Adopting the Aqua Interface

This document describes what you need to do in order to revise your application for the Mac OS X interface (known as Aqua). This document is primarily intended for developers who want their application to look and behave correctly when running under Mac OS X and assumes the reader is familiar with *Macintosh Human Interface Guidelines*, which can be found at:
<http://developer.apple.com/techpubs/macos8/HumanInterfaceToolbox/HumanInterfaceGuide/humaninterfaceguide.html>

Note: This draft document applies as of Developer Preview 4. Note that control sizes are now more compatible with platinum appearance metrics, but that layout guidelines for Mac OS X differ somewhat from Mac OS 8 and 9. Carbon developers, in particular, should review the material in "Control Layout Guidelines" (page 16) and "Menu Layout" (page 26). Apple reserves the right to make changes in future releases.

New in Aqua

This section describes the Aqua interface and highlights how it differs from the Platinum Appearance introduced with Mac OS 8.

Window Layering

The window layering model in Mac OS X has changed. In previous versions of Mac OS, all windows belonging to a particular application are in the same layer.

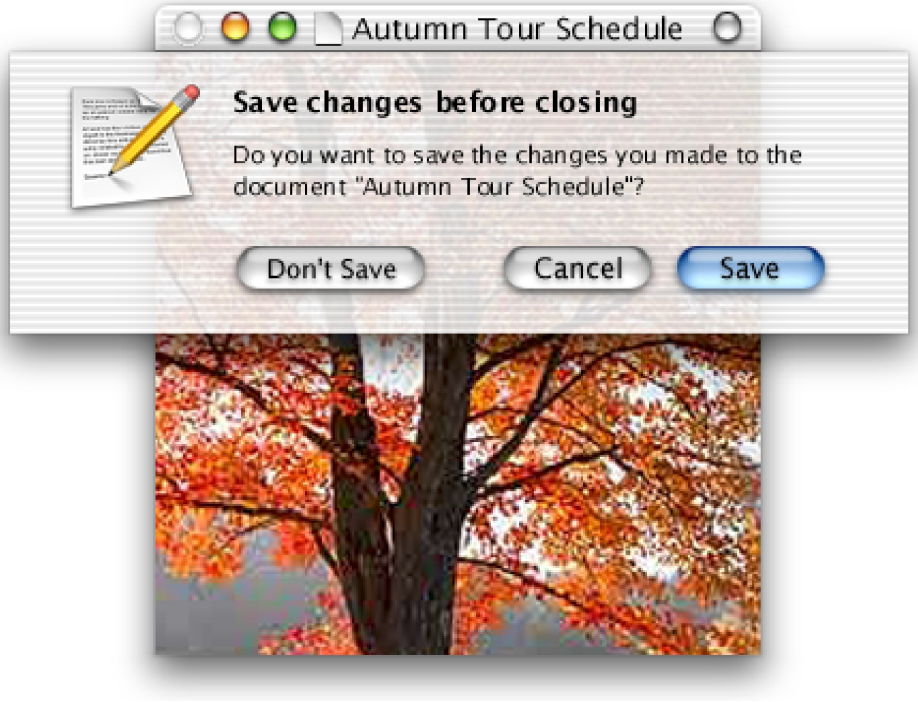
In Mac OS X, document windows and an application's main windows are in their own individual layers. This allows a user to have documents from different applications interleaved. Three or four TextEdit documents, three or four Preview documents, and so on, may be interleaved so that the user can click on any one of them, bring them to the front, and not disturb the layering of any other window. The user determines the layering order of windows by when each window was last accessed. It is possible to bring all windows of a given application to the front or near the front, but the ordering is generally determined by what the user accesses.

In the section called “Sheets”, a new type of dialog is introduced. This new type of dialog is designed to ensure that a dialog does not get “buried” when a user accesses windows from different applications.

Sheets

Sheets are a new way to present certain dialogs to the user. Sheets are “attached” to a particular document or window; they are displayed as an animation that appears to emerge from the window’s title bar. This attachment ensures that the user never loses track of the association of a sheet with its pertinent window, even when other windows are opened and closed. The ability to keep a dialog attached to its pertinent window enables users to take full advantage of Mac OS X’s window layering model. Figure 1-1 (page 3) shows an example of a sheet presenting a modal dialog.

Figure 1-1 Example of using sheet to display a modal dialog



Sheet Behavior

You lay out sheets like any other dialog in Mac OS X. Sheet behavior is determined by the Toolkit, but sheets are basically modal dialog boxes that emerge from the title bar of a window. If a sheet is attached to a window near the edge of the screen, the sheet moves the window away from the edge until the dialog is dismissed, at which time the window returns to its previous position.

Only one sheet may be open for any window at any one time. A sheet presenting a modal dialog prevents any other operation on that window until the dialog is dismissed. If the user opens a sheet presenting a dialog, then requests a second sheet, the first sheet closes before the second sheet opens.

When to Use Sheets

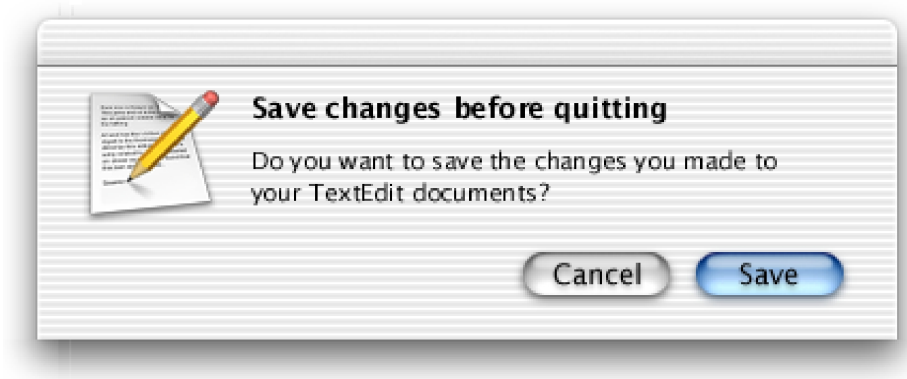
You should use sheets for modal dialogs or for modeless dialogs that the user dismisses before proceeding with work. Here are some examples of when to use sheets:

- A modal dialog that is specific to a particular document, such as saving or printing.
- A modal dialog that is specific to a single-window application that does not create documents. A single-window installer program might use a sheet to request acceptance of a licensing agreement from the user, for example.
- Other window-specific modeless dialogs, when the user benefit of associating a dialog with a specific document or window outweighs the benefit of keeping the dialog modeless.

When Not to Use Sheets

Do not use sheets for dialogs that apply to several windows. For example, the Save Changes dialog that your application displays when the user attempts to quit with unsaved work should be a standard movable modal dialog box (as shown in Figure 1-2), because it may pertain to several open windows. Sheets are strictly intended to be used in situations when a particular dialog is associated only with the window to which it is attached.

Figure 1-2 Example of modal dialog box used in preference to sheet



Sheets are not appropriate for modeless operations where the dialog should be left open to allow the user to observe the effects of changes applied. Such tasks are better suited to separate modeless dialog boxes or palettes.

Sheets should not be used in place of a preference dialog. Most applications use multiple panels to display preferences to the user. Application preferences should be presented in a consistent manner across all applications.

Font Usage

Mac OS X supports six different font usages, as described in Table 1-1 (page 5).

Table 1-1 Mac OS X font usages

Preference	Font
System Font	Lucida Grande Regular 13 pt
System Font (Emphasized)	Lucida Grande Bold 13 pt
Small System Font	Lucida Grande Regular 11 pt
Small System Font (Emphasized)	Lucida Grande Bold 11 pt
Application Font	Lucida Grande Regular 13 pt
Label Font	Lucida Grande Regular 10 pt

Note: In Interface Builder, the Message font is the System font, the Informational font is the Small System font, and the Tool Tip font is the Small System font.

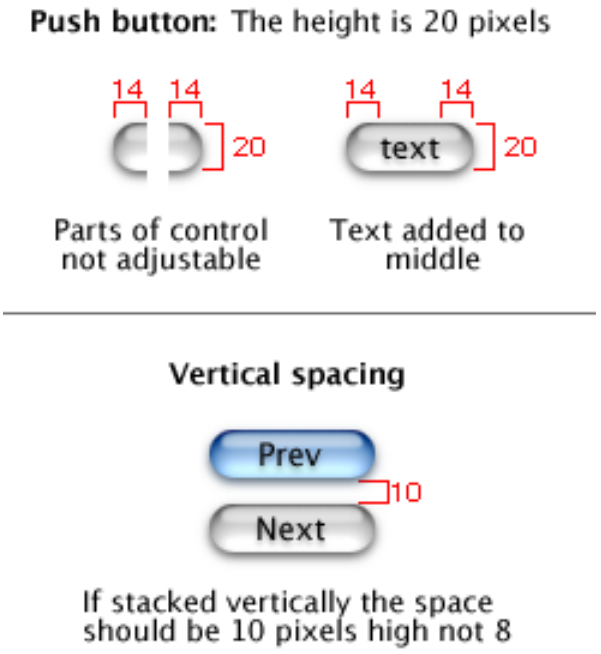
Control Sizing

The Toolbox determines the height of all controls except for bevel buttons, group boxes, and text input fields.

Push Button Sizing

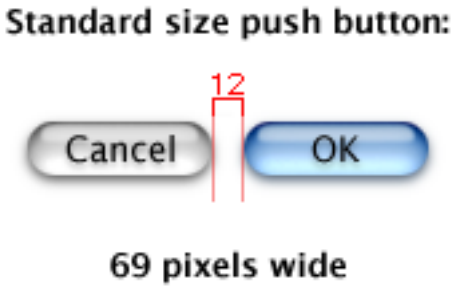
Push buttons have a fixed height of 20 pixels. They have fixed-sized end caps (14 pixels wide) that clip text if you don't specify a button that is wide enough to accommodate the text string. For an illustration of push button dimensions, see Figure 1-3 (page 6). If you need to use a font that is larger than the system font, use a bevel button instead of a push button.

Figure 1-3 Push button dimensions



The standard width for OK and Cancel buttons is 69 pixels, as shown in Figure 1-4 (page 7).

Figure 1-4 Example of standard push buttons



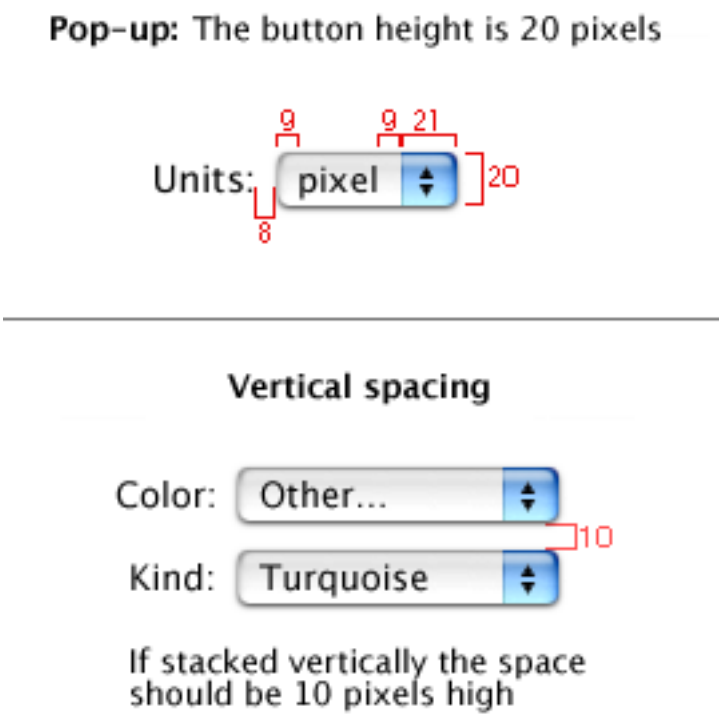
Using Color in Push Buttons

All push buttons are clear except the default button (the button selected by pressing the Return Key), which uses the default color (in addition to pulsing). For example, in a dialog box containing a default OK button and a Cancel button, the Cancel button is clear and the OK button is colored and pulsing. If you do not specify a default button in a dialog box, you may wish to designate a "main" push button that you draw in the default color. This may be useful if one of the push buttons is a higher-priority option for the user, for example. Designating a main button is unusual; most dialog boxes use clear buttons exclusively. Figure 1-2 (page 4) shows an example of push button color use. We do not encourage the use of color in more than one button in the same dialog box.

Pop-up Button Sizing

Pop-up buttons are 20 pixels high. The label portion of a pop-up button begins 9 pixels from the left edge of the button and ends 9 pixels from the control portion of the button, as illustrated in Figure 1-5 (page 8).

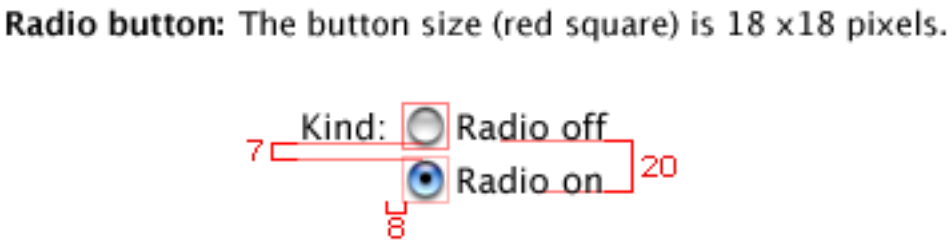
Figure 1-5 Pop-up button dimensions



Radio Button and Checkbox Sizing

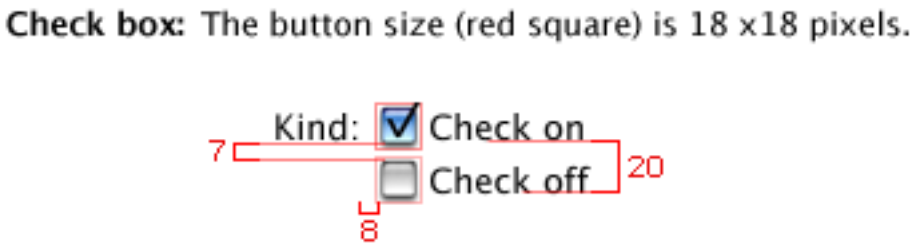
Radio button dimensions are shown in Figure 1-6 (page 9). Note that the size indicated for each control includes the shadow. The actual size of the hit-testing region is 12 x 12 pixels.

Figure 1-6 Radio button dimensions



Checkbox dimensions are shown in Figure 1-7 (page 9). Note that the size indicated for each control includes the shadow. The actual size of the hit-testing region is 12 x 12 pixels.

Figure 1-7 Check box dimensions



Bevel Button Sizing

Mac OS X bevel buttons use the same bevel height for all sizes of button, as shown in Figure 1-8 (page 10). The Carbon Control Manager provides 3 different bevel height settings, so Carbon developers should specify the small bevel height when creating a bevel button in order to provide the largest area in which to draw icons. We recommend you keep a border of 6 pixels around the icon; if your icon is too large, the button may appear dark or greyed-out because the icon obscures the button's highlighting. The minimum size for a bevel button is 20 x 20 pixels.

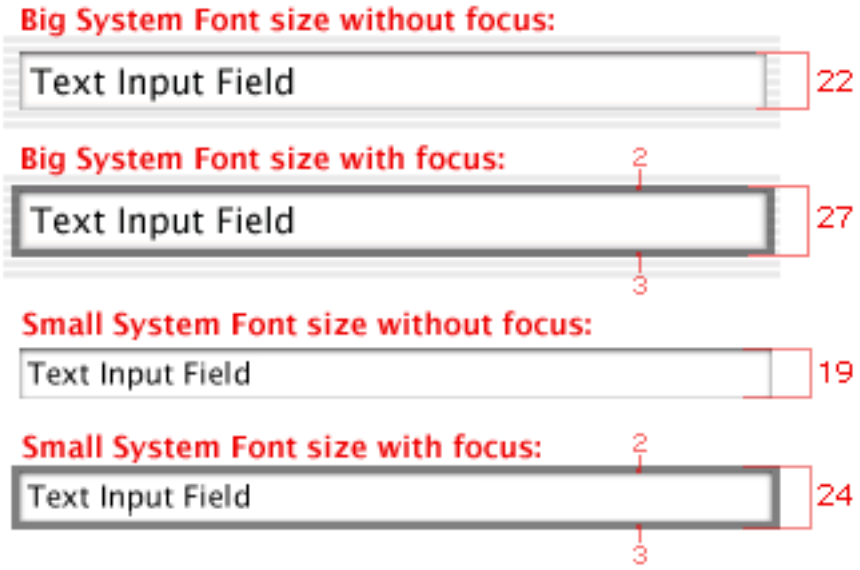
When you use a Carbon Control Manager function to specify a bevel button with an icon and a label, the Toolbox automatically places the text label below the icon. For example, bevel buttons used in the Finder toolbar have 32 x 32 pixel icons and 10 point labels, as shown in Figure 1-8 (page 10).

Figure 1-8 Bevel button dimensions

Text Input Field Sizing

Text input fields should be 22 pixels high to accommodate the System font (which is 16 pixels high without including line spacing). When the user selects text in a text input field, the selection rectangle is also 16 pixels high. The relationship of the text rectangle to the field is shown in Figure 1-9 (page 11). If you specify the small System font, the field dimensions are reduced proportionally. When a text input field has keyboard focus, a dark, translucent rectangle (which is two pixels wide at the top and three pixels wide on the other three sides) appears around the outside edge of the field. You should adjust your layout to make room for this focus ring,

Figure 1-9 Text input field dimensions



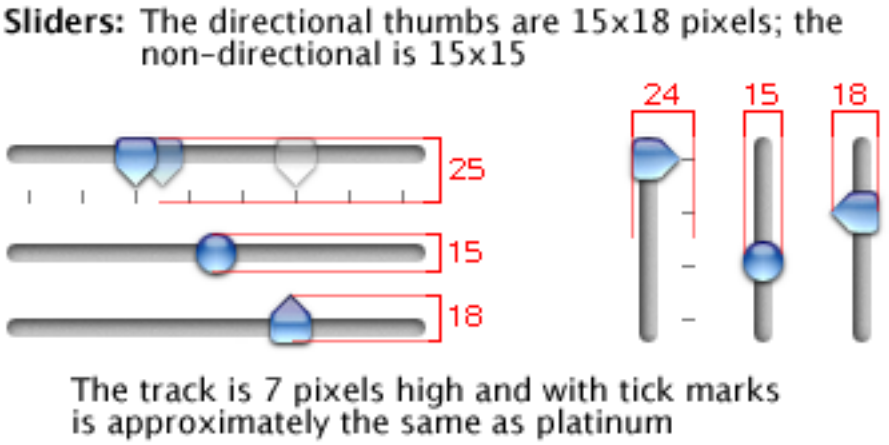
List Element Sizing

List elements, including Browser and Table views, should be vertically sized to show only full lines of text. If you use the default font, this works out to (number of lines x 16) + 4 pixels.

Slider Sizing

Sliders have a 7-pixel track but the thumb extends the size 2 pixels beyond platinum appearance metrics for both vertical and horizontal orientations. You can specify tick marks and labels. Directional and non-directional thumbs are available. Examples of vertical and horizontal sliders are shown in Figure 1-10 (page 12).

Figure 1-10 Slider dimensions



Tab Control Sizing

Tab controls are horizontally oriented and located at the top of the pane, as in platinum appearance. At this time, tab controls support the Large System Font only. Individual tabs are automatically centered within the pane and can be inset up to 24 pixels on either side of the pane. You can calculate the width of the tab by determining the width of the tab label in pixels and adding 12 pixels to each side. Note that the accent bar adds 6 pixels to the height of the active tab. You can also define the pane so that space remains for controls such as push buttons below the pane. An example of tab control dimensions is shown in Figure 1-11 (page 13).

Figure 1-11 Tab control dimensions

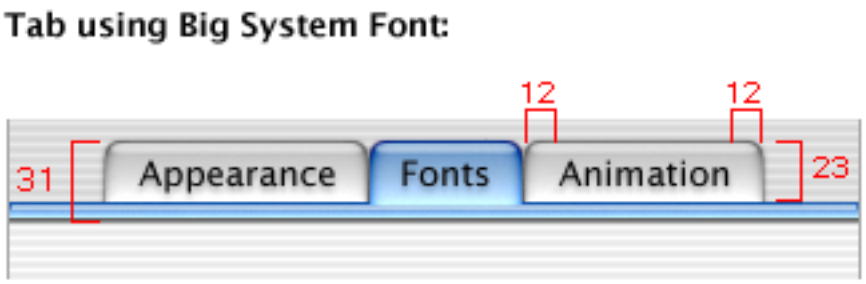


Figure 1-12 (page 14) shows an example of tab controls used from one edge of a pane to the other.

Figure 1-12 Tabs located from edge-to-edge

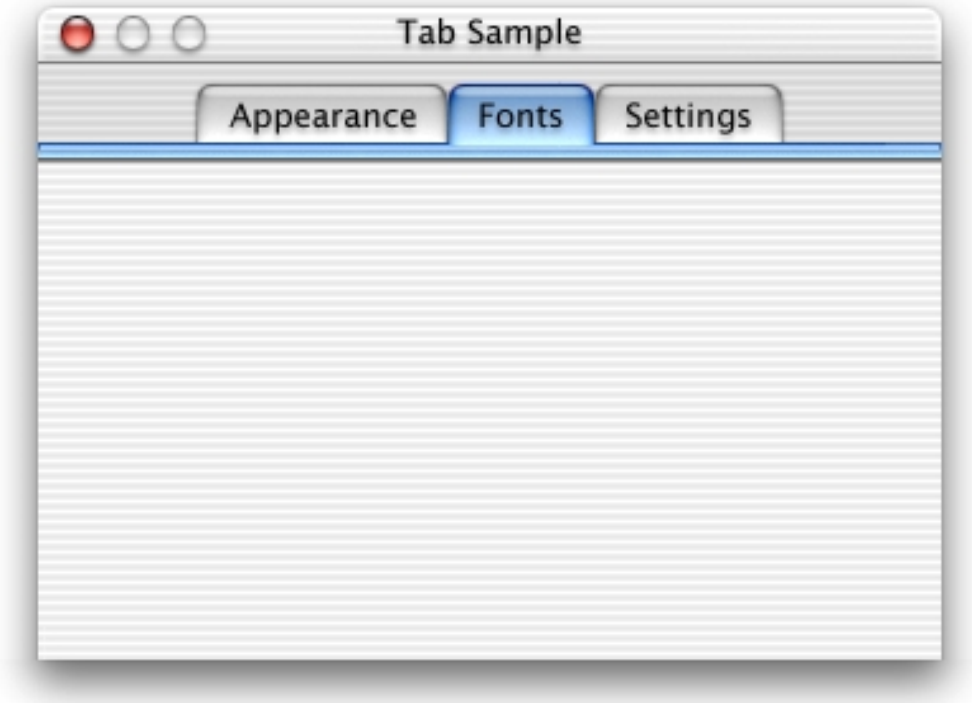
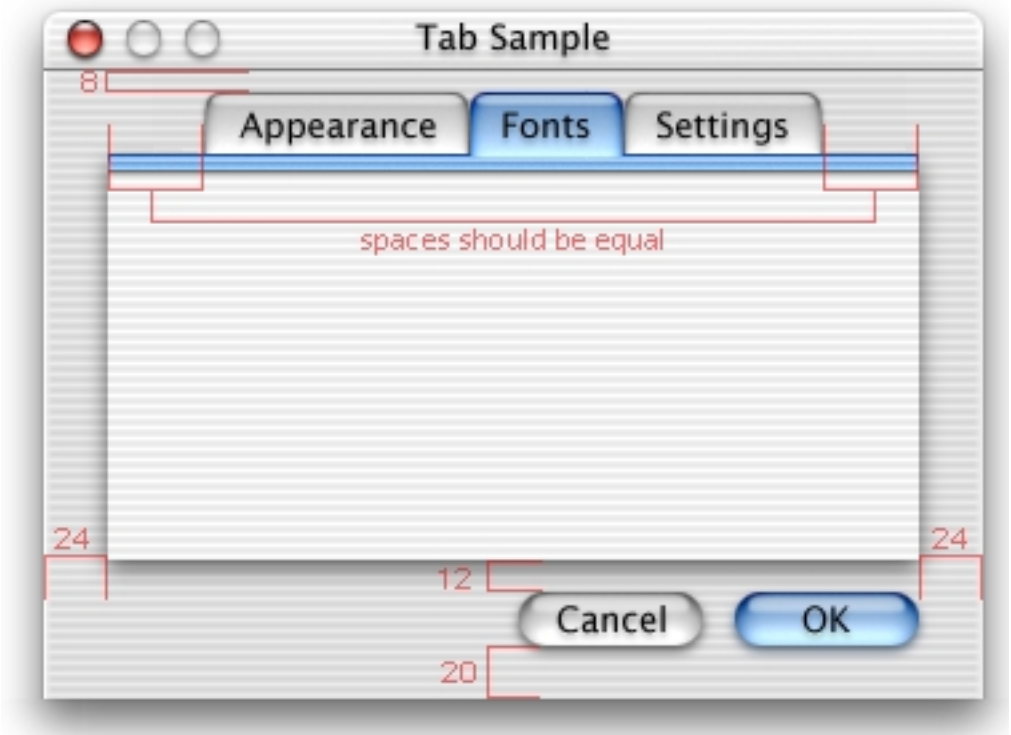


Figure 1-13 (page 15) shows an example of tab controls inset within a pane.

Figure 1-13 Tabs inset from edge of pane



Control Layout Guidelines

This section describes a basic set of control layout guidelines. In an effort to simplify the process of resizing and repositioning existing layouts, most values are based on a multiple of 4 pixels. These guidelines maintain many of the default control sizes defined in Interface Builder; exceptions are described below. When creating or changing dialog box layouts, remember to use the default font preferences, as described in "Font Usage" (page 5).

Control Positioning in Dialog Boxes

All spacing between dialog box elements involves a multiple of 4 pixels; generally, this translates to 4, 8, 12, 16, 20, or 24 pixel spacing. Here is a list of items to check when laying out dialog boxes:

- For most document windows that contain a single view (scrolling text or tables, for example), do not specify any space between the window edge and scrollbars (for a Carbon application) or the frame of the view (in IB).
- For dialog boxes that contain a mix of controls, set 16 pixels of vertical space between controls. Try to maintain a 24-pixel space between the left and right edge of the window and any controls. Keep 20 pixels of space between the bottom edge and any controls; this can include the shadow of any push buttons in that area. Top spacing is determined by which controls are placed closest to the top of the dialog box. For example, Figure 1-14 (page 19) uses a group box label as the topmost control, so the spacing is set to 12 pixels. In contrast, Figure 1-15 (page 20) uses a column of radio buttons and checkboxes as the topmost element, so the spacing is set to 16 pixels.
- In general, you should try for a more center-biased approach to dialog box layout, as opposed to the strongly left-biased approach of the traditional Mac OS dialog box. Most of the sample layouts in this document illustrate the center-biased approach.
- Group boxes are shown in several of the sample dialog boxes, but we recommend that, whenever possible, you use additional space between controls to create groups, in preference to group boxes. Excessive use of group boxes

creates visual clutter, as the user is distracted by too many lines and edges. Within a group box, no control or label should be positioned within 16 pixels of the box's top, bottom, left, or right borders.

- Groups of controls should be separated by 16 pixels of vertical spacing and subgroups of controls within groups should be separated by 12 pixels.
- Vertical spacing between controls is determined by the tallest control in the row.
- Horizontal spacing between controls is 12 pixels. A control associated with a label (such as a text input field or a pop-up button) should be spaced 8 pixels from its label.
- Check boxes and radio buttons are spaced 20 pixels baseline-to-baseline. Using this spacing automatically provides 7 pixels of separation between each control. For an illustration of this approach, see "Radio Button and Checkbox Sizing" (page 9) and Figure 1-15 (page 20).
- Text associated with controls like pop-up buttons, checkbox groups, and radio button groups should be spaced 8 pixels from the control.
- Bevel button spacing is based on the button's use. Toolbar bevel buttons, as used in the Finder or MailViewer applications, for example, are spaced 8 pixels apart. This spacing would not be appropriate for a set of smaller buttons, however. We do not recommend that you overlap buttons in a palette.
- The "OK" or default button should be positioned in the lower-right corner of the window, dialog box or alert box. If you use a Cancel button, it should be positioned to the left of the default button. If you use a third, or alternate, button option ("Don't Save", for example), then it should be positioned to the left of the Cancel button, as shown in Figure 1-1 (page 3). The preferred ordering of buttons is Alternate, Cancel and Default. Push buttons should be spaced 12 pixels apart horizontally and 10 pixels apart if stacked vertically.

Spacing Summary

When to Use 8-Pixel Spacing

- Between a control and its text label
- Between a tab control and the top of the window

- Between two controls that are closely associated (such as the text input field and pop-up menu used together in Figure 1-14)
- Between a control and an icon used to identify the control's use

When to Use 12-Pixel Spacing

- Between push buttons
- Between pop-up buttons
- Between text input fields
- Between text labels for groups of controls
- Between sub-groups of controls in larger groups

When to Use 16-Pixel Spacing

- Between the inside edge of a group box and enclosed controls
- Between the top and bottom edges of a group box and enclosed controls
- Between the left and right edges of a group box and any other group box
- Between primary groups of controls
- Between the top edge of a window and its topmost controls, when appropriate to the controls

When to Use 20-Pixel Spacing

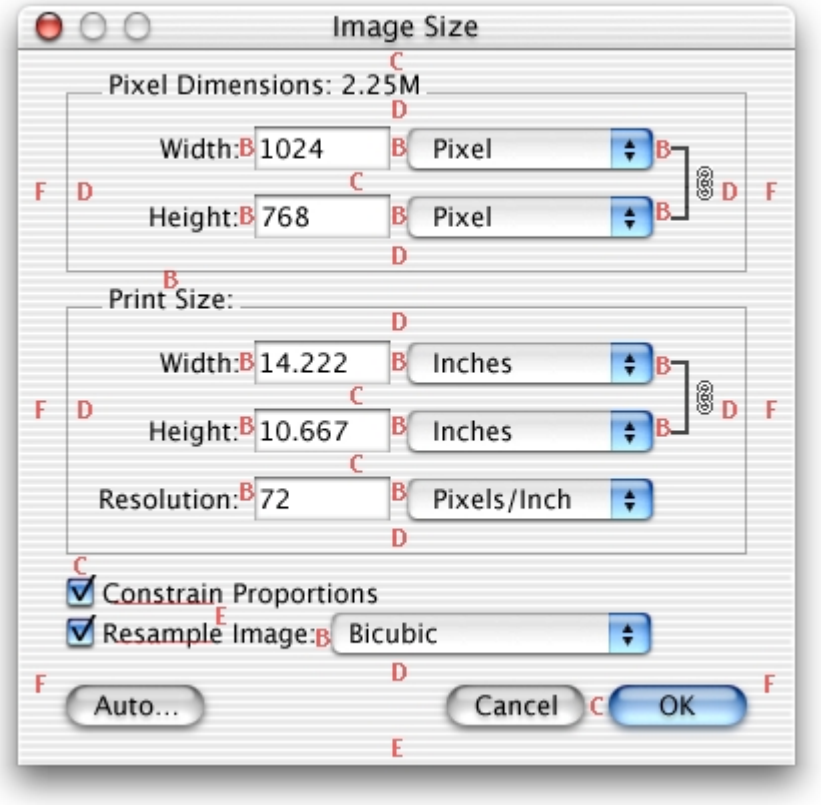
- Between the bottom edge of a window and its enclosed controls
- Between groups of controls when no group box is used or when the mixture of controls requires more space
- Between the baselines of text labels for radio buttons and check boxes

When to Use 24-Pixel Spacing

- Between the left or right edge of a window and its enclosed controls

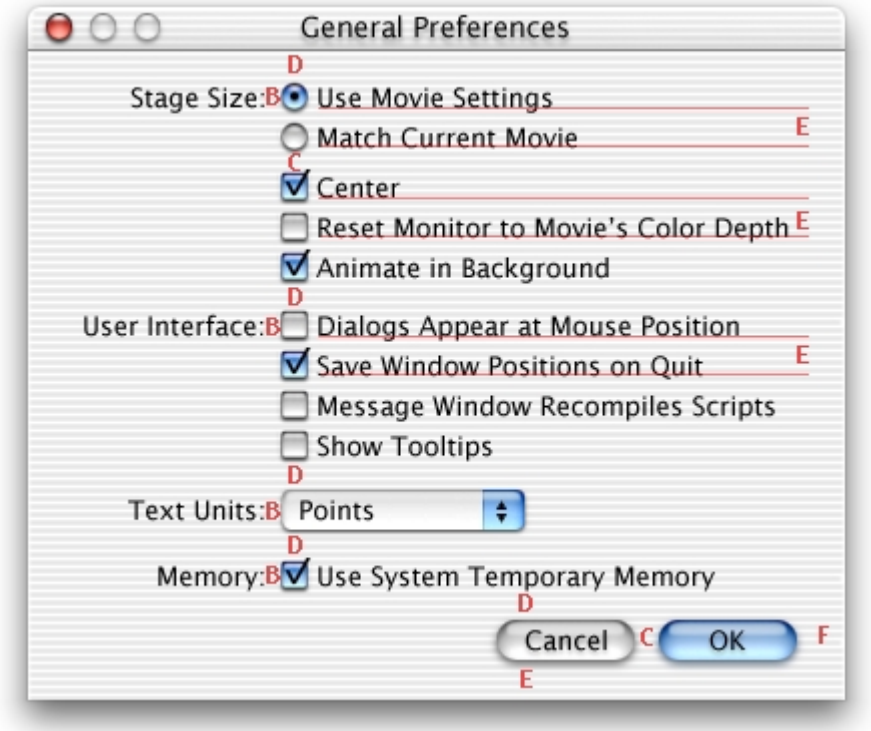
Sample Dialog Box Layouts

Figure 1-14 Sample Image Size dialog box



B = 8 D = 16 F = 24
C = 12 E = 20

Figure 1-15 Sample General Preferences dialog box



B = 8 D = 16 F = 24
C = 12 E = 20

Icon Design

Icon design in Mac OS X is significantly different from previous versions of Mac OS. Icon design in Mac OS 8 and 9 is highly abstracted and conceptualized due to graphical limitations. These limitations constrained designers to use a style that emphasized straight lines rotated in increments of 45 degrees. A familiar example of this is the standard diamond-like application icon, as shown in Figure 1-16.

Figure 1-16 Standard application icon for Mac OS 8 and 9

Building on the icon style defined in previous versions of Mac OS, Mac OS X offers icons that are stylistically vibrant and emotive. Icons can be represented in 128 x 128 pixels to allow ample room for detail. Anti-aliasing makes curves and non-rectilinear lines possible. Alpha channels and transparency allow for complex shading and dimensionality. All of these qualities pave the way for lush and vibrant imagery that allows you to create icons that communicate in ways never before possible.

Designing User Application Icons

Stylistically, the diamond-like rotation in a Mac OS X user application icon is arbitrary; it is designed to reinforce the emotional quality of an icon. The rotated object carries a direct meaning for the application and should represent the media created or viewed by the application. The perspective can be compared to objects on a desk. If you look at your keyboard as it appears when you are sitting in your chair, you obtain the perspective from which Mac OS X user application icons are designed. This perpective (that is, looking down at a layered set of elements), combined with the superior aesthetic capabilities of Mac OS X, allows the user to perceive icons as representing familiar objects in a familiar way, which helps the application become more approachable and non-threatening to novice users enabling you to design icons with the desired emotive quality. Figure 1-17 shows an example of a user application icon that clearly communicates its purpose in a way that invites the user to learn more about it.

Figure 1-17 Example user application icon for Mac OS X

Observe the use of color, curve, and shading in Figure 1-17 (page 22). Alpha channels allow for complex, soft shadows that emphasize the effect of layered elements as well as perspective. Alpha channels can also be used for transparency, so you can provide the effect of translucency, as seen in Figure 1-18 (page 23). 32-bit color provides a vibrant color palette.

Figure 1-18 Sample application icon for Mac OS X

Aside from the overall shape and formal elements that you can use to design an application icon, you should also provide a supportive "tool" whenever possible to communicate the type of task that an application allows the user to accomplish. The magnifying glass in Figure 1-18 is a good example of a tool element. The tool should closely relate to the base object that it rests upon and should be rotated in an opposite direction. Ideally, the application icon should tell a story of how the application can be used. More examples of the supporting tool concept are shown in Figure 1-19 (page 23).

Figure 1-19 Examples of icons with supporting "tools"

Designing Utility Icons

In contrast to the "object on a desktop" approach discussed in "Designing User Application Icons" (page 22), utility application icons are designed to reinforce the typically "serious" nature of utility applications. This change in style shifts the appropriate visual perspective to that of an object on a shelf at eye level, directly in front of the user. Unlike richly colored user application icons, utility icons are generally greyscale, with color applied sparingly. Figure 1-20 (page 24) shows an example of a utility icon.

Figure 1-20 Example of a utility icon

Since utility applications are normally focused on a narrow set of tasks, it's best to keep the number of elements to a minimum. The supporting tool should be an integrated element, as opposed to a layered object.

Designing Icon Genres

A new concept in Mac OS X is the notion of icon "genres." Classifying applications by role (such as user applications, utilities, and administrator's tools, for example) provides for simple, easy division of icons into useful categories. Each genre is identified by a distinctive icon style. The value of this division becomes more apparent when you look at icons in juxtaposition. Figure 1-21 (page 25) shows two rows of icons, with application icons in the top row and utility icons in the bottow row.

Figure 1-21 Examples of different icon genres

The Dock, for example, can contain icons of many different genres, so it is crucial that you provide clear visual indication of an icon's genre. This indication allows the user to easily classify and organize each icon on the Dock.

Menu Layout

This section describes changes and additions to Mac OS X menus.

Notable Changes From Mac OS 9

The Apple menu has been replaced with an Application menu containing general application items. Note that the familiar “Quit” and “Preferences” menu items have been moved to the Application menu. By introducing the Application menu in the left-most position in the menu bar (in left-to-right script systems), Mac OS X creates a logical progression of menu functions:

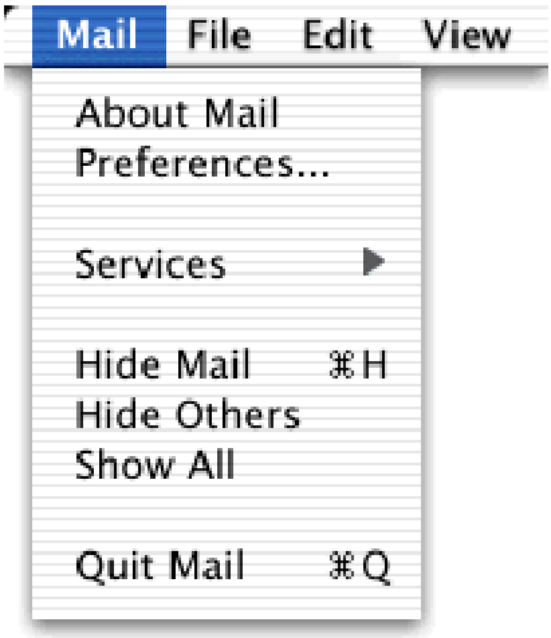
- The first menu (the Application menu) provides items that pertain to the entire application, such as Hide and Quit.
- The second menu (the File menu) provides items for managing an individual document file, such as Open and Close Window.
- The third menu (the Edit menu) and all subsequent menus provide items for managing document content, such as Cut and Paste.

An optional Window menu has also been introduced, intended to aid in managing multiple documents open within an application. Note that in Developer Preview 4, this menu is available to Cocoa applications only.

The Application Menu

The Application menu, which is new in Mac OS X, replaces the Apple menu from previous releases of Mac OS. This menu is populated with menu items that are general to the application; that is; items that are not specific to a document or other window.

Figure 1-22 Example of Application menu



The title of the Application menu is the name of the application. In Figure 1-22, for example, the menu title is "Mail." The limited space available for this name requires that you provide a short application name (16 characters or less) as part of the application package. You should ensure that your title does not exceed 128 pixels in width when drawn in the standard menu font. Note that the "About" item, the "Hide" command and the "Quit" command should all use this short application name, as well.

The first item below the Application menu title is the familiar "About" item. It provides access to the application's About dialog box and should be named appropriately. For example, the Mail application's About item is "About Mail". Any menu items that provide access to the application's preference dialogs should be installed below the "About" item. You should include any other application-specific items here, as well. If you specify a single "About" item and a single preference item, do not use a menu separator between them. If you include a number of application-specific items, you can use a menu separator to group them.

The Services menu, currently only available for Cocoa applications, is the next item in the Application menu. This item is preceded by a menu separator. The "Hide" command, found in the process menu on Mac OS 9, is the next item in Mac OS X's Application menu. This command is preceded by a menu separator and followed by the "Hide Others" and "Show All" commands. The last menu item of the application menu is the Quit command. This menu item has been moved from its previous location, the File menu. Quit should be preceded by a menu separator.

The File and Edit Menus

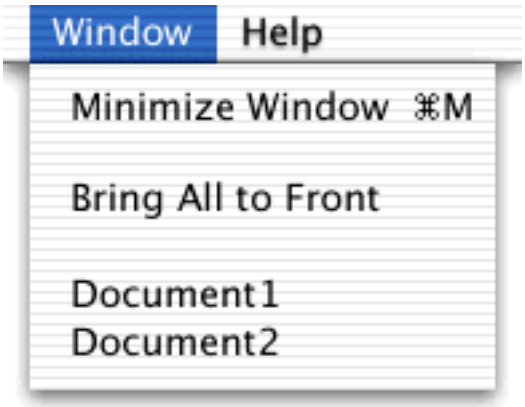
These menus should remain similar to their Mac OS 9 counterparts, with the exceptions of the relocated Preferences and Quit items described in "The Application Menu" (page 26). If an application is not document-centered, you can rename the File menu to something more appropriate. For example, the System Preferences application's first menu is called Panels. If there are no relevant commands in the File menu, you can eliminate it altogether. For example, if your single-window application automatically quits when its window is closed, you do not have to provide a Close Window command; the Quit command in the Application menu is sufficient.

The Window and Help Menus

Note that in Developer Preview 4, the Window menu is only available to Cocoa applications. The Window menu (which is optional) provides a listing of open document windows, plus provides equivalent menu commands and keyboard shortcuts to some of the window titlebar controls. If you choose to implement the Window menu, you can add commands to show and hide auxiliary windows such as palettes and modeless dialogs.

If an application uses a single window, it is not necessary to create a Window menu simply to provide the Minimize Window command; you can simply provide a minimizing control in the window's title bar and support for the appropriate keyboard shortcut. For more information, see "Mac OS X-Reserved Keyboard Shortcuts" (page 30).

Figure 1-23 Example of a Window menu



The ordering of items in the Window menu is: Minimize Window, <separator>, Bring All to Front. The Cocoa AppKit then appends a separator, followed by the window list. The "Close Window" menu item has been relocated to the File menu; it was previously located in the Window menu in the menu layout guidelines for Mac OS X Server.

Mac OS X appends a Help menu after the Window menu. The Help menu contains one default item, the Help Center. As with Mac OS 8 and 9, you can add more items to the Help menu if you wish.

Using Ellipses in Menus

An ellipsis (…) after a menu item or button label indicates to the user that additional information is required to complete a command. You should use an ellipsis in the following cases:

- An action that requires further user input to complete, or presents an alert allowing the user to cancel the action. Examples include: "Open...", "Page Setup..." and "Print...". Typically these windows are modal and are dismissed after completing the action.
- An action that opens a non-modal dialog box that requires additional user input to complete task. Examples include: "Find...", "Go To...", and "Spelling...".

- An action that opens a settings window. The main function of settings windows is to allow the user to change some aspect of the application, not the document content. The main function of these dialogs is not to provide information (see opening informational windows below). Examples: “Set Title...”, “Preferences...”, and “Options...”.

You should not use an ellipsis in the following cases:

- An action that requires no further user input to complete and does not present an alert. Often the item to be acted upon is already selected. Examples: “New”, “Cut”, “Bold”, “Print One”, and “Quit”.
- An action that opens an informational, accessory or tool window. These windows can be implemented as either utility windows (i.e. Colors), or non-modal windows (i.e. Help, About, Fonts, and Inspectors). These windows provide tools that help create or manage the content in the main window and are frequently left open to assist the task of the main window.

Mac OS X-Reserved Keyboard Shortcuts

Mac OS X reserves two key combinations as new shortcuts to menu commands that affect all applications.

- <Command-H> is reserved as a shortcut to the “Hide <appname>” menu command that appears in the Application menu of all applications.
- <Command-M> is reserved as a shortcut to the “Minimize Window” menu command that appears in the Window menu. It applies to any active window that can be minimized.