Mobile gaming is one of the fastest-growing industries, and compatibility with iOS is now becoming the norm for game developers. SpriteKit is part of the native SDK from Apple, and enables developers to make a simple entry into game development without unnecessary overheads and a long learning process. SpriteKit also provides other functionality that is useful for games, including basic sound playback support and physics simulation. In addition, Xcode provides built-in support for SpriteKit so that you can create complex special effects and texture atlases directly in Xcode.

Become a master of iOS game development through this fun and fast-paced guide. In the beginning, we’ll tell you everything you need to plan and design your game. You’ll then start developing your game through step-by-step instructions using the various built-in technologies of Xcode. From then on, we discuss how to deploy your game to the iOS App Store, as well as monetizing it to make more revenue. You will also learn advanced techniques to improve the game-playing experience, including better multitasking, improved performance optimization, battery management, and more. To end the book, we’ll show you how to update your game with different features, then how to port the updates to the App Store.

Who this book is written for

This book is for those who have created an iOS game already and want to hone their skills. A reasonable level of knowledge and an understanding of the core elements and applications would be helpful.

What you will learn from this book

- Blast off and start developing fun games
- Optimize assets for the best performance of the game app
- Master player movements using touch controls and motion controls
- Perform bug squashing by testing and debugging
- Design various interesting characters and enemies for your game
- Test your game on both the simulator on the computer and on real devices
- Tweak and fine tune the code and graphics before the release of your app
- Uncover some awesome new gameplay by adding multiplayer elements to get more people playing together

Mastering iOS Game Development

Master the advanced concepts of game development for iOS to build impressive games

Miguel DeQuadros

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'What to Expect in Xcode and Game Development'
- A synopsis of the book’s content
- More information on Mastering iOS Game Development
Miguel DeQuadros is a game developer and the founder of the independent development studio, Wurd Industries, based in Ontario, Canada. He has been developing iPhone games since the release of the App Store back at the exciting release of iOS 2.0. Since then, he has released 10 games and 1 entertainment app world-wide on the App Store with more to come from Wurd Industries.

He was originally interested in 3D animation and graphical design, which he focused on mainly in 2004. But, he then got the game development bug and has been developing iPhone apps since 2008, which also allows him to use his creativity and knowledge of 3D animation for cut scenes and videos within his apps, and he is loving every minute of it. Starting from his first project, Toy Tennis, back in 2008, down to his current project, SpaceRoads, for PC, Mac, Wii U, and other platforms, he continues to develop high-quality apps and games alike. Moving away from simple game development tools, he now primarily uses Unity3D, 3D Studio Max, and the Unreal Engine for his current project in an aim to create very high-quality games.

His games can be seen on the App Store on iOS, Steam Greenlight, Amazon, and IndieCity, and of course on his website, www.wurdindustries.com. His games have been reviewed on YouTube by Action Soup Studios, and you can also find his interviews there.
Preface

Welcome! We all love game development and many of us dream of games we want to play, but alas, we never see them released. You are here to make that game, the one you've been planning, drawing, and dying to create for a long time. With the help of this book and with your awesome ideas, you will succeed in creating your dream game. In this book, you will learn how to use the free software available to you to create, develop, and release your awesome creations.

Welcome and enjoy the ride!

What this book covers

Chapter 1, What to Expect in Xcode and Game Development, covers how to become an Apple developer, how to download Xcode and the iOS development kit, and what to expect of the game development industry.

Chapter 2, Creating the Stuff, demonstrates how to use the various tools available to you to create sprites and various other images, sound effects, and even videos for your game.

Chapter 3, Blast Off! Starting with Development, talks about beginning our project. You will learn how to import various frameworks for our project and create our character and get him moving.

Chapter 4, Let's Keep Going! Adding More Functionality, discusses how to add some cool effects, such as particles for flames and rain, to our game now that our project is coming together.

Chapter 5, Bug Squashing – Testing and Debugging, covers testing our game and debugging it to ensure that there are no issues in it once we port our game.
Preface

Chapter 6, *Making Our Game More Efficient*, shows how we can make our game run better on all devices now that it is running flawlessly. By doing this, we will increase battery life, performance, and the overall user experience.

Chapter 7, *Deploying and Monetizing*, discusses how to show our game to the world and increase our revenue outlets by adding monetizing options.

Chapter 8, *It’s Too Dangerous to Go Alone, Take a Friend!*, talks about making our game multiplayer after much planning, as we are now ready to update our game. We will incorporate multiplayer connectivity to find friends to play with!
Welcome to the world of mobile game development! Whether you are a seasoned developer or a fresh new developer, you are in an exciting fast-paced industry. You have purchased this book expecting to master iOS game development, and that's exactly what you are going to do by the end of this book! However, you need to start somewhere, right? Let's see what will discuss in this chapter:

- Signing up to be an iOS developer
- Setting up your Mac for development
- What to expect from the game development market

The mobile game development industry is so exciting! Get ready to take your first steps into game development. We will get you all signed up to be an Apple Developer, set up your Mac with the development environment, and take a tour of some of the features of Xcode and we are going to see what source files (the files that contain our code) look like. Then, we will discuss the game development market and how AMAZING it isn't. I'm kidding, it is a fun market!

Let's dive right into it, shall we?
Registering to be an iOS developer
As with many platforms (Mac, PC, Android, Blackberry, and Windows Mobile), iOS requires you to sign up to be a developer. Don't worry! It's super easy, simply go to www.developer.apple.com, where you will see the home page, as shown in the following screenshot:

Scroll down to find iOS Apps. Click on the iOS Apps button, and you will be greeted with the iOS Dev Center. Get used to this site as well as iTunes Connect, which we will cover later in this book — you will be using these sites a lot during your game development.
If you don't have a developer account, click on the register for free text so that you can get signed up for a new Apple ID. Another great thing about Apple is that their accounts are all linked, so when you register for a developer account, you can use your Apple ID. On the next screen, you will be asked to either sign in with your Apple ID or create a new one. Simply follow the prompts, fill in all your information, and pay the annual fee ($99 USD/ $119 CAD). When you are done, you should have access to the SDK. If you aren't redirected to the iOS Dev Center again, go to www.developer.apple.com/ios and scroll down to the Downloads section. As of the time of writing this book, the current version of Xcode is 6.2. Click on the Download button. You will be taken to a download screen; however, you will be ported over to the Mac AppStore. Xcode is now hosted on the Mac AppStore, which is a lot better because it automatically updates Xcode when needed. You will see the Xcode installation page, as shown in the following screenshot:

Click on the Get button to download Xcode. When it's done downloading (it will probably take a while unless you have fibre optic extreme Internet: It's 2.5 GB. The AppStore will automatically install Xcode for you.
Now, you should have a glittery new icon on your dock. When you open it, it may prompt you to do additional installation to support debugging or what not, go ahead and let it download any extra components needed, as shown in the following screenshot:

You will now be greeted with the Xcode welcome screen. From here, you can create new projects, open old ones, and even see a list of recent projects.

We will start setting up our game project, so click on Create a new Xcode project, and then the new project wizard will pop up. Here, you have a lot of empty project templates to choose from, but for this project, we will select the Game template under the iOS selection, under Application. Click on Game, and then click on Next. The next screen will ask for all the game information, which will be broken down like this:

- **Product Name**: This is the name of the project or game.
- **Organization Name**: This is the name of the company or your personal name that you used when signing up with the Apple Developer site.
- **Organization Identifier**: This will show up in your bundle identifier (certificates will be discussed later), usually certificates read `com.yourcompany.yourproduct`, but you can customize it to be `yourcompany.yourproduct`. 
Bundle Identifier: This is the identifier for your project. You will see this on your developer certificates, which we will install later, and you will see it when uploading to the AppStore as well.

Language: There are now two languages that we can choose from when developing: Swift and Objective-C. For this book, we will use Objective-C.

Game Technology: This includes the various "kits" that you can use when developing such as SceneKit, SpriteKit, OpenGL ES, and Metal. For this project, we will select SpriteKit.

Devices: The default device is Universal; however, if you want to select your target device, you can do so. Sometimes, an iPad app won't look right on iPhone due to the smaller screen size, things won't fit properly as the resolution is different, and won't display properly.

Fill in all the fields of information as shown in the following screenshot:

Choose options for your new project:

<table>
<thead>
<tr>
<th>Product Name:</th>
<th>SuperSolderBoys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization Name:</td>
<td>Miguel DeQuadros</td>
</tr>
<tr>
<td>Organization Identifier:</td>
<td>WurdIndustries</td>
</tr>
<tr>
<td>Bundle Identifier:</td>
<td>WurdIndustries.SuperSolderBoys</td>
</tr>
<tr>
<td>Language:</td>
<td>Objective-C</td>
</tr>
<tr>
<td>Game Technology:</td>
<td>SpriteKit</td>
</tr>
<tr>
<td>Devices:</td>
<td>Universal</td>
</tr>
</tbody>
</table>

To recap, for this project, we will select **Objective-C**, **SpriteKit**, and **Universal**. When you have filled in all the information, click on the **Next** button, and save your project in a location that's easy to remember.

Now you are in Xcode! Confused by what you see? Don't worry, it's actually pretty easy to navigate.
The following screenshot shows the general project settings, which is the first thing you see when you load an Xcode project:

For this project, we only want to support the landscape orientations; so, from the Deployment Info section under Device Orientation, deselect the Portrait option, and leave the two landscape orientations checked.

If you want, you can add in different capabilities for our game. Click on the Capabilities button on the topmost bar in the center of the screen, where you will see a ton of different options that you can add to your app. For example, we can add Game Center for leader boards and achievements, or we can add iCloud functionality by clicking the button from off to on. We can use iCloud to store game save data remotely, as shown in the following screenshot:
There are a lot of settings and variables that can be changed in the various sections of the topmost bar. We will get into some of them later on in this book.

Let's take a look around our project files to familiarize ourselves with the functions of the files. On the left-hand side bar, you will see .h and .m files, these are where all your programming goes.

The .h files are your header files, we will declare variables (such as integers and Booleans) and outlets (if we were building through the storyboard, we would declare buttons, labels, and a lot more. We would declare them in the header file, and connect them in the storyboard).

Don't worry, we are going to go into a lot more detail later on in this book.

The .m file is your main file where the majority of the coding goes. Declarations made in the header files are accessible as long as it's the same set of files, for example, you can access an integer from the AppDelegate.h file in the AppDelegate.m file. There are ways to access variables from other files, such as frameworks; or as long as you import the header files into the files you are working on. Again these are things we will discuss later. I don't want to confuse you at this point.

Let's look at the AppDelegate.h and AppDelegate.m files, starting with the .h file where you will see the following text:

```c
//
//AppDelegate.h
//SuperSolderBoys
//
//Created by Miguel DeQuadros on 2015-03-25.
//Copyright (c) 2015 Miguel DeQuadros. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface AppDelegate : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@end
```

There will be variations in the text that is commented (you can comment text out by typing // following the comment). You will know text is commented out when it is green in color.
Almost all code files will begin with importing files. The AppDelegate class seen in the preceding code shows #import <UIKit/UIKit.h>, which will allow you to access commands and functions from the UIKit headers. You can import any other header files here, for example, if you want to create an app with an option to make payments via PayPal, you can import the PayPal SDK, then import it into your file by typing #import <PayPal/PayPal.h> or whatever the file is that needs to be imported. When you start typing, Xcode will display a list of files that you can import.

We will now look at the next line, that is, @interface AppDelegate : UIResponder <UIApplicationDelegate>. This will vary depending on the type of interface you are working with (whether you are working with the app delegate, a view controller for our main menu, or on a PayPal payment view controller); it can be an AppDelegate method in this file, or it can be a UIViewController or SKScene method. Again, it depends on the type of interface you are working with. You will see variations of different interfaces throughout this book.

Finally, we will see the last highlighted line, that is, @property (strong, nonatomic) UIWindow *window;. This is where your declarations will go. This line declares the app window. Properties are declarations of various items we use within our classes. For example, a property can be an integer, a Boolean, a window (as mentioned earlier), or even a button.

Finally, we have the @end line, which is simply the end of the file.

Now, let's look at the .m file:

```
//
//  AppDelegate.m
//  SuperSolderBoys
//
//  Created by Miguel DeQuadros on 2015-03-25.
//  Copyright (c) 2015 Miguel DeQuadros. All rights reserved.
//

#import "AppDelegate.h"

@interface AppDelegate ()
@end

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
```
// Override point for customization after application launch.
return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to
    inactive state. This can occur for certain types of temporary
    interruptions (such as an incoming phone call or SMS message) or when
    the user quits the application and it begins the transition to the
    background state.
}

- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data,
    invalidate timers, and store enough application state information to
    restore your application to its current state in case it is terminated
    later.
    // If your application supports background execution, this method
    is called instead of applicationWillTerminate: when the user quits.
}

- (void)applicationWillEnterForeground:(UIApplication *)application {
    // Called as part of the transition from the background to the
    inactive state; here you can undo many of the changes made on entering
    the background.
}

- (void)applicationDidBecomeActive:(UIApplication *)application {
    // Restart any tasks that were paused (or not yet started) while
    the application was inactive. If the application was previously in the
    background, optionally refresh the user interface.
}

- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if
    appropriate. See also applicationWillEnterBackground:.
}
@end

In this file, you will call various functions when the application reaches certain
states. If you read through some of the functions, you will see commented out text
(to comment out text type // before the line of code) on what each state does, like
when the application is terminated, or when it enters the background. With these
functions, you can save data, log information, or even pause the game.
What to Expect in Xcode and Game Development

Seems fairly simple, right? Once you get the hang of things, it really is!

We will now have to add our developer account to the Xcode accounts preferences. This makes signing certificates and provisioning profiles much easier.

To add your account, click on Xcode in the topmost bar (where you see file, edit and all that fun stuff), then click on Preferences. Alternatively, you can press CMD +, for quick access to the menu.

When the preferences window appears, click on the Accounts tab, which is the second tab in the window.

In the lower-left corner of the window, click on the + button to add an account. When the drop-down menu appears, select Add Apple ID.

Again, Xcode uses your normal Apple ID login to sign in to the Developers' Member Center and iTunes Connect, which is super convenient because you don't have to remember multiple logins.
Now that we have added our account to Xcode, we need to add all the developer certificates into our key chain.

Go to developer.apple.com, click on Member Center, and log in using your Apple ID. Under the Technical Resources and Tools section, click on the Certificates, Identifiers & Profiles icon.

We will now see a selection for iOS, Mac, and Safari. These will display all the certificates for each platform, organized nice and neatly.

Click on the Certificates tab under iOS, and if your membership is new, you will be greeted with a Getting Started webpage. Simply follow all the steps, which will guide you into the creation and installation of the certificates. More of this will be discussed later on in this book.

When you have all your certificates installed, you are all done! Now that we have Xcode installed and set up, let’s talk about asset creation.

**Asset creation for your awesome game**

There are literally a plethora of programs that you can use to create the assets for your game. Wait! What's an asset?! Simply put, assets are the various images, sound effects, or videos you use in your game. For example, we have sprites. Wait! What are sprites?! Sprites are images you use in your game; this can include characters, platforms, backgrounds, items, and objects. Assets can also be sound effects, music, and videos.

For sprites, I like to use Adobe Photoshop, especially if the game I'm making is a pixel art 2D game. It's excellent because you can make use of the layers for each individual body part that needs to be moved around for the animations. However, if I'm making something that I would want to look a little more realistic, I'll use 3D Studio Max to create my models then render them into a small image file.

These software suites are not cheap. 3DS Max is over $3,000, and the Adobe Suite used to be about $1,600, though they now offer monthly plans.

If you're looking for a free option, check out programs such as Gimp for your image creation. I've used it: it's laid out just like Photoshop, and it is just as easy and powerful to use. Alternatively, use Pixlr, which is a browser-based image manipulation program exactly like Photoshop.

If you want to do some cool 3D stuff, check out Blender, Wings3D, and DAZ Studio.

For music, many like to use good ol' Garage Band. There are lots of instrument options, premade loops, and the ability to use your computer keyboard as a musical keyboard.
For audio manipulation, the main choice between developers and others alike is Audacity. You can manipulate audio files by adding fades, clicks, silence, reversing audio, adding, and a lot more. It’s pretty awesome.

For videos, I personally like to use 3DS Max, then import into iMovie. The iMovie app is great because it can export your video files for use with an iPhone, so we need not worry about the compatibility of the file, or about screen size. You can use Blender to create some awesome 3D animations in a short time for your introductions, or you can animate them using Adobe Flash, which can also export them to specific formats for iPhone and iPad.

We will talk more about the actual creation of assets in the next chapter. In the meantime, let’s talk about the game development scene, and what exactly to expect when you get into it.

**Entering the game development market**

As I mentioned in the introduction of this chapter, you have entered a super exciting market. The game development market is fast paced, always changing, and super fun with a huge potential to make a lot of money as Doge will explain:
However, while it is a fun business with huge potential to make money, it's not all fun and games and instant success.

In the ten and more years that I've been developing games, I have learned that what you think will sell, may not sell. One of the first games I made was called iMMUNE and I thought it was quite decent when I was programming it. After much play testing, and getting people's opinions, I felt the game was ready to release, and I hoped for much success, as we all do.

The day of release came and passed, and I saw little or no sales. Naturally, I was pretty bummed, but now I realize I made many mistakes when developing and releasing the game. Let me explain what happened—I got the idea from my grandmother, who (as most grandmothers do) takes a lot of medication for her ailing body. One day, we were discussing the whole swine flu epidemic, and my late teenage brain automatically thought, "Hey! All of us take pills, and this swine flu thing is pretty crazy right now. Why not cash in on that?" I wasn't being uncaring of the state of the world's health, I just thought that, with the current situation, people would be searching for swine flu or pills at the time and my game would be presented to a lot of people.

So, I came up with an idea for iMMUNE. It was simple, you play a pill going throughout the human body and you have to fight the viruses, culminating in having to destroy the swine flu viruses at the end of the game. Here is a screenshot of iMMUNE:
I thought it was a great idea, and so did those around me. So, after many months of rushing to get the game finished, and admittedly, leaving some major features out, such as a main menu with a help button, the first mistake I made was that I uploaded the game to the AppStore. I began to publicize the game, create hype with screenshots and cut scene videos and trailers.

At the time, I felt, "You know what, I’ve been working on this game for about nine months, I want to get a half decent return from it. I'll set the price to $2.99." This was my second mistake.

The game released on 5 August, 2009, approximately a year after its development. I started to see one star reviews popping up. Ah! Haters gonna hate. This was my third mistake.

A few days later, my best friend purchased the game and called me up, and this was literally our conversation:

**Him:** Hey, I just bought your game. How do I play it?

**Me:** Oh, just tilt the device to move the character and press the crosshair button to shoot the viruses. Then...

I'm pretty sure he fell asleep as I was trying to explain it. (I'll tie this one into my first mistake.) As you can see by the following image, the user interface wasn't well labelled or explained. The interface was just thrown together.

There were so many things on screen the reader had to understand and none of it was explained.
There were so many on screen elements the player had to understand and none of it was explained.

It was an unfortunate error.
Needless to say, I went on to create a sequel using a program called **GameSalad**, which I've written two books on. I love the program. However, when I was creating iMMUNE 2, GameSalad was in its very early development stages, which meant that it would crash a lot, and there were a lot of features that were missing. Long story short, I miffed that one up too. And I was trying to improve on the first program's mistakes.

So, what went wrong? Let me break it down for you.

- **Mistake 1**: I rushed to get the game out, and I left out crucial features. As noted, I didn't include a main menu, which in itself isn't exactly a terrible thing. What is terrible is the fact I threw the player right into the game without telling them what to do. When the player got frustrated, they went on YouTube to see how to play, only to find out that if they tilted the device too far, the main character would run off the screen. Again, that little bit of code I forgot to write, *if player.position > screen.height = STOP MOVING*, was completely left out. How did I miss that?

- **Mistake 2**: I set the price for what seemed to be a reasonable $2.99. However, what else can you get for $2.99? A cup of coffee, or a ticket for the Toronto Maple Leafs, considering how bad they are lately? There's a lot $2.99 can buy, just go into a dollar store and see. Do people want to pay $2.99 for an app? Nope. Have they ever? Not really. This is why most games have a freemium payment structure. The game is free to download but you can purchase 1,800 carrot seeds for a nickel.

- **Mistake 3**: When I started to see the one star reviews come in, I figured it was just people "hating" because they do! I mean, come on, my game was pretty awesome. Right? *right*? Nope.

What did I learn from these critical errors that caused a waste of a year of my time to be a? For one, don't rush projects. High budget projects are rushed to meet deadlines, and we as the players, hate the game because it's so broken. Check every line of code, play test the heck out of it to ensure that everything is working 100 percent. Granted, there are things you will miss that 14 year olds smashing their hand-me-down iPad with their greasy fingers at a rate of 1,100 rpm will catch because that's not how the game is suppose to be played. Ensure that it's working right the first time. Can I suggest focus groups to play test before you release?
The high price tag was a major error. I wanted to get a large return on the time I spent developing the game, which is why I set the price too high—way too high. I think we all have that mindset when we first develop a game, and we don't consider the fact our price cannot hold the market. I should have priced it the lowest it could go, which is $0.99 at the time, "freemium" wasn't really a thing. I felt I had put a decent amount of effort into the game, as you can see by the following image, I even put stomach acid that would tilt according to how you were holding the device. I felt I had made a decent, quality game. Oh how wrong I was.

Finally, I should have read the reviews and taken them to heart. I realized what I did to the players, and how it wasn't fair. My game was a little broken, and I admitted it. I worked to fix it and updated it to version 1.1. I added a main menu, a help screen, a help popup at the beginning of the game, and I fixed that stupid bug where the character would float off the edge of the screen instead of just stopping. Also, I reduced the price tag to $0.99. You can probably guess what I'm going to say next. It was too late. By the time iMMUNE 1.1 was released to the general public, the disinterest and general disdain for the previous errors caused the game to ultimately fail.
But, what about new players? Wouldn't they see the new screenshots and gameplay when the game was updated? Yep, they would—for a time frame that is smaller than the thermal exhaust port on the death star. The AppStore is a great place, but apps swarm it like zombies swarm "Rick Grimes", you will see your app go on the new list, and after maybe an hour or so, it's down ten places or more.

This is my personal experience in the game development market. I have had successes, but only after my failures. When I released my latest game paceRoads on Desura and IndieRoyale, I was lucky enough to be told by my developer relations contact to lower the price I had set before the game was even released, which ultimately led to the success of the game.

I know all this sounds like doom and gloom, and that I'm "hating" on the market. But, I'm not, I love game development and the whole community of developers. I just don't want you to make the same mistakes that I did.

I talk about how there's a huge potential to make money in the market, and there really is! If you have time look up the creator of Trism, or just look at Angry Birds. Enough said, but those are normally huge corporations with billions of dollars to throw at advertising. Normal people like you and I don't have that much money, so our creation, as amazing as it may be, could get lost in the swarm. We will discuss how to overcome this in Chapter 7, Deploying and Monetizing.

Don't let any of this discourage you because game development and the market is actually a lot of fun. If you have any questions, you can go on a developers' forum and ask a group of developers who are way more knowledgeable than you are, and you'll get an answer—a good one too. I've even had a developer ask me to send him my project so that he could see what was wrong, and he reprogrammed the set of code that was causing me grief.

Trust me when I say you've entered an exciting world! Game development is an expression of, not only your technical knowhow, but also your creativity.

I had recommended in my previous book, GameSalad Essentials by Packt Publishing, and I will recommend it again in this book, please take a look at the movie Indie Game The Movie. You will see some real world experiences of developers who made it huge in the market, and you'll see the struggles and hardships they endured. I highly recommend watching it.

Enough of my ranting! Let's get to the fun part, let's start creating the assets for our game in the next chapter.
Summary

In this chapter, we signed up to become an iOS developer, set up Xcode on our Mac, and discussed setting up the certificates so that we can begin with the development of our awesome game! We had a quick overview of Xcode and how the coding files work, and we discussed some programs that we can use to create the assets for our game (the ones that I use and the free alternatives). Then, we discussed my personal experience in the game development market, what to expect, and which mistakes to avoid.

Now that we are done with all of that, let's begin with asset creation!
Where to buy this book

You can buy Mastering iOS Game Development from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.