Contents

1 Preample

Welcome to **the** most compact yet detailed guide to iOS game programming. You will be guided through absolutely everything you need to know about Cocos2d and SpriteBuilder and 2d game programming in general.

While we will cover the very basics of game programming, such as scene graphs, animations and game loops - Objective-C, the language we will be using throughout the book is not in the scope of things you will learn. When starting this guide, you are expected to have a solid foundation of Objective-C knowledge.

The structure in which you will learn is the following:

- Tools: Get familiar with the very basics of Cocos2d and SpriteBuilder
- Infrastructure: Understand that on a high level a game consists of scenes. Understand how to create scenes and navigation paths through these scenes with Cocos2d and SpriteBuilder
- Action and Movement: Understand how objects in your game can be moved and animated. With Cocos2d and SpriteBuilder
- Interaction: Understand how user interaction can be captured, including Touch interaction and Accelerometer.
- Interobject Interaction: Understand how to use the delightfully integrated Chipmunk physics engine
- Beyond the Basics; Recipes and Best Practices: Once we have the basics, we will look at a ton of recipes and exciting Cocos2d classes, which you can use to create any kind of 2d game. Particle Effects, Custom Drawing, Custom Shaders, Tile Maps, Networking, Audio, cocos2d UI in depth, etc.

1.1 Structure of this book

This book shall function as a learning guide and a reference book. Therefore most examples will be small and self-contained. Instead of builind a game throughout the whole book, you will learn by implementing very small projects that are limited to the material we are currently discussing. That shall give you a better chance of understanding the concepts/code snippets and using them in your original game, instead starting of from an example game you have built in this book.

After we have discussed all the basics and you have a good understanding of the Cocos2d API I will point you to resources that provide example implementations for specific game types.

There are two different ways to read this book. From the front to the beginning, gaining knowledge in logical groups. Or if you aren't a beginner and would like to use this book as an example driven extension of the API reference you can look up pages by Class names or concept names. There is a special glossar in the back of this book.

1.2 Tools used throughout this book

The two main tools we will be using are Cocos2d and SpriteBuilder. Many of the problems that occur during game development can be solved by both of these tools. Wherever it makes sense I will point out both ways, one using only Cocos2d and one using SpriteBuilder. This will allow you to see the advantages of each approach and finally decide which tool you want to use in certain situations for your own games.

1.3 What is a 2D game engine?

2 Introduction to SpriteBuilder and Cocos2d

Now it's time to dive into 2D Game Development! For this chapter I will assume that you haven't written a game with a game engine so I will explain the relevant concepts fairly detailed.

2.1 Introduction to Cocos2d

To understand what Cocos2d is, it is helpful to look at the history of game development. The first video games were written in assembler (a very low level programming language) and images were drawn to the screen by manually setting colors for certain pixels. Since then a wealth of frameworks and libraries has been written to make the life of a game developer easier.

Here are some of the most important features of Cocos2d that make 2D game development a lot easier:

Structure like most game engines Cocos2d demands a specific structure of the code for your game, making most design decisions simple

Scene Graph and Sprites loading an image for a character and adding it to the gameplay scene can happen in two lines of code

Action System a sophisticated action system allows to define movements of objects and animations

Integrated Physics Engine automatically calculates movements of objects, collisions and more.

There are a whole bunch of more features - but this brief outline shows the most important ones and should give you an idea why almost all game developers these days use game engines.

Cocos2d is built on top of OpenGL ES 2.0. If you have ever written OpenGL code before you know that it takes many lines of code to draw a textured rectangle on the screen. Cocos2d abstracts all of these tasks for you, you can go a very long way without writing any OpenGL code. The following diagram shows which technologies are used by Cocos2d:

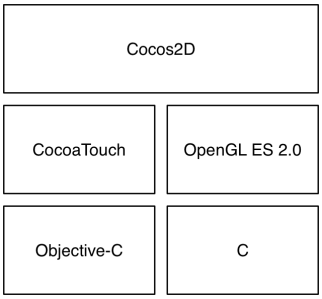

Figure 2.1: Cocos2D Technology Stack

As a Cocos2d developer you define which scenes exist in your game, which objects are part of these scenes and which size, position and appearance these objects have and Cocos2d will use OpenGL to render the scene you have described.

You have probably realized this already - when working with a 2D game engine for the first time you will be introduced to a whole set of new terminology. Just as a framework to write desktop applications knows the concept of windows, buttons and mouse clicks a 2D game engine comes with its own set of terms and techniques. We will spent the next sections discussing the important terms in detail.

2.1.1 Scenes

For explaining the important terms and concepts of Cocos2d I will use a top down approach. First we will learn how games that use Cocos2d are structured. On the highest level of structure we have a concept called *scenes*. By default every scene in Cocos2d is full-screen. This means for every screen in your game you will use one scene.

Here's an example from the MakeGamesWithUs game *Deep Sea Fury*: You can see that the game consists of the start scene, the gameplay scene and the game over scene.

A Cocos2d developer creates scenes by using the CCScene class. Another important Cocos2d class for scene handling is CCDirector. The CCDirector class is responsible for deciding which scene is currently active in the game (Cocos2d only allows one active scene at a time). Whenever a developer wants to display a scene or transition between two scenes he needs to use the CCDirector class.

2.1.2 Nodes

2.1.3 Scene Graphs

2.2 Introduction to SpriteBuilder

As of this writing SpriteBuilder itself is a very new tool, released in early 2014.

2.2.1 The Editor

2.2.2 CCB Files

2.2.3 Publishing - How SpriteBuilder and Xcodework together

2.2.4 How SpriteBuilder and Cocos2d

2.2.5 Code Connections

2.3 A first SpriteBuilderproject