

Enhance your virtual world with the  
power of iOS augmented reality



Pro  
iOS 5 Augmented  
Reality

Kyle Roche

Apress®

# Pro iOS 5 Augmented Reality



**Kyle Roche**

**Apress®**

## **Pro iOS 5 Augmented Reality**

Copyright © 2011 by Kyle Roche

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. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13 (pbk): 978-1-4302-3912-3

ISBN-13 (electronic): 978-1-4302-3913-0

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.

President and Publisher: Paul Manning

Lead Editor: Kate Blackham

Technical Reviewer: Yosun Chang, Peter Ma, Graham Wood

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell,

Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson,

Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper,

Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing,

Matt Wade, Tom Welsh

Coordinating Editor: Corbin Collins

Copy Editor: Vanessa Moore

Compositor: MacPS, LLC

Indexer: BIM Indexing & Proofreading Services

Artist: SPi Global

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](mailto:orders-ny@springer-sbm.com), or visit [www.springeronline.com](http://www.springeronline.com).

For information on translations, please e-mail [rights@apress.com](mailto:rights@apress.com), or visit [www.apress.com](http://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](http://www.apress.com/info/bulksales).

Any source code or other supplementary materials referenced by the author in this text is available to readers at [www.apress.com](http://www.apress.com). For detailed information about how to locate your book's source code, go to [www.apress.com/source-code/](http://www.apress.com/source-code/).

---

# Contents at a Glance

<b>Contents</b> .....	<b>v</b>
<b>About the Author</b> .....	<b>ix</b>
<b>About the Technical Reviewers</b> .....	<b>x</b>
<b>Acknowledgments</b> .....	<b>xi</b>
<b>Preface</b> .....	<b>xii</b>
■ <b>Chapter 1: Introduction</b> .....	<b>1</b>
■ <b>Chapter 2: Hardware Comparison</b> .....	<b>15</b>
■ <b>Chapter 3: Using Location Services</b> .....	<b>31</b>
■ <b>Chapter 4: iOS Sensors</b> .....	<b>63</b>
■ <b>Chapter 5: Sound and User Feedback</b> .....	<b>87</b>
■ <b>Chapter 6: Camera and Video Capture</b> .....	<b>101</b>
■ <b>Chapter 7: Using cocos2D for AR</b> .....	<b>123</b>
■ <b>Chapter 8: Building a cocos2D AR Game</b> .....	<b>141</b>
■ <b>Chapter 9: Third-Party Augmented Reality Toolkits</b> .....	<b>181</b>
■ <b>Chapter 10: Building a Marker-Based AR Application with OpenGL ES</b> .....	<b>211</b>
■ <b>Chapter 11: Building a Social AR Application</b> .....	<b>225</b>
■ <b>Chapter 12: Facial-Recognition Techniques</b> .....	<b>263</b>
■ <b>Chapter 13: Building a Facial Recognition AR App</b> .....	<b>297</b>
<b>Index</b> .....	<b>333</b>

---

# Contents

<b>Contents at a Glance</b> .....	<b>iv</b>
<b>About the Author</b> .....	<b>ix</b>
<b>About the Technical Reviewers</b> .....	<b>x</b>
<b>Acknowledgments</b> .....	<b>xi</b>
<b>Preface</b> .....	<b>xii</b>
<b>■ Chapter 1: Introduction</b> .....	<b>1</b>
Augmented Reality in the Real and Cyber World .....	1
Pop Culture .....	3
Gaming and Location-Based AR .....	4
Getting Your House in Order .....	4
Signing Up for GitHub.....	4
Accessing GitHub from Your Machine.....	5
Setting Up Xcode 4.2 and Your Developer Account .....	7
Linking an Xcode Project to GitHub.....	8
Creating Our Xcode Project.....	10
Connecting Our Project to the Remote Repository.....	12
What's Next?.....	12
Location Services.....	12
Sensor Programming .....	12
Lights, Camera, Action . . . ..	12
Gaming Frameworks.....	12
Third-Party Frameworks .....	13
Summary .....	13
<b>Hardware Comparison</b> .....	<b>15</b>
Out with the Old .....	15
Hardware Components .....	16
Camera Support.....	16
Detecting Location Capabilities .....	20
Wired for Sound .....	24
Checking for Video Capabilities .....	24
Acceleration and Gyroscope .....	26

Enforcing Hardware Requirements .....	27
Summary .....	29
<b>Using Location Services .....</b>	<b>31</b>
You Are Here .....	31
Standard Location Service .....	34
Significant-Change Location Service .....	38
Geographic Region Monitoring Service.....	39
Altitude.....	47
Viewing on the Map .....	48
Centering the Map and Setting the Displayed Region .....	50
Changing the Map Type .....	52
Adding Annotations to the Map.....	54
Reverse Geocoding .....	58
Summary .....	60
<b>iOS Sensors.....</b>	<b>63</b>
Orientation Sensors .....	63
Using the Accelerometer .....	64
Low-Pass Filtering .....	69
Using the Gyroscope .....	69
Magnetometer .....	77
Summary .....	85
<b>Sound and User Feedback .....</b>	<b>87</b>
Audio Data Formats .....	87
So, Which Data Format Is for Me? .....	88
What About File Formats?.....	88
Bit Rates and Quality.....	88
Sample Rates.....	89
Converting Audio for Use in iOS.....	89
Playing Sound in an iOS Application .....	92
System Sound Services .....	93
AVAudioPlayer Class .....	93
Experimenting with the Multiple Audio Players .....	94
Playing Positional Sound .....	96
User Feedback Through Vibration.....	96
Recording Sound.....	96
Initializing the Audio Recorder.....	96
Summary .....	100
<b>Camera and Video Capture .....</b>	<b>101</b>
Quick Review .....	101
Photo Capture .....	102
Using Storyboards.....	103
Using the Camera .....	109
Saving Images in Different Formats .....	110
E-mailing an Image.....	111
Video Capture.....	114
Building a Base on the Video Preview .....	114

Building a Base for Frame Capture .....	116
Summary .....	121
<b>Using cocos2D for AR .....</b>	<b>123</b>
Overview .....	123
Installation .....	123
Installing the Project Templates .....	124
Creating a Project .....	125
Hello Augmented World .....	126
Adjusting the Default View .....	126
Adding the Camera View .....	128
Scaling the Camera View .....	129
cocos2D Concepts .....	130
Scenes .....	130
Director .....	130
Layers .....	131
Adding Effects .....	131
Handling Touch Events .....	131
Visual Effects .....	133
Adding Sound Effects .....	135
Adding a HUD Layer .....	135
Summary .....	139
<b>Building a cocos2D AR Game .....</b>	<b>141</b>
Overview .....	141
Creating the Project .....	142
Camera View .....	145
Creating the Game Menu .....	147
Artwork .....	149
Helper Code Directory .....	152
Finishing the Menu Screen .....	152
Adding the Menu Option .....	160
Enable Camera Support .....	163
Finishing the Action Layer .....	171
Here Come the Pumpkins .....	172
Ending the Game .....	176
Summary .....	179
<b>Third-Party Augmented Reality Toolkits .....</b>	<b>181</b>
Overview .....	181
Powered by String .....	182
String's Basic Workflow .....	184
Extra Functionality .....	185
Unity Integration .....	186
Advanced Shaders and OpenGL Features .....	186
Qualcomm SDK .....	187
Building Our Own QCAR Demo .....	190
Creating the Xcode Project .....	192
EAGLView .....	194
Redirecting the UIView .....	206

ARKit .....	208
Summary .....	208
<b>Building a Marker-Based AR Application with OpenGL ES .....</b>	<b>211</b>
Building a Marker .....	211
Our Marker .....	212
OpenGL ES .....	212
Creating the Project .....	213
Adding the String SDK .....	213
EAGLView .....	213
Creating the AR ViewController .....	219
Summary .....	223
<b>Building a Social AR Application .....</b>	<b>225</b>
Getting Set Up .....	225
Creating a Facebook Application .....	225
Cloning the Facebook iOS SDK .....	226
Vocabulary Lesson .....	227
Azimuth .....	227
Corrected Heading .....	228
Building the Application .....	228
Credits .....	228
Required Frameworks .....	228
Adding the Facebook iOS SDK .....	229
And, We're Off! .....	229
Listening for Sensor Updates .....	235
Storing Coordinates .....	238
Adding Social Context .....	249
Graph for Friends .....	251
Summary .....	262
<b>Facial-Recognition Techniques.....</b>	<b>263</b>
Choices for Facial Recognition .....	263
OpenCV .....	263
iOS 5 CIDetector Class .....	264
Face.com .....	264
Using the OpenCV Approach .....	264
Capturing the Image for Testing .....	265
Haar Cascades .....	271
OpenCV Review .....	277
Using the CIDetector Class Approach .....	277
CIDetector Review .....	280
Using the Face.com API Approach .....	280
Faces.detect API Call .....	280
Adding Face.com Support to Our Sample .....	281
Face.com API Key .....	282
Adding the Face.com Callout .....	283
Measuring Performance .....	287
Summary .....	295



<b>Building a Facial Recognition AR App .....</b>	<b>297</b>
The Application's Purpose .....	297
Technology Used.....	297
Getting Set Up.....	298
Face.com .....	298
cocos2D .....	299
Setting Up Our Twilio Account .....	299
Downloading the ASI-HTTP-Request Library .....	300
JSON-Framework .....	300
Project Structure.....	300
Setting Up the Main Scene .....	302
Enabling the Camera.....	304
Face.com API .....	313
Using the ASI-HTTP-Request Library .....	314
Creating the POST Request Method.....	316
Creating the NSTimer.....	317
Parsing the Output.....	320
Constructing Our HUD Layer .....	323
Adding a Twilio Callout .....	330
Summary .....	331
<b>Index .....</b>	<b>333</b>

---

# About the Author



**Kyle Roche** has been focused on emerging technologies since 2000. During his time at Appirio, he led some of the world's first and largest Google and Force.com cloud platform migrations. He is the chief architect behind RingDNA ([ringdna.com](http://ringdna.com)) and the co-founder of 2lemetry ([2lemetry.com](http://2lemetry.com)). Mobile applications and connected electronics (M2M) are the main focus of all of Kyle's projects. Augmented reality and gaming frameworks play a large part in how these applications are visualized. Kyle studied mathematics and was on the wrestling team at the University of New Mexico. He currently lives in Denver, Colorado with his wife Jessica and their four children: Aodhan, Avery, Kelly, and Timmy. If there is ever free time outside of family life, Kyle spends it playing hockey or building iOS applications for local nonprofits. You can find him at [kyleroche.com](http://kyleroche.com).

---

# About the Technical Reviewers



**Yosun Chang** has been creating apps for iOS and Android since early 2009, and is currently working on a next generation 3D and augmented reality mobile games startup called nusoy. Prior to that, since 1999 she did web development on the LAMP stack and Flash. She has also spoken at several virtual world, theater, and augmented reality conferences under her artist name of Ina Centaur. She has a graduate level background in physics and philosophy from UC San Diego and UC Berkeley. An avid reader who learned much of her coding chops from technical books like the current volume, she has taken care to read every single word of the chapters she reviewed — and vet the source. Contact her @yosunchang on Twitter.



**Peter Ma** has been working with web, iOS, Android, WebOS, and WP7 since 2007. He has been building projects from database design to mobile presentation. Peter has won many hackathons and developer challenges, all using native tools. He has won a TED Prize sponsored challenge and gave a TED talk about building mobile apps during TED Global2010. The mobile app Pickup Sports was the foundation for Spotvite and had over 80,000 signups. Peter is also involved in many open source projects; he has pioneered the TEDx app that helps organizers to build their own iOS and Android applications. Contact him @Nyceane on Twitter.



**Graham Wood** is a mobile application developer whose primary focus is the iOS platform. He has 11 years of experience in software development, with most of that time spent writing software for safety critical embedded systems for commercial aircraft. Graham holds a Bachelor of Science degree in Computer Science from the University of Minnesota. His company, Wood App Developers LLC, develops mobile applications for clients, along with its own suite of iOS applications. He can be reached at [graham@woodapps11c.com](mailto:graham@woodapps11c.com) or followed on Twitter at @woodapps11c.

---

# Acknowledgments

I wrote this book in the middle of a transition between startups. We were transitioning from one company and starting two new projects. Furthermore, iOS 5 was in beta for most of the timeline. It was a very difficult time to be writing a book. It wouldn't have been possible without the support from the Apress team, led by Corbin Collins and Steve Anglin.

From a technical perspective, I want to thank my colleague Sergey Loshchilov. Sergey is a post-graduate student from Nizhny Novgorod State Technical University. He was a huge help on the OpenCV sections and the new iOS 5 APIs. Sergey is publishing a paper comparing the algorithms in the more popular AR frameworks. I'll post links to it from [kyleroche.com](http://kyleroche.com).

On a personal note, my wife and four kids have been very supportive with the long nights and weekends it takes to get a book published. I'd like to thank them for their contribution of patience and time. I had a chance to have the kids participate in the facial recognition chapter, which was fun for both them and me.

---

# Preface

This was a fun and interesting book to write! Augmented reality is a fascinating new field with tons of potential to reshape how we integrate technology into our everyday lives. Companies and toolkits are popping up each week trying to capture a piece of this emerging market.

The goal of this book was to provide you with a jump-start on building these types of applications. I begin by discussing the basic foundations of the app, such as the compass and accelerometer, and move on to more advanced ideas behind image processing.

This book is intended for experienced iOS developers. You should have moderate experience with Xcode and objective-C. I use third-party frameworks and some of the new iOS5 APIs to show you how to build augmented reality applications for location, social, and gaming purposes. You can download the source code for this book from the book's page on Apress.com, or check out [www.apress.com/source-code/](http://www.apress.com/source-code/).

# Introduction

Welcome to *Pro iOS 5 Augmented Reality*. Augmented reality (AR) has existed in sci-fi movies for decades, is used in the military for head-up displays (HUDs), and until recently, has been a thing of the future. With the upswing in mobile applications since the introduction of the iPhone and the Android operating system, applications such as Layar ([www.layar.com](http://www.layar.com)), Metaio's Junaio ([www.junaio.com](http://www.junaio.com)), and Wikitude ([www.wikitude.com](http://www.wikitude.com)) have put augmented reality in the hands of the everyday consumer. In this book, I'll walk you through how to create your own augmented reality applications for iOS.

*Time* magazine named augmented reality among the top-ten technology trends for 2010. *Time* barely scratched the surface on the potential applications of AR. They selected a few vendor application platforms, such as Layar, and also discussed some more day-to-day applications, such as that employed by the United States Postal Service (USPS).

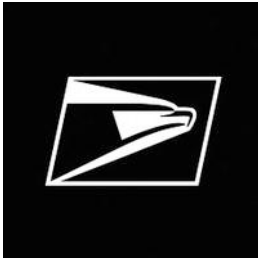
## Augmented Reality in the Real and Cyber World

The USPS introduced an augmented reality application to its web site in 2010. If you've ever mailed something from the post office, you can attest to the fact that quickly selecting a box that fits your needs without holding up the line is a near impossible task. Either you're stuck wasting a lot of space with a bigger box or you're holding up the 20 people behind you while you jam all your items into the box that *almost* fits everything. The USPS took a shot at making this easier, without requiring you to leave your home or office. Basically, you go to the USPS web site ([www.prioritymail.com](http://www.prioritymail.com)) and use the Virtual Box Simulator and your webcam to try out different box sizes before you head out for the post office. It works like this:

Print out a special icon (the USPS eagle) so the simulator knows where to put the hologram of the virtual box. See Figure 1-1.

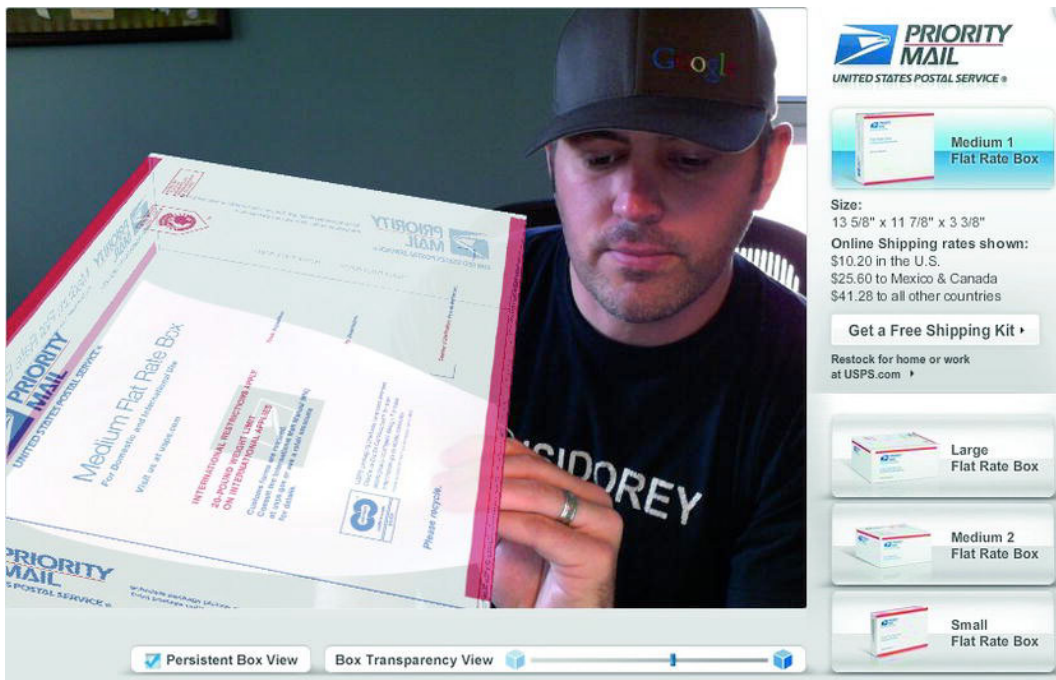
1. Make sure your webcam is enabled.

2. Launch the Virtual Box Simulator. Put the printed image in the view of the webcam and the simulator puts a hologram of different options for shipping containers around the image. See Figure 1–2.



**Figure 1–1.** This eagle icon is printed and used by the USPS to augment your camera's view with a simulated shipping container.

There are a few basic principles to follow when creating icons or markers for recognition. For traditional markers, you want high-contrast objects that carry a certain uniqueness and aren't found in common scenarios. In fact, random images are often more effective. Also, you want to use images that have a certain rotation and aren't symmetrical either horizontally or vertically. This helps the AR program recognize orientation and adjust accordingly. The USPS marker is a good example of these principles.



**Figure 1–2.** The hologram is overlaying the printed icon.

Notice in Figure 1–2 that the simulator allows you to adjust transparency, move your to-be-shipped item on different angles and rotations, and experience exactly which shipping container you need to ship your materials. The USPS uses the marker and some sort of recognition algorithm to find it in the live camera view, track its orientation, and augment the picture with the current box you’ve selected.

## Pop Culture

There are hundreds of other applications for AR in advertising, real estate, the automotive industry, and especially in consumer spending. Although statistics suggest that well over half the population of the United States has tried online shopping, the revenue accounts for only eight percent of consumer spending, according to Wikipedia. Obviously, there are various theories as to why the traction hasn’t taken more market share. Among them are the basic concerns about privacy and security online, but there are equally as many theories on the lack of physical interaction accounting for an unknown product. In some cases, such as with clothing, you just need to see and feel what you’re buying.

Sometime in late 2010, we started seeing multiple AR experiences penetrate the retail market. Growing up in the late ’70s, I recall Jane Jetson trying out new hairstyles with the push of a button, or Luke Skywalker listening to the brief about the approach methods for the Death Star over a holographic 3D display. This type of experience is now available for consumers. From trying on new clothing and accessories, to finding out where your grocer’s apples are grown, consider some of these recent examples:

- *Lego’s Digital Box*: An in-store kiosk by Lego lets a child hold up the box set he or she is considering in front of a camera on the kiosk, which then overlays the fully constructed set right on top of the box. The child can move it around, turn it over, and get a feel for whether this is the set they really want to put on their Christmas list.
- *Zugara*: Zugara uses its Magic Mirror, which lets an online shopper stand in front of a webcam and try on different clothing styles, without the aid of a mouse or keyboard. In addition to overlaying the clothes from the online catalog, Zugara overlays controls in the camera’s view so that the user can use gestures to interact with menu options or share their new outfit over their social network.
- *FoodTracer*: This project by Giuseppe Costana uses image recognition in AR to give grocery shoppers more information about the food they are buying. Simply wave a smartphone’s camera in front of the grocer’s shelf and information becomes available.

There are obvious advantages and appeals to the interactive experience. However, also consider some of the supplemental values of AR. The back end of most of these applications lives on the cloud. Image-recognition algorithms and the camera’s interpretation itself are primarily running on the device, but advertising data, contextual information, location directories, and other dynamic content linked to the AR view can



be loaded from the cloud and in a centralized location where updates are seamless and the applications can always remain current.

## Gaming and Location-Based AR

Retail and in-store kiosks are not the only places that AR is becoming a trend. Social networks, location-based services, and gaming are leveraging AR as well. Imagine using your camera to interact with the real world in a gaming scenario. I recently saw a demo at a conference in which 3D models of zombies were rendered in the AR view of an iPhone and the user could shoot them by just tapping on the screen. It has spawned a secondary market for accessories like the iPhone gun, covered on [www.augmentedplanet.com](http://www.augmentedplanet.com). This rifle-sized accessory mounts your iPhone to the scope, so you can have a realistic experience of shooting 3D zombies in an AR fashion.

In this book, we'll cover the basics for creating your own AR game. We'll look at various approaches to this project, including some available SDKs to speed your time to market.

## Getting Your House in Order

There are a few steps you'll need to take to make sure everything on your machine is ready to go for iOS programming. In this book, we'll be using Xcode 4.2 only, and we'll be storing all our projects on GitHub. Xcode shipped with native Git integration for source-code management, so we'll be taking advantage of that to make things easier and save setup time.

## Signing Up for GitHub

If you already have a GitHub account, you can skip this section. If not, you're going to need one to download the assets and starting points for each chapter. Open a browser to [www.github.com](http://www.github.com) and click the big **Signup** button in the middle of the page, as shown in Figure 1-3.



**Figure 1-3.** *The Signup button is easy to find on GitHub.*

For this book, we're going to be accessing the Git repositories that I've already set up for each chapter; and, if you're into sharing, we'll be posting any variations back for fellow readers. With that in mind, we really only need the "Free for open source" account type. Click the **Create a free account** button and fill out your information.

## Accessing GitHub from Your Machine

If you've used GitHub before, you may skip this section, which is for users who have not yet created an SSH key for use with GitHub.

There are a few ways to access GitHub's remote repositories from your machine. We'll be using SSH access, which means we'll need to generate a token and post it to GitHub. Open Terminal (**Applications** ► **Utilities** ► **Terminal**) from your Mac. Take a look at Listing 1-1. Follow this same pattern in your Terminal window. I'll explain the steps next.

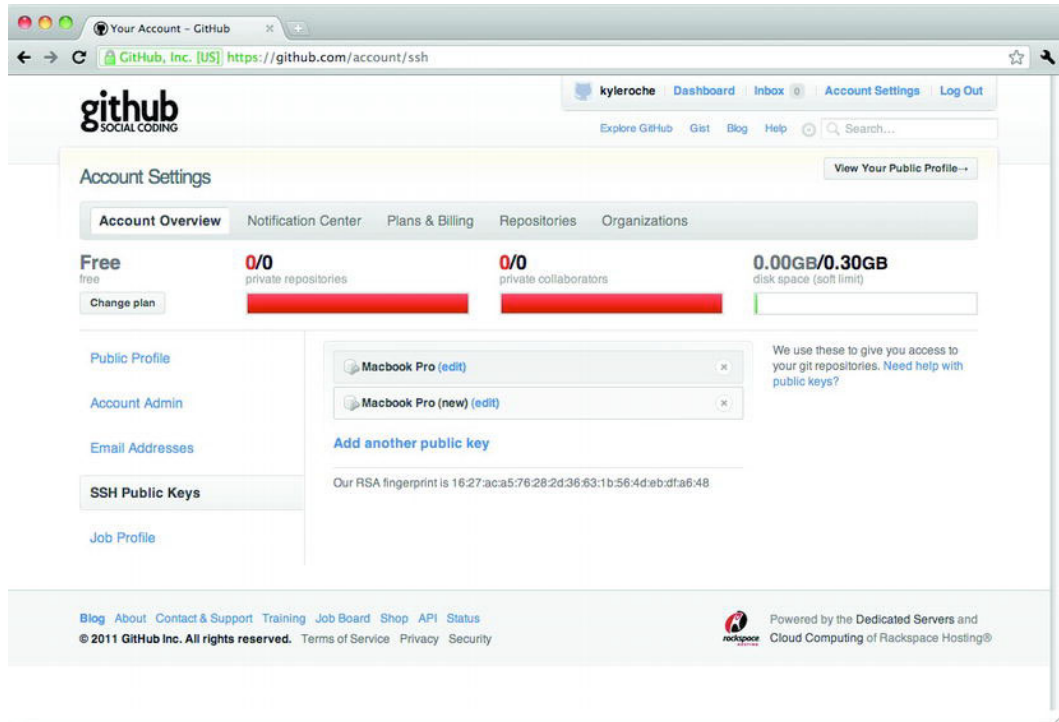
### Listing 1-1. Create Your SSH Key on Your Mac

```
Kyle-Roches-MacBook-Pro-2:~ kyleroches$ cd ~/.ssh
Kyle-Roches-MacBook-Pro-2:~.ssh kyleroches$ ls
known_hosts
Kyle-Roches-MacBook-Pro-2:~.ssh kyleroches$ ssh-keygen -t rsa -C "kyle@isidorey.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/kyleroches/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): [enter a passphrase here]
Enter same passphrase again: [enter your passphrase again]
Your identification has been saved in /Users/kyleroches/.ssh/id_rsa.
Your public key has been saved in /Users/kyleroches/.ssh/id_rsa.pub.
The key fingerprint is:
26:9d:3a:82:fe:r9:gf:ba:39:30:6b:98:16:fe:3b:2c kyle@isidorey.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|
|       . . . . 4
|      . . . . N
|     . o . . +R
|    . . . . -|=
|   . . + . - E . o
|  + . = 0000
+-----+
Kyle-Roches-MacBook-Pro-2:~.ssh kyleroches$ ls
id_rsa          id_rsa.pub     known_hosts
```

The directory listing commands might have different results if you have existing keys already. In this case, you probably want to back up your key directory, just to be safe. First, we're going to use the `ssh-keygen` utility to create a public/private rsa key pair. The utility will ask us for a passphrase. This is optional, but passphrases do increase security. Passwords, as most of us realize, aren't all that secure on their own. Generating a key pair without a passphrase is equivalent to saving your passwords in a plain-text file on your machine. Anyone who gains access can now use your key. If you're lazy and concerned about typing it in every time, don't fret. Keychain (since we're all on a Mac) will allow you to store it after the first time you use this key pair.

So, we have a key pair. It's stored in the newly created `id_rsa.pub` file you see in your directory listing. Open this file in your favorite plain-text editor and copy all of its contents. It's important that you copy everything, even the headers.

Return to Github, which should be open to your account in your browser. Open your **Account Settings** page from the top-left navigation menu. Then open the subtab on the left-hand side called **SSH Public Keys**. You should see something similar to Figure 1–4.



**Figure 1–4.** Open the SSH Public Keys dialog on GitHub.

Find the **Add another public key** link in the middle of the page. That will open a dialog where you will paste the contents of the `id_rsa.pub` file we just created. That's it! You're now set up in GitHub and your machine can access your repositories using SSH.

Because we'll be using SSH access in this book, let's quickly set up our default preferences before we move on.

We need to configure our local Git client to use the credentials that we received when signing up for GitHub. First, run the following commands from Listing 1–2 in your Terminal window to set some global flags for Git. This, in combination with your SSH keys, will authenticate your Git client to the remote repository.

**Listing 1–2.** Create Your SSH Key on Your Mac

```
Kyle-Roches-MacBook-Pro-2: kyleroche$ git config --global user.name "Kyle Roche"
Kyle-Roches-MacBook-Pro-2: kyleroche$ git config --global user.email "kyle@isidorey.com"
```

## Setting Up Xcode 4.2 and Your Developer Account

If you have Xcode 4.2 already set up, you may skip this section.

To publish an app to the App Store, you need Xcode and an Apple Developer account. We can take care of both of these steps at the same time. Open your browser to <http://developer.apple.com/programs/register/> and click the **Get Started** button in the header. There are a few paths to follow here. If you want to use an existing Apple ID, you can fill that in and continue along. See Figure 1–5. Alternatively, you can create a new ID for iOS development. That might not seem reasonable, but there are a few pitfalls with using one account.

### Do you have an existing Apple ID you would like to use?


**Create an Apple ID**

If you have not registered as an Apple developer or do not have an iTunes, Apple Online Store or MobileMe account, you will need to create an Apple ID.

**Use an existing Apple ID**

If you have already registered as an Apple developer or have an iTunes, Apple Online Store or MobileMe account, you can use your existing Apple ID to sign in.

Note: If you intend to enroll in a paid Developer Program for business purposes, you may prefer to create a new Apple ID that is dedicated to your business transactions and used for accounting purposes with Apple. If your Apple ID is associated with an existing iTunes Connect account, please create a new Apple ID to avoid accounting and reporting issues.

 **What is an Apple ID?**

You will use your Apple ID to access certain information and resources, or to register for an event.



**Figure 1–5.** Use existing Apple ID or create a new one?

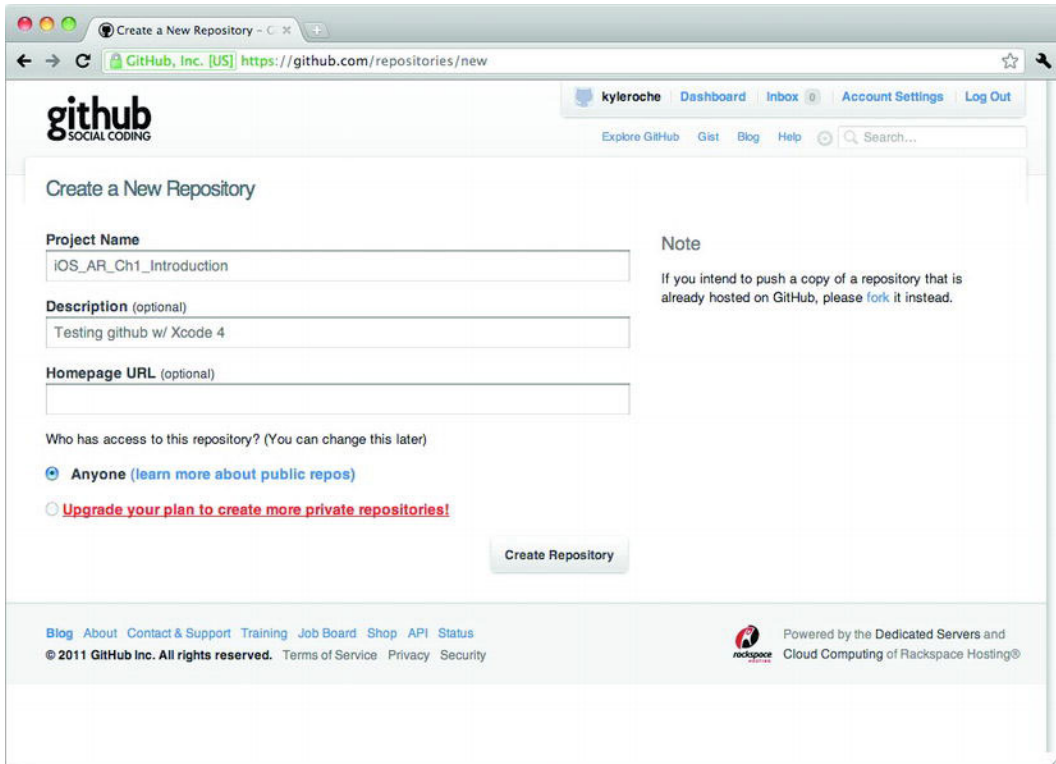
**NOTE:** Choosing whether to consolidate your Apple IDs or create a second one depends on your intent in regards to publishing your apps in the future. Apple has a restriction on which publishing type you can link to an account. There are two ways to publish applications: through the App Store and through Apple’s Enterprise Distribution program. An Apple ID cannot be tied to both publishing methods. Make sure you decide which ID will be responsible for which method of publishing, if you are going to cover both scenarios.

If you just want to use your account to develop and debug, then use an existing account. It’s probably the simplest path. After you are registered, log in to the **iOS Dev Center**. Find the link for **Downloads**. At the time of this writing, there are only two choices: **Download Xcode 4.2** and a series of links around **iAd Producer 1.1**. Download Xcode 4.2 to your machine. The download is fairly large. This is one of the drawbacks of Xcode. Each upgraded version, which have started coming more frequently since iOS, requires a new full download of the IDE.

We now have our IDE and our source control strategy set up. Let's connect the two and make sure we're ready to get started.

## Linking an Xcode Project to GitHub

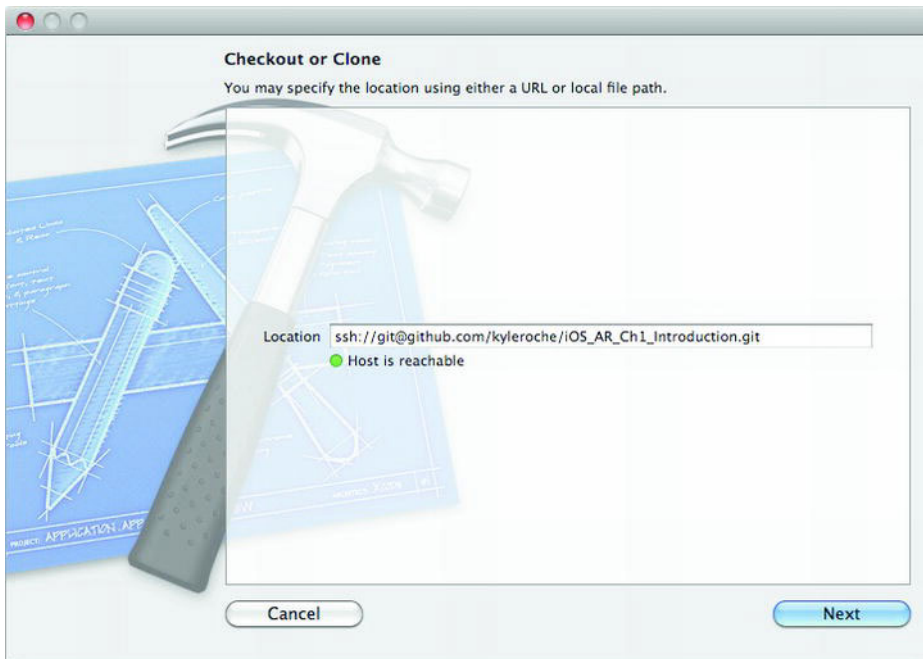
Return to GitHub in your browser. Click on **Dashboard** in the top-left navigation bar and find the **New Repository** button. For **Project Name** I'm going to use `iOS_AR_Ch1_Introduction`. Feel free to choose your own name, or if you're an experienced GitHub user, you can fork my repository from <https://github.com/kyleroche>. See Figure 1–6 for the options I've chosen.



**Figure 1–6.** Create a new repository at GitHub.

Next, take note of your repository's SSH URL. You will see it in the header of the confirmation page. It will be in a form similar to `git@github.com:kyleroche/iOS_AR_Ch1_Introduction.git`. You are going to need this in the next step.

Launch Xcode on your local machine. In the **Welcome to Xcode** dialog box that launches on startup, you should have an option on the left side called **Connect to a Repository**. Click this option and enter the SSH URL for your GitHub repository. See Figure 1–7 for my configuration.



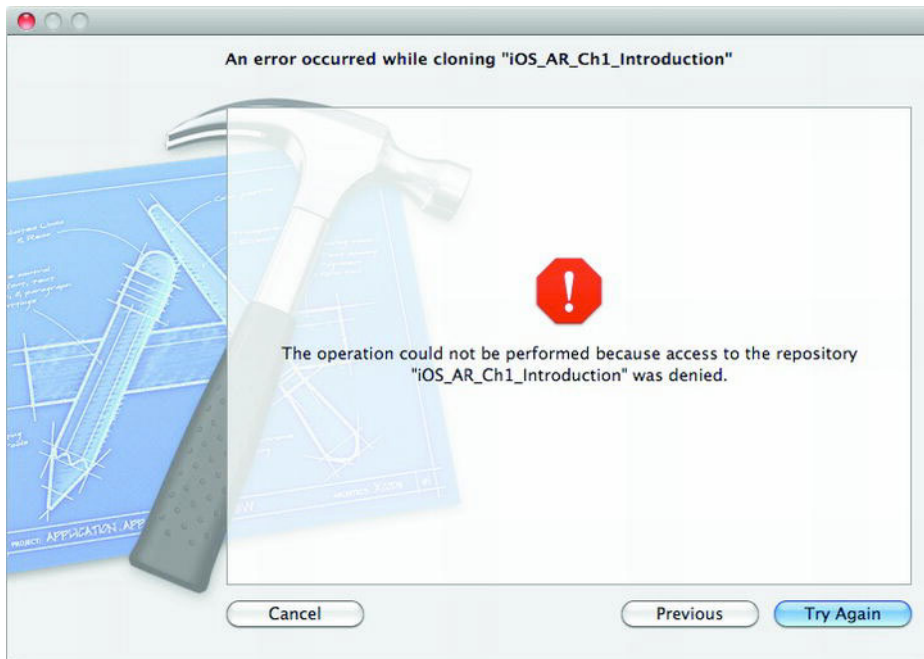
**Figure 1–7.** Clone your GitHub repository for local access.

Xcode validates your location and the ability to clone the repository. Wait a few moments until your indicator is green and the message states **Host is reachable**, then click **Next**.

You are presented with a prompt to name your new project. I am using the same name as my GitHub repository, `iOS_AR_Ch1_Introduction`, for simplicity. Make sure that **Type** is set to **Git** and click **Clone**.

Next, choose a location for your local repository and click **Next**.

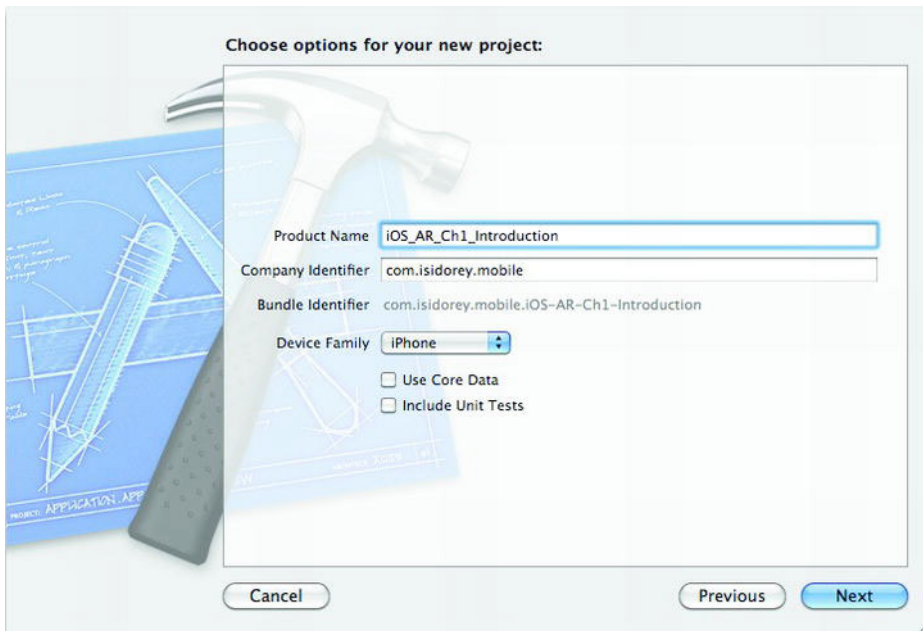
**NOTE:** At the time of this writing, Xcode 4.2 still has a few bugs in regards to using Git. The first of them should have manifested itself in this last step. If your version still has issues, you will get an error similar to that shown in Figure 1–8. If this is the case, simply click **Try Again**, select the same location, choose **Replace** and everything should be fine.



**Figure 1–8.** A defect in early releases of Xcode 4.2 threw invalid errors. Simply click *Try Again* and it goes away.

## Creating Our Xcode Project

From Xcode's **Welcome to Xcode** screen, select **Create a New Xcode Project**. We're not going to be doing much coding in this project, so the template type isn't all that important. I'm going to select a Windows-based application template for simplicity. The next screen has a few more important options. You are now being asked for your **Product Name**. This is used as a suffix to your fully qualified **Bundle Identifier**. This is where things will start to diverge a bit. Unless you're involved in team-based development, this option will be unique to your machine. I'm going to, again, use the same name as my GitHub repository to make things easy. My options are shown in Figure 1–9.



**Figure 1–9.** Choose your new project options.

Your **Company Identifier** is going to be different as well. Until we discuss distributing applications either over the App Store or over the Enterprise Distribution options, the **Company Identifier** can be a reverse DNS format. Click **Next** when you have everything filled out.

You are finally prompted to find a local location for this project. Make sure that you select the same directory as you did for cloning the GitHub repository. Also, make sure that **Create local git repository for this project** is *not* selected, as shown in Figure 1–10.





## Connecting Our Project to the Remote Repository

There are quite a few online tutorials covering the integration of Xcode and GitHub. To get started connecting your project, and to learn the latest features and changes, visit <http://help.github.com/mac-set-up-git/>.

## What's Next?

Here's a quick walk-through of some of the key sections in this book.

### Location Services

Most AR use cases, at least the early ones, have some element that relates to the user's location. Whether it's presenting nearby restaurants in AR view or finding your friends recommendations on night life, location-based AR kicked off the trend. In Chapter 3, we'll learn about the mapping and geo-location capabilities of iOS using real examples that we can apply to our own AR apps.

### Sensor Programming

Creating an AR application requires quite a bit of integration with the native sensors on the iPhone or iPad. Sensors include the accelerometer, the digital compass, and the gyroscope. In Chapter 4, I'll introduce you to sensor development with small projects demonstrating the key features we'll reuse in our AR applications.

### Lights, Camera, Action . . .

In Chapter 5, I cover sound and user feedback. Sound isn't the most prominent feature in AR apps, but it does lead to a better user experience. After that, in Chapter 6, we'll dive into camera and video programming. Because AR apps are all overlaid on our camera view, this is an essential chapter to understand before we start constructing the larger AR projects at the end of the book.

### Gaming Frameworks

I choose to use Cocos2D to demonstrate AR gaming capabilities. In Chapter 7, you'll get a primer on Cocos2D's essentials and we'll follow that up with a real application in Chapter 8.

---

## Third-Party Frameworks

In Chapter 9, I talk about a few third-party frameworks that make marker-based augmented reality application development easier. We follow that up with a real example, then move on to more complicated frameworks, such as OpenCV (Open Computer Vision), which is an open source library for things such as facial or complex-object recognition. Facial recognition on the device itself has some limitations. Mostly, these limitations are related to hardware capabilities. We'll discuss a few more creative ways to supplement facial recognition using publicly available APIs.

## Summary

I hope you learn much from this book. AR is such a new concept in mobile apps and has endless possibilities. The developer community is just beginning to scratch the surface of possibilities. I hope this book gives you a jump-start on your own journey into the AR world.

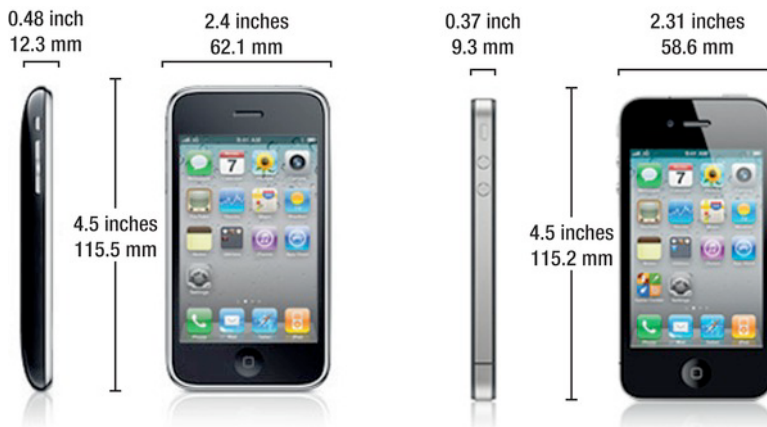
Let's get started by reviewing some of our application layout options and frameworks for putting together our own AR applications. In the next chapter, we'll discuss the hardware we'll be using in this book and the major features of the different models.

# Hardware Comparison

Every mobile developer worries about hardware compatibility. However, the main benefit of developing for Apple's iOS line is standardization among hardware. True, there are different evolutions of the devices, but there is only one vendor: Apple! With other mobile operating systems, you have to worry about OEM vendors and their unlimited variations of hardware configurations. Let's take a look under the hood of the more recent iOS devices.

## Out with the Old

We're going to be using both the iPhone and the iPad for our sample projects. However, whenever the code is completely portable between platforms, we'll be coding for only the iPhone. Figure 2-1 illustrates the physical dimensions of the iPhone 4.



**Figure 2-1.** Here we see the physical dimensions of the iPhone 4.

The iPad hasn't been on the market as long as the iPhone, but has no less traction for its purpose. In this book, we're only going to use the iPad 2 for our examples. There are a few reasons for this. Most important, the iPad (first-generation) is missing a front-facing

camera, and we're going to work on some facial recognition programming later in this book. Figure 2-2 shows the physical characteristics of the iPad 2.



**Figure 2-2.** Here we see the physical characteristics of the iPad 2.

## Hardware Components

Instead of just Listing out the hardware specs for each of these devices, let's take a minute to discuss what an augmented-reality application might require. First, and most obviously, the device would need a camera. Most augmented-reality use cases have some sort of requirement for location and direction, so a GPS and a compass would be useful. It might be useful to either listen for sound or output sound, so we should look for a microphone and speaker support. We'll incorporate some gaming features into our samples, so we probably need some hardware acceleration capabilities and a graphics toolkit. Let's translate this to Apple's terminology and walk through how to check for these various hardware components.

**NOTE:** As we look at the different hardware components of the iOS devices, I'll be posting some small code snippets. In this chapter, they will be out of context of a sample project. However, if you want to follow along, all the code in this chapter is in the following GitHub repository:

<https://github.com/kyleroché>

## Camera Support

The camera has come a long way since Apple launched the iPhone 3GS. It still leaves a lot to be desired when compared to some other hardware models that make their differentiation around the camera, but it'll more than suffice for our purposes.

There are two ways to build augmented-reality applications over the video capabilities of the phone. First, you can actively inspect the video capture for elements, recognizable objects, and so forth. Or, you can use the video capture as the background for your application, while completely ignoring the contents. We see this approach quite a bit in augmented-reality browsers because of the heavy processing involved with constantly inspecting the video capture. In this book, we'll walk through samples of both of these approaches.

Table 2–1 details the specifics about the camera and video capability of the hardware we'll be using in this book.

**Table 2–1.** *Camera Details for iPhone 3GS, iPhone 4, and iPad 2*

	iPhone 3GS	iPhone 4	iPad 2
Back Camera Video	VGA, 30 frames per second with audio	HD (720p), 30 frames per second with audio	HD (720p), 30 frames per second with audio
Front Camera Video	—	VGA, 30 frames per second with audio	VGA, 30 frames per second with audio
Back Still Camera	3-megapixel still camera	5-megapixel still camera	HD still camera with 5x digital zoom
Flash	—	LED flash	—

## Detecting the Camera

We can programmatically detect what camera is available on our device by using the UIImagePickerController class. There is a method called `isSourceTypeAvailable` we can use to determine whether the camera we want to use is available on this device. Listing 2–1 shows an example of how to use the `isSourceTypeAvailable` method.

**Listing 2–1.** *Checking for a Camera, Then for a Front-Facing Camera*

```

BOOL cameraAvailable = [UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera];
BOOL frontCameraAvailable = [UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerCameraDeviceFront];

if (cameraAvailable) {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Camera"
                                                         message:@"Camera Available"
                                                         delegate:self
                                                         cancelButtonTitle:@"OK"
                                                         otherButtonTitles:nil, nil];

    [alert show];
    [alert release];
} else {

```

```

        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Camera"
                                                         message:@"Camera NOT Available"
                                                         delegate:self
                                                         cancelButtonTitle:@"OK"
                                                         otherButtonTitles:nil, nil];

        [alert show];
        [alert release];
    }

    if (frontCameraAvailable) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Camera"
                                                         message:@"Front Camera
Available"
                                                         delegate:self
                                                         cancelButtonTitle:@"OK"
                                                         otherButtonTitles:nil, nil];

        [alert show];
        [alert release];
    } else {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Camera"
                                                         message:@"Front Camera NOT
Available"
                                                         delegate:self
                                                         cancelButtonTitle:@"OK"
                                                         otherButtonTitles:nil, nil];

        [alert show];
        [alert release];
    }
}

```

This code block will demonstrate the results in a quick UIAlertView pop-up. Actually, you'll get two pop-ups from this example. You can see in the first few lines of Listing 2-1 where we are checking for the existence of UIImagePickerControllerSourceTypeCamera to see whether the camera is available. We next check for the existence of the front-facing camera using the UIImagePickerControllerCameraDeviceFront parameter. The isSourceTypeAvailable method returns a BOOL value. We use that in our if/else statements, and display the appropriate UIAlertView for each check.

Now, why would we have to check for a camera when we're only using iPhone 3GS, iPhone 4, and the iPad 2? Don't they all have cameras? Yes, they do all have cameras. However, we'll be coding in Xcode using the simulators, as well. Unlike some other mobile operating systems' IDEs, Xcode simulators do not have camera support.

Take a look at Figure 2-3. Both of these screenshots were taken from an iPhone 4 device running the previous sample code block. Since both of the if/else blocks execute in the code block, we get two UIAlertView dialogs. As you can see from the dialogs, both checks returned True, meaning the camera and the front-facing camera are available to use.