Objective-C for Absolute Beginners

iPhone, iPad and Mac Programming Made Easy

Third Edition

Gary Bennett
Brad Lees
Mitchell Fisher

Apress®

Objective-C for Absolute Beginners

iPhone, iPad and Mac Programming Made Easy

Third Edition

Gary Bennett
Brad Lees
Mitchell Fisher

Apress®

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

Gary Bennett
Scottsdale, Arizona, USA

Brad Lees
Phoenix, Arizona, USA

Mitchell Fisher
Boston, Massachusetts, USA

ISBN-13 (pbk): 978-1-4842-1903-4
ISBN-13 (electronic): 978-1-4842-1904-1
DOI 10.1007/978-1-4842-1904-1

Library of Congress Control Number: 2016960200

Copyright © 2016 by Gary Bennett, Brad Lees and Mitchell Fisher

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr
Lead Editor: Aaron Black
Technical Reviewer: Stefan Kaczmarek
Editorial Board: Steve Anglin, Pramila Balan, Laura Berendson, Aaron Black, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Nikhil Karkal, James Markham, Susan McDermott, Matthew Moodie, Natalie Pao, Gwenan Spearing
Coordinating Editor: Jessica Vakili
Copy Editor: Kim Wimpsett
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 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`. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

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/bulk-sales`.

Any source code or other supplementary materials referenced by the author in this text are available to readers at `www.apress.com`. For detailed information about how to locate your book's source code, go to `www.apress.com/source-code/`. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

Printed on acid-free paper

Gary would like to dedicate this book to his wife, Stefanie, and children, Michael, Danielle, Michelle, and Emily, for always supporting him.

Mitch would like to dedicate this book to his family that supported him through all the long nights, especially his wife Heather, and his children, Jade, Eric, and Matthew

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 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 *Swift 3 for Absolute Beginners* for Apress. Gary lives in Scottsdale, Arizona, with his wife, Stefanie, and their four children.

Brad Lees has more than 12 years of 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.

Mitchell Fisher is a software developer in the Boston, Massachusetts area. He was introduced to PCs in the 1980s when 64 KB 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 has led several teams of developers on multimillion-dollar projects. Mitch now divides his time between writing iOS applications and server-side UNIX technologies.

About the Technical Reviewer

Stefan Kaczmarek has more than 15 years of software development experience specializing in mobile applications, large-scale software systems, project management, network protocols, encryption algorithms, and audio/video codecs. As chief software architect and cofounder of SKJM, LLC, Stefan has developed a number of successful mobile applications including iCam (which has been featured on CNN, *Good Morning America*, and *The Today Show*, and which was chosen by Apple to be featured in the "Dog Lover" iPhone 3GS television commercial) and iSpy Cameras (which held the #1 Paid iPhone App ranking in a number of countries around the world including the United Kingdom, Ireland, Italy, Sweden, and South Korea). Stefan resides in Phoenix, Arizona, with his wife, Veronica, and their two children.

Introduction

Over the past seven 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 was also written for people who may have never programmed before using object-oriented programming (OOP) languages. There are lots of Objective-C books, 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 seven years we have taught thousands 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" 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 beginning. In the past, students had to learn the following topics all at once:

- 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 you to focus on learning the principles of OOP without having to focus on learning a complex IDE and Objective-C principles all at once. You 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 you feel comfortable with the material, we then move into Xcode, where you will get to use your new OOP knowledge in writing Objective-C applications. This enables you 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 humbling, even for the most experienced 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. One of the most satisfying accomplishments you can have is seeing your 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. 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, in other words, a sample iPhone app, in a Table View. See Figure I-1.

Required Software, Materials, and Equipment

One of the great things about Alice is it is 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.

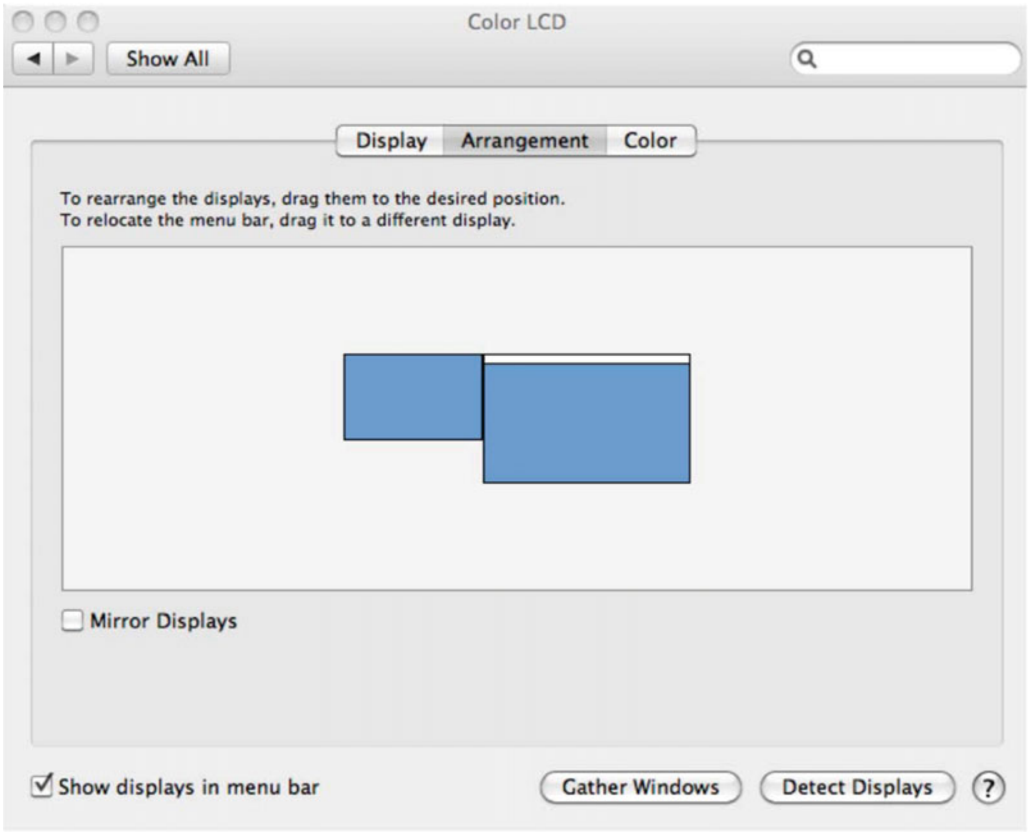
Operating System and IDE

Although you can use Alice on many platforms, the IDE that developers use to develop iPhone/iPad apps is Xcode. The IDE is free and is available in the Mac App Store.

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. 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.

To access the dual-monitor set up feature, go to Apple System Preferences and select Displays (see Figure I-1).



To rearrange the displays, drag them to the desired position.
To relocate the menu bar, drag it to a different display.

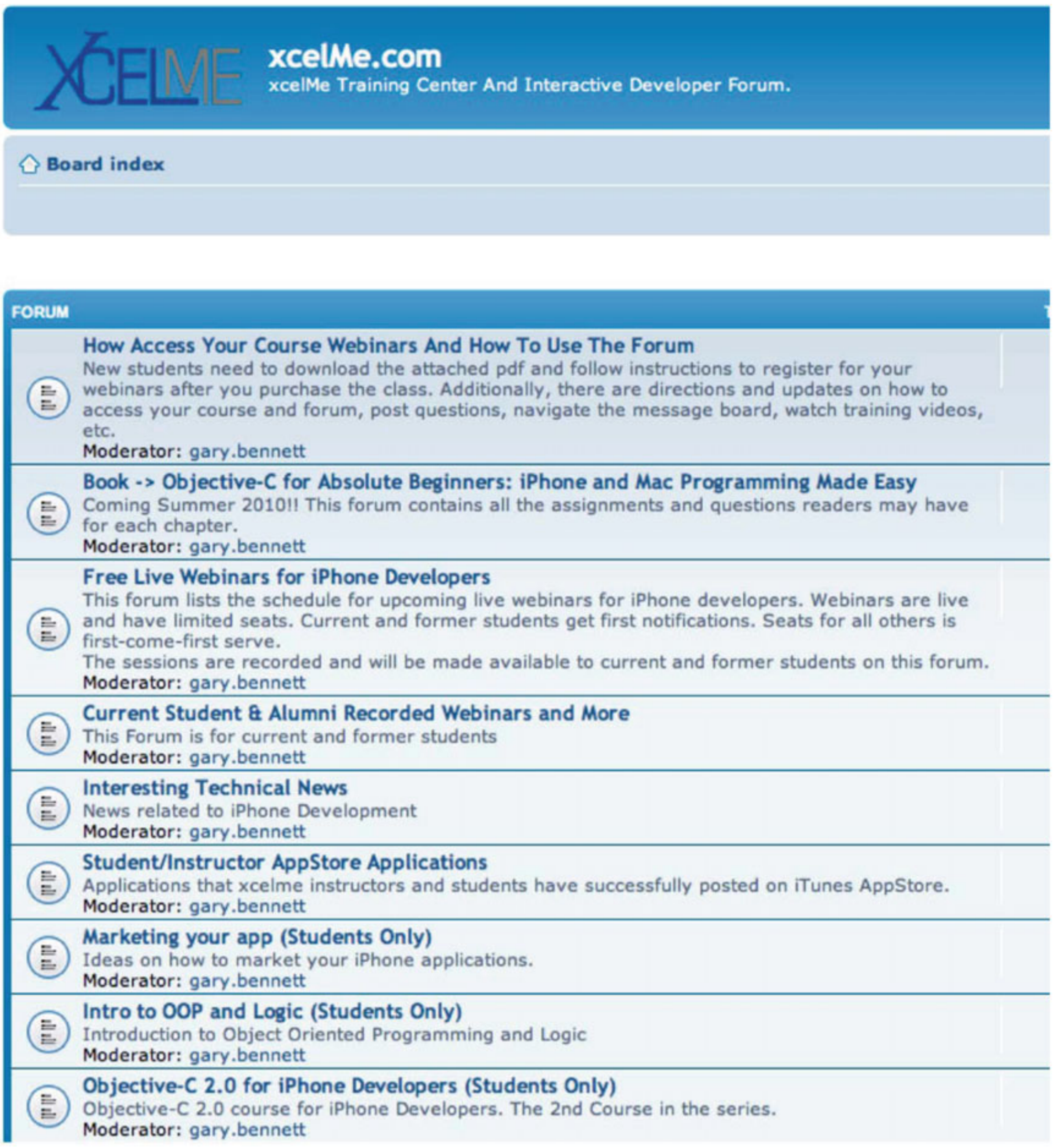
☐ Mirror Displays

☑ Show displays in menu bar

Figure I-1. *Dual monitors*

Book Forum

We have developed an online forum for this book at `http://forum.xcelme.com`, where you can ask us questions while you are learning Objective-C. See Figure I-2.


xcelMe.com
xcelMe Training Center And Interactive Developer Forum.

Board index

FORUM

How Access Your Course Webinars And How To Use The Forum
New students need to download the attached pdf and follow instructions to register for your webinars after you purchase the class. Additionally, there are directions and updates on how to access your course and forum, post questions, navigate the message board, watch training videos, etc.
Moderator: gary.bennett

Book -> Objective-C for Absolute Beginners: iPhone and Mac Programming Made Easy
Coming Summer 2010!! This forum contains all the assignments and questions readers may have for each chapter.
Moderator: gary.bennett

Free Live Webinars for iPhone Developers
This forum lists the schedule for upcoming live webinars for iPhone developers. Webinars are live and have limited seats. Current and former students get first notifications. Seats for all others is first-come-first serve.
The sessions are recorded and will be made available to current and former students on this forum.
Moderator: gary.bennett

Current Student & Alumni Recorded Webinars and More
This Forum is for current and former students
Moderator: gary.bennett

Interesting Technical News
News related to iPhone Development
Moderator: gary.bennett

Student/Instructor AppStore Applications
Applications that xcelme instructors and students have successfully posted on iTunes AppStore.
Moderator: gary.bennett

Marketing your app (Students Only)
Ideas on how to market your iPhone applications.
Moderator: gary.bennett

Intro to OOP and Logic (Students Only)
Introduction to Object Oriented Programming and Logic
Moderator: gary.bennett

Objective-C 2.0 for iPhone Developers (Students Only)
Objective-C 2.0 course for iPhone Developers. The 2nd Course in the series.
Moderator: gary.bennett

Figure I-2. *Reader forum for accessing answers to exercise and posting questions for authors*

CHAPTER 1

Becoming a Great iOS 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 several 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. Often, what you think you've told your program to do and what it actually does are two different things.

Key to Success If you haven't already, take a few minutes to read the introduction of this book. The introduction shows you where to go to access the free webinars, forums, and YouTube videos that go with each chapter. Also, you'll better understand why we are using the Alice programming environment and how to be successful in developing your iOS 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 want and the satisfaction associated with this.

Thinking like a Developer

Software development involves writing a computer program and then having a computer execute that program. A **computer program** is the set of instructions that you want the computer to perform. Before beginning to write a computer program, it is helpful to list the steps that you want your program to perform, in the order you want them accomplished. This step-by-step process is called an **algorithm**.

If you wanted to write a computer program to toast a piece of bread, you 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.

© Gary Bennett, Brad Lees and Mitchell Fisher 2016

G. Bennett et al., *Objective-C for Absolute Beginners*, DOI 10.1007/978-1-4842-1904-1_1

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, the algorithm leaves out many details and makes many assumptions. Here are some examples:

- What kind of toast does the user want? Does the user want white bread, wheat, or some other kind of bread?
- How does the user want the bread toasted? Light or dark?
- What does the user want on the bread after it is toasted: butter, margarine, honey, or strawberry jam?
- Does this algorithm work for all users in their cultures and languages? Some cultures may have another word for toast or not know what toast is.

Now, you might be thinking we are getting too detailed for just making a simple toast program. Over the years, software development has gained a 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.

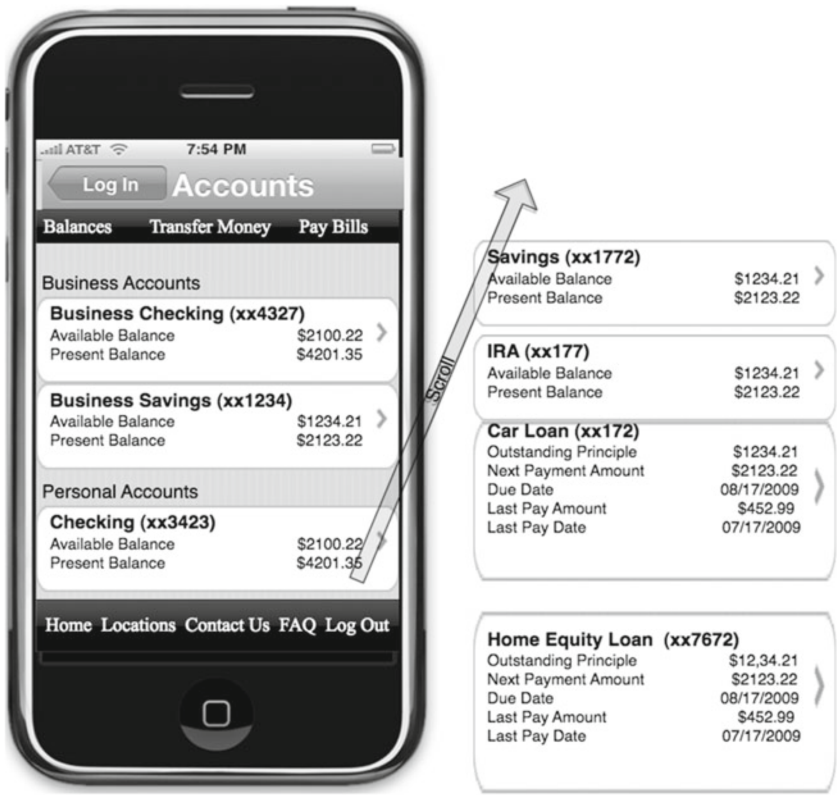
The key ingredients to making successful applications are **design requirements**. Design requirements can be formal and detailed or as simple as a list on a piece of paper. Design requirements are important because they help the developer flesh out what the application should do and not do when complete. Design requirements should not be completed in a programmer's vacuum but should be produced as the result of 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 least expensive) 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 you spend more than 50 percent of the entire development process focusing on the UI design. The design can be done using simple pencil and paper or using Xcode's storyboard feature to lay out your screen elements. 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 the design requirements from their screen layouts.

After 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 shows a mock-up of a mobile banking app screen prior to development using OmniGraffle. Developing mock-up screens along with design requirements forces developers to think through many of the applications usability issues before coding begins. This enables the application development time to be shortened and makes for a better user experience and better reviews on the App Store. Figure 1-2 shows how the view for the mobile banking app actually appears when completed.



Business Accounts

Business Checking (xx4327)
Available Balance $2100.22
Present Balance $4201.35

Business Savings (xx1234)
Available Balance $1234.21
Present Balance $2123.22

Personal Accounts

Checking (xx3423)
Available Balance $2100.22
Present Balance $4201.35

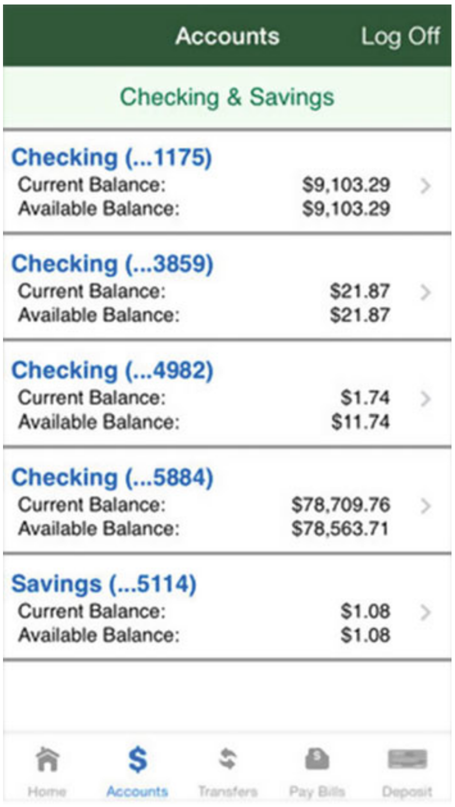
Savings (xx1772)
Available Balance $1234.21
Present Balance $2123.22

IRA (xx177)
Available Balance $1234.21
Present Balance $2123.22

Car Loan (xx172)
Outstanding Principle $1234.21
Next Payment Amount $2123.22
Due Date 08/17/2009
Last Pay Amount $452.99
Last Pay Date 07/17/2009

Home Equity Loan (xx7672)
Outstanding Principle $12,34.21
Next Payment Amount $2123.22
Due Date 08/17/2009
Last Pay Amount $452.99
Last Pay Date 07/17/2009

Figure 1-1. *This is a UI mock-up of the account balance screen for an iPhone mobile banking app before development begins on the original iPhone in 2010. This UI design mock-up was completed using OmniGraffle*



Checking (...1175)
Current Balance: $9,103.29
Available Balance: $9,103.29

Checking (...3859)
Current Balance: $21.87
Available Balance: $21.87

Checking (...4982)
Current Balance: $1.74
Available Balance: $11.74

Checking (...5884)
Current Balance: $78,709.76
Available Balance: $78,563.71

Savings (...5114)
Current Balance: $1.08
Available Balance: $1.08

Figure 1-2. *This is a completed iPhone mobile banking application as it appeared on the App Store after several revisions in 2015. This app is called Woodforest Mobile Banking*

Completing the Development Cycle

Now that you have your design requirements and user interface designs and have written your program, what's next? After programming, you need to make sure your program matches the design requirements and user interface design and ensure that there are no errors. In programming vernacular, errors are called **bugs**. Bugs are undesired results of our programming and must be fixed before the app is released to the App Store. The process of finding bugs in 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)**.

■ **Note** When an application is ready to be submitted to the App Store, Xcode gives the file an `.app` or `.ipa` extension, for example, `appName.app`. That is why iPhone, iPad, and Mac applications are called **apps**. This book uses **program**, **application**, and **app** to mean the same thing.

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.

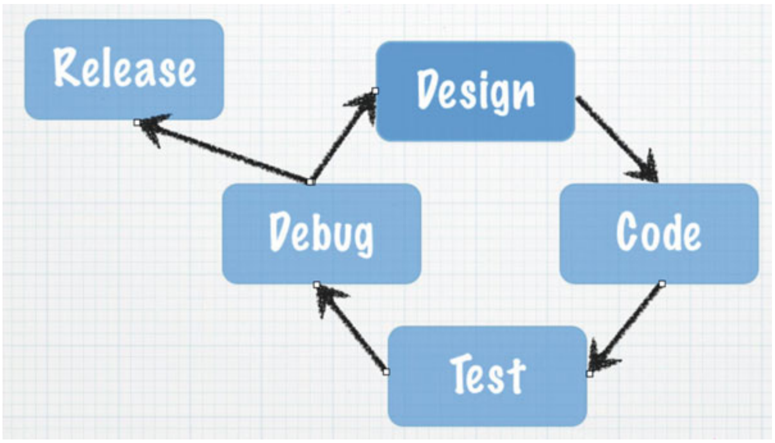


Figure 1-3. *The typical software development cycle*

Frequently during testing and debugging, changes to the requirements (design) must occur to make the application more usable for the customer. After 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 App Store. Many considerations are taken into account when this happens.

- Cost of development
- Budget
- 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 date were left up to the developers, the app would likely 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 uploaded to the App Store so it can do what it was meant to do.

Introducing Object-Oriented Programming

As discussed in detail in the introduction, Alice enables you 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, you can focus on learning the basic principles of OOP and using those principles quickly to write your first programs.

For decades, developers have been trying to figure out a better way to develop code that is 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. You may want to act on the plane by making the plane bank. You 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.

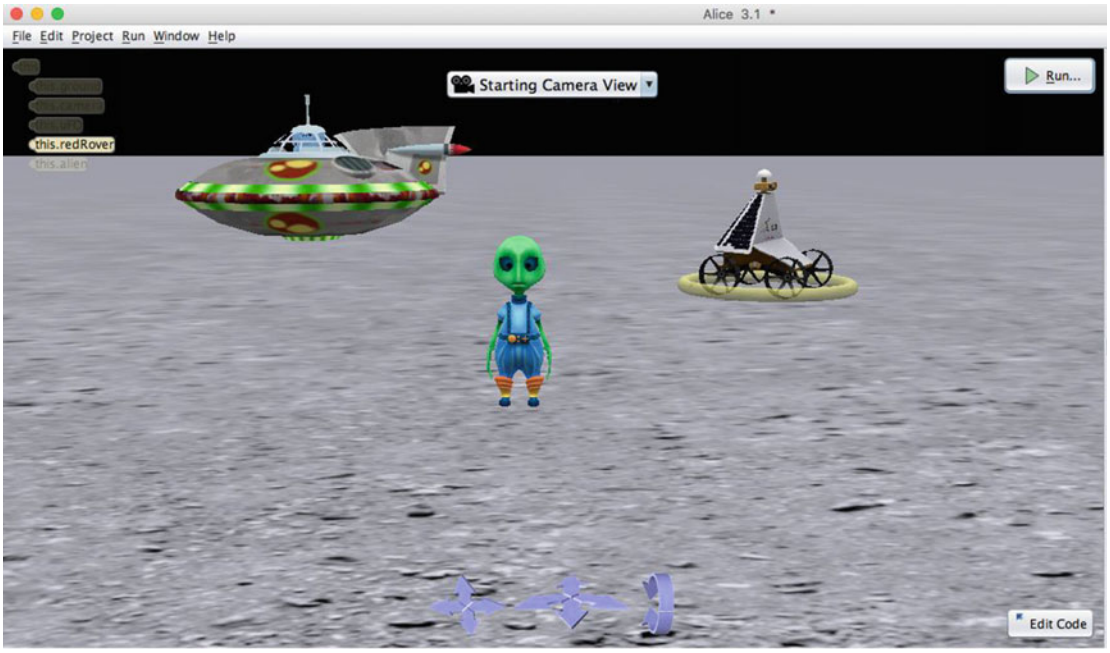


Figure 1-4. *There are three objects in this Alice application: UFO, Rover, and Alien. The UFO object can have actions applied—takeoff and landing, turn right and turn left*

Alice will run a program, such as the one shown in Figure 1-4, for you if you click the play button. When you run your Alice applications, the user can apply actions to the objects in your application. Similarly, Xcode is an **integrated development environment (IDE)** that enables you to run your application from within your programming environment. You can test your applications on your computers first before running them on your iOS devices by running the apps in Xcode's iPhone simulator, as shown in Figure 1-5.

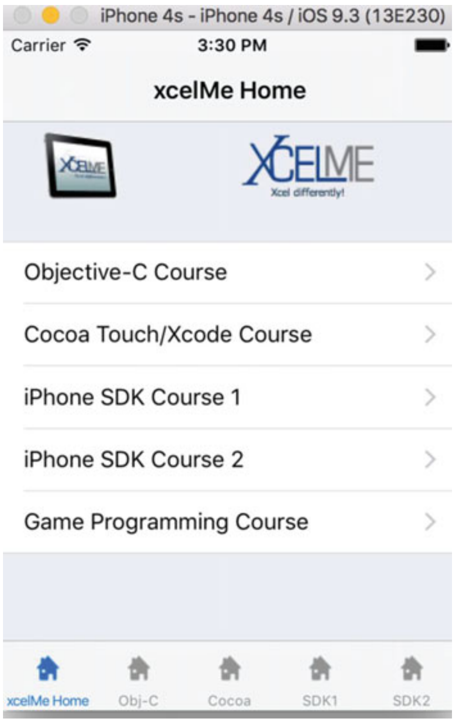


Figure 1-5. *This sample iPhone app contains a table object to organize a list of courses. Actions such as "rotate left" or "user did select row 3" can be applied to this view object*

Actions that are performed on objects are called **methods**. Methods manipulate objects to accomplish what you want your app to do. For example, for a jet object, you might have the following methods:

```
goUp
goDown
bankLeft
turnOnAfterburners
lowerLandingGear
```

The table object in Figure 1-5 is actually called `UITableView` when you use it in a program, and it could have the following methods:

```
numberOfRowsInSection
cellForRowAtIndexPath
canEditRowAtIndexPath
commitEditingStyle
didSelectRowAtIndexPath
```

Most objects have data that describes those 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 the `UITableView` object in Figure 1-5, the following might be the properties:

`backGroundColor` = Red

`selectedRow` = 3

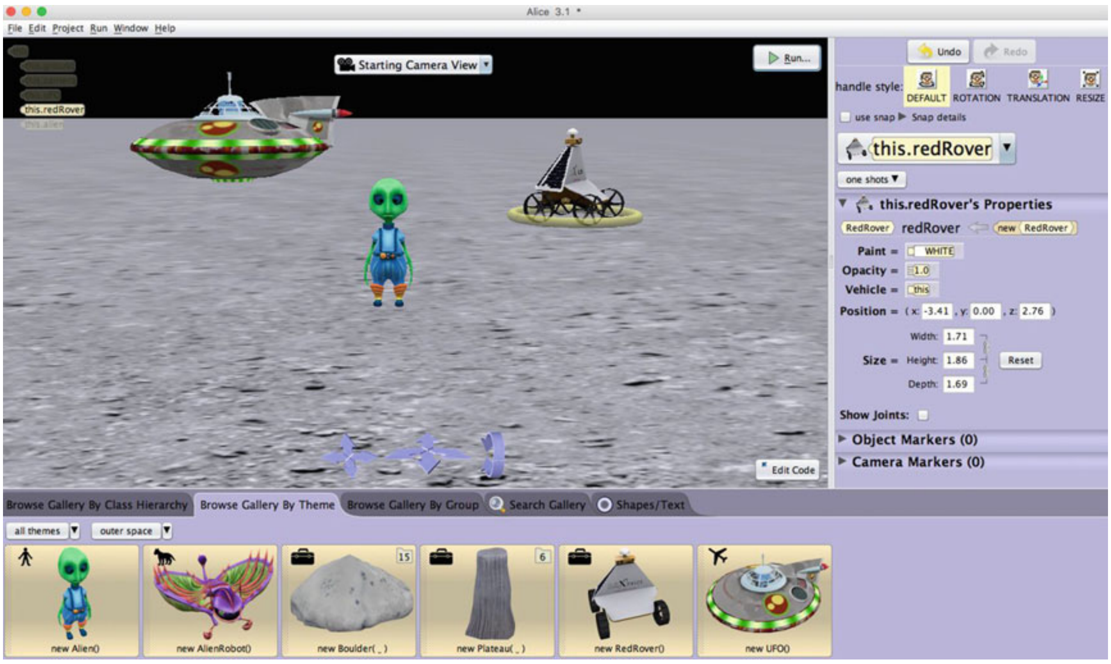
`animateView` = No

An object's properties can be changed at any time when your program is running, when the user interacts with the app, or when 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.

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 takes only a few minutes to familiarize yourself with the Alice interface and begin writing a program.

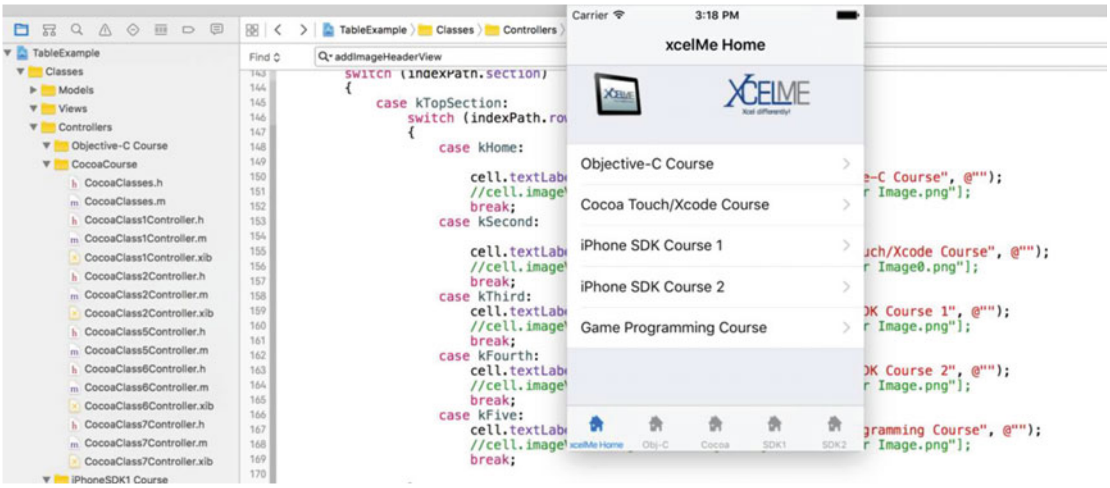
The introduction of this book describes how to download Alice. After it's downloaded and installed, you need to open Alice. It will look like Figure 1-6.



this.redRover
this.redRover's Properties
RedRover redRover ⇐ new RedRover()
Paint = WHITE
Opacity = 1.0
Vehicle = this
Position = (x: -3.41 , y: 0.00 , z: 2.76)
Size = Width: 1.71 Height: 1.86 Depth: 1.69 Reset
Show Joints:

Figure 1-6. *Alice IDE running*

Technically speaking, Alice is not a true IDE like Xcode, but it is pretty close and much 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 1-7. However, Alice offers a similar look, feel, and features to Xcode. This will serve you well later when you start writing Objective-C code.



TableExample
Classes
Models
Views
Controllers
Objective-C Course
CocoaCourse
CocoaClasses.h
CocoaClasses.m
CocoaClass1Controller.h
CocoaClass1Controller.m
CocoaClass1Controller.xib
CocoaClass2Controller.h
CocoaClass2Controller.m
CocoaClass2Controller.xib
CocoaClass5Controller.h
CocoaClass5Controller.m
CocoaClass6Controller.h
CocoaClass6Controller.m
CocoaClass6Controller.xib
CocoaClass7Controller.h
CocoaClass7Controller.m
CocoaClass7Controller.xib
iPhoneSDK1 Course

```
    switch (indexPath.section)
    {
        case kTopSection:
            switch (indexPath.ro
            {
                case kHome:

                    cell.textLab                              e-C Course", @"");
                    //cell.image'                             r Image.png"];
                    break;
                case kSecond:

                    cell.textLab                              uch/Xcode Course", @"");
                    //cell.image'                             r Image0.png"];
                    break;
                case kThird:
                    cell.textLab                              DK Course 1", @"");
                    //cell.image'                             r Image.png"];
                    break;
                case kFourth:
                    cell.textLab                              DK Course 2", @"");
                    //cell.image'                             r Image.png"];
                    break;
                case kFive:
                    cell.textLab                              gramming Course", @"");
                    //cell.image'                             r Image.png"];
                    break;
```

Objective-C Course
Cocoa Touch/Xcode Course
iPhone SDK Course 1
iPhone SDK Course 2
Game Programming Course

Figure 1-7. *The Xcode IDE with the iPhone simulator*

In the next chapter, you will go through the Alice interface and write your 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

- Answer the following questions:
    - Why is it so important to spend time on your user requirements?
    - What is the difference between design requirements and an algorithm?
    - What is the difference between a method and a property?
    - What is a bug?
    - What is state?
- Write an algorithm for how a soda machine works from the time 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.