

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



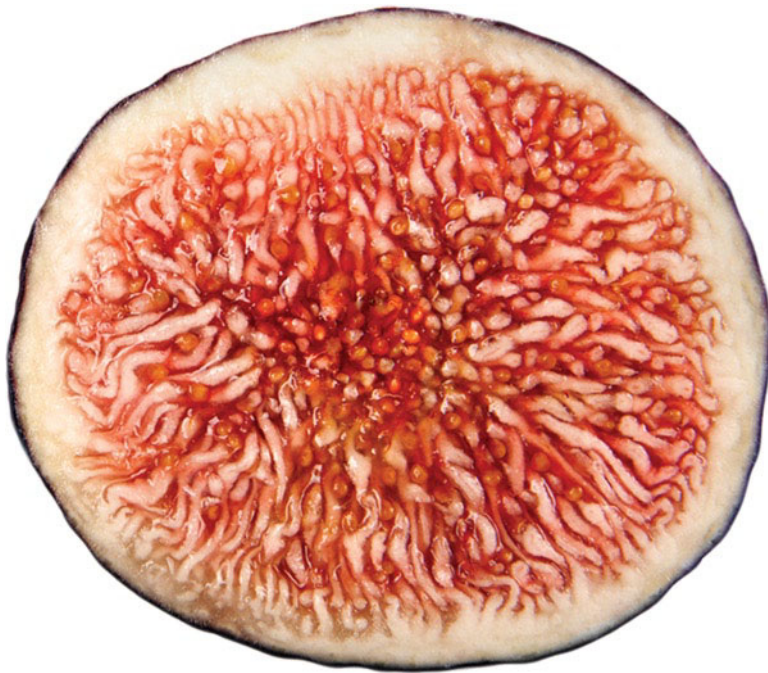
Learn cocos2d 2

Game Development for iOS

Steffen Itterheim | **Andreas Löw**

Apress®

Learn cocos2D 2



Steffen Itterheim
Andreas Löw

Apress®

Learn cocos2D 2

Copyright © 2012 by Steffen Itterheim and Andreas Löw

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN 978-1-4302-4416-5

ISBN 978-1-4302-4417-2 (eBook)

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

President and Publisher: Paul Manning

Lead Editor: Steve Anglin

Development Editor: Tom Welsh

Technical Reviewers: Boon Chew & Tony Hillerson

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Brigid Duffy

Copy Editors: Corbin Collins

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

Cover Designer: Anna Ishchenko

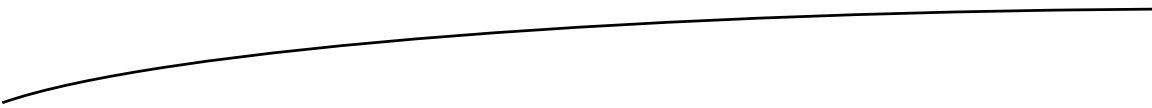
Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

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 information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

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

About the Authors	xix
About the Technical Reviewers	xxi
Acknowledgments	xxiii
Preface	xxv
■ Chapter 1: Introduction	1
■ Chapter 2: Getting Started	15
■ Chapter 3: Essentials	51
■ Chapter 4: Your First Game	101
■ Chapter 5: Game Building Blocks	133
■ Chapter 6: Sprites In-Depth	159
■ Chapter 7: Scrolling with Joy	187
■ Chapter 8: Shoot 'em Up	213
■ Chapter 9: Particle Effects	239
■ Chapter 10: Working with Tilemaps	265
■ Chapter 11: Isometric Tilemaps	291
■ Chapter 12: Physics Engines	321

■ Chapter 13: Pinball Game	347
■ Chapter 14: Game Center	391
■ Chapter 15: Cocos2d and UIKit Views	429
■ Chapter 16: Kobold2D Introduction	459
■ Chapter 17: Out of the Ordinary.....	479
Index.....	505

Contents

About the Authors	xix
About the Technical Reviewers	xxi
Acknowledgments	xxiii
Preface	xxv
■ Chapter 1: Introduction	1
What's New in the Third Edition?	2
All About ARC	2
Why Use cocos2d for iOS?	3
It's Free.....	3
It's Open Source	4
It's Objective, See?	4
It's 2D.....	4
It's Got Physics	5
It's Less Technical.....	5
It's Still Programming	5
It's Got a Great Community.....	5
Why Use Kobold2D over cocos2d-iphone?.....	6
Other cocos2d Game Engines	7
This Book Is for You	8

Prerequisites	8
Programming Experience	8
Objective-C	8
What You Will Learn	9
What Beginning iOS Game Developers Will Learn	10
What iOS App Developers Will Learn	10
What Cocos2d Developers Will Learn	10
What's in This Book	11
Chapter 2, "Getting Started"	11
Chapter 3, "Essentials"	11
Chapter 4, "Your First Game"	11
Chapter 5, "Game Building Blocks"	11
Chapter 6, "Sprites In-Depth"	11
Chapter 7, "Scrolling with Joy"	12
Chapter 8, "Shoot 'em Up"	12
Chapter 9, "Particle Effects"	12
Chapter 10, "Working with Tilemaps"	12
Chapter 11, "Isometric Tilemaps"	12
Chapter 12, "Physics Engines"	12
Chapter 13, "Pinball Game"	12
Chapter 14, "Game Center"	12
Chapter 15, "Cocos2d and UIKit Views"	13
Chapter 16, "Kobold2D Introduction"	13
Chapter 17, "Conclusion"	13
Where to Get the Book's Source Code	13
Questions and Feedback	13
Chapter 2: Getting Started	15
What You Need to Get Started	15
System Requirements	15
Register as an iOS Developer	16
Certificates and Provisioning Profiles	17

Download and Install Xcode and the iOS SDK	17
Download cocos2d or Kobold2D	18
Install Kobold2D.....	18
Create a Kobold2D Project.....	19
Install cocos2d and its Xcode Project Templates.....	22
How to Create a cocos2d Project.....	23
How to Enable ARC in a cocos2d Project.....	25
The Anatomy of cocos2d and Kobold2D Applications	34
The Supporting Files Group	36
Memory Management with ARC.....	41
Changing the World	42
What Else You Should Know	44
The iOS Devices.....	44
About Memory Usage	46
The iOS Simulator.....	46
About Performance and Logging.....	48
Summary.....	49
Chapter 3: Essentials.....	51
The cocos2d Scene Graph.....	51
The CCNode Class Hierarchy	55
CCNode.....	56
Working with Nodes	56
Working with Actions.....	57
Scheduled Messages.....	57
Director, Scenes, and Layers	61
The Director	61
CCScene	62
Scenes and Memory.....	63
Pushing and Popping Scenes	64
CCTransitionScene.....	65
CCLayer	68

CCSprite	73
Anchor Points Demystified	74
CCLabelTTF	75
Menus.....	76
Menu Items with Blocks	78
Actions	81
Interval Actions.....	82
Instant Actions.....	89
Orientation, Singletons, Tests, and API References	93
Orientation Course in Device Orientation.....	93
Singletons in cocos2d	95
Cocos2d Test Cases	98
Cocos2d API Reference.....	98
API References in Xcode and elsewhere	99
Summary.....	99
Chapter 4: Your First Game.....	101
Create the DoodleDrop Project.....	102
Start with an ARC-enabled cocos2d Project.....	102
Create the DoodleDrop Scene	104
Adding the Player Sprite.....	108
Simple Accelerometer Input.....	112
First Test Run.....	113
Player Velocity	113
Adding Obstacles	116
Collision Detection.....	122
Labels and Bitmap Fonts.....	123
Adding the Score Label.....	124
Introducing CCLabelBMFont	125
Creating Bitmap Fonts with Glyph Designer	126
Simply Playing Audio.....	128

iPad Considerations.....	129
Supporting the Retina iPad.....	129
One Universal App or Two Separate Apps?	130
Restricting Device Support.....	131
Summary.....	132
Chapter 5: Game Building Blocks	133
Working with Multiple Scenes.....	133
Adding More Scenes.....	133
Loading Next Paragraph, Please Stand By	136
Working with Multiple Layers.....	139
How to Best Implement Levels	145
CCLayerColor and CCLayerGradient.....	146
Subclassing Game Objects from CCSprite	147
Composing Game Objects Using CCSprite.....	148
Curiously Cool CCNode Classes.....	152
CCProgressTimer	153
CCParallaxNode	154
CCMotionStreak.....	156
Summary.....	158
Chapter 6: Sprites In-Depth	159
Retina Display	160
CCSpriteBatchNode	163
When to Use CCSpriteBatchNode	164
The Sprites01 Demo Project.....	164
Sprite Animations the Hard Way.....	170
Animation Helper Category	172
Working with Texture Atlases	174
What Is a Texture Atlas?	175
Introducing TexturePacker.....	175
Preparing the Project for TexturePacker.....	176

Creating a Texture Atlas with TexturePacker	177
Using the Texture Atlas with cocos2d.....	181
Updating the CCAnimation Helper Category	183
All into One and One for All.....	184
Summary.....	185
Chapter 7: Scrolling with Joy	187
Advanced Parallax Scrolling.....	187
Creating the Background as Stripes.....	187
Re-creating the Background in Code.....	190
Moving the ParallaxBackground	192
Parallax Speed Factors.....	194
Scrolling to Infinity and Beyond.....	196
Fixing the Flicker	199
Repeat, Repeat, Repeat	200
A Virtual Joypad.....	201
Introducing SneakyInput.....	202
Touch Button to Shoot	204
Skinning the Button	206
Controlling the Action	209
Digital Controls	212
Summary	212
Chapter 8: Shoot 'em Up.....	213
Adding the BulletCache Class.....	213
Let's Make Some Enemies	219
The Enemy Class	221
The EnemyCache Class	226
The Component Classes.....	229
Shooting Things.....	232
A Healthbar for the Boss	235
Summary	238

Chapter 9: Particle Effects	239
Example Particle Effects	239
Creating a Particle Effect the Hard Way	243
Subclassing CCParticleSystem.....	244
CCParticleSystem Properties.....	246
Particle Designer	255
Introducing Particle Designer	256
Using Particle Designer Effects	258
Sharing Particle Effects	259
Shoot 'em Up with Particle Effects.....	261
Summary.....	264
Chapter 10: Working with Tilemaps	265
What Is a Tilemap?	265
Preparing Images with TexturePacker.....	269
Tiled (Qt) Map Editor.....	270
Creating a New Tilemap.....	270
Designing a Tilemap	273
Using Orthogonal Tilemaps with Cocos2d.....	277
Locating Touched Tiles	281
Working with the Object Layer	284
Drawing the Object Layer Rectangles	286
Scrolling the Tilemap.....	288
Summary.....	290
Chapter 11: Isometric Tilemaps	291
Designing Isometric Tile Graphics	292
Isometric Tilemap Editing with Tiled	295
Creating a New Isometric Tilemap.....	295
Creating a New Isometric Tileset.....	297
Laying Down Some Ground Rules.....	298

Isometric Game Programming	300
Loading the Isometric Tilemap in Cocos2d	300
Set up Cocos2d for Isometric Tilemaps	300
Locating an Isometric Tile.....	302
Scrolling the Isometric Tilemap	305
This World Deserves a Better End	306
Adding a Movable Player Character.....	310
Adding More Content to the Game	319
Summary.....	319
Chapter 12: Physics Engines	321
Basic Concepts of Physics Engines.....	321
Limitations of Physics Engines.....	322
The Showdown: Box2D vs. Chipmunk.....	322
Box2D	323
The World According to Box2D	325
Restricting Movement to the Screen	328
Converting Points	330
Adding Boxes to the Box2D World	330
Updating the Box2D World.....	332
Collision Detection.....	333
Joint Venture.....	335
Chipmunk	337
Chipmunks in Space.....	337
Boxing-In the Boxes.....	339
Adding Ticky-Tacky Little Boxes	340
Updating the Chipmunk Space	341
A Chipmunk Collision Course.....	342
Joints for Chipmunks.....	343
Summary.....	345

Chapter 13: Pinball Game	347
Shapes: Convex and Counterclockwise.....	348
Working with PhysicsEditor.....	349
Defining the Plunger Shape.....	351
Defining the Table Shapes.....	353
Defining the Flippers.....	356
Defining the Bumper and Ball.....	358
Save and Publish.....	358
Programming the Pinball Game.....	358
Forcing Portrait Orientation.....	359
The BodySprite Class.....	359
Creating the Pinball Table.....	363
Box2D Debug Drawing.....	368
Adding the Ball.....	369
Forcing the Ball to Move.....	372
Adding the Bumpers.....	376
The Plunger.....	377
The Flippers.....	386
Summary.....	390
Chapter 14: Game Center	391
Enabling Game Center.....	391
Creating Your App in iTunes Connect.....	392
Setting Up Leaderboards and Achievements.....	393
AppController and UINavigationController.....	393
Configuring the Xcode Project.....	394
Game Center Setup Summary.....	397
Game Kit Programming.....	397
The GameKitHelper Delegate.....	398
Checking for Game Center Availability.....	399
Authenticating the Local Player.....	400
Block Objects.....	403

Receiving the Local Player's Friend List.....	405
Leaderboards.....	408
Achievements.....	413
Matchmaking.....	418
Sending and Receiving Data.....	423
Summary	427
Chapter 15: Cocos2d and UIKit Views	429
What Is Cocoa Touch?	429
Using Cocoa Touch and cocos2d Together	430
Why Mix Cocoa Touch with cocos2d?.....	430
Limitations of Mixing Cocoa Touch with cocos2d.....	431
How Is Cocoa Touch Different from cocos2d?	431
Alert: Your First UIKit View in cocos2d.....	433
Embedding UIKit Views in a cocos2d App	435
Adding Views in Front of the cocos2d View.....	435
Skinning the UITextField with a UIImage	438
Adding Views Behind the cocos2d View	439
Adding Views Designed with Interface Builder.....	445
Embedding the cocos2d View in Cocoa Touch Apps.....	448
Creating a View-Based Application Project with cocos2d	448
Designing the User Interface of the Hybrid App.....	450
Start Your cocos2d Engine.....	452
Changing Scenes.....	455
Summary.....	457
Chapter 16: Kobold2D Introduction	459
Benefits of Using Kobold2D.....	459
Kobold2D Is Ready to Use.....	460
Kobold2D Is Free	460
Kobold2D Is Easy to Upgrade.....	460
Kobold2D Includes Popular Libraries.....	461
Kobold2D Takes Dual-Platform to Heart	462

The Kobold2D Workspace.....	462
The Hello Kobold2D Template Project	464
The Hello World Project Files	464
How Kobold2D Launches an App.....	466
The Hello Kobold2D Scene and Layer.....	469
Running Hello World with iSimulate	473
DoodleDrop for Mac with KKInput	474
Summary	477
Chapter 17: Out of the Ordinary.....	479
Additional Resources for Learning and Working	480
Where to Find Help	480
Source Code Projects to Benefit From.....	482
Cocos2D Podcast.....	487
Tools, Tools, Tools.....	488
Cocos2d Reference Apps.....	489
The Business of Making Games	492
Working with Publishers.....	492
Finding Freelancers.....	493
Finding Free Art and Audio	494
Finding the Tools of the Trade.....	494
Marketing	495
Engaging Players for More Revenue.....	498
Summary.....	502
Index.....	505

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 industry. 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 Reviewers



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.

Tony Hillerson is a mobile developer and cofounder at Tack Mobile. He graduated from Ambassador University with a bachelor's degree in Management Information Systems. On any given day, he may be working with Objective-C, Java, Ruby, CoffeeScript, JavaScript, HTML, or shell scripts. Tony has spoken at RailsConf, AnDevCon, and 360|Flex. He is the creator of the popular O'Reilly Android screencasts.

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

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, Corbin Collins and Brigid Duffy, the coordinating editors. When chaos ensued, they were the ones 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 tremendous 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.

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 five months in 2010 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 the many months updating the chapters to the second and then the third 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 I can provide a solution for cocos2d that removes most of the initial hurdles for cocos2d developers. The result of that is Kobold2D, which you'll find an introduction to in Chapter 16 and of course on www.kobold2d.com. What sets it apart from cocos2d is that it has an installer, all the documentation, bundles the important libraries, has ARC enabled, over a dozen example projects and additional features on top of that. Obviously one of my goals for the third edition was to make it 100% compatible with Kobold2D 2.0.

Steffen Itterheim

Chapter 1

Introduction

Have you ever imagined yourself writing a computer game and being able to make money selling it? With Apple's iTunes App Store and the accompanying mobile iPhone, iPod touch, and iPad devices, doing that is 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 come from a huge variety of backgrounds. Some, like me, have been professional game developers for years, or 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 pays homage to that, yet doesn't forget about the tools that 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 (application programming interface). 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 that 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 because 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 should definitely visit the book's companion web site at www.learn-cocos2d.com. Since the first edition of this book, I began improving cocos2d with the Kobold2D project, which you can also use while reading this book. You can learn more about Kobold2D by visiting www.kobold2d.com.

What's New in the Third Edition?

This third edition is a complete overhaul to bring the book up to date with the latest developments, including but not limited to iOS 6 and Xcode 4.4.

For one, cocos2d 2.0 is fresh out of the box. All the source code and descriptions have been adapted to the latest incarnation of cocos2d and OpenGL ES 2.0. Although you cannot deploy cocos2d 2.0 apps on 1st- and 2nd-generation devices, in March 2012 these devices only made up about 16% of all iOS devices sold thus far. There's no doubt that by the time you're reading this book this number will have gone down to around 10% and be still falling as the newer devices continue to sell at an ever increasing rate.

I've also received plenty of questions from many readers all wanting to know the same thing: can the book be used with Kobold2D as well? Previously I had to say "Yes, but there are a few things that are slightly different." The third edition changes my answer to a resounding "Yes, by all means!" Whenever there's anything that works differently in Kobold2D, I mention that. I also highlight all the things that you don't need to do anymore if you were using Kobold2D, because that's what Kobold2D is all about: making cocos2d easier to use.

In case you passed on the second edition, here's a quick recap of what was new in the second edition and was further adapted for this edition. First Andreas Löw joined as a co-author for the book, providing valuable help to improve the book with instructions for his Texture Packer and Physics Editor tools. Together we overhauled a lot of the graphics and significantly improved several chapters with additional code and new features. And two new chapters were added: one about integrating cocos2d in a UIKit app and the other an introduction to the (then newly released) Kobold2D.

All About ARC

Then there's ARC. Woof? What? ARC? Yes, automatic reference counting (ARC) is Apple's new and proven technology to simplify memory management for Objective-C applications. In essence it does away with reference counting, meaning you no longer have to concern yourself with remembering how many times you or other code has retained an object, requiring you to release it the exact same number of times. Autoreleasing objects further complicated that matter. If you didn't get retain, release, and autorelease 100% matched up correctly every time for every object, you were either leaking memory or the app would be prone to crashing.

It's also good to know that ARC is not garbage collection. It works fully deterministic, which means if you run the same program through the exact same process every time, it will behave exactly the same. ARC is also not a runtime component—it's the compiler inserting retain, release, and

autorelease statements automatically for you whenever they are needed. There is a set of simple rules followed by ARC that every Objective-C programmer previously had to follow to avoid memory leaks and crashes. You can imagine that if a human being does that job, a lot more errors sneak in, whereas ARC is not only able to perform this job without fail, it also optimizes your code along the way. For example, in manual reference counting you had the ability to retain objects multiple times. ARC, however, realizes when additional retains are unnecessary and omits them.

Now, the compiler has taken over the tedious job of retaining and releasing objects in memory, and all the source code throughout the book has been converted to use ARC. This means fewer code lines to write, and fewer potentially lethal drugs ... err, bugs.

The two biggest concerns from Objective-C programmers towards ARC are loss of control and learning a new programming paradigm. Both are terrible arguments against using ARC. I offer an analogy: think of ARC being like automatic transmission in a car, and manual reference as shifting gears manually. If you leave the world of the clutch and gear stick, are you really giving up control? Yes, but it's not control that you really need. Do you need to learn something new? No, you only need to unlearn a few things and you actually have to do less.

With manual transmission, you can easily miss a gear even after years of driving, possibly damaging the engine. That is equivalent to over-releasing or over-retaining an object. You may also kill the engine with the clutch every now and then. That's the equivalent of crashing your app due to a dangling (over-released) pointer. And with an automatic transmission, the engine always changes gears at the optimal rate, typically reducing the fuel consumption rate to a comparable manually shifted car (and a typical driver). Wasting gas is the equivalent of leaking memory.

To sum it up: ARC does not take control away from you. You can still control everything that is important to your app. Code that requires more memory management control than is available to you under ARC does not exist. The issue of loss of control is purely psychological and possibly based on misconceptions about how ARC works. Which brings me to the learning aspect: yes, there are a few things you can and should learn about ARC. But this is so little compared to what a programmer learns every day, and it's far easier than picking up (let alone mastering) manual reference counting if you're new to Objective-C. You only need to read Apple's relatively short Transitioning to ARC Release Notes summary: <http://developer.apple.com/library/ios/#releasenotes/ObjectiveC/RN-TransitioningToARC/Introduction/Introduction.html>. You may also want to read my blog post covering everything you need to know about automatic reference counting: www.learn-cocos2d.com/2011/11/everything-know-about-arc.

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 also create Mac OS X apps with it. You don't have to pay any 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.

It's Objective, See?

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 easy to understand Apple's documentation and implement iOS SDK functionality.

A lot of other useful APIs, such as Facebook Connect and OpenFeint, are also written in Objective-C, which makes it easy to integrate those APIs, too.

Note I advise learning Objective-C, 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 currently offer.

It doesn't 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. Unfortunately, cocos3d is not compatible with cocos2d 2.0 at this point since cocos3d is still using OpenGL ES 1.1.

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, and even a lot of 3D games are essentially 2D games, gameplay-wise. 2D games are generally easier to develop, and the algorithms in 2D games are easier to understand and implement, making them ideal for beginners. In almost all cases, 2D games are less demanding on hardware, allowing you to create more vibrant and more detailed graphics.

It's Got Physics

You can also choose from two physics engines that are already integrated with cocos2d. On the 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 in the two products. 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. For most developers, that tends to be Box2D. On the other hand Chipmunk has a commercial Pro version which, among other things, gives you a native Objective-C interface to its API.

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. 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, including vertex and fragment shaders, at any time for any game object that needs it. Shaders are a way to program the graphics hardware and are beyond the scope of this book. 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 OpenGL 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 requiring truly excellent programming skills first and foremost. Other iOS game engines such as Unity, Unreal, iTorque 2D, and Shiva focus their efforts on providing tool sets and workflows to reduce the amount of programming knowledge required. In return, 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 who will answer a question quickly, 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 on my own forum Cocos2D Central (<http://cocos2d-central.com>). Cocos2D Central is the best place to reach me personally.

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

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 and Kobold2D as well: <http://twitter.com/gaminghorror>

<http://twitter.com/kobold2d>

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.

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.

Why Use Kobold2D over cocos2d-iphone?

First of all, it's a lot easier to get started with cocos2d-iphone development if you use Kobold2D from the start. You only need to run the Kobold2D installer to get everything you need: the source code, additional useful source code libraries, the template projects, the documentation, the tools, and more. You can start a new project right away.

You get 15 template projects instead of just 3, and all of them are ARC enabled. You'll recognize that most of the template projects are variations of the projects you'll create throughout the book. So you'll feel right at home.

The most commonly used source code libraries are also integrated and ready to use. This includes the cocos2d-iphone-extensions project, the Lua scripting language, ObjectAL, iSimulate, SneakyInput, and more.

Kobold2D has additional convenience features that cocos2d-iphone doesn't have. There's helper code for Game Center, Ad Banners, Gesture Recognizers, PiXEL-Perfect Collision Detection, and simplified user input handling. KKInput gives you an easy-to-use, platform-independent wrapper to use Gesture Recognizers, the Gyroscope, Keyboard, and Mouse.

Additionally, most Kobold2D projects have targets for both iOS and Mac OS X. So if you plan on releasing your game to both App Stores, Kobold2D makes this dual-platform development a lot smoother and offers other enhancements under the hood, such as optimized Build Settings for each platform.

Then there's KoboldScript. It's a modern, powerful, yet simple Lua interface backed by Kobold2D to develop iOS and Mac OS X apps with. At the time of this writing it's still a work in progress, so please visit www.koboldscript.com for the latest news and updates.

Finally, I use Kobold2D each and every day for my own work. I continue to tweak and improve it, and I make sure it stays up-to-date with the latest developments, be they new iOS devices, new Xcode versions, or simply new releases of cocos2d and the other libraries provided by Kobold2D. You'll get a timely update and you'll be able to upgrade your projects safely to a newer Kobold2D version with just one mouse click.

Other cocos2d Game Engines

You may have noticed that versions of cocos2d exist for a great variety of platforms, including Windows, JavaScript, and Android. There's even a C++ version of cocos2d dubbed cocos2d-x that supports multiple mobile platforms all in one, 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 by a lot. Porting your cocos2d-iphone 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 in name and philosophy only. If cross-platform development is your goal, you should take a look at cocos2d-x, which has most of the features of cocos2d-iphone, is backed financially by China Unicom, and continues to be updated and improved at an incredible pace.

In any case, I think 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 didn't include in this list cocos2d ports that are significantly out of date and haven't been updated for months, if not years. That includes the defunct cocos2d for Windows project, whose only release dates back to May 2010, and the long obsolete ShinyCocos, a Ruby Wrapper based on cocos2d-iphone v0.8.2.