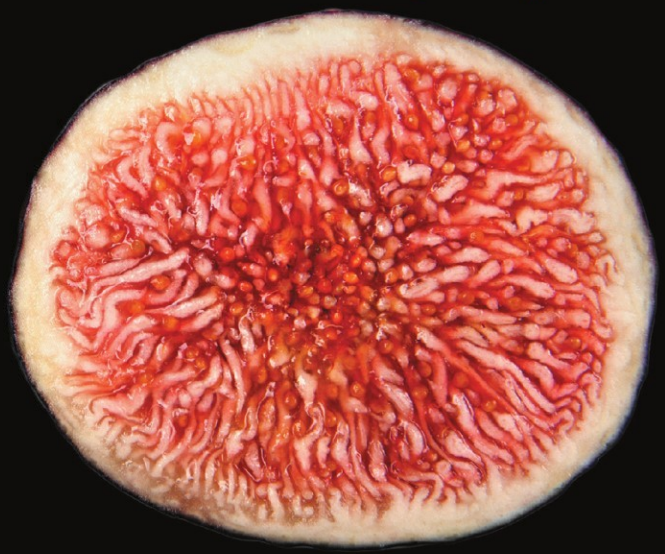
Use cutting-edge tools to create exciting iPhone and iPad game apps

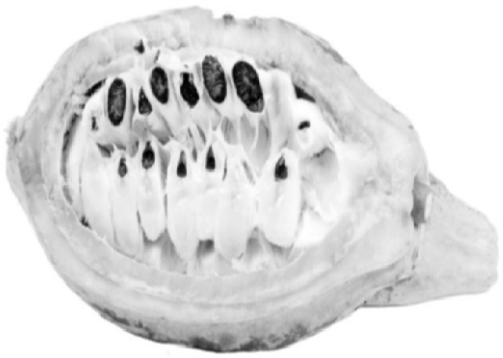


Learn cocos2d Game Development with iOS 5

Steffen Itterheim | Andreas Löw

Apress®

Learn cocos2D Game Development with iOS 5



Steffen Itterheim
Andreas Löw

Apress®

Learn cocos2D Game Development with iOS 5

ISBN 978-1-4302-3813-3

ISBN 978-1-4302-3814-0 (eBook)

President and Publisher: Paul Manning
Lead Editor: Steve Anglin
Development Editor: Chris Nelson
Technical Reviewer: Boon Chew
Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Coordinating Editor: Kelly Moritz
Copy Editors: Kim Wimpsett
Compositor: MacPS, LLC
Indexer: SPi Global
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, 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/info/bulksales.

The source code for this book is available to readers at www.apress.com and www.learn-cocos2d.com/store/book-learn-cocos2d.

To Gabi, the one and only space ant.
Sometimes alien, often antsy, always loved.

(Steffen)

To Saskia & Renate for making it possible to
spend my time with things I love most.

(Andreas)

Contents at a Glance

Contents

About the Authors

Steffen Itterheim has been a game development enthusiast since the early 1990s. His work in the Doom and Duke Nukem 3D communities landed him his first freelance job as a beta tester for 3D Realms. He has been a professional game developer for more than a decade, having worked most of his career as a game play and tools programmer for Electronic Arts Phenomic. His first contact with cocos2d was in 2009, when he cofounded an aspiring iOS games start-up company called Fun Armada. He loves to teach and enable other game developers so that they can work smarter, not harder. Occasionally you'll find him strolling around in the lush vineyards near his domicile at daytime, and the desert of Nevada at night, collecting bottle caps.

Andreas Löw has been a computer feak since he the age of 10 when he got is first Commodore C16. Teaching himself how to write games, he released his first computer game, Gamma Zone, for Commodore Amiga in 1994, written in pure assembly language. After his diploma in electrical engineering, he worked for Harman International, in the department responsible for developing navigation and infotainment systems with speech recognition for the automotive industy. He invented his own programming language and development tools, which are in use by every car with speech recognition technology around the world.

With the iPhone, he found his way back to his roots and began developing a game called TurtleTrigger. He realized there is a huge demand for good tools in the cocos2d community. With his knowledge in both game and tool development, his products TexturePacker and PhysicsEditor quickly became essential development tools for any cocos2d user.

About the Technical Reviewer

Boon Chew is the managing director for Nanaimo Studio, a game studio based in Seattle and Shanghai that specializes in web and mobile games. He has extensive experience with game development and interactive media, having previously worked for companies such as Vivendi Universal, Amazon, Microsoft, and various game studios and advertising agencies. His passion is in building things and working with great people. You can reach Boon at `boon@nanaimostudio.com`.

Acknowledgments

This is the part of the book that makes me a little anxious. I don't want to forget anyone who has been instrumental and helpful in creating this book, yet I know I can't mention each and every one of you. If you're not mentioned here, that doesn't mean I'm not thankful for your contribution! Give me a pen, and I'll scribble your name right here in your copy of the book, and I'll sincerely apologize for not having mentioned you here in the first place.

My first thanks go to you, dear reader. Without you, this book wouldn't make any sense. Without knowing that you might read and enjoy this book, and hopefully learn from it, I probably wouldn't even have considered writing it in the first place. I've received valuable insights and requests from my blog readers and other people I've met or mailed during the course of this book. Thank you, all!

My first thanks go to Jack Nutting, who put the idea of writing a book about cocos2d in my head in the first place. I'm grateful that he did not sugarcoat how much work goes into writing a book so that I wasn't unprepared.

Clay Andres I have to thank for being such a kind person, whose input on my chapter proposals were invaluable and to the point. He helped me form the idea of what the book was to become, and he's generally a delightful person to talk to. Clay, I hope that storm did not flood your house.

Many thanks to Kelly Moritz, the coordinating editor, who though incredibly busy always found the time and patience to answer my questions and follow up on my requests. When chaos ensued, she was the one to put everything back in order and made it happen.

Lots and lots of feedback and suggestions I received from Brian MacDonald and Chris Nelson, the development editors for the book, and Boon Chew, the technical reviewer. They made me go to even greater lengths. Brian helped me understand many of the intricacies of writing a book, while Boon pointed out a lot of technical inaccuracies and additional explanations needed. Many thanks to both of you. Chris was a tremendeous help for the second edition; he pointed out a lot of the small but crucial improvements. He shall forever be known as CCCC: Code Continuation Character Chris.

Many thanks go to the copy editor, Kim Wimpsett. Without you, the book's text would be rife with syntax errors and compiler warnings, to put it in programmer's terms.

I also wish to thank Bernie Watkins, who managed the Alpha Book feedback and my contracts. Thanks also to Chris Guillebeau for being an outstanding inspirational blogger and role model.

Of course, my friends and family all took some part in writing this book, through both feedback and plain-and-simple patience with putting up with my writing spree. Thank you!

Preface

In May 2009 I made first contact. For the first time in my life, I was subjected to the Mac OS platform and started learning Xcode, Objective-C, and cocos2d. Even for experienced developers like me and my colleagues, it was a struggle. It was then that I realized cocos2d was good, but it lacked documentation, tutorials, and how-to articles—especially when compared with the other technologies I was learning at the time.

Fast-forward a year to May 2010. I had completed four cocos2d projects. My Objective-C and cocos2d had become fluent. It pained me to see how other developers were still struggling with the same basic issues and were falling victim to the same misconceptions that I did about a year earlier. The cocos2d documentation was still severely lacking.

I saw that other cocos2d developers were having great success attracting readers to their blogs by writing tutorials and sharing what they know about cocos2d. To date, most of the cocos2d documentation is actively being created in a decentralized fashion by other developers. I saw a need for a web site to channel all of the information that's spread over so many different web sites.

I created the `www.learn-cocos2d.com` web site to share what I knew about cocos2d and game development, to write tutorials and FAQs, and to redirect readers interested in cocos2d to all the important sources of information. In turn, I would be selling cocos2d-related products, hoping it might one day bring me close to the ultimate goal of becoming financially independent. I knew I could make the web site a win for everyone.

From day one, the web site was a success—beyond my wildest imaginations. Then, within 24 hours of taking the web site live, Jack Nutting asked me if I had considered writing a cocos2d book. The rest is history, and the result is the book you're reading right now.

I took everything I had in mind for the web site and put it in the book. But that alone would have amounted to maybe a quarter of the book, at most. I hope the four months I spent writing the book full-time paid off by being able to provide an unprecedented level of detail on how cocos2d works and how to work with cocos2d.

I learned a lot in the process, and even more so during months updating the chapters of the second edition. I wish nothing more than for you to learn a great deal about cocos2d and game development from this book.

What I learned from writing about cocos2d is that there's a lot of room for improvement. I strongly believed that the world needed a better cocos2d that's more approachable for beginning game developers. The result of that is Kobold2D, which you'll find an introduction to in Chapter 16 and of course on `www.kobold2d.com`. Of course, almost all of what you'll learn throughout the book still applies to Kobold2D.

Steffen Itterheim

Chapter 1

Introduction

Did you ever imagine yourself writing a computer game and being able to make money selling it? With Apple's iTunes App Store and the accompanying mobile devices iPhone, iPod touch, and iPad, it's now easier than ever. Of course, that doesn't mean it's easy there's still a lot to learn about game development and programming games. But you are reading this book, so I believe you've already made up your mind to take this journey. And you've chosen one of the most interesting game engines to work with: cocos2d for iOS.

Developers using cocos2d have a huge variety of backgrounds. Some, like me, have been professional game developers for years and even decades. Others are just starting to learn programming for iOS devices or are freshly venturing into the exciting field of game development. Whatever your background might be, I'm sure you'll get something out of this book.

Two things unite all cocos2d developers: we love games, and we love creating and programming them. This book will pay homage to that yet won't forget about the tools that will help ease the development process. Most of all, you'll be making games that matter along the way, and you'll see how this knowledge is applied in real game development.

You see, I get bored by books that spend all their pages teaching me how to make yet another dull Asteroids clone using some specific game-programming API. What's more important, I think, are game programming concepts and tools the things you take with you even as APIs or your personal programming preferences change. I've amassed hundreds of programming and game development books over 20 years. The books I value the most to this day are those who went beyond the technology and taught me why certain things are designed and programmed the way they are. This book will focus not just on working game code but also on why it works and which trade-offs to consider.

I would like you to learn how to write games that matter games that are popular on the App Store and relevant to players. I'll walk you through the ideas and technical concepts behind these games in this book and, of course, how cocos2d and Objective-C make these games tick. You'll find that the source code that comes with the book is enriched

with a lot of comments, which should help you navigate and understand all the nooks and crannies of the code.

Learning from someone else's source code with a guide to help focus on what's important is what works best for me whenever I'm learning something new and I like to think it will work great for you too. And since you can base your own games on the book's source code, I'm looking forward to playing your games in the near future! Don't forget to let me know about them! You can share your projects and ask questions on Cocos2D Central (`www.cocos2d-central.com`), and you can reach me at `steffen@learn-cocos2d.com`. You might also want to visit my web site dedicated to learning cocos2d at `www.learn-cocos2d.com` and you should check out how I'm improving cocos2d with Kobold2D by visiting `www.kobold2d.com`.

What s New in the Second Edition?

First, I'm proud to have had Andreas Löw as the coauthor for the second edition. Andreas is the developer of the TexturePacker and PhysicsEditor tools, and in particular he went out of his way to update the projects for several chapters with new code and better graphics.

Most importantly, the goal of the second edition was to bring the book up to date with recent developments, one being the final 1.0.1 version of cocos2d as well as compatibility with Xcode 4 and iOS 5. The text, the code, and the figures have all been updated to reflect the new versions of cocos2d, Xcode and the iOS SDK.

Many more changes were made based on reader feedback. Chapter 3 has been overhauled to improve and extend the descriptions of essential cocos2d features. It has also become more visual, with a lot more figures illustrating key concepts and classes. The number of figures in the book has increased throughout.

Over the course of a year, new tools for cocos2d game development have emerged. To reflect the changing tool landscape, the book now refers to TexturePacker in favor of Zwoptex as the leading texture atlas creation tool. Since Löw works full-time on his tools, his customers benefit by receiving frequent updates, new features, and great support. Similarly, PhysicsEditor is used in the second edition in place of VertexHelper since it offers a far better workflow and powerful convenience features. Finally, the second edition introduces you to Glyph Designer, which is essentially the Hiero Bitmap Font tool but with a native Mac OS X user interface and with none of the bugs that plagued Hiero.

The shoot 'em up project first introduced in Chapter 6 and used throughout Chapters 7 to 9 has seen a graphic overhaul. Let's just say it looks a lot better than the programmer's art it used before, courtesy of myself. Likewise, Chapter 13, the pinball physics game, has been improved with new code and improved graphics. Accordingly, the aforementioned chapters have seen some of the more substantial changes.

Last but certainly not least, the second edition adds two entirely new chapters.

Chapter 15 explores the frequently misunderstood and underutilized possibility of mixing cocos2d with regular UIKit views. You'll learn how to add UIKit views to a cocos2d app as well as add cocos2d to an already existing UIKit app. This chapter will make it a lot easier for you to cross the chasm between pure UIKit and pure cocos2d apps, regardless of which way you're going.

Chapter 16 introduces you to Kobold2D (`www.kobold2d.com`), my idea of making cocos2d a much better game development environment. Kobold2D aims to make commonly performed tasks easy while adding preconfigured libraries such as wax (Lua Scripting), ObjectAL (OpenAL audio), and cocos3d (3D rendering) to the distribution. It also comes with a lot of project templates, many of them based on the projects discussed in this book.

Why Use cocos2d for iOS?

When game developers look for a game engine, they first evaluate their options. I think cocos2d is a great choice for a lot of developers, for many reasons.

It s Free

First, it is free. It doesn't cost you a dime to work with it. You are allowed to create both free and commercial iPhone, iPod, and iPad apps. You can even create Mac OS X apps with it. You don't have to pay royalties. Seriously, no strings attached.

It s Open Source

The next good reason to use cocos2d is that it's open source. This means there's no black box preventing you from learning from the game engine code or making changes to it where necessary. That makes cocos2d both extensible and flexible.

You can download cocos2d from `www.cocos2d-iphone.org/download`.

It s Objective, See?

Furthermore, cocos2d is written in Objective-C, Apple's native programming language for writing iOS apps. It's the same language used by the iOS SDK, which makes it easier to understand Apple's documentation and implement iOS SDK functionality.

A lot of other useful APIs like Facebook Connect and OpenFeint are also written in Objective-C, so it makes it easier to integrate those APIs, too.

> **NOTE:** Learning Objective-C is advised, even if you prefer some other language. I have a strong C++ and C# background, and the Objective-C syntax looked very odd at first glance. I wasn't happy at the prospect of learning a new programming language that was said to be old and outdated. Not surprisingly, I struggled for a while to get the hang of writing code in a programming language that required me to let go of old habits and expectations.
>
> Don't let the thought of programming with Objective-C distract you, though. It does require some getting used to, but it pays off soon enough, if only for the sheer amount of documentation available. I promise you won't look back!

It s 2D

Of course, cocos2d carries the 2D in its name for a reason. It focuses on helping you create 2D games. It's a specialization few other iOS game engines are currently offering.

It does not prevent you from loading and displaying 3D objects. In fact, an entire add-on product aptly named cocos3d has been created as an open source project to add 3D rendering support to cocos2d.

But I have to say that the iOS devices are an ideal platform for great 2D games. The majority of new games released on the iTunes App Store are still 2D-only games even today. 2D games are generally easier to develop, and the algorithms in 2D games are easier to understand and implement. In almost all cases, 2D games are less demanding on the hardware, allowing you to create more vibrant, more detailed graphics.

It s Got Physics

There are also two physics engines you can choose from that are already integrated with cocos2d. On one hand there's Chipmunk, and on the other there's Box2d. Both physics engines superficially differ only in the language they're written in: Chipmunk is written in C, and Box2d is written in C++. The feature set is almost the same. If you're looking for differences, you'll find some, but it requires a good understanding of how physics engines work to base your choice on the differences. In general, you should simply choose the physics engine you think is easier to understand and better documented, and for most developers that tends to be Box2d. Plus, its object-oriented nature makes it a little easier to use with Objective-C.

It s Less Technical

What game developers enjoy most about cocos2d is how it hides the low-level OpenGL ES code. Most of the graphics are drawn using simple sprite classes that are created from image files. In other words, a sprite is a texture that can have scaling, rotation, and color applied to it by simply changing the appropriate Objective-C properties of the

`CCSprite` class. You don't have to be concerned about how this is implemented using OpenGL ES code, which is a good thing.

At the same time, cocos2d gives you the flexibility to add your own OpenGL ES code at any time for any game object that needs it. And if you're thinking about adding some Cocoa Touch user interface elements, you'll appreciate knowing that these can be mixed in as well.

And cocos2d doesn't just shield you from the Open GL ES intricacies; it also provides high-level abstraction of commonly performed tasks, some of which would otherwise require extensive knowledge of the iOS SDK. But if you do need more low-level access or want to make use of iOS SDK features, cocos2d won't hold you back.

It s Still Programming

In general, you could say that cocos2d makes programming iOS games simpler while still truly requiring excellent programming skills first and foremost. Other iOS game engines such as Unity, iTorque, and Shiva focus their efforts on providing tool sets and workflows to reduce the amount of programming knowledge required. In turn, you give away some technical freedom and cash too. With cocos2d, you have to put in a little extra effort, but you're as close to the core of game programming as possible without having to actually deal with the core.

It s Got a Great Community

The cocos2d community always has someone quick to answer a question, and developers are generally open to sharing knowledge and information. You can get in touch with the community on the official forum (`www.cocos2d-iphone.org/forum`) or in my own forum dubbed Cocos2D Central (`http://cocos2d-central.com`).

New tutorials and sample source code are released on an almost daily basis, most of it for free. And you'll find scattered over the Internet plenty of resources to learn from and get inspired by.

Once your game is complete and released on the App Store, you can even promote it on the cocos2d web site. At the very least, you'll get the attention of fellow developers and ideally valuable feedback.

> **TIP:** To stay up to date with what's happening in the cocos2d community, I recommend following cocos2d on Twitter: `http://twitter.com/cocos2d`.
>
> While you're at it, you might want to follow me as well: `http://twitter.com/gaminghorror`.
>
> Next, enter **cocos2d** in Twitter's search box and then click the Save this search link. That way, you can regularly check for new posts about cocos2d with a single click. More often than not, you'll come across useful cocos2d-related information that would otherwise have passed you by. And you'll definitely get to know your fellow developers who are also working with cocos2d.

The Future of the cocos2d-iphone Project

In May 2011, Zynga announced that it hired Ricardo Quesada and Rolando Abarca, key contributors to the cocos2d-iphone project.

Zynga is the developer of the blockbuster social game Farmville. The iPhone version of Farmville was created with cocos2d-iphone and published in June 2010. It's expected that Zynga plans to create more iOS games based on cocos2d-iphone now that Quesada and Abarca work for them.

Quesada is the creator and lead developer of cocos2d-iphone. In fact, he has been running all aspects of cocos2d-iphone since 2008. He managed the web site, he fostered the community, and he is the driving force behind the development and success of the cocos2d-iphone project. Abarca is a contributor to the project, best known for his Ruby wrapper for cocos2d-iphone.

Both developers relocated from Argentina and Chile, respectively, to work for Zynga in San Francisco. At the same time, their previous company, Sapus Media, has stopped selling their two flagship products, Sapus Tongue Source Code and Level SVG, after they had been bought by Zynga.

This means Quesada and Abarca now get a regular paycheck and aren't required to sell, support, and maintain their commercial products to make a living. On the other hand, they will now work primarily for Zynga, and time will have to tell how much of their work will actually find its way into the publicly available open source version of cocos2d-iphone. Currently Ricardo is working on cocos2d 2.0 which will use OpenGL ES 2.0 exclusively.

Although Zynga promises to advance the development of the cocos2d-iphone project and community, there is reason to doubt that the cocos2d-iphone project will continue to be developed at the same rate as in the past. Still, the community can help itself, and

some of them have already started a project dubbed cocos2d-iphone-extensions, which provides additional functionality for the cocos2d-iphone game engine.

I do not think we have to generally worry about the future of cocos2d. The cocos2d-iphone project has a strong community and has seen widespread adoption, and even though there are competing game engines, most of them are commercial and proprietary, not free and open source like cocos2d-iphone.

Other cocos2d Game Engines

You may have noticed that cocos2d ports exist for various platforms, including Windows, JavaScript, and Android. There's even a C++ version of cocos2d dubbed cocos2d-x that supports multiple mobile platforms, including iOS and Android.

These cocos2d ports all share the same name and design philosophy but are written in different languages by different authors and are generally quite different from cocos2d for iOS. For example, the Android cocos2d port is written in Java, which is the native language when developing for Android devices.

If you're interested in porting your games to other platforms, you should know that the various cocos2d game engines differ a lot. Porting your game to Android, for example, isn't an easy task. First there's the language barrier all your Objective-C code must be rewritten in Java. When that's done, you still need to make a lot of modifications to cope with numerous changes in the cocos2d API or possibly unsupported features of the port or the target platform. Finally, every port can have its own kind of bugs, and every platform has its own technical limitations and challenges.

Overall, porting iOS games written with cocos2d to other platforms that also have a cocos2d game engine entails almost the same effort as rewriting the game for the target platform using some other game engine. This means there's no switch you can flip and it'll work. The similarity of the cocos2d engines across various platforms is mostly in name and philosophy. If cross-platform development is your goal, you should take a look at cocos2d-x, which has most of the features of cocos2d-iphone and is backed financially by China Unicom.

In any case, you should still know about the most popular cocos2d game engines. Table 1–1 lists the cocos2d game engines that are frequently updated and are stable enough for production use. I did not include cocos2d ports in this list that are significantly out of date and haven't been updated for months, if not years.

Table 1–1. *Most Popular cocos2d Game Engine Ports*

Name	Language	Platforms	Web Site
cocos2d-iphone	Objective-C	iOS, Mac OS X	www.cocos2d-iphone.org
cocos2d-x	C++	iOS, Android, Windows	www.cocos2d-x.org
cocos2d-javascript	JavaScript	Web browsers	www.cocos2d-javascript.org
cocos2d-android-1	Java	Android	http://code.google.com/p/cocos2d-android-1
cocos2d	Python	Mac OS, Windows, Linux	www.cocos2d-javascript.org

This Book Is for You

I'd like to imagine you picked this book because its title caught your interest. I suppose you want to make 2D games for iPhone, iPod touch, and iPad, and the game engine of your choice is cocos2d for iOS. Or maybe you don't care so much about the game engine but you do want to make 2D games for the iOS devices in general. Maybe you're looking for some in-depth discussion on cocos2d, since you've been using it for a while already. Whatever your reasons for choosing this book, I'm sure you'll get a lot out it.

Prerequisites

As with every programming book, there are some prerequisites that are nice to have and some that are almost mandatory.

Programming Experience

The only thing that's mandatory for this book is some degree of programming experience, so let's get that out of the way first. You should have an understanding of programming concepts such as loops, functions, classes, and so forth. If you have written a computer program before, preferably using an object-oriented programming language, you should be fine.

Still with me? Good.

Objective-C

So, you do have programming experience, but maybe you've never written anything in that obscure language called Objective-C.

You don't need to know Objective-C for this book, but it definitely helps to know the language basics. If you are already familiar with at least one other object-oriented programming language, such as C++, C#, or Java, you may be able to pick it up as you go. But to be honest, I found it hard to do that myself even after roughly 15 years of programming experience with C++, C#, and various scripting languages. There are always those small, bothersome questions about tiny things you just don't get right away, and they tend to steal your attention away. In that case, it's handy to have a resource you can refer to whenever there's something you need to understand about Objective-C.

Objective-C may seem scary with its square brackets, and you may have picked up some horror stories about its memory management and how there's no garbage collection on iOS devices. Worry not.

First, Objective-C is just a different set of clothes. It looks unfamiliar, but the underlying programming concepts such as loops, classes, inheritance, and function calls still work in the same way as in other programming languages. The terminology might be different; for example, what Objective-C developers call sending messages is in essence the same as calling a method. As for memory management, let's just say cocos2d makes it as easy for you as possible, and I'll help you understand the very simple and basic rules you can follow.

I had one invaluable Objective-C book to learn from, and I recommend it wholeheartedly as a companion book in case you want to learn more about Objective-C and Xcode. It's called *Learn Objective-C on the Mac* by Mark Dalrymple and Scott Knaster, published by Apress.

There is also Apple's □Introduction to the Objective-C Programming Language,□ which proved valuable as an online reference. It's available here: `http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html`.

What You Will Learn

I will provide you with a fair share of my game development experiences to show how interactive games are made. I believe that learning to program is not at all about memorizing API methods, yet a lot of game development books I've read over the past two decades follow that □reference handbook□ approach. But that's what the API documentation is for. When I started programming some 20 years ago, I thought I'd never learn to program just by looking at a huge stack of compiler reference handbooks and manuals. Back at that time, compiler manuals were still printed and, obviously, didn't come with online versions. The World Wide Web was still in its infancy. So, all that information was stacked some 15 inches high on my desk, and it seemed very daunting to try to learn all of this.

Today, I still don't recall most methods and APIs from memory, and I keep forgetting about those I used to know. I look them up time and time again. After 20 years of programming, I do know what's really important to learn: the concepts. Good

programming concepts and best practices stick around for a long time, and they help with programming in any language. Learning concepts is done best by understanding the rationale behind the choices that were made in designing, structuring, and writing the source code. That's what I'll focus on the most.

What Beginning iOS Game Developers Will Learn

I'll also ease you into the most important aspects of cocos2d. I'll focus on the kind of classes, methods, and concepts that you should be able to recall from memory just because they are so fundamental to programming with cocos2d.

You'll also learn about the essential tools supporting or being supported by cocos2d. Without these tools, you'd be only half the cocos2d programmer you can be. You'll use tools like TexturePacker and ParticleDesigner to create games that will be increasingly complex and challenging to develop. Because of the scope of this book, these games will not be complete and polished games, nor will I be able to discuss every line of code. Instead, I'll annotate the code with many helpful comments so that it's easy to follow and understand.

I leave it up to you to improve on these skeleton game projects, and I'm excited to see your results. I think giving you multiple starting points to base your own work on works better than walking you through the typical Asteroids games over the course of the whole book.

I chose the game projects for this book based on popularity on the App Store and relevance for game developers, who often inquire about how to solve the specific problems that these games present. For example, the line-drawing game genre is a huge favorite among cocos2d game developers, yet line-drawing games require you to overcome deceivingly complex challenges.

I've also seen a fair bit of other developers' cocos2d code and followed the discussions on code design, structure, and style. I'll base my code samples on a framework that relies on composition over inheritance and will explain why this is preferable. One other frequent question that has to do with code design is how different objects should communicate with each other. There are interesting pros and cons for each approach to code design and structure, and I want to convey these concepts because they help you write more stable code with fewer bugs and better performance.

What iOS App Developers Will Learn

So, you are an iOS app developer, and you've worked with the iOS SDK before? Perfect. Then you'll be most interested in how making games works in a world without Interface Builder. In fact, there are other tools you'll be using. They may not be as shiny as Apple's tools, but they'll be useful nonetheless.

The programming considerations will change, too. You don't normally send and receive a lot of events in game programming, and you let a larger number of objects decide what to do with an event. For performance reasons and to reduce user input latency,

game engine systems often work more closely connected with each other. A lot of work is done in loops and update methods, which are called at every frame or at specific points in time. While a user interface-driven application spends most of the time waiting for a user's input, a game keeps pushing a lot of data and pixels behind the scenes, even when the player is not doing anything. So, there's a lot more going on, and game code tends to be more streamlined and efficient because of concerns for performance.

What Cocos2d Developers Will Learn

You're already familiar with cocos2d? You may be wondering whether you can learn anything new from this book. I say you will. Maybe you need to skip the first chapters, but you'll definitely get hooked by the games' sample source code supplied with the book. You'll learn how I structure my code and the rationale behind it. You'll probably find inspiration reading about the various games and how I implemented them. There's also a good number of tips you'll benefit from.

Most importantly, this book isn't written by some geek you've never heard of and never will hear from again, with no e-mail address or web site where to post your follow-up questions. Instead, it's written by a geek you may not have heard of but who will definitely be around. I'm actively engaged with the cocos2d community at my `www.learn-cocos2d.com` blog, where I'll basically keep writing this book.

What s in This Book

Here's a brief overview of the chapters in this book. The second edition of the book features two entirely new chapters: Chapter 15 discusses integration of UIKit views with cocos2d, and Chapter 16 introduces Kobold2D, my own take on a cocos2d development environment with additional convenience features.

Chapter 2, Getting Started

I'll cover setting up cocos2d for development, installing project templates, and creating the first □Hello World□ project. You'll learn about cocos2d basics, such as scenes and nodes.

Chapter 3, Essentials

I'll explain the essential cocos2d classes that you'll need most often, such as sprites, transitions, and actions. And you'll learn how to use them, of course.

Chapter 4, Your First Game

Enemies drop from the top, and you have to avoid them by tilting your device. This will be our first simple game using accelerometer controls.

Chapter 5, Game Building Blocks

Now prepare yourself for a bigger game, one that requires a better code structure. You'll learn how scenes and nodes are layered and the various ways that game objects can exchange information.

Chapter 6, Sprites In-Depth

You'll learn what a texture atlas is and why we'll be using it for our next game and how to create a texture atlas with the TexturePacker tool.

Chapter 7, Scrolling with Joy

With the Texture Atlas ready, you'll learn how to implement a parallax scrolling shooter game, controlled by touch input.

Chapter 8, Shoot em Up

Without enemies, our shooter wouldn't have much to shoot at, right? I'll show you how to add game-play code to spawn, move, hit, and animate the enemy hordes.

Chapter 9, Particle Effects

By using the ParticleDesigner tool, you'll add some particle effects to the side-scrolling game.

Chapter 10, Working with Tilemaps

Infinitely jumping upward, you'll apply what you've learned from the side-scrolling game in portrait mode to create another popular iOS game genre.

Chapter 11, Isometric Tilemaps

Since cocos2d supports the TMX file format, you'll take a look at how to create tile-based games using the Tiled editor.

Chapter 12, Physics Engines

Directing where things go with the move of your fingertips you'll learn here how that's done.

Chapter 13, Pinball Game

This is a primer on using the Chipmunk and Box2d physics engines and the crazy things you can do with them.

Chapter 14, Game Center

This time, you'll use real physics for a gravity-defying, planet-bouncing, ball-shooter in space. It's not going to be realistic, but it's going to have real physics. It's a conundrum, maybe, but fun in any case.

Chapter 15, Cocos2d with UIKit Views

This chapter goes into depth on how to mix and match cocos2d with regular Cocoa Touch, particularly UIKit views. You'll learn how to add UIKit views to a cocos2d game or, conversely, how to make use of cocos2d in an existing UIKit app.

Chapter 16, Kobold2D Introduction

Kobold2D is my take on improving the cocos2d game engine by adding popular libraries and combining them into a single, ready-to-use package. In this chapter, you'll learn how to set up new Kobold2D projects and the role of Lua scripting in Kobold2D, and you'll get a primer on 3D game development with cocos3d.

Chapter 17, Conclusion

This is where the book ends. Worry not, your journey won't. You'll get inspiration on where to go from here.

Where to Get the Book s Source Code?

One of the most frequently asked questions following the release of the first edition of this book was about where to get the book's source code. I've added this little section to answer this question.

You can get the book's source code on the Apress web site under Source Code/Downloads if you follow this link: `www.apress.com/9781430233039`. Alternatively, you can download the source code in the Downloads section on Cocos2D Central: `cocos2d-central.com/files/file/2-source-code`.

Of course, you can always type the code directly from the book if you prefer.