CHAPTER 1

Building a User Interface

Your application's user interface is agrueably the most important part any application. The old saying "You don't get a second chance to make a first impression" couldn't be more true when it comes to your applications user interface. If the interface is intuitive and sloppy, the user will react the same way they might react to someone who has poor communication skills and cares little for their appearance. Using your application will be frustrating at best and at worst, the user will give up and go looking for another solution to their problem that your application could have solved. This leaves you with whatever goals you had for your application unfulfilled.

Fortunately, REALbasic makes building your application's user interface so fast and easy that you can spend the time you need to get the interface just right. REALbasic's built-in Interface Assistant™ actually helps you build a proper, clean interface.

In this chapter you will learn just about everything you need to know about creating all of the elements that make up your applications' user interface. You will learn how to build windows and menus and some guidelines to following when creating your interface.

Contents

Working with Windows

Typically, most of an application's user interface will be in windows. This of course is highly application-specific. Some applications have no windows at all; relying completely on menus to provide the user interface. REALbasic makes it easy to create new windows of just about any type, remove them, and fill them with interface controls like pushbuttons and checkboxes.

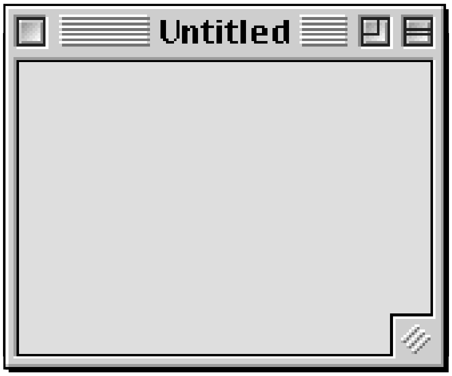
Window Types

REALbasic supports 7 different types of windows. The type you choose for a particular window depends mostly on how the window will be used.

Document

Document windows are the most common type of window used. They are most often used when the window will stay open until the user dismissed it by clicking it's close box (if it has one) or clicking a button programmed to close the window. The user can click on other windows to bring them to the foreground, moving the document window behind the others. Figure 1 on page 3 shows an example of a small, blank document window.

FIGURE 1. **A Document window**

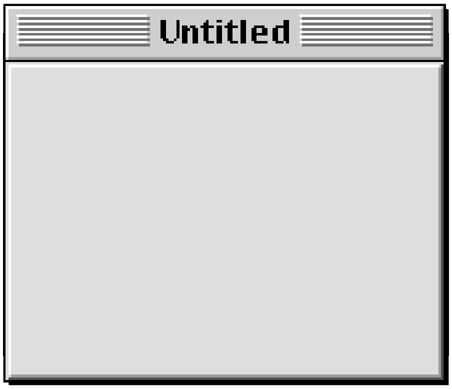


Document windows can have a close box, zoom box grow icon (making them user-resizable).

Movable Modal

This type of window stays in front of all other windows the application has open until it is closed. Use a Movable Modal window when you need to briefly communicate with the user without the user having access to the rest of the application. Because the window is movable, the user will be able to drag the window to another location on the screen in case they need to see information in other windows in order to finish what they are doing in the Movable Modal window. Figure 2 on page 4 shows an example of a blank Movable Modal window.

FIGURE 2. **A Movable Modal window**



Movable Modal windows cannot have a close box so you will need to include a button that the user can click to dismiss the window unless the window will dismiss itself after the application finishes a particular task. They also cannot have zoom box or grow icon so they are not re-sizable by the user. This means you will have to consider the amount of available screen space the user will have in determining the size you will make a Movable Modal window.

Note: There is one exception to the rule regarding Movable Modal windows being in front of all other windows. If a Movable Modal window or one of it's controls executes code that opens a Floating window, the Floating window will be in front of the Movable Modal window. However, it is poor interface design for a Movable Modal window to open another window because Movable Modal windows are mostly used in situations where the interaction with the user will be brief.

Modal Dialog

These windows are very similar to Movable Modal windows. The only differences are that unlike Movable Modal windows, Modal Dialog windows have no titlebar so they cannot be moved. The Page Setup dialog box is an example of a Modal Dialog window.

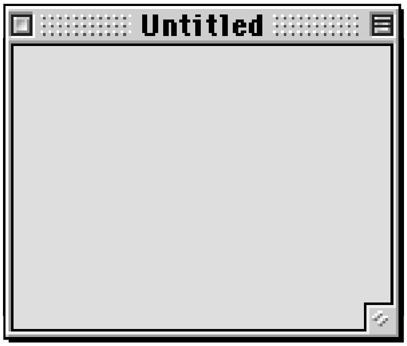
FIGURE 3. A Modal Dialog window

Note: Because Modal Dialog windows and Movable Modal windows are both modal, the same exception applies regarding floating windows opening in front of Modal windows. See the note for Movable Modal windows on page 4.

Floating

Like Movable Modal and Modal Dialog windows, a Floating window (also known as a Windoid) stays in front of all other windows. The difference is that the user can still click on other windows to access them. If you have more than one Floating window open, clicking on another Floating window will bring that window to the front but all open Floating windows will be in front of all non-floating windows. Because they are always in front of other types of windows, their size should be kept to a minimum or they will quickly get in the user's way. This type of window is most commonly used to provide tools the use will frequently access.

FIGURE 4. **A Floating window**

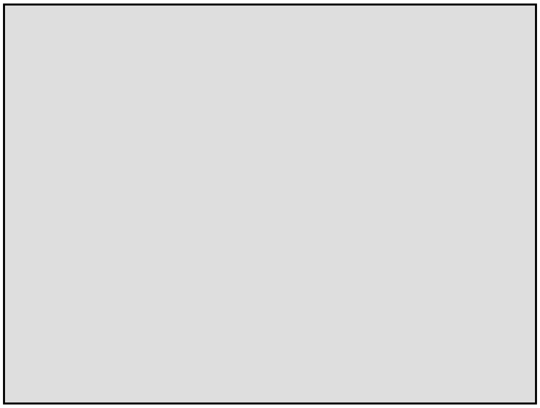


Like Document windows, Floating windows can have a close box and can be user-resizable. However, they cannot have a zoom box.

Plain Box

These windows are effectively Modal Dialog windows. The only real difference is in their appear as you can see in Figure 5 on page 6. Plain Box windows are commonly used for About Box windows and for applications that need to hide the desktop.

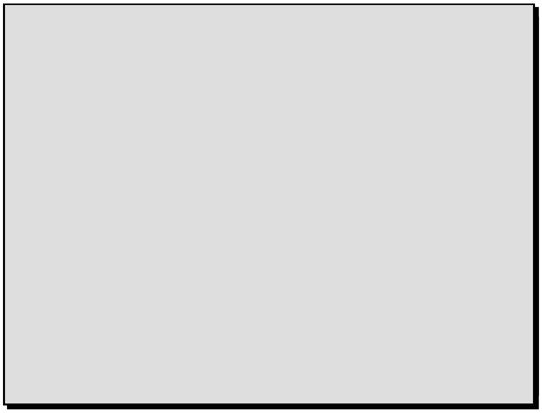
FIGURE 5. **A Plain Box**



Shadowed Box

Like Plain Box windows, Shadowed Box windows are effectively Modal Dialog windows. The only difference is their appearance as you can see in Figure 6 on page 7. Shadowed Box windows are commonly used for About Box windows.

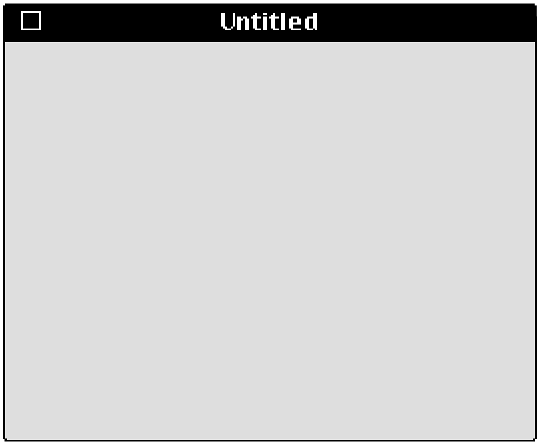
FIGURE 6. A Shadowed Box



Rounded

Rounded windows act like Document windows. The only differences are appearance (as you can see in Figure 7 on page 8) and the fact that Rounded windows cannot have a zoom box or be resizable. They are not commonly used anymore and there is really no reason to use them instead of a Document window.

FIGURE 7. **A Rounded window**



Creating Windows

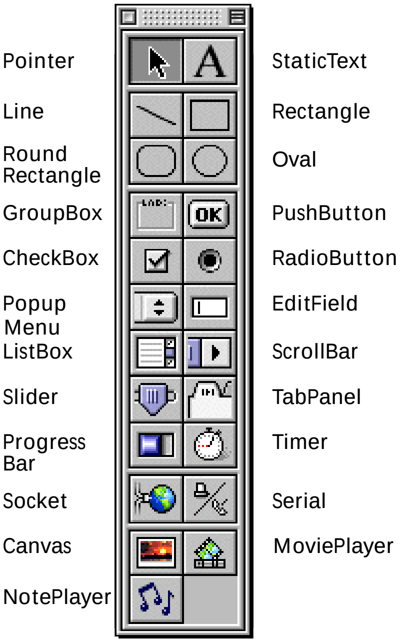
When you create a new project, REALbasic adds a window named "Window1" to your project automatically. To add additional windows, choose File ➞ New Window. The windows you create act as templates. When your application opens one of these windows, it's really opening a copy of the window. This means that your application can open several copies of the same window at the same time. It's important to understand this when creating your user interface because there is no need to go to the extra trouble of duplicating a window in the Design environment if your application needs to open two of them at the same time.

Removing Windows

To remove a window from your project, simply click on it once in the Project window to select it and press the Delete key. REALbasic supports the Undo feature better than most applications. If you delete a window by mistake, choose Edit ➞ Undo (⌘-Z).

Interacting with the User Through Controls

Users provide information to your application through user interface controls. REALbasic provides a tremendous amount of flexibility in this area. Not only are there many built-in controls, but you can even create your own controls (you will learn more about this later).



Adding, Changing and Removing Controls

REALbasic makes adding, changing and removing controls easy.

Adding Controls

To add a control to a window in your project, do this:

1. Bring the window to the front. If it's not open, double-click on it in the Project window to open it.
2. Drag the desired control from the Tools window and drop it on the window.

Selecting Controls

Controls can be selected with one of two ways: using the mouse button or with the Tab key. If you click on a control, it will be selected. When a control is selected, REALbasic draws a border around the control using your highlight color selected in your Appearance control panel.

You can also move through the controls in a window by pressing the Tab key. Each time you press the Tab key, REALbasic will move from one control to another. This is also the order the user will move through the controls when using the Tab key. For more information, see “Changing The Tab Order” on page 12. Holding down the Shift key while pressing the Tab key will select controls in reverse Tab order. If only one control is selected, REALbasic will draw resize squares at each corner of the control. You can select several controls by holding down the Shift key as you click on controls.

Changing a Control’s Position

A controls position can be changed by dragging the control using the mouse, by using the arrow keys (to move it one pixel at a time in any direction) and by changing the Position properties in the Properties window.

Changing a Control’s Properties with the Properties window

Some changes to a control must be made with the Properties window. For example, controls can be rearranged by simply dragging them from one place to another inside the window. However, most of the changes you make to controls will be made using the Properties window.

The Properties window displays the properties of the currently selected control that can be changed from the Design environment. If more than one control is selected, the Properties window displays only those properties common to all of the selected controls.

Some properties are entered by typing, while others with on/off-type values are represented by a checkbox. Some properties that require you to choose a value from a fixed list are displayed as popup menus. Color properties display the selected color. These colors can be changed by clicking on the color and using the Color Picker to choose a color or by dragging a color from the Colors window and dropping on a color property.

Removing Controls

To remove a control from a window, do this:

1. Bring the window that contains the control to the front. If it's not open, double-click on it in the Project window to open it.
2. Click on the control to select it.
3. Choose Edit ➞ Cut (⌘-X), or press the Delete key.

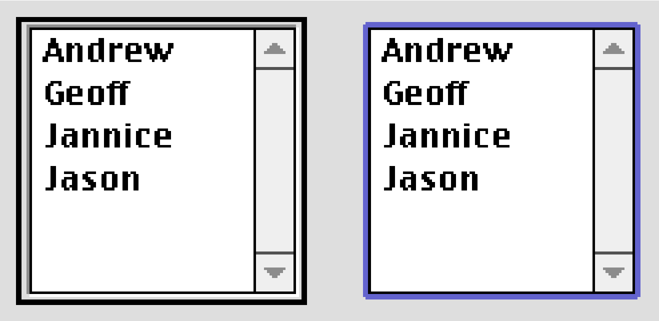
Understanding Control Layers

Each control in a window has it's own layer. This layer is like a clear piece of plastic and determines whether one control is in front of the other. The Arrange menu provides commands for moving a control forward one layer, to the front, backwards one layer and to the very back of the layers. These layers will usually only be important when controls are going to overlap. For example, when you place controls on top of a GroupBox control or a TabPanel control, the GroupBox or TabPanel must be in back of the other controls. Otherwise, the GroupBox or TabPanel will be in front of one or more of the controls, obscuring them from view. Control layers also determine the order that your application will go through the controls as the user presses the Tab key. However, you don't have to rearrange the layers of controls in order to determine their tab order. Instead, you can use the Control Order dialog to determine the tab order. See "Changing The Tab Order" on page 12 for more details.

Understanding The Focus

The focus is a visual cue that tells the user which control will receive keystrokes. Only EditFields and ListBoxes can receive the focus. EditFields display the focus by showing a blinking cursor. When a ListBox has focus, REALbasic draws a border around the ListBox. If the user is running System 7, this border will be a black rectangle. If the user is running MacOS8, the border will drawn in the Accent Color chosen in the user's Appearance control panel. When a ListBox has the focus, it automatically responds to the arrow keys. It also receives any other keys the user types. This allows you to provide *type selection* functionality where typing automatically selects the item that matches the characters being typed. An example of type selection is provided with REALbasic.

FIGURE 8. **A ListBox with the focus (System 7 and MacOS8)**



Note: A Listbox will not receive the focus if it's the only item in the window that can receive the focus.

Changing The Tab Order

You can change the order of controls that will receive the focus as the user presses the Tab key by using the Control Order dialog box. This dialog box displays a list of controls from the current window in the current tab order (from first to last).

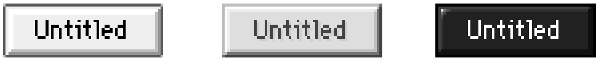
The Appearance of Controls

The part of the Macintosh Operating System (MacOS) that handles how menus, windows and controls will appear is called the Appearance Manager. If you are running MacOS8 or greater, you have probably used the Appearance Control Panel to select a highlight color and perhaps an accent color. A future release of the MacOS will add a new feature called “Themes” to the Appearance Manager. Themes will provide several “looks” that allow the user to subtly or radically change the appearance of menus, windows and controls. This in no way changes the functionality of the interface. This is simply a way to take the idea of allowing the user to customize their computing environment one step further.

REALbasic supports the Appearance Manager. This means that REALbasic itself will appear differently based on your Appearance Control Panel settings. It also means that when Themes become available in a future release of the MacOS, REALbasic’s interface will change based on the Theme the user chooses. The applications you create with REALbasic also support the Appearance Manager automatically.

If you would like to have Themes now, there is a shareware system extension that provides the equivalent of Themes. It’s called Kaleidoscope and it runs under System 7 and MacOS8. You can download it from www.download.com. If you are planning on distributing the application you build to a large number of people, it would probably be worth your time to install Kaleidoscope and check out how your interface works with Themes. This will allow you to make small changes if you need to so that when your users install a MacOS that supports Themes, your interface will look just right. Figure 9 on page 13 shows a standard pushbutton as it appears with different Kaleidoscope themes.

FIGURE 9. A standard PushButton displayed in 3 different Kaleidoscope Themes



Apple Computer updated the look for many interface elements in MacOS8. You (or your users) may have even installed a system extension called Aaron that changed the appearance of the interface in System 7 to make it look like MacOS8. Part of this change was giving the controls a more “3D” look. REALbasic, by default, will draw controls with this 3D look regardless of whether the user is running System 7, System 7 with the Aaron extension or MacOS8. If you want your interface to appear as similar as possible to the interface of other applications the user is running, you will want to uncheck the Use 3D Controls option in the Project Settings dialog box available under the Edit menu.

Button Controls for Performing Actions

There are three controls that are commonly used to perform actions when clicked: the CheckBox, the PushButton, and the RadioButton.

PushButton

When clicked, a PushButton appears to depress giving the user feedback that they have clicked it. Pushbuttons are typically used to take an immediate and obvious action when pressed, like printing a report or closing a window.

FIGURE 10. **A PushButton pressed and unpressed**

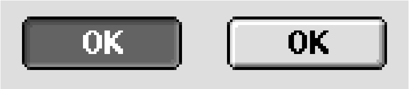


TABLE 1. **PushButton properties**

Name	Description
Super	The class of object the PushButton is based on.
Name	The internal name of the PushButton used to identify it in programming code.

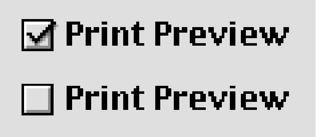
TABLE 1. **PushButton properties**

Name	Description
Index	The PushButton's position in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the PushButton.
Top	The distance (in pixels) between the top edge of the window and the top edge of the PushButton.
Width	The width (in pixels) of the PushButton.
Height	The height (in pixels) of the PushButton.
LockLeft	Keeps the distance between the left side of the window and the left side of the PushButton from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the PushButton from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the PushButton from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the PushButton from changing when the window is resized.
Visible	The PushButton will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the PushButton.
Caption	The text that appears on the PushButton.
Default	Adds the standard default ring to the PushButton and associates the Return and Enter keys with the it.
Cancel	Associates the Escape key and Command-Period key combination with the PushButton.
Enabled	The PushButton will be initially enabled.
TextFont	The font used to display the PushButton's caption.
TextSize	The font size used to display the PushButton's caption.
Bold	Adds the bold style to the PushButton's caption.
Italic	Adds the italic style to the PushButton's caption.
Underline	Adds the underline style to the PushButton's caption.

CheckBox

Checkboxes are used to let the user state a preference that has only two possible choices where one of the choices can be selected by default. Checkboxes should not cause an immediate and obvious action to occur when pressed except perhaps to enable or disable other controls.

FIGURE 11. **A CheckBox checked and unchecked**



If space permits, consider using two RadioButton controls instead of a single CheckBox control as it will make the user's choice more obvious especially to the new computer user.

TABLE 2. **CheckBox properties**

Name	Description
Super	The class of object the CheckBox is based on.
Name	The internal name of the CheckBox used to identify it in programming code.
Index	The position of the CheckBox in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the CheckBox.
Top	The distance (in pixels) between the top edge of the window and the top edge of the CheckBox.
Width	The width (in pixels) of the CheckBox.
Height	The height (in pixels) of the CheckBox.
LockLeft	Keeps the distance between the left side of the window and the left side of the CheckBox from changing when the window is resized.

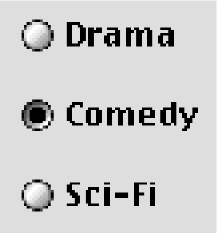
TABLE 2. **CheckBox properties**

Name	Description
LockTop	Keeps the distance between the top of the window and the top of the CheckBox from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the CheckBox from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the CheckBox from changing when the window is resized.
Visible	The CheckBox will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the CheckBox.
Caption	The text that appears on the PushButton.
Enabled	The CheckBox will be initially enabled.
TextFont	The font used to display the CheckBox caption.
TextSize	The font size used to display the CheckBox caption.
Bold	Adds the bold style to the CheckBox caption.
Italic	Adds the italic style to the CheckBox caption.
Underline	Adds the underline style to the CheckBox caption.
Value	The default value of the CheckBox.

RadioButton

RadioButtons are used to present the user with two or more choices where one of the choices can be selected by default. Selecting one RadioButton causes the RadioButton that is currently selected to become unselected. They are called RadioButtons because they act just like the buttons on old car radios where pushing one button would cause the previously pushed button to pop back out. RadioButtons should always be displayed in groups of at least two.

FIGURE 12. **A group of RadioButtons with one selected**



If you are creating a window that will have two or more independent sets of RadioButtons, you will need to use a GroupBox control to make your RadioButton groups respond independently. See “GroupBox” on page 33.

TABLE 3. **RadioButton properties**

Name	Description
Super	The class of object the RadioButton is based on.
Name	The internal name of the RadioButton used to identify it in programming code.
Index	The position of the RadioButton in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the RadioButton.
Top	The distance (in pixels) between the top edge of the window and the top edge of the RadioButton.
Width	The width (in pixels) of the RadioButton.
Height	The height (in pixels) of the RadioButton.
LockLeft	Keeps the distance between the left side of the window and the left side of the RadioButton from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the RadioButton from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the RadioButton from changing when the window is resized.

TABLE 3. **RadioButton properties**

Name	Description
LockBottom	Keeps the distance between the bottom of the window and the bottom of the RadioButton from changing when the window is resized.
Visible	The RadioButton will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the RadioButton.
Caption	The text that appears on the RadioButton.
Enabled	The RadioButton will be initially enabled.
TextFont	The font used to display the RadioButton caption.
TextSize	The font size used to display the RadioButton caption.
Bold	Adds the bold style to the RadioButton caption.
Italic	Adds the italic style to the RadioButton caption.
Underline	Adds the underline style to the RadioButton caption.
Value	The default value of the RadioButton.

Controls for Displaying and Entering Text

REALbasic provides controls that let you display text the user can't select, display text the user can select but not edit and display text the user can both select and edit.

StaticText

Used to display text that the user shouldn't be able to select. StaticText controls are most commonly used to label other controls (like PopupMenus) or provide titles for groups of controls.

FIGURE 13. **A StaticText control used to label a PopupMenu control**

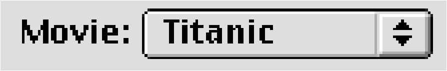


TABLE 4. **StaticText properties**

Name	**Description**
Super	The class of object the StaticText is based on.
Name	The internal name of the StaticText used to identify it in programming code.
Index	The position of the StaticText in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the StaticText.
Top	The distance (in pixels) between the top edge of the window and the top edge of the StaticText.
Width	The width (in pixels) of the StaticText.
Height	The height (in pixels) of the StaticText.
LockLeft	Keeps the distance between the left side of the window and the left side of the StaticText from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the StaticText from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the StaticText from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the StaticText from changing when the window is resized.
Visible	The StaticText will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the StaticText.
Text	The text that appears in the window.
TextAlign	The alignment of the text within it's area (left, middle, right).

TABLE 4. **StaticText properties**

Name	Description
TextFont	The font used to display the StaticText caption.
TextColor	The color of the text.
Multiline	Causes the text to start at the top of it's area rather than being centered vertically within it.
TextSize	The font size used to display the StaticText caption.
Bold	Adds the bold style to the StaticText caption.
Italic	Adds the italic style to the StaticText caption.
Underline	Adds the underline style to the StaticText caption.

EditField

EditFields can be used to allow the user to enter text or to display text that can copied to the clipboard but not changed in the EditField. They can also be configured to allow multiple lines of text, display a scrollbar if necessary, and display text in multiple fonts, styles and sizes.

FIGURE 14. **A Empty EditField**

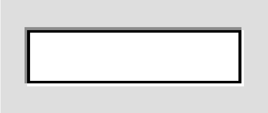
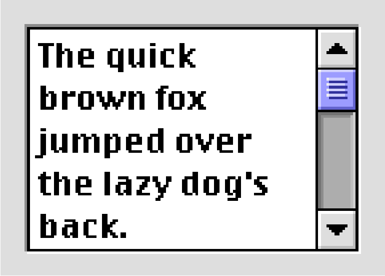
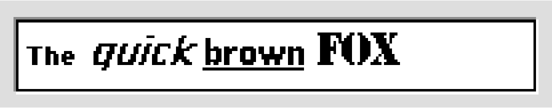


FIGURE 15. **An EditField configured for multiple lines of text**



The quick brown fox jumped over the lazy dog's back.

FIGURE 16. **An Editfield with multiple fonts, styles and sizes**



The quick brown FOX

TABLE 5. **EditField properties**

Name	**Description**
Super	The class of object the EditField is based on.
Name	The internal name of the EditField used to identify it in programming code.
Index	The position of the EditField in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the EditField.
Top	The distance (in pixels) between the top edge of the window and the top edge of the EditField.
Width	The width (in pixels) of the EditField.
Height	The height (in pixels) of the EditField.
LockLeft	Keeps the distance between the left side of the window and the left side of the EditField from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the EditField from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the EditField from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the EditField from changing when the window is resized.
Visible	The EditField will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the EditField.
Border	Draws a border around the EditField.

TABLE 5. **EditField properties**

Name	Description
Multiline	Causes the text to start at the top of it's area rather than being centered vertically within it.
ScrollBar	Displays a scrollbar if Multiline property is checked.
ReadOnly	Allows copying of text to the clipboard but no editing.
Styled	Allows EditField to contain styled (multiple fonts, styles and sizes) text.
Password	Every character entered is replaced with a bullet character. The actual characters typed are stored in the Text property.
LimitText	The maximum number of characters allowed (0=no limit).
Enabled	The EditField will be enabled when the window opens.
TextFont	The font used to display the EditField caption.
TextSize	The font size used to display the EditField caption.
Bold	Adds the bold style to the EditField caption.
Italic	Adds the italic style to the EditField caption.
Underline	Adds the underline style to the EditField caption.
Text	The default value of the EditField.

Controls for Displaying and Entering Numeric Values

REALbasic provides controls that can be used to let the user choose a numeric value from a range or to display a numeric value from a range. In some cases, these controls can also be used to control the display of another control. For example, a ScrollBar control might be used to determine which portion of a picture in a Canvas control is displayed (in other words, act as the Canvas control's scrollbar).

ScrollBar

ScrollBars can be presented vertically or horizontally. By default, they are horizontal. To make a vertical ScrollBar, simply resize the

Scrollbar object so that the height is greater than the width. While you can resize a ScrollBar to any proportions, Horizontal ScrollBars should always be 16 pixels tall and vertical ScrollBars should be 16 pixels wide.

FIGURE 17. **Horizontal and vertical ScrollBars**

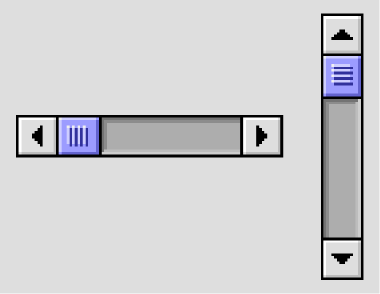


TABLE 6. **ScrollBar Properties**

Name	Description
Super	The class of object the ScrollBar is based on.
Name	The internal name of the ScrollBar used to identify it in programming code.
Index	The position of the ScrollBar in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the ScrollBar.
Top	The distance (in pixels) between the top edge of the window and the top edge of the ScrollBar.
Width	The width (in pixels) of the ScrollBar.
Height	The height (in pixels) of the ScrollBar.
LockLeft	Keeps the distance between the left side of the window and the left side of the ScrollBar from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the ScrollBar from changing when the window is resized.

TABLE 6. **ScrollBar Properties**

Name	Description
LockRight	Keeps the distance between the right side of the window and the right side of the ScrollBar from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the ScrollBar from changing when the window is resized.
Visible	The ScrollBar will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the ScrollBar.
Enabled	The ScrollBar will be initially enabled.
Minimum	The value of the Value property when the scroll indicator is all the way left (for horizontal scrollbars) or at the very top (for vertical scrollbars).
Value	The current position of the scroll indicator.
Maximum	The value the Value Property will be set to when the scroll indicator is all the way to the right (for horizontal scrollbars) or at the bottom (for vertical scrollbars).
Span	
LineStep	The amount by which the Value property will change when the user clicks on one of the ScrollBar's arrows.
PageStep	The amount by which the Value property will change when the user clicks inside the ScrollBar on either side of the scroll indicator.
LiveScroll	

Slider

This control was added in MacOS8. It is effectively identical to a ScrollBar control in functionality. However, ScrollBar controls have come to be associated with scrolling text or a picture and less with assigning numeric values. The Slider control provides an interface that is clearly for increasing or decreasing a numeric value. Like the ScrollBar, the Slider control can appear horizontally (which is the default) or vertically. You can create a vertical Slider by changing its height so that it’s greater than it’s width. Unlike the ScrollBar control, the Slider control automatically maintains the correct proportions regardless of the dimensions you give it. Because the Slider was added in MacOS8, for System 7 users the Slider control will appear as a ScrollBar.

FIGURE 18. **A horizontal and vertical Slider controls**

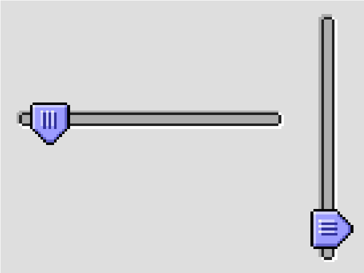


TABLE 7. **Slider properties**

Name	**Description**
Super	The class of object the Slider is based on.
Name	The internal name of the Slider used to identify it in programming code.
Index	The position of the Slider in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the Slider.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Slider.
Width	The width (in pixels) of the Slider.
Height	The height (in pixels) of the Slider.
LockLeft	Keeps the distance between the left side of the window and the left side of the Slider from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the Slider from changing when the window is resized.

TABLE 7. **Slider properties**

Name	Description
LockRight	Keeps the distance between the right side of the window and the right side of the Slider from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the Slider from changing when the window is resized.
Visible	The Slider will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the Slider.
Enabled	The Slider will be initially enabled.
Minimum	The value of the Value property when the indicator is all the way left (for horizontal Sliders) or at the very top (for vertical Sliders).
Value	The current position of the indicator.
Maximum	The value the Value Property will be set to when the indicator is all the way to the right (for horizontal Sliders) or at the bottom (for vertical Sliders).
LineStep	This property is only used when the user is running System 7 as the Slider appears as a Scrollbar. The amount by which the Value property will change when the user clicks on one of the ScrollBar's arrows.
PageStep	This property is only used when the user is running System 7 as the Slider appears as a Scrollbar. The amount by which the Value property will change when the user clicks inside the ScrollBar on either side of the scroll indicator.
LiveScroll	

ProgressBar

ProgressBars are designed for showing that some function of your application is making progress (hence the name) towards its goal or to show capacity. Unlike ScrollBars and Sliders, ProgressBars are designed for displaying a value. They cannot be used for data entry. They can also appear only in a horizontal orientation. When using a ProgressBar to show duration, the ProgressBar can be configured to show progress where the length is determinate and indeterminate. Indeterminate ProgressBars are often referred to as "Barber Poles."

FIGURE 19. **Determinate and indeterminate ProgressBars**

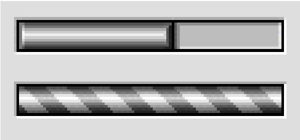


TABLE 8. **ProgressBar properties**

Name	**Description**
Super	The class of object the ProgressBar is based on.
Name	The internal name of the ProgressBar used to identify it in programming code.
Index	The position of the ProgressBar in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the ProgressBar.
Top	The distance (in pixels) between the top edge of the window and the top edge of the ProgressBar.
Width	The width (in pixels) of the ProgressBar.
Height	The height (in pixels) of the ProgressBar.
LockLeft	Keeps the distance between the left side of the window and the left side of the ProgressBar from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the ProgressBar from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the ProgressBar from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the ProgressBar from changing when the window is resized.
Visible	The ProgressBar will initially be visible when the window opens.

TABLE 8. **ProgressBar properties**

Name	Description
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the ProgressBar.
Value	The current position of the indicator.
Maximum	The value the Value Property will be set to when the indicator is all the way to the right.

Controls for Presenting the User a List of Choices

RadioButton and CheckBox controls can of course be used to provide the user with a list of choices. There are situations, however, when using these controls is either an inefficient use of space or impossible. Some of these situations are:

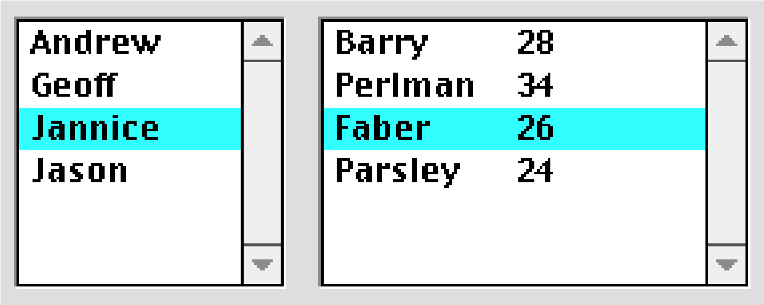
- When the number of choice items is quite long making it difficult or impossible to use RadioButton or CheckBox controls
- When the choices change dynamically based on the application's logic
- When the choice items need to display more than one column of information

If your situation doesn't match one of the above, consider using RadioButton or CheckBox controls instead. They are easier for a new computer user to handle because all of their choices will be right in front of them.

ListBox

ListBox controls display a scrolling list of values. The user can use the mouse or the arrow keys to choose an item. ListBox controls can contain one or more columns of data.

FIGURE 20. Single and multi-column ListBoxes



Andrew	
Geoff	
Jannice	
Jason	

Barry	28
Perlman	34
Faber	26
Parsley	24

TABLE 9. ListBox properties

Name	Description
Super	The class of object the ListBox is based on.
Name	The internal name of the ListBox used to identify it in programming code.
Index	The position of the ListBox in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the ListBox.
Top	The distance (in pixels) between the top edge of the window and the top edge of the ListBox.
Width	The width (in pixels) of the ListBox.
Height	The height (in pixels) of the ListBox.
LockLeft	Keeps the distance between the left side of the window and the left side of the ListBox from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the ListBox from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the ListBox from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the ListBox from changing when the window is resized.
Visible	The ListBox will initially be visible when the window opens.

TABLE 9. **ListBox properties**

Name	Description
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the ListBox.
Column-Count	The number of columns the ListBox can display.
Column-Widths	A list of comma-separated values, with each value controlling the width of the associated column. Each value can be express in pixels or as a percentage.
Enabled	The ListBox will be initially enabled.
TextFont	The font used to display the ListBox caption.
TextSize	The font size used to display the ListBox caption.
Bold	Adds the bold style to the ListBox caption.
Italic	Adds the italic style to the ListBox caption.
Underline	Adds the underline style to the ListBox caption.

PopupMenu

PopupMenu controls are useful when you have a single column of data to present and have limited space with which to work.

FIGURE 21. **A Popupmenu control**

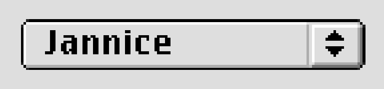


TABLE 10. **PopupMenu properties**

Name	Description
Super	The class of object the PopupMenu is based on.
Name	The internal name of the PopupMenu used to identify it in programming code.
Index	The position of the PopupMenu in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the PopupMenu.

TABLE 10. **PopupMenu properties**

Name	Description
Top	The distance (in pixels) between the top edge of the window and the top edge of the PopupMenu.
Width	The width (in pixels) of the PopupMenu.
Height	The height (in pixels) of the ListBox.
LockLeft	Keeps the distance between the left side of the window and the left side of the PopupMenu from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the PopupMenu from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the PopupMenu from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the PopupMenu from changing when the window is resized.
Visible	The PopupMenu will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the PopupMenu.
Enabled	The PopupMenu will be initially enabled.
TextFont	The font used to display the PopupMenu caption.
TextSize	The font size used to display the PopupMenu caption.
Bold	Adds the bold style to the PopupMenu caption.
Italic	Adds the italic style to the PopupMenu caption.
Underline	Adds the underline style to the PopupMenu caption.

Controls for Visually Grouping Other Controls

If a window contains groups of controls where each group of controls is serving a different purpose, it can be confusing to the user to see all of these groups simply lumped together in a window. It often makes sense (and is sometimes necessary) to visually group related controls. Fortunately, REALbasic provides two built-in controls to make grouping controls simple.

GroupBox

A GroupBox can be displayed with or without a caption. If a window has more than one group of RadioButton controls, one of the groups must be contained within a GroupBox control in order for the RadioButton groups to function independently.

FIGURE 22. **A GroupBox control with and without a caption**

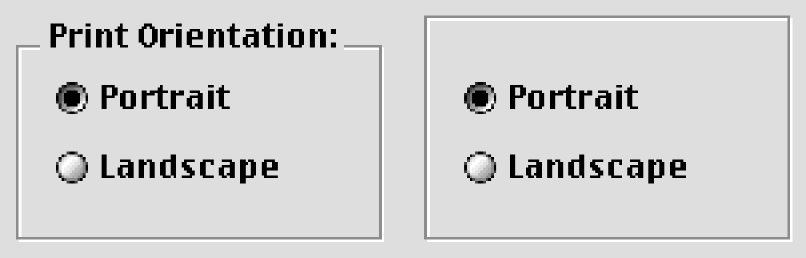


TABLE 11. **GroupBox properties**

Name	**Description**
Super	The class of object the GroupBox is based on.
Name	The internal name of the GroupBox used to identify it in programming code.
Index	The position of the GroupBox in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the GroupBox.
Top	The distance (in pixels) between the top edge of the window and the top edge of the GroupBox.
Width	The width (in pixels) of the GroupBox.
Height	The height (in pixels) of the GroupBox.
LockLeft	Keeps the distance between the left side of the window and the left side of the GroupBox from changing when the window is resized.

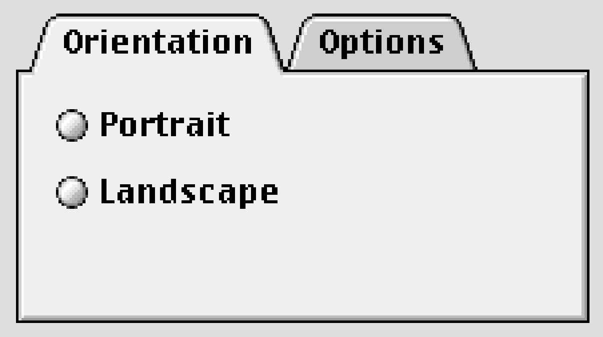
TABLE 11. **GroupBox properties**

Name	Description
LockTop	Keeps the distance between the top of the window and the top of the GroupBox from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the GroupBox from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the GroupBox from changing when the window is resized.
Visible	The GroupBox will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the GroupBox.
Caption	The text that appears on the GroupBox.
Enabled	The GroupBox will be initially enabled.
TextFont	The font used to display the GroupBox caption.
TextSize	The font size used to display the GroupBox caption.
Bold	Adds the bold style to the GroupBox caption.
Italic	Adds the italic style to the GroupBox caption.
Underline	Adds the underline style to the GroupBox caption.
Value	The default value of the GroupBox.

TabPanel

When you have several groups of controls and space is very limited, TabPanels are most appropriate. TabPanels allow several groups of controls to occupy the same space in the window. When the user clicks on tab in the TabPanel, REALbasic automatically hides the controls associated with the currently selected tab and displays those that are associated with the tab the user has clicked on.

FIGURE 23. **A two-panel TabPanel control**



Orientation
Options
○ Portrait
○ Landscape

TABLE 12. **TabPanel properties**

Name	Description
Super	The class of object the TabPanel is based on.
Name	The internal name of the TabPanel used to identify it in programming code.
Index	The position of the TabPanel in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the TabPanel.
Top	The distance (in pixels) between the top edge of the window and the top edge of the TabPanel.
Width	The width (in pixels) of the TabPanel.
Height	The height (in pixels) of the TabPanel.
LockLeft	Keeps the distance between the left side of the window and the left side of the TabPanel from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the TabPanel from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the TabPanel from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the TabPanel from changing when the window is resized.

TABLE 12. **TabPanel properties**

Name	Description
Visible	The TabPanel will initially be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the TabPanel.
Enabled	The TabPanel will be initially enabled.

Controls for Displaying Graphics and Pictures

REALbasic is very flexible when it comes to displaying graphics and pictures. You can use the built-in graphic controls, display pictures from documents or draw the graphics using REALbasic's programming language.

Line

Draws a line that can be of any length, width, color and direction. Lines are 100 pixels in length, 1 pixel in width, black and horizontal by default.

TABLE 13. **Line properties**

Name	Description
Super	The class of object the Line is based on.
Name	The internal name of the Line used to identify it in programming code.
Index	The position of the Line in a control array.
X1	The distance (on the horizontal axis) from the left side of the window to the end of the Line that is leftmost by default.
X2	The distance (on the horizontal axis) from the left side of the window to the end of the Line that is right most by default.
Y1	The distance (on the vertical axis) from the top of the window to the end of the Line that is leftmost by default.

TABLE 13. **Line properties**

Name	Description
Y2	The distance (on the vertical axis) from the top of the window to the end of the Line that is right most by default.
Visible	The Line will be visible when the window opens.
Border-Width	The width (in pixels) of the Line.
LineColor	The color of the Line.

Rectangle

Draws a rectangle that can be of any length, width, border color, and fill color. Rectangles are 100 pixels in length and width, 1 pixel in width, have black borders and a white center by default. Because you can control the color of the left and top borders independently from the right and bottom borders, you can easily create rectangles that appear to be sunken or raised.

FIGURE 24. **A Rectangle with default, sunken and raised appearances**

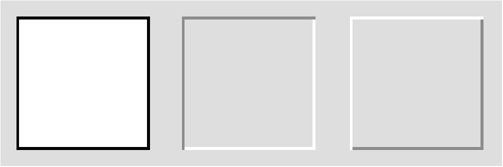


TABLE 14. **Rectangle properties**

Name	Description
Super	The class of object the Rectangle is based on.
Name	The internal name of the Rectangle used to identify it in programming code.
Index	The position of the Rectangle in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the Rectangle.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Rectangle.

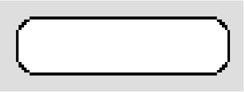
TABLE 14. **Rectangle properties**

Name	Description
Width	The width (in pixels) of the Rectangle.
Height	The height (in pixels) of the Rectangle.
LockLeft	Keeps the distance between the left side of the window and the left side of the Rectangle from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the Rectangle from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the Rectangle from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the Rectangle from changing when the window is resized.
Visible	The Rectangle will be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the Rectangle.
FillColor	The color that will fill the interior of the Rectangle.
BorderWidth	The width (in pixels) of the sides of the Rectangle.
TopLeftColor	The color of the lines that make up the top and left sides of the Rectangle.
BottomRightColor	The color of the lines that make up the right and bottom sides of the Rectangle.

RoundRectangle

RoundRectangles are similar to regular Rectangle controls. The differences are that you don’t have the independent color control for the border (because it’s one continuous line) but you can control the width and height of the arcs that make up the round corners.

FIGURE 25. A RoundRectangle control



Oval

Draws an oval with a single pixel, black border and filled with white. All of these properties can be modified. The “ovalness” of the Oval is controlled by it’s height and width. For example, an Oval with the same width and height would be a perfect circle.

FIGURE 26. An Oval control

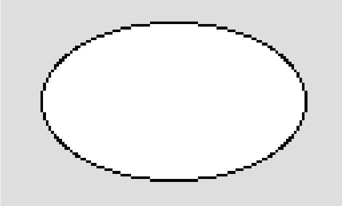


TABLE 15. Oval properties

Name	Description
Super	The class of object the Oval is based on.
Name	The internal name of the Oval used to identify it in programming code.
Index	The position of the Oval in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the Oval.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Oval.
Width	The width (in pixels) of the Oval.
Height	The height (in pixels) of the Oval.

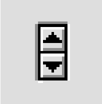
TABLE 15. **Oval properties**

Name	Description
LockLeft	Keeps the distance between the left side of the window and the left side of the Oval from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the Oval from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the Oval from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the Oval from changing when the window is resized.
Visible	The Oval will be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the Oval.
FillColor	The color that will fill the interior of the Oval.
Border-Width	The width (in pixels) of the sides of the Oval.
BorderColor	The color of the Oval's border.
OvalWidth	The width of the arcs that make up the corners.
OvalHeight	The height of the arcs that make up the corners.

Canvas

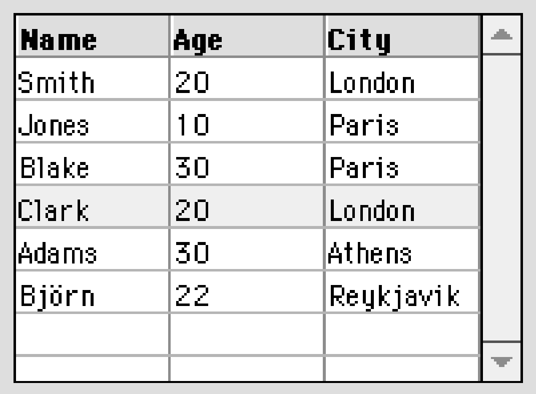
A Canvas control can be used to display a picture from a file or a picture drawn using REALbasic's programming language. If your application requires a type of control that is not built-in to REALbasic, you can use a Canvas control along with the REALbasic programming language drawing commands to create the controls you need.

FIGURE 27. **An Canvas control used to create a “Little Arrows” control**



Canvas controls can be used to create extremely sophisticated controls. In Figure 28 on page 41, a Canvas control is used to provide a table of data with rows that can be selected and columns that can be sorted by clicking on the column title.

FIGURE 28. **A sophisticated control created using a Canvas control**



Name	Age	City
Smith	20	London
Jones	10	Paris
Blake	30	Paris
Clark	20	London
Adams	30	Athens
Björn	22	Reykjavik

The “Little Arrow” and Table controls above were created by Björn Eiríksson.

TABLE 16. **Canvas properties**

Name	Description
Super	The class of object the Canvas is based on.
Name	The internal name of the Canvas used to identify it in programming code.
Index	The position of the Canvas in a control array.

TABLE 16. **Canvas properties**

Name	Description
Left	The distance (in pixels) between the left edge of the window and the left edge of the Canvas.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Canvas.
Width	The width (in pixels) of the Canvas.
Height	The height (in pixels) of the Canvas.
LockLeft	Keeps the distance between the left side of the window and the left side of the Canvas from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the Canvas from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the Canvas from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the Canvas from changing when the window is resized.
Visible	The Canvas will be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the Canvas.
Backdrop	A picture from the Project window that will be displayed inside the Canvas control.
Enabled	The Enabled will be initially enabled.

Controls for Playing Movies and Music

If the user has QuickTime™ installed, your application can play QuickTime™ movies and use the QuickTime™ Musical Instruments to play music.

MoviePlayer

The MoviePlayer control displays the standard QuickTime™ movie controller. From the Design environment, you can select the QuickTime™ movie that should be associated with a particular MoviePlayer control. You can also determine if the controller is displayed, a badge (a small icon that when clicked, reveals the controller) is displayed, or no controls are displayed at all.

TABLE 17. **MoviePlayer properties**

Name	Description
Super	The class of object the MoviePlayer is based on.
Name	The internal name of the MoviePlayer used to identify it in programming code.
Index	The position of the MoviePlayer in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the MoviePlayer.
Top	The distance (in pixels) between the top edge of the window and the top edge of the MoviePlayer.
Width	The width (in pixels) of the MoviePlayer.
Height	The height (in pixels) of the MoviePlayer.
LockLeft	Keeps the distance between the left side of the window and the left side of the MoviePlayer from changing when the window is resized.
LockTop	Keeps the distance between the top of the window and the top of the MoviePlayer from changing when the window is resized.
LockRight	Keeps the distance between the right side of the window and the right side of the MoviePlayer from changing when the window is resized.
LockBottom	Keeps the distance between the bottom of the window and the bottom of the MoviePlayer from changing when the window is resized.
Visible	The MoviePlayer will be visible when the window opens.
Balloon Help	The text that will appear if the user has Balloon Help on and moves the pointer over the MoviePlayer.

TABLE 17. **MoviePlayer properties**

Name	Description
AutoResize	Resizes the MoviePlayer at runtime to the size of the QuickTime movie.
Border	Draws a border around the MoviePlayer
Speaker	Adds the volume slider to the MoviePlayer.
HasStep	Adds the previous and next frame buttons to the MoviePlayer.
Movie	The movie to be played in the MoviePlayer.
Controller	Determines how the controller will appear at the bottom of the MoviePlayer (none, badge, or controller).
Looping	Plays the movie continuously once it has started.
Palindrome	Plays the movie backwards once it reaches it's end.

NotePlayer

While the NotePlayer control does display an icon when placed in a window in the Design environment, it has no interface. It is designed only for playing musical notes using the QuickTime™ musical instruments. See "NotePlayer Control" on page 58 of the Language Reference for more details.

TABLE 18. **NotePlayer properties**

Name	Description
Super	The class of object the NotePlayer is based on.
Name	The internal name of the NotePlayer used to identify it in programming code.
Index	The position of the NotePlayer in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the NotePlayer.
Top	The distance (in pixels) between the top edge of the window and the top edge of the NotePlayer.
Instrument	The number that represents the QuickTime Musical Instrument to be used to play notes. See "NotePlayer Control" on page 59 for a list of instruments.

Controls for Handling Communications

REALbasic provides controls that allow your application to communicate through the serial port (for communicating via a modem or through a serial cable to another device) and over a network to other computers using the Internet’s communication protocol called “TCP/IP.”

Serial

While the Serial control does display an icon when placed in a window in the Design environment, it has no interface. It is designed only for executing code to communicate via the serial port. See “Serial Control” on page 85 of the Language Reference for more details.

TABLE 19. **Serial control properties**

Name	Description
Super	The class of object the Serial control is based on.
Name	The internal name of the Serial control used to identify it in programming code.
Index	The position of the Serial control in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the Serial control.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Serial control.
Port	Determines which port (serial or printer) port will be used to read and write data.
Baud	The speed at which data will be read or written through the chosen port.
Bits	Determines the number of data bits used during communications.
Parity	Determines the type of parity (no parity, odd parity, even parity).
Stop	Determines the number of stop bits used during communications

TABLE 19. **Serial control properties**

Name	Description
XON	Enables XON flow control.
CTS	Enables CTS flow control.
DTR	Enables DTR flow control.

Socket

While the Socket control does display an icon when placed in a window in the Design environment, it has no interface. It is designed only for executing code to communicate with other computers on the Intranet or Internet using TCP/IP. See "Socket Control" on page 88 of the Language Reference for more details.

TABLE 20. **Socket control properties**

Name	Description
Super	The class of object the Socket control is based on.
Name	The internal name of the Socket control used to identify it in programming code.
Index	The position of the Socket control in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the Socket control.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Socket control.
Address	The IP address to send data to.
Port	The TCP/IP port to transmit/receive data on.

The Timer Control

The Timer control is designed to execute some code once or repeatedly after a period of time has passed. While the Timer control does display an icon when placed in a window in the

Design environment, it has no interface. See “Timer Control” on page 103 of the Language Reference for more details.

TABLE 21. Timer control properties

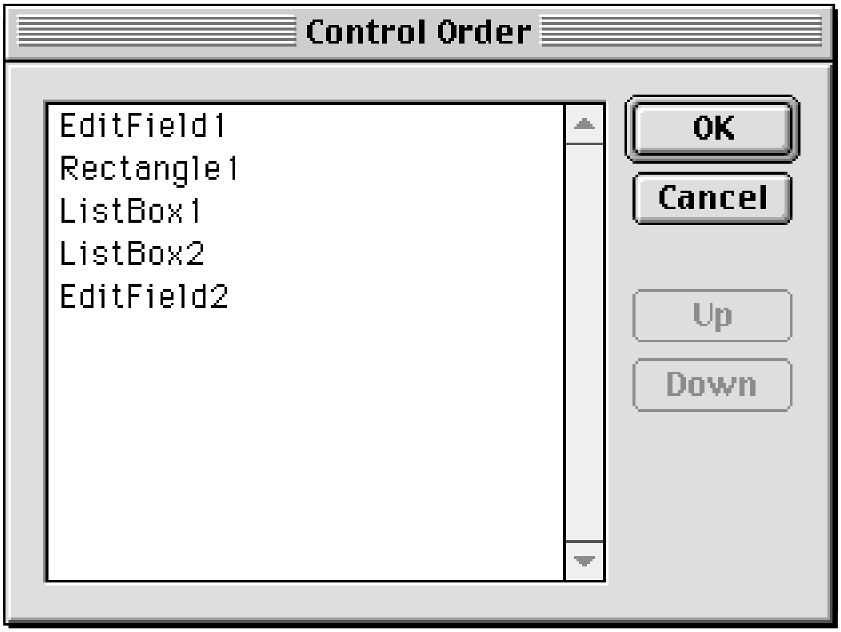
Name	Description
Super	The class of object the Timer control is based on.
Name	The internal name of the Timer control used to identify it in programming code.
Index	The position of the Timer control in a control array.
Left	The distance (in pixels) between the left edge of the window and the left edge of the Timer control.
Top	The distance (in pixels) between the top edge of the window and the top edge of the Timer control.
Mode	Determines the number of times the Timer will execute (off, single, multiple).
Period	The time in milliseconds between executions.

Changing The Tab (Control) Order

The order in which the user move through controls that receive the focus (EditFields and ListBoxes) when they press the Tab key, is called the Control Order (also know as the Tab Order). The control order is actually controlled by the control layers. When a window opens, REALbasic places the focus in the control that is farthest back that can also receive the focus. You could change the control order by using the Arrange menu to move controls through the control layers.

Instead, the Control Order dialog box makes the job much easier.

FIGURE 29. **The Control Order dialog box**



EditField1
Rectangle1
ListBox1
ListBox2
EditField2

To change the tab order, do this:

1. Choose Arrange ➞ Control Order.
2. Select the control in the list, whose tab order you wish to change.
3. Use the Up button to move the control up one position in the tab order or the Down key to move the control down one position in the tab order.

Aligning Controls with Other Controls

REALbasic’s Interface Assistant makes it easy to align a particular control with another control. Simply drag the control until it is close to being aligned with the other control. When you get close to aligning the two controls, REALbasic will snap the control you are dragging into place and display a dotted line so you can tell the controls are aligned.

Note: If the Interface Assistant is getting in your way, you can turn it off temporarily by holding down the Command key while dragging a control.

If you need to align several controls, do this:

1. Click on the control whose position is already correct to select it.
2. Choose Arrange → Move to Back to insure that the selected control will remain in place while the other controls move to align with it.
3. While holding down the Shift key, select each of the controls you wish to align together.
4. Depending on which edges you wish to align, choose Arrange → Align Objects then choose Align Left Edges, Align Right Edges, Align Top Edges or Align Bottom Edges from the Align Objects submenu.

Spacing Controls Evenly

REALbasic provides an easy way to reposition controls to evenly distribute empty space between them.

To evenly distribute the space between controls, do this:

1. Click on a control to select it.
2. Hold down the Shift key and select at least two other controls.
3. To distribute the horizontal space, choose Arrange → Align Objects then choose Space Horizontally from the Align Objects submenu.
4. To distribute the vertical space, choose Arrange → Align Objects then choose Space Vertically from the Align Objects submenu.

Adding Menus

REALbasic has a built-in Menu editor that makes adding menus and menu items to your project easy. The menus displayed in the Menu Editor will be displayed when you choose Debug → Run (⌘-R) in the Design environment or when in a stand-alone version of your application.

Adding Menus

REALbasic adds File and Edit menus to your project automatically. Every application should have at least a File menu with a Quit menu item. You can remove the Edit menu if your application has no controls that could be edited by the Edit menu items.

To add a menu to your project, do this:

1. Bring the Project window to the front by clicking on it. If it’s obscured by other windows, Choose Window → Project (⌘-0).
2. Double-click on the Menu object to open the Menu Editor. The Menu Editor window appears.
3. Click on the dotted rectangle in the Menu Editor’s menu bar to select it.
4. In the properties window, enter the Name of the menu and the Text that will appear in the menu bar.

TABLE 22. **Menu properties**

Name	Description
Name	The internal name of the Menu used to identify it in programming code.
Super	The class of object the Menu control is based on.
Text	The text that will appear in the Menu bar.

Adding a Help Menu

Most Macintosh applications have a Help menu that is the rightmost menu in the application. At a minimum, this menu has an About Balloon Help menu item and a Show Balloons menu item. The Help menu may also contain menu items that give the user access to Apple Guide files or an application specific help system. You can add a Help (complete with About Balloon Help and Show Balloons menu items) menu to your project.

To add a Help menu, do this:

1. Add a menu to the end of your menu bar.
2. Set the Text property of the menu to **Help**.

Any menu items you add to the Help menu will be displayed after the About Balloon Help and Show Balloons menu items.

Note: If you are planning on including Apple Guide files with your application, there is a handy application for generating Guide files. It's called Guide Composer and you can download a demo version from www.downloads.com.

Adding Menu Items

The Menu Editor makes it easy to add menu items to your menus. You can assign keyboard shortcuts to menu items but remember that the Macintosh looks for a shortcut starting from the left most menu. That means that if you assign the same keyboard shortcut to two different menu items, one of them won't work. There are also several specific keyboard shortcuts that should be reserved for specific functions. According to Apple's Macintosh Human Interface Guidelines, these are:

TABLE 23. **Reserved keyboard shortcuts**

Menu	Keys	Command
File	⌘-N	New
File	⌘-O	Open…
File	⌘-W	Close
File	⌘-S	Save
File	⌘-P	Print…
File	⌘-Q	Quit
Edit	⌘-Z	Undo
Edit	⌘-X	Cut
Edit	⌘-C	Copy

TABLE 23. **Reserved keyboard shortcuts**

Menu	Keys	Command
Edit	⌘-V	Paste
Edit	⌘-A	Select All
Edit	⌘-period	Terminate an operation

To add a menu item to a menu, do this:

1. In the Menu Editor, select the menu you wish to add a menu item to by clicking on it.
2. Click on the dotted rectangle at the bottom of the menu to select it.
3. In the Properties window, enter the Name and Text for the menu item.

Note: While the Menu Editor will allow you to use lowercase characters as keyboard shortcuts, only uppercase characters should be used.

Adding a Submenu

Submenus are menu items that when selected, display an additional menu to their right. The menu item itself is not selectable. It acts only as a title for the submenu.

To add a submenu to an existing menu item, do this:

1. Click on the menu item in the Menu Editor to select it.
2. In the Properties window, place a checkmark in the Submenu property. A new submenu item appears in the Menu Editor.
3. In the Menu Editor, click on the dotted rectangle that appears in the new submenu item just to the right of the menu item you selected.
4. In the Properties window, enter the Name and Text for the submenu Item.

Submenus can give the user fast access to a group of menu items. However, they can be difficult to navigate for the new computer user. They also hide menu items from view. If the user scans through the menu items looking for a particular menu item, they may not look at the submenus. Consider the audience for your application before using submenus. If many of your users will be new computer users, consider displaying a dialog box to choose the functions you would have put in a submenu.

Submenu items themselves can be submenus. Seriously consider your audience when choosing to have multiple level submenus. Many of your users may find navigating multiple level submenus difficult.

Moving Menu Items

A menu item can be moved to a new position by dragging the menu item. You can only move a menu item to another position on the same menu. If you need to move the menu item to another menu, you will have to delete it, then re-create it on the other menu.

To move a menu item, do this:

1. Click on the menu item you want to move to select it.
2. Drag the menu item towards the position on the menu where you want it. A bold line appears above the menu item.
3. When the bold line is in the position where you want to move the menu item to, release the mouse button.

Removing Menu Items

To remove a menu item from a menu, do this:

1. In the Menu Editor, click on the menu item to select it.
2. Press the Delete key or choose Edit → Clear.

Apple's Macintosh User Interface Guidelines

The quality of your application's interface will determine how successful your user will be in using it. It's absolutely critical that your user find the interface intuitive. Studies have shown that if a user can't accomplish something within the first 15 minutes of using an application, they will give up in frustration. Beyond simply being intuitive, the more polished an application's interface is, the more professional it will appear to the user. Remember that without realizing it, the user, will be comparing your application's interface to all of the other applications they have used.

REALbasic's Interface Assistant™ helps you create a nice interface by making it easy to align controls with other controls. But there is more to a professional, polished interface than simply aligning controls. We all think we know how to create a nice interface because we have used lots of applications. But using an interface is a lot different from designing one. If you haven't done so already, read Apple's Macintosh Human Interface Guidelines. This comprehensive guide will teach you what you need to know to give your application a professional interface. You will also learn the reasons behind the implementation of many of the features of the Macintosh user interface. Apple's Macintosh Human Interface Guidelines is part of the Inside Macintosh series published by Addison-Wesley and can purchased through most bookstores. You can also download it for free through Apple's Developer World web site at www.devworld.apple.com.