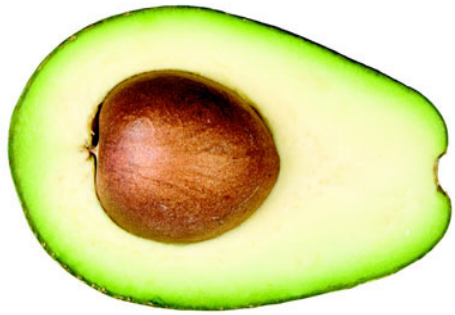
Beginning iPhone Games Development



Peter Bakhirev
PJ Cabrera
Ian Marsh
Scott Penberthy
Ben Britten Smith
Eric Wing

Apress®

Beginning iPhone Games Development

ISBN-13 (pbk): 978-1-4302-2599-7

ISBN-13 (electronic): 978-1-4302-2600-0

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Publisher and President: Paul Manning
Lead Editor: Clay Andres
Developmental Editor: Jeffrey Pepper
Technical Reviewer: Peter Bakhirev, Eric Wing and Ben Britten Smith
Editorial Board: Clay Andres, 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 Editor: Marilyn Smith
Compositor: MacPS, LLC
Indexer: Potomac Indexers
Artist: April Milne
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 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. You will need to answer questions pertaining to this book in order to successfully download the code.

To the dearest people, my family: Mom, Dad, Olya, Mike and Lena
– Peter Bakhirev

To my family, for always being supportive of my dreams.
– PJ Cabrera

y loving wife Gina, who puts up with my addiction to iPhone game development.
– Ian Marsh

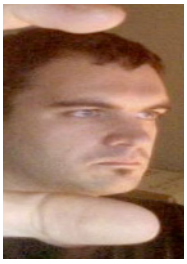
To my lovely wife Leonie.
– Ben Britten Smith

With great love, to my mother, who lost her fight to GIST cancer shortly before this book came to press. Her courage and determination will always be an inspiration to me.
– Eric Wing

Contents at a Glance

Contents

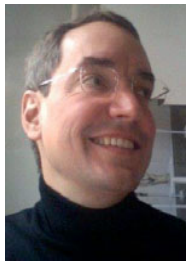
About the Authors



Peter Bakhirev is a longtime software developer, with over a decade of experience in Internet technologies and network programming, and an aspiring writer and entrepreneur. During the pre-iPhone era, he helped architect and implement one of the largest online poker sites. More recently, he participated in the creation of one of the first multiplayer games for the iPhone called Scramboni.

PJ Cabrera is a software engineer with more than 12 years of experience developing information systems in various industries, programming in C, C++, Java, PHP, Python, Ruby and Objective-C. He lives in the San Francisco Bay area and works as an iPhone and Rails developer for a stealth mode start-up in San Francisco.

Ian Marsh is the co-founder of the independent game studio NimbleBit based in San Diego, CA. He has been developing games for the iPhone since the advent of the App Store, with such successes as the #1 kids game "Scoops" and the #1 free game "Hanoi". When not developing games, Ian enjoys reading about science, tweeting about game development, and finger painting.



Scott Penberthy began coding shortly after the Apple II was launched in the 70's. His addiction to writing software fully bloomed with a scholarship to MIT, where he wrote a multiplayer online game that brought his school's antique computer to its knees. After graduating, Scott took a job at IBM Research, the birthplace of IBM's web products and services. After running up the corporate ladder in the 90's building massive web sites, he jettisoned in 2005 to return to his true love of coding. Now a successful entrepreneur, Scott runs an app studio in New York City.

Ben Britten Smith has been writing software on Apple platforms for 15 years. Most notably he was given an Academy Award for Technical Achievement for his feature film work with Mac-based suspended camera control systems. Lately he has switched his efforts from the big screen to the small screen.

His first iPhone game, "SnowDude" was published to the App Store a few months after the SDK became available. Since then he has written a dozen apps for various clients including the games: "Snowferno", The award winning: "Mole - A quest for the Terracore Gem", and the "Gambook Adventures" series of games. Ben lives in Melbourne, Australia with his wife Leonie and their pet bunnies.

Feeling he was living too extravagant of a lifestyle of ramen and subsidized bus passes, **Eric Wing** graduated (kicking and screaming) from the University of California at San Diego with a Masters degree in Computer Engineering just days before 9/11. In the following challenging world, he worked a wide range of jobs in the field from automated testing on satellite systems to scientific visualization with a variety of different operating systems and programming languages. But in a stroke of genius (actually, it was more likely just a stroke), he figured out how he could work even harder for no money and started working on open source projects. He has been a contributor to projects such as SDL (Simple DirectMedia Layer), OpenSceneGraph, and the Lua/Objective-C Bridge (and its successor LuaCocoa). And when he was offered a co-authorship of Beginning iPhone Games Development, how could he possibly refuse the idea of even more hard work for virtually no pay? It was a match made in heaven!

About the Technical Reviewer

The chapters of *Beginning iPhone Games Development* were peer-reviewed by the authors themselves. Peter Bakhirev reviewed chapters by Ian Marsh, PJ Cabrera, and Ben Britten Smith. Eric Wing reviewed the chapters written by Peter Bakhirev, and Ben Britten Smith was responsible for the tech review of Scott Penberthy's chapter.

Acknowledgments

Peter Bakhirev
Writing this book is part of a journey that started long time ago. Many better, kinder and smarter people than myself stood along the way and made this journey possible. Papa, thanks for putting that first computer together, for that tape with Basic on it, and for the wisdom (and all of the games, too). Mike, thanks for giving me a chance at that interview, for teaching me everything that I know about programming today, and for always being there ("We should do this again some time"). Lenny, thanks for believing that we could go from A to B (otherwise, I'd be going from 9 to 5 instead of writing this). Mama and Olya, thanks for the encouragement and asking about how the book is going almost every day. Lena, thanks for being and for believing, ILU and IWTHKWU. Thanks to Keith Shepherd for the introduction to Clay and Apress. Thanks to Clay Andres, Kelly Moritz, Douglas Pundick, Marilyn Smith and all of the wonderful people at Apress for pushing and pulling and making this happen.

PJ Cabrera
My gratitude goes out to Peter Bakhirev, Jeff Pepper, and Marilyn Smith for their help improving my chapters. Many thanks to Kelly Moritz and Dominic Shakeshaft for their patience. Kudos to Clay Andres and Douglas Pundick for their support. And last but not least, to my friends in the good ol' SNAP team, Kevin, Lenny, Ernesto, Cesar, Alejandro, Javier, Xavier, Edwin, thanks for keeping computing fun amidst a sea of chaos.

Ben Britten Smith
Ben would like to thank Neil from Tin Man Games for the use of his art assets.

Eric Wing
I want to acknowledge and give my everlasting thanks to a wonderful group of volunteers who reviewed my chapters.

First, my thanks to Daniel Peacock and Ryan Gordon for bringing to bear the full force of their technical expertise of OpenAL and audio in general to catch any mistakes and omissions. Also my thanks to Ian Minett for providing details about the Xbox implementations of OpenAL and Garin Hiebert for tracking down some obscure information concerning OpenAL for me.

Next, I wish to thank Josh Quick, Victor Gerth, Carlos McEvilly, and Wang Lam who reviewed my chapters from the perspective of the general audience for this book, thus enabling me to improve things before the book went to press.

Finally, there are several people who wish or need to stay anonymous. I wish to thank them nonetheless because their contributions were also significant.

And of course, my thanks to my co-authors and to the people at Apress for all their support.

Introduction

Hey there, curious reader! My name is Peter, and I'd like you to meet my fellow co-authors Ian, PJ, Scott, Ben and Eric. We are here for one simple reason: to help you learn how to make awesome iPhone games. You have probably noticed the word "beginning" in the title of this book. Here is what that means: you can develop games even if you have never thought of yourself as a "game developer" before. Even though it might seem like "rocket science" at first, in reality it is anything but (although one of the games that we will build in this book does have a rocket in it, along with spectacular explosions, out-of-this-world sounds and evil aliens.) We believe that anybody can learn the necessary skills, and we'll walk you through all of the steps and explain everything as we go along.

Speaking of ideas and games, we have a whole bunch of those for you to play with. This book includes half a dozen fully playable games that we will help you develop. In the process, you'll learn how to build 2D and 3D games with music, sound effects, support for multiple players and networking. But the question of "what to build" is at least as important as "how to build it", and you will find plenty of discussion about how to design something that's fun, as well.

In case you haven't developed for the iPhone before and need a crash course in how to use the tools of the trade, we have included a brief introduction to Objective-C and Xcode, but you can find a much more in-depth exploration of the iPhone development environment in "Beginning iPhone 3 Development" by Dave Mark and Jeff LaMarche, which we highly recommend. If you need a more thorough introduction to programming in general or C and Objective-C languages in particular, take a look at "Learn C on the Mac" by Dave Mark and "Learn Objective-C on the Mac" by Mark Darlymple and Scott Knaster.

Writing, coding and debugging this book was a lot of fun, and we hope that you will find it enjoyable and engaging. Good luck and see you in the App Store!

On behalf of the author team,

Peter Bakhirev

Chapter 1

A Revolutionary Gaming Platform: Games for Everyone, Anytime, Anywhere

The iPhone platform has drastically changed the landscape of next-generation mobile gaming. The iPhone is a device of many firsts, and its multifaceted nature is what pushes it above and beyond traditional mobile gaming platforms. The iPhone's ubiquity, connectivity, personal integration, popularity, and innovative interface make it one of the most potent and exciting platforms to develop for today.

The Ever-Present iPhone

Because the iPhone platform is first and foremost a mobile phone and/or digital music player, the vast majority of iPhone and iPod touch owners carry their devices around with them everywhere they go, every day. The combination of a phone, music player, and game player into a single, svelte package means that people no longer need to choose which devices to take with them each morning. This fact makes the iPhone a groundbreaking game platform.

For iPhone and iPod touch owners, any game is only seconds away—whether they're waiting at a gas pump or sitting on a trans-Atlantic flight. A quick glance around any public place illustrates the iPhone's proliferation and the quickly expanding market for iPhone games. For the gamer on the go, there's no better use of pocket space than the iPhone.

With games always within reach for players, developers can design either quick "pick-up-and-play" games or longer "appointment" games that may take 30 minutes or more to play. Great "time-waster" games, such as Veiled Game's Up There (see Figure 1–1),

rarely last for more than a couple minutes and appeal to casual gamers who can't dedicate long spans of time to gaming. Others enjoy spending time in absorbing titles like Electronic Arts' SimCity (also shown in Figure 1–1). Either type of gamer can be an iPhone owner, and developers can even design a game that caters to both types. One thing is for certain: As an iPhone game developer, you can count on your game being carried around by your users every waking minute of their day.

Figure 1–1. *Up There offers players a quick balloon-soaring game. Fans of simulation games can easily spend long stretches of time playing SimCity.*

Always having your game on hand not only means it will get more play time by your users, but it also helps to market your game. One of the best ways a high-quality iPhone game can be advertised is through word of mouth. iPhone users are likely to associate with other iPhone users. With their entire game collection right there in their pockets, gamers can show off their favorite games in an instant, and their friends can buy the game for themselves in a matter of seconds.

Mass Appeal—There's a Gamer Born Every Minute

Users are drawn to the iPhone for many reasons besides gaming. Some people are primarily interested in its web-browsing capabilities and multimedia features. But even those who have no history of playing video games find the App Store and its thousands of games very appealing.

The App Store's ease of use, combined with the availability of so many games, can turn anyone into a gamer, casual or otherwise. A developer can create games that will be enjoyed by all types of people—from a child on a car ride, to a Halo fan away from his Xbox, to a grandfather relaxing in his chair. The iPhone makes your games available to

people who previously never considered gaming important enough to warrant buying one on any device.

The diversity of iPhone apps is actually blurring the definition of what a game is. Entertainment apps such as Snappy Touch's Flower Garden, The Blimp Pilots' Koi Pond (see Figure 1–2), and Bolt Creative's Pocket God are popular. These interactive experiences may not be games by literal definition, but they share many elements with games and can attract a huge fan base.

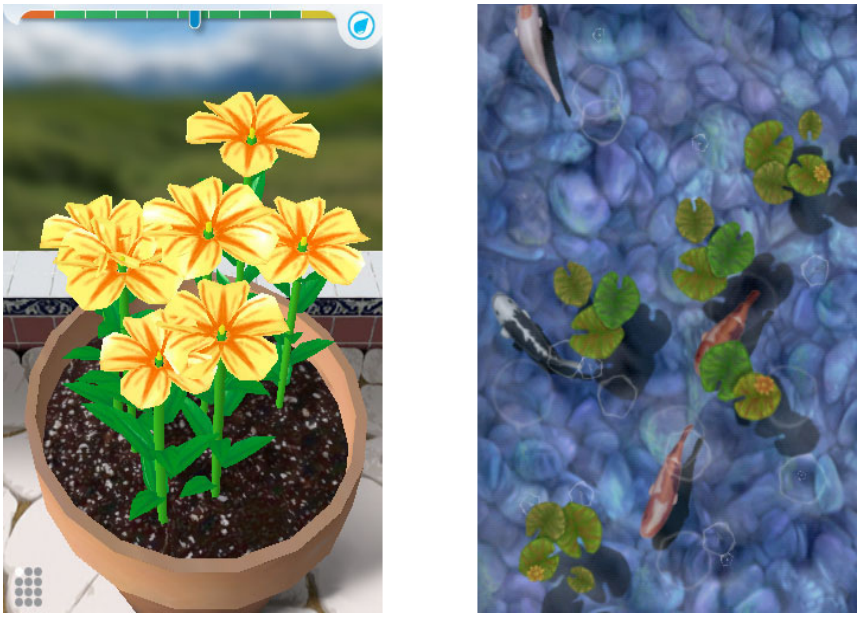


Figure 1–2. *Flower Garden lets users manage a virtual garden and send flowers to friends. Koi Pond presents users with an interactive virtual koi pond.*

Because many developers, rather than publishers, are deciding what kinds of apps to make, innovation and experimentation are running rampant. With so many potential customers and different types of gamers, there can be a market for almost any type of game—whether it's classic or something the world has never seen.

As an iPhone game developer, your customer base is growing every day and shows no sign of slowing down. Even when upgrading their iPhones or iPod touches, your customers can be sure to hang on to your games due to an easily supported standardized model lineup—a rarity in traditional cell phone gaming. Your game can target each and every iPhone and iPod touch gamer, regardless of which model they own. Even with a hugely successful title, it's impossible to saturate the market since there are new iPhone gamers born every minute.

User Interfaces—Death of the D-Pad

The iPhone's user interface is another part of the equation for this revolutionary platform. In the same way the Nintendo Wii made console gaming accessible to the general public's touch screen, accelerometer, camera, and microphone let game developers create intuitive natural interfaces to their interactive experiences.

With the tilt, touch, and microphone controls at their disposal, developers can make controls for their games transparent, creating an experience that is both immersive and easily comprehensible. Directly manipulating game objects with your finger or moving the physical device in your hands provides a visceral game interface to users. No longer is there a learning period of mentally mapping game buttons, pads, and joysticks to game actions. Interfaces on the iPhone must be immediately obvious to be usable at all.

Developers are utilizing these nontraditional control methods in new and unexpected ways. The iPhone's unique user interfaces have even spawned entirely new genres of games. Angling devices from side to side controls tilt games such as Lima Sky's Doodle Jump and NimbleBit's Scoops (see Figure 1–3). Multitouch games like Bed Bugs by Igloo Games put the players' concentration to the test by forcing them to manipulate multiple game objects at the same time. Entertainment apps Ocarina and Leaf Trombone by Smule allow users to play virtual instruments by blowing into the iPhone microphone.

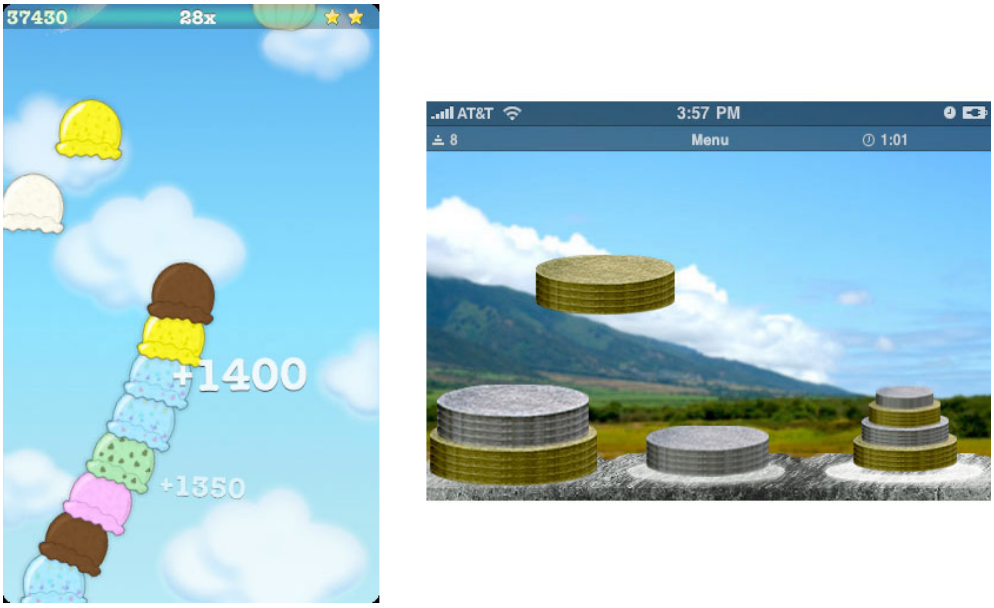


Figure 1–3. *Scoops utilizes the iPhone's accelerometer to control a wobbling tower of ice cream scoops. Hanoi's natural interface lets players drag and drop game pieces.*

An iPhone game's interface can *be* the game, by presenting such a compelling and natural means of interaction that it becomes invisible to the user. A great example of the iPhone's capability for transparent user interfaces is NimbleBit's Hanoi (also shown in

Figure 1–3). Hanoi is the classic Towers of Hanoi puzzle used in many computer science algorithm classes. On most other platforms, this simple game would need to let users select a disk, perhaps with a directional pad and button, indicate on the screen which piece is selected, and then give users a way to drop the selected disk. When you think about it, this seems like a relatively complex interface to such a simple real-world task. In Hanoi for the iPhone, the user manipulates the pieces in a more natural manner: The user simply touches a disk, drags it to the correct position, and lets go. No directional pads, buttons, joysticks, or styluses are required.

Connectivity—Plays Well with Others

Another feature setting the iPhone apart from past mobile gaming platforms is its unprecedented connectivity. Developers targeting the iPhone (and iPod touch) can count on their game almost always having a data connection. This means that games on the iPhone can utilize Internet connectivity—whether it is a core part of the game play (multiplayer), a feature that improves an offline game (high score boards), or integration with social networks such as Facebook and Twitter. Games like Baseball Slugger by Com2uS (see Figure 1–4) pit you in a head-to-head contest against another player anywhere in the world from the comfort of your living room, a park bench, or a bus seat.

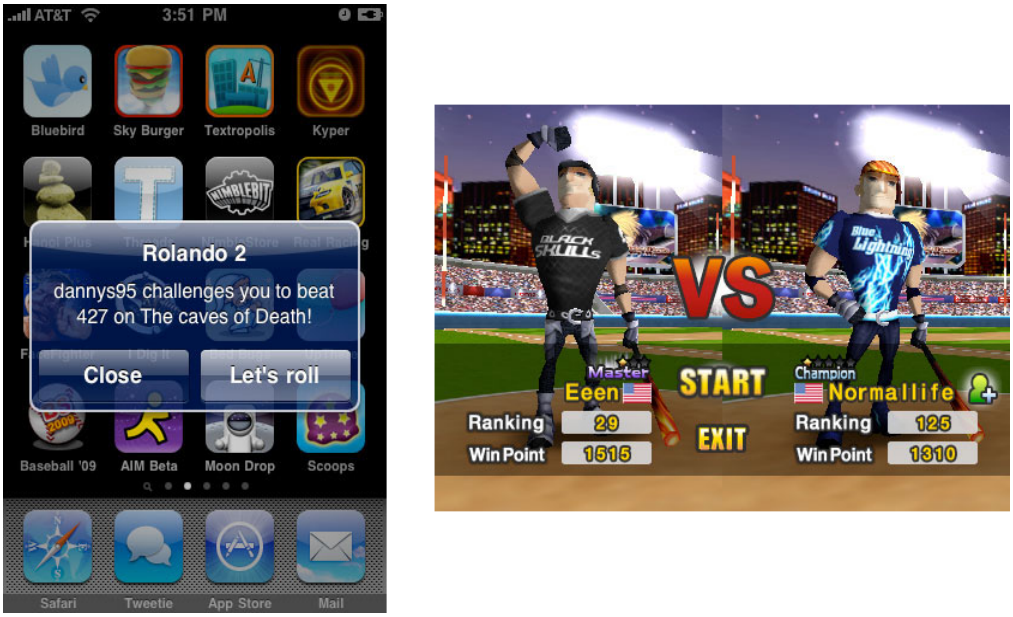


Figure 1–4. *Rolando 2 uses push notifications to challenge friends even when they aren't playing. Baseball Slugger lets you challenge players from around the world, anytime, anywhere.*

Even when devices are away from an Internet connection, a developer can use Bluetooth to connect players. With the addition of push notifications, you can keep your users involved and part of the game even when they're not playing. Games not designed for real-time multiplayer interaction, like Rolando 2 by ngmoco:) (also

shown in Figure 1–4), utilize iPhone's connectivity by sending asynchronous challenges to other players. This type of technology is perfectly suited for things like notifying players when they're up in turn-based games, have been challenged by an opponent, or their record score has been dethroned.

Being a connected device not only improves games themselves, but also makes your games available to purchase and download at anytime and from anywhere. On traditional mobile platforms, gamers needed to visit retail stores and hope the game they wished to purchase was in stock. Previously, a good amount of time and effort stood between a customer's money and your games. With the live App Store only a tap away on the iPhone, it is easier and quicker than ever for developers to get their games on users' devices. Even with the recent arrival of digital distribution for some other mobile game platforms, the App Store's use of iTunes accounts avoids the need for purchasing or refilling points or credits to buy games; purchasing a game is as painless as buying a song. With the addition of support for paid downloadable content, developers also have multiple ways to monetize their creations.

The connectivity of the iPhone opens up a huge number of possibilities for game developers. They can create games that are dynamically updated with real-time data or new content. Not only can players play others across the world, but they can also voice chat with them in real time. A developer can integrate any number of Internet technologies into a game—be it social networking or streaming audio and video. Developers can even learn valuable things about how their games are being played by recording and retrieving usage data.

User Data—This Time It's Personal

The iPhone is the first gaming platform to have access to a wealth of personal information. Having the user's contacts, music, photos, videos, and location available for game developers to access opens the door for extremely personalized and customized experiences. How many other gaming platforms know who your friends are? With access to a user's contacts, iPhone games can use the names of friends and family for characters, contact them directly, or make multiplayer matchup a breeze. In ngmoco:)'s Dr. Awesome (see Figure 1–5), patients admitted to your hospital take the names of contacts from your address book, giving the game a very personal feel.

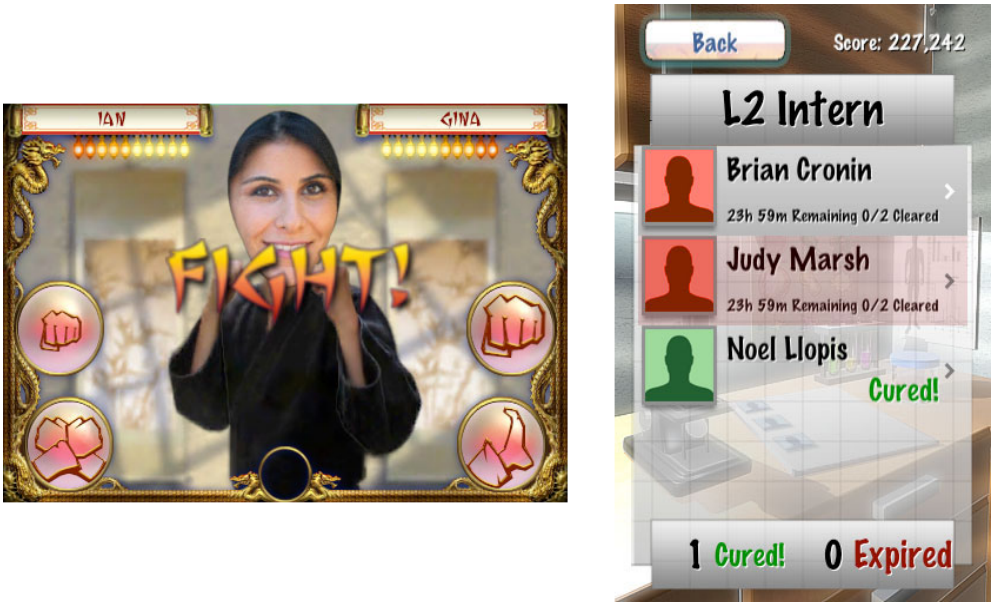


Figure 1–5. *Face Fighter players can customize their opponents with photos on their iPhone. The patients admitted in Dr. Awesome by ngmoco:) are the player's contacts.*

Not only can iPhone owners carry around their music collection in their pockets, but they can also choose their own game soundtracks from their favorite music. Players can customize their game's characters or environments with pictures or video taken or stored on their device.

In addition to all of this personal information, developers also have access to the device's physical location. While location-based games are still in their infancy, location can be used in more passive ways to match up nearby players or show maps of users worldwide.

Other devices have toyed with integrated or external cameras as a way to allow players to customize their games, but most fell short due to the fact you could use only pictures taken with that particular camera. Games and applications on the iPhone not only have access to the integrated camera, but also can use images saved from the browser, contained in an e-mail, saved to a contact, or sent in a multimedia message. This means that in games such as Appy Entertainment's Face Fighter (also shown in Figure 1–5), you can battle kung fu style against your best friend, a coworker, or a celebrity whose photo you saved from a web page.

With the ability to tap into an iPhone owner's music, photos, videos, friends, and location, game developers have unprecedented access to the personal lives of their players. In the right situations, iPhone games can allow customization or use fragments of personal information to evoke a more emotional response or give a greater feeling of ownership in the game. Used appropriately, access to this user data is another valuable tool in the inventive iPhone developer's toolbox.

Device Performance—A Multimedia Powerhouse

Most important to many developers is the power of the hardware. The processing power of a mobile gaming platform determines the extent that 3D graphics, physics simulations, and other technologies can be utilized in any game. Lucky for us, the iPhone is no slouch in this department. Veteran developers put the abilities of pre-3GS devices ahead of the Nintendo DS, and on par with the Sony PlayStation Portable (PSP). With the introduction of the 3GS model, the iPhone has advanced to the front of the pack in terms of hardware capability and performance.

With faster processers, more memory, and advanced 3D graphics support, future iPhone and iPod hardware will push the limits of what mobile games are capable of achieving. Each improved device will be able to create more polygons and simulate more physics. Established developers such as Electronic Arts, Gameloft, and Firemint have already produced iPhone titles that rival or surpass similar DS and PSP games (see Figure 1–6).

Figure 1–6. *Games like Firemint's Real Racing and Gameloft's Let's Golf show off the graphical power of the platform.*

With support for common game technologies such as OpenGL ES and OpenAL, experienced developers can make graphically rich experiences in no time. Since the iPhone Software Development Kit (SDK) supports C/C++ in addition to Objective-C, many existing libraries and a lot of game code can be reused or ported with little (or no) effort.

Along with the powerful hardware inside the devices, developers appreciate the presentation aspects. One glance at the crisp, high-resolution, glass-covered display will make developers forget the cheap, tiny screens of old mobile gaming. At 320 by 480 pixels, the screen is a wonderful canvas for high-quality game art and beautifully rendered 3D environments. The low response time of the display prevents any image ghosting, even with games running at 60 frames per second (fps). The smooth glass shield with the integrated capacitive touch screen makes tapping, dragging, pinching, and zooming a pleasure in any game.

To complete the package, the iPhone's hardware gives developers powerful audio capabilities. With hardware audio decoding, the CPU can concentrate on game play processing instead of background music. And when the highest quality lossless sounds

are needed, there is plenty of disk space to hold them. While the single mono speaker might leave something to be desired, developers can count on most users having headphones on hand for truly immersive audio experiences.

Dev Kit? You're Holding It!

By far, the most revolutionary thing about the iPhone as a gaming platform is the fact that nearly anyone can develop for it. This fact alone dwarfs every other feature of the platform. A platform might have the most amazing groundbreaking features the gaming world has ever seen, but it's all for nothing if you're prevented from developing for it.

Traditionally, console manufacturers such as Nintendo, Sony, and Microsoft have put a huge number of restrictions on who can even purchase development kits. Developers must apply and provide detailed company information and design docs for the games they wish to make. It is then up to the manufacturers to decide whether the developers are competent enough and whether their game ideas are worthwhile. If you are lucky enough to become an authorized developer for one of these platforms, you're then hit with dev kit, licensing, testing, and certification fees—totaling thousands of dollars.

While some may complain that the iPhone's App Store is still a "walled garden," it is nowhere near as restrictive as the exclusive platforms of the past. All that Apple requires of its developers is a small annual fee. Apple doesn't care how many employees you have, how you're funded, or what kind of games you intend to make. Anyone—whether an experienced team or a single student just learning game development—can create games that will stand alongside titles made by huge developers like Electronic Art, Sega, and Gameloft. Shelf space is given to every developer free of charge, thanks to digital distribution, and any developer's game has the chance to be placed on the "end cap" of the App Store by being featured (see Figure 1–7).

Cost of development for the iPhone is significantly cheaper as well. Even if you don't already own an Intel-powered Mac, the total cost of computer, device, and developer fees would run around a thousand dollars or less. For most platforms, the development kits alone cost the same or more before adding fees for testing and certification.

Unlike traditional platforms, Apple is totally hands-off when it comes to the actual development itself. Apple doesn't require design docs or set milestones for you to meet. The only time Apple enforces its liberal requirements on your game is when it's finished and submitted to the App Store. The iPhone is the first popular gaming platform to be open to all developers.

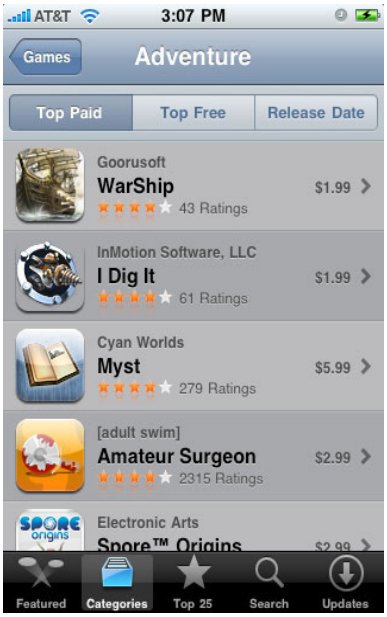


Figure 1–7. *Games from small independent developers are in direct competition with those made by large and established developers and publishers.*

Innovation—Good Things Come from Small Developers

Innovation comes hand in hand with an open platform. The number and variety of people developing for the iPhone lead to the creation of things never before seen or even imagined. The iPhone is a wonderful new outlet for independent game developers to deliver their work to the world. Radical new games that would have a hard time seeing the light of day on a normal console face no such difficulty on the App Store.

Unique and innovative games such as Steph Thirion's Eliss and Mobigame's Edge (see Figure 1–8) can make it to market and start generating waves much quicker than would be possible on any other platform. Titles from relatively unknown developers can become hits overnight if they attract a large fan base that spreads news of the game by word of mouth, or have their games featured by Apple on iTunes or in a TV commercial. The App Store is one of the only platforms where games created by independent developers compete with the larger established developers and publishers, and even outnumber them.

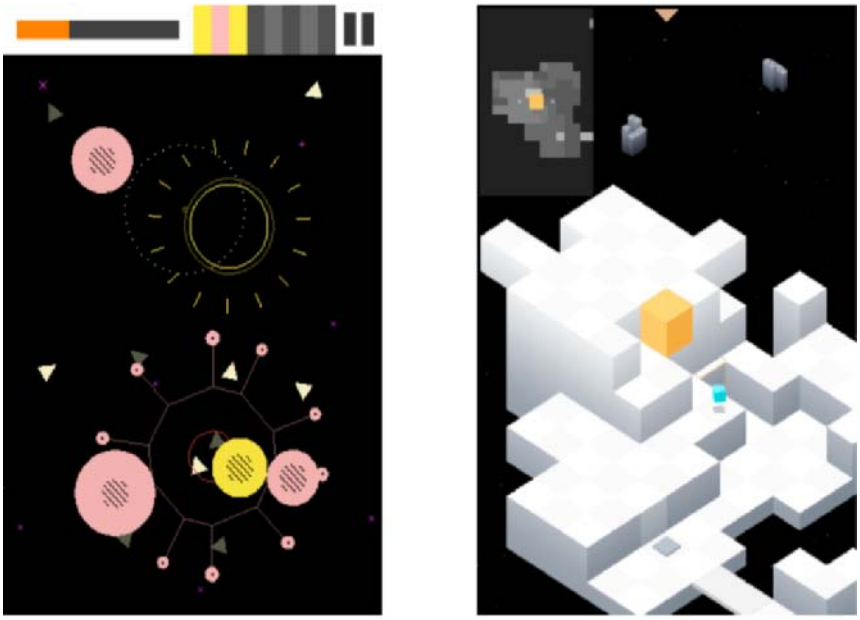


Figure 1–8. *Eliss and Edge are two examples of unique and innovative games that have found success on the App Store.*

Chances are good that if you can dream it and build it, you can distribute it. While the large number of apps available for the iPhone can diminish the signal-to-noise ratio of high-quality games, consumers and Apple will continue to find ways of separating the wheat from the chaff.

Summary

The iPhone's ubiquity, mass appeal, user interface, connectivity, power, and low barrier to entry all combine to make it a game-changing, revolutionary new mobile gaming platform. Groundbreaking interactive experiences such as augmented reality are finding their way into people's hands. While still in its infancy, the iPhone and the App Store have already rewritten the rules of portable gaming. It's an exciting time to be a mobile game developer. In the next chapter, you'll get a better look at the tools and technologies you'll be using to get started.

Chapter 2

Developing iPhone Games: Peeking Inside the iPhone Toolbox

Now that we've established the iPhone's platform credentials and described why you should be excited about developing for it, let's take a peek at some of the tools you'll be using. These technologies include Objective-C or C/C++, Xcode, UIKit, Quartz 2D, Core Animation, OpenGL, audio APIs, networking, and GameKit. This chapter provides a brief overview of these technologies, how they can be used when developing a game, and examples of how they are employed in existing games.

Development Tools and Environment

The language of iPhone development is Objective-C. As the name implies, Objective-C is an extension of the American National Standards Institute (ANSI) C language designed to give C simple and straightforward object-oriented capabilities. While most of the iPhone APIs have Objective-C interfaces, it is possible for the noninterfacing parts of an application to be written in C/C++, since Objective-C syntax is a superset of the GNU C/C++ syntax. You'll need at least some understanding of Objective-C and experience with C/C++.

Lucky for us, Apple prides itself on providing high-quality software to its developers. These tools have been enabling the creation of amazing software for the Mac for quite some time, and you'll be using nearly all the same tools for iPhone development. The foundation of iPhone development is Xcode, which allows for interface design, code editing, debugging, and performance analysis. All of this software is provided free of charge and will run on any Intel-based Mac computer.

The Xcode integrated development environment (IDE) is a full-featured code editor, project manager, and graphical debugger. Xcode contains all the amenities of a modern IDE, including robust syntax coloring, error reporting, code completion, and code folding. Compiling, installing, and launching your application requires a single click, and the on-device debugging is great for hunting down bugs. Make yourself comfortable with its user interface and shortcuts, because you'll be spending a lot of time writing your C/C++ and Objective-C inside Xcode.

Once your game is up and running, you can take advantage of the iPhone simulator. Able to simulate nearly every facet of the iPhone operating system apart from the accelerometer, the simulator is a quick and convenient way to test changes to your app. But make sure to test your app on a real device from time to time, since the simulator doesn't replicate device CPU performance or memory conditions.

Completing the package are a few other tools aimed at helping you design and optimize your iPhone apps. Interface Builder provides a graphical user interface (UI) editor, which automates the loading and positioning of UIKit elements such as buttons and labels. If you're not using OpenGL to build your game, Interface Builder can greatly simplify the creation of items like menus and other static elements. Once you've reached the optimization phase of development, Instruments will come in handy. A powerful profiling tool, Instruments collects and visualizes data such as disk, memory, and CPU usage, allowing you to quickly find the bottlenecks or memory hogs in your game.

UIKit

UIKit provides one of the simplest ways to draw images and other useful UI elements. Displaying and positioning bitmaps is very simple using UIKit, yet still remains relatively fast due to underlying hardware acceleration. UIKit is a great choice for games that don't have a large number of graphical elements or animations and don't need to run at the maximum of 60 fps. Aside from drawing bitmaps, UIKit makes it easy for developers to add other UI elements useful to games, such as alert boxes, text labels, and text-input fields. UIKit also gives access to user input, such as screen touches and accelerometer readings.

NimbleBit's Sky Burger (see Figure 2–1) is a tilt-controlled, burger-stacking game that was developed entirely in UIKit, without the direct use of OpenGL ES. While Sky Burger has a lot of graphics and animated elements, it is near the limit of what UIKit can do graphically with acceptable frame rates. If you wanted to add more graphical effects to a game like this, you would probably need to employ OpenGL ES to ensure that it runs quickly on all devices.

Textropolis by NimbleBit (also shown in Figure 2–1) is another example of a game that doesn't need the powerful graphical rendering provided by OpenGL ES. Because Textropolis is a word game with only small background animations, UIKit was a perfect fit for its development.