



Objective-C for Absolute Beginners

iPhone, iPad and
Mac Programming Made Easy

Stefan Kaczmarek
Brad Lees
Gary Bennett
Mitch Fisher

Apress®

Objective-C for Absolute Beginners

**iPhone, iPad and Mac
Programming Made Easy**

Stefan Kaczmarek

Brad Lees

Gary Bennett

Mitch Fisher

Apress®

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

Stefan Kaczmarek
Phoenix, Arizona, USA

Brad Lees
Phoenix, Arizona, USA

Gary Bennett
Scottsdale, Arizona, USA

Mitch Fisher
Glendale, Arizona, USA

ISBN-13 (pbk): 978-1-4842-3428-0
<https://doi.org/10.1007/978-1-4842-3429-7>

ISBN-13 (electronic): 978-1-4842-3429-7

Library of Congress Control Number: 2018937904

Copyright © 2018 by Stefan Kaczmarek, Brad Lees, Gary Bennett, Mitch 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, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Cover designed by eStudioCalamar

Cover image by Freepik (www.freepik.com)

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/rights-permissions.

Apress titles 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 Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-3428-0. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Authors..... xi

Introduction xiii

Chapter 1: Becoming a Great Objective-C Developer..... 1

 Thinking Like a Developer..... 2

 Completing the Development Cycle 5

 Introducing Object-Oriented Programming..... 7

 Working with the Alice Interface..... 11

 Summary..... 12

 Exercises..... 13

Chapter 2: Programming Basics..... 15

 Taking a Tour with Alice 15

 Application Menu..... 17

 Editing a Scene..... 18

 Classes, Objects, and Instances in Alice..... 20

 Object Tree..... 21

 Editor 22

 Methods Panel..... 23

 Creating an Alice App: To the Moon, Alice 23

 Your First Objective-C Program..... 32

 Launching and Using Xcode 32

 Summary..... 41

 Exercises..... 42

TABLE OF CONTENTS

Chapter 3: It's All About the Data 43

 Numbering Systems Used in Programming 43

 Bits 43

 Bytes 46

 Hexadecimal 47

 Unicode 49

 Data Types 49

 Using Variable and Data Types with Alice 50

 Data Types and Objective-C 64

 Identifying Problems 71

 Summary 72

 Exercises 73

Chapter 4: Making Decisions About and Planning Program Flow 75

 Boolean Logic 76

 Truth Tables 77

 Comparison Operators 80

 Designing Apps 80

 Pseudocode 81

 Flowcharting 84

 Designing and Flowcharting an Example App 85

 The App's Design 87

 Using Loops to Repeat Program Statements 88

 Coding the Example App in Alice 91

 Coding the Example App in Objective-C 92

 Nested If Statements and Else-If Statements 97

 Removing Extra Characters 97

 Improving the Code Through Refactoring 98

 Running the App 98

 Moving Forward Without Alice 99

 Summary 99

 Exercises 100

Chapter 5: Object-Oriented Programming with Objective-C	101
The Object.....	102
What Is a Class?.....	103
Planning Classes.....	104
Planning Properties	104
Planning Methods.....	107
Implementing the Classes	109
Inheritance	116
Why Use OOP?.....	117
It Is Everywhere.....	118
Eliminate Redundant Code	118
Ease of Debugging.....	118
Ease of Replacement.....	118
Advanced Topics	119
Interface	119
Polymorphism.....	119
Summary.....	120
Exercises.....	120
Chapter 6: Learning Objective-C and Xcode	121
A Brief History of Objective-C	121
Understanding the Language Symbols and Basic Syntax.....	122
Create a Variable	123
Begin and End a Section of Code	124
Signify the End of a Line of Code.....	124
Write a Comment.....	125
Define a Class.....	126
Define a Method	127
Define an Objective-C Variable	127

TABLE OF CONTENTS

Putting the “Objective” into Objective-C.....	128
Writing Another Program in Xcode.....	133
Creating the Project.....	133
Summary.....	155
Exercises.....	155
Chapter 7: Objective-C Classes, Objects, and Methods	157
Creating an Objective-C Class.....	157
Declaring Interfaces and Properties.....	159
Calling Methods.....	160
Working with the Implementation File.....	163
Coding Your Methods.....	165
Using Your New Class	167
Updating MyFirstApp	168
Adding Objects	169
Writing the Implementation File	173
Updating the User Interface.....	174
Hooking Up the Code	177
Accessing the Xcode Documentation.....	186
Summary.....	188
Exercises.....	188
Chapter 8: Programming Basics in Objective-C	191
Collections	191
Using NSSet.....	192
Using NSArray	194
NSDictionary.....	196
Using the Mutable Container Classes	197
NSMutableSet.....	198
NSMutableArray	199
NSMutableDictionary.....	200

Creating the Bookstore Application.....	201
Introducing Properties	208
Accessing Properties.....	209
Custom Getter and Setter	211
Finishing the MyBookstore Program.....	212
Creating the Initial View.....	218
The Bookstore Object	224
Using the Bookstore Object	227
Preparing the Table View	228
The Book Detail View	231
Setting Up the Outlets.....	246
Plugging in the Book Details	250
Summary.....	253
Exercises.....	254
Chapter 9: Comparing Data	255
Revisiting Boolean Logic.....	255
Using Relational Operators	256
Comparing Numbers.....	257
Creating an Example Xcode App.....	259
Using Boolean Expressions.....	263
Comparing Strings.....	265
Comparing Dates	267
Combining Comparisons.....	270
Using the switch Statement.....	271
Summary.....	273
Exercises.....	274

TABLE OF CONTENTS

Chapter 10: Creating User Interfaces 275

 Understanding Interface Builder 276

 The Model-View-Controller 277

 Human Interface Guidelines..... 279

 Creating an Example iPhone App with Interface Builder..... 281

 Using Interface Builder 287

 The Document Outline 288

 The Object Library 289

 Creating the View 290

 Using Outlets 291

 Connecting Actions and Objects 294

 Implementation File..... 294

 Summary..... 296

 Exercises..... 296

Chapter 11: Storing Information..... 297

 Storage Considerations..... 297

 Preferences..... 298

 Writing Preferences..... 298

 Reading Preferences 300

 Databases 301

 Storing Information in a Database 301

 Getting Started with Core Data 303

 The Model 305

 Managed Object Context 316

 Setting Up the Interface..... 317

 Summary..... 331

 Exercises..... 331

Chapter 12: Protocols and Delegates	333
Multiple Inheritance	333
Understanding Protocols.....	334
Protocol Syntax.....	335
Understanding Delegates.....	336
Next Steps.....	337
Summary.....	337
Exercise	337
Chapter 13: Introducing the Xcode Debugger.....	339
Getting Started with Debugging.....	340
Setting Breakpoints	341
Using the Breakpoint Navigator	344
Debugging Basics.....	346
Working with the Debugger Controls.....	348
Using the Step Controls	350
Looking at the Thread Window and Call Stack	352
Debugging Variables.....	353
Dealing with Code Errors and Warnings	355
Warnings	356
Summary.....	358
Exercise	358
Index.....	359

About the Authors



Stefan Kaczmarek has 20 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 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.



Brad Lees has more than a decade of experience in application development and server management. He 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 AUTHORS



Gary Bennett 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 App 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, chief information officer, and resident. 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.

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 iOS 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 those 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 nine years we have taught thousands of students at xcelMe.com to be iOS 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 many developers who programmed years ago or programmed in a non-OOP language and need some 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 for new developers to create rich graphical applications. 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. They get to focus on each topic individually. This helps the students feel a real sense of accomplishment as they progress.

Alice removes all of the complexity of learning an IDE and programming language syntax. It is drag-and-drop programming. You'll see that 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 to write 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 you'll follow step-by-step examples to reinforce it all.

Often we will repeat previous topics to reinforce what you have learned and apply these skills in new ways. This enables new programmers to reapply 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 must 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 now. 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. One of the most satisfying accomplishments you can have is seeing your app on the iOS 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.

INTRODUCTION

- 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!

Required Software, Materials, and Equipment

One of the great things about Alice is that it's 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 <http://www.alice.org/>.

Operating System and IDE

Although you can use Alice on many platforms, the IDE that developers use to develop iOS apps is Xcode, which is free and is available from 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 iOS simulator at the same time on dual, independent monitors. Apple hardware makes this easy. Note that 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.

1. To access the dual-monitor set-up feature, go to **Apple System Preferences** and select **Displays**, as shown in Figure I-1.

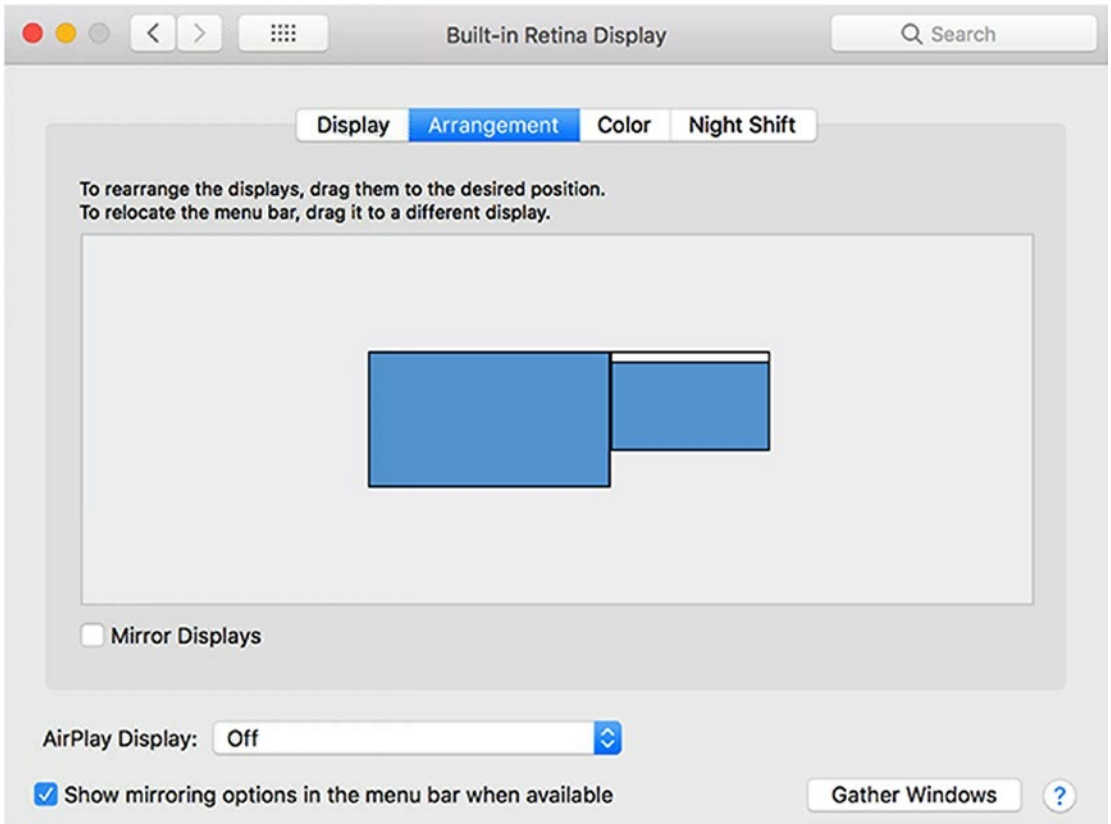



Figure I-1. Dual monitors

Book Forum

We 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. See Figure I-2.



xcelMe.com
xcelMe Training Center And Interactive Developer Forum.

Board index

[FAQ](#) [Members](#) [Register](#) [Login](#)

It is currently Sat Jan 27, 2018 9:18 am













FORUM	TOPICS	POSTS	LAST POST
 How To 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	3	12	by zenith9356  Thu Mar 13, 2014 10:24 am
 Book -> Swift 3.0 for Absolute Beginners: iPhone and Mac Programming Made Easy 3rd Edition This forum contains answers readers may have for each chapter as well as any corrections to the book. The forum also contains the Source Code for the book. Moderator: gary.bennett	17	22	by schurms  Wed Jan 17, 2018 6:04 pm
 Book -> Swift 2.0 for Absolute Beginners: iPhone and Mac Programming Made Easy 2nd Edition This forum contains answers readers may have for each chapter as well as any corrections to the book. The forum also contains the Source Code for the book. Moderator: gary.bennett	17	96	by zany76  Thu Aug 31, 2017 3:11 pm
 Book -> Developing for Apple TV using tvOS and Swift This forum contains answers readers may have for each chapter as well as any corrections to the book. The forum also contains the Source Code for the book. Moderator: gary.bennett	10	12	by mdstebel  Mon Jun 13, 2016 11:26 am
 Book -> Objective-C for Absolute Beginners: (2nd Edition) iPhone and Mac Programming Made Easy This forum contains all the assignments and questions readers may have for each chapter. Moderator: gary.bennett	20	224	by Drago  Mon Jun 16, 2014 9:27 pm
 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	1	9	by Miptinguaw  Tue Nov 29, 2011 3:48 am

Figure I-2. The Reader Forum for accessing answers to exercises and posting questions for authors

CHAPTER 1

Becoming a Great Objective-C Developer

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 apps in Objective-C.

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 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 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 want to write a computer program to toast a piece of bread, you first write an algorithm. This algorithm might look something like the following:

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 the 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, medium, 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 the **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 and should 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.

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.

Note If you take anything away from this chapter, let it be 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.

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 are easily accommodated by the development process. See Figures 1-1 and 1-2.

Figure 1-1 shows a mock-up of a rental report app screen prior to development. Developing mock-up screens along with design requirements forces developers to think through many of the application's usability issues before coding begins. This shortens the application development time and makes for a better user experience and better reviews on the App Store. Figure 1-2 shows how the view for the rental report app appears when completed. Notice how mock-up tools enable you to model the app to the real thing.

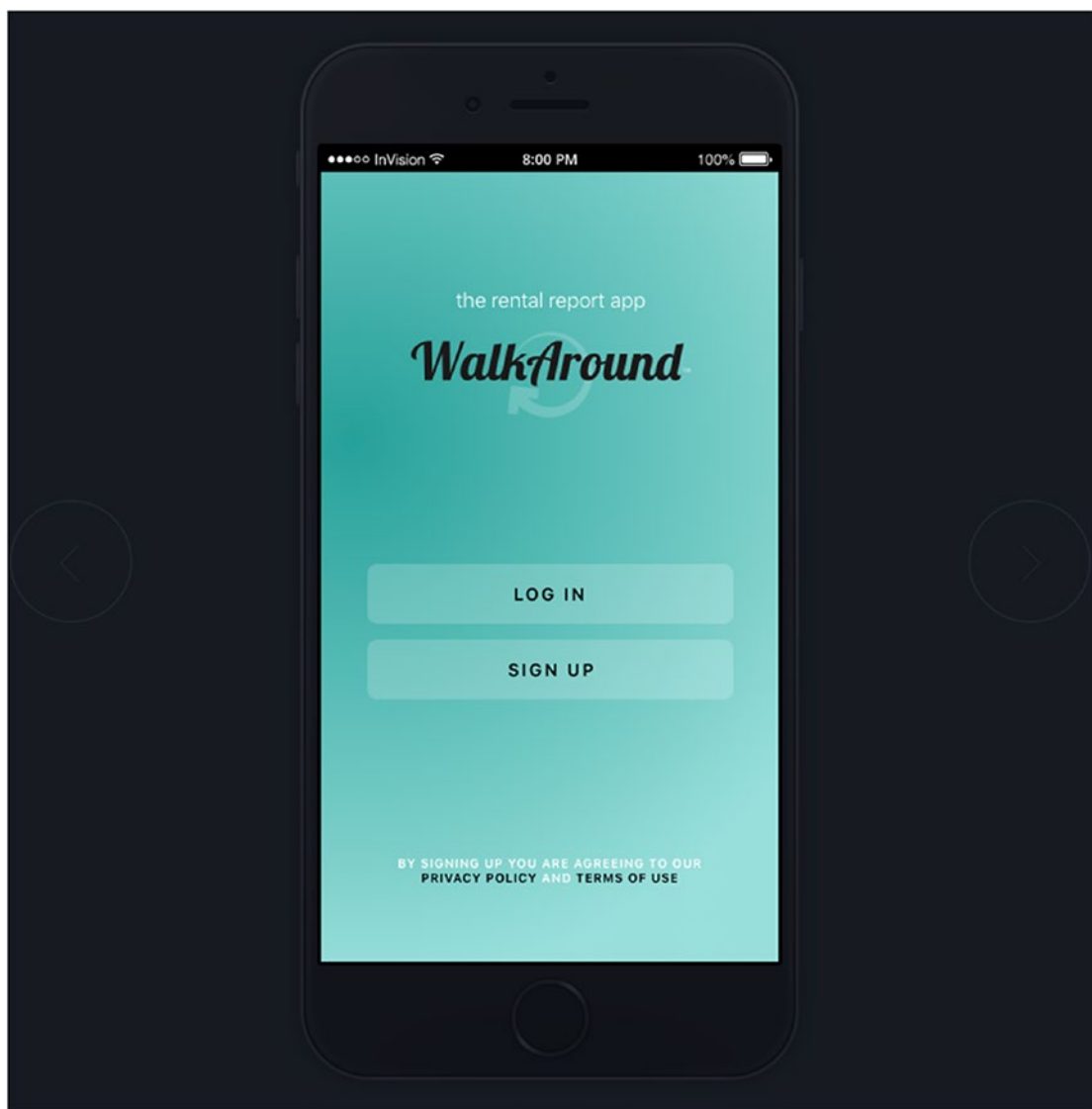


Figure 1-1. This is a UI mock-up of the Log In screen for an iPhone mobile rental report app before development begins. This UI design mock-up was completed using InVision.

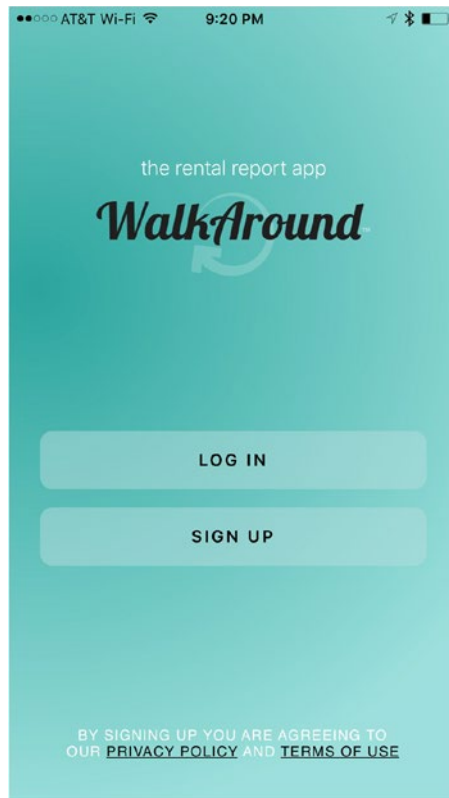


Figure 1-2. This is the completed iPhone rental report app. This app is called *WalkAround*.

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 your programming and must be fixed before the app is released. 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, such as `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.

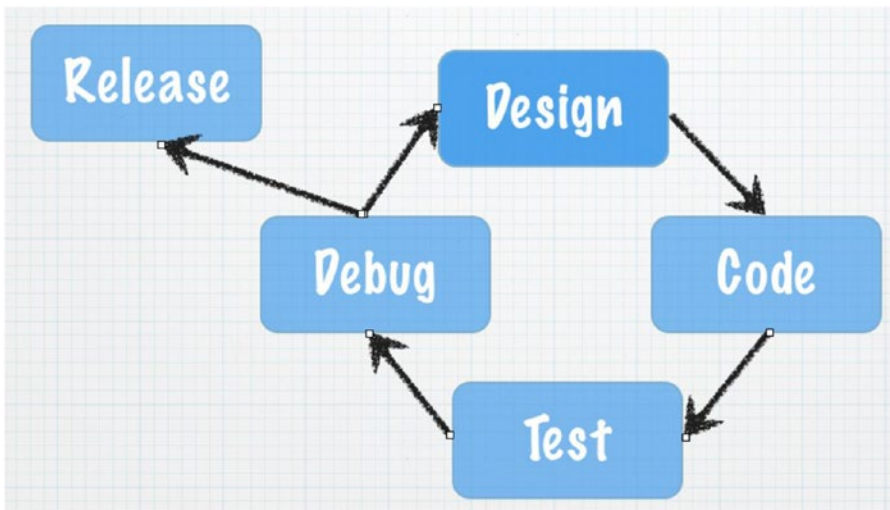


Figure 1-3. *The typical software development cycle*

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 released. 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. 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 released to the end users.

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 an 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 screen color within an iPad app. 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, 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 Run button. When you run your Alice applications, you 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 iOS simulator, as shown in Figure 1-5.

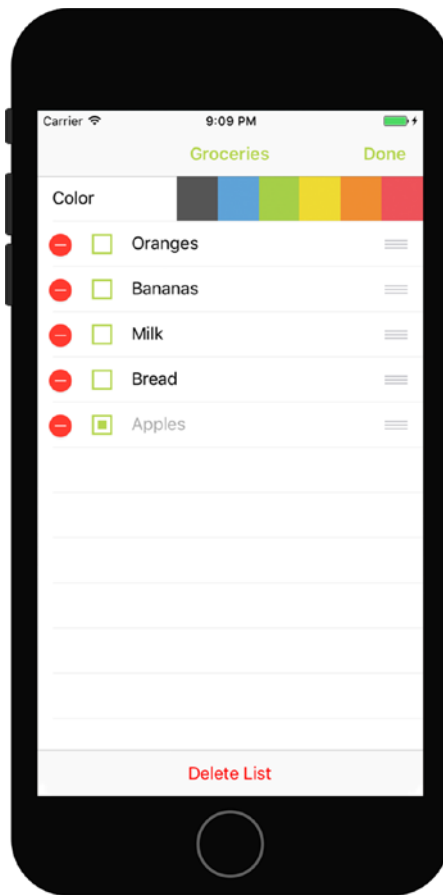


Figure 1-5. This sample iPhone app running in the iOS Simulator contains a table object to organize a list of groceries. Actions such as “rotate left” or “user selected row 3” can be applied to this view object.

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 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.

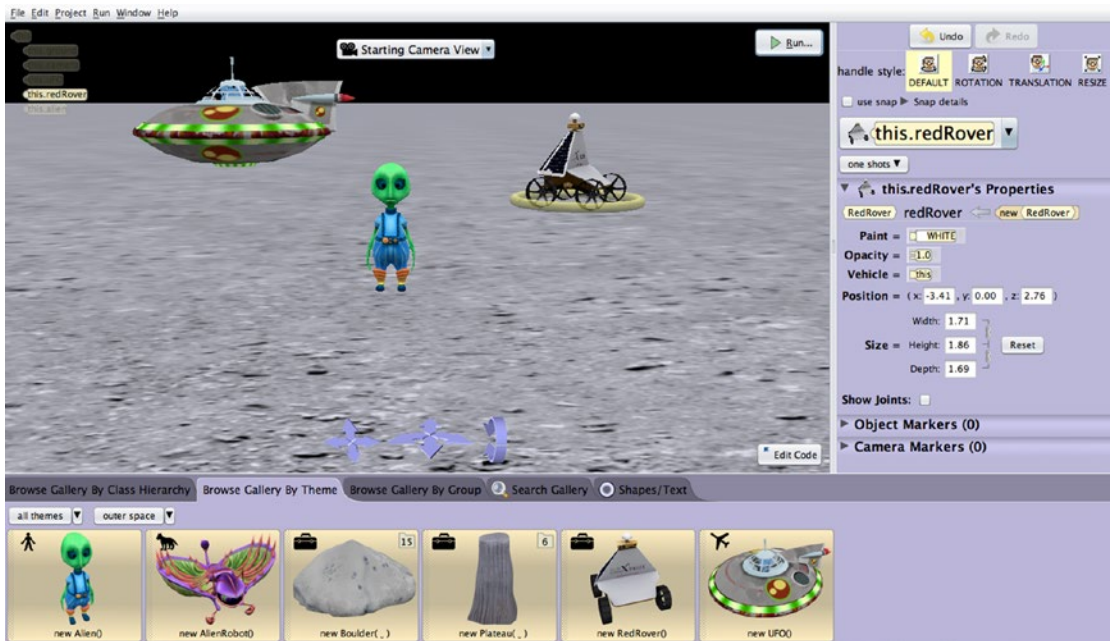


Figure 1-6. Alice IDE running

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.

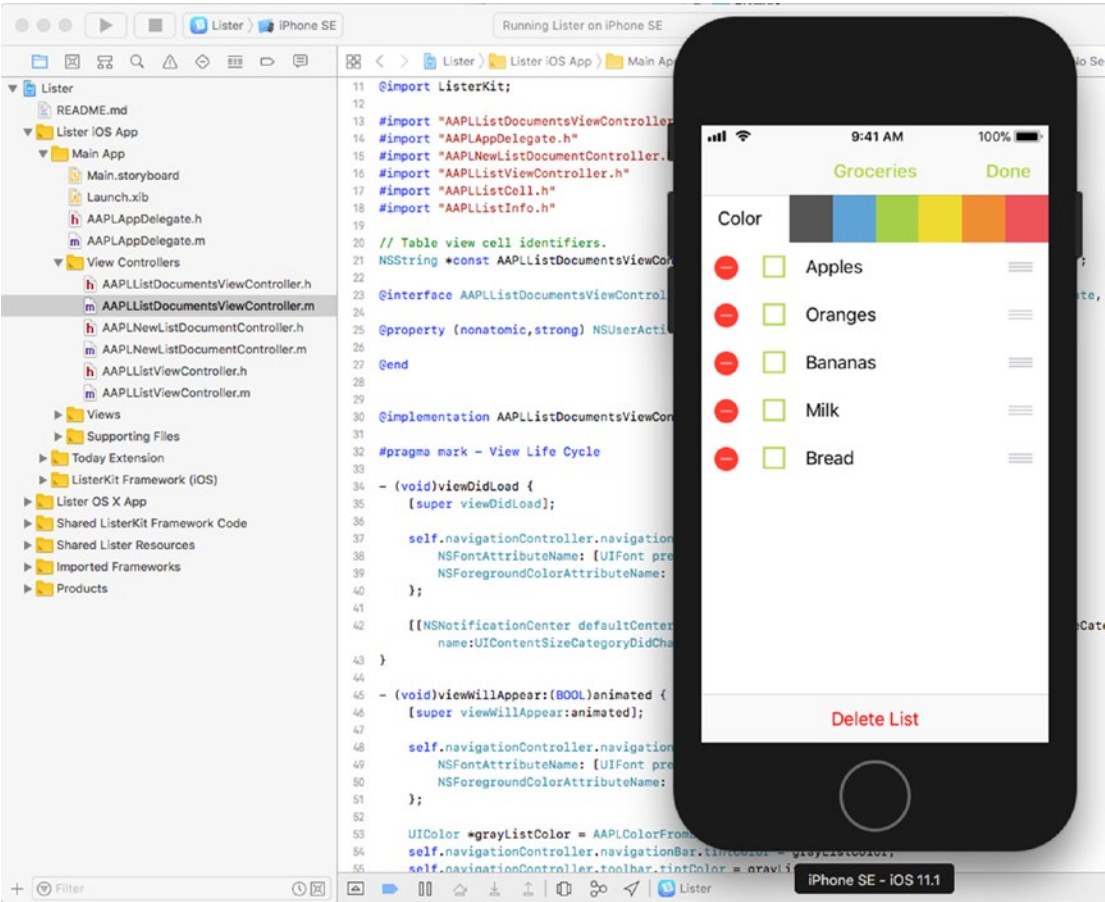


Figure 1-7. The Xcode IDE with the iPhone simulator

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