

Chapter 18

Ten Tips on iPad App Design

In This Chapter

- ▶ Figuring out what makes a great iPad application *icon*
- ▶ Discovering the features of the iPad that can inspire you
- ▶ Understanding Apple's expectations for iPad applications
- ▶ Making a plan for developing iPad software

When John Reed wrote *Ten Days that Shook the World* in 1919, he was writing about a different kind of revolution than the one Steve Jobs referred to in his announcement of the iPad. For this revolution, I put together ten tips that will shake up your thoughts about application design, especially if you're familiar with iPhone app design. There are important differences to know about, and these tips will help you make your iPad app more successful.

You can find more details about each and every one of these tips in the *iPad Human Interface Guidelines* inside the iPhone OS Reference Library. Chapter 4 shows you how to register as a developer and gain access to this library and other resources in Apple's iPhone Dev Center.

Making an App Icon for the Masses

With over 150,000 apps in the App Store, it's a challenge to come up with an icon for your app that would make it stand out in the App Store and look unique and inviting to touch on an iPad display.

Don't even think about using an Apple image, such as an iPad (or iPhone or iPod) in your icon, or you'll most likely receive a polite, but firm, e-mail rejecting the application.

Chapter 9 shows you how to add your icon to your app, and Chapter 6 spells out the details of what form your icon should take — including the fact that you need to use the same (or very similar) graphic image for the small app

icon on the iPad display (at 72 x 72 pixels) you'll use for the larger App Store icon (at 512 x 512 pixels). You also need to supply an approximately 48 x 48-pixel version of this icon for display in Spotlight search results and in the Settings application (if you provide settings).

As with iPhone application icons, the iPhone OS on the iPad automatically adds rounded corners, a reflective shine, and a drop shadow, so you shouldn't add those effects to your icon. Create an icon with 90-degree corners and without any shine or gloss (or alpha transparency), and save it in the .png format to submit to Apple.

Make sure you fill the entire 72 x 72-pixel area — if your image boundaries are smaller, or you use transparency to create see-through areas within it, your icon will appear to float on a black background with rounded corners. Although this may seem fine at first, remember that users can display custom pictures on their Home screens, and an icon with a visible black background looks bad.

Launching Your App Into View

I go into considerable detail in Chapter 8 about how an app starts up and displays a view. Although the chapter is long and takes a while to read, the iPhone OS on the iPad performs these functions instantaneously — so fast that the view appears instantly.

You should take advantage of the speed of the app launch to display an image that represents the heart of your app's functionality — such as one that resembles the most common view of the app's user interface — in the iPad's current orientation. (See “Supporting All Display Orientations,” later in this chapter.) You may think you want to use an About window (with your brand image) or a splash screen, but that slows app startup, and your users will see it every time they start your app. Better to use a simple, stripped-down screen shot of your app's initial user interface or a similar image with only the constant, unvarying elements of the user interface. Avoid all text because you don't want to go through the nightmare of providing different images for different countries.

Your goal during startup should be to present your app's user interface as quickly as possible. Don't load large data structures that your app won't use right away. If your app requires time to load data from the network (or perform other tasks that take a noticeable amount of time), get your interface up and running first and then launch the slow task on a background thread. Then you can display a progress indicator or other feedback to the user to indicate that your app is loading the necessary data or otherwise doing something important.

Stopping Your App on a Dime

When the user presses the Home button, your app comes to an immediate stop, but it shouldn't crumble as if it hit a brick wall. You need to provide a good stopping experience — or more to the point, a good restarting experience for the user who quits and then returns to your app.

If you save data in your app frequently, your app will quit more gracefully without requiring the user to tap a Save button. (See “Saving Grace with Your App's Data,” later in this chapter.) Your app needs to save user data while it's running, and as often as reasonable, because an exit or terminate notification — like Immigration or the Spanish Inquisition — can arrive at any time. And be sure to save the current state when stopping, at the finest level of detail possible, because users expect to return to their earlier context when they restart your app. For example, if you use a Split view in your app, store the current selection in the master pane and be sure to display that selection again when the user restarts your app.

For example, in Chapter 10 and 11 you add code to the DeepThoughts app that saves the user's preference settings as they're set (the text for the flowing words, and the animation speed). When the user restarts DeepThoughts, it uses the user's settings for the falling words and speed.

Saving Grace with Your App's Data

Don't force your users to tap a Save button. iPad apps should take responsibility for saving the user's input, not only periodically but also when a user opens a different document or quits the app.

This design goal addresses the very essence of the iPad experience: that users should feel comfortable consuming information and have complete confidence that their work is always preserved (unless they explicitly delete the work or cancel). If your app lets users create and edit documents, design it so that users don't have to explicitly perform saves. If your app doesn't create content but lets users switch between viewing information and editing it (such as Contacts), your app can offer an Edit button that turns into a Save button when users tap it, and the app can include a Cancel button when that happens. By doing both, your app reminds the users that they're in edit mode and that they can either save the edits or cancel.

If your app uses popovers, you should always save information that users enter in a popover (unless they explicitly cancel) because users may dismiss the popover inadvertently.

Supporting All Display Orientations

As you probably know (or read in Chapter 2), when you rotate the iPad from a vertical view (portrait) to a horizontal view (landscape), the accelerometer detects the movement and changes the display accordingly. Motion detection happens so quickly that you can control a game with these movements.

This is important: iPad users expect apps to run well in the device orientation they're currently using. As much as possible, your app should enable users to interact with it from any side by providing a great experience in all orientations.

For example, the iPad app's launch image should be ready to launch in any of the four orientations — so you need to provide four unique launch images. Each launch image should be in the .png format and should match the size of the iPad display in either portrait orientation (1,024 x 768 pixels) or landscape orientation (768 x 1,024 pixels).

Flattening Information Levels

If you've developed for the iPhone, you want to rethink your app design goals for the iPad. One technique in particular is the one-level-per-screen structure of iPhone apps, which forces your information into a hierarchy of screens resembling an upside-down tree (with the first screen acting as the root). As you tap an option on a screen, you go deeper into the upside-down tree into more detailed or more specific screens.

Although this structure makes sense for the iPhone's smaller display that can hold only one screen at a time, for your iPad app you need to flatten this structure — spread the information out horizontally rather than in a vertically-oriented tree structure — so that it doesn't force iPad users to visit many different screens of information to find what they want. They have one large display, so use it. Focus the app's Main view on the primary content and provide additional information or tools in an auxiliary view, such as a popover. (See "Popping Up All Over," later in this chapter.)

Your app needs to provide easy access to the functionality users need, without requiring them to leave the context of the main task. For example, in Chapter 15 you take a Table view for the iPadTravel411 sample app, which is appropriate for an iPhone app, and implement it as one of the views of a *Split view* (two views on the display at once), which is more appropriate for an iPad app. Use a Split view to persistently display the top

level of a hierarchy in the left pane and content that changes in the right pane, as shown in Figure 18-1. This flattens your information hierarchy by at least one level, because two levels are always onscreen at the same time.



Figure 18-1:
The split
screen
design for
iPad
Travel411 in
landscape
orientation.

Popping Up All Over

A popover appears, or pops up on top of, the Main view when a user taps a control or an area of the display. You can use a popover to enable actions or provide tools that affect objects in the Main view. A popover can display these actions and tools temporarily on top of the current view, which means people don't have to move on to another view to perform the actions or use the tools.

You can put almost anything in a popover, from tables, images, maps, text, or Web views to Navigation bars, toolbars, and controls. A popover can be useful for displaying the contents of the landscape orientation's left pane when a Split view-based application is in portrait orientation. For example, in Chapter 15 you find out how to display the Master view that you see in landscape orientation in a Popover view in portrait orientation, as shown in Figure 18-2.

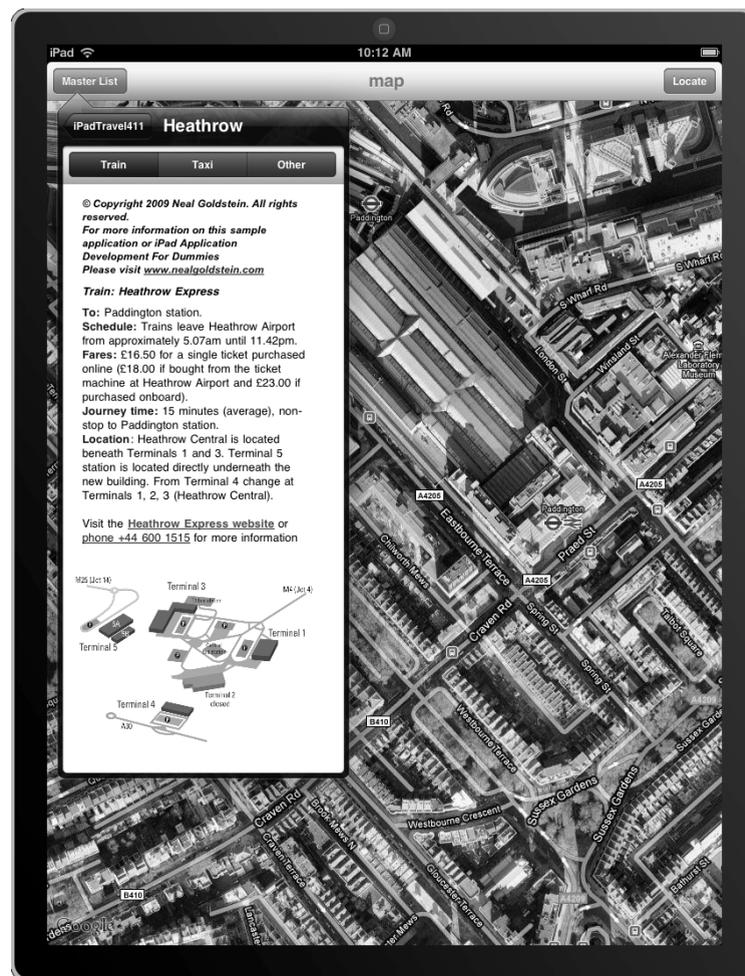


Figure 18-2:
 A popover shows the Master view for iPad Travel411 in portrait orientation.

Minimizing Modality to Maximize Simplicity

Yo! Keep it simple! Keep those Modal views to a minimum. A Modal view is a child window, like a dialog in Mac OS X, that appears on top of the parent window (the Main view) and requires the user to interact with it before returning to the parent window. I know, I showed you how to add a Modal view in DeepThoughts in Chapter 11, but that one is for one purpose only — to change the preference settings — and users can choose whether or not to tap the display (or the Info button) to bring it up.

You don't want to annoy users with modal dialogs that force them to perform a task or supply a response. iPad apps should react to taps in flexible, nonlinear ways. Remember that modality prevents freedom of movement through your app by interrupting the user's workflow and forcing the user to choose a particular path. In general, you should use modality only when it's critical to get the user's attention, or a task must be completed (or explicitly abandoned) to avoid leaving the user's data in an ambiguous state, such as unsaved. Got that?

And if you must use a Modal view, keep the modal tasks fairly short and narrowly focused. You don't want your users to experience a Modal view as a mini application within your application.

Turning the Map into the Territory

As you discover in Chapter 14, working with maps is one of the most fun things you can do on the iPad because Apple makes it so easy — you can display a map that supports the standard panning and zooming gestures by simply creating a view controller and a nib file. You can also center the map on a given coordinate, specify the size of the area you want to display, and annotate the map with custom information.

You can also specify the map type — regular, satellite, or hybrid — by changing a single property. For many apps one type may work better than another, but consider using hybrid so that your app displays streets and highways superimposed over the satellite image. What you should do is provide a control for the user to make the choice between all three types.

Making Smaller Transitions (Don't Flip the View)

The iPad's display is inherently immersive; many things can be going on inside a single view. It's far better to change or update only the areas of the view that need it, rather than swapping in a whole new full-page view when some embedded information changes (as you would probably do in an iPhone app).

Don't flip the entire view if something needs to change. Do transitions with smaller views and objects. Associate any visual transitions with the content that's changing. Use a Split view so that only one part of the view changes (see "Flattening Information Levels," earlier in this chapter), or use a popover for information that changes, to lessen the need for a full-screen transition. (See "Popping Up All Over," earlier in this chapter.) As a result, your app will appear to be more visually stable, and users will feel confident that they know where they are in a given task.

