iOS Game Development By Example

Game development has always been an exciting subject for game enthusiasts, and iOS game development takes a big leap in terms of perpetuating growth and creativity. With iOS 9 and Sprite Kit, comes a series of breathtaking features to improve your game development experience. Sprite Kit, an exciting framework supported by Apple, allows you to create stunning games at blazing speed.

Learn the basics and create your own fully functional platformer game on your iOS platform by using the Sprite Kit game engine. Take a detailed look at game development with the various features implemented in iOS 8 that further increase the essence of game development. Explore and implement features such as physics bodies, character animations, scoring, and other essential elements in your game. You will successfully conceive a 2D game along with discovering the path to reach the pinnacle of iOS game development.

Who this book is written for
This book is targeted at novice, intermediate, and proficient game developers coming from a different development platform and wanting to migrate to game development with the Sprite Kit engine. The reader does not need any knowledge of Sprite Kit and building games on the iOS platform.

What you will learn from this book
- Learn about the Sprite Kit game engine and create indie games in no time
- Set sail on the quest of game development by successfully creating a runner game
- Find out more about the IDE provided by Apple for game development – Xcode
- Get an overview of Apple’s latest programming language, Swift
- Discover the functionalities of scenes and nodes in a game
- Explore how physics bodies work and how to add this feature into your game
- Grasp knowledge of particle effects and shaders
- Add a scoring system into your game to visualize high scores

iOS Game Development By Example

Learn how to develop an ace game for your iOS device using Sprite Kit.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'An Introduction to Sprite Kit'
- A synopsis of the book’s content
- More information on iOS Game Development By Example
About the Author

**Samanyu Chopra** is a developer, entrepreneur, and game developer with a bank of experience in conceptualizing, developing, and producing computer and mobile software. He has been programming since the age of 11. He is proficient in programming languages such as JavaScript, Scala, C#, C++, Swift, and so on. He has a wide range of experience in developing for computers and mobiles. He has worked on a majority of game engines and mobile platforms, and also has a strong proficiency in the Scala programming language.

He is the cofounder and CEO of Daphnis Labs, a mobile app and game development studio. He has experience in managing development, leading the tech jargon, and has published over 150 apps and games for himself and his clients. In his studio, he leads a team of more than 20 members. He is known for executing eccentric projects in the app and game development spaces. He also conducts mobile development workshops at various engineering institutes.

He is ardent about his work, and his colleagues, students, and coworkers think of him as a very dedicated and open-minded person. He is inclined toward investing his time in the research and development of new technologies.

He is an avid traveler and adventurer and a very fun-loving and eccentric personality, as described by the people around him. He has an undying love for travel, tech, and comedy. He is inspired by all the people who have made a stand on their own from scratch. He loves his family and dedicates his life to them.
You can know more about Daphnis Labs at www.DaphnisLabs.com. You can write a tweet to him at @samonly1 or find him on Facebook for any updates. You can also reach him at samanyu@DaphnisLabs.com.
Preface

Sprite Kit is a set of tools to develop 2D games for the Apple iOS platform. Sprite Kit provides powerful features for graphics and the animation of images having texture, and so on. It is one of the best available game engines for iOS devices. It is very simple and powerful, with full support provided by Apple, hence it is more reliable and convenient than any third-party game engine available today.

The Integrated Development Environment (IDE), Xcode, provided by Apple for app development can also be used for Sprite Kit game development. Objective-C or Swift, either of the two programming languages can be used for Sprite Kit game development. Sprite Kit is already being used by many developers for iOS game development. There is a good amount of information available that focuses on Sprite Kit set up and development. However, a structured and concise resource discussing about the complete development process and feature set is not currently available. This book explains the basics of Sprite Kit development and allows a beginner to become skilled in Sprite Kit game development using the Swift programming language with its complete set of development features. This book is a complete guide for Sprite Kit and a perfect starting point for those wanting to set sail on the iOS game industry.

What this book covers

Chapter 1, An Introduction to Sprite Kit, introduces you to the Sprite Kit game engine, along with its various elements and features. It also helps in setting up a new Xcode project for developing a Sprite Kit game.

Chapter 2, Scenes in Sprite Kit, explains an important topic in Sprite Kit, that is, scenes. Along with this, there is a brief about the node tree drawing order.

Chapter 3, Sprites, explains sprites and their properties. It also applies some of the properties in the example game.
Chapter 4, *Nodes in Sprite Kit*, discusses about nodes and its subclasses in detail. It also explains the implementation of various node subclasses in the example game.

Chapter 5, *Physics in Sprite Kit*, talks about physics simulation in a Sprite Kit game. It explains the types of physics bodies. Physics capabilities are applied to the example game in this chapter.

Chapter 6, *Animating Sprites, Controls, and SceneKit*, covers animating nodes and adding controls to a Sprite Kit game. These features are added in the example game. It also talks about SceneKit.

Chapter 7, *Particle Effects and Shaders*, discusses about particle effects and shaders, along with their implementation in the example game.

Chapter 8, *Handling Multiple Scenes and Levels*, helps us understand the need for different levels in a game. This chapter also explains how to create multiple scenes.

Chapter 9, *Performance Enhancement and Extras*, discusses in detail how to improve the performance of a Sprite Kit game, along with performance measuring using instruments. It also explains the scoring system, sound, and player running animation in the example game.

Chapter 10, *Revisiting Our Game and More on iOS 9*, discusses the various steps involved in the development of a game, and introduces readers to the Game Center and discusses new features that will be introduced in iOS 9.
An Introduction to Sprite Kit

In this book we will be discussing about iOS game development using Sprite Kit. We will be taking a fun approach and shall make an actual 2D platform game on the iPhone in the process. We are going to develop a 2D (two dimensional) game; a game which relies on only two coordinates. Some famous 2D games include Mario, Hill Climb Racing, Angry Birds, Cut the Rope, and so on.

A 2D game only deals with two dimensions along $x$ and $y$ axes (left/right and up/down) but not along the $z$ axis (forward/backward). So basically, players cannot rotate or move the camera freely in a 3D space to view objects from other angles and perspectives. Although there are exceptions such as 2.5D games; we will be talking about that in later chapters. So, let's not keep things waiting and dive into the book.

What's new in iOS 8?

You might be familiar with Apple's mobile operating system, popularly known as iOS; the latest version of this operating system is iOS 8. This version has a lot of new additions over its predecessor, iOS 7. Some of the additions in this version are the introduction of the Swift programming language, loads of new API's, and most importantly, improvements in Sprite Kit and its peripheral frameworks.

In this book, we will be using the Swift programming language over Objective-C. Although, you can use Sprite Kit with either Objective-C or Swift, Swift offers much easier syntax, and has a simpler learning curve.
An Introduction to Sprite Kit

**Getting to know Swift**

Swift is Apple's entirely new multi-paradigm programming language for developing applications on Apple devices. Swift has been in development for 4 years, and was announced in 2014 at the **Worldwide Developer Conference (WWDC)**. Swift is both, a scripting and programming language; it has the ability to return multiple return values. Swift takes different constructs that are loved from many languages including Objective-C, Rust, Haskell, Ruby, Python, C#, CLU, and more. It has type safety feature that is, to prevent you passing string as int thus minimizing possible errors in your code.

We will be discussing more about Swift, as and when required, in the further topics covered.

**Getting to know Sprite Kit**

Sprite Kit is a framework from Apple, meant for developing 2D games for iOS devices. It is one of the best ways to make games for iOS devices. It is easy to learn, powerful and fully supported by Apple, which makes it more reliable to use than third-party game development engines.

Sprite Kit was introduced in iOS 7 and allowed easy, fast game development; it has similarities with Cocos2d, which is a popular library for game development. If you are somewhat familiar with Cocos2d, Sprite Kit will be a breeze for you.

Sprite Kit provides various functionalities that are useful for games, such as graphics rendering, animation utilities, sound playback, a particle system, and physics simulation. In Sprite Kit, every node will have a property name and physics body, which can consist of arbitrary shapes such as rectangles, polygons, circles, paths, and so on. Sprite Kit provides a richer particle system, where any aspect can be changed by code during the animation. In Sprite Kit's particle system, you can also add custom actions to the particles created. In addition, Xcode provides built-in support for Sprite Kit so that you can create complex special effects and texture atlases directly in Xcode. This combination of framework and tools makes Sprite Kit a good choice for games and other apps that require similar kinds of animation.

Because Sprite Kit supports a rich rendering infrastructure, and handles all of the low-level work to submit drawing commands to OpenGL, you can focus your efforts on solving higher-level design problems and creating your game functionality.

As Sprite Kit is a native framework of iOS, it provides in-built support for using the particle effects, texture effects, and physics simulations. The performance of Sprite Kit is better than other third-party frameworks/gaming engines, as it is a native framework.
Advantages of Sprite Kit

The main advantage of Sprite Kit is that it’s built into iOS. There is no need to download any other third-party libraries or depend on external resources to develop 2D games. Other iOS APIs such as, iAd, In-App purchases, and so on, can be easily used without banking on extra plugins. You don’t have to get familiar with any new programming language, the languages supported for Sprite Kit can also be used for app development on iOS. The best thing of all is that it is free, you get all the functionalities of Sprite Kit at no cost. You can run your game on both Mac and iOS without much effort, all you need to do is change its controls.

Elements of Sprite Kit

Now we are going to discuss some elements of Sprite Kit, which are essential for game development. A game made in Sprite Kit consists of many scenes which are made of nodes, and the functioning of a node in a scene is determined by actions.

Scenes

A level or environment in a game is termed as a scene. We make scenes as per our requirement, such as menus, levels, and so on. So, there are different scenes for different levels and also for different menus in a game. It’s like a canvas where you position your elements.

A scene in Sprite Kit is represented by an SKScene object. A scene holds sprites and other contents to be rendered. To switch scenes, we can use the SKTransition class.

Nodes

Nodes are fundamental building blocks for all content in a scene. The SKScene class is a descendant of the SKNode class, so a scene is a root node. The SKNode class does not draw anything on scene by itself; we can think of it as a base class for other node classes. There are node subclasses as follows:

- SKSpriteNode: This can be used for drawing textured sprites, playing video content, and more
- SK3DNode: This can be used for rendering a Scene Kit scene as a 2D textured image
- SKVideoNode: This can be used for playing video content
- SKLabelNode: This can be used for rendering a text string
An Introduction to Sprite Kit

- **SKShapeNode**: This can be used for rendering shape, based on a core graphics path
- **SKEmitterNode**: This can be used for creating and rendering particles
- **SKCropNode**: This can be used for cropping child nodes using a mask
- **SKEffectNode**: This can be used for applying a core image filter to its child node
- **SKLightNode**: This can be used for applying lighting and shadows to a scene
- **SKFieldNode**: This can be used for applying physics effects to a specific portion of the scene

**Actions**
An action tells a node what to do and allows you to perform different things, such as:

- Moving nodes in any direction
- Making any node follow a path
- Rotating nodes
- Scaling of nodes
- Showing or hiding a node
- Changing the content of a sprite node
- Playing sound
- Removing nodes from a scene
- Performing action on a child's node, and so on

To create a run action, first, create the action using the particular action class, configure the properties for the created action, and call a run action by passing action object as a parameter. When the scene processes the node, the actions of that particular node will be executed.

**Features of Sprite Kit**
Sprite Kit provides many features to facilitate the development of a game. These features can be used for enhancing the experience as well as performance of the game. Let's discuss them in brief.
Particle editor
This feature was introduced in iOS 7. Particle editor is used to add special effects in a game, like adding a mist effect in a game scene. Here, we can customize many things, such as:

- The number of particles
- Limit of particles allowed
- The color of particles
- The size of a particle
- The life of a particle
- The location of a particle in a scene, and so on

Texture atlas generator
Texture atlas generator combines all image files into one or more large images, in order to improve performance. We will discuss this in detail in the later chapters. It is recommended to use a lesser number of images to reduce draw calls (number of images rendering on a scene).

Shaders
Shaders were introduced in iOS 8. They are used to produce a variety of special effects; they calculate rendering effects on graphic hardware with a high degree of flexibility, for example, we have seen ripple effects in many apps/games. Wherever a user touches the screen, a ripple effect will be produced.

In Sprite Kit, shaders are represented by the SKShaderNode class object.

Lighting and shadows
Lighting and shadows were introduced in iOS 8. These effects are produced using the SKLightNode class object. The SKLightNode object can:

- Spread a lighting effect at any desirable position on the scene
- Add lighting in any sprite
- Support colors and shadows

It's just a type SKNode, so we can apply any property that we apply to any SKNode.
An Introduction to Sprite Kit

Physics
Simulating physics in Sprite Kit can be achieved by adding physics bodies to the scenes. A physics engine has the sole purpose of moving objects around in a simulated world. The physics bodies take the properties of objects, such as mass, shape material, current trajectory, and so on, and calculate a new position for all those objects.

Every object on the Sprite Kit game scene will have a physics body. A physics body object is connected to a node on the node tree of a particular scene. The scene will simulate the effect of forces and collisions on those particular physics bodies that are connected to the node tree, whenever the scene computes a new frame of animation. We can apply a particular physics property on those nodes using their particular physics properties such as gravity, mass, force, friction, and so on.

The game loop
Following is a frame life cycle diagram:

At the start, the update function is called to where we set up the logic of the game. After that, the scene evaluates the actions. After the actions are evaluated, we get a callback. After that, we set up physics, if any. When the physics simulation is finished, we get another call with didSimulatePhysics. Then, we apply constraint and get another callback, didApplyConstraints. The last callback method is didFinishUpdate; we get it just before frame is completed and view is ready to render. Finally SKView renders the scene; the frame is complete and it continues 60 times per second.
Setting up a project
We have discussed many things about Sprite Kit, now it's time to see a project in action and gain some practical knowledge.

The Hello World project
We'll need to create a new project to build Hello World. An Xcode project organizes everything your app needs into one convenient place. Let's begin by creating a brand new game project in Xcode by carrying out either of the first two points, and then continuing as shown in the list:

1. Click on **Create a new Xcode project** on the welcome screen:
2. Instead, you can also select **File | New | Project...** from the file menu:

![File menu screenshot](image)

3. Select **Game** from the new project selection window:

![Project selection window](image)
4. The next window asks you to customize options for your project. Fill out the fields as shown in the following screenshot:

![Screenshot of customizing options](image)

- **Product Name**: It is the name of the game
- **Organization Name**: If you are an individual, then your name, or the name of the organization
- **Organization Identifier**: A unique identifier of your organization
- **Bundle Identifier**: It is a default ID generated automatically using organization identifier and product name.
- **Language**: The programming language you are using, that is, Objective-C or Swift
- **Game Technology**: The game framework being used, like Scene Kit, Sprite Kit, Metal, and so on
- **Devices**: The devices you want your game to run on; iPad, iPhone, or both
- These fields can be anything you want

5. Press **Next** and Xcode will ask where to save your new project. Choose a directory and then click on **Create**.
6. After saving, it should open Xcode to your brand new Hello World project, specifically to the project properties screen. On this screen, unselect the Portrait option under Device Orientation. This file will be automatically saved, so you won't have to do anything further:
Result

Run the default game project by pressing \( \text{⌘} + R \) on your keyboard, or by clicking on the little play button in the top left corner. If a simulator isn't present, Xcode will download one for you before launching the app. The result will look as follows:

![Hello, World!](image)

Summary

We also learned how to create a Sprite Kit project and run Hello World in it.

In the next chapter, we will be diving deeply into scenes, and also into adding scenes to our Platformer game.
Where to buy this book
You can buy iOS Game Development By Example 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.