iPad Human Interface Guidelines

User Experience

2010-09-08

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

Simultaneously published in the United States and Canada.

Contents

Introduction **Introduction 7**

Organization of This Document 7

Chapter 1 **Key iPad Features and Characteristics 9**

iPad Device Characteristics 9
New UI Elements and Behaviors 10

Chapter 2 **From iPhone Application to iPad Application 11**

Design Strategies for Translating Your iPhone Application 11
Case Study: From Mail on iPhone to Mail on iPad 12

Chapter 3 **iPad User Experience Guidelines 17**

Aim to Support All Orientations 17
Enhance Interactivity (Don't Just Add Features) 19
Flatten Your Information Hierarchy 19
Reduce Full-Screen Transitions 22
Enable Collaboration and Connectedness 22
Add Physicality and Heightened Realism 23
Delight People with Stunning Graphics 24
De-emphasize User Interface Controls 25
Minimize Modality 25
Rethink Your Lists 25
Consider Using Multifinger Gestures 26
Consider Using Popovers for Some Modal Tasks 26
Restrict Complexity in Modal Tasks 27
Downplay File-Handling Operations 28
Ask People to Save Only When Necessary 28
Migrate Toolbar Content to the Top 29
Start Instantly 30
Always Be Prepared to Stop 31
Create Custom Icons and Images 31
- Application Icons 32
- Small Icons 34
- Document Icons 35
- Web Clip Icons 37
- Icons for Navigation Bars, Toolbars, and Tab Bars 39
- Launch Images 40

Follow Established Principles 41

Chapter 4 iPad UI Element Guidelines 43

Bars 43
The Status Bar 43
Navigation Bar 44
Tab Bar 46
Toolbar 47
Content Views 48
Popover 48
Split View 50
Text View 52
Controls 52
Date and Time Picker, Picker 52
Info Button 53
Page Indicator 54
Search Bar 54
Segmented Control 55
Action Sheets, Alerts, and Modal Views 56
Action Sheet 56
Alert 58
Modal View 63
Edit Menu Additions 65
Keyboard Customization 65

Document Revision History 67

Figures and Tables

Chapter 2 **From iPhone Application to iPad Application 11**

Figure 2-1 Mail on iPhone presents streamlined email functionality in a series of screens 13
Figure 2-2 Mail on iPad works well in all orientations (landscape shown) 14
Figure 2-3 Mail on iPad focuses on message content in all orientations (portrait shown) 15
Figure 2-4 In editing mode, Mail on iPad displays messages marked for deletion as a stack of papers 16

Chapter 3 **iPad User Experience Guidelines 17**

Figure 3-1 Settings uses a navigation bar in the right pane 20
Figure 3-2 Mail uses a navigation bar in the left pane 21
Figure 3-3 Mail uses a popover to display account and mailbox information in portrait 21
Figure 3-4 iTunes uses a segmented control to provide perspectives on the content 21
Figure 3-5 iTunes uses a tab bar to provide categories of content 22
Figure 3-6 Contacts depicts a realistic address book that behaves as people expect 23
Figure 3-7 The appearance of realistic, high-quality materials enhances Notes 24
Figure 3-8 Mail on iPhone provides important functionality in a toolbar 29
Figure 3-9 Mail on iPad moves functionality to the top of the screen 30
Figure 3-10 A 64 x 64 pixel document icon, before and after processing 36
Figure 3-11 A 320 x 320 pixel document icon, before and after processing 37
Table 3-1 Custom icons and images 31

Chapter 4 **iPad UI Element Guidelines 43**

Figure 4-1 The status bar fades into the background on iPad 44
Figure 4-2 A navigation bar can contain navigational and other controls 45
Figure 4-3 A tab bar switches subtasks, views, or modes 46
Figure 4-4 A tab bar in landscape displays the same number of tabs as it does in portrait 47
Figure 4-5 A toolbar provides controls that act upon objects in the screen or view 47
Figure 4-6 A popover provides choices or functionality in a temporary view 48
Figure 4-7 A split view contains two related views 51
Figure 4-8 A date and time picker makes it easy to select values with multiple parts 53
Figure 4-9 A search bar accepts input from users 54
Figure 4-10 A scope bar helps people focus their searches 55
Figure 4-11 A segmented control can provide access to different perspectives or views 55
Figure 4-12 A segmented control works well in a popover to filter information or switch modes 56
Figure 4-13 An action sheet is displayed inside a popover 57
Figure 4-14 An animated action sheet typically includes a Cancel button 58
Figure 4-15 A modal view presents a self-contained task 64

Introduction

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

For iPad applications, the guidelines in this document supplement, and occasionally supersede, the guidelines in *iPhone Human Interface Guidelines*. If you read only this document, and not *iPhone Human Interface Guidelines*, you miss out on information that applies to all iOS-based devices, including iPad.

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

- *Your First iOS Application*
- *iOS Application Programming Guide*

If you're new to user interface design for iOS 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.

Organization of This Document

iPad Human Interface Guidelines contains the following chapters:

- "Key iPad Features and Characteristics" (page 9) provides an overview of the device, focusing on characteristics and features that differ from iPhone.
- "From iPhone Application to iPad Application" (page 11) describes some strategies for transitioning iPhone applications to iPad.
- "iPad User Experience Guidelines" (page 17) lists guidelines that help you deliver a great user experience in your iPad application.
- "iPad UI Element Guidelines" (page 43) describes how to use UI elements in iPad applications.

Key iPad Features and Characteristics

iPad runs iOS 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 offers you a unique opportunity to create 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 Device Characteristics

iPad introduces new 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.

To help you get your bearings, note that iPad and iPhone share the following 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 iOS 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 17). For usage guidelines, see "Split View" (page 50).
- **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 Using Popovers for Some Modal Tasks" (page 26). For usage guidelines, see "Popover" (page 48).
- **Results list button**. You can use this system-provided button to reveal search results from a search bar. (For usage guidelines, see "Search Bar" (page 54).)
- **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. (For usage guidelines, see "Modal View" (page 63).)
- **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. (For usage guidelines, see "Toolbar" (page 47).)
- **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. (For usage guidelines, see "Edit Menu Additions" (page 65).)
- **The keyboard view can be customized**. You can replace the system-provided keyboard view with a custom view that contains custom buttons. (For usage guidelines, see "Keyboard Customization" (page 65).)
- **The keyboard view can include custom input accessories**. You can supply a separate view of auxiliary keyboard controls that users can tap to input application-defined content.
- **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 iOS 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 17). 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 rearchitecting 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 19) 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. Also consider displaying data in more appealing ways, such as graphically instead of textually.
- Look into ways to provide more interactivity in your iPad application. You always want to maintain a clear focus on the main task, but it's often a good idea to expose more of the data for people to see and control.

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

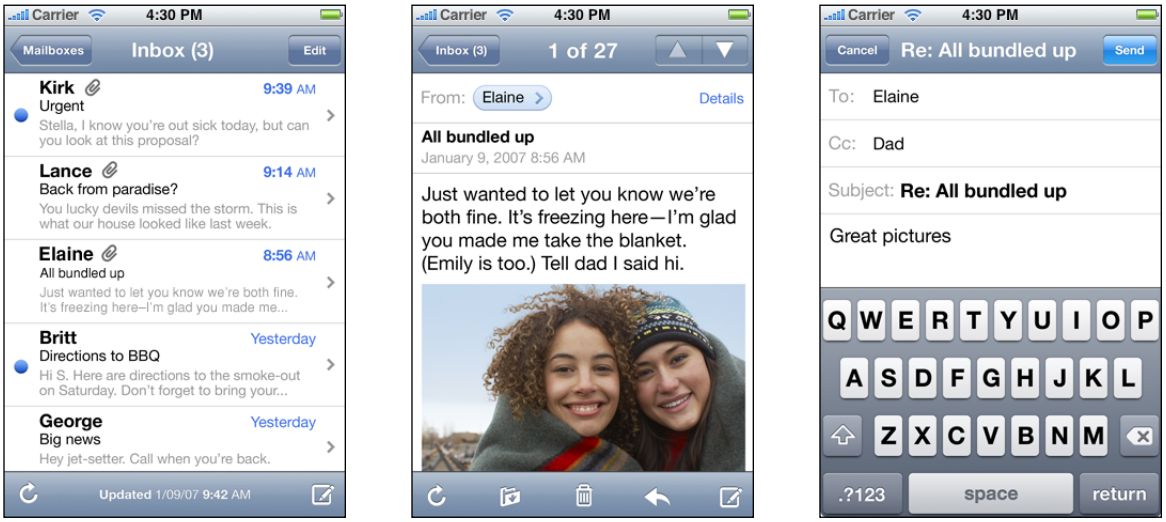
- Consider ways to combine the content from separate iPhone screens into one screen on iPad.
- Think about giving users more in-depth information about the task. If your iPhone utility application provides a summary view of the content, you might allow people to get an expanded view on the same screen.
- 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 17).

Case Study: From 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.

Figure 2-1 Mail on iPhone presents streamlined email functionality in a series of screens



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

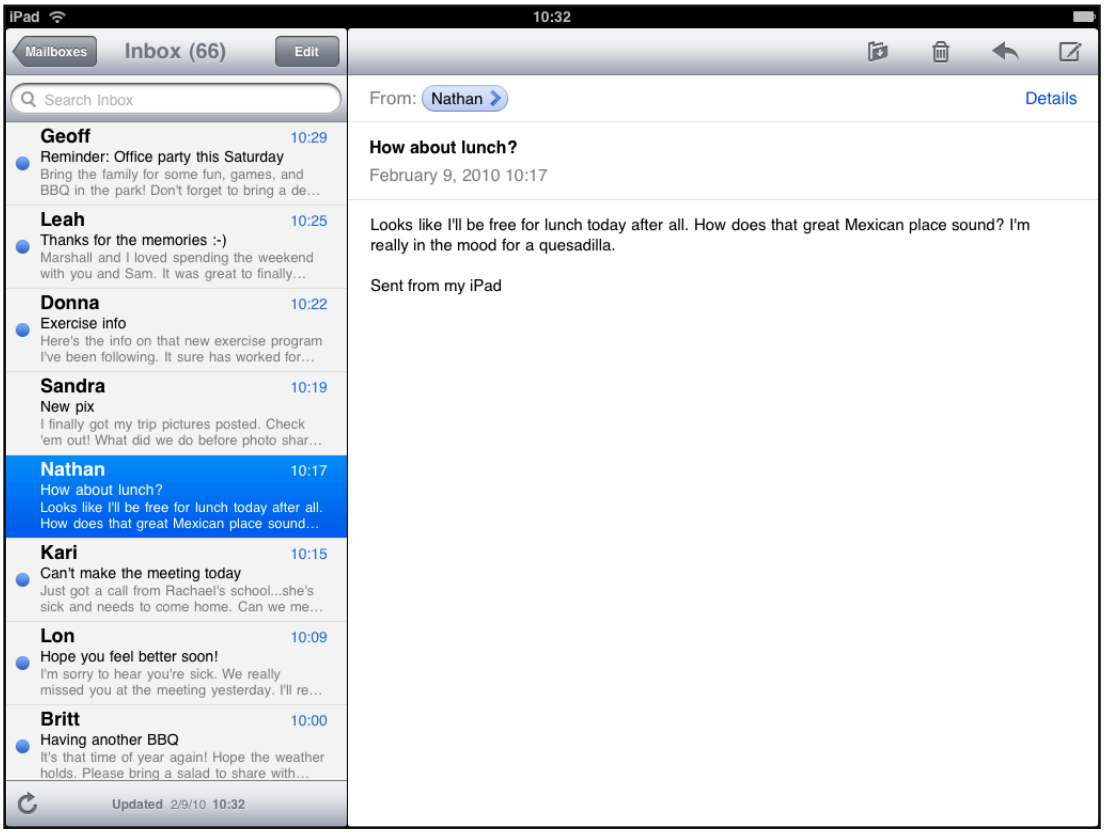
The differences between Mail on iPhone and Mail on iPad reflect the different user experiences of each device. Mail on iPhone is designed to help mobile users handle their email while they're standing in line or walking to a meeting. Mail on iPad is efficient enough for people to use on the go, but it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

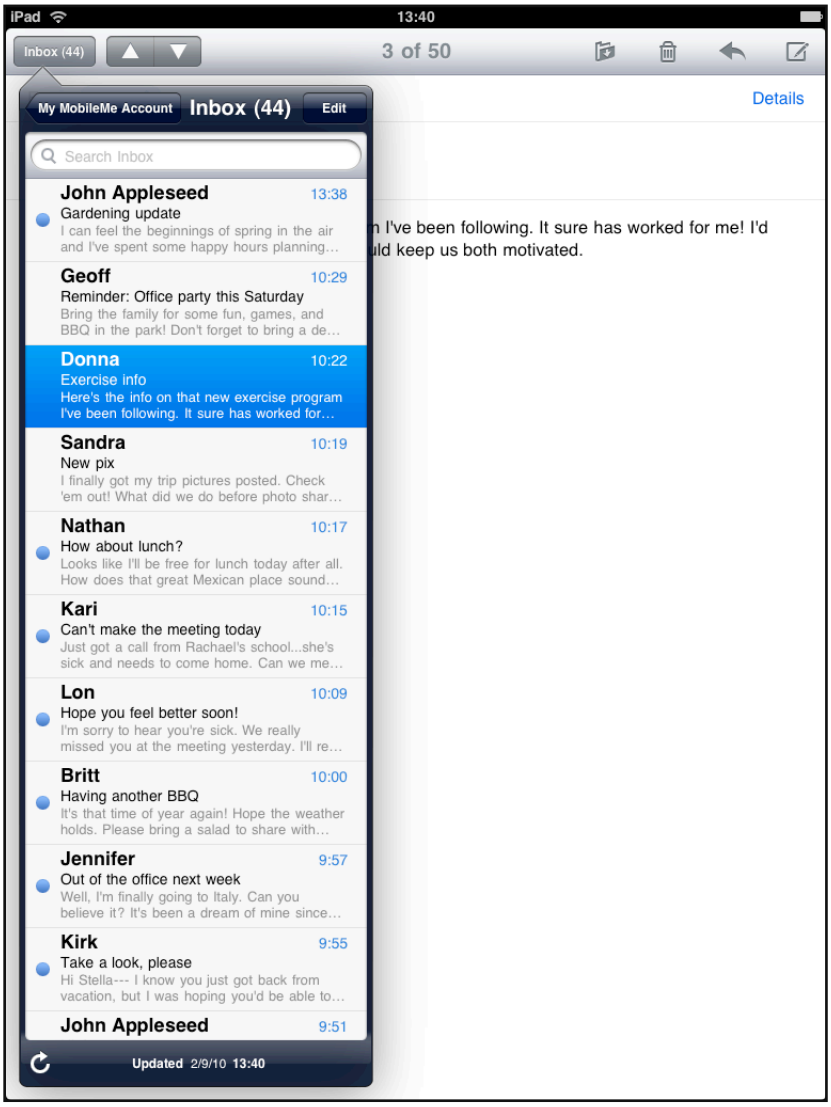
Expanded support for device orientations. People can use Mail on iPad in any of the four orientations. Although the landscape layout differs somewhat from the portrait layout, people can easily access all the functionality they care about in any orientation. Figure 2-2 shows Mail in landscape; Figure 2-3 shows Mail in portrait.

Figure 2-2 Mail on iPad works well in all orientations (landscape shown)



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, people can read longer messages with less scrolling. And when people want to view the message list, they can still see a large portion of the current message.

Figure 2-3 Mail on iPad focuses on message content in all orientations (portrait shown)

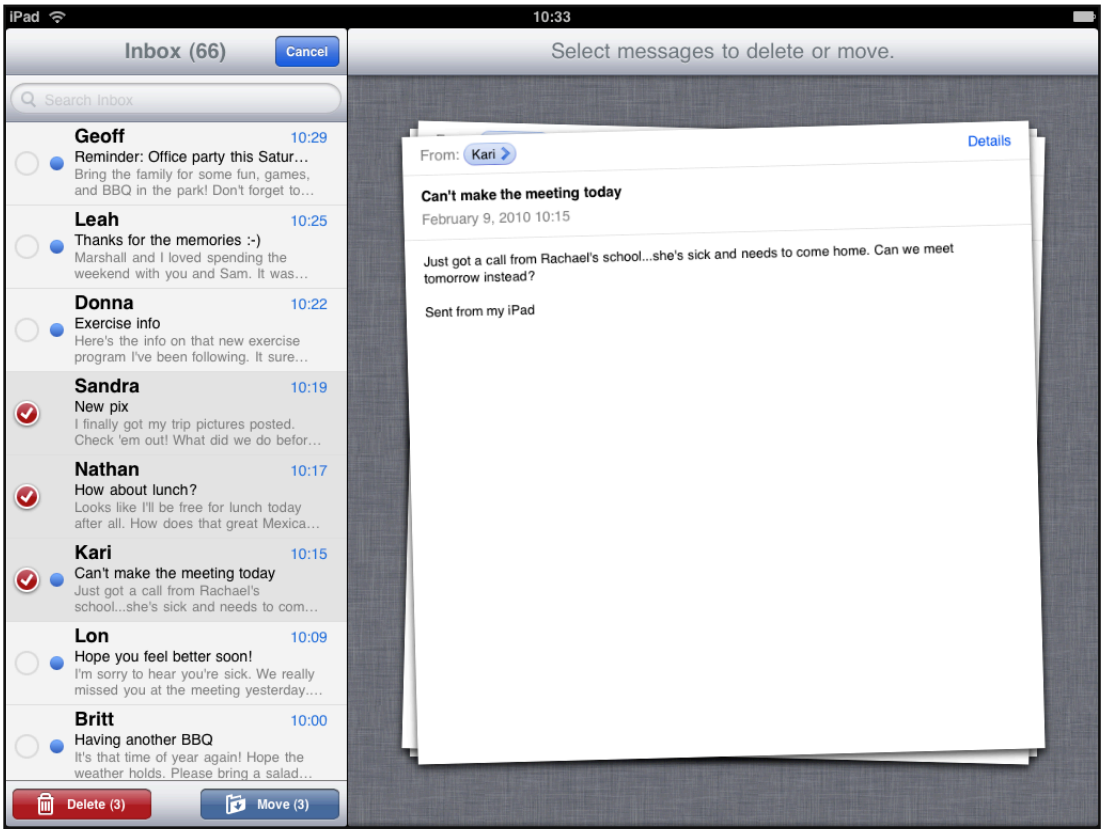


Flatter hierarchy. Mail on iPad effectively flattens the account > mailbox > message list > message hierarchy by confining all levels above the message itself to a separate UI element. In landscape, this element is the left pane of a split view (shown in Figure 2-2); in portrait, this element is a popover (shown in Figure 2-3).

Drastically reduced full-screen transitions. Because most of the hierarchy is available in a separate onscreen element, people can access most of what they need in a single screen. When people 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 people mark a message for deletion, it slides onto the message view like a physical sheet of paper. As they choose additional messages to delete, the messages form a realistic stack of papers, complete with slightly untidy edges, as shown in Figure 2-4.

Figure 2-4 In editing mode, Mail on iPad displays messages marked for deletion as a stack of papers



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.

Aim to Support All Orientations

Being able to run in all orientations is an important factor in the success of your iPad application. The large screen mitigates people's desire to rotate the device to landscape to "see more." And, because people don't pay much attention to the minimal device frame or the location of the Home button, they don't view the device as having a default orientation. This leads people to expect applications to run well in the device orientation they're currently using. As much as possible, your application should encourage people to interact with iPad from any side by providing a great experience in all orientations.

The difference between landscape and portrait dimensions 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. Or, consider a game that displays a rectangular game board in landscape. In portrait, the game needs to redraw the board to fit well on the screen, which might result in additional space above or below the board. Instead of vertically stretching the game board to fit the space or leaving the space empty, the game could display supplemental information or objects in the additional space. 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 primary 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 (although you might adjust the dimensions of the grid). 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, it's important to avoid causing them to 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.

Avoid providing a UI element or defining a rotation gesture that rotates your content. Instead, people should be able to rotate your content by rotating the device.

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. In contrast with the Home screen 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 "Launch Images" (page 40) for more information on iPad launch images.

Think twice before preventing your application from running in all orientations. People expect to use your application in whichever orientation they're currently holding their iPad, and it's best when you can fulfill that expectation. In certain cases, however, an application needs to run in portrait only or in landscape only. If it's essential that your application run in only one orientation, you should:

- **Launch in your supported orientation, regardless of the current device orientation**. For example, if your game or media-viewing application runs in landscape only, it's appropriate to launch in landscape, even if the device is currently in portrait. This way, if people start your application in portrait, they know to rotate the device to landscape to view the content.
- **Avoid displaying a UI element that tells people to rotate the device**. Launching in your supported orientation clearly tells people to rotate the device, if required, without adding unnecessary clutter to your UI.
- **Support both variants of an orientation**. For example, if your application runs only in landscape, people should be able to use it whether they're holding the device with the Home button on the right or on the left. And, if people rotate the device 180 degrees while using your application, it's best if you can respond by rotating your content 180 degrees.

If your application interprets changes in device orientation as user input, you can handle rotation in application-specific ways. For example, if your application is a game that allows people to move game pieces by rotating the device, you can't respond to device rotation by rotating the screen. In a case like this, you should launch in either variant of your required orientation and allow people to switch between the variants until they start the main task of the application. Then, as soon as people begin the main task, you can begin responding to device movement in application-specific ways.

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 screenwriting 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

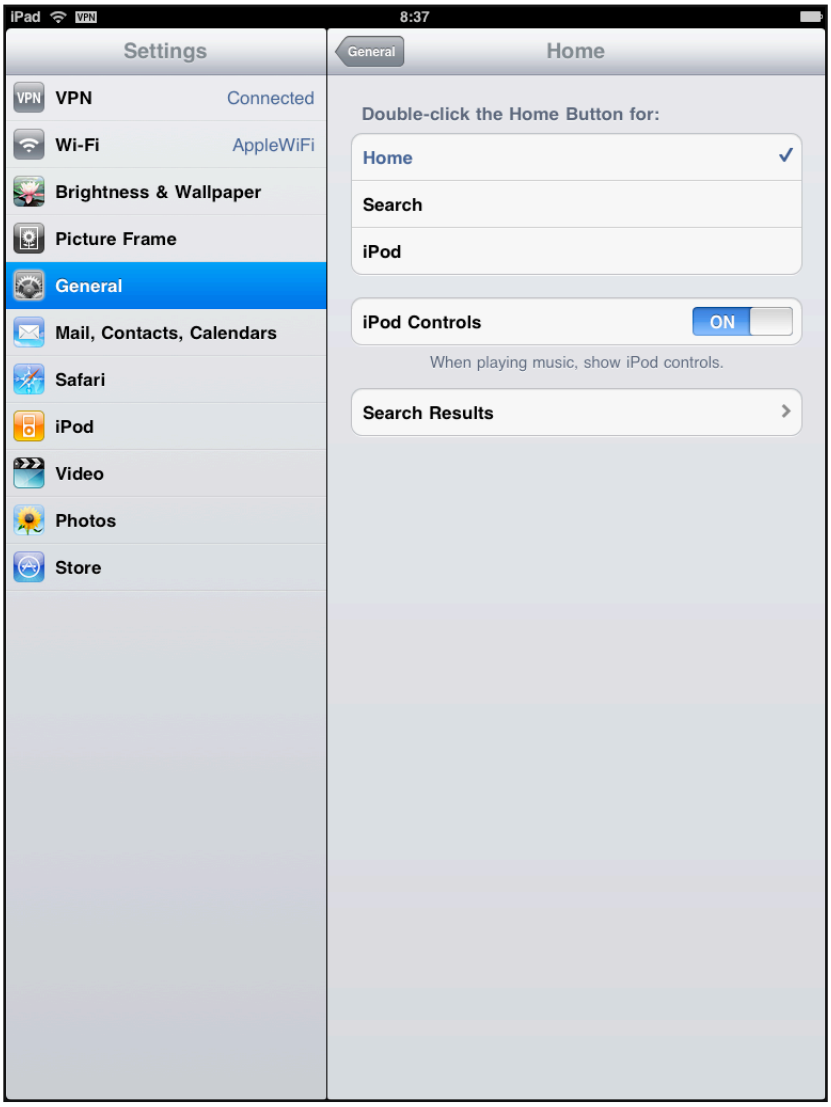
Use the large iPad screen and new UI elements to give people access to more information in one place. Although you don't want to pack too much information into one screen, you also want to prevent people from feeling that they must visit many different screens to find what they want.

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:

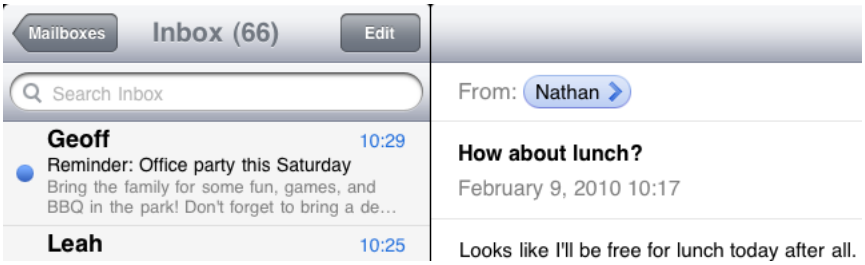
Use a navigation bar in the right pane of a split view to allow people to drill down into a top-level category that is persistently displayed in the left pane. This flattens your information hierarchy by at least one level, because two levels are always onscreen at the same time. Settings displays device and application settings in this way, as shown in Figure 3-1. (See "Split View" (page 50) for usage guidelines.)

Figure 3-1 Settings uses a navigation bar in the right pane



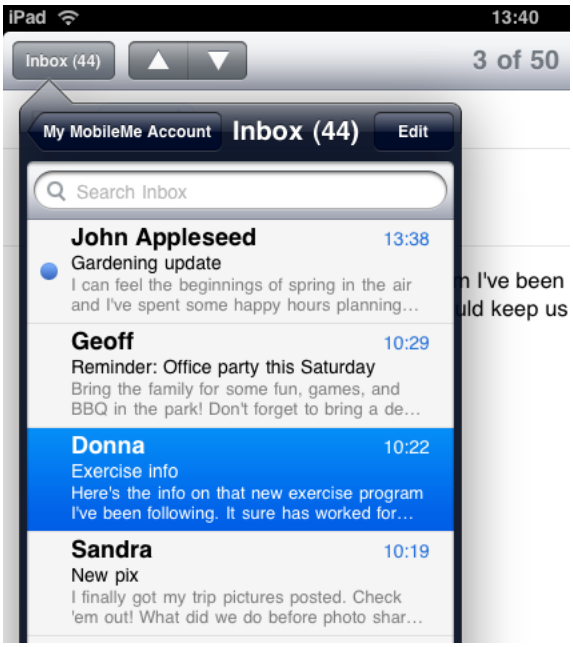
Use a navigation bar in the left pane of a split view to allow people to drill down through a fairly shallow hierarchy. Then, display the most specific information (that is, the leaf nodes in the hierarchy) in the right pane. This, too, flattens your hierarchy by displaying two levels onscreen at one time. Mail in landscape uses this design to display the user's mailbox hierarchy in the left pane, as shown in Figure 3-2; individual messages are displayed in the right pane, as shown in Figure 2-2 (page 14). (See "Navigation Bar" (page 44) for usage guidelines.)

Figure 3-2 Mail uses a navigation bar in the left pane



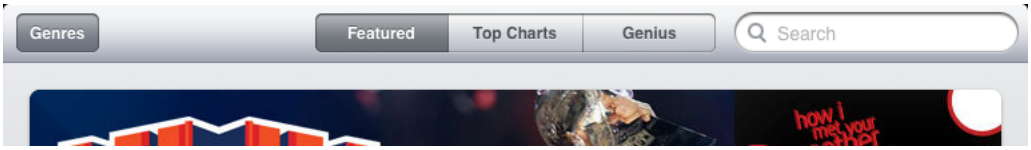
Use a popover to enable actions or provide tools that affect onscreen objects. A popover can display these actions and tools temporarily on top of the current screen, which means people don't have to transition to another screen to get them. Mail in portrait uses a popover to display the user's account and mailbox hierarchy, as shown in Figure 3-3. (See "Popover" (page 48) for usage guidelines.)

Figure 3-3 Mail uses a popover to display account and mailbox information in portrait



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. iTunes uses a segmented control in a top-edge toolbar to provide different perspectives on the content in a category, as shown in Figure 3-4. (See "Toolbar" (page 47) and "Segmented Control" (page 55) for usage guidelines.)

Figure 3-4 iTunes uses a segmented control to provide perspectives on the content



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. As shown in Figure 3-5, iTunes uses a tab bar to give people access to different categories of media. 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 46) for usage guidelines.)

Figure 3-5 iTunes uses a tab bar to provide categories of content

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, try to 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 most cases, flipping the entire screen is not recommended.

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.

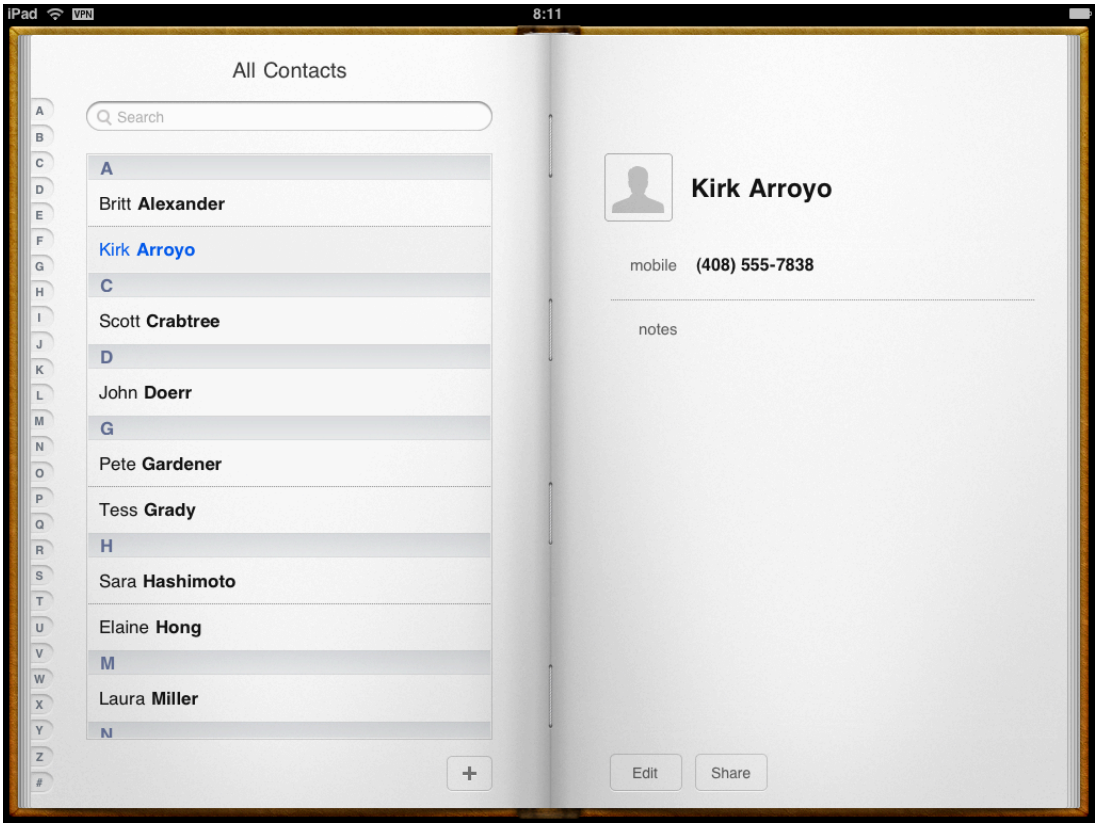
People expect to be able to share information that's important to them. When it makes sense in your application, make it easy for people to interact with others and share things like their location, opinions, and high scores.

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. 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. For example, people instantly know how to use the realistic address book that Contacts portrays (shown in Figure 3-6).

Figure 3-6 Contacts depicts a realistic address book that behaves as people expect



As you work on adding realistic touches to your application, don't feel that 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.

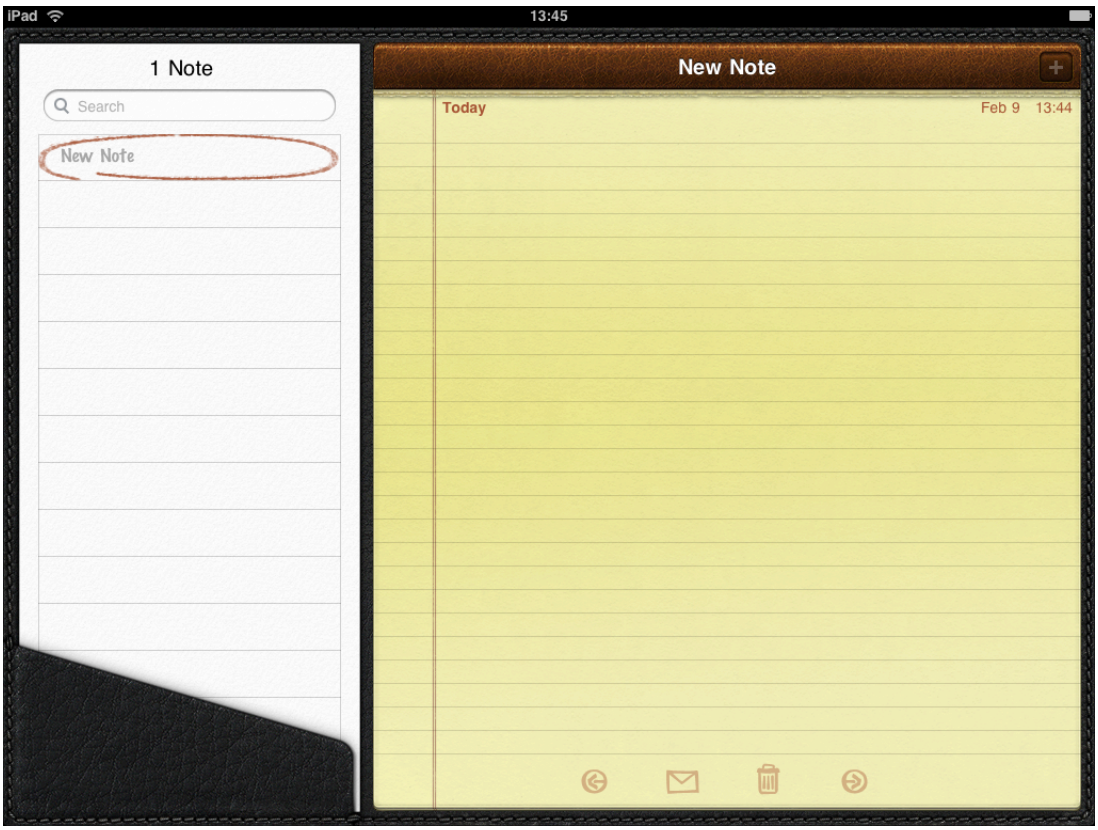
Use animation to further enhance realism in your application. In general, it's more important to strive for accuracy in movement than in appearance. This is because people accept artistic license in appearance, but they can feel disoriented when they see movement that appears to defy physical laws. As 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. As shown in Figure 3-7, Notes reproduces the look of fine leather and meticulous stitching.

Figure 3-7 The appearance of realistic, high-quality materials enhances Notes



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 "Launch Images" (page 40)).

Create a large application icon (see "Application Icons" (page 32)).

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. Photos does this by placing a few unobtrusive controls on translucent bars.

Consider creating custom controls that subtly integrate with your application's graphical style. In this way, controls are discoverable, without being 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. For example, Photos fades the controls and bars after a few moments of noninteraction, which encourages people to immerse themselves in the content.

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.

Modality is most appropriate when:

- It's critical to get the user's attention.
- A task must be completed (or explicitly abandoned) to avoid leaving the user's data 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 (this is shown in Figure 3-6 (page 23)).
- Consider presenting some of the information as objects instead of list items. For example, the iPod application displays albums in a grid of album cover thumbnails. And in Mail, messages that people mark for deletion are displayed as a stack of realistic sheets of paper (this is shown in Figure 2-4 (page 16)).
- 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.

- When possible, 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 sparse on iPad. Take advantage of the extra room to provide additional information or details on each row.

Consider Using Multifinger Gestures

The large iPad screen provides great scope for custom 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. Always bear in mind that nonstandard gestures aren't discoverable and should rarely, if ever, be the only way to perform an action.

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. (For more information about the standard gestures, see "Support Gestures Appropriately" in *iPhone Human Interface Guidelines*.)

Consider Using 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.
- **Nonmodal**, in which case the popover does not dim the screen area around it and people can tap anywhere outside its bounds (including the control that reveals the popover) to dismiss it. This behavior makes a nonmodal 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 nonmodal popover.

  Because people can see the main view around a nonmodal 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 (for complete usage guidelines, see "Popover" (page 48)). Also, be aware that iPad applications display action sheets inside popovers (for more information, see "Action Sheet" (page 56)).

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 63)). 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 views, because people can get lost and forget how to retrace their steps. If a modal task must contain subtasks in separate views, 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 a hierarchy of modal views, make sure your users understand what happens if they tap a Done button in a view that's below the top level. Examine the task to decide whether a Done button in a lower-level view should finish only that view's part of the task or the entire task. When possible, avoid adding Done buttons to subordinate views, because of this potential for confusion.

Downplay File-Handling Operations

Although iPad applications can allow people to create and manipulate files, 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 permissions 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 scroll 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. For more guidelines specific to using popovers, see "Popover" (page 48).

Migrate Toolbar Content to the Top

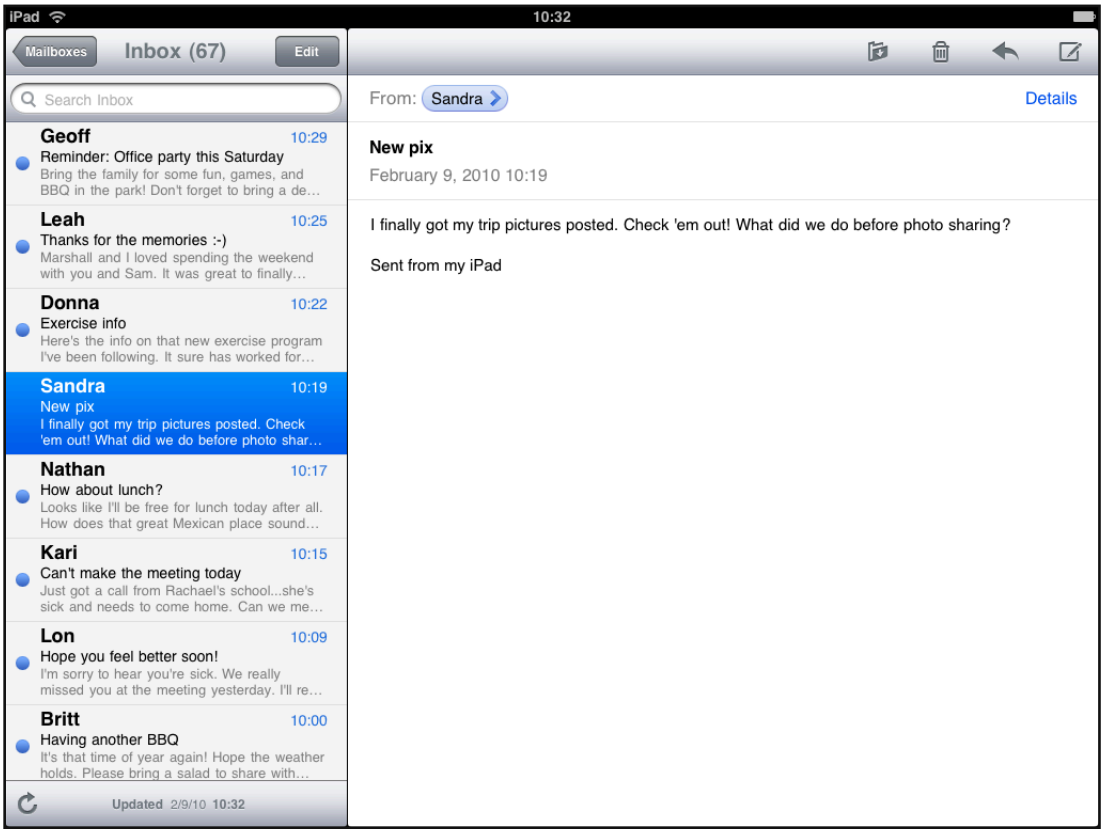
If your iPhone application has a toolbar, consider moving it to the top of the screen instead of leaving it at the bottom. With the additional width of the iPad screen, you should be able to provide all of your toolbar functionality in a single toolbar at the top. This gives you more vertical space for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

For example, Mail on iPhone uses a toolbar to give people access to the refresh, organize, trash, reply, and compose actions while they view messages, as shown in Figure 3-8.

Figure 3-8 Mail on iPhone provides important functionality in a toolbar

Mail on iPad provides access to all but one of these actions in the toolbar above the message, as shown in Figure 3-9. The Refresh control is in the mailbox list, which is in a popover in portrait and in the left pane of the split view in landscape.

Figure 3-9 Mail on iPad moves functionality to the top of the screen



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 application screen in the current orientation**. This decreases the perceived launch time of your application and helps to reassure people that the device is active. (For more information, see "Launch Images" (page 40).)
- **Avoid displaying an About window or a splash screen that slows application startup**. You're not prohibited from displaying information about your brand, but you need to remember that your users will see this content *every* time they start your application. If you feel you must display a splash screen, make sure that it doesn't remain visible for too long and that it disappears without requiring any user interaction. Your main concern should be to make your application launch quick and pleasant.
- **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 setup information**. Instead, you should:
  - **Focus your solution on the needs of 80 percent of your users**. When you do this, most 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 28)).
- **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 Custom Icons and Images

Every application needs to supply a custom application icon, even if it doesn't include any other custom artwork. You should also supply a small icon for display in Spotlight search results. Depending on your application, you might need to supply custom bar icons, a Settings icon, or an icon that represents a custom document type.

Unlike other custom artwork in your application, these icons and images must meet specific criteria so that iOS can display them properly. Table 3-1 contains a summary of information about these icons and images and provides links to specific guidelines for creating them.

To learn what to name these files, and how to specify them in your code, see "Application Icons" in *iOS Application Programming Guide* and "Application Launch Images" in *iOS Application Programming Guide*.

Table 3-1 Custom icons and images

Description	Size for iPhone and iPod touch (in pixels)	Size for iPad (in pixels)	Guidelines
Application icon (**required**)	57 x 57 114 x 114 (high resolution)	72 x 72	"Application Icons" (page 32)
App Store icon (**required**)	512 x 512	512 x 512	"Application Icons" (page 32)

Description	Size for iPhone and iPod touch (in pixels)	Size for iPad (in pixels)	Guidelines
Small icon for Spotlight search results and Settings (recommended)	29 x 29 58 x 58 (high resolution)	50 x 50 for Spotlight search results 29 x 29 for Settings	"Small Icons" (page 34)
Document icon (recommended for custom document types)	22 x 29 44 x 58 (high resolution)	64 x 64 320 x 320	"Document Icons" (page 35) (For iPhone, see "Document Icons" in *iPhone Human Interface Guidelines*)
Web Clip icon (recommended for web applications and websites)	57 x 57 114 x 114 (high resolution)	72 x 72	"Web Clip Icons" (page 37)
Toolbar and navigation bar icon (optional)	Approximately 20 x 20 Approximately 40 x 40 (high resolution)	Approximately 20 x 20	"Icons for Navigation Bars, Toolbars, and Tab Bars" (page 39)
Tab bar icon (optional)	No larger than 48 x 32 No larger than 96 x 64 (high resolution)	No larger than 48 x 32	"Icons for Navigation Bars, Toolbars, and Tab Bars" (page 39)
Launch image (**required**)	320 x 480 640 x 960 (high resolution)	For portrait: 768 x 1004 For landscape: 1024 x 748	"Launch Images" (page 40)

Note: For all images and icons, the PNG format is recommended.

The standard bit depth for icons and images is 24 bits (8 bits each for red, green, and blue), plus an 8-bit alpha channel.

You do not need to constrain your palette to web-safe colors. Although you can use alpha transparency in the icons you create for navigation bars, toolbars, and tab bars, do not use it in application icons.

For information on creating high-resolution versions of icons and images for iPhone 4, see "Tips for Creating Great High-Resolution Artwork" in *iPhone Human Interface Guidelines*.

Application Icons

An **application icon** is an icon users put on their Home screens and tap to start an application. 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. Every application needs an application icon.

You can view your application icon as an opportunity 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.

Create different sizes of your application icon for different devices. If you're creating a universal application, you need to supply application icons in all three sizes.

For iPad:

- 72 x 72 pixels

For iPhone and iPod touch both sizes are required:

- 57 x 57 pixels
- 114 x 114 pixels (high resolution)

When iOS displays your application icon on the Home screen of a device, it automatically adds the following visual effects:

- Rounded corners
- Drop shadow
- Reflective shine (unless you prevent the shine effect)

For example, a very simple 72 x 72 pixel iPad application icon might look like this:

When it's displayed on an iPad Home screen, the same application icon would look like this:

Note: You can prevent iOS from adding the shine to your application icon. To do this, you need to add the `UIPrerenderedIcon` key to your application's `Info.plist` file (to learn about this file, see "The Information Property List" in *iOS Application Programming Guide*).

The presence (or absence) of the added shine does not change the dimensions of your application icon.

Ensure your icon is eligible for the visual enhancements iOS provides. You should produce an image that:

- Has 90° corners
- Does not have any shine or gloss (unless you've chosen to prevent the addition of the reflective shine)
- Does not use alpha transparency

Give your application icon a discernible background. Icons with visible backgrounds look best on the Home screen primarily because of the rounded corners iOS adds. This is because uniformly rounded corners ensure that all the icons on a user's Home screen have a consistent appearance that invites tapping. If you create an icon with a background that disappears when it's viewed on the Home screen, users don't see the rounded corners. Such icons often don't look tappable and tend to interfere with the orderly symmetry of the Home screen that users appreciate.

Be sure your image completely fills the required area. If your image boundaries are smaller than the recommended sizes, or you use transparency to create "see-through" areas within them, your icon can appear to float on a black background with rounded corners.

For example, an application might supply an icon on a transparent background, like the blue star on the far left. When iOS displays this icon on a Home screen, it looks like the image in the middle (if no shine is added) or it looks like the image on the right (if shine is added).

An icon that appears to float on a visible black background looks especially unattractive on a Home screen that displays a custom picture.

Create a 512 x 512 pixel version of your application icon for display in the App Store. Although it's important that this version be instantly recognizable as your application icon, it can be subtly richer and more detailed. In other words, you should not simply scale up your application icon to create an icon for the App Store. There are no visual effects added to this version of your application icon.

If you're developing an application for ad-hoc distribution (that is, to be distributed in-house only, not through the App Store), you must also provide a 512 x 512 pixel version of your application icon. This icon identifies your application in iTunes.

iOS might also use this large image in other ways. In an iPad application, for example, iOS uses the 512 x 512 pixel image to generate the large document icon, if a custom document icon is not supplied.

Small Icons

Every application should supply an icon that iOS can display when the application name matches a term in a Spotlight search. Applications that supply settings should also supply this icon to identify them in the built-in Settings application.

This icon should clearly identify your application so that people can easily recognize it in a list of search results or in Settings.

For iPad, you supply separate icons for Settings and Spotlight search results. Create icons that measure::

- 50 x 50 pixels (for Spotlight search results)

  Note that the final visual size of this icon is 48 x 48 pixels. iOS trims 1 pixel from each side of your artwork and adds a drop shadow. Be sure to take this into account as you design your icon.
- 29 x 29 pixels (for Settings)

For iPhone and iPod touch, iOS uses the same icon for both Spotlight search results and Settings. If you do not provide this icon, iOS might shrink your application icon for display in search results and in Settings. Create icons that measure:

- 29 x 29 pixels
- 58 x 58 pixels (high resolution)

Document Icons

If your iOS application creates documents of a custom type, you might want to create a custom icon that identifies this type to users. If you don't provide a custom document icon, iOS creates one for you by default, using your application icon (including the added visual effects). For example, using the 72 x 72 pixel white star iPad application icon, a default document icon would look like this:

Optionally, you can provide custom artwork for iOS to use instead of your application icon. Because people will see your document icon in different places, it's best to design an image that's memorable and clearly associated with your application. Your custom artwork should be attractive, expressive, and detailed.

Note: For iPhone apps, you can find out how a default document icon might look, and learn how to create a custom document icon, in "Document Icons" in *iPhone Human Interface Guidelines*. The remainder of this section describes how to create a custom document icon for an iPad app only.

iOS uses two sizes of document icons for iPad applications: 64 x 64 pixels and 320 x 320 pixels. It's a good idea to create both sizes so that your document icons look good in different contexts.

For both sizes, the overall dimensions include specific amounts of padding, leaving a smaller "safe zone" for your artwork. It's essential to make sure your artwork fits well in these safe zones, or it may get cropped or scaled up.

Although your artwork can fill an entire safe zone, the upper right corner will always be partially obscured by the page curl effect that iOS adds. In addition, iOS adds a gradient that transitions from black (near the top, just below the page curl) to transparent (at the bottom edge).

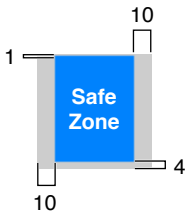
Important: If you decide to create a custom document icon for your iPad application, be sure to follow the guidelines in this section. If your icon is too large, too small, or improperly padded, the resulting document icon will not look good.

To create a 64 x 64 pixel document icon:

1. Create a 64 x 64 pixel image in PNG format.
2. Add the following margins to create the safe zone:
    - 1 pixel on top

- 4 pixels on bottom
- 10 pixels on each side

Your safe zone should look similar to the colored area shown below:



3. Place your custom artwork within the 44 x 59 pixel safe zone. The artwork can be centered, offset, or it can fill the entire safe zone. (Remember that iOS adds the page curl to the upper right corner and a gradient that runs from the page curl to the bottom edge.)

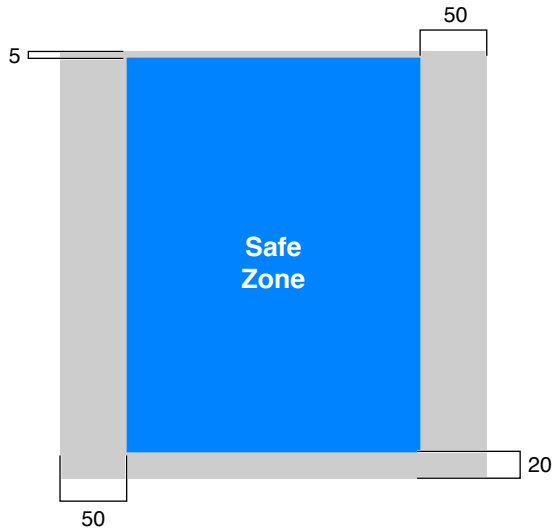
For example, if you supply an icon that looks like the image on the left in Figure 3-10, iOS creates a document icon that looks like the image on the right.

Figure 3-10 A 64 x 64 pixel document icon, before and after processing

To create a 320 x 320 pixel document icon:

1. Create a 320 x 320 pixel image in PNG format.
2. Add the following margins to create the safe zone:
   - 5 pixels on top
   - 20 pixels on bottom
   - 50 pixels on each side

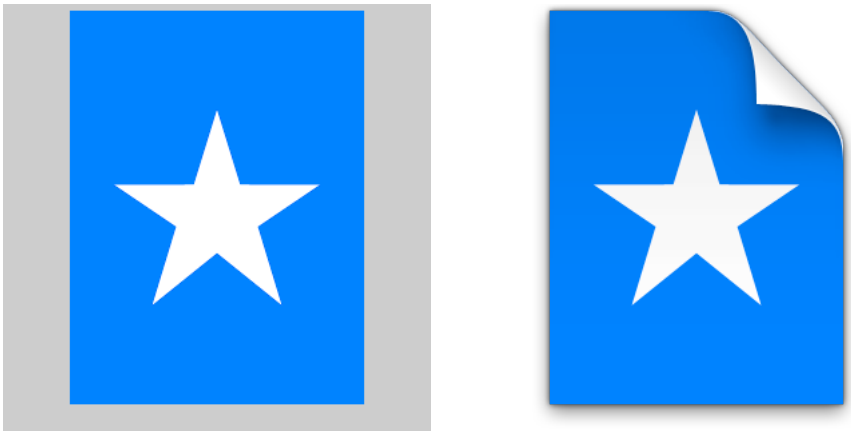
Your safe zone should look similar to the colored area shown below:



3. Place your custom artwork within the 220 x 295 pixel safe zone. The artwork can be centered, offset, or it can fill the entire safe zone. (Remember that the page curl will obscure some of the artwork in the upper right corner of the safe zone.)

For example, if you supply an icon that looks like the image on the left in Figure 3-11, iOS creates a document icon that looks like the image on the right.

Figure 3-11 A 320 x 320 pixel document icon, before and after processing



Web Clip Icons

If you have a web application or a website, you can provide a custom icon that users can display on their Home screens using the Web Clip feature. Users tap the icon to reach your web content in one easy step. You can create an icon that represents your website as a whole or an icon that represents a single webpage.

If your web content is distinguished by a familiar image or recognizable color scheme, it makes sense to incorporate it in your icon. However, to ensure that your icon looks great on the device, you should also follow the guidelines in this section. (To learn how to add code to your web content to provide a custom icon, see *Safari Web Content Guide*, available in the Safari Reference Library.)

For iPad create an icon that measures:

- 72 x 72 pixels

For iPhone and iPod touch create icons that measure:

- 57 x 57 pixels
- 114 x 114 pixels (high resolution)

As it does with application icons, iOS automatically adds some visual effects to your icon so that it coordinates with the built-in icons on the Home screen. Specifically, iOS adds:

- Rounded corners
- Drop shadow
- Reflective shine

For example, a simple 57 x 57 pixel webpage icon might look like this:

When it's displayed on an iPhone Home screen, the same icon would look like this:

Note: You can prevent the addition of all effects by naming your icon `apple-touch-icon-precomposed.png` (this is available in iOS 2.0 and later).

Ensure your icon is eligible for the visual enhancements iOS adds (if you want them). You should produce an image that:

- Has 90° corners
- Does not have any shine or gloss

Icons for Navigation Bars, Toolbars, and Tab Bars

As much as possible, you should use the system-provided buttons and icons to represent standard tasks in your application. For a complete list of standard buttons and icons, and guidelines on how to use them, see "System-Provided Buttons and Icons".

Of course, not every task your application performs is a standard one. If your application supports custom tasks users need to perform frequently, you need to create custom icons that represent these tasks in your toolbar or navigation bar. Similarly, if your application displays a tab bar that allows users to switch among custom application modes or custom subsets of data, you need to design tab bar icons that represent these modes or subsets.

Before you create the art for your icon, spend some time thinking about what it should convey. As you consider designs, aim for an icon that is:

- **Simple and streamlined**. Too many details can make an icon appear sloppy or indecipherable.
- **Not easily mistaken for one of the system-provided icons**. Users should be able to distinguish your custom icon from the standard icons at a glance.
- **Readily understood and widely acceptable**. Strive to create a symbol that most users will interpret correctly and that no users will find offensive.

Important: Be sure to avoid using images that replicate Apple designs and products in your artwork. These symbols are copyrighted and can change frequently.

After you've decided on the appearance of your icon, follow these guidelines as you create it:

- Use pure white with appropriate alpha.
- Do not include a drop shadow.
- Use anti-aliasing.
- If you decide to add a bevel, be sure that it is 90° (to help you do this, imagine a light source positioned at the top of the icon).

For toolbar and navigation bar icons, create an icon in the following sizes:

- For iPad:
  - About 20 x 20 pixels
- For iPhone and iPod touch:
  - About 20 x 20 pixels
  - About 40 x 40 pixels (high resolution)

For tab bar icons, create an icon in the following sizes:

- For iPad:
  - No larger than 48 x 32 pixels

- For iPhone and iPod touch:
  - No larger than 48 x 32 pixels
  - No larger than 96 x 64 pixels (high resolution)

These sizes represent the maximum dimensions a tab bar icon can have. If you provide a larger icon, iOS will center it and clip the excess.

Note: The icon you provide for toolbars, navigation bars, and tab bars is used as a mask to create the icon you see in your application. It is not necessary to create a full-color icon.

Don't include a pressed or selected appearance with your icons. iOS automatically provides these appearances for items in navigation bars, toolbars, and tab bars, so you do not need to provide them. Because these visual effects are automatic, you cannot change their appearance.

Give all icons in a bar a similar visual weight. Aim to balance the overall size, level of detail, and use of solid regions across all icons that can appear in a specific bar. In general, it does not look good to combine in the same bar icons that are large, blocky, and completely filled with icons that are small, detailed, and unfilled.

Launch Images

Because most iPad applications should 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 localized).

Supply a launch image to improve user experience; avoid using it as an opportunity to provide:

- An "application entry experience," such as a splash screen
- An About window
- Branding elements, unless they are a static part of your application's first screen

Because users are likely to switch among applications frequently, you should make every effort to cut launch time to a minimum, and you should design a launch image that downplays the experience rather than drawing attention to it.

For iPad create launch images that do not include the status bar region in the following sizes:

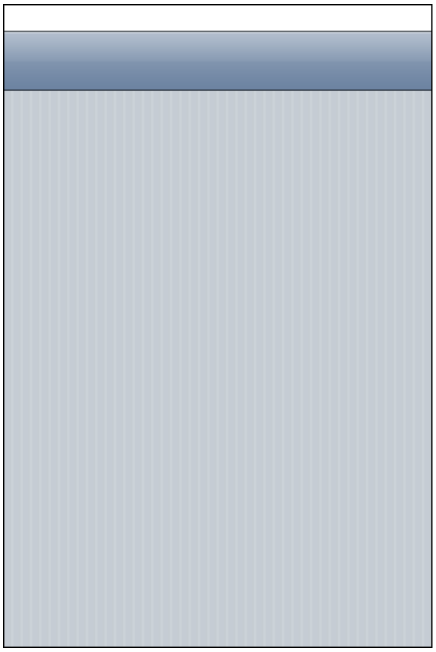
- 768 x 1004 pixels (portrait)
- 1024 x 748 pixels (landscape)

For iPhone and iPod touch create launch images that include the status bar region in the following sizes:

- 320 x 480 pixels
- 640 x 960 pixels (high resolution)

If you think that following these guidelines will result in a plain, boring launch image, you're right. Remember, the launch image is not meant to provide an opportunity for artistic expression; it is solely intended to enhance the user's perception of your application as quick to launch and immediately ready for use.

For example, the iPhone Settings launch image displays only the background of the application, because no other content in the application is guaranteed to be static.



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 principles and 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 iOS 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.

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. On iPad, the subtle appearance of the status bar does not compete with your application for the user's attention. As you can see in Figure 4-1, the small size of the status bar and the slightly rounded corners of the application's upper bar combine to make the status bar seem like part of the device background.

Figure 4-1 The status bar fades into the background on iPad

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. Figure 4-2 shows a navigation bar in Mail.

Figure 4-2 A navigation bar can contain navigational and other controls

To learn more about using a navigation bar in your code, see "Navigation Controllers".

Note: Navigation bars are used in some iPad applications, but they are not as prevalent as they are in iPhone applications. As you consider using a navigation bar in your iPad application, be sure to read "Flatten Your Information Hierarchy" (page 19).

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

- Either pane of a split view
- A popover
- A modal view
- A full-screen application view (although this usage 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. (For usage guidelines, see "Segmented Control" (page 55).)

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. For more information, see "Standard Buttons for Use in Toolbars and Navigation Bars" in *iPhone Human Interface Guidelines*. 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** typically appears at the bottom edge of an application screen and gives people the ability to switch between different subtasks, views, or modes. Figure 4-3 shows a tab bar in iTunes.

Figure 4-3 A tab bar switches subtasks, views, or modes

To learn more about using a tab bar in your code, see "Tab Bar Controllers".

In general, use a tab bar to organize information at the application level. A tab bar is well suited for use in the main application view because it's a good way to flatten your information hierarchy and provide access to several peer information categories or modes at one time.

In limited circumstances, it might make sense to use a tab bar in a split view pane or a popover if the tabs switch or filter the content within that view. However, it often works better to use a segmented control at the bottom edge of a popover or split view pane, because the appearance of a segmented control coordinates better with the popover or split view appearance. (For more information on using a segmented control, see "Segmented Control" (page 55).)

Avoid crowding the tab bar with too many tabs. Putting too many tabs in a tab bar can make it physically difficult for people to tap the one they want. Also, with each additional tab you display, you increase the complexity of your application. In general, try to limit the number of tabs in the main view or in the right pane of a split view to about seven. In a popover or in the left pane of a split view, up to about five tabs fit well.

Avoid creating a More tab. On iPad, a screen devoted solely to a list of additional tabs is a poor use of space.

Display the same tabs in each orientation. In portrait, the recommended seven tabs fit well across the width of the screen. In landscape orientation, you should center the same tabs along the width of the screen. This guidance also applies to the usage of a tab bar within a split view pane or a popover. For example, if you use a tab bar in a popover in portrait, it works well to display the same tabs in the left pane of a split view in landscape.

When you avoid changing the tabs or their spacing, you increase the visual stability of your application. As you can see in Figure 4-4, iTunes displays precisely the same tabs in landscape as it does in portrait (shown in Figure 4-3).

Figure 4-4 A tab bar in landscape displays the same number of tabs as it does in portrait

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. Figure 4-5 shows a toolbar in Maps.

Figure 4-5 A toolbar provides controls that act upon objects in the screen or view

To learn more about using a toolbar in your code, see "Displaying a Navigation Toolbar" in *View Controller Programming Guide for iOS*.

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. (For usage guidelines, see "Segmented Control" (page 55).)

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.

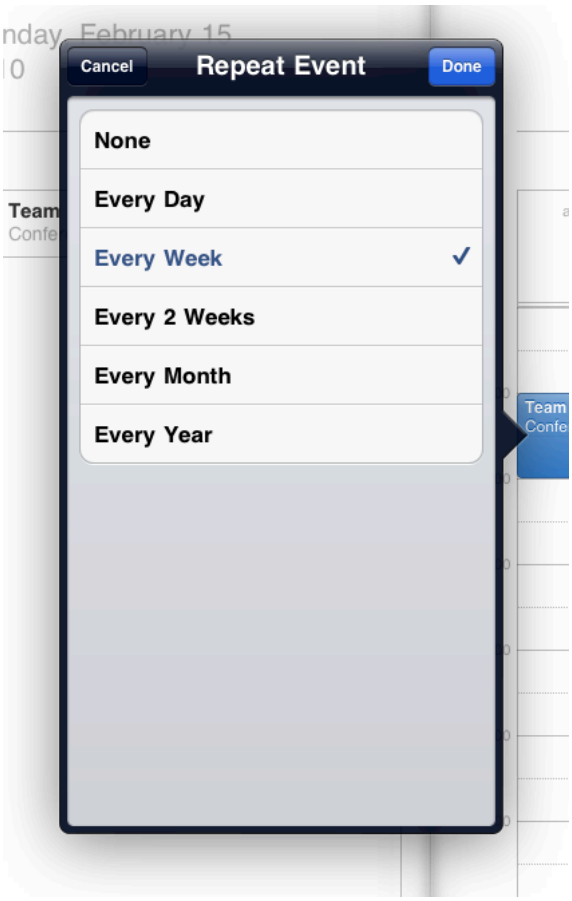
Content Views

In addition to the existing image, map, table, text, and web views, iOS 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 iOS.

Popover

A **popover** is a transient view that can be revealed when people tap a control or an onscreen area. Figure 4-6 shows a popover in Calendar.

Figure 4-6 A popover provides choices or functionality in a temporary view



To learn more about using a popover in your code, see "Using Popovers to Display Content" in *iPad Programming Guide*.

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, toolbars, or tab bars.
- 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 border.

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.

  A popover that provides menu or inspector functionality should close when people tap anywhere outside its bounds, including the control that reveals the popover. 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 their input when they tap outside a popover's borders, just as you would if they tapped Done.

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

Ensure that the popover arrow points as directly as possible to the element that revealed it. Doing 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. You should not display more than one popover (or custom view designed to look and behave like a popover) at the same time. In particular, you should avoid displaying a cascade or hierarchy of popovers simultaneously, in which one popover emerges from another.

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.

Prefer standard UI controls and views within a popover. In general, popovers look best, and are easier for users to understand, when they contain standard controls and views.

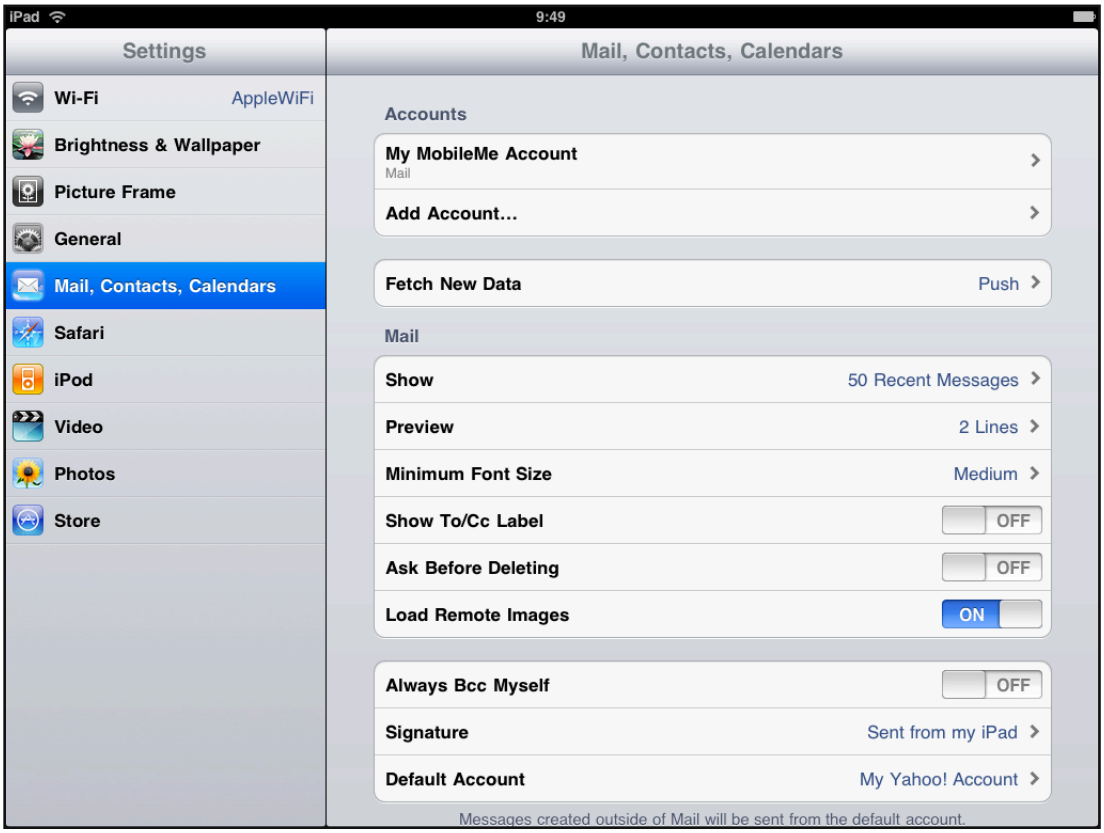
Take care if you combine a customized background color or texture with standard controls and views. Be sure the standard UI elements look good and are easy to read when they're seen on top of your custom background appearance.

If appropriate, change a popover's size while it remains visible. You might want to change a popover's size if you use it 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. Figure 4-7 shows a split view in Settings.

Figure 4-7 A split view contains two related views



To learn more about using a split view in your code, see "Creating a Split View Interface" in *iPad Programming Guide*.

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 and Settings.

- 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.
- 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, toolbars, or tab bars. (Note, however, that it does not make sense to put a navigation bar in both panes of a split view at the same time.)

In general, when an application uses a split view in landscape, it displays the contents of the left pane in a popover when it rotates to portrait. However, this does not mean that you must follow this pattern. If it makes sense in your application, you can design your UI 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.

You can use your own fonts in your iPad application. Include the fonts in your application bundle, and specify them using the `UIAppFonts` key in your `Info.plist` file. To learn about this file, see "The Application Bundle" in *iOS Application Programming Guide*; for guidelines specific to iPad, see "Updating Your Info.plist Settings" in *iPad Programming Guide*.

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.

To learn more about using a text view in your code, see *UITextView Class Reference*.

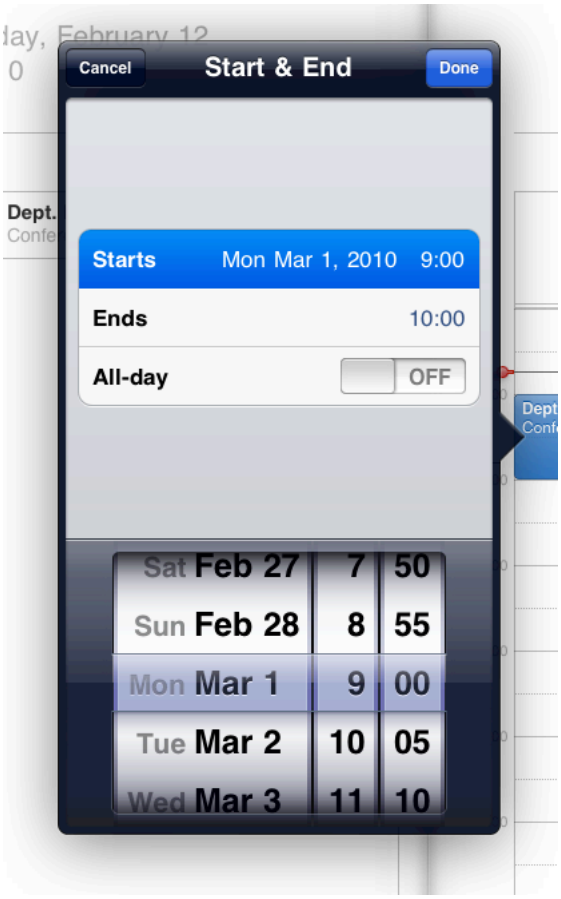
Controls

All UIKit controls introduced in previous versions of iOS 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

A **date and time picker** allows people to select a value that can consist of multiple parts, such as a date. A picker is a generic version of a date and time picker, which allows people to select from a set of single-part values. Figure 4-8 shows a date and time picker in Calendar.

Figure 4-8 A date and time picker makes it easy to select values with multiple parts



To learn more about using a picker or a date and time picker in your code, see *UIPickerView Class Reference* and *UIDatePicker Class Reference*, respectively.

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 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 of a utility application to reveal configuration options.

To learn more about using an Info button in your code, see *UIButton Class Reference*.

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.

To learn more about using a page indicator in your code, see *UIPageControl Class Reference*.

Search Bar

A **search bar** is a field that accepts text from people, which your application can use as input for a search. When the user taps a search bar, a keyboard appears; when the user is finished typing search terms, the input is handled in an application-specific way. Figure 4-9 shows a search bar in a Mail popover.

Figure 4-9 A search bar accepts input from users

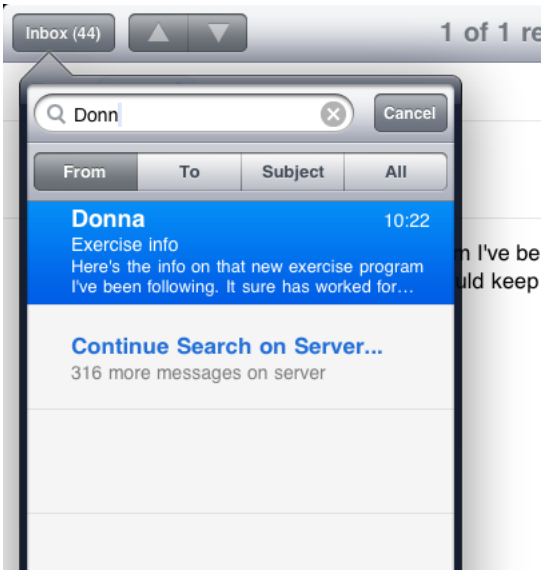
To learn more about using a search bar in your code, see *UISearchBar Class Reference*.

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.

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, as shown in Figure 4-10.

Figure 4-10 A scope bar helps people focus their searches



To learn more about using a scope bar in your code, see *UISearchDisplayController Class Reference*.

Segmented Control

A **segmented control** is a linear set of segments, each of which functions as a button that can display a different view. Figure 4-11 shows a segmented control in the iTunes toolbar.

Figure 4-11 A segmented control can provide access to different perspectives or views

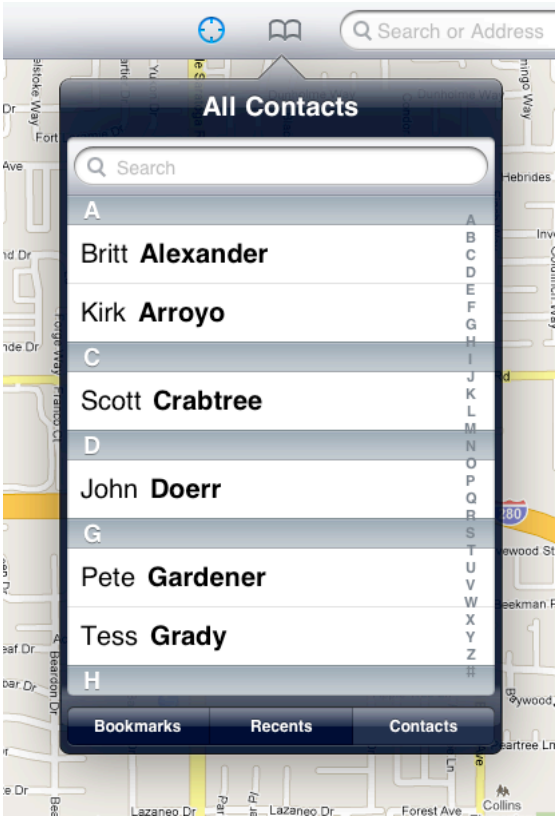
To learn more about using a segmented control in your code, see *UISegmentedControl Class Reference*.

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, the iPod application uses a segmented control in a bottom-edge toolbar to let people group their libraries in different ways.

Another example of a view-specific segmented control is in the Maps bookmarks popover, shown in Figure 4-12. Because the segmented control is contained in the popover, people understand that it acts upon the popover contents, and does not filter information or change modes in the rest of the application. If you need to provide mode-switching or filtering within a popover or split view pane, consider using a segmented control instead of a tab bar.

Figure 4-12 A segmented control works well in a popover to filter information or switch modes



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. Figure 4-13 shows the action sheet that appears when the user taps the Reply button in Mail.

Figure 4-13 An action sheet is displayed inside a popover

To learn more about using an action sheet in your code, see "Using Popovers to Display Content" in *iPad Programming Guide*.

An action sheet can be displayed with or without animation.

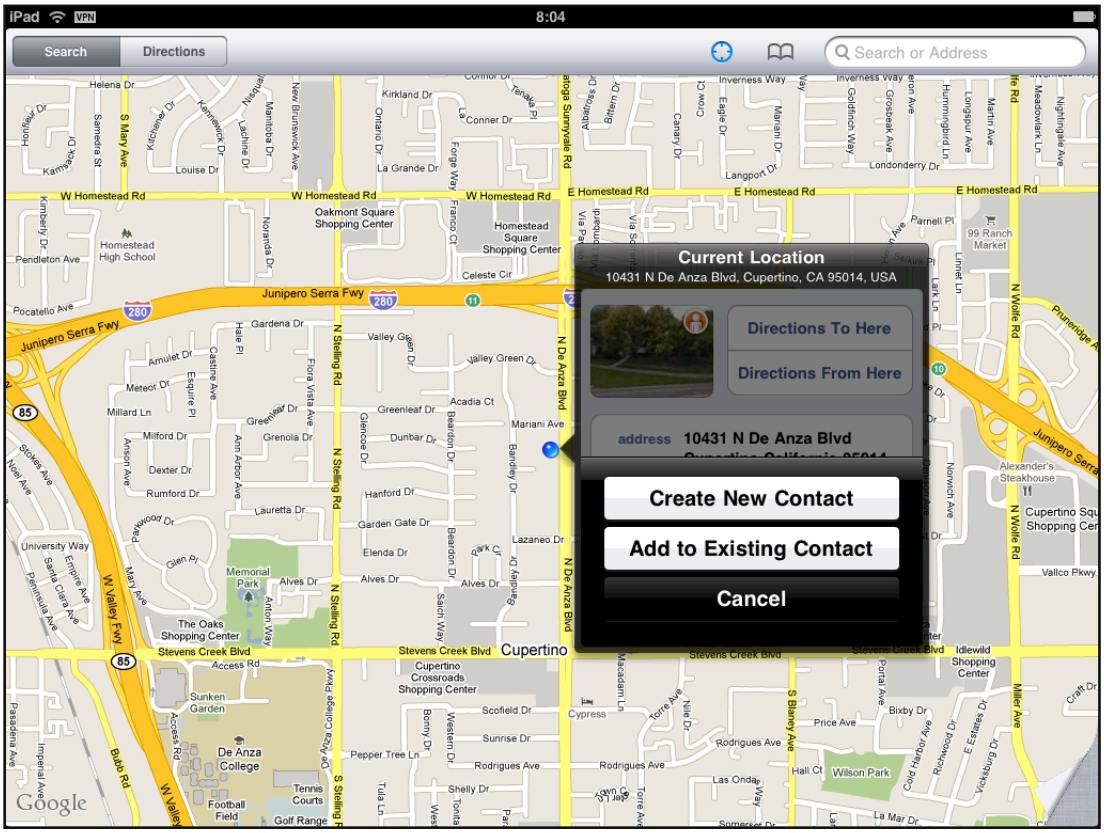
- **Without animation, an action sheet and its popover appear simultaneously**. Display an action sheet without animation to provide alternatives related to a task that the user initiates from outside a popover. When you display an action sheet this way, the popover's arrow points to the control or area 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. (Note that there is no Cancel button in the action sheet in Figure 4-13.)

- **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. Figure 4-14 shows an action sheet that appears in the location popover in Maps.

Figure 4-14 An animated action sheet typically includes a Cancel button



Alert

An **alert** gives people important information that affects their use of the application or the device. Alerts behave the same on iPad as they do on iPhone.

On iOS-based devices, alerts should seldom be necessary and are best used to:

- Describe a problem and give people a choice of ways to handle it
- Give people a chance to accept or reject an outcome that is potentially dangerous

If you merely want to increase the visibility of some information, especially information related to the standard functioning of your application, you should avoid using an alert. Instead, you should design an eye-catching way to display the information that harmonizes with your application's style.

To learn more about using an alert in your code, see *UIAlertView Class Reference*.

You can specify the text, the number of buttons, and the button contents in an alert. You can't customize the width or the background appearance of the alert view itself, or the alignment of the text (it's center-aligned).

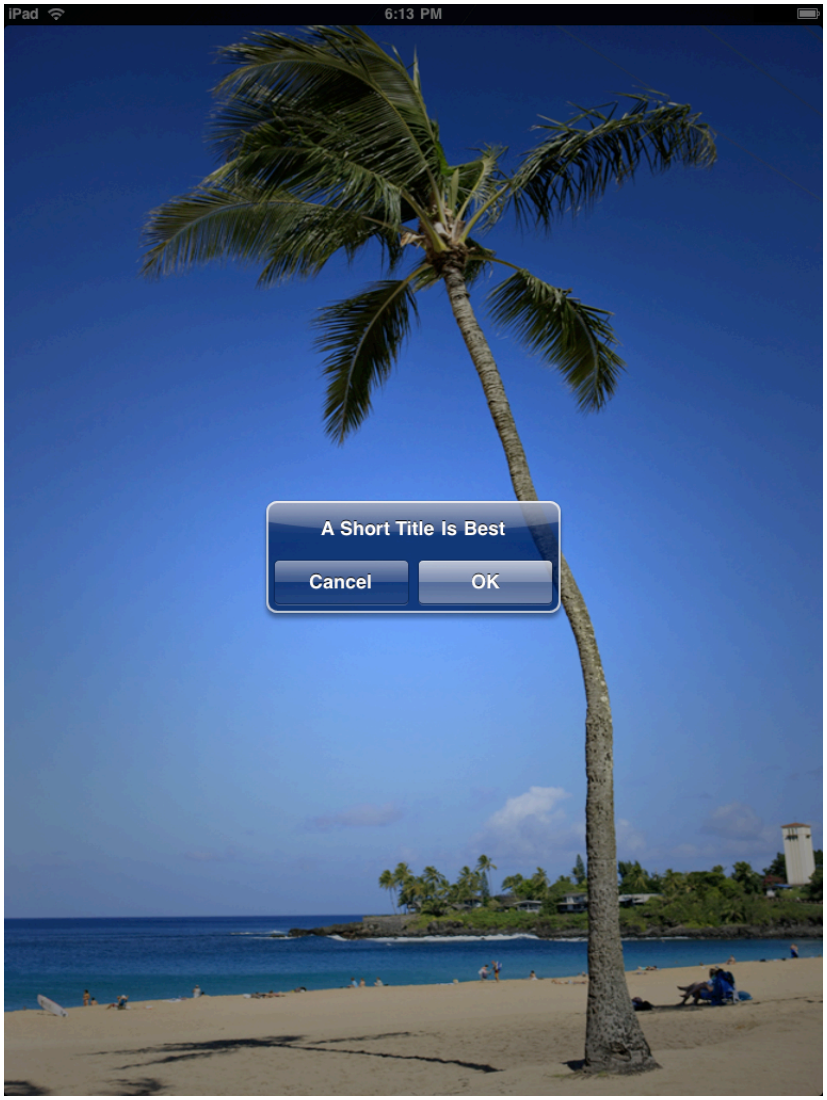
Note: As you read these guidelines, be aware of the following definitions:

- **Title-style capitalization** means that every word is capitalized, except articles, coordinating conjunctions, and prepositions of four or fewer letters.
- **Sentence-style capitalization** means that the first word is capitalized, and the rest of the words are lowercase, unless they are proper nouns or proper adjectives.

The alert title (and optional message) should succinctly describe the situation and explain what people can do about it. Ideally, the text you write gives people enough context to understand why the alert has appeared and to decide which button to tap.

As you compose the required alert title:

- Keep the title short enough to display on a single line, if possible. A long alert title is difficult for people to read quickly, and it might get truncated or force the alert message to scroll.



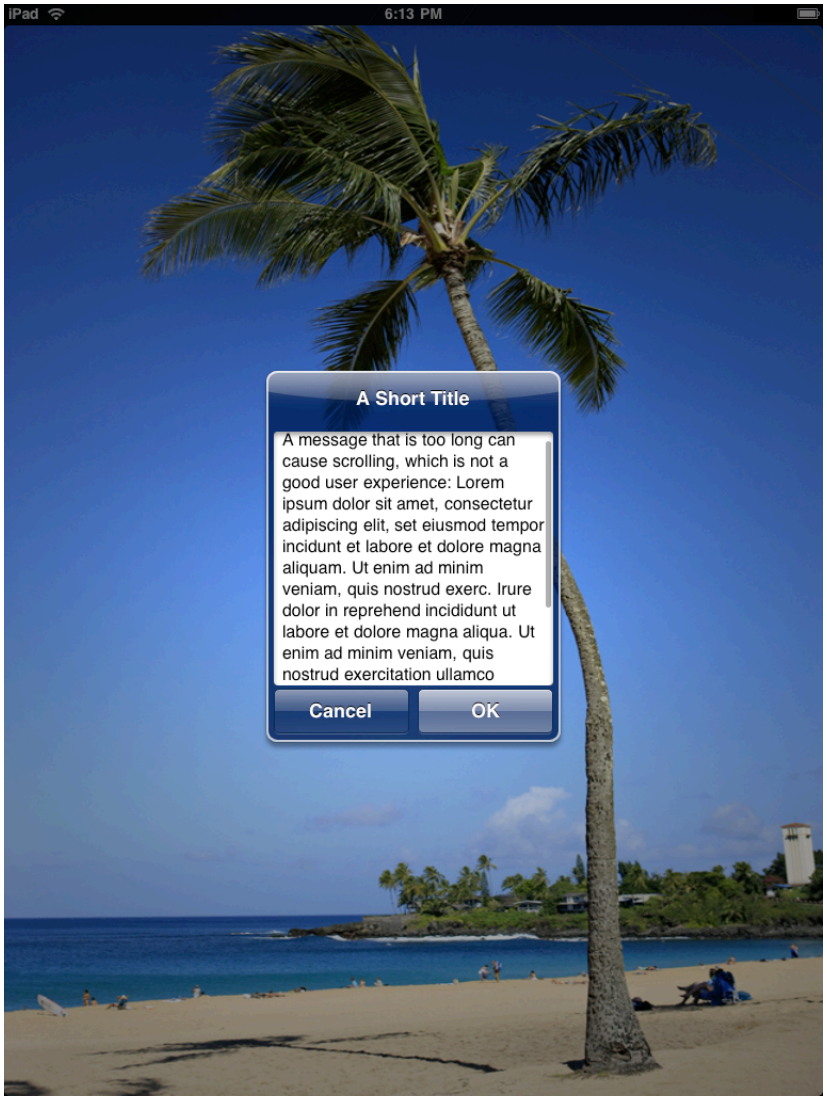
- Avoid single-word titles that don't provide any useful information, such as "Error" or "Warning."

- When possible, use a sentence fragment. A short, informative statement is often easier to understand than a complete sentence.
- Don't hesitate to be negative. People understand that most alerts tell them about problems or warn them about dangerous situations. It's better to be negative and direct than it is to be positive but oblique.
- Avoid using "you," "your," "me," and "my" as much as possible. Sometimes, text that identifies people directly can be ambiguous and can even be interpreted as an insult.
- Use title-style capitalization and no ending punctuation when:
  - The title is a sentence fragment
  - The title consists of a single sentence that is not a question
- Use sentence-style capitalization and an ending question mark if the title consists of a single sentence that is a question. In general, consider using a question for an alert title if it allows you to avoid adding a message.
- Use sentence-style capitalization and appropriate ending punctuation for each sentence if the title consists of two or more sentences. A two-sentence alert title should seldom be necessary, although you might consider it if it allows you to avoid adding a message.

If you provide an optional alert message:

- Create a short, complete sentence that uses sentence-style capitalization and appropriate ending punctuation.

- Avoid creating a message that is too long. If possible, keep the message short enough to display on one or two lines. If the message is too long it will scroll, which is not a good user experience.



Avoid lengthening your alert text with descriptions of which button to tap, such as "Tap View to see the information." Ideally, the combination of unambiguous alert text and logical button labels gives people enough information to understand the situation and their choices. However, if you must provide detailed guidance, follow these guidelines:

- Be sure to use the word "tap" (not "touch" or "click" or "choose") to describe the selection action.
- Don't enclose a button title in quotes, but do preserve its capitalization.

Prefer a two-button alert. A two-button alert is often the most useful, because it is easiest for people to choose between two alternatives. It is rarely a good idea to display an alert with a single button because such an alert is merely informative; it does not give people any control over the situation. An alert that contains three or more buttons is significantly more complex than a two-button alert, and should be avoided if possible.

Use alert button colors appropriately. Alert buttons are colored either dark or light. In an alert with two buttons, the button on the left is always dark-colored and the button on the right is always light-colored. In a one-button alert, the button is always light-colored.

- In a two-button alert that proposes a potentially risky action, the button that cancels the action should be on the right (and light-colored).
- In a two-button alert that proposes a benign action that people are likely to want, the button that cancels the action should be on the left (and dark-colored).

Note: A Cancel button may be either light-colored or dark-colored and it may be on the right or the left, depending on whether the alternate choice is destructive. Be sure to properly identify which button is the Cancel button in your code (for more information, see *UIAlertView Class Reference*).

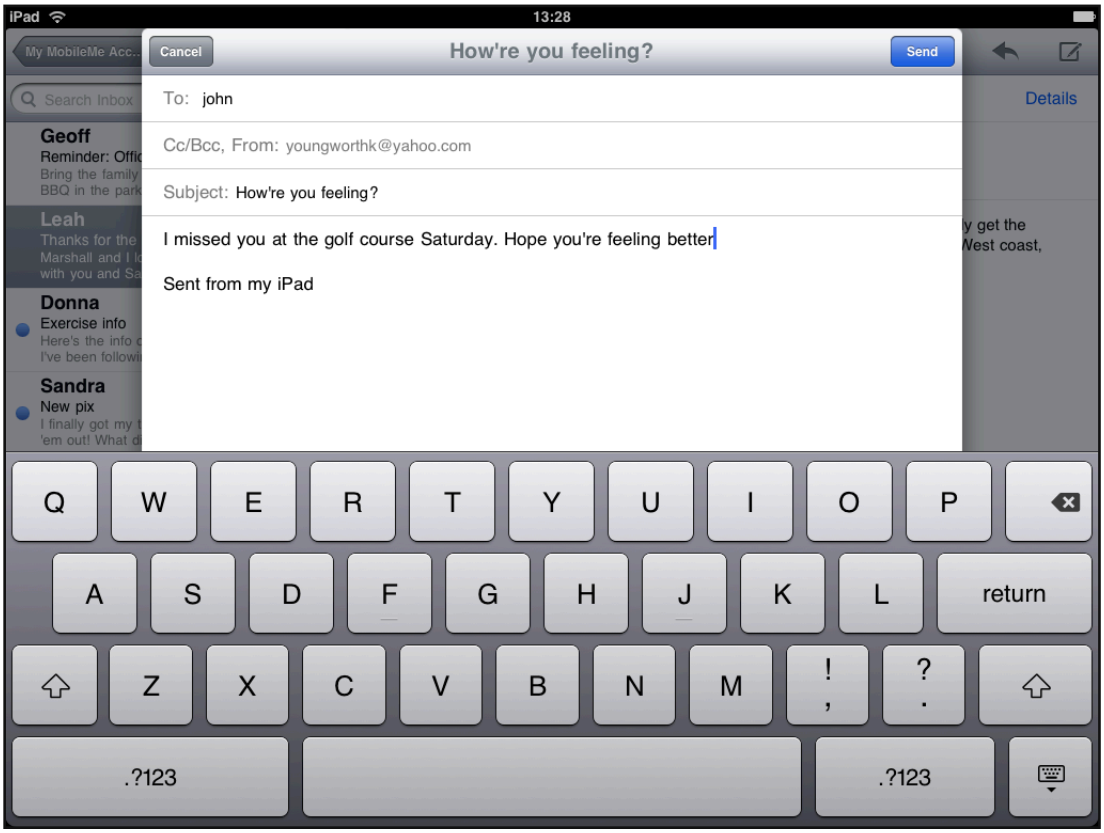
Give alert buttons short, logical titles. The best titles consist of one or two words that make sense in the context of the alert text. Follow these guidelines as you create titles for alert buttons:

- As with all button titles, use title-style capitalization and no ending punctuation.
- Prefer verbs and verb phrases, such as “Cancel,” “Allow,” “Reply,” or “Ignore” that relate directly to the alert text.
- Prefer “OK” for a simple acceptance option if there is no better alternative. Avoid using “Yes” or “No.”
- Avoid “you,” “your,” “me,” and “my” as much as possible. Button titles that use these words are often both ambiguous and patronizing.

Modal View

A **modal view** (that is, a view presented modally) provides self-contained functionality in the context of the current task or workflow. Figure 4-15 shows a modal view in Mail.

Figure 4-15 A modal view presents a self-contained task



To learn more about using a modal view in your code, see "Configuring the Presentation Style for Modal Views" in *iPad Programming Guide*.

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, the iPod application 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 (as shown in Figure 4-15).
- **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.
- **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-full-screen view.

In addition to the existing transition styles (vertical and flip), iOS 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 27) for some guidance.

Edit Menu Additions

In general, the items you add to the edit menu should edit, alter, or otherwise act directly upon the user's selection. People expect the standard edit menu items to act upon text or objects within the current context, and it's best when your custom menu items behave similarly.

Note: If you need to enable actions that use the selected text or object in a way that's external to the current context, it's better to use an action sheet. For example, if you want to allow people to share their selection with others, you might display an action sheet that lists social networking sites to which they can send their selection. To learn about the usage guidelines for action sheets, see "Action Sheet" (page 56).

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 describe the action to be performed. 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

iOS 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
2010-09-08	Included existing guidelines for alerts and custom icons and images. Also clarified how many popovers can be visible at one time.
2010-08-03	Clarified how iOS handles custom document icons.
2010-05-07	Corrected launch image dimensions and clarified the behavior of taps outside a popover.
2010-04-03	New document describing how to design a compelling user interface for an iPad application.

Document Revision History