iPad Human Interface Guidelines

General

2010-01-22

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Contents

Introduction

Important: This is a preliminary document for an API or technology in development. Although this document has been reviewed for technical accuracy, it is not final. Apple is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future seeds of the API or technology.

iPad Human Interface Guidelines contains guidelines you should follow to design a great iPad application that takes full advantage of iPad and iPhone OS 3.2 features.

If you're new to iPhone OS, it's strongly recommended that you begin by visiting the iPhone Reference Library to familiarize yourself with the platform. If you're new to iPhone OS programming, you should start by reading:

- *Your First iPhone OS Application*
- *iPhone Application Programming Guide*

If you're new to user interface design for iPhone applications, begin by reading:

- *iPhone Human Interface Guidelines*

Most of the guidance in those documents applies equally to iPad and to iPhone and iPod touch. Where there are differences, *iPad Human Interface Guidelines* and *iPad Programming Guide* provide device-specific guidance.

Introduction

Key iPad Features and Characteristics

iPad runs iPhone OS 3.2, and iPad applications use many of the same UIKit views and controls you used in your iPhone application. This makes your experience in iPhone application development (and your knowledge of *iPhone Human Interface Guidelines*) very valuable.

Built on this familiar foundation, iPad ushers in a new user experience that differs significantly from the iPhone user experience. With its large screen and its compelling, highly interactive interface, iPad provides a unique platform on which to explore a new class of applications.

Expect to spend a large portion of your development time absorbing the iPad user experience and using this insight to design an application that feels at home on the device.

iPad Platform Characteristics

iPad introduces new platform characteristics that have a significant impact on your application's user interface:

- A large screen size of 1024 x 768 pixels.
- No default or user-expected orientation.
- The option for users to plug in an external keyboard and use it in place of the onscreen keyboard.
- The ability for users to dock the device and share files with a computer.

To help you get your bearings, note that iPad and iPhone share the following platform characteristics:

- Memory is limited.
- One application runs at a time.
- Preferences are available in the Settings application.
- Device orientation can change.
- Onscreen user help is minimal and understated.
- Applications respond to manual gestures, not mouse clicks.
- Native, web-only, and hybrid software run on the device.
- Artwork has a standard bit depth, specifically: 24 bits (8 bits each for red, green, and blue), plus an 8-bit alpha channel. In general, the PNG format is recommended.

New UI Elements and Behaviors

In iPhone OS 3.2, UIKit includes some new UI elements and defines new behaviors for a few elements that are available in earlier versions:

- **Split view**. You can use this iPad-only element to display more than one view onscreen at a time, such as to present data in a master-detail or source list–style arrangement. The split view is a common organizational element in iPad applications because it helps flatten the information hierarchy. Find out more about the ways you can use a split view in your application by reading "iPad User Experience Guidelines" (page 13). For usage guidelines, see "Split View" (page 30).
- **Popover**. You can use this iPad-only view to temporarily display additional information, controls, or choices related to content in the main view. The main benefit of the popover is that it can contain information or choices that don't need to be in the main interface all the time. To find out more about some of the ways you can use a popover, see "Consider Popovers for Some Modal Tasks" (page 18). For usage guidelines, see "Popover" (page 28).
- **Results list button**. You can use this system-provided button to reveal search results from a search bar. (See "Search Bar" (page 31) for usage guidelines.)
- **Modal views have new presentation styles**. You can use styles such as full screen, partial screen, and form to present a modal interface that's more closely tailored to your application's user experience and visual design. (See "Modal View" (page 32) for usage guidelines.)
- **Toolbars can be in additional locations**. You can place a toolbar at the top or the bottom of a screen. You can also use a toolbar inside a split view or a popover. (See "Toolbar" (page 27) for usage guidelines.)
- **The edit menu can display custom items**. You can supply menu items to augment or replace the standard Cut, Copy, Paste, Select, and Select All commands. (See "Edit Menu Additions" (page 33) for usage guidelines.)
- **The keyboard view can be customized**. You can replace the system-provided keyboard view with a custom view that contains custom buttons.
- **The keyboard view can include custom input accessories**. You can supply auxiliary keyboard controls that users can tap to input application-defined content. (See "Keyboard Customization" (page 34) for usage guidelines.)
- **Custom text views can support text in multiple styles and offer advanced editing features**. You can offer word-processing capabilities and support spell checking and autocompletion for text entry.

From iPhone Application to iPad Application

If you already have an iPhone application, you need to know how to revise your application so that it can take full advantage of iPad. Your familiarity with iPhone OS helps you change your code with relative ease, but evolving the user interface and user experience can be more challenging.

Important: Unmodified, your iPhone application runs in a compatibility mode on iPad. This allows your users to access your application on iPad, but it does not give them the device-specific experience they want.

This chapter describes some general strategies for revising the user interface of your iPhone application. For specific guidelines you should follow, see "iPad User Experience Guidelines" (page 13). For help with transitioning your iPhone application code to iPad, see *iPad Programming Guide*.

Design Strategies for Translating Your iPhone Application

It's vital that you revise the user interface and user experience of your iPhone application to take advantage of the large iPad screen and to give people the enhanced interactivity they expect. Precisely how you do this depends on the specifics of your iPhone application.

Games and other immersive iPhone applications may not need much change in information architecture, because they're experience-driven, rather than data-driven. But games generally require significant revision of artwork and interaction to deliver a compelling experience on iPad. As you plan the revision of your game, think about the following things:

- It's essential that you make your artwork as high-fidelity as possible. On iPad, people expect games to furnish a lavish visual environment they can explore in depth. Don't just scale up your artwork. Instead, increase the resolution and take every opportunity to add details that encourage people to immerse themselves in your application.
- You should make sure that your visual effects are also high-fidelity. Clear, finely detailed, lifelike portrayals of events and actions enhance people's emotional response and strengthen their sense of entering a game's unique environment.
- You should investigate ways to let users interact more with your application, without changing the fundamental experience. For example:
  - Consider adding multiplayer capabilities.
  - Give people more things to see and control, by adding more camera angles; more knobs, switches, and dials; or more ways to customize game pieces and gameplay.
  - When it makes sense, support multifinger gestures, especially when you can integrate them with custom controls.

iPhone productivity applications tend to require some re-architecting of the information hierarchy, in addition to an enriched UI and an enhanced user experience. As you plan the revision of your productivity application, keep the following things in mind:

- You want to let users see more content and do more work in one screen. Try to avoid making them drill down through many screens to accomplish their goal. (See "Flatten Your Information Hierarchy" (page 15) for some ways to do this.)
- Your user interface should still be understated enough to avoid distracting users from the task, but don't let that prevent you from making it beautiful. Even though people use productivity applications to accomplish an important task, they expect to enjoy the experience on iPad. Consider using subtly custom elements that harmonize with the task, or making the objects that users interact with look more realistic.
- You should look into ways to provide more interactivity in your iPad application. You always want to maintain a clear focus on the main task, but you should consider exposing more of the data for people to see and control.

Utility applications need to be re-envisioned for iPad so that they take advantage of the larger screen. Because flipping the entire screen is not recommended, many utility applications need to change their interaction model. If you have a utility-style iPhone application:

- Consider ways to combine the content from separate iPhone screens into one screen on iPad.
- Think about giving users more in-depth information about the task. If your iPhone utility application provides a summary view of the content, you might allow people to get an expanded view on the same screen.
- Give people more ways to interact with the content. For example, it might make sense to allow people to manipulate a graph in real time, see a simulation of data, or customize details of the content they see.

Regardless of the style of your iPhone application, be sure to follow the iPad-specific guidelines described in "iPad User Experience Guidelines" (page 13).

Case Study: Mail on iPhone to Mail on iPad

Mail is one of the premier built-in applications on iPhone. People appreciate the clear, streamlined way it organizes large amounts of information, its ease of use, and its scalability. Mail on iPad delivers the same core functionality with a modified user experience that includes:

- More onscreen space for people's messages
- Meaningful touches of realism
- Powerful organizing and editing tools that are always accessible, but not obtrusive
- The visual stability of a single, uncluttered screen that provides what users need in one place, with minimal context changes

The differences between Mail on iPhone and Mail on iPad reflect the different user experiences of each device. Mail on iPhone is designed to help mobile users handle their email while they're standing in line or walking to a meeting. Mail on iPad is certainly efficient enough for people to use on the go, but it also encourages more in-depth usage.

It's crucial to note that, although Mail on iPad tailors the user experience to the device, it does not alter the core functionality people are used to. Nor does it gratuitously change the location or effect of individual functions. iPhone Mail users easily recognize the toolbar items and mailbox structure in iPad Mail, and immediately know how to use them because they're virtually identical.

To enhance the mobile email experience, Mail on iPad evolves the iPhone Mail UI in the following ways:

- **Expanded support for device orientations**. People can use Mail on iPad in any of the four orientations. Although the landscape layout differs somewhat from the portrait layout, people can easily access all the functionality they care about in any orientation.
- **Increased focus on message content**. Mail on iPad reserves most of the screen for the current message. This includes moving the toolbar to the top of the message view to increase the vertical space available for the message content. With the extra space, users can read longer messages with less scrolling. And, when users want to view the message list, they can still see the current message.
- **Flatter hierarchy**. Mail on iPad effectively flattens the account > mailbox > message list > message hierarchy by confining all levels above the message itself to a separate UI element. In landscape, this element is the left pane of a split view; in portrait, this element is a popover.
- **Drastically reduced full-screen transitions**. Because most of the hierarchy is available in a separate onscreen element, users can access most of what they need in a single screen. When users drill down through the hierarchy, it's the view inside the split view pane or popover that transitions, not the entire screen.
- **Realistic messages**. When users mark a message for deletion, it slides onto the message view like a physical sheet of paper. As users choose additional messages to delete, they form a realistic stack of papers, complete with slightly untidy edges.

iPad User Experience Guidelines

Content and interactivity are paramount in the iPad user experience. The best iPad applications elevate content and interactivity by doing three things really well:

- They downplay application UI so that the focus is on the content that people want.
- They present the content in beautiful, often realistic ways.
- They take full advantage of device capabilities to enable enhanced interaction with the content.

The primacy of content and interactivity inform the guidelines in this chapter. Keep this in mind as you design your iPad application.

Support All Orientations

Being able to run in all orientations is key to the success of your iPad application. The reason is that people don't view the device as having a default orientation, because they don't pay much attention to the minimal device frame and they're unconcerned with the location of the Home button. And, the large screen mitigates people's desire to rotate the device to landscape to "see more." Your application should encourage people to interact with iPad from any side by providing a great experience in all orientations.

Important: Although you might not have supported all orientations in your iPhone application, you must do so in your iPad application.

Rotation causes the screen dimensions to switch between 768 x 1024 pixels and 1024 x 768 pixels, which can significantly affect how your UI fits onscreen. Precisely how you respond to rotation might vary, but you should make every effort to abide by the following guidelines.

Maintain focus on the primary content. This is your highest priority. People use your application to view and interact with the content they care about. Altering the focus on that content in different orientations can make people feel that they've lost control over the application.

Consider changing how you display auxiliary information or functionality. Although you should make sure that the most important content is always in focus, you can respond to rotation by changing how you provide secondary content. In Mail, for example, the lists of accounts and mailboxes comprise secondary content (the main content is the selected message). In landscape, secondary content is displayed in the left pane of a split view; in portrait, it's displayed in a popover. As another example, if you provide supplemental information in, say, dials or gauges to the right of the main content in landscape, you could display these objects below the main content in portrait. Both of these examples maintain the user's focus on the primary content and take best advantage of the current screen dimensions without altering the functionality of the application.

Avoid radical or gratuitous changes in layout. The large iPad screen allows you to provide a similar UI layout in all orientations. For example, if you display images in a grid while in landscape, you don't need to display the same information in a list while in portrait. Focus on providing a consistent experience in all orientations, even if the layout of secondary information might change. A comparable experience in all orientations allows people to maintain their usage patterns when they rotate the device.

When possible, avoid reformatting information and rewrapping text on rotation. Strive to maintain a similar format in all orientations. Especially if people are reading text, you should try to avoid causing them lose their place when they rotate the device.

If some reformatting is unavoidable, use animation to help people track the changes. For example, if you must add or remove a column of text in different orientations, you might choose to hide the movement of columns and simply fade in the new arrangement. To help you design appropriate rotation behavior, think about how you'd expect your content to behave if you were physically interacting with it in the real world.

Provide a unique launch image for each orientation. When each orientation has a unique launch image, people experience a smooth application start regardless of the current device orientation. Be aware that, unlike on iPhone, the iPad Home screen supports all orientations, so people are likely to start your application in the same orientation in which they quit the previous application. See "Create a Launch Image for Each Orientation" (page 22) for more information on iPad launch images.

Enhance Interactivity (Don't Just Add Features)

The best iPad applications give people innovative ways to interact with content while they perform a clearly defined, finite task. Resist the temptation to fill the large screen with features that are not directly related to the main task. In particular, you should not view the large iPad screen as an invitation to bring back all the functionality you pruned from your iPhone application.

To make your iPad application stand out, concentrate on ways to amplify the user experience, without diluting the main task with extraneous features. For example:

- A book-reader application that allows people to read books and keep track of reading lists can provide a much more enjoyable reading experience on the large screen. Instead of making people transition to another screen to manage their reading lists, the application can put the list in a popover and allow people to copy favorite passages into it. The application can also let people add bookmarks and annotations to the text, and help them trade their lists with others or compare their progress against a central repository of lists.
- A fighter pilot game might enable a translucent heads-up display over the main view. Players can tap realistic cockpit controls to engage the enemy or locate themselves on a map overlay.
- A soccer playing game can display a larger, more realistic playing field and more detailed characters, and allow people to manage their teams and customize the characters. It can also allow people to see information about characters without leaving the field view. Finally, it can enable a multiplayer mode in which two people can pit their teams against each other.
- A screen writing application might provide ways to switch between a plot view and a character view without leaving the main context. Writers can switch between these views to check details as they write in the main view.

Flatten Your Information Hierarchy

Use the large iPad screen and new UI elements to give people access to more information in one place. Although you don't want to pack too much information into one screen, you also want to prevent people from feeling that they must visit many different screens to find what they want.

In general, focus the main screen on the primary content and provide additional information or tools in an auxiliary view, such as a popover. This gives people easy access to the functionality they need, without requiring them to leave the context of the main task.

With the large iPad screen, and UI elements such as split view and popover, you have alternatives to the one level per screen structure of many iPhone applications. For example, you can:

- Use a split view to persistently display the top level of a hierarchy in the left pane and allow people to drill down through content in the right pane. This flattens your information hierarchy by at least one level, because two levels are always onscreen at the same time. Settings displays device and application settings in this way. (See "Split View" (page 30) for usage guidelines.)
- Use a navigation bar inside the left pane of a split view to allow people to drill down through a fairly shallow hierarchy. Then, display only the most specific information (that is, the leaf nodes in the hierarchy) in the right pane. This, too, flattens your hierarchy by displaying two levels onscreen at one time. Mail in landscape uses this design to display the user's mailbox hierarchy in the left pane and individual messages in the right pane.
- Use a popover to enable actions or provide tools that affect onscreen objects. A popover can display these actions and tools temporarily on top of the current screen, which means people don't have to transition to another screen to get them. (See "Popover" (page 28) for usage guidelines.)
- Use a segmented control in a toolbar to display different perspectives on the content or different information categories. In this way, you can provide access to these perspectives or categories from a single bar at the top (or the bottom) of the screen. (See "Toolbar" (page 27) and "Segmented Control" (page 31) for usage guidelines.)
- Use a tab bar to display different information categories or, less often, different application modes. In iPad applications, a tab bar is more likely to be used as a filter or category switcher than as a mode switcher. It's worth changing your information architecture to avoid multiple, parallel modes, if this allows you to avoid using a tab bar to swap in different screens. (See "Tab Bar" (page 27) for usage guidelines.)

Reduce Full-Screen Transitions

Closely associate visual transitions with the content that's changing. Instead of swapping in a whole new screen when some embedded information changes, update only the areas of the user interface that need it. As a general rule, prefer transitioning individual views and objects, not the screen. In particular, avoid flipping the entire screen.

When you perform fewer full-screen transitions, your application has greater visual stability, which helps people keep track of where they are in their task. You can use UI elements such as split view and popover to lessen the need for full-screen transitions.

Enable Collaboration and Connectedness

People view iPad as a personal device, but its convenient size also encourages physical collaboration and sharing with others.

Think of ways people might want to use your application with others. Expand your thinking to include both the physical sharing of a single device and the virtual sharing of data. For example, two people might be able to play a game on opposing sides of an onscreen board. Or a band application might allow different people to play different instruments together on a single device.

People expect to be able to share information that's important to them. When it makes sense in your application, make it easy for people to interact with others and share things like their location, opinions, and high scores.

Most applications can add value by allowing people to go beyond the application and share data with other tools they use. For example, an iPad application can act as a mobile complement to a computer application, and allow people to transfer and manipulate shared information. Or, an iPad application might allow its users to communicate with the users of the iPhone version of the application.

Add Physicality and Heightened Realism

Whenever possible, add a realistic, physical dimension to your application. The more true to life your application looks and behaves, the easier it is for people to understand how it works and the more they enjoy using it.

As you work on adding realistic touches to your application, don't feel you must strive for scrupulous accuracy. Often, an amplified or enhanced portrayal of something can seem more real, and convey more meaning, than a faithful likeness. As you design objects and scenes, think of them as opportunities to communicate with your users and to express the essence of your application.

Use animation to further enhance realism in your application. In general, it's more important to strive for accuracy in movement than in appearance. People sometimes feel disoriented when they see movement that appears to defy physical laws. As much as possible, make sure your virtual views and controls mimic the behavior of the physical objects and controls they resemble. Convincing animation heightens people's impression of your application as a tangible, physical realm in which they want to spend time.

Delight People with Stunning Graphics

The high-resolution iPad screen supports rich, beautiful, engaging graphics that draw people into an application and make the simplest task rewarding. iPad showcases your application's artwork, so you should consider hiring a professional artist to create first-rate graphics that people will admire.

One way to increase the perceived value of your application is to replicate the look of high-quality or precious materials. For example, if the effect of wood, leather, or metal is appropriate in your application, take the time to make sure the material looks realistic and valuable.

If your artwork is not already high resolution, you may need to recreate it. In most cases, scaling up your artwork is not recommended as a long-term solution. Instead, try creating your artwork in a dimension that is larger than you need, so you can add depth and details before scaling it down. This works especially well

when the dimension of the original art file is a multiple of the dimension you need. Then, if you also use an appropriate grid size in your image-editing application, you'll be able to keep the scaled-down art file crisp and reduce the amount of retouching and sharpening you need to do.

In addition to updating all your textures, effects, and images, make sure you remove from your code any hard-coded values that identify screen dimensions.

Update your existing launch images and create additional ones, if necessary (see "Create a Launch Image for Each Orientation" (page 22)).

Create a large application icon (see "Create a Beautiful Application Icon" (page 22)).

De-emphasize User Interface Controls

Help people focus on the content by designing your application UI as a subtle frame for the information they're interested in. Downplay application controls by minimizing their number and prominence.

Consider creating custom controls that subtly integrate with your application's graphical style. In this way, controls are discoverable, but not too conspicuous.

Also, consider fading controls after people have stopped interacting with them for a little while, and redisplaying them when people tap the screen. Sometimes you may want to fade the rest of your application UI, too. This gives even more of the screen space to the content people want to see.

Minimize Modality

When possible, minimize the number of times people must be in a modal environment to perform a task or supply a response. iPad applications should allow people to interact with them in nonlinear ways. Modality prevents this freedom by interrupting people's workflow and forcing them to choose a particular path. In general, you should use modality only when:

- It's critical to get the user's attention.
- A task must be completed (or explicitly abandoned) to avoid leaving the application in an ambiguous state.

Rethink Your Lists

Lists (that is, table views) are a common way to efficiently display large amounts of information in iPhone applications. Lists are very useful in iPad applications, too, but you should take this opportunity to investigate whether you can present the same information in a richer way. For example:

- Consider a more real-world vision of your application. For example, on iPhone, Contacts is a streamlined list, but on iPad, Contacts is an address book with a beautifully tangible look and feel.

- Consider presenting some of the information as objects instead of list items. For example, iPod displays albums in a grid of album cover thumbnails. Messages that people mark for deletion are displayed as a stack of realistic sheets of paper.
- Constrain the width of a list by embedding it in another view. For example, it might be appropriate to display a list inside a popover or on the flip side of a view instead of in the full width of the screen.
- Try to avoid displaying list information in precisely the same format as you do in your iPhone application. On iPad, lists are much wider, so content that fills a list row on iPhone can look unattractively sparse on iPad. Take advantage of the extra room to provide additional information or details on each row.

Consider Multifinger Gestures

The large iPad screen provides great scope for multifinger gestures, including gestures made by more than one person. Although complex gestures are not appropriate for every application, they can enrich the experience in applications that people spend a lot of time in, such as games or content-creation environments.

Be sure the gestures you use make sense in the context of your application's functionality and the expectations of your users. If, for example, your application enables an important task that users perform frequently and want to complete quickly, you should probably use only standard gestures. But if your application contains realistic controls that dictate a specific usage, or provides an environment that users expect to explore, custom or multifinger gestures can be appropriate.

Consider Popovers for Some Modal Tasks

Popovers and modal views are similar, in the sense that people typically can't interact with the main view while a popover or modal view is open. But a modal view is always modal, whereas a popover can be used in two different ways:

- **Modal**, in which case the popover dims the screen area around it and requires an explicit dismissal. This behavior is very similar to that of a modal view, but a popover's appearance tends to give the experience a lighter weight.
- **Non-modal**, in which case the popover does not dim the screen area around it and people can tap outside its bounds to dismiss it. This behavior makes a non-modal popover seem like another view in the application, not a separate state.

In addition, a popover always has an arrow that points to the control or area the user tapped to reveal it. This visual tie-in helps people remember their previous context. It also makes a modal popover seem like a more transient state than a modal view, which takes over the screen without indicating where it came from.

If you use modal views to enable self-contained tasks in your iPhone application, you might be able to use popovers instead. To help you decide when this might be appropriate, consider these questions:

- Does the task require different types of input? If so, use a popover.

  Although a keyboard can accompany either a popover or a modal view, a popover is better for displaying a picker or a list of options.
- Does the task require people to drill down through a hierarchy of views? If so, use a popover.

The frame of a popover is better suited to displaying multiple pages, because there is less chance people will confuse it with the main view.

- Might people want to do something in the main view before they finish the task? If so, use a non-modal popover.

  Because people can see the main view around a non-modal popover and they can dismiss it by tapping in the main view, you should allow them to suspend the popover's task and come right back to it.

- Is the task fairly in-depth and does it represent one of the application's main functions? If so, you might want to use a modal view.

  The greater context shift of a modal view helps people stay focused on the task until they finish it. The greater screen space of most modal view styles makes it easier for people to provide a lot of input.

  If, on the other hand, the task represents an important part of application functionality, but it is not in-depth, a modal popover can be a better choice. This is because the lighter visual weight of a popover can be more pleasant for frequently performed tasks.

- Is the task performed only once or very infrequently, as with a setup task? If so, consider using a modal view.

  People aren't as concerned about staying in the current context when they perform a task only once or very infrequently.

There are a number of other uses for popovers, such as to provide auxiliary tools. See "Popover" (page 28) for additional usage guidelines. Also, be aware that iPad applications display action sheets inside popovers (see "Action Sheet" (page 32) for more information).

If you decide to use a modal view, be sure to read about the different presentation styles you can use (they're described in "Modal View" (page 32)). In your iPad application, you can choose the presentation style that's best suited to the modal task you need to enable.

Restrict Complexity in Modal Tasks

People appreciate being able to accomplish a self-contained subtask in a modal view, because the context shift is clear and temporary. But if the subtask is too complex, people can lose sight of the main task they suspended when they entered the modal view. This risk increases when the modal view is full screen and when it includes multiple subordinate views or states.

Try to keep 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. Be especially wary of creating a modal task that involves a hierarchy of modal views, because people can get lost and forget how to retrace their steps. If a modal task must contain subviews, be sure to give users a single, clear path through the hierarchy, and avoid circularities.

Always provide an obvious and safe way to exit a modal task. People should always be able to predict the fate of their work when they dismiss a modal view or popover.

If the task needs multiple modal views, make sure your users understand what happens if they tap a Done button on a view that's below the top level. Examine the task to decide whether a Done button in a subview should finish only that subview's part of the task or the entire task. When possible, avoid adding Done buttons to subviews, because of this potential for confusion.

Downplay File-Handling Operations

Although iPad applications can allow people to create and manipulate files and share them with a computer (when the device is docked), this does not mean that people should have a sense of the file system on iPad.

On iPad, there is no application analogous to the Mac OS X Finder, and people should not be asked to interact with files as they do on a computer. In particular, people should not be faced with anything that encourages them to think about file types or locations, such as:

- An open or save dialog that exposes a file hierarchy
- Information about the share status of files

Instead, a document-handling iPad application should encourage people to view their content as objects in the application.

If your iPad application allows people to create and edit documents, it's appropriate to provide some sort of document picker that allows them to open an existing document or create a new one. Ideally, such a document picker:

- **Is highly graphical**. People should be able to easily identify the document they want by looking at visual representations of the documents onscreen.
- **Allows people to make the fewest possible gestures to do what they want**. For example, people might drag horizontally through a carousel of existing documents and open the desired one with a tap.
- **Includes a new document function**. Instead of making people go somewhere else to create a new document, a document picker can allow them to tap a placeholder image to create a new document.

Ask People to Save Only When Necessary

People should have confidence that their work is always preserved unless they explicitly cancel or delete it. If your application helps people create and edit documents, make sure they do not have to take an explicit save action. iPad applications should take responsibility for saving people's input, both periodically and when they open a different document or quit the application.

If the main function of your application is not content creation, but you allow people to switch between viewing information and editing it, it can make sense to ask them to save their changes. In this scenario, it often works well to provide an Edit button in the view that displays the information. When people tap the Edit button you can change it to a Save button and add a Cancel button. The transformation of the Edit button helps remind people that they're in an editing mode, and the Cancel button gives them the opportunity to exit without saving their changes.

In general, save information that people enter in a popover (unless they cancel their work), because they might dismiss the popover without meaning to. See "Popover" (page 28) for more guidelines specific to using popovers.

Migrate Toolbar Content to the Top

If your iPhone application has a toolbar, consider moving it to the top of the screen instead of leaving it at the bottom. With the additional width of the iPad screen, you should be able to provide all of your toolbar functionality in a single toolbar at the top. This gives more vertical space to the focused content.

Start Instantly

iPad applications should start as quickly as possible so that people can begin using them without delay. When starting, iPad applications should:

- **Display a launch image that closely resembles the first screen of the application in the current device orientation**. This decreases the perceived launch time of your application and helps to reassure people that the device is active. (See "Create a Launch Image for Each Orientation" (page 22) for more information.)
- **Avoid displaying an About window or a splash screen that slows application startup**. Try to avoid providing any type of startup experience that prevents people from using your application immediately. You're not prohibited from displaying information about your brand, but you need to make sure that doing so does not delay application startup.
- **Restore state from the last time the application ran**. People should not have to remember the steps they took to reach their previous location in your application.
- **Avoid asking people to supply set-up information**. Instead, you should:
  - **Focus your solution on the needs of 80 percent of your users**. When you do this, the majority of people do not need to supply settings because your application is already set up to behave the way they expect. If there is functionality that only a few people might want, or that most people might want only once, leave it out.
  - **Get as much information as possible from the system**. If you can use any of the information people supply in built-in application or device settings, query the system for these values; don't ask people to enter them again.
  - **Let people benefit from your application before prompting them for input**. If you must get information from people before they can use all the features of your application, first help them accomplish something that doesn't require their input. Later, ask for information when it's necessary, and store it as soon as possible so that you don't have to ask for it again.

Always Be Prepared to Stop

Like iPhone applications, iPad applications stop when people press the Home button to open another application. Therefore, to provide a good stopping experience, an iPad application should:

- **Save user data as soon as possible and as often as reasonable**, because an exit or terminate notification can arrive at any time. When you save frequently, your application quits more quickly and people don't have to tap a Save button (see "Ask People to Save Only When Necessary" (page 20)).

- **Save the current state when stopping, at the finest level of detail possible**. People expect to return to their earlier context when they restart your application. For example, if you use a split view, remember the current selection in the master pane and display it when the user starts your application again.

Create a Launch Image for Each Orientation

Because an iPad application must be able to launch in any of the four orientations, you need to provide four unique launch images. As with iPhone applications, a launch image is a simple, stripped-down snapshot of your application's initial UI. It should include only the constant, unvarying elements of the initial UI and avoid all text (because the launch image is not localizable).

Each launch image should be in the PNG format and should match the size of the device screen in that orientation: either 1024 x 768 pixels or 768 x 1024 pixels. See "Providing Launch Images for Different Orientations" in *iPad Programming Guide* for more information about naming the launch image files and adding them to your application bundle.

Create a Beautiful Application Icon

In the App Store, your application icon introduces your application to potential customers. On their Home screens, your application icon is the first and last thing people see every time they use your application. It's worthwhile to spend the time it takes to create an appealing icon that attracts people and makes a statement about your application.

An iPad application needs a 72 x 72 pixel application icon. These dimensions give you a sizable area in which to tell a story about your application. One way to think about this is to imagine that you're designing a poster for your application. Design a scene that introduces your characters or objects, gives a hint about interaction or storyline, and sets the tone for your application. Use these ideas to create a richly detailed icon that people will enjoy seeing on their Home screens.

As with iPhone application icons, iPhone OS 3.2 automatically adds the following visual effects to your iPad application icon:

- Rounded corners
- Drop shadow
- Reflective shine

To ensure that your icon can take advantage of these visual enhancements, produce a 72 x 72 pixel image in PNG format that:

- Has 90-degree corners
- Does not have any shine or gloss
- Does not use alpha transparency

Important: Be sure to create an image that completely fills the 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. Icons like this are strongly discouraged on iPhone, but they are especially undesirable on iPad because people can display custom pictures on their Home screens. An icon with a visible black background looks very unattractive on an iPad Home screen.

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).

Follow Established Principles

As you develop your iPad application, don't lose sight of the guidelines in *iPhone Human Interface Guidelines*. Most of the information in that document also applies to iPad application design, so you should look there for guidelines on tasks, interactions, and UI elements that are not described in this document.

iPad UI Element Guidelines

Follow these guidelines as you use the new UI elements, and the new behaviors for existing UI elements, introduced in iPhone OS 3.2. If you need to learn how to use an existing element that is not covered in this chapter, see the relevant section in *iPhone Human Interface Guidelines*.

Bars

The status bar, navigation bar, tab bar, and toolbar are views that have specifically defined appearances and behaviors in an application. Although bars in iPad applications look and behave much as they do in iPhone applications, there are a few differences you should be aware of.

The Status Bar

The status bar appears at the upper edge of the device screen (in all orientations) and contains information people need, such as the network connection, the time of day, and the battery charge.

Think twice before hiding the status bar if your application is not a game or full-screen media-viewing application. Most iPad applications do not need to hide the status bar to gain extra space, because the status bar occupies such a small fraction of the screen. And although you can change the status bar color to match your application's appearance, this is not essential because the subtle appearance of the status bar does not compete with your application for the user's attention.

Consider hiding the status bar (and all other application UI) while people are viewing full-screen media. If you do this, be sure to allow people to retrieve the status bar (and appropriate application UI) with a single tap.

Although a game might permanently hide the status bar, you should be aware of the ramifications of this design decision. Permanently hiding the status bar means that users must quit your application to find out, for example, whether they need to recharge their device.

Navigation Bar

A navigation bar appears at the upper edge of an application screen or view. A navigation bar usually displays the title of the current view, and it can contain controls that manage the view's contents, in addition to navigational controls when appropriate.

Note: Navigation bars are used in iPad applications, but they are not as prevalent as they are in iPhone applications. Be sure to read "Flatten Your Information Hierarchy" (page 15) as you consider using a navigation bar in your iPad application.

In your iPad application, you can use a navigation bar in:

- One or both panes of a split view
- A popover
- A modal view
- A full-screen application view (although this is unusual in an iPad application)

Use a navigation bar if you need to allow people to drill down into an information hierarchy. You can do this at the top level of your application or within a discrete view, such as a tab, a split view pane, or a popover. For example:

- Settings uses a navigation bar in the right pane of a split view to help people drill down through the settings associated with the application or feature they selected in the left pane.
- Calendar uses a navigation bar in the Add Event popover to organize the set of detail screens within the popover. To input some of the values, people navigate to a new popover screen, but they remain within the popover.
- iTunes uses navigation bars to allow people to drill down through the content in several of its tabs.

Use the title of the current view as the title of the navigation bar. When the user navigates to a new level, two things should happen:

- The bar title should change to the new level's title.
- A back button should appear to the left of the title, and it should be labeled with the previous level's title.

Use a toolbar instead of a navigation bar if you need to offer a larger set of controls, or you do not need to enable navigation.

Consider putting a segmented control in a navigation bar at the top level of an application. This is especially useful if doing so helps to flatten your information hierarchy and make it easier for people to find what they're looking for. If you use a segmented control in a navigation bar, be sure to choose accurate back-button titles for subsequent information levels. (See "Segmented Control" (page 31) for usage guidelines.)

Avoid crowding a navigation bar with additional controls, even though there might be enough space. In addition to a view's current title, the navigation bar should contain no more than the back button and one control that manages the view's contents. If, instead, you use a segmented control in the navigation bar, the bar should not display a title and it should not contain any other controls.

Use only bordered-style controls in a navigation bar. If you place a plain (borderless) control in a navigation bar, it automatically converts to the bordered style.

Use system-provided buttons according to their documented meaning. See "Standard Buttons for Use in Toolbars and Navigation Bars" in *iPhone Human Interface Guidelines* for more information. If you decide to create your own navigation bar controls, see "Icons for Navigation Bars, Toolbars, and Tab Bars" in *iPhone Human Interface Guidelines* for advice on how to design them.

Be aware that a navigation bar does not change its height or translucency with rotation. This behavior differs from the behavior of a navigation bar in an iPhone application.

Specify the color or translucency of a navigation bar, when appropriate. If you want the navigation bar to coordinate with the overall look of your application, you can specify a custom color. You can make a navigation bar translucent if you want to encourage people to pay more attention to the content underneath the bar. If you customize a navigation bar in these ways, try to make sure it's consistent with the look of the rest of your application.

Tab Bar

A tab bar appears at the bottom edge of an application screen and gives people the ability to switch between different modes, subtasks, or views in an application.

In general, use a tab bar to organize information at the application level. A tab bar is not well-suited for use inside a view, such as a popover or a split view pane.

Limit the number of tabs to about seven. If you have more than about seven application modes or data perspectives, you should review your information architecture and make sure it's not too complex. Consider other options, such as using a segmented control in a tab to give people access to subcategories, or redesigning your application to use a split view.

Avoid creating a More tab. Not only should you limit the total number of tabs, but on iPad, a screen devoted solely to a list of additional tabs is a poor use of space.

Display the same number of tabs in each orientation. In portrait, the recommended seven tabs fit well across the width of the screen. In landscape orientation, you can center the same tabs along the width of the screen. When you do this, people don't experience any jarring shifts in the positions of the tabs.

Use system-provided tab icons according to their documented meaning. See "Standard Icons for Use in Tab Bars" in *iPhone Human Interface Guidelines* for more information. If you decide to create your own tab icons, see "Icons for Navigation Bars, Toolbars, and Tab Bars" in *iPhone Human Interface Guidelines* for advice on how to design them.

Be aware that a tab bar does not change its color, opacity, or height, regardless of orientation. This behavior is the same behavior as in an iPhone application.

Toolbar

A toolbar usually appears at the top edge of a screen or view, but it can also appear at the bottom edge. It contains controls that perform actions related to objects in the screen or view.

Use a toolbar to give people a selection of frequently used commands that make sense in the current context. You can also put a segmented control in a toolbar to give people access to different perspectives on your application's data or to different application modes. With a segmented control, you can provide mode-switching functionality from a single bar at the top of the screen. (See "Segmented Control" (page 31) for usage guidelines.)

Maintain a hit target area of at least 44 x 44 pixels for each toolbar item. If you crowd toolbar items too closely together, people have difficulty tapping the one they want.

Use system-provided toolbar items according to their documented meaning. See "Standard Buttons for Use in Toolbars and Navigation Bars" in *iPhone Human Interface Guidelines* for more information. If you decide to create your own toolbar items, see "Icons for Navigation Bars, Toolbars, and Tab Bars" in *iPhone Human Interface Guidelines* for advice on how to design them.

Try to avoid mixing plain style (borderless) and bordered toolbar items in the same toolbar. You can use either style in a toolbar, but mixing them does not usually look good.

Content Views

In addition to the existing image, map, table, text, and web views, iPhone OS 3.2 introduces two new views to manage content: the **popover** and the **split view**. With the exception of new capabilities for text views, the behaviors of the other existing content views remain the same as in previous versions of iPhone OS.

Popover

A popover is a transient view that can be revealed when people tap a control or an onscreen area.

Important: Popovers are available in iPad applications only.

A popover can contain a wide variety of objects and views. For example, a popover can contain:

- Table, image, map, text, web, or custom views.
- Navigation bars or toolbars. You should not use a tab bar inside a popover.
- Controls or objects that act upon objects in the current application view.

You can use a popover to:

- Display additional information or a list of items related to the focused (or selected) object.
- Display the contents of the left pane when a split view–based application is in portrait. If you do this, be sure to provide an appropriately titled button that displays the popover, preferably in a navigation bar or toolbar at the top of the screen.
- Display an action sheet that contains a short list of options that are closely related to something on the screen.

Note: A popover always displays an arrow, and you cannot change the appearance of a popover's background.

Avoid providing a "dismiss popover" button. A popover should close automatically when its presence is no longer necessary. For example:

- When a popover's only function is to provide a set of options or items that have an effect on the main view, it should close as soon as people make a choice. This behavior is very similar to that of a menu in a computer application. Note that this behavior also applies to a popover that contains only an action sheet: As soon as people tap a button in the action sheet, the popover should close.

Occasionally, it can make sense to provide a popover that contains items that affect the main view, but that does *not* close when people make a choice. You might want to do this if you implement an inspector in a popover, because people might want to make an additional choice or change the attributes of the current choice.

Popovers that provide menu or inspector functionality should close when people tap outside their bounds. In a popover that provides a menu of choices, this gesture means that the user has decided not to make a choice (so the main view remains unaffected). In a popover that contains an action sheet, this gesture takes the place of tapping a Cancel button.

- If a popover enables a task, it can be appropriate to display buttons that complete or cancel the task, and simultaneously dismiss the popover. In general, popovers that enable an editing task display a Done button and a Cancel button. These buttons help remind people that they're in an editing environment and allow them to explicitly keep or discard their input. When people tap either button, the popover should close.

  If it makes sense in your application, you can prevent people from closing such a popover when they tap outside its borders. This might be a good idea if it's important that people finish (or explicitly abandon) a task. Otherwise, you should save the user's input when they tap outside a popover's borders, just as you would if they tapped Done.

In general, save the user's work when they tap outside a popover's borders. Because a popover does not require an explicit dismissal, the user might dismiss it mistakenly. You should discard the user's work only if they tap a Cancel button.

Try to ensure that the popover arrow points directly to the element that revealed it. This helps people remember where the popover came from and what task or object it's associated with.

Make sure people can use a popover without seeing the application content behind it. A popover obscures the content behind it, and people cannot drag a popover to another location.

Ensure that only one popover is visible onscreen at a time. A popover is closely associated with a user action, so it would be confusing for people to see more than one popover onscreen.

When possible, allow people to close one popover and open a new one with one tap. This behavior is especially desirable when several different bar buttons each open a popover, because it prevents people from having to make extra taps.

Avoid making a popover too big. A popover should not appear to take over the entire screen. Instead, it should be just big enough to display its contents and still point to the place it came from.

Ideally, the width of a popover should be at least 320 points, but no greater than 600 points. The height of a popover is not constrained, so you can use it to display a long list of items. In general, though, you should try to avoid scrolling in a popover that enables a task or that presents an action sheet. Note that the system might adjust both the height and the width of a popover to ensure that it fits well on the screen.

If appropriate, change a popover's size while it remains visible. You might want to change a popover's size if you use a popover to display both a minimal and an expanded view of the same information. If you adjust the size of a popover while it's visible, you can choose to animate the change. Animating the change is usually a good idea because it avoids making people think that a new popover has replaced the old one.

Split View

A split view is a full-screen view that consists of two side-by-side panes. Even though the left pane is often called the master pane and the right pane is often called the detail pane, you are not limited to using a split view to implement a master-detail design.

Important: Split views are available in iPad applications only.

Typically, you use a split view to display persistent information in the left pane and related details or subordinate information in the right pane. When people select an item in the left pane, the right pane displays the information related to that item. (You're responsible for making this happen in code.) Two examples that use split views are:

- Mail. In landscape, Mail displays the user's account and mailbox hierarchy in the left pane and the selected message in the right pane. People drill down through the content in the left pane and view the most detailed information in the right pane.
- Settings. In all orientations, Settings displays top level device and application categories in the left pane and specific settings in the right pane. People select a category in the left pane and drill down through related settings in the right pane, if necessary.

Both panes can contain a wide variety of objects and views, such as:

- Table, image, map, text, web, or custom views.
- Navigation bars and toolbars. You should not use a tab bar inside a split view pane.

An application that uses a split view in landscape uses a popover to display the contents of the left pane when it rotates to portrait. However, this does not mean that you cannot design your user interface to display side-by-side views in all orientations.

Avoid creating a right pane that is narrower than the left pane. The width of the left pane is fixed at 320 points in all orientations. Although the width of the right pane is up to you, it does not look good to use a width of less than 320 points.

In general, indicate the current selection in the left pane in a persistent way. This behavior helps people understand the relationship between the item in the left pane and the contents of the right pane. This is important because the contents of the right pane can change, but they should always remain related to the item selected in the left pane.

Text View

Be sure to avoid using Core Text to compute layout of text and a `UITextView` object to draw the result, because this use is not supported. Core Text capabilities are primarily intended to help implement very full-featured word processing applications. Core Text is not necessary or suitable for the vast majority of applications that need simpler text-handling capabilities.

Note that fonts of type `.ttf` or `.otf` work on iPad; traditional Mac OS fonts (that is, Classic font suitcases) do not work on iPad.

Controls

All UIKit controls introduced in previous versions of iPhone OS are available to iPad applications. With a couple of slight differences, the controls look and behave as you and your users expect.

Date and Time Picker, Picker

Present a picker or date and time picker only within a popover. This placement differs from the placement recommendation for an iPhone application. For more information about these what these controls do and how you can customize them, see "Date and Time Pickers" in *iPhone Human Interface Guidelines* and "Pickers" in *iPhone Human Interface Guidelines*.

Info Button

Avoid using an Info button to flip the entire screen. Instead, you might use an Info button to show people that they can access an expanded view that contains related information or additional details. This usage differs from the most common usage of an Info button on iPhone, which is to flip the screen to reveal configuration options.

Page Indicator

The large iPad screen lessens the need for the page indicator control. If you used a page indicator in your iPhone application, investigate ways to display your content on a single screen, instead of on multiple, parallel screens.

Search Bar

Use a search bar in a navigation bar or a toolbar to give people quick access to search. If search is a primary function in your application, you might want to provide a search tab, as iTunes does.

Display the search results button to let people retrieve or change the results of a previous search. People tap this button to see a list of search results in a popover. If it makes sense in your application, you can filter the list (or supply autocompletions or suggestions) while people type. See *UISearchBar Class Reference* to learn how to display the search results button in a search bar.

Consider using a scope bar to allow people to change the scope of the search. In iPad applications, you can put a scope bar below the search bar inside a subview, for example, in the left pane of a split view or in a popover.

Segmented Control

A **segmented control** is a linear set of segments, each of which functions as a button that can display a different view.

Use a segmented control in a top-edge toolbar to provide different perspectives or modes in the application. When you put a segmented control at the top of the screen, people tend to see it as a way to manage data categories or modes at the application level. For example, Maps uses a segmented control in a top-edge toolbar to let people switch between search mode and directions mode.

Use a segmented control in a bottom-edge toolbar to provide different perspectives or modes in a view. When you put a segmented control at the bottom of a view, people tend to see it as a way to manage data categories or modes on a view level. For example, iPod uses a segmented control in a bottom-edge toolbar to let people group their libraries in different ways.

Action Sheets, Alerts, and Modal Views

Action sheets, alerts, and modal views are temporary views that appear when something requires the user's attention or when additional choices or functionality need to be offered. People cannot interact with an application while one of these views is on the screen.

Action Sheet

An action sheet displays a set of choices related to a task the user initiates.

In an iPad application, an action sheet is displayed within a popover; it never has full-screen width.

An action sheet can be displayed with or without animation:

- **With animation, an action sheet slides up over an open popover's content**. Use animation to display alternatives related to a task that the user initiates from within an open popover.

  An animated action sheet should include a Cancel button, because people need to be able to dismiss the action sheet without closing the popover.

- **Without animation, an action sheet and its popover appear simultaneously**. Display an action sheet without animation to provide alternatives related to a task that the user initiates from outside a popover. When you display an action sheet this way, the popover's arrow points to the control or area the user tapped to initiate the task.

  Do not include a Cancel button, because people can tap outside the popover to dismiss the action sheet without selecting one of the other alternatives.

Alert

Alerts behave the same on iPad as they do on iPhone. See "Using Alerts" in *iPhone Human Interface Guidelines* for guidance on when to use an alert and how to design a good alert experience.

Modal View

A modal view (that is, a view presented modally) provides self-contained functionality in the context of the current task or workflow.

A modal view can be presented in a style that suits the current task and the visual style of your iPad application:

- **Full screen**. Covers the entire screen. This style is good for presenting a potentially complex task that people can complete within the context of the modal view. For example, iPod uses this style for its Genius playlist creation task.
- **Page sheet**. Has a fixed width of 768 points; the sheet height is the current height of the screen. In portrait, the modal view covers the entire screen; in landscape, the screen that is visible on both sides of the modal view is dimmed to prevent user interaction. For example, Mail uses this style for its message composition task.
- **Form sheet**. Has fixed dimensions of 540 x 620 points and is centered in the screen. The area of the screen that is visible outside the modal view is dimmed to prevent user interaction. When the keyboard is visible in landscape, a form sheet view moves up to just below the status bar. This style is good for gathering structured information from the user. For example, Mail uses this style for its account setup task.
- **Current context**. Uses the same size as its parent view. This style is good for displaying a modal view within a split-view pane, popover, or other non-fullscreen view.

In addition to the existing transition styles (vertical and flip), iPhone OS introduces the partial-curl transition. Using this transition, one corner of the current view curls up to reveal the modal view underneath. When the user leaves the modal view, the current view uncurls to its original position. Note that a modal view revealed by a partial-curl transition cannot itself reveal another modal view.

The partial-curl transition style is similar to the page-curl behavior of Maps on iPhone. You might want to use this style when the modal view you reveal is not large and does not require much user interaction, such as a view that holds configuration options.

If you're considering displaying a modal view that can contain other modal views, be sure to avoid convoluted interactions. See "Restrict Complexity in Modal Tasks" (page 19) for some guidance.

Edit Menu Additions

If you add custom items to the edit menu, be sure to list them together after the system-provided items. Don't intersperse your custom items with the system-provided ones.

Keep the number of custom menu items reasonable. You don't want to overwhelm your users with too many choices.

Create succinct names for your custom menu items and make sure the names precisely describe what the commands do. In general, item names should be verbs that declare the action the item performs on the user's selection. Although you should prefer a single capitalized word for an item name, use title-style capitalization if you must use a short phrase. (Briefly, title-style capitalization means to capitalize every word except articles, coordinating conjunctions, and prepositions of four or fewer letters.)

Keyboard Customization

iPhone OS 3.2 allows you to design a custom input view that replaces the system-provided onscreen keyboard. If you provide a custom input view, be sure its function is obvious to people. Also, be sure to make the controls in your input view look tappable.

Document Revision History

This table describes the changes to *iPad Human Interface Guidelines*.

Date	Notes
2010-01-22	New document describing how to design a compelling user interface for an iPad application.

Document Revision History