Objective-C for Absolute Beginners

iPhone, iPad, and Mac Programming Made Easy

Gary Bennett
Mitch Fisher
Brad Lees

Apress®

Objective-C for Absolute Beginners: iPhone, iPad, and Mac Programming Made Easy

ISBN-13 (pbk): 978-1-4302-2832-5

ISBN-13 (electronic): 978-1-4302-2833-2

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

President and Publisher: Paul Manning
Lead Editor: Clay Andres
Development Editor: Douglas Pundick
Technical Reviewer: James Bucanek
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Coordinating Editor: Kelly Moritz
Copy Editor: Heather Lang and Tracy Brown
Compositor: MacPS, LLC
Indexer: BIM Indexing & Proofreading Services
Artist: April Milne
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/info/bulksales.

The source code for this book is available to readers at www.apress.com.

I would like to dedicate this book to my wife Stefanie and to my children, Michael, Danielle, Michelle, and Emily. Thank you for always supporting me when I decide to do crazy things like write a book.

Also, I want to thank two of my friends, Mitch Fisher and Brad Lees, for co-authoring this book with me. They are two of the finest developers in the country, and are great friends. It was great being able to work with them again.

—Gary Bennett

I would like to thank all my family and friends who have patiently supported my work on this book. You know who you are. I would like to especially thank Lisa, Jade, Eric, and Patti for the long nights and forever-busy weekends that I was spending on this book and not with them.

—Mitch Fisher

I would like to thank my wife Natalie and my kids for the support and time they have given me to work on this book. I am also grateful for good friends who convince me to take on crazy endeavors.

—Brad Lees

Contents at a Glance

Contents

About the Authors

Gary Bennett is president of `xcelMe.com`. xcelMe teaches iPhone/iPad programming courses online. Gary has taught hundreds of students how to develop iPhone/iPad apps, and has several very popular apps on the iTunes Apps Store. Gary's students have some of the best-selling apps on the iTunes App Store. Gary also worked for 25 years in the technology and defense industries. He served 10 years in the U.S. Navy as a Nuclear Engineer aboard two nuclear submarines. After leaving the Navy, Gary worked for several companies as a software developer, CIO, and President. As CIO, he helped take VistaCare public in 2002. Gary also co-authored *iPhone Cool Projects* for Apress. Gary lives in Scottsdale, Arizona with his wife Stefanie and their four children.

Mitch Fisher is a software developer in the Phoenix, Arizona area. He was introduced to PCs back in the 1980s when 64K was a lot of memory and 1 Mhz was considered a fast computer. Over the last 25 years, Mitch has worked for several large and medium-sized companies in the roles of software developer and software architect, and had led several teams of developers on multi-million dollar projects. Mitch now divides his time between writing iOS applications and server-side UNIX technologies.

Brad Lees has more than 12 years' experience in application development and server management. He has specialized in creating and initiating software programs in real-estate development systems and financial institutions. His career has been highlighted by his positions as information systems manager at The Lyle Anderson Company; product development manager for Smarsh; vice president of application development for iNation; and IT manager at The Orcutt/Winslow Partnership, the largest architectural firm in Arizona. A graduate of Arizona State University, Brad and his wife Natalie reside in Phoenix with their five children

About the Technical Reviewer

James Bucanek has spent the past 30 years programming and developing microcomputer systems. He has experience with a broad range of technologies, from embedded consumer products to industrial robotics. James is currently focused on Macintosh and iPhone software development. When not programming, James indulges in his love of the arts. He earned an Associate's degree from the Royal Academy of Dance in classical ballet, and occasionally teaches at Adams Ballet Academy.

Acknowledgments

We would like to thank Apress for all their help in making this book possible. Specifically, we would like to thank Kelly Moritz, our coordinating editor, for helping us stay focused and overcoming many obstacles. Without Kelly, this book would not have been possible.

Special thanks to Douglas Pundick, development editor, for all his suggestions during the editorial review process to help make this a great book. Thanks to Heather Lang and Tracy Brown, the copy editors who made the book look great.

We would also like to thank the Alice Community and Carnegie Mellon University for developing Alice and making learning object-oriented programming fun and easy!

Introduction

Over the last two years, we've heard this countless times: "I've never programmed before, but I have a great idea for an iPhone/iPad app. Can I really learn to program the iPhone or iPad?" We always answer, "yes, but you have to believe you can." Only you are going to tell yourself you can't do it.

For the Newbie

This book assumes you may have never programmed before. It is also written for someone who may have never programmed before using object-oriented programming (OOP) languages. There are lots of Objective-C books out there, but all of these books assume you have programmed before and know OOP. We wanted to write a book that takes readers from knowing nothing about programming to being able to program in Objective-C.

Over the last two years we of have taught hundreds of students at xcelMe.com to be iPhone/iPad developers. We have incorporated what we have learned in our first two courses, Introduction to Object Oriented Programming and Logic along and Objective-C for iPhone/iPad developers, into this book.

For the More Experienced

There are lots of developers who programmed years ago or programmed in a non-OOP language and need the background in OOP and Logic before they dive into Objective-C. This book is for you. We gently walk you through OOP and how it is used in iPhone/iPad development.

Why Alice: An Innovative 3D Programming Environment

Over the years, universities have struggled with several issues with their computer science departments:

- High male-to-female ratios
- High drop-out rates
- Longer than average time to graduation

One of the biggest challenges to learning OOP languages like Java, C++, or Objective-C is the steep learning curve from the very beginning. In the past, students had to learn at once the following topics:

- Object-oriented principles
- A complex Integrated Development Environment (IDE)
- The syntax of the programming language
- Programming logic and principles

Carnegie Mellon University received a grant from the U.S. government and developed Alice. Alice is an innovative 3D programming environment that makes it easy to create rich graphical applications for new developers. Alice is a teaching tool for students learning to program in an OOP environment. It uses 3D graphics and a drag-and-drop interface to facilitate a more engaging, less frustrating first programming experience.

Alice enables the students to focus on learning the principles of OOP without having to focus on learning a complex IDE and Objective-C principles all at once. We get to focus on each topic individually. This helps readers feel a real sense of accomplishment as they progress.

Alice removes all the complexity of learning an IDE and programming language syntax. It is drag-and-drop programming. You'll see it is actually fun to do, and you can develop really cool and sophisticated apps in Alice.

After the OOP topic has been introduced and readers feel comfortable with the material, we then move into Xcode, where readers get to use their new OOP knowledge in writing Objective-C applications. This enables readers to focus on the Objective-C syntax and language without having to learn OOP at the same time.

How This Book Is Organized

You'll notice that we are all about successes in this book. We introduce the OOP and Logic concepts in Alice and then move those concepts into Xcode and Objective-C. Most students are visual and learn by doing. We use both of these techniques. We'll walk you through topics and concepts with visual examples and then take you step-by-step examples reinforcing these.

Often we will repeat previous topics to reinforce what you have learned and apply these skills in new ways. This enables new programmers to re-apply development skills and feel a sense of accomplishment as they progress.

The Formula for Success

Learning to program is an interactive process between you and your program. Just like learning to play an instrument, you have to practice. You must work through the examples and exercises in this book. Just because you understand the concept, doesn't mean you will know how to apply it and use it.

You will learn a lot from this book. You will learn a lot from working through the exercises in this book. *But you will really learn when you debug your programs.* Spending time walking through your code and trying to find out why it is not working the way you want is a learning process that is unparalleled. The downside of debugging is it can be especially frustrating to the new developer. If you have never wanted to throw your computer out the window, you will. You will question why you are doing this, and whether you are smart enough to solve the problem. Programming is very humbling, even for the most experience developer.

Like a musician, the more you practice the better you get. You can do some amazing things as a programmer. The world is your oyster. It is one of the most satisfying accomplishments you can have, seeing you app on the iTunes App Store. However, there is a price, and that price is time spent coding.

Here is our formula for success:

- Believe you can do it. You'll be the only one who says you can't do this. So don't tell yourself that.
- Work through all the examples and exercises in this book.
- Code, code, and keeping coding. The more you code, the better you'll get.
- Be patient with yourself. If you were fortunate enough to have been a 4.0 student who can memorize material just by reading it, this will not happen with Objective-C coding. You are going to have to spend time coding.
- DON'T GIVE UP!

The Development Technology Stack

We will walk you through the process of understanding the development process for your iPhone/iPad apps and what technology is needed. However, it is helpful to briefly look at all the pieces together: a sample iPhone app, in a Table View. See Figure 1.

Figure 1. *The iPhone/iPad technology stack*

Required Software, Materials, and Equipment

One of the great things about Alice is it available on the three main operating systems used today:

- Windows
- Mac
- Linux

The other great thing about Alice is it is free! You can download Alice at `www.Alice.org`.

Operating System and IDE

Although you can use Alice on many platforms, the Integrated Development Environment (IDE) that developers use to develop iPhone/iPad apps Xcode, **has to be an Intel-based Mac!** The IDE is free and is available on your Mac DVD operating system. The operating system has to be 10.5 or later to develop iPhone apps, and 10.6 for iPad apps and iOS 4 apps.

Software Development Kits

You will need to download the iPhone/iPad IDE from Apple. This is available at `http://developer.apple.com/iphone`. You will need to register as an iPhone developer.

When you are ready to upload your app to the iTunes App Store, you will need to pay $99/yr to do this.

Dual Monitors

It is highly recommended that developers have a second monitor connected to their computer. It is great to step through your code and watch your output window and iPad simulator at the same time on dual, independent monitors. Apple hardware makes this easy. Just plug your second monitor in to the display port of any Intel-based Mac, with the correct mini display port adapter, of course, and you're able to have two monitors working independently from one another. See Figure 2. Note it is not required to have dual monitors. You will just have to organize your open windows to fit on your screen if you don't.

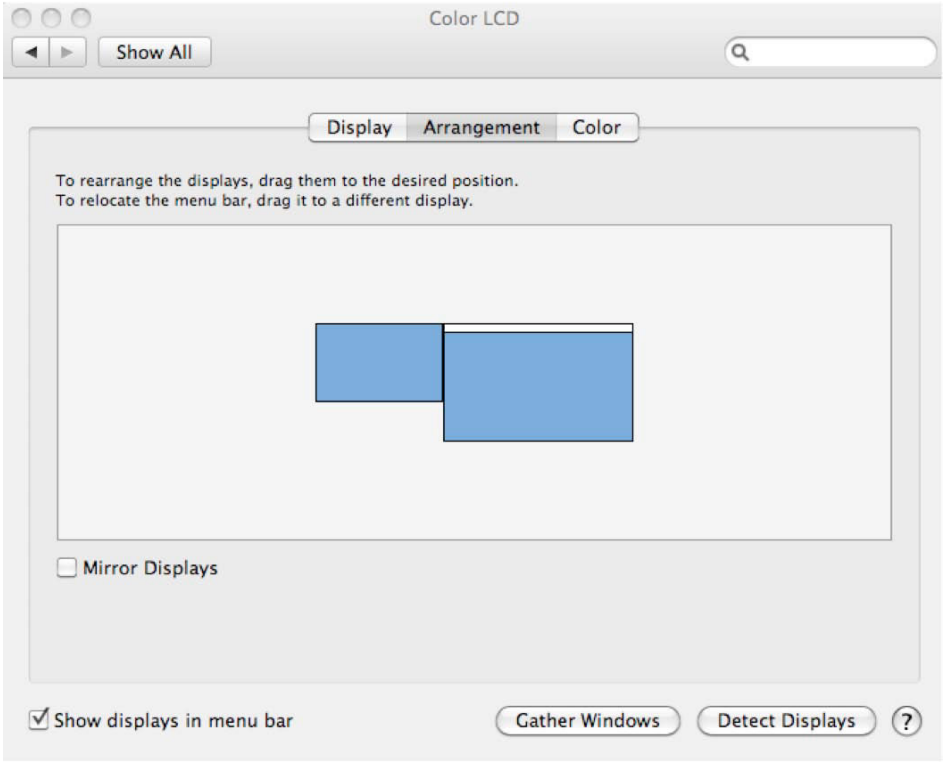

Figure 2. *Dual monitors*

Book Forum

We have developed an online forum for this book at http://forum.xcelme.com, where readers can go to ask questions of the authors while they are learning Objective-C. There you will find answers to the exercises and additional exercises to help you learn.

See Figure 3. Readers can also access answers to exercises and discover helpful links to help them be successful iPhone/iPad develops and great amazing apps. So let's get started!

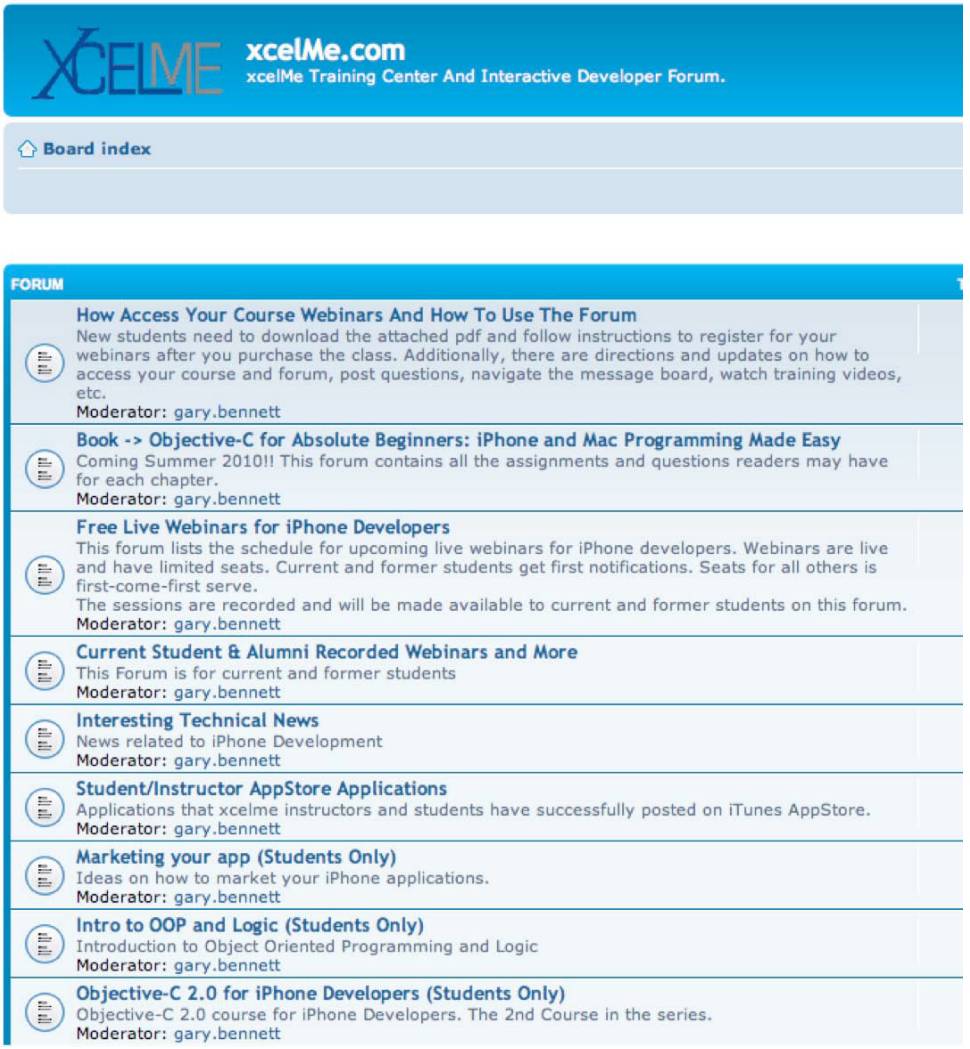

Figure 3. *Reader Forum for accessing answers to exercise and posting questions for authors*

Chapter 1

Becoming a Great iPhone/iPad or Mac Programmer

Now that you're ready to become a software developer and have read the Introduction of this book, you need to become familiar with some key concepts. Your computer program will do exactly what you tell it to do, no more and no less. It will follow the programming rules that were defined by the operating system and programming language. Your program doesn't care if you are having a bad day or how many times you ask it to perform something. Your program will do whatever you tell it to do. Often, what you think you've told your program to do and what it actually does are to different things.

> **NOTE:** If you haven't already, take a few minutes to read the Introduction to this book. You will better understand why we are using the Alice programming language and how to be successful in developing your iPhone/iPad and Mac apps.

Depending on your background, working with something absolutely black and white may be frustrating. Many times, programming students have lamented, "That's not what I wanted it to do!" As you begin to gain experience and confidence programming, you'll begin to think like a programmer. You will understand software design and logic, and you will experience having your programs perform exactly as you tell them to do as enormously satisfying.

Thinking Like a Developer

Software development involves writing a computer program and then having the computer execute the program. A **computer program** is the set of instructions that we

want the computer to perform. Before we begin to write a computer program, it is helpful to list the steps that we want our program to perform, in the order we want them accomplished. This step-by-step process is called an **algorithm**.

If we want to write a computer program to toast a piece of bread, we would first write an algorithm. This algorithm might look something like this:

1. Take the bread out of the bag.
2. Place the bread in the toaster.
3. Press the toast button.
4. Wait for the toast to pop up.
5. Remove the toast from the toaster.

At first glance, this algorithm seems to solve our problem. However, our algorithm leaves out many details and makes many assumptions, for example:

1. What kind of toast does our user want? Does the user want white bread, wheat, or some other kind of bread?
2. How does the user want the bread toasted, light or dark?
3. What does the user want on the bread after it is toasted: butter, margarine, honey, or strawberry jam?
4. Maybe the user wanted another kind of toast, like French toast or garlic toast, which is prepared by means other than a toaster.
5. Does this algorithm work for all your users in their cultures and languages?

Now, you might be thinking we are getting too detailed for just doing a simple toast program. Over the years, software development has the reputation of taking too long, costing too much, and not being what the user wants. This reputation came to be because computer programmers often start writing their programs before they have really thought through their algorithms.

The key ingredients to making successful applications starts with **design requirements**. Design requirements can be very formal and detailed or a simple list on a piece of paper. The importance of design requirements is they help the developer flesh out what the application should do and not do when complete. Design requirements should not be completed in a programmer's vacuum but should be a collaboration between developers, users, and customers.

NOTE: If you take anything away from this chapter, take away the importance of considering design requirements and user interface design before starting software development. This is the most effective (and cheapest) use of time in the software development cycle. Using a pencil and eraser is a lot easier and faster than making changes to code because you didn't have others look at the designs before starting to program.

Another key ingredient to your successful app is the **user interface (UI)** design. Apple recommends that you spend over 50% of the entire development process focusing on the UI design. The design can be simple pencil-and-paper layouts created using the *iPhone Application Sketch Book* by Dean Kaplan (Apress, 2009) or on-screen layout created with the Omni Group's OmniGraffle software application with the Ultimate iPhone Stencil plug-in. Many software developers start with the UI design, and after laying out all the screen elements and having many users look at paper mock-ups, they then write out the design requirements from their screen layouts.

Once you have done your best to flesh out all the design requirements, laid out all the user interface screens, and had the client(s) or potential customers look at your design and give you feedback, coding can begin. Once coding begins, design requirements and user interface screens can change, but the changes are typically minor and easily accommodated by the development process. See Figures 1–1 and 1–2.

Figure 1–1. *This is a UI mock-up of the Account Balance screen for an iPhone mobile banking app before development begins. This UI design mock-up was completed using OmniGraffle.*

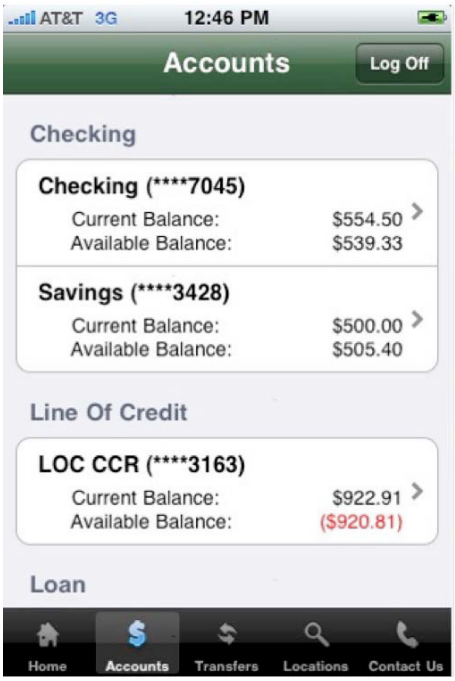

Figure 1–2. *This screenshot shows a completed iPhone mobile banking application as it appeared on the iTunes App Store. This app is called Woodforest Mobile Banking.*

Completing the Development Cycle

Now that we have our design requirements and user interface designs and have written our program, what's next? After programming, we need to make sure our program matches the design requirements and user interface design and that there are no errors. In programming vernacular, errors are called **bugs**. Bugs are undesired results of our programming and must be fixed before released to the App Store. The process of finding bugs in our programs and making sure the program meets the design requirements is called **testing**. Typically, someone who is experienced in software testing methodology and who didn't write the app performs this testing. Software testing is commonly referred to as **Quality Assurance (QA)**. Figure 1–3 shows the complete software development cycle.

> **NOTE:** When an application is ready to be submitted to the iTunes App Store, Xcode gives the file an `.app` extension, for example, `appName.app`. That is why iPhone, iPad, and Mac applications are called **apps**. We will use "program," "application," and "app" to mean the same thing throughout this book.

During the testing phase, the developer will need to work with QA staff to determine why the application is not working as designed. The process is called **debugging**. It requires

the developer to step through the program to find out why the application is not working as designed. Figure 1–3 shows the complete software development cycle.

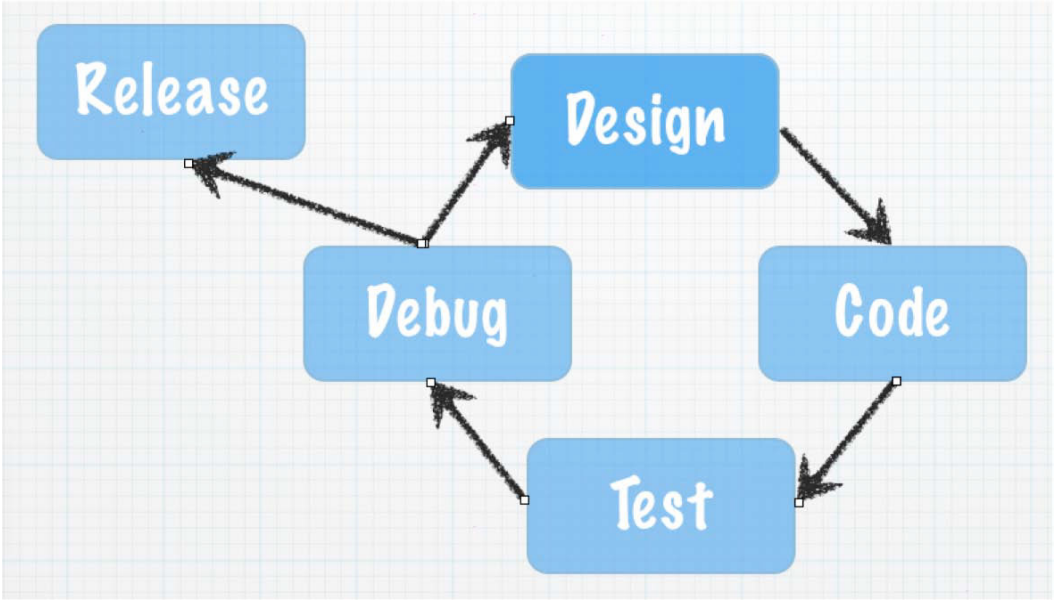

Figure 1–3. *The typical software development cycle*

Frequently during testing and debugging, changes to the requirements (design) need to occur to make the application more usable for the customer. Once the design requirements and user interface changes are made, the process begins over again.

At some point, the application that everyone has been working so hard on must be shipped to the iTunes App Store. Lots of considerations are taken into account when this happens:

- Cost of development
- Competition
- Stability of the application
- Return on investment

There is always the give-and-take between developers and management. Developers want the app perfect, and management wants to start realizing revenue from the investment as soon as possible. If the release were left up to the developers, the app would never ship to the App Store. Developers would continue to tweak the app forever, making it faster, more efficient, and more usable. At some point, however, the code needs to be pried from the developers' hands and shipped to the user, so it can do what it was meant to do.

Introducing Object Oriented Programming

As discussed in detail in the Introduction to this book, Alice enables us to focus on **object oriented programming (OOP)** without having to cover all the Objective-C programming syntax and complex Xcode development environment in one big step. Instead, we can focus on learning the basic principles of OOP and using those principles quickly to write our first programs.

For decades, developers have been trying to figure out a better way to develop code that was reusable, manageable, and easily maintained over the life of a project. OOP was designed to help achieve code reuse and maintainability while reducing the cost of software development.

OOP can be viewed as a collection of objects in a program. Actions are performed on these objects to accomplish the design requirements.

An **object** is anything that can be acted on. For example, an airplane, person, or screen/view on the iPad can all be objects. We may want to act on the plane by making the plane bank. We may want the person to walk or to change the color of the screen of an app on the iPad. Actions are all being applied to these objects; see Figure 1–4.

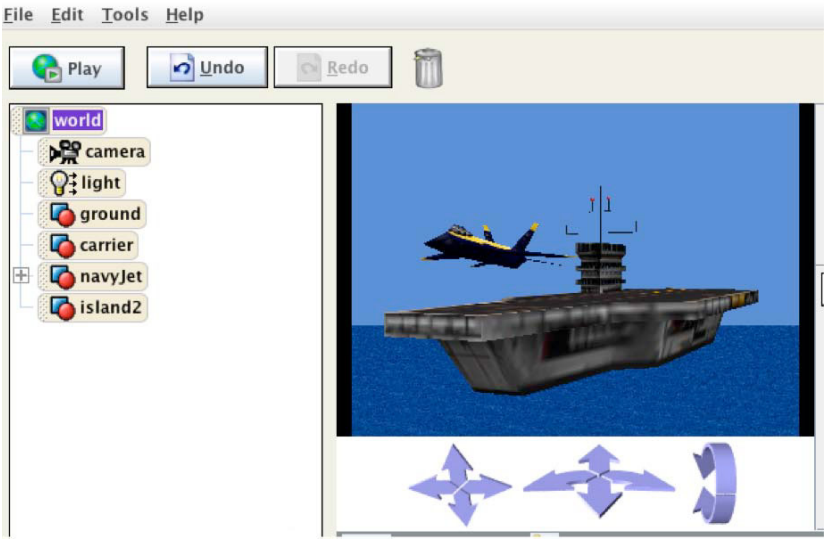

Figure 1–4. *These are two objects in an Alice application, a plane and aircraft carrier. Both objects can have actions applied—take off and landing for the plane and "turn to port" and "ahead flank" for the aircraft carrier.*

Like the play button in Alice, the Xcode **integrated development environment (IDE)** enables us to run our application from within our programming environment See Figure 1–5.

Figure 1–5. *This sample iPhone app contains UITableView objects. Actions such as "rotate left" or "user did select row 3" can be applied to this object.*

Actions that are performed on objects are called **methods**. Methods manipulate objects to accomplish what we want our app to do. For example, for our jet object in Figure 1–4, we might have the following methods:

```
goUp
goDown
bankLeft
turnOnAfterBurners
lowerLandingGear
```

For our iPhone application's `UITableView` object in Figure 1–5, we could have the following methods:

```
loadView
shouldAutorotateToInterfaceOrientation
numberOfSectionsInTableView
cellForRowAtIndexPath
didSelectRowAtIndexPath
```

All objects have data that describes those objects. Our properties hold values that describe the state of the objects. This data is defined as **properties**. Each property describes the associated object in a specific way. For example, the jet object's properties might be as follows:

```
altitude = 10,000 feet
heading = North
speed = 500 knots
```

```
pitch = 10 degrees
yaw = 20 degrees
latitude = 33.575776
longitude = -111.875766
```

For our `UITableView` object in Figure 1–5, these might be our properties:

```
backGroundColor = Red
selectedRow = 3
animateView = No
```

An object's properties can be changed at any time as our program is running, as the user interacts with the app, or as the programmer designs the app to accomplish the design requirements. The values stored in the properties of an object at a specific time are collectively called the **state of an object**.

State is an important concept in computer programming. When teaching students about state, Gary asks them to go over to a window and find an airplane in the sky. He then asks them to snap their figures and make up some of the values that the plane's property might have at that specific time. Those values might be

```
altitude = 10,000 feet
latitude = 33.575776
longitude = -111.875766
```

Those values represent the state of the object at the specific time that they snapped their fingers.

After waiting a couple of minutes, Gary asks the students to find that same plane, snap their fingers again, and record the plane's possible state at that specific point in time.

The values of the properties might then be something like this:

```
altitude = 10,500 feet
latitude = 33.575665
longitude = -111.875777
```

Notice how the state of the object changes over time.

Working with the Alice Interface

Alice offers a great approach in using the concepts that we have just discussed without all the complexity of learning Xcode and the Objective-C language at the same time. It just takes a few minutes to familiarize yourself with the Alice interface and begin writing a program.

The Introduction of this book describes how to download Alice. Once it's downloaded, you need to open Alice. See Figure 1–6

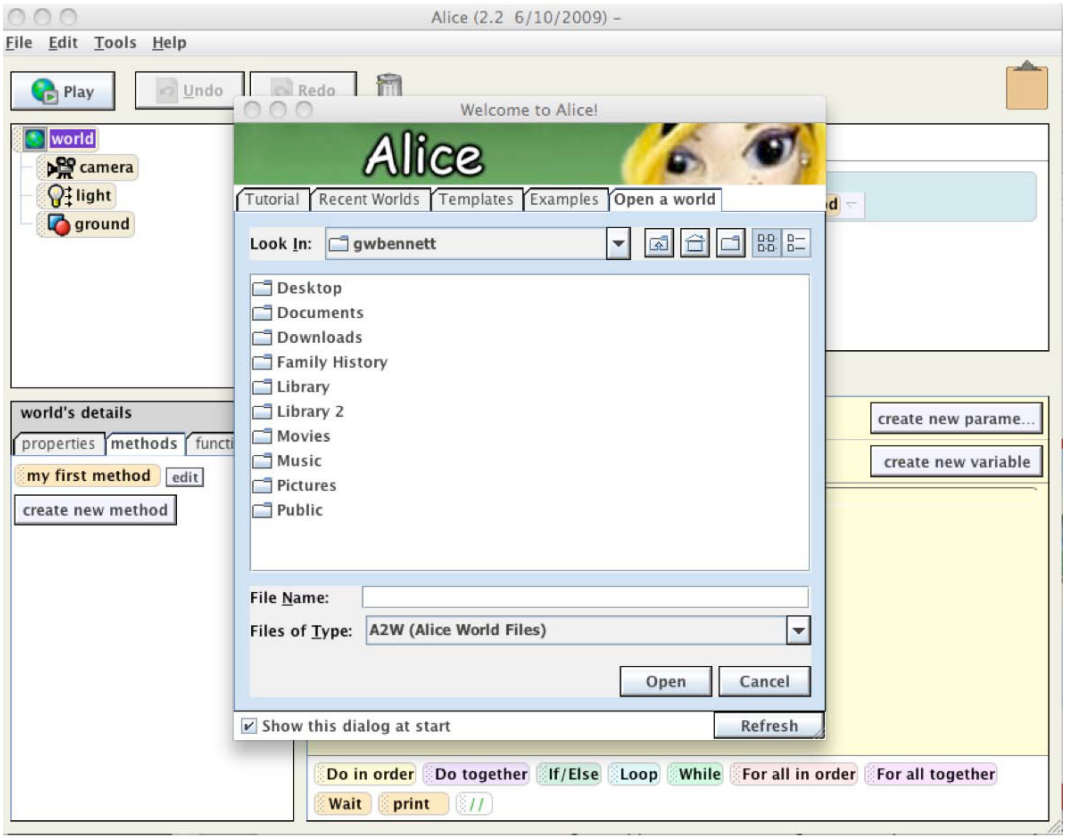

Figure 1–6. *It is worth your time to click on the Tutorial tab to familiarize yourself with the Alice application and user interface. Additionally, there are several great examples on the Examples tab.*

Alice has great tutorials and examples that are highly recommended for developers to work through, like the one shown in Figure 1–7.

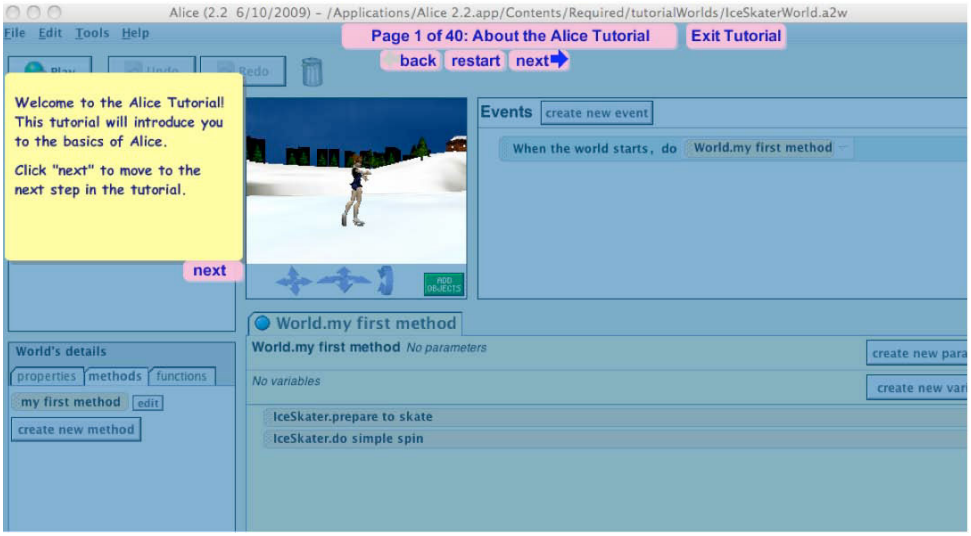

Figure 1–7. *An Alice tutorial describing the user interface*

Technically speaking, Alice is not a true IDE like Xcode, but it is pretty close and a whole lot easier to learn than Xcode. A true IDE combines code development, user interface layout, debugging tools, documentation and simulator/console launching for a single application; see Figure 1–8. However, Alice offers similar look, feel, and features to Xcode. This will serve you well later when we start writing Objective-C code.

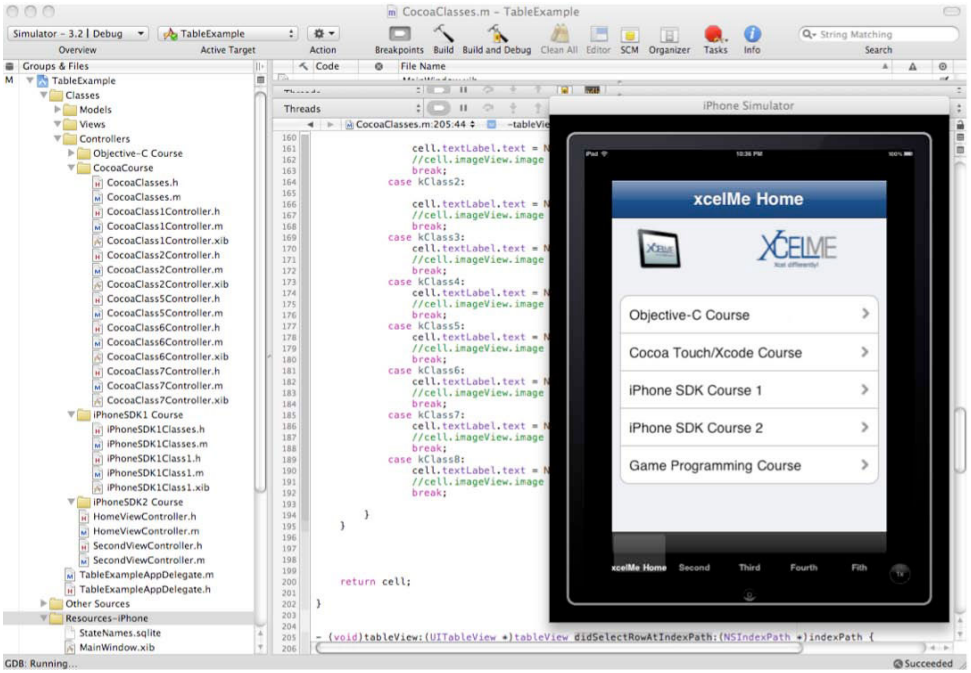

Figure 1–8. *The Xcode integrated development environment (IDE)) with the iPad Simulator*

In the next chapter, we will go through the Alice interface and write our first program.

Summary

Congratulations, you have finished the first chapter of this book. It is important that you have an understanding of the following terms, because they will be reinforced throughout this book:

- Computer program
- Algorithm
- Design requirements
- User interface
- Bug
- Quality assurance (QA)
- Debugging
- Object oriented programming (OOP)
- Object
- Property
- Method
- State of an object
- Integrated development environment (IDE)

Exercises

- Write an algorithm for how a soda machine works, from the time when a coin is inserted until a soda is dispensed. Assume the price of a soda is 80 cents.
- Write the design requirements for an app that will run the soda machine.

Chapter 2

Programming Basics

This chapter will focus on the building blocks necessary to become a great Objective-C programmer. We are going to use the Alice user interface, write our first Alice program, explore some new OOP terms and write our first Objective-C program.

NOTE: We want to introduce new concepts in Alice that later enable you to use these concepts in Objective-C. We believe this unique approach will help you learn the concepts quickly, without discouragement, and give you a great foundation to build on.

Taking a Tour with Alice

Alice's 3D programming environment makes it easy to write your first program using some of the principles that you learned about in Chapter 1. First, you need to learn a little more about Alice's user interface. When we first launch Alice, we are presented with a screen that looks like Figure 2–1.

You can start with the default blue sky and green grass template or pick another template with different backgrounds. Feel free to explore and have fun. This is where we will spend most of our time and write our first Alice application.

The Alice user interface is set up to help us efficiently write our applications. The user interface is very similar in form and function to the Xcode IDE. We will now explore the major sections of Alice.