 150–151

G

game events, catching, 283
 Game interface, 127
 Game Kit (GameKit.framework), 93
 game states, for Lunar Lander
 application, 118
 GameDifficulty, 129–130, 138
 GameKit.framework (Game Kit), 93
 games, implementing physics in,
 281–282
 gravity, 281
 rotation, 282
 thrust, 282
 GameState, 129–130, 138
 GameView class, 120, 126, 128–129,
 135–137
 GameView header, for Lunar Lander
 application, 128–137
 GameView interface, 126
 GameView object, 137
 GameViewController class, 120,
 123–126, 128–129, 132
 GameViewController property, 125
 GameView.h file, 128
 GameView.xib file, 138
 GDI (Graphics Display Interface), 92
 General Public License (GPL), 219
 GeoCoordinate class, 265
 GeoCoordinateWatcher object, 265
 GeoPositionAccuracy property, 265
 gesture detection, 274–275
 swipes, 275

 touch events, 274–275
 getter method, 98
 Github library, 221
 Global Positioning System. *See* GPS
 GPL (General Public License), 219
 GPS (Global Positioning System),
 261–265
 location-based services
 implementing, 262–264
 overview, 262
 uses for, 264–265
 graphical user interface (GUI), 145
 Graphics Display Interface (GDI), 92
 graphics, handling, 283
 gravity, 281
 GUI (graphical user interface), 145

H

hardware requirements, for iOS SDK, 28
 header file, 38–39, 45, 48, 51
 Hello, World application
 using Appcelerator Titanium Mobile
 platform, 70–75
 using Marmalade SDK, 77–78
 using MonoTouch component,
 66–68
 HelloWorldAppDelegate, 44
 HelloWorld.cs file, 61
 HelloWorld.exe, 62
 HelloWorldViewController, 45, 48–49,
 51–52
 HelloWorldView.xib file, 68
 high-score class, persistent, 197–201
 initializing, 203
 testing, 201–203
 high-score example, 197–203
 vs. .NET implementation, 204–205
 persistent high-score class, 197–201
 initializing, 203
 testing, 201–203
 high scores, displaying, 284
 HighScore class, 199
 HighScoreEntry class, 198–201, 203
 HighScore.h file, 212, 216
 Home feature, of simulator, 239

-
- iAd (iAd.framework), 92
- IBAction property, 123, 126, 128–129, 167, 169
- IBOutlet property, 51–52, 167
- iCloud applications, 277–278
- Icon file, 252
- icon.png file, 252
- ID type, 102
- IDE (integrated development environment), 7, 9, 29
- IDE workspace, 106–108
 - code completion in, 107
 - project editor, 108
 - schemes, 107–108
- Identity Inspector, 108
- Image file, 12
- Image I/O (ImageIO.framework), 94
- Image property, 267
- ImageIO.framework (Image I/O), 94
- iMessage service, iOS 5, 279
- Implementation file, 45, 226
- info button, 176–177
- Info.plist file, 85–86, 179
- information bars, navigation bars and, 179–180
- initializing
 - application, 45–47
 - views, 48–53
- initWithParameters method, 198–199, 201, 203
- Insert() method, 204
- Inspector pane, 48
- inspectors, in XCode 4, 108
- installing, iOS SDK, 30–35
- integrated development environment (IDE), 7, 9, 29
- integration testing, 225
- Interface Builder, 12, 47–48
- interface controls, 153, 174, 178, 186
- interfaces, 82, 84, 88, 91–92, 96–97, 100–103
- Internet-aware table, 220
- Internet, using to store data, 192
- iOS (Apple Operating System), 9–10, 278–280
 - changing version in simulator, 238
 - iMessage service, 279
 - integrated Twitter service capability, 279
 - libraries, vs. .NET framework
 - libraries, 209–210
 - Newsstand application, 279
 - Notification Center feature, 278–279
 - Reminders application, 279
 - SDK, 12–13
 - updated features, 279–280
- iOS Dev Center, 3, 8
- iOS Developer, 3–4, 10
- iOS-embedded databases
 - creating or opening, 194
 - creating table in, 195
 - reading data from, 196–197
 - SDK options for, 193–194
 - writing data to, 195–196
- iOS Human Interface Guidelines, 30, 115
- iOS SDK, 27–56
 - ARC, 54–56
 - enabling, 55
 - migrating to, 55
 - overview, 55
 - programming with, 55
 - creating user interface, 47–53
 - initializing view, 48–53
 - using Interface Builder, 47–48
 - hardware requirements for, 28
 - initializing application, 45–47
 - installing, 30–35
 - Objective-C, 35–40
 - classes in, 38–39
 - exception handling, 39–40
 - importing, 38
 - memory management, 40
 - naming conventions, 38
 - object model, 36–37
 - square brackets, 37–38
 - terminology for, 36
 - projects in
 - creating, 41–44
 - file structure of, 44–45
 - resources for, 30

- Xcode, new features for, 29
- iOS user-interface controls, 115
- iPad
 - controllers specific to, 163–174
 - Popover view, 163–171
 - split-view, 171–174
 - targeting multiple devices with code, 276–277
- iPhone, targeting multiple devices with code, 276–277
- ISerializable class, 190
- iTunes Connect, 254–257, 259–260

J

- jailbreaking, 24

K

- kCLLocationAccuracyBest, 263–264
- kUTTypeImage, 266, 268–269
- kUTTypeMovie, 266

L

- Label (UILabel), 178
- lander_nothrust property, 130, 134–135, 139–140
- Lander.tiff, 139
- last-in-first-out (LIFO), 157
- libraries, 207–221
 - Apple Developer Agreement, 218–219
 - definition of, 208
 - dynamic, 208–209
 - iOS vs. .NET framework, 209–210
 - static, 208–218
 - assemblies in .NET framework, 217–218
 - with Xcode 4 tool, 210–217
 - third-party, 219–221
 - categories of, 219
 - Github library, 221
 - list of useful, 220
 - SourceForge library, 221
 - types of, 208

- Library pane, 48, 50
- libsqlite3.dylib library, 193, 209, 217
- licensing, 5
- life cycle, of applications, 85–86
- LIFO (last-in-first-out), 157
- linker, 208
- LLVM compiler, 107
- loadRequest, 182–183
- Localizable.string file, 100
- location-based services, implementing, 262–264
- LocationManager class, 262–264
- Lock feature, of simulator, 239
- low memory, triggering in simulator, 238
- Lunar Lander application, 113–143, 281–284
 - catching game events, 283
 - creating project, 119–121
 - displaying high score, 284
 - enabling user interaction, 282–283
 - GameView header, 128–137
 - handling graphics, 283
 - implementing game physics, 281–282
 - gravity, 281
 - rotation, 282
 - thrust, 282
 - implementing navigation, 127–128
 - initializing XIB resource, 138–140
 - manually drawing user interface, 140
 - planning for, 114–118
 - design resources, 115–116
 - game states, 118
 - requirements specification, 116–117
 - user interfaces, 118
 - resources for, 284
 - self-documenting code, 137–138
 - using constants, 137
 - using enumerated types, 138
 - testing, 141–143
 - user interface, 121–126
 - using bespoke methods, 141
- Lunar Lander graphic, 118
- LunarLanderAppDelegate, 119, 122, 124

LunarLanderViewController class, 119,
121–124, 126, 128
LunarLanderViewController.xib file, 123,
126
Lunary Lander game, 226

M

.m extension, 39
main() method, 45–47
main .nib file, 85
MainViewController.m file, 216
MainWindow.xib file, 44, 48, 85, 119,
161
makeKeyAndVisible, 48–49
Managed memory model, 83
Map Kit (MapKit.framework), 92
Marmalade SDK (Software
Development Kit), 19–21, 75–78
Hello, World application using,
77–78
installing, 75–76
Marmalade Studio, 20, 75–76
Marmalade System, 20, 75–76
Media layer, 10, 94
Media Player (MediaPlayer.framework),
94
MediaLibrary object, 267
MediaPlayer.framework (Media Player),
94
mediaType, 269–270
mediatypes property, 266
memory, low, 238
memory management, 40, 95
message file, 39
Message UI (MessageUI.framework), 92
methods
calling with square brackets, 37
in Objective-C, declaring, 97–98
Microsoft Developer Network (MSDN),
83
Microsoft.Devices.PhotoCamera class,
267
Microsoft.Devices.Sensors namespace,
273
migrating, to ARC, 55

mobile device, 85, 93–95
MobileCoreServices.framework, 267
.mobileprovision file, 253
Model-View-Controller (MVC), 5, 83
Mono environment, 14–16
Core Mono component, 14–15
installing, 59–62
MonoDevelop component, 15–16
MonoTouch component, 15
and MonoTouch component, 58–68
Hello, World application using,
66–68
installing, 59–66
MonoDevelop component
installing, 62–64
overview, 15–16
MonoDroid, 58
MonoMac, 58
MonoTouch component, 15
installing, 64–66
Mono environment and, 58–68
Hello, World application using
MonoTouch component, 66–68
installing, 59–66
Motion class, 273
movement, simulating, 238
MSDN (Microsoft Developer Network),
83
multidevice capable, 277
multitasking, 89, 96
multitaskingSupported property, 89
MVC (Model-View-Controller), 5, 83
MyStaticLibrary, 212

N

Name property, 162
naming conventions, for Objective-C,
38
native applications, development of,
6–7
Navigate() method, 183
navigation bars, and information bars,
179–180
navigation, 180
status, 179

- toolbar, 179
- navigation-based applications, and associated view controllers, 156–157
- navigation, for Lunar Lander application, 127–128
- Navigator view, 110
- navigators, in XCode 4, 109–110
- NDA (nondisclosure agreement), 218
- NET control, 156, 178, 265
- .NET Framework, 90–96
 - application services, 93–94
 - assemblies in, 217–218
 - libraries, iOS libraries vs., 209–210
 - runtime services, 95–96
 - tools for, vs. Xcode tools, 105–106
 - user-interface services, 91–92
- .NET implementation, vs. high-score example, 204–205
- NewGame() method, 130, 132, 134, 139–141
- newMediaAvailable, 267–269
- Newsstand application, iOS 5, 279
- NIB file, 106, 120, 162, 173
- Nil object, 35, 37
- nondisclosure agreement (NDA), 218
- NormalizeAngle() method, 282
- Notification Center feature, iOS 5, 278–279
- NSArray, 190–191
- NSAutoRelease class, 46
- NSCachesDirectory, 188
- NSClassFromString() method, 277
- NSData, 190
- NSDate, 190
- NSDictionary, 190–191
- NSLog command, capturing diagnostics with, 232
- NSLog() method, 109, 196
- NSMutableArray, 190–192, 198–200, 203–204
- NSMutableDictionary class, 190
- NSNumber, 190
- NSObject, 160
- NSSet, 274–275

- NSString class, 190, 192, 194, 196–198, 200–203
- NSTimeInterval() method, 281
- NSTimer class, 128, 135, 143

O

- Object class, 190
- object model, for Objective-C, 36–37
- Objective-C, 35–40, 96–104
 - classes in
 - declaring, 97
 - overview, 38–39
 - comments, 104
 - delegation, 103–104
 - exception handling, 39–40
 - importing, 38
 - interfaces and protocols, 100–103
 - memory management, 40
 - methods in, declaring, 97–98
 - naming conventions, 38
 - object model, 36–37
 - properties, 98–99
 - square brackets, 37–38
 - calling methods, 37
 - passing and retrieving with, 38
 - strings, 99–100
 - terminology for, 36
- ODBC (Open Database Connectivity), 197, 204
- OpenAL and OpenGL ES (OpenAL.framework), 94
- OpenGLES.framework, 94
- orientation changes, 88
- orientation, of devices, supporting, 150–154

P

- page indicator, 177
- passing the call along the chain, 125
- passing, with square brackets, 38
- pathForResource method, 134, 139
- PDF (Portable Document Format), 277
- persistent high-score class, 197–201
 - initializing, 203

- testing, 201–203
- photos, updated features in iOS 5, 279–280
- physics, implementing in games, 281–282
 - gravity, 281
 - rotation, 282
 - thrust, 282
- Picker class, 176
- Picker control, 115
- pickers, date and time and general, 175–176
- pixels per inch (PPI), 150
- pixels, points vs., 149
- planning
 - for Lunar Lander application, 114–118
 - design resources, 115–116
 - game states, 118
 - requirements specification, 116–117
 - user interfaces, 118
- platforms
 - and devices, constraints for, 146–154
 - and technologies, Apple, 7–13
- points, vs. pixels, 149
- Popover view controllers, 163–171
- PopoverExampleViewController, 167
- PopoverSelection, 165
- PopoverSelection class, 166–167, 169–170
- PopoverSelection.h file, 166–167
- PopoverSelection.m file, 166
- PopoverSelection.xib file, 166
- Portable Document Format (PDF), 277
- PPI (pixels per inch), 150
- presentPopoverFromBarButtonItem, 169–170
- profiling applications, 233–237
- programming, with ARC, 55
- progress indicator, activity indicator and, 174–175
- ProgressBar class, 175
- project editor, IDE workspace, 108
- project navigator, 109

- projects
 - creating, 41–44
 - file structure of, 44–45
- properties, in Objective-C, 98–99
- property lists, using as storage, 191
- protocols, in Objective-C, 100–103
- Provisioning Portal feature, 244–250
- Public class, 101
- publishing, 253–259
 - resources on, 259–260
 - via Adhoc mechanism, 253–254
 - via App Store, 254–259
 - preparing for submission, 254–256
 - uploading application binary, 256–259

Q

- Quick Help, 108
- quitGame, 128–129, 133, 141
- QuitGame() method, 132

R

- RDBMS (relational database management system), 192
- readHighScores method, 202
- registering, device, 243–244
- relational database management system (RDBMS), 192
- Reminders application, iOS 5, 279
- Remove() method, 204
- requirements capture stage, 114
- resolution, size and, of displays, 146–150
- resources folder, 73
- resources, for iOS SDK, 30
- retain method, 51–52, 54–55
- retrieving, with square brackets, 38
- rootViewController, 161, 173–174
- rotateAt() method, 283
- RotateLeft() method, 129–130, 132–133, 135, 141
- RotateRight() method, 129–130, 132–133, 135, 141

- rotation, 282
 - Round rect button (UIButton), 178
 - runtime services, 91, 95–96
- S**
- s3dHelloWorld.mkb file, 77
 - .s3e files, 77
 - Safari Browser, updated features in iOS 5, 279–280
 - sandboxes, file system-based storage using, 188–189
 - scheduledTimerWithTimeInterval method, 133, 135–136
 - Schema-driven object, 193
 - schemes, IDE workspace, 107–108
 - Score class, 204
 - Score table, 181
 - screens, size of, 149–150
 - SDKs (Software tools Kits)
 - completeness, 58
 - DragonFire, 16–18
 - iOS, 12–13
 - Marmalade, 19–21
 - options for iOS-embedded databases, 193–194
 - search bars, 177
 - SecondView, 161–162
 - security, 96
 - Security (Security.framework), 96
 - segmented controls, 178
 - SELECT statement, 196, 203
 - selector parameter, 133, 135–136
 - self-documenting code, 137–138
 - using constants, 137
 - using enumerated types, 138
 - self.view property, 153
 - Server database, 197
 - setNeedsDisplay method, 133, 136–137
 - setter method, 98
 - Short Message Service (SMS), 92
 - ShowDialog() method, 126
 - signal strength, 239
 - Simple Object Access Protocol (SOAP), 190
 - Simulate Hardware Keyboard feature, of simulator, 239
 - simulators, debugging with, 237–239
 - changing device, 238
 - changing iOS version, 238
 - Home feature, 239
 - Lock feature, 239
 - Simulate Hardware Keyboard feature, 239
 - simulating movement, 238
 - Toggle In-Call Status Bar feature, 239
 - triggering low memory, 238
 - TV Out feature, 239
 - Size Inspector, 108
 - Slider (UISlider), 178
 - SMS (Short Message Service), 92
 - SOAP (Simple Object Access Protocol), 190
 - Software Development Kit. See SDK
 - Sort() method, 204
 - sortArrayUsingSelector method, 203–204
 - SourceForge library, 221
 - split-view controllers, 171–174
 - SplitContainer class, 174
 - sprintf() method, 232
 - SQL command, 195
 - SQL statement, 195–196, 201, 204
 - SQL (Structured Query Language), 192
 - SQLite database, 13
 - sqlite3_bind_text() method, 196
 - sqlite3_prepare() method, 196, 200–202
 - sqlite3_step() method, 196
 - sqlite_open() method, 194
 - square brackets, 37–38
 - calling methods, 37
 - passing and retrieving with, 38
 - Start Game button, 121, 123, 125–126
 - Start Touch Down event, 126
 - startAnimating method, 175
 - startUpdatingLocation, 262, 264
 - states, of applications, 86–87
 - static analysis, in XCode 4, 111

- static libraries, 208–218
 - assemblies in .NET framework, 217–218
 - with Xcode 4 tool, 210–217
- statically linked libraries, 208
- status bar, 179
- StatusBar class, 179
- STFail() method, 226
- stopUpdatingLocation, 262
- storage, 187–205
 - high-score example, 197–203
 - vs. .NET implementation, 204–205
 - persistent high-score class, 197–201
- options for data, 188–197
 - databases, 192–197
 - file system-based storage using sandbox, 188–189
 - Internet, 192
 - managing within application, 189–191
 - property lists, 191
- Store Kit (StoreKit.framework), 95
- strings, in Objective-C, 99–100
- Structured Query Language (SQL), 192
- Such class, 98
- super initWithCoder:aDecoder
 - command, 139
- swipes, detecting, 275
- switch control, 177
- symbol navigator, 109
- synthesizing, 98
- System.Collections.Generic.Dictionary
 - class, 190
- System.Date class, 190
- System.Device.Location namespace, 265
- System.Diagnostics namespace, 232
- System.Runtime.Serialization.Formatter
 - s.Binary, 190
- Systems Configuration
 - (SystemsConfiguration.framework), 95
- System.String class, 190
- System.Threading.Timer class, 135

T

- tab bar-based applications
 - and associated view controllers, 155–156
 - implementing, 157–163
- Tab object, 73
- tabBarController property, 160–161
- TabBarExample, 158–160
- TabBarExampleAppDelegate.h file, 160
- TabBarExampleAppDelegate.m file, 161
- TabControl, 156
- TABLE command, 195
- table view, 180–181
- tables, creating in database, 195
- Target-action, 83
- TDD (Test Driven Development), 224–226
- technologies, platforms and, 7–13
- terminology, for Objective-C, 36
- Test & Package option tab, 72
- Test Driven Development (TDD), 224–226
- testExample() method, 226–228
- testing, 223–229
 - on devices, 239–240
 - integration, 225
 - Lunar Lander application, 141–143
 - persistent high-score class, 201–203
 - resources on, 259–260
 - unit, 224–229
 - defining approach to, 224–226
 - writing and running, 226–229
- TestMethod() method, 217–218
- Text field (UITextField), 178
- text view, 181–182
- third-party libraries, 219–221
 - categories of, 219
 - Github library, 221
 - list of useful, 220
 - SourceForge library, 221
- third-party tools, 5–6, 13–22, 57–78
 - Adobe Flash Professional Creative Studio 5 platform, 21–22
 - Appcelerator Titanium Mobile platform, 18–19, 69–75

- Hello, World application using, 70–75
 - installing, 69–70
 - constraints, 57–58
 - DragonFire SDK, 16–18
 - Marmalade SDK, 19–21, 75–78
 - Hello, World application using, 77–78
 - installing, 75–76
 - Mono environment, 14–16
 - Core Mono component, 14–15
 - MonoDevelop component, 15–16
 - MonoTouch component, 15
 - Mono environment and MonoTouch component, 58–68
 - Hello, World application using, 66–68
 - installing, 59–66
- thrust, 282
- Thrust() method, 128–129, 132
- thrustEngine, 130, 134–135, 141
- ThrusterState, 129–130, 138
- thrustspeed value, 282
- Titanium Developer icon, 69
- Titanium.UI namespace, 73
- Toggle In-Call Status Bar feature, of simulator, 239
- togglePopOverController, 167–169
- toolbar, 179
- Toolbar class, 165, 179
- tools
 - for .NET Framework, vs. Xcode tools, 105–106
 - accelerometer, 271–273
 - App Store platform, 22–25
 - selling apps at, 23–24
 - submitting apps to, 24–25
 - Apple platforms and technologies, 7–13
 - application development using Apple components, 10–13
 - iOS, 9–10
 - terminology and concepts used by, 7–9
 - application development
 - native, 6–7
 - web, 6
 - camera
 - basics of, 266–267
 - example application for, 267–271
 - development principles, 5–6
 - future directions in, 277–280
 - iCloud applications, 277–278
 - iOS 5, 278–280
 - gesture detection
 - swipes, 275
 - touch events, 274–275
 - GPS, 261–265
 - location-based services, 262
 - uses for, 264–265
 - registering as Apple Developer, 2–4
 - targeting multiple devices with code, 276–277
 - third-party options, 13–22
 - Adobe Flash Professional Creative Studio 5 platform, 21–22
 - Appcelerator Titanium Mobile platform, 18–19
 - DragonFire SDK, 16–18
 - Marmalade SDK, 19–21
 - Mono environment, 14–16
 - totalseconds value, 282
 - touch events, detecting, 274–275
 - touchesBegan method, 274–275
 - touchesCancelled:withEvent, 274
 - touchesEnded:withEvent, 274
 - transform, 283
 - TV Out feature, of simulator, 239
 - Twitter service, integrated capability in iOS 5, 279
 - type management, 95

U

- UIAccelerometer class, 272
- UIAccelerometerDelegate protocol, 272–273
- UIActionSheet class, 185
- UIActivityIndicatorView class, 175
- UIAlertView class, 184
- UIApplication, 179

- UIApplicationDelegate protocol, 86, 160
- UIApplicationMain() function, 46–47, 85
- UIButton class, 123, 176
- UIButton (Round rect button), 178
- UIDatePicker class, 175
- UIDevice, 89
- UIEvent, 274–275
- UIGestureRecognizer class, 274
- UIImage variable, 130, 134, 139–140, 236
- UIImagePickerController class, 266–270
- UIImagePickerController object, 269
- UIImagePickerControllerSourceTypePhotoLibrary, 269
- UIImageView control, 268
- UIInterfaceOrientation, 150, 153, 169
- UIKit (UIKit.framework), 92, 214
- UILabel control, 50
- UILabel object, 49, 51–52
- UIImage class, 143
- UINavigationController class, 180
- UINavigationController interface, 157
- UIPageControl class, 177
- UIPopoverController class, 164, 167, 170
- UIProgressView class, 175
- UIs (User Interfaces), 145–186
 - Apple resources on, 185–186
 - application types and associated view controllers, 154–157
 - navigation-based applications, 156–157
 - tab bar-based applications, 155–156
 - utility-based applications, 154–155
- controls, 174–185
 - action sheets, 185
 - activity and progress indicators, 174–175
 - alerts, 184–185
 - common, 178
 - content views, 180–183
 - date and time and general pickers, 175–176
 - detail disclosure button, 176
 - info button, 176–177
 - navigation and information bars, 179–180
 - page indicator, 177
 - search bar, 177
 - segmented, 178
 - switch, 177
- creating, 47–53
 - initializing view, 48–53
 - using Interface Builder, 47–48
- implementing tab bar-based application, 157–163
- iPad-specific controllers, 163–174
 - Popover view, 163–171
 - split-view, 171–174
- for Lunar Lander application, 118–121, 126
- manually drawing, 140
- platform and device constraints, 146–154
 - display size and resolution, 146–150
 - supporting device orientation, 150–154
- UISearchBar class, 177
- UISegmentedControl class, 178
- UISlider (Slider), 178
- UISplitViewController class, 172–173
- UIStatusBarHidden, 179
- UIStatusBarStyle, 179
- UISwitch class, 177
- UITabBarController class, 156, 161
- UITabBarController interface, 157
- UITabController class, 161
- UITabControllerDelegate protocol, 160
- UITableView class, 180
- UITableViewController class, 156, 165, 170
- UITextField (Text field), 178
- UITextView class, 181
- UIToolbar class, 179
- UITouch, 274–275
- UIView class, 136, 274–275
- UIViewController class, 120, 123, 129, 267, 273–275
- UIWebView class, 182

- UIWindow object, 48
- unit testing, 224–229
 - defining approach to, 224–226
 - integration testing, 225
 - TDD, 225–226
 - writing and running, 226–229
- updateInterval property, 272–273
- UpdateLander() method, 131–132, 137
- useCamera method, 267–268
- user interaction, enabling, 282–283
- user-Interface guidelines, 115
- user-interface services, .NET
 - Framework comparison, 91–92
- User Interfaces. *See* UIs
- UTCORETypes.h file, 267
- UTF8String method, 194, 196, 200
- Utilities view, 110
- utility-based applications, and
 - associated view controllers, 154–155

V

- view controllers, associated with
 - application types, 154–157
 - navigation-based, 156–157
 - tab bar-based, 155–156
 - utility-based, 154–155
- ViewController class, 68, 135–136
- viewController array, 48, 173–174, 273
- viewDidLoad event, 124, 132–133, 135, 169–170, 182
- ViewDidLoad() method, 52, 68
- viewDidLoad method, 169–170
- views
 - initializing, 48–53
 - in XCode 4, 110
- visibility modifier, 125

W

- WAP (Wireless Access Protocol), 91
- web applications, development of, 6

- web view, 182–183
- WebBrowser class, 183
- Windows Forms, 82, 90, 92–93
- Windows method, 112
- Windows Presentation Foundation (WPF), 90
- Wireless Access Protocol (WAP), 91
- World Wide Developer Conference (WWDC), 277
- WPF (Windows Presentation Foundation), 90
- writeToFile method, 191, 198
- writeToURL method, 192
- WWDC (World Wide Developer Conference), 277

X, Y, Z

- XCode 4, 106–112
 - code snippets, 111
 - debugger, 229–231
 - IDE workspace, 106–108
 - code completion in, 107
 - project editor, 108
 - schemes, 107–108
 - inspectors, 108
 - navigators, 109–110
 - static analysis, 111
 - static libraries with, 210–217
 - views in, 110
- Xcode interface, 106, 109
- Xcode tools
 - new features for, 29
 - overview, 11–12
 - vs. tools for .NET Framework, 105–106
- XIB file, 108, 121, 126, 128, 136, 138, 166
- XIB resource, initializing, 138–140
- XML file, 85, 191, 204
- XMLSerializer class, 204