Chapter 1

What Makes a Great iOS App

In This Chapter

- ▶ Figuring out what makes an insanely great iOS app
- ▶ Discovering the iOS features that can inspire you
- ▶ Understanding Apple's expectations for iOS apps
- ▶ Making a plan for developing iOS software

July 10, 2008.

That was the day the App Store opened. The next day, Apple launched the iPhone 3G, which came with the brand-new iPhone OS 2.0.1. Owners could choose from the 500 apps in the App Store to expand their phone's power and features. You can find the latest numbers and versions in the App Store article on Wikipedia, which is updated regularly. I think it's fair to say that no one in 2008 envisioned the world of apps that we have today.

This is a world that some people have dreamed of from the dawn of the computer age in the early 1950s. It's a world of computers that are highly portable, that boast terrific connectivity without wires and cables, and that do things that people find useful, such as providing entertainment, education, practical support, and information. Part of that dream was a world in which the ability to program computers is accessible to the largest nations and corporations on almost equal terms as it is to the individual developer or hobbyist.

April 3, 2010.

That was the day the first iPad was shipped. Its operating system was iPhone OS 3.2. iOS 4 was released a few months later, and, today, iOS and its development tools have been substantially rearchitected to make both the user experience and the development process easier.

Today.

Because of the App store and the new development tools, this is a great time to start developing for iOS. Welcome!

Figuring Out What Makes a Great iOS App

A great iOS app can be described simply: It helps people do something that they want to do; it does it well; it does it when and where people want to do it; and it disappears. Because you can leverage the power of the App Store, your app can be successful globally. Yes, that may mean millions of users for your app, but it also may mean that you can find the 100 widely scattered around the world for whom your app may become a necessity.

For the most part, choosing your app's topic and market is beyond the scope of this book, but you can find many articles on the Internet and in books and magazines. You can even study the topic in colleges.

What this book does focus on is the rest of that description — building an app that works well, works when and where people want to use it, and has a user interface that helps people use the app but does not draw attention to itself. If people are thinking about your app when they should be thinking about a plot, a high score, or a trip to a store or another country, your app isn't great.

Making your app work well

The nuts and bolts of making your app work well are described in this book. At the most basic level, they are technical and organizational, but the first step is understanding what your app is going to be and do. Before you write your first line of code, think through the app. Who will the audience be? What will the app look like?

For large app development projects as well as small ones, sketching out a wireframe sequence of your app's screens is a good idea. You may think it's unnecessary in a one-person project, but it may even be more essential there. Show your sketches to friends who will be honest with you. You can search for "ios wireframe" on your favorite search engine to get recommendations for tools to help develop wireframes.

If your app delivers content, make certain that you have the content. If you have expertise in a specific area, that can serve as the basis for an app. If you have access to experts in other areas, see if they will advise you. If you go to the App Store and look at the reviews for apps, you'll see that people quickly provide low ratings for apps that don't work or that don't provide reliable data.

Making your app work well is easier than ever before with Xcode 5. Powerful debugging tools are built in so that you can even watch your app's performance on real-time gauges. Compared to earlier versions of iOS, iOS 7 has a host of improvements for users and simplifications for developers. In my opinion, the milestone was iOS 5. Although significant changes occurred in the later releases, iOS 5 was the first to provide the concept of universal apps where you could write a single code base for both iPhone and iPad. That entailed making some changes to the APIs, but we're over that hump and proceeding full speed ahead.

Handling networking, social media, and location

Networking, social media, and location can make your app great. There certainly are many successful and even great apps that don't use them, but they add additional layers of greatness to your app. Networking means that you can access Internet resources directly from your app. You can display web pages within your app, and you can also access data that you display in your app's interface. The tools to do this are available to you in the APIs.

Today, social media integration scarcely needs promotion: It has become part and parcel of our daily lives. Allowing users to promote your app on Facebook or Twitter with a simple tap is a no-brainer for many developers. iOS lets users enter their sign-in information whenever those taps occur.

Location awareness has opened a wide range of opportunities for apps. The most obvious opportunities involve integration with Maps, but "near me" functionality intrigues developers and users. With iOS 7, developers now have two sets of location tools to use. For traditional mapping, a set of tools uses GPS and cell tower locations to locate the device. Now, iBeacon adds tools to handle low-energy Bluetooth beacons over much shorter distances, such as individual paintings in a museum or specific shelves in a store.

Designing a powerful and intuitive interface that disappears

Designing a disappearing interface is one of the most challenging aspects of app design, and you'll find tips to achieve this throughout the book. A *disappearing interface* is one that works (or as many people say, "just works") without people having to think about it.

When someone looks at your app's interface and notices the interface, you're on the wrong track. What you want to achieve is a situation where someone looks at the screen and immediately sees how to get a weather report, the current temperature, or the prediction for tomorrow's weather in New York City. Users should not notice the interface. Instead, they should notice what the interface can *do*.

Using the iOS Platform to the Fullest

Okay, enough talk about the user experience. Just what exactly is the iOS platform, and what are its features?

Exploiting advantages of the system

One of the keys to creating a great app is taking advantage of what the device offers. In the case of a new platform with new possibilities, exploiting advantages is especially important. The combination of hardware and system software opens up design advantages that depart from the typical design approach for desktop and laptop apps. For example:

- **Multifinger gestures:** Apps respond to multifinger gestures, not mouse clicks. If you design an app that simply uses a single finger tap as if it were a mouse click, you may be missing an opportunity to design a better user experience.
- **Movement and orientation:** iOS devices have a variety of sensors that collect movement and orientation data. The new M7 chip in iPhone 5S, iPad Air, and iPad Mini (second generation) collects sensor information from the integrated accelerometers, gyroscopes, and compasses. This enhances existing location services and takes some of the workload off the main chip.
- **Split views and unique keyboards:** You can use a split view on an iPad to display more than one view onscreen at a time. Both iPad and iPhone provide a special keyboard unique to a task, such as the numbers-and-formulas keyboard that appears in the Numbers app.
- **Internet access:** With quick and easy access, your app doesn't need to store lots of data — all it really needs to do is jump on the Internet and grab what it needs from there. However, to be truly useful, your app needs to be ready to function when the Internet is unavailable to it.

- **Television or projection system connection:** Users can connect an iPhone or iPad to an HDTV or projection system to show content to larger audiences. With iOS's AirPlay feature and an Apple TV, users don't even need a physical connection.
- **Consistent system environment:** The Home button quits your app, and the volume controls take care of audio, just like you'd expect them to. User preference settings can be made available in the Settings application to avoid cluttering your app's user interface. Your native iOS apps can coexist with web services and apps created in HTML5.
- **Breathtaking imagery:** Photos and video already look fantastic on this display, but the artwork you create yourself for your app should be set to 24 bits (8 bits each for red, green, and blue), plus an 8-bit alpha channel to specify how a pixel's color should be merged with another pixel when the two are overlaid one on top of the other. In general, the PNG format is recommended for graphics and artwork that are included as part of your iOS app.

In the following sections, you get to dive into some of the major features, grouped into the following major areas:

- Accessing the Internet
- Tracking location
- Tracking motion
- Supporting multifinger gestures and touches
- Playing content
- Accessing the content of Apple's supplied apps (such as Contacts and Photos)
- Taking advantage of the display

Accessing the Internet

An iOS device can access websites and servers on the Internet through Wi-Fi or optional data services from the same carriers that support the iPhone's voice communication. This Internet access gives you the capability to create apps that can provide real-time information. An app can tell a user, for example, that the next tour at the Tate Modern in London is at 3 p.m, how to get there, and how long the line for the tour is.

This kind of access also allows you, as the developer, to go beyond the limited memory and processing power of the device and access large amounts of data stored on servers, or even offload the processing. You don't need all the information for every city in the world stored on the device, nor do you have to strain the device processor to compute the best way to get someplace on the Tube. You can send the request to a server for all that information, especially information that changes often.

Knowing the location of the user

You can create an app that can determine the device's current location or even be notified when that location changes, using iOS location services. As people move, it may make sense for your app to tailor itself to where the user is, moment by moment.

Many iPad and iPhone apps use location information to tell you where the nearest coffeehouse is or even where your friends are.

When you know the user's location, you can even put it on a map, along with other places she may be interested in. You find out how easy it is to add a map to your app in Chapter 17.

Tracking orientation and motion

All iOS devices contain an *accelerometer with three-dimensional data* — a component that detects changes in movement. The accelerometer measures change along one of the primary axes in three-dimensional space. An app can, for example, know when the user has turned the device from vertical to horizontal orientation, and it can change the orientation from Portrait to Landscape if doing so makes for a better user experience. Newer devices add a gyroscope and — together with the accelerometer — improve the ability of the device to measure in the direction you are moving it in space.

You can also determine other types of motion such as a sudden start or stop in movement (think of a car accident or a fall) or the user shaking the device back and forth. It makes some way-cool features easy to implement — for example, the Etch A Sketch metaphor of shaking the device to undo an operation. You can even control a game by moving the iPhone or iPad like a controller.

Tracking users' fingers on the screen

People use their fingers to select and manipulate objects on the device screen. The moves that do the work, called *gestures,* give the user a heightened sense of control and intimacy with the device. Several standard gestures — tap, double-tap, pinch-close, pinch-open, flick, and drag — are used in the apps supplied with iOS.

TIP

You may want to stick with the standard gestures in your app just because folks are already aware of (and comfortable with) the current pool, but iOS Multi-Touch gesture support lets you go beyond standard gestures when appropriate. Because you can monitor the movement of each finger to detect gestures, you can create your own.

Playing content

Your iOS app can easily play audio and video. You can play sound effects or take advantage of the multichannel audio and mixing capabilities available. You can even create your own music player that has access to all the audio synced to the device from the user's iTunes Library, or from Apple's iCloud service. You can also play back many standard movie file formats, configure the aspect ratio, and specify whether controls are displayed. You can put up pages that look like web pages or book pages if you want, and you can easily mix content for an immersive experience.

Accessing information from Apple's apps

Your app can access the user's information in the Contacts app and display that information in a different way or use it as information in your app. For example, a user could enter the name and address of a hotel, and the app would file it in the user's Contacts database. Then, when the user arrives in New York City, for example, the app can retrieve the address from the Contacts app and display directions. What's more, your app can also present standard interfaces for picking and creating contacts.

What you can do with Contacts, you can do in a similar fashion with the Calendar app. Your app can remind a user when to leave for the airport or create calendar events based on what's happening this week in New York. These events show up in the Calendar app and in other apps that support that framework.

Your app can also access the Photo library in the Photos app, not only to display photos but also to use or even modify them. For example, Apple's Photos app lets you add a photo to a contact, and many apps enable you to edit your photos on the device itself. You can develop your own photo-editing app for the iPhone or iPad using, for example, Apple's Core Image framework.

Copying, cutting, and pasting between apps

iOS provides support for Copy, Cut, and Paste operations within and between apps. It also provides a context-sensitive Edit menu that can display the Copy, Cut, Paste, Select, Select All, and Delete system commands. That means that although each iOS app is generally expected to play only in its own sandbox, you actually do have ways to send small amounts of data between apps.

Multitasking, background processing, and notifications

Recent releases of iOS have implemented and improved background processing for apps. (This has been made possible by new hardware as well as software.) As you will see throughout this book, a lot of iOS development takes place in an *asynchronous* environment. In older software, programs typically executed in a linear (or *synchronous*) manner — one line of code after the other. From time to time, this linear process was interrupted with conditional statements and branches such as *if* and *switch* statements, but the overall flow was linear.

With asynchronous processing, you can execute a section of code (often in the form of what is called a *block*) and not continue on to another line of code. When the block finishes executing, it notifies the app that it is done, and, at that point, the app executes some other code. You basically never know when you will receive such notifications, but iOS makes it easy to manage them. This architecture provides for a peppy user experience.

iOS also offers *push* notifications for receiving alerts from your remote servers even when your app isn't running, and *local* notifications that you can use in your app to alert users of scheduled events and alarms in the background (no servers required). You can use local notifications to get a user's attention; for example, a driver navigation app running in the background can use local notifications to alert the user when it's time to make a turn. Applications can also schedule the delivery of local notifications for a future date and time and have those notifications delivered even if the app isn't running.

Living large on the big screen

The iPad display offers enough space to show a laptop-style app (which is one reason why web pages look so great). You can organize your app with a master list and detailed list of menu choices, or in a layout for Landscape orientation with a source column on the left and a view on the right — similar to the OS X versions of iTunes and iPhoto and exemplified by the Contacts app on the iPad.

Note: Although the iPhone screen is smaller than the iPad screen, don't think of the iPhone screen as being tiny. The iPhone 5 screen, for example, at 1136 x 640 pixels, displays more pixels (on a smaller physical screen) than the original Macintosh screen (512 x 342 pixels). The first Mac had a 0.18-megapixel monochrome display. The iPhone 5 clocks in at a 0.73-megapixel (four times larger) dazzling full-color display. Progress is a wonderful thing, eh?

For example, to crop and mask out parts of an image in Apple's Keynote app for iPad (the app that lets you create slide shows), you don't have to select a photo and then hunt for the cropping tool or select a menu item — just double-tap the image, and a mask slider appears. In Apple's Numbers app for the iPad, if you double-tap a numeric formula, the app displays a special numeric-and-function keyboard rather than a full-text keyboard — and the app can recognize what you're doing and finish the function (such as a Sum function) for you.

These are examples of redesigning a known type of app to get rid of (or at least minimize) that modal experience of using a smartphone app — that sinking feeling of having only one path of communication to perform a task or supply a response. iOS apps should allow people to interact with them in nonlinear ways. Modality prevents this freedom by interrupting a user's workflow and forcing the user to choose a particular path.

TIP

Lists are a common way to efficiently display large amounts of information in iPhone apps. Lists are very useful in iPad apps, too, but you should take this opportunity to investigate whether you can present the same information in a richer way on the larger display.

Embracing Device Limitations

Along with all those features, however, the iPhone, and even the iPad, have some limitations. The key to successful app development — and to not making yourself too crazy — is to understand those limitations, live and program within them, and even learn to love them. (It can be done. Honest.) These constraints help you understand the kinds of apps that are right for this device.

Often, it's likely that if you *can't* do something (easily, anyway) because of device limitations, maybe you shouldn't.

- **Users have fat fingers.** You may think that the iPad's larger display makes that relatively easy to deal with, but keep in mind that you may want to design a multiuser app for the iPad that takes into account multiple fingers. (Anyone for a nice game of Touch Hockey?)
- **Memory and battery power are limited.** This limitation may or may not be a decisive factor, depending on what kind of app you want to create, but smaller apps generally perform better.

The next sections help get you closer to a state of iOS enlightenment.

Designing for fingers

Although the Multi-Touch interface is a feature of the iPad, iPhone, and iPod touch, it brings with it some limitations.

First of all, fingers aren't as precise as a mouse pointer, which makes some operations even more difficult on an iPhone or iPod touch than on an iPad (text selection, for example). Still, due to fat fingers, user-interface elements need to be large enough and spaced far enough apart so that users' fingers can find their way around the interface comfortably. Apple recommends that anything a user has to select or manipulate with a finger be a minimum of 44 x 44 points in size.

Because it's so much easier to make a mistake using fingers, you also need to ensure that you implement a robust — yet unobtrusive — Undo mechanism. You don't want to have your users confirm every action (it makes using the app tedious), but on the other hand, you don't want your app to let anybody mistakenly delete a page without asking, "Are you *sure* this is what you *really* want to do?" Lost work is worse than tediousness.

Balancing memory and battery life

As an app designer, you have several balancing acts to keep in mind:

- **Limited memory:** When compared to the original Macintosh's standards, the computer power and amount of memory on the iPad may seem significant . . . but that is so yesterday. No ifs, ands, or buts; the computer power and amount of memory on the iPhone and iPad are limited. But this is an issue much more with older devices. The newer iPhones and iPads do have fairly large amounts of memory. However, as experienced developers know, the actual amount of memory is pretty much irrelevant: there is never enough for you to relax.

- **Limited battery power:** Access to the Internet can mitigate the device's power and memory limitations by storing data and (sometimes) offloading processing to a server, but those Internet operations eat up the battery faster.

  Although it's true that the iOS power-management system conserves power by shutting down any hardware features that aren't currently being used, a developer must manage the trade-off between all those busy features and a shorter battery life. Any app that takes advantage of Internet access, core location, and the accelerometer is going to eat up the batteries.

As with memory, there is never enough power that you can afford not to think about it.

REMEMBER

iOS devices are particularly unforgiving when it comes to memory usage. If you run out of memory, in order to prevent corruption of other apps and memory, the system will simply shut down your app.

Why Develop iOS Apps?

Because you can. Because it's fun. And because the time has come (today!). iOS apps are busting out all over, and developers have been very successful.

Developing iOS apps can be the most fun you've had in years, with very little investment of time and money (compared with developing for platforms like Windows). Here's why:

- **iOS apps are usually bite-sized, which means that they're small enough to get your head around.** A single developer — or one with a partner and some graphics support — can do them. You don't need a 20-person project team with endless procedures and processes and meetings to create something valuable.
- **The apps tend to be crisp and clean, focusing on what the user wants to do at a particular time and/or place.** They're simple but not simplistic. This makes app design (and subsequent implementation) much easier and faster.
- **The apps use the most innovative platform available for mobile computing.** It's completely changing the Internet as a publishing medium, the software industry with regard to applications, and the mobile device industry with regard to the overall digital media experience.
- **The free iOS Software Development Kit (SDK) makes development as easy as possible.** This book reveals the SDK in all its splendor and glory in Chapter 2. If you can't stand waiting, you *could* register as an iOS developer, and download the SDK . . . but (fair warning) jumping the gun leads to extra hassle. It's worth getting a handle on the ins and outs of iOS app development beforehand.

iOS has these two other advantages that are important to you as a developer:

- **You can distribute your app through the App Store.** Apple will list your app in the App Store in the category you specify, and the store takes care of credit-card processing (if you charge for your app), hosting, downloading, notifying users of updates, and all those things that most developers hate doing. Developers name their own prices for their creations or distribute them free; Apple gets 30 percent of the sales price of commercial apps, with the developer getting the rest. However, keep in mind that Apple must approve your app before it appears in the App Store.
- **Apple has a robust yet inexpensive developer program.** To place your app in the store and manage it, you have to pay $99 per year to join the Individual or Company version of the iOS Developer Program (which includes iPhone and iPad development support). (Apple also offers an Enterprise version for $299 per year to develop proprietary, in-house iOS apps that you can distribute to employees or members of your organization, and a free University version for educational institutions to include iOS development as part of a curriculum.) But that's it. You don't find any of the infamous hidden charges that you often encounter, especially when dealing with credit card companies. Go to the Apple iOS Developer site (`http://developer.apple.com/programs/ios`) and click the Enroll Now button to get started.

Developing with Apple's Expectations in Mind

Just as the iPhone and iPad can extend the reach of the user, the device possibilities and the development environment can extend your reach as a developer. To make sure that you're reaching in the right direction, it helps to understand Apple's perspective on what iOS apps should be — the company clearly has done some serious thinking about it, far longer than anybody else out there, having taken years to bring iOS devices to market under a veil of secrecy.

So what does Apple think? Spokespeople often talk about three different app styles:

- **Productivity apps use and manipulate information.** The RoadTrip sample app that I show in this book is an example, and so are my own Minutes Machine app as well as FileMaker Go (FileMaker), and Apple's iWork apps — Keynote, Pages, and Numbers. Common to all these apps is the use and manipulation of multiple types of information.

- **Utility apps perform simple, highly defined tasks.** Google's YouTube app is an example — it deals only with the YouTube videos. The Brushes app for painting (by Steve Sprang) is considered a utility because it performs a simple, highly defined task.
- **Immersive apps are focused on delivering — and having the user interact with — content in a visually rich environment.** A game is a typical example of an immersive app.

Although these categories help you understand how Apple thinks about iOS apps (at least publicly), don't let them get in the way of your creativity. You've probably heard *ad nauseam* about thinking outside the box. But hold on to your lunch; the iOS "box" isn't even a box yet. So here's a more extreme metaphor: Try diving into the abyss and coming up with something really new.

Thinking About You, Apps, and Money

This book focuses on technology, and you can find many books and articles about the business side of apps. Nevertheless, it's worth spending a moment to review the financial world of apps as it has developed over the last few years.

First of all, consider the fact that it is likely that most apps are given away. They may be given away under various circumstances:

- They may be used as cross-promotion for products that are priced (think of Apple's iWork apps, which became free with the release of iOS 7). People who have iOS 7 (or any other version) have bought at least one iOS device.
- They may be used to provide added value for services that are priced. Think of the free apps for many banks and the free apps from hotels, airlines, and tour companies.
- They may be given away, but they support in-app purchases, whereby you can add more advanced game levels or additional functionality to the basic free app.

Beyond the question of the app's price, you may be wondering if you can be paid for building apps. Although it's hard to find accurate numbers, it is also likely that most developers are not paid. These include students (and remember that only a few years ago everyone was a student when it comes to iOS) as well as would-be professional app developers who are building a portfolio — often with free work for friends or non-profit organizations.

Organizations that are building apps to give away often hire developers (and graphic designers and marketers) as do organizations that are selling apps.

Then there are individual developers or small groups thereof who attempt to do one of the hardest things of all: They attempt to make money from selling the apps they have written.

This is a broad overview of you, apps, and money. You don't have to decide where you're going to land as you become a proficient app developer; you can wait and see what you like to do most. In addition, be aware that many developers today mix and match roles. They may work for free for a non-profit and for pay for an app developer as well as for themselves. (Does every app developer have at least one "skunkworks" project to work on in his free time? Probably.)

Two things are important for you right now:

- Learn iOS with this book.
- Look for groups of app developers near you or online. (Meetup is a good resource.) Many developers share their experiences.

Enter the Cloud

Apple, of course, created a great deal of excitement when it announced iCloud. However, iCloud is more than just an integral part of the built-in applications; it can also be used by developers to implement new functionality for their apps.

iCloud lets you create apps that share data among all of a user's devices. For example, you could create a RoadTrip app that allowed the user to plan a trip on an iPad, and then access and even update that data on an iPhone.

iCloud is available on iOS as well as on the Mac's operating system, OS X. This means that sharing iCloud data among all of a user's devices can mean iPad, iPod touch, iPhone, MacBook Pro, Mac Pro, and the remarkable new device that's still under wraps in Cupertino at Apple's headquarters. (It doesn't matter when you read this: there will always be a remarkable new device under wraps. And, if the past is any indication, we'll all try to figure out how we ever lived without it.) All of this sharing relies on one simple fact: You must use the same (free) Apple ID on all of your devices that you want to share.

Because this book deals only with iOS, I look at iCloud only from that side, but remember that you can make a round trip from iOS to OS X and back again.

Developing an App the Right Way Using the Example App in This Book

As I mention in the Introduction, the point of this book isn't to find out how to program in Objective-C using the iOS frameworks. Instead, the point is to discover how to build apps, and that's what you'll be doing — finding out the *right way* to develop iOS apps.

The best way I can think of to show you how to build an app is to build one, and I take you through doing that throughout this book. The app you build is called RoadTrip. It allows you to plan a trip, check the weather along the way, find the events happening at your destination, and display the destination, sights, and your current location on a map, as well as display any other location by entering the address or the name of a point of interest. You can also choose between destinations.

As simple as it is, RoadTrip shows you how to do many of the tasks that are common to iOS apps. You add animation and sound, display views and navigate through them, and use controls such as buttons, as well as use the Navigation controllers like the kind you find on the iPhone and the split screen on the iPad that allows you to see two views side by side, or one view with another in a popover window.

As you build the RoadTrip app, you even find out how to display a web page and navigate its links (and return) from inside the app. You also download and display data from the Internet.

In addition, I have you build a *universal app* — one that can run either on the iPad or iPhone. Let's get started!

What's Next

You must be raring to go by now and probably can't wait to download the Software Development Kit (SDK). That's exactly what many new developers do — and later are sorry that they didn't spend more time up front understanding the iOS user experience, how apps work in the iOS environment, and the guidelines that Apple enforces for apps to be approved for the App Store.

The following chapters cover all the aspects of development you need to know before you spend time coding. Then, I promise, it's off to the stars.