Learn iPhone and iPad game apps development using iOS 6 SDK

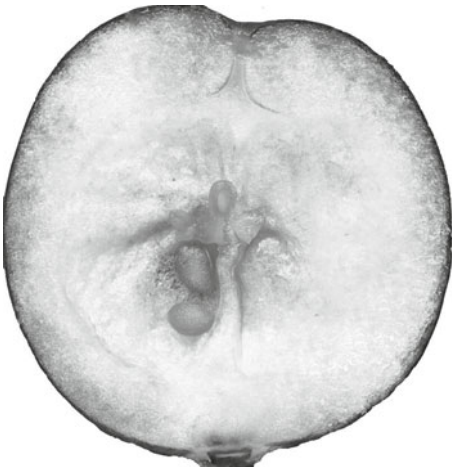
# Beginning iOS 6 Games Development

Lucas Jordan

ClayWare Games™

Apress®

# Beginning iOS6 Games Development



Lucas Jordan

Apress®

**Beginning iOS6 Games Development**



ISBN 978-1-4302-4422-6

ISBN 978-1-4302-4423-3 (eBook)



President and Publisher: Paul Manning
Lead Editor: Steve Anglin
Developmental Editor: Tom Welsh
Technical Reviewer: Tony Hillerson
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, 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: Katie Sullivan
Copy Editor: Carole Berglie
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

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.

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 is 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.

*To being my own boss.*

# Contents at a Glance

# Contents

# About the Author

**Lucas L. Jordan** is a lifelong computer enthusiast who has worked for many years as a developer, with a focus on user interface. He is the author of *JavaFX Special Effects: Taking Java RIA to the Extreme with Animation, Multimedia, and Game Elements* and the co-author of *Practical Android Projects*, both by Apress. Lucas is interested in mobile application development in its many forms. He has recently quit his day job to pursue a career developing apps under the name ClayWare, LLC. Learn more at `http://claywaregames.com`.

# About the Technical Reviewer

**Tony Hillerson** is a mobile developer and cofounder at Tack Mobile. He graduated from Ambassador University with a BA in MIS. On any given day he may be working with Objective-C, Java, Ruby, Coffeescript, Javascript, HTML, or shell scripts. Tony has spoken at RailsConf, AnDevCon, 360|Flex, and has created popular O'Reilly Android screencasts.

In his free time Tony enjoys playing the bass and Warr Guitar, and making electronic music. Tony lives outside Denver, CO with his wife Lori and sons Titus and Lincoln.

# Acknowledgments

As always this book would not exists without the fine people at Apress who can turn a bunch of Word documents into a book. Amazing! Specifically, Katie Sullivan did a great job keeping me on track. Tom Welsh helped track down those elusive Code Inline formatting issues. Tony Hillerson ran the code and insured I wasn't just making stuff up. Thank you all.

Chapter 1

# A Simple First Game

In this book you are going to learn a lot about working with iOS. The goal, of course, is to be able to build a game that runs on iOS. To do that, you must learn about a lot of different elements that a full game will incorporate, such as basic UI widgets, audio, complex touch input, Game Center, in-app purchases, and of course graphics. This book will explore these concepts and many others. Think of it as a guide to the building blocks that you will need to make a compelling game that is specific to iOS and Apple's mobile devices. All iOS applications have one thing in common—the application Xcode—so it makes sense to start with that.

In this first chapter, we are going to build a very simple game of Rock, Paper, Scissors. We will use the Storyboard feature of Xcode to create an application with two views and the navigation between them.

Included with this book are sample Xcode projects; all of the code examples are taken directly from these projects. In this way, you can follow along with each one in Xcode. I used version 4.5 of Xcode when creating the projects for this book. The project that accompanies this chapter is called Sample 1; you can easily build it for yourself by following the steps outlined in this chapter.

The project is a very simple game in which we use Storyboard to create two scenes. The first scene is the starting view, and the second scene is where the user can play the Rock, Paper, Scissors game. The second scene is where you will add a `UIView` and specify the class as `RockPaperScissorView`. The source code for the class `RockPaperScissorView` can be found in the project Sample 1.

We will walk through each of these steps, but first let's take a quick look at our game, shown in Figure 1-1.


***Figure 1-1.*** *The two views of our first game: Sample 1*

On the left of Figure 1-1 we see the starting view. It just has a simple title and a Play button. When the user clicks the Play button, he is transitioned to the second view, shown on the right of the figure. In this view, the user can play Rock, Paper, Scissors. If the user wishes to return to the starting view, or home screen, he can press the Back button. This simple game is composed of a Storyboard layout in Xcode and a custom class that implements the game.

Let's take a look at how I created this game and at some ways you can customize a project.

# Creating a Project in Xcode: Sample 1

Creating this game involves only a few steps, which we'll walk through as an introduction to Xcode.

Start by launching Xcode. From the File menu, select New Project. You will see a screen showing the types of projects you can create with Xcode (See Figure 1-2).

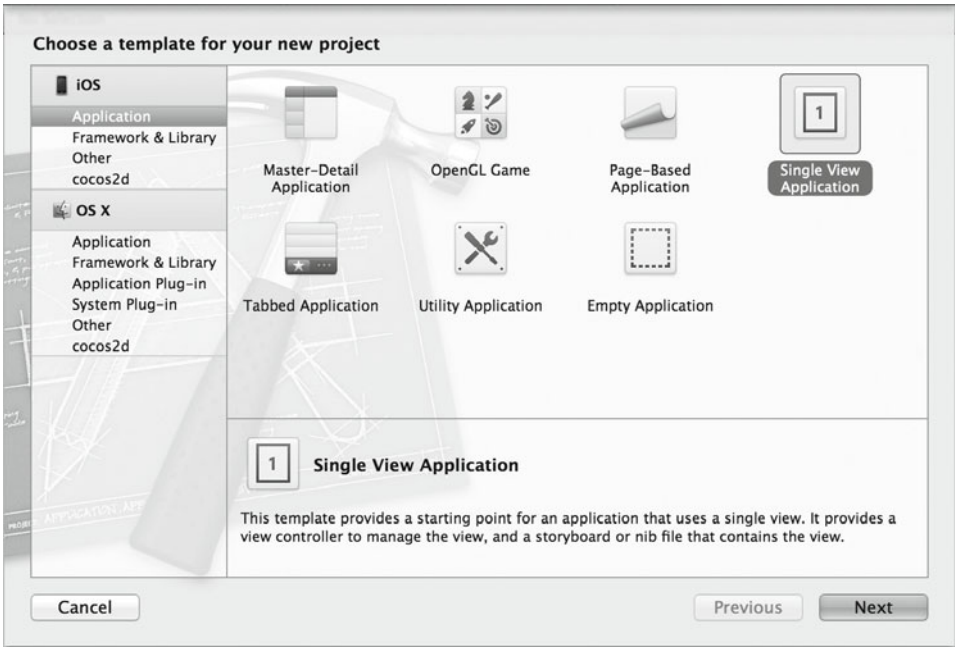



***Figure 1-2.*** *Project templates in Xcode*

For this project, select the template Single View Application. Click Next, and you will be prompted to name the project, as shown in Figure 1-3.

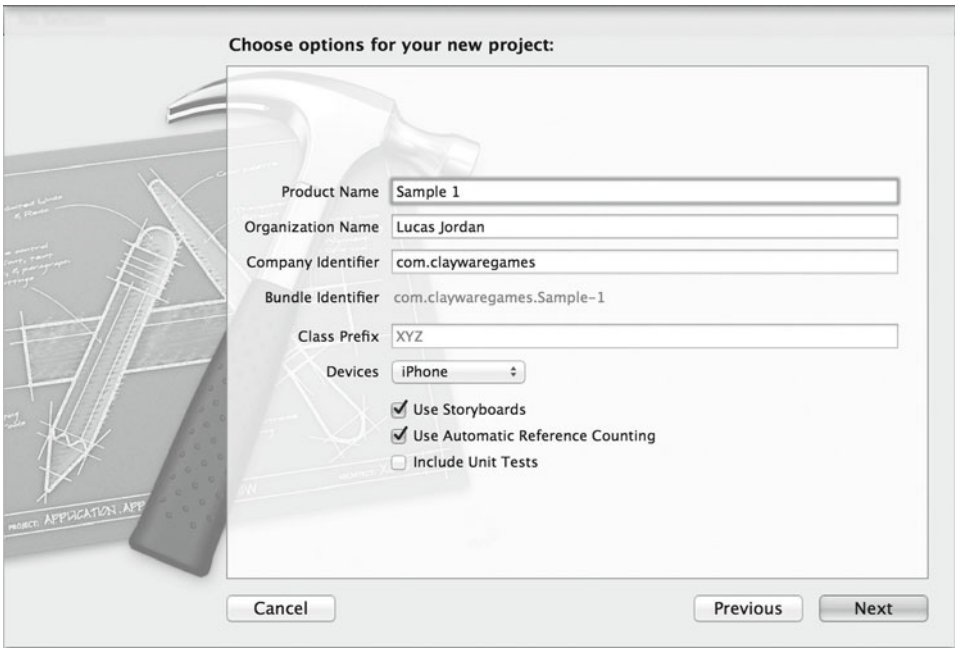



***Figure 1-3.*** *Naming an Xcode project*

Name your project whatever you want. The name you give your project will be the name of the root folder that contains it. You also want make sure Use Storyboard and Use Automatic Reference Counting are selected.

This time we will be making an application just for the iPhone, but from the Device Family pull-down menu you could also select iPad or Universal. After you click Next, you will be prompted to pick a place to save your project. The project can be saved anywhere on your computer.

Before moving on, let's take a moment to understand a little about how an Xcode project is organized.

# A Project's File Structure

After saving a new project, Xcode will create a single new folder within the folder you select. This single folder will contain the project. You can move this folder later if you want without affecting the project. Figure 1-4 shows the files created by Xcode.

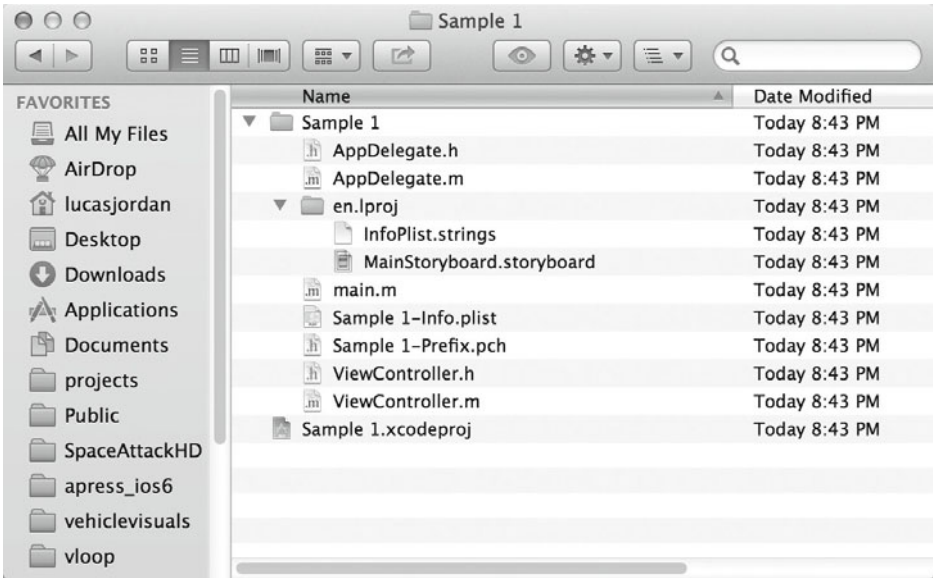



***Figure 1-4.*** *Files created by Xcode*

In Figure 1-4, we see a Finder window showing the file structure created. I selected that I wanted the project saved on my desktop, so Xcode created a root folder name `Sample 1` that contains the `Sample 1.xcodeproj` file. The `xcodeproj` file is the file that describes the project to Xcode, and all resources are by default relative to that file. Once you have saved your project, Xcode will open your new project automatically. Then you can start customizing it as you like.

# Customizing Your Project

We have looked at how to create a project. Now you are going to learn a little about working with Xcode to customize your project before moving on to adding a new UIView that implements the game.

## Arranging Xcode Views to Make Life Easier

Once you have a new project created, you can start customizing it. You should have Xcode open with your new project at this point. Go ahead and click the `MainStoryboard.storyboard` file found on the left so your project looks like Figure 1-5.

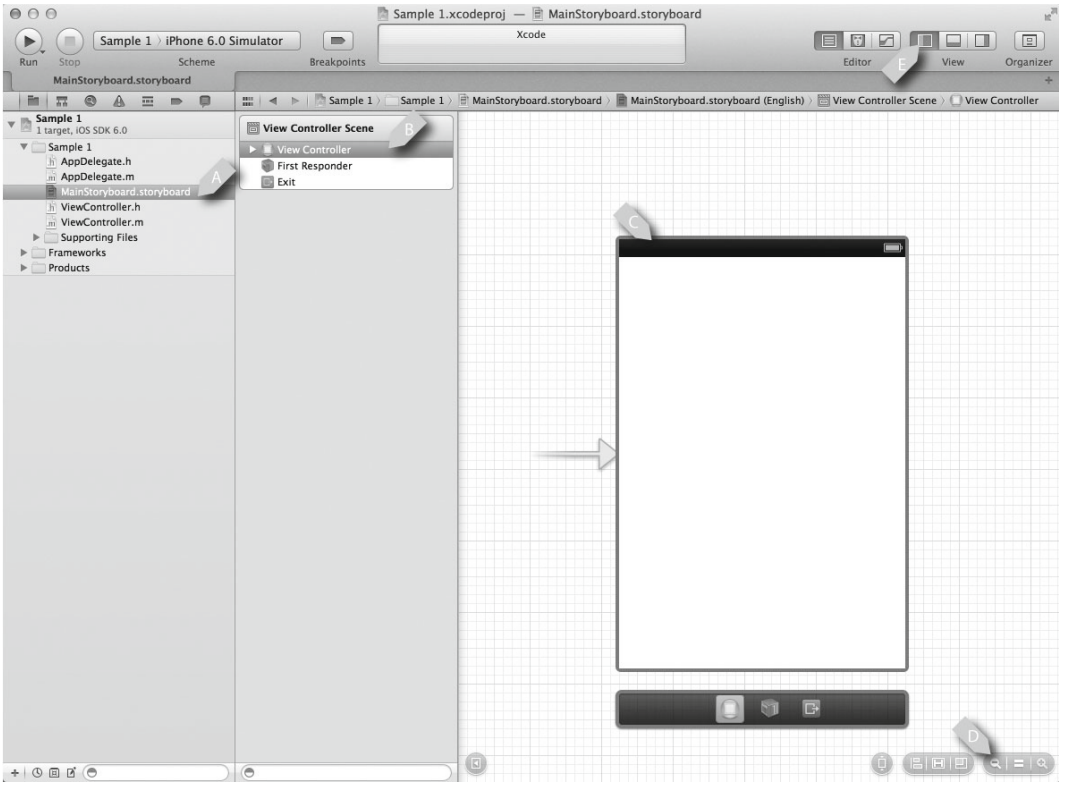



***Figure 1-5.*** *MainStoryboard.storyboard before customization*

In Figure 1-5, we see the file `MainStoryboard.storyboard` selected (item A). This file is used to describe multiple views and the navigation relationships between them. It shows the selected storyboard file and describes the content of the right side of the screen. In item B, we see an item called View Controller. This is the controller for the view described in item C. The items at D are used to zoom in and out of a storyboard view, and are critical to successfully navigating your way around. Additionally, the buttons in item E are used to control which of the main panels are visible in Xcode. Go ahead and play around with those buttons.

Next, let's look at how to add a new view.

# Adding a New View

Once you have had a chance to play a little with the different view setups available in Xcode, you can move on and add a new view to your project. Arrange Xcode so the right-most panel is visible, and hide the left-most panel if you want. Xcode should look something like Figure 1-6.

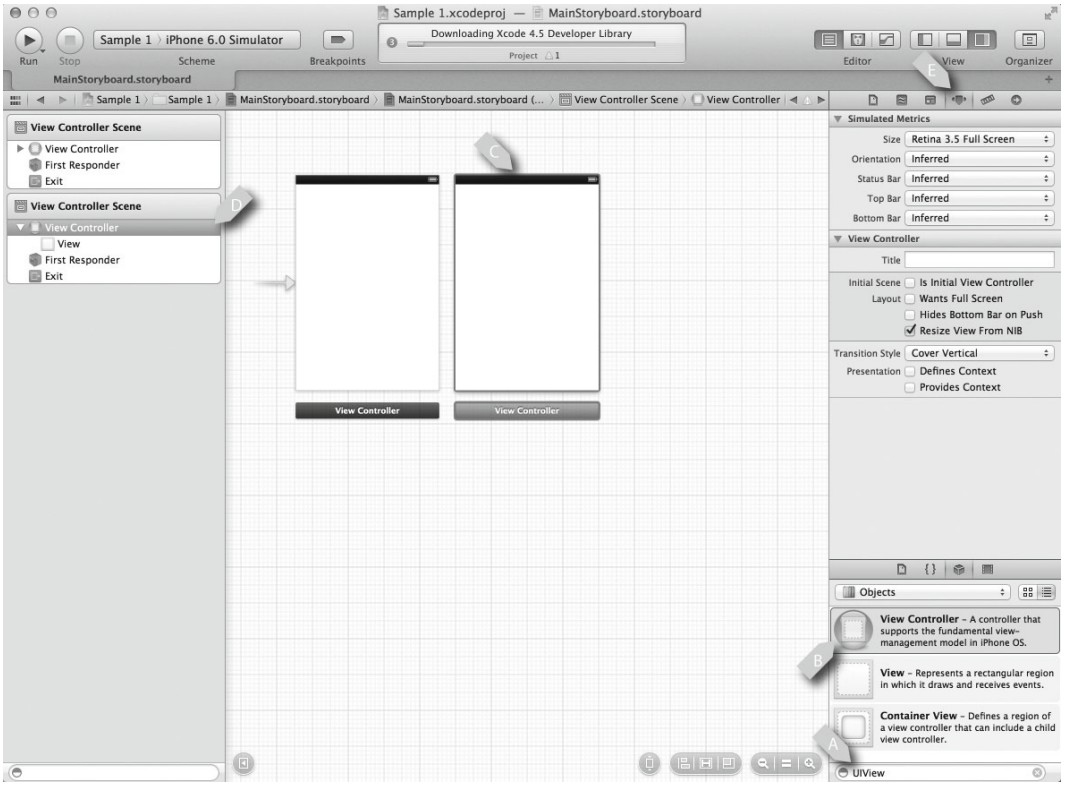



***Figure 1-6.** Storyboard with second view*

In Figure 1-6, we see that we have added a second view to the storyboard. Like any good Apple desktop application, most of the work is done by dragging and dropping. To add the second view, we enter the word “UIView” into the bottom-right text field, at item A. This filters the list so we can drag the icon labeled item B on the work area in the center. Click on the new view so it is selected (see item C), which we can see correlates to the selected icon in item D. Item E shows the properties for the selected item.

Now that we have a new view in the project, we want to set up a way to navigate between our views.

# Simple Navigation

We now want to create some buttons that enable us to navigate from one view to the other. The first step is to add the buttons, and the second is to configure the navigation. Figure 1-7 shows these views being wired up for navigation.

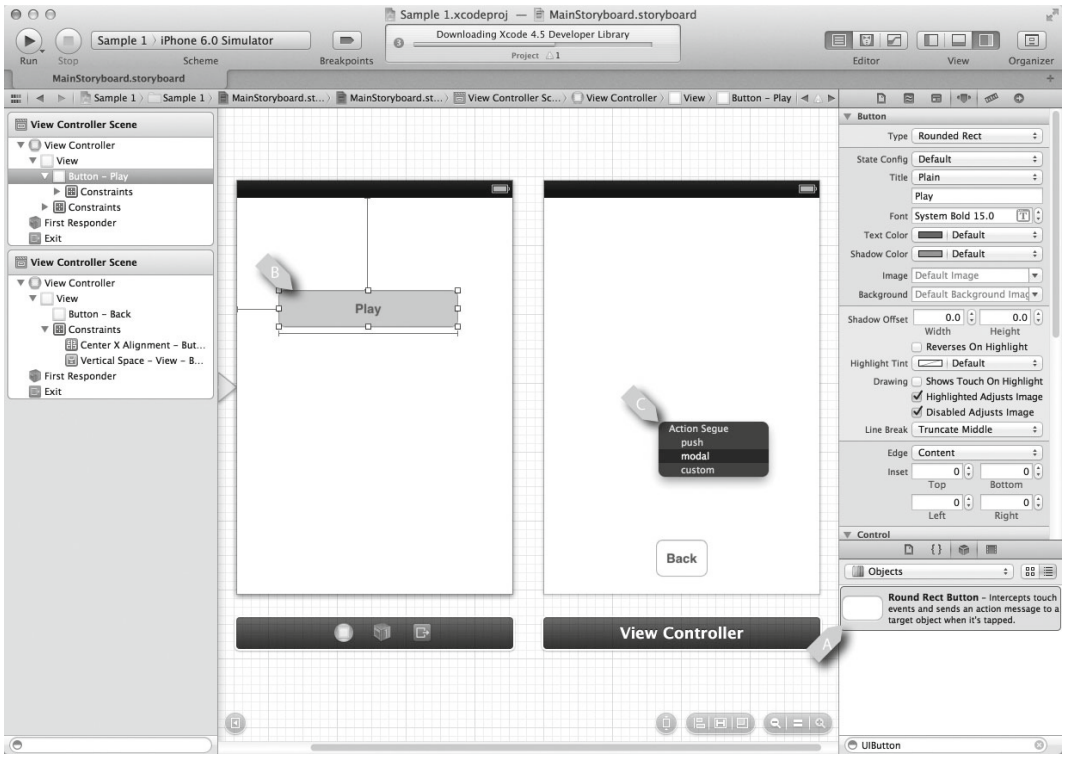



*Figure 1-7. Storyboard with navigation*

In Figure 1-7, we see that we have dragged a `UIButton` from the library item A onto each of the views. We gave the `UIButton` on the left the label Play, and the `UIButton` on the right the label Back. To make the Play button navigate to the view on the right, we right-drag from the Play button (item B) to the view on the right and release at item C. When we do this, a context dialog pops up, allowing us to select which type of transition we want. I selected Model. We can repeat the process for the Back button: right-drag it to the view on the left and select the transition you want for the return trip. You can run the application at this point and navigate between these two views. In order to make it a game, though, we need to include the Rock, Paper, Scissors view and buttons.

# Adding the Rock, Paper, Scissors View

To add the Rock, Paper, Scissors view, we need to include a class from the sample code in the project you are building. The easiest way to do this is to open the sample project and drag the

files `RockPaperScissorsView.h` and `RockPaperScissorsView.m` from the sample project to your new project. Figure 1-8 shows the dialog that pops up when you drag files into an Xcode project.

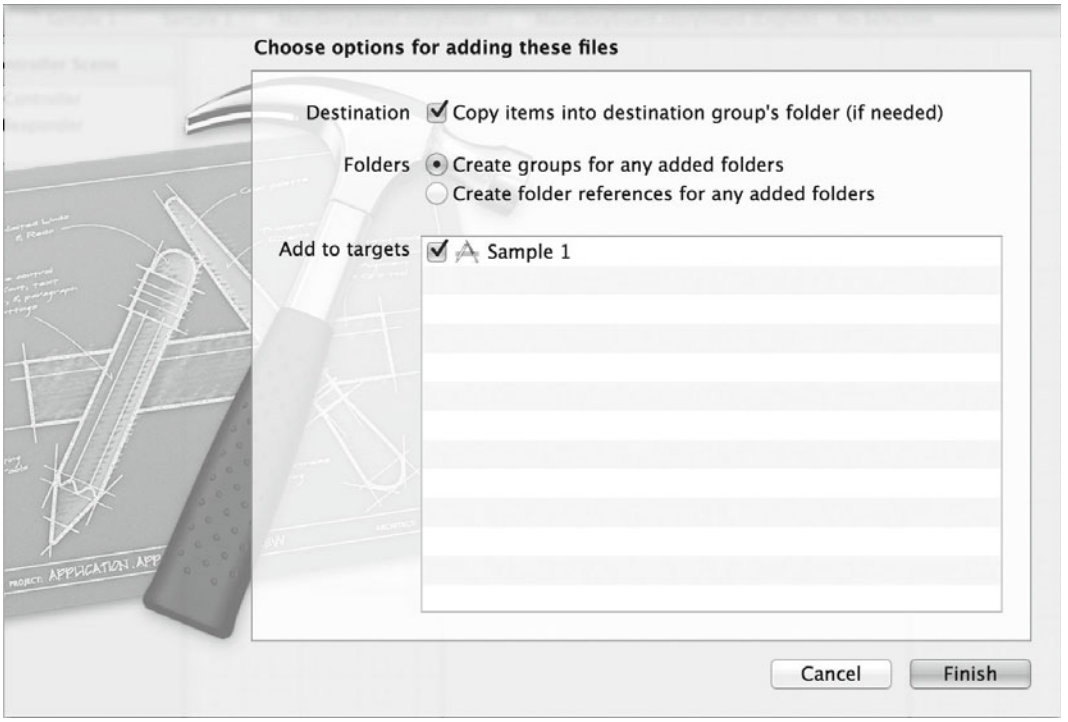



***Figure 1-8.** Dragging files into an Xcode project*

In Figure 1-8, we see the dialog confirming that we want to drag new files into an Xcode project. Be sure the Destination box is checked. Otherwise, Xcode will not copy the files to the location of the target project. It is good practice to keep all project resources in the root folder of a project. Xcode is flexible enough to not require that you do this, but I have been burned too many times by this flexibility. Anyway, now that we have the required class in our project, let's wire up our interface to include it.

## Customizing a UIView

The last step in preparing a simple application is to create a new `UIView` in our interface that is of the class `RockPaperScissorsView`. Figure 1-9 shows how this is done.

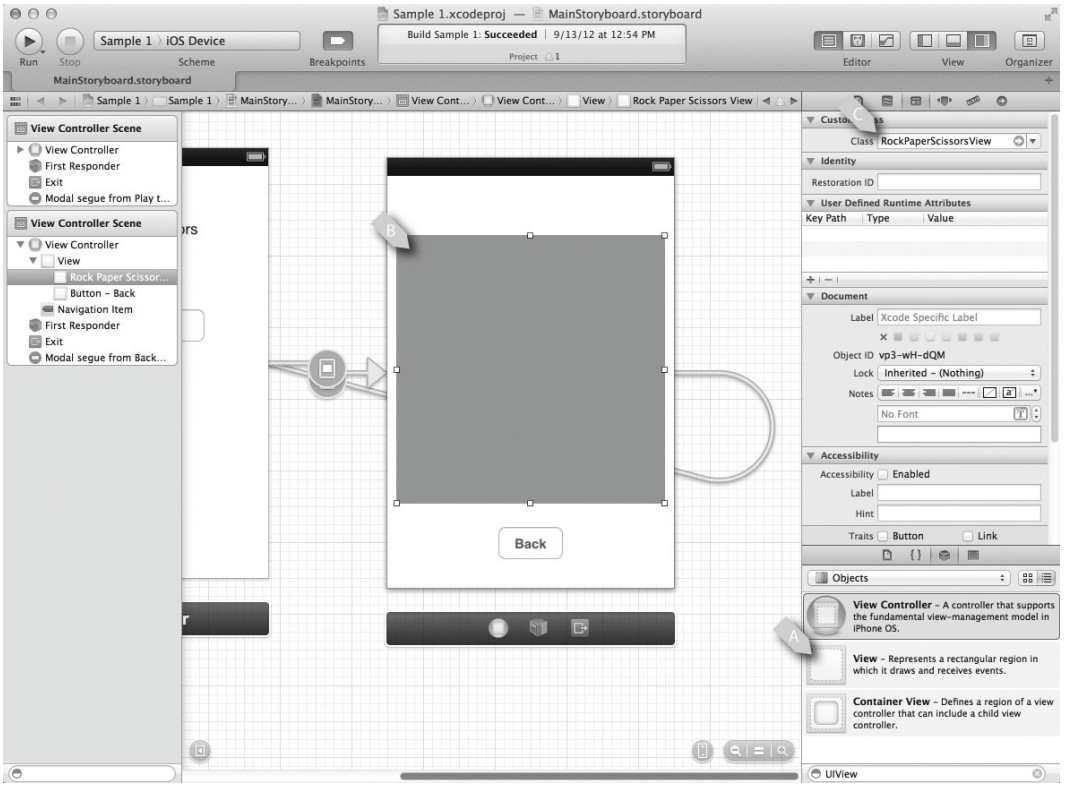



***Figure 1-9.** A customized UIView*

In Figure 1-9, we see a `UIView` added to the view on the right. We did this by dragging the icon from item A onto the storyboard in item B. After adjusting the size of the new `UIView`, we set its class to be `RockPaperScissorsView`, as shown in item C. At this point, we are technically done. We have created our first game! Obviously, we have not looked at the implementation of `RockPaperScissorsView`, which is discussed on the next chapter.

The rest of this book will use Sample 1 as a starting place. You will learn many new techniques for customizing a simple app to make a truly complete game.

# Summary

In this chapter, we have taken a quick tour through Xcode, learning how to create a project with it and build a simple navigation using Storyboard. The chapters that follow will add to the basic lessons given here to show you how to build a complete game.

Chapter 2

# Setting up Your Game Project

Like all software projects, iOS game development benefits from starting on good footing. In this chapter, we will discuss setting up a new Xcode project that is a suitable starting point for many games. This will include creating a project that can be used for the deployment on the iPhone and the iPad, handling both landscape and portrait orientations.

We look at how an iOS application is initialized and where we can start customizing behavior to match our expectations of how the application should perform. We will also explore how user interface (UI) elements are created and modified in an iOS application, paying special attention to managing different devices and orientations.

The game we create in this chapter will be very much like the simple example from Chapter 1—in fact, it will play exactly the same. But we will be building a foundation for future chapters while practicing some key techniques, such as working with `UIViewControllers` and Interface Builder.

We will explore how an iOS application is put together, and explain the key classes. We'll also create new UI elements and learn how to customize them with Interface Builder, and we will explore using the MVC pattern to create flexible, reusable code elements. At the end of this chapter, we will have created the Rock, Paper, Scissors application shown in Figure 2-1.

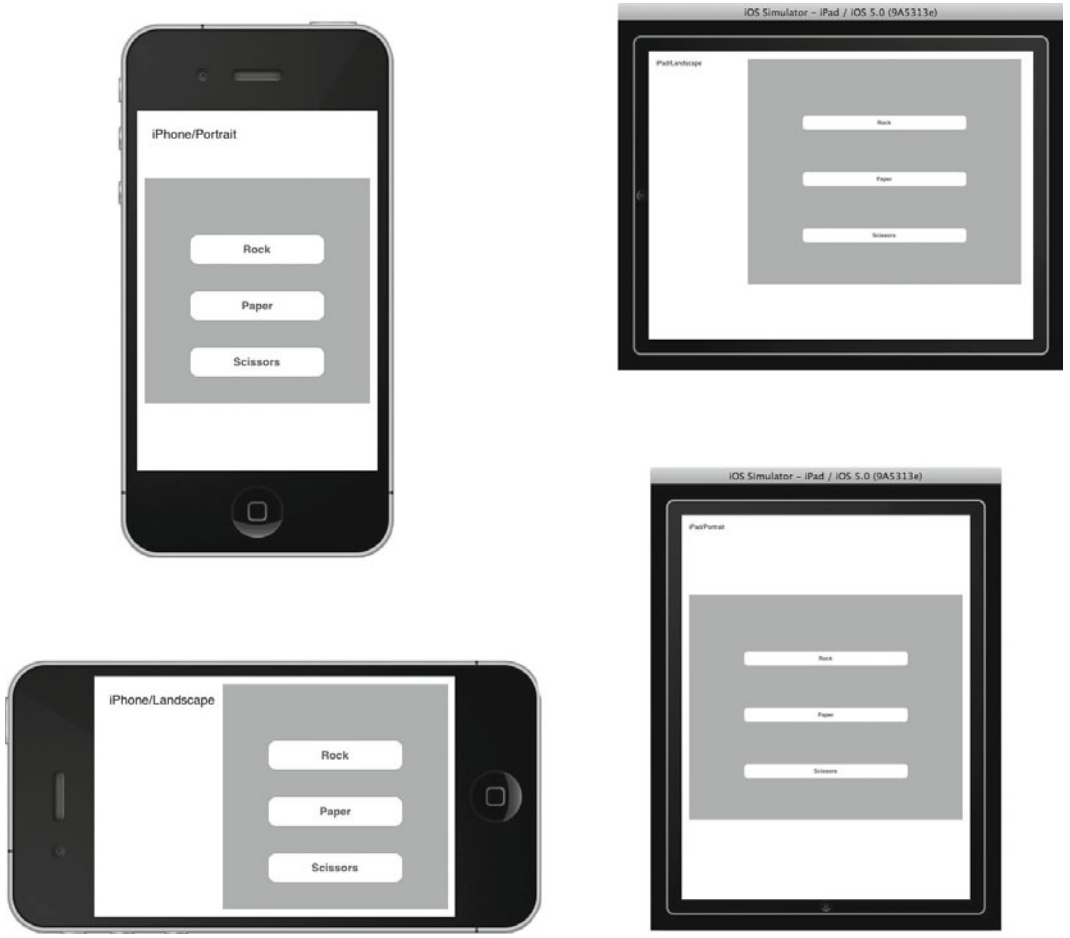



***Figure 2-1.*** *An application design to work on the iPhone and iPad in all orientations*

Figure 2-1 shows the application running on an iPhone and iPad simulator. This is a so-called universal application: it can run on both devices and would be presented in the App Store as such. Unless there are specific business reasons for writing an app that only works on the iPhone or the iPad, it makes a lot of sense to make your app universal. It will save you time down the road, even if you only intend to release your app on one of the devices to start.

Our sample application is so simple that it may be difficult to see the differences between the four states presented in Figure 2-1. On the upper left, where the iPhone is in portrait orientation, the position of the gray area is laid out differently from the iPhone in landscape at the bottom left. The layout of the text is also different. The same goes for the application when it is running on the iPad in landscape vs. portrait. Let's get going and understand how we can set up a project to accommodate these different devices and orientations.

# Creating Your Game Project

To get things started with our sample game, we first have to create a new project in Xcode.

Create a new project by selecting File ➤ New ➤ New Project…. This will open a wizard that allows you to select which type of project you want, as shown in Figure 2-2.

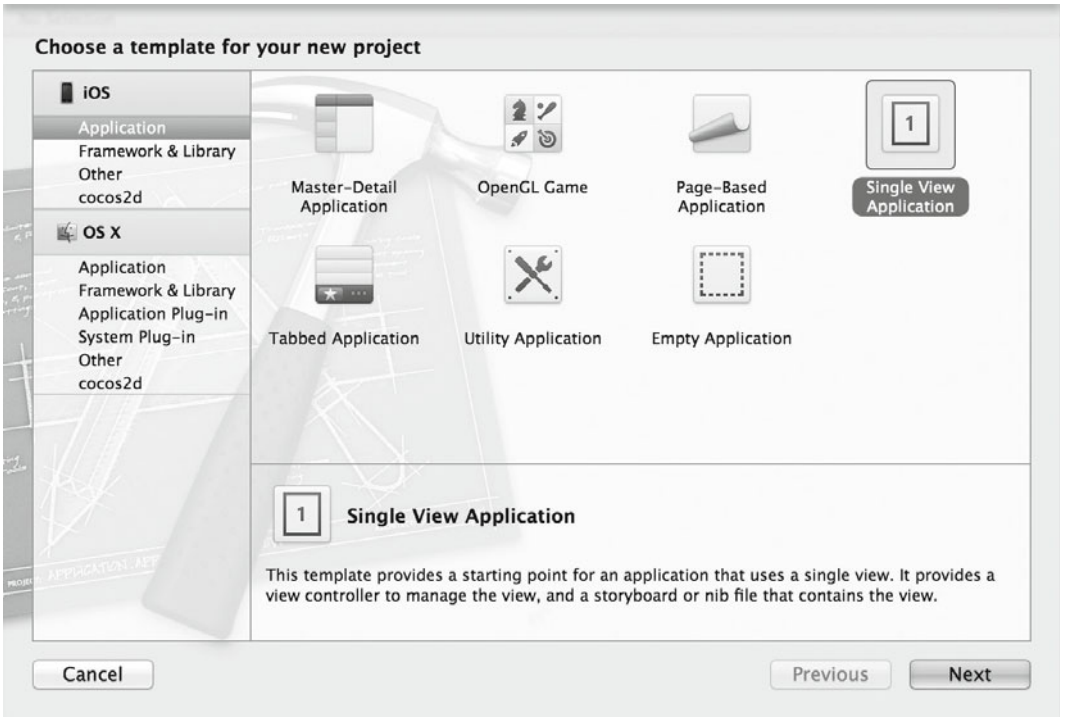



***Figure 2-2.*** *Creating a new Single View Application*

On the left side of Figure 2-2, we have selected Application from the iOS section. On the right are the available project types for creating iOS applications. The choices presented here help developers by giving them a reasonable starting place for their applications. This is particularly helpful for developers new to iOS, because these templates get you started for a number of common application navigation styles. We are going to pick a Single View Application, because we require only a very minimal starting point, and the Single View Application provides good support for universal applications. After clicking Next, we see the options shown in Figure 2-3.

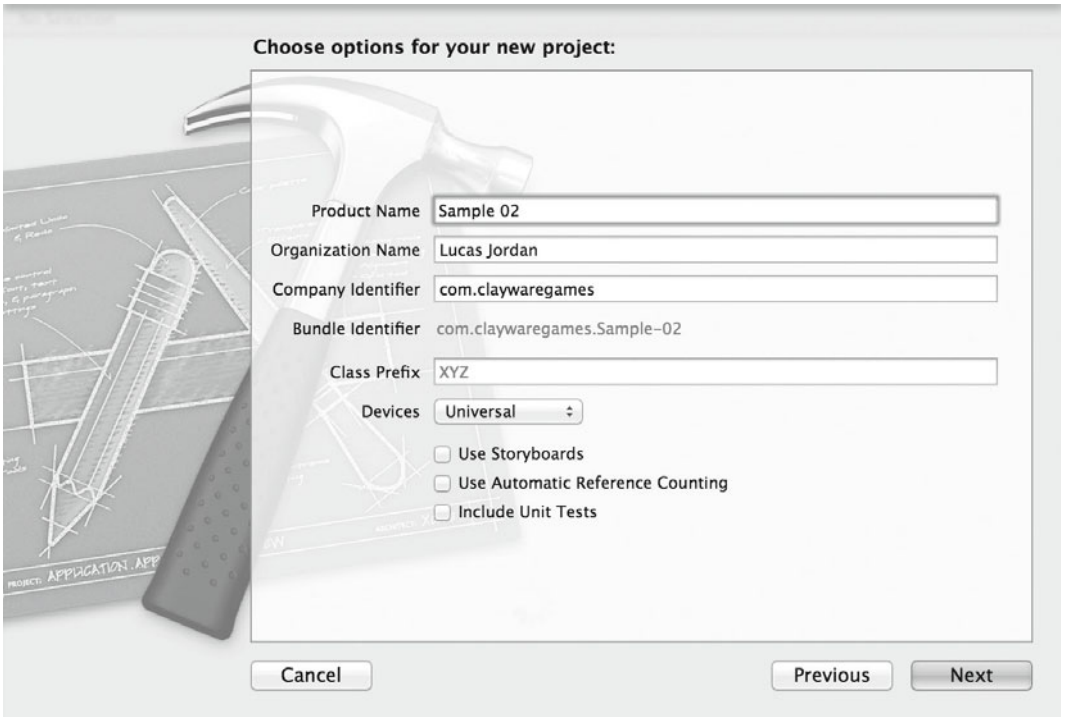



***Figure 2-3.*** *Details for the new project*

The first thing we will do is name our product. You can pick anything you want. The company identifier will be used during the app submission process to identify it. You can put any value you want as the company identifier, but it is common practice to use a reverse domain name. As can be seen in Figure 2-3, the bundle identifier is a composite of the product name and the company identifier. The bundle identifier can be changed later—the wizard is simply showing what the default will be. When you submit your game to the App Store, the bundle identifier is used to indicate which application you are uploading.s

By selecting Universal from the Device list, you are telling Xcode to create a project that is ready to run on both the iPhone and the iPad. For this example, we will not be using Storyboard or Automatic Reference Counting. Similarly, we won't be creating any unit test, so the Include Unit Tests option should be unchecked as well. Clicking Next prompts you to save the project. Xcode will create a new folder in the selected directory, so you don't have to manually create a folder if you don't want to. When the new project is saved, you will see something similar to Figure 2-4.

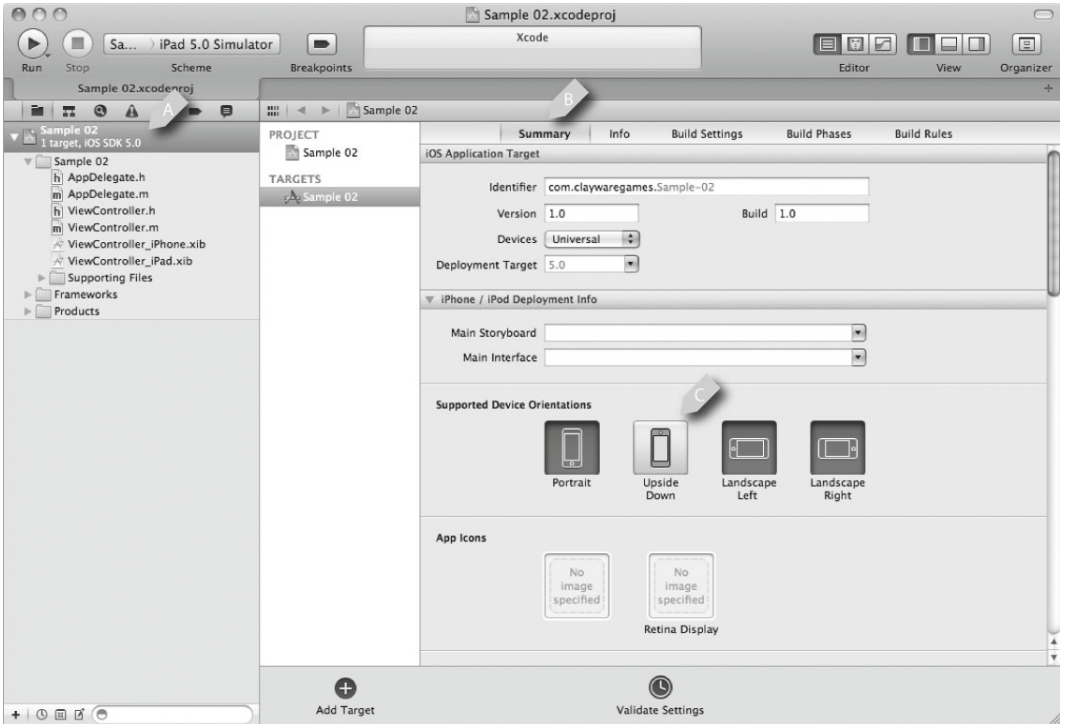



*Figure 2-4. A newly created project*

On the left side of Figure 2-4, there is a tree containing the elements of the project with the root-most element selected (A). On the right, the Summary tab is selected (B). From the Summary tab, we want to select the supported device orientations (C). To support both orientations on each device, click the Upside Down button. Scroll down and make sure that all of the orientations for the iPad are depressed as well. Figure 2-5 shows the correct settings. Now that the project is created, it is time to start customizing it to fit our needs.


*Figure 2-5. Supporting all device orientations*