Beginning REALbasic

From Novice to Professional

Jerry Lee Ford, Jr.

Apress®

Beginning REALbasic: From Novice to Professional

ISBN-13 (pbk): 978-1-59059-634-0

ISBN-10 (pbk): 1-59059-634-X

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Chris Mills
Technical Reviewer: Allan Kent
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade
Project Manager: Richard Dal Porto
Copy Edit Manager: Nicole LeClerc
Copy Editor: Marcia Baker
Assistant Production Director: Kari Brooks-Copony
Production Editor: Lori Bring
Compositor: Pat Christenson
Proofreader: Linda Seifert
Indexer: Broccoli Information Management
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available on the accompanying CD. Any corrections to code and errata are posted for download/viewing at `http://www.apress.com`. Just search for the book by title, author, or ISBN, in the search bar on our homepage.

CHAPTER 4

Working with REALbasic Menus

In Chapter 3, you learned the fundamentals of creating graphical user interfaces for your REALbasic applications. In this chapter, you learn how to finish your application's user interface by adding professional-looking menus. This includes the development of menus, submenus, and menu items, and the configuration of these resources using features such as shortcuts. On top of all this, you learn how to enable and disable menu items, and to associate program statements with each menu item. Specifically, you learn how to

- Customize a menu system for your application windows
- Add shortcuts and accelerator keys to your application menus
- Control access to application menus and menu items
- Add additional menu bars to your applications

Working with REALbasic Menu Bars

Today, most users expect their desktop applications to come equipped with a menu system that provides access to application commands. A *menu system* provides you with a means for organizing commands that make your REALbasic applications work. A menu system is also a great space saver, taking up minimal space while also enabling you to remove PushButtons and other controls you would otherwise have to add to your applications.

If you want your applications to be well-received by the people who will use them, you must develop menu systems for your REALbasic applications. Fortunately, REALbasic makes this easy to do.

Note In the context of this book, the term "menu system" refers to the collection of menus, submenus, and menu items for any given window in a REALbasic application. Different windows can have different menu systems, each of which is customized to meet a specific set of needs.

REALbasic provides you with a built-in Menu Editor, which provides all the tools you need to add menus, menu items, and submenus to your application's menu system. REALbasic implements menus through MenuBar items managed in the Project Editor and configured using REALbasic's built-in Menu Editor.

The usefulness of a well-designed menu system is especially apparent on Windows and Linux applications, where each menu is displayed directly on application windows, as Figure 4-1 and Figure 4-2 show.

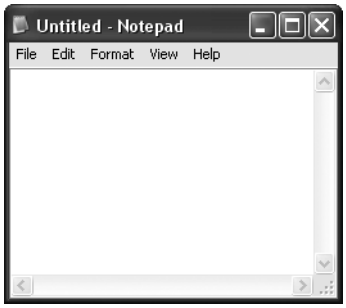

Figure 4-1. *All the functionality provided by the Windows Notepad applications is made available through its menu system.*

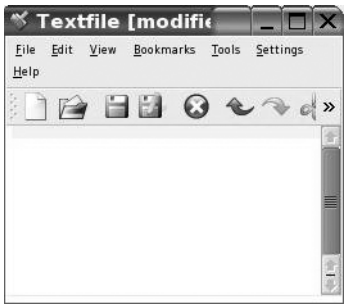

Figure 4-2. *Programs, such as the Textfile application, found on SuSe Linux, provide access to key application commands via their menu systems.*

On Macintosh, menus are displayed at the top on the screen, as opposed to directly on application windows, as Figure 4-3 shows.

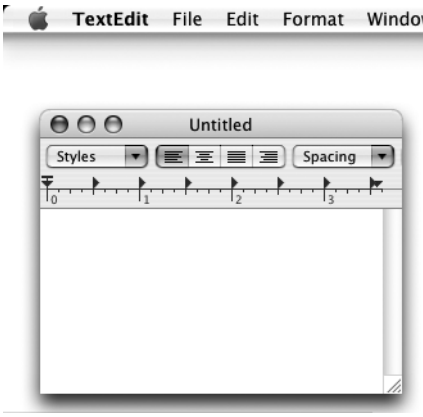

Figure 4-3. *Macintosh applications display their menu systems at the top of the desktop display area, instead of directly on top of application windows.*

Examining Menu Components

A typical application's menu system is made up of a number of different parts, including the high-level menu headings you see displayed on menu bars. In addition, under each menu, you can find different menu elements, including submenus and menu items. REALbasic provides you with all the tools you need to create professional-looking menus. This includes providing you with the capability to define shortcuts and to visually organize menu contents using separator bars. The following list provides a complete overview of the options REALbasic provides for you when developing a menu system for your applications:

- **Menus**. High-level headers displayed on the menu bar (File, Edit, Help, and so forth).
- **Menu Items**. Text items located under menus that represent commands users can select.
- **Submenus**. Collections of menu items grouped together and accessed through a parent menu (for example, a submenu).
- **Shortcuts**. Keyboard characters, or character sequences, that provide the capability to access menu items directly from the keyboard.
- **Accelerator Keys**. Keyboard keys that activate a menu or menu item when pressed in conjunction with the Alt key.
- **Separator Bars**. Horizontal lines you can use to visually group and separate submenus and menu items into logical groups.

Figure 4-4 provides a demonstration of how each of these menu components can be used in building an application's menu system.

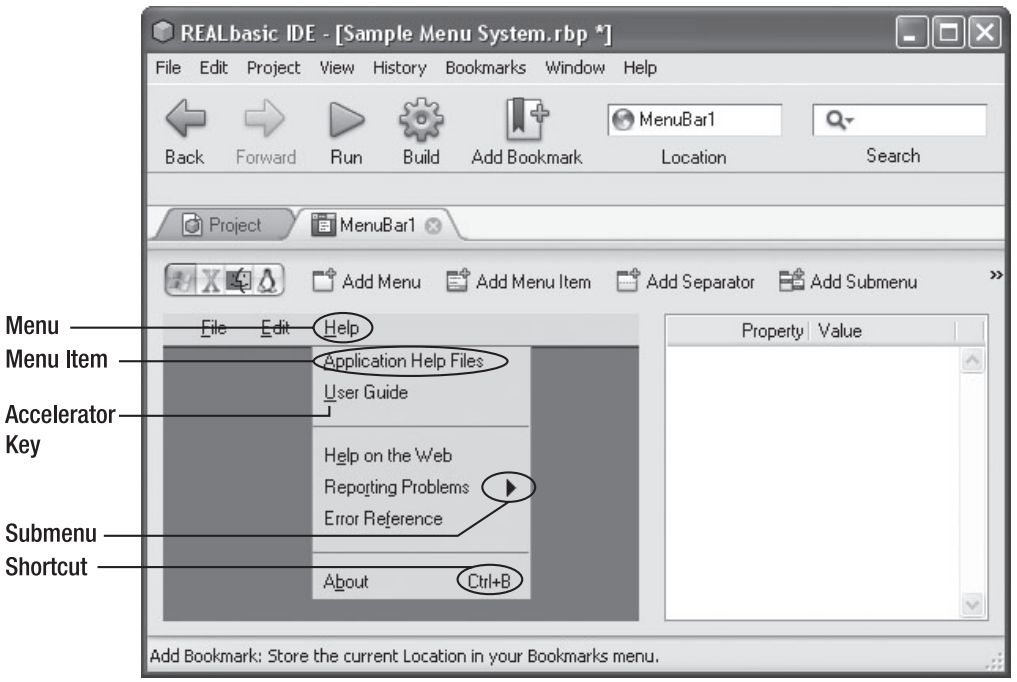

Figure 4-4. *An example of a Help menu as viewed using REALbasic's built-in menu preview option*

The Default MenuBar

By default, REALbasic provides every new desktop application with a new menu bar, as Figure 4-5 shows. By default, this menu bar is named MenuBar1.

By default, this menu bar is assigned to the application's default window (Window1). In addition, you have the option of assigning this menu to any other windows you add to your application, making it a shared menu bar.

■ **Note** Programmers with a Visual Basic background immediately notice that REALbasic approaches menu design a little differently than they are used to. Instead of adding toolbars to windows using the Visual Basic MenuStrip control, REALbasic menu bars are added to REALbasic projects as items managed on the Project Editor. In addition, REALbasic makes it easy for windows to share access to the same menu bar. Visual Basic programmers will be happy to find out, however, that the process of adding menus, menu items, and submenus is nearly the same in REALbasic as it is in Visual Basic.

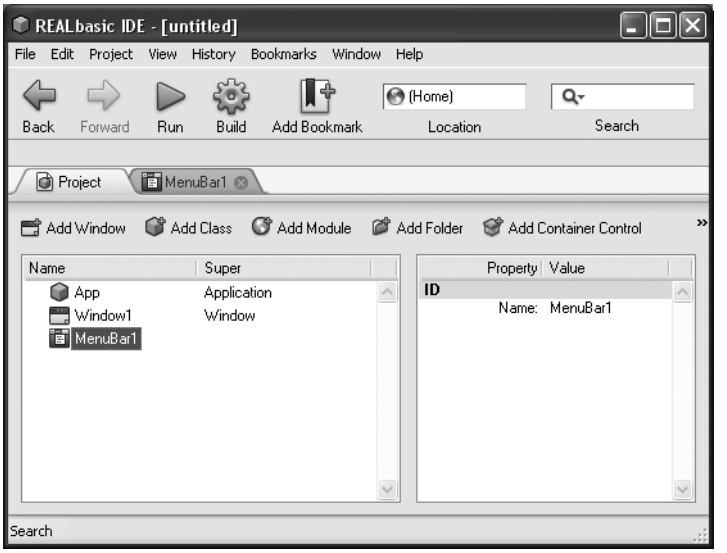

Figure 4-5. *Every new REALbasic dekstop application starts out with a default menu bar.*

By default, the default menu bar and every menu bar you add to a REALbasic application contains a File menu and an Edit menu. Table 4-1 outlines the menu items REALbasic automatically adds to these menus.

Table 4-1. *REALbasic Default Menus and Menu Items*

Menu	Command	Description
File	Exit or Quit*	Close the application
Edit	Undo	Undo a previous operation or command
	Cut	Remove selected data and place on clipboard
	Copy	Copy selected data and place on clipboard
	Paste	Copy data from clipboard to insertion point
	Delete	Delete selected data
	Select All	Select all data in the file

** Use of the Exit or Quit command depends on which operating system an application is running on (for example, Exit on Windows and Linux, and Quit on Macintosh).*

On Mac OS X, REALbasic also automatically adds the Apple and Applications menus. You learn more about these menus and how to configure them in the section "Customizing the Apple and Macintosh Menus."

■ **Tip** If your application does not use the Edit menu, you can delete it by selecting Edit menu and clicking Delete.

Adding a New Menu Bar and Assigning It to a Window

While you can use the default menu bar on windows throughout your application, often is it necessary to create and assign different menu bars to different windows. This enables you to customize a different menu bar to meet the requirements for each window. The following procedure outlines the steps involved in adding a new menu bar to a REALbasic application and assigning it to a window.

1. Open the Project Editor and click the Add Menu Bar button or click Project ➤ Add ➤ Menu Bar. REALbasic responds by adding a new menu bar to your application with a default name of MenuBar2, as Figure 4-6 shows.

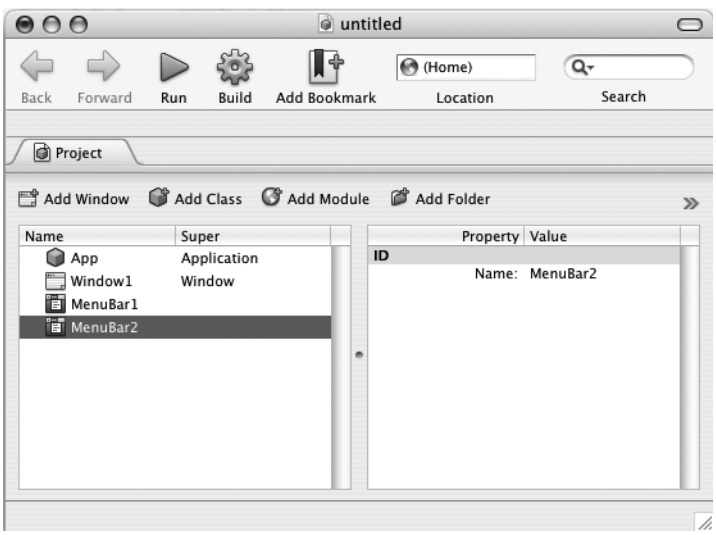

Figure 4-6. *Adding a new menu bar to a REALbasic application*

2. Select the window to which you want to add the new menu bar and modify its MenuBar property by selecting the new menu bar from the property's drop-down list, as Figure 4-7 shows.

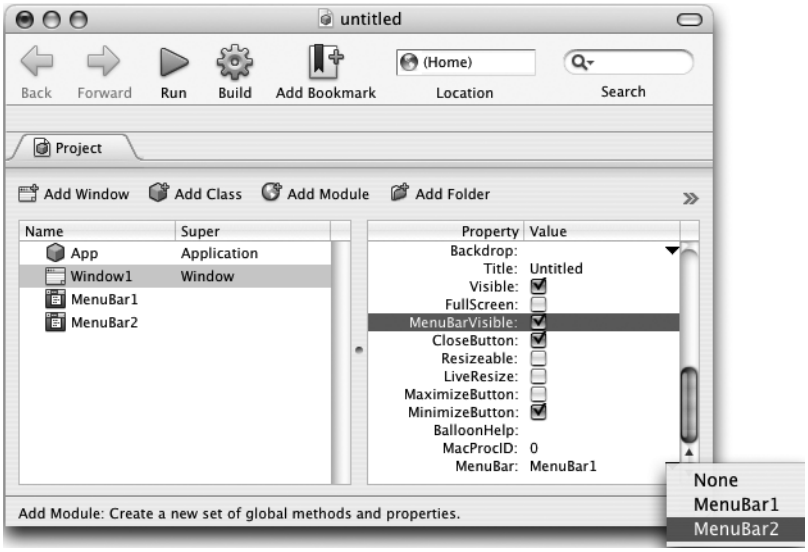

Figure 4-7. *Assigning the new menu bar to a window*

Adding Menus, Submenus, and Menu Items

One of the strengths of REALbasic is the ease with which it assists you in creating menu systems for your application windows. Using the REALbasic Menu Editor, you can create a professional menu system for any application in a matter of minutes.

Adding a New Menu

The first step in customizing a menu bar is usually to add additional menus to it. Menus do not do anything within your applications, other than provide access to menu items and submenus.

The menu items provide access to application commands. The following procedure outlines the steps involved in adding a new menu to a menu system.

1. Double-click the new menu bar item in the Project Editor that you want to modify. REALbasic responds by opening the menu bar to the REALbasic Menu Editor, as Figure 4-8 shows.

Figure 4-8. *A new, unmodified menu bar as shown on Linux*

2. Click the Add Menu button located on the Menu Editor toolbar or click Project ➤ Add ➤ Menu. REALbasic responds by adding a new menu, as Figure 4-9 shows.

Figure 4-9. *A newly added menu, as shown on Linux*

■ **Note** By default, REALbasic adds a new menu just to the right of the currently selected menu. If you do not select a menu prior to adding a new one, REALbasic adds the new menu to the end of the menu (on the far right-hand side).

---

3. If necessary, click the new menu and drag, and then drop it to the desired position on the menu bar.

4. Assign a name to the new menu by selecting it, and then modifying the value of the Name property in the Property pane. The name you assign to the new menu will be the name you use when you programmatically refer to the menu from within your program code.

5. Assign the text to be displayed on the menu by modifying the value of the Text property in the Property pane. Figure 4-10 shows how the new menu will look once its properties have been modified.

Figure 4-10. *Viewing the modified menu, as shown on Linux*

Repeat this procedure as many times as necessary to add additional menus to the menu bar. Once you finish adding menus, you are ready to customize them by adding menu items and submenus.

■ **Tip** It's important for you to be consistent when assigning text to menus, submenus, and menu items. Always make sure you capitalize the first letter of each word that makes up your menu, submenu, and menu item text properties.

Adding Menu Items

To make your applications useful, you need to provide users with access to application commands. One way to do this is by adding PushButtons and other types of controls to your application windows. If your application provides access to many commands, you may find not enough room is available on your application window to comfortably display a control representing each available command.

Too many controls make your applications seem cluttered and difficult to work with. Instead, you will probably want to organize and group related commands together, and display them as menu items located under the menus on your application's menu bars. The following procedure outlines the steps involved in adding new menu items to your menus.

1. Double-click the menu bar in the Project Editor whose menus you want to modify. REALbasic responds by opening the menu bar in the REALbasic Menu Editor. Select the menu to be modified.

2. Click the Add Menu Item button located on the Menu Editor toolbar or click Project ➤ Add ➤ Menu Item. REALbasic responds by adding a new menu item to the selected menu, as Figure 4-11 shows.

Figure 4-11. *A newly added menu item, as shown on Linux*

3. If necessary, click the new menu item and drag, and then drop it to the desired position on the menu.

4. Assign a name to the new menu item by selecting it, and then modifying the value of its Name property in the Property pane. The name you assign to the new menu item is the name you use when you programmatically refer to the menu from within your program code.

5. Assign the text to be displayed on the menu item by modifying the value of the Text property in the Property pane. Figure 4-12 shows how the new menu will look once its properties have been modified.

Figure 4-12. *Viewing the modified menu item, as shown on Linux*

6. If desired, modify the Bold, Italic, or Underline properties for the menu items by enabling the corresponding properties in the Property pane.

Repeat this procedure as many times as necessary to add an additional menu item to the selected menu. Once you finish adding menu items, you are ready to set them up to execute program code, as explained in the section "Using Menu Items to Trigger Command and Code Execution."

■ **Note** REALbasic also provides you with the capability to populate a menu with a list of menu items using menu item arrays. Information on how to use arrays is available in Chapter 5.

Creating a Submenu

If you find your menus are becoming too crowded with menu items, you may want to improve how things are organized by grouping together related menu items and making them accessible through a submenu. A *submenu* is a menu item that displays a list of menu items when it's clicked.

The following procedure outlines the steps involved in adding a new submenu to a menu by converting an existing menu item to a submenu.

1. Double-click the menu bar item in the Project Editor where you want to define the submenu. REALbasic responds by opening the menu bar in the REALbasic Menu Editor.
2. Select the menu item to be turned into a submenu.
3. Enable the menu item's Submenu property by selecting it in the Property pane. REALbasic responds by displaying an indicator to the right of the menu item to identify it as a submenu, as Figure 4-13 shows.

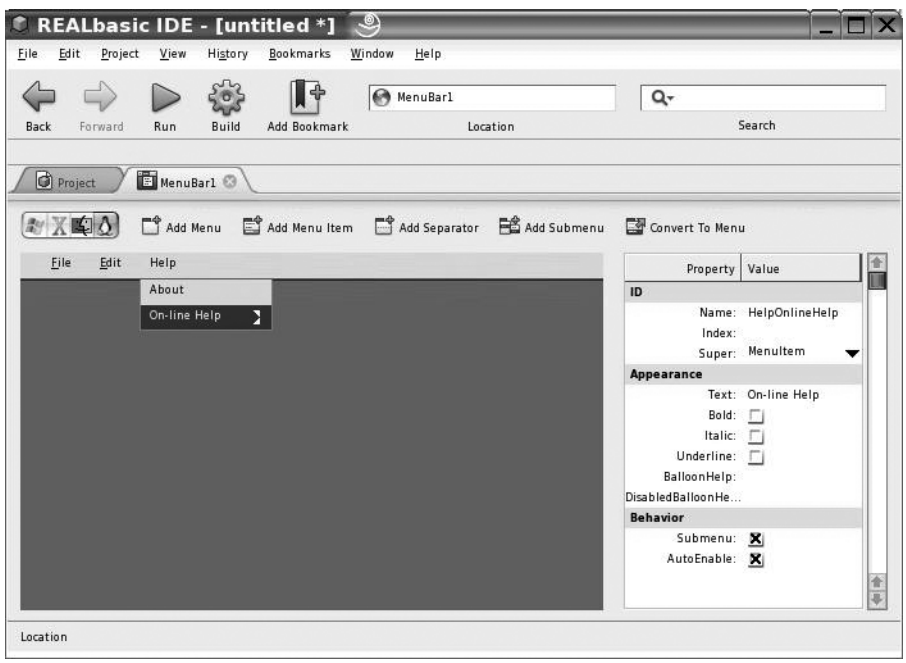

Figure 4-13. *Submenus provide the capability to reduce the size of menus by organizing and temporarily hiding the display of menu items.*

USE SUBMENUS PRUDENTLY

Submenus are a useful organizational tool, which you can use to significantly improve the overall organization of your applications. In fact, you can even add submenus to your submenus to further organize how your menu items are organized and presented. However, submenus also have a downside, especially for inexperienced users. While intermediate to advanced users should not have problems working with them, less-experienced users may not realize that additional functionality has been tucked away in submenus. Depending on the target audience for your applications, you may either want to reorganize menu items by creating additional menus under which to store them or you may decide to create a menu item that displays a new window in place of a submenu. You could then customize the new window by adding whatever controls are necessary to enable the user to select the appropriate command.

Once you create a submenu, you can add menu items to it by following the steps outlined in the next procedure.

1. Select the submenu to which you want to add menu items.
2. Click the Add Menu Item button or select Project ➤ Add ➤ Menu Item. REALbasic will respond by adding a new menu item under the selected submenu.
3. Give the new menu item a name by selecting it and then modifying the value of its Name property. The name assigned will be the name you have to use when you programmatically refer to the menu from within your program code.
4. Assign the text you want displayed on the menu item by changing the value assigned to the Text property.
5. Repeat steps 2 through 4 to add as many additional menu items as necessary to the submenu.

■ **Note** You can also add a new submenu directly to a menu by clicking the Add Submenu button located on the Menu Editor toolbar.

Previewing Your REALbasic Menus

When you design and create a menu system for REALbasic applications, which you plan on compiling for multiple operating systems (OSs), visualizing exactly how the menu system you develop will look when executed on other OSs can be difficult. Obviously, you can stop midway through the development process and compile a temporary copy of your new application to test it on target OSs to see how your menu system will look.

Fortunately, REALbasic negates this process by providing you with the capability to get a sneak peek at how your menu system will look. This functionality is provided by four buttons located on the far left-hand side of the Menu Editor toolbar, as shown in Figure 4-14.

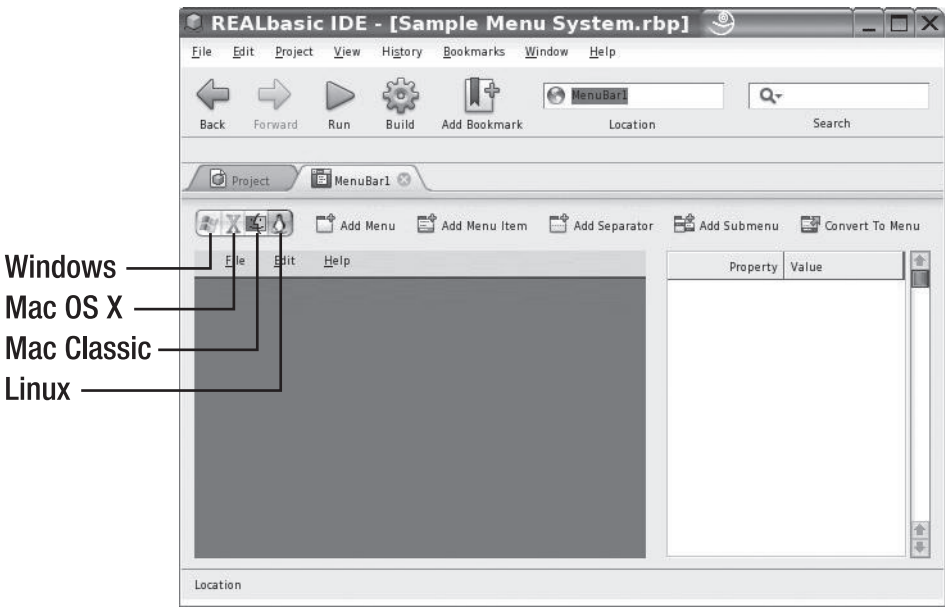

Figure 4-14. *You can preview the appearance of your REALbasic menu system for any supported OS from within the Menu Editor.*

By clicking one of these buttons, you can see how your menu system will look on different OSs. For example, Figure 4-15 shows a preview of how a menu system will look on Windows, even though the application is currently being developed on Linux.

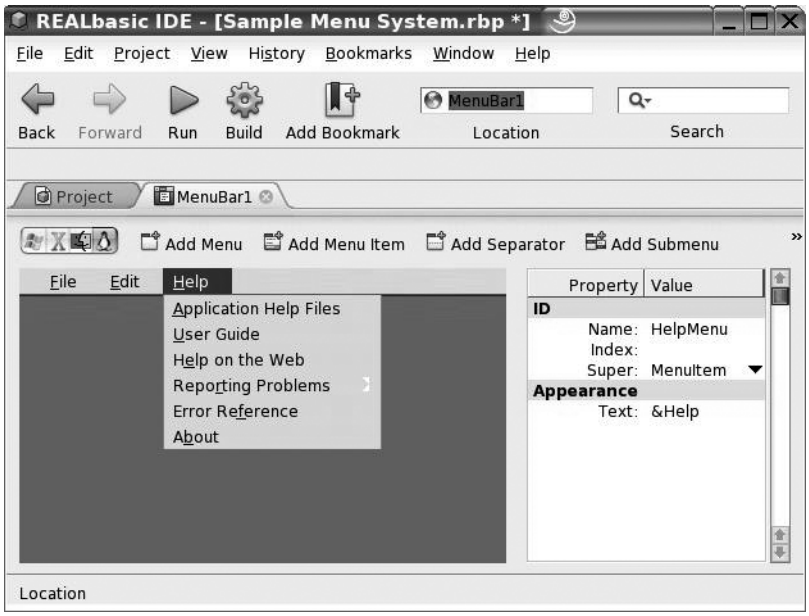

Figure 4-15. *Previewing the Windows version of an application's menu system*

Enhancing Menu Systems

Building effective application menu systems is more than simply creating menus, menu items, and submenus. To meet user expectations and to provide a fully functional menu system, you also need to think about other menu system features. For example, as you add more and more functionality to your application's menus, they can grow considerably. You can add separator bars in between groups of related menu items to visually group them and make your menu systems more intuitive.

You might also want to consider providing your users with the capability to access the menu system or to execute menu item functionality directly from the keyword, instead of limiting menu access strictly to the mouse. Options for providing menu-item access to the keyboard include the following:

- **Shortcuts**. Provide users with the capability to access a menu item's functionality directly from the keyboard by pressing keystrokes associated with each menu item.
- **Accelerator Keys**. Provides Windows and Linux users with the capability to access menus, menu items, and submenus directly from the keyboard by pressing the Alt key and a designated accelerator key.
- **Keyboard Equivalents**. Provides Windows XP users with the capability to execute menu-item functionality directly from the keyboard by pressing a predefined combination of keys.

Using Separator Bars to Organize Menu Items

To make your application's menu system easier and more intuitive to work with, make sure to group menu items and submenus together in a logical manner. For example, on the File menu of most Windows and Linux applications, the customary presentation is to have menu items related to creating, opening, or closing documents first, followed by menu items for saving or printing. The last menu item should be an Exit or Quit option.

Once you organize your menu items into related groups, visually separating these groups by inserting a horizontal separator bar in between them is often helpful. The following procedures outlines the steps required to insert a separator bar in between your menu items.

1. Finish building the menu system for your REALbasic application, making sure your group-related menu items are together in a logical order.
2. Select a menu where you want to insert a separator bar.
3. Click the menu item, and then insert the separator bar.
4. Click the Add Separator button located in the Menu Editor's toolbar or click Project ➤ Add ➤ Separator. The separator bar is immediately visible, as Figure 4-16 shows.

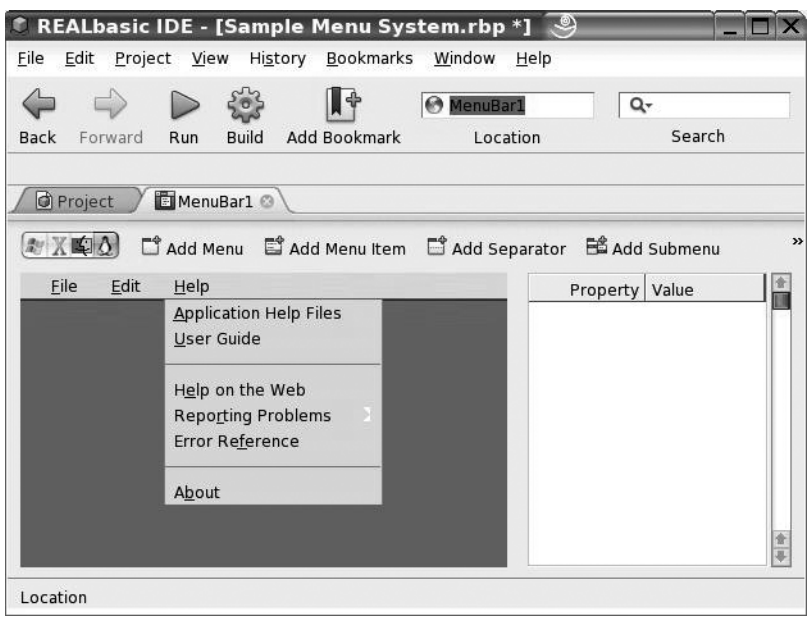

Figure 4-16. *Visually improving the organization of menu items using separator bars*

5. Repeat steps 2 through 4 to add additional separator bars to other menus, as needed.

■ **Tip** REALbasic automatically inserts separator bars just beneath the currently selected menu item. If, after adding the separator bar, you decide it needs to be moved to a different location within the menu, you can do so by selecting the separator bar, and then dragging-and-dropping it at the appropriate spot.

Setting Up Shortcut Keys

Shortcuts are designed to provide users with easy access to menu item functionality via the keyboard. For example, in most Macintosh applications, users can save their work by pressing Command+S. Likewise, users can save their work on Windows and Linux applications by pressing Ctrl+S.

While not required, most users today have become savvy enough to look for and expect to find application shortcuts, so it is important that you provide them. Otherwise, you run the risk of disappointing your users and, perhaps, making them think your applications are not yet ready for prime time. Fortunately, REALbasic makes the process of assigning shortcuts to your menu items a snap. Given how easy defining shortcuts is, it is important that you add them to your applications to make your applications look as professional as possible.

To specify shortcuts, you need to modify the following properties for each menu or menu items:

- **Key**. A single letter pressed in conjunction with a modifier keys to trigger a menu event handler.
- **MenuModifier**. When enabled, the user must press and hold Control on Windows (or Command on Macintosh) in conjunction with the specified key to trigger the menu's event handler.

■ **Note** Macintosh, Windows, and Linux all begin looking for shortcuts starting with the left-most menu. Therefore, if you assign the same shortcut to a menu item located on two separate menus, only the first instance (the left-most instance) will work as expected.

For example, the following procedure demonstrates how to assign a shortcut of Command+I or Ctrl+I to a menu item named Properties on a File menu.

1. Finish building the menu system for your REALbasic application.
2. Select the File menu to expand and see its contents.
3. Select the Properties menu item.
4. Specify *I* as the value of the Properties menu item's Key property.

5. Enable the MenuModifier property by selecting it. The new shortcut is immediately displayed just to the right of the menu item, as Figure 4-17 shows.

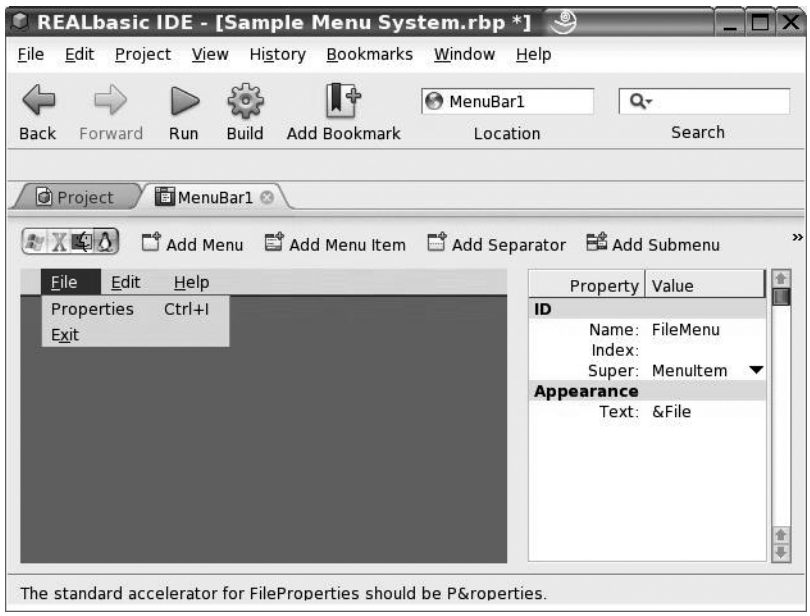

Figure 4-17. *Assigning a shortcut to a menu item, as shown on Linux*

REALbasic provides additional options for specifying more complex forms of shortcuts. For example, you might want to set up a shortcut that is accessed as Ctrl+Shift+A (on Windows and Linux) or Command+Shift+A (on Macintosh). To set up this type of shortcut, you must specify a value for the Key property, and then enable both the MenuModifier and AlternateMenuModifier properties.

If you are developing applications only for Windows or Linux, you also have the option of enabling the PCAltKey property, which requires the Alt key also be pressed to access a shortcut. For example, to set up a shortcut on Windows or Linux as Ctrl+Alt+H, you need to specify *H* has the value of the Key property and enable both the MenuModifier and PCAltKey properties.

REALbasic also provides two Macintosh-only modifier keys, which you can use to create additional shortcut variations on those OSs. For starters, when you select the MacControlKey property, it requires you to hold down Control in addition to any other specified keys. And, when you enable the MacOptionKey, it requires you also to press Option.

Defining Accelerator Keys on Windows and Linux Menus

Another important menu system feature to implement when you are developing applications for Windows and Linux is accelerator keys. *Accelerator keys* provide you with the capability to activate menus and menu items using only the keyboard. A menu or menu item's accelerator key is activated by pressing the Alt key, and then pressing the appropriate key.

FOLLOW ACCEPTED DESIGN CONVENTIONS

Today's computer users have become sophisticated and demanding, so it's critical that your applications measure up to their expectations. One expectation most users have is that their applications follow a predictable design pattern. This means, among other things, that an application's menu system presents its menus in the manner they expect. For example, menu headings should be labeled File, Edit, and Help, not Document, Modify, and Assistance. In addition, users expect to see certain features on certain menus. For example, the File menu should contain menu items for opening and closing files. Putting these menu items under a different menu can frustrate and confuse users.

Also important is that your menus and menu items include shortcuts, where appropriate, and that you follow standards appropriate for naming your shortcuts as appropriate for each OS where your applications will run. For example, if your application provides users with the capability to select and copy data, then your users will expect to see a Copy menu item located under an Edit menu. In addition, Macintosh users will expect your application to provide a Command+C shortcut. Likewise, Windows and Linux users will expect to see a Ctrl+C shortcut. If you fail to provide a shortcut or you assign a different shortcut key, you run the risk of missing your users' expectations. To help provide you with some guidance, Table 4-2 provides a listing of shortcut key definitions that represent common application-shortcut standards.

Table 4-2. *Recommended Reserved Operating System Shortcut Keys*

Menu	Macintosh	Windows	Linux	Command	Description
File	⌘-N	Ctrl+N	Ctrl+N	New	Create a new file
	⌘-O	Ctrl+O	Ctrl+O	Open	Open an existing file
	⌘-W	Ctrl+W	Ctrl+W	Close	Close a file
	⌘-S	Ctrl+S	Ctrl+S	Save	Save a file
	⌘-P	Ctrl+P	Ctrl+P	Print	Print a file
	⌘-Q	Ctrl+Q	Ctrl+Q	Quit	Close the application
Edit	⌘-Z	Ctrl+Z	Ctrl+Z	Undo	Undo a previous operation or command
	⌘-X	Ctrl+X	Ctrl+X	Cut	Remove selected data; place on clipboard
	⌘-C	Ctrl+C	Ctrl+C	Copy	Copy selected data; place on clipboard
	⌘-V	Ctrl+V	Ctrl+V	Paste	Copy data from clipboard to insertion point
	⌘-A	Ctrl+A	Ctrl+A	Select All	Select all data
	⌘-Period	Esc	Esc	Terminate	Terminate the current operation
	⌘-M	Ctrl+M	Ctrl+M	Minimize	Minimize the window

Accelerator keys are identified by the presence of an underscore character somewhere in the text of the menu, submenu, or menu item. For example, the letter *F* is generally used as the accelerator key for the File menu and is displayed as File. When the user presses the Alt key, and then presses the *F* key, an application displays the contents of its File menu. Once displayed, the user can access any of the submenus of menu items located under that menu by continuing to press the Alt key, and then press the accelerator key assigned to the submenu or menu item. Using accelerator keys, a user may access any part of an application's menu system without ever touching their mouse. Figure 4-18, shows how accelerator keys are implemented in a typical Windows application.

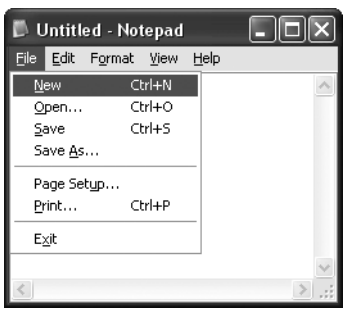

Figure 4-18. *Examining the accelerator key provided on the File menu of Microsoft's Notepad application*

Accelerator keys are specified by placing an ampersand (&) character in the Text property belonging to a menu, submenu, or menu item. For example, to assign an accelerator key of *B* for a menu item that has a Text value of Background, you would place the ampersand character just at the beginning of the word, as the following shows.

1. Finish building the menu system for your REALbasic application.
2. Select the menu where the menu items reside.
3. Select the menu item and locate its Text property.
4. Add the ampersand (&) character to the beginning of the specified value.
5. Press Enter or Return. The results should be immediately visible.

Using Menu Items to Trigger Command and Code Execution

Once you finish putting together your application's menu system, you need to associate program code statements with each menu item to make the menu functional. Users can then activate menus and execute associated programming statements by clicking the appropriate menu item.

The process of associating program code with individual menu items is relatively straightforward, as the following shows.

1. Finish building the menu system for your REALbasic application.
2. Open the Project Editor and assign the menu bar to a window by selecting the window, and then selecting the name of the appropriate menu bar from the drop-down list of property values associated with the MenuBar property.
3. Double-click the specified window in the Project Editor to open it.
4. Click the Code View icon located on the far left-hand side of the Windows Editor toolbar.
5. To add code to your application for each menu item, add a menu event handler. This enables you to associate specific code statements with specific menu items. Click the Add Menu Handler button located on the Windows Editor toolbar.

■ **Note** A *menu handler* is a collection of code statements executed whenever you click an enabled menu item.

---

6. The entry of a menu handler is displayed in the left-hand browser area, with a single entry underneath it. By default, this entry is selected and the right-hand side of the Windows Editor displays a new procedure, as Figure 4-19 shows.

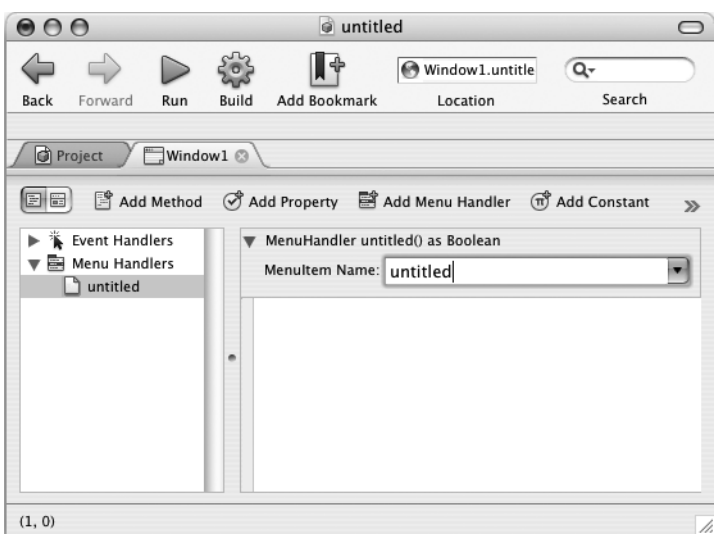

Figure 4-19. *Adding a menu handler to your REALbasic application, as shown on Macintosh*

7. Select the name of a menu item from the drop-down list for the MenuItem Name field, as you see in Figure 4-20.

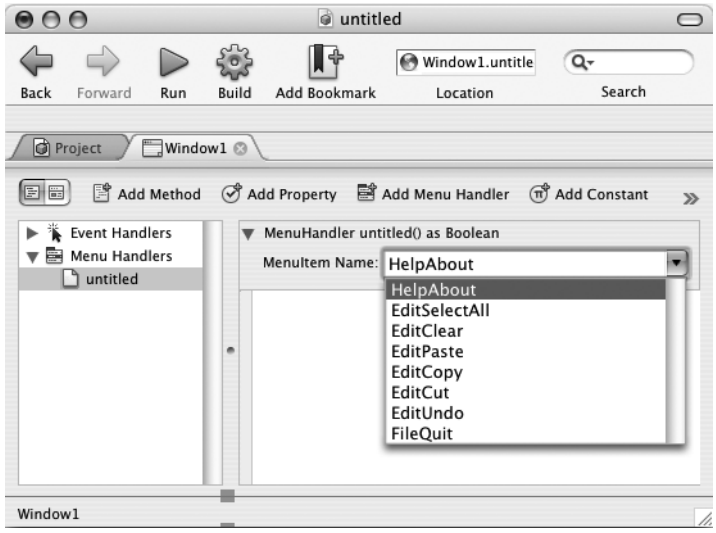

Figure 4-20. *Specify the menu for which you want to add code statements, as shown on Macintosh.*

8. Enter the code statements to be executed when the user clicks the menu item in the text edit area, as Figure 4-21 shows.

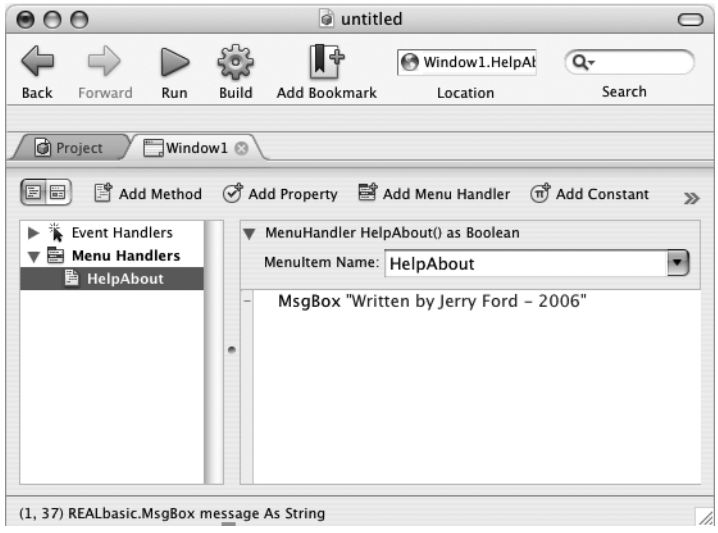

Figure 4-21. *Enter the code statements associated with the specifed menu item, as shown on Macintosh.*

9. Repeat steps 5 through 8 as many times as necessary to provide program code for each remaining menu item.

Controlling Access to Menu Items

By default, REALbasic automatically enables every menu item you add to a menu system. At times, though, you may want to prevent users from being able to click a particular menu item. You may want to disable a menu item to prevent the user from being able to click it at certain times during the execution of the application. For example, you may want to disable a menu item that enables the user to save a file as long as no file has been opened or as long as the user has not made any changes to an opened file. Later on, once the user has opened a file and made changes to it, you can programmatically enable the menu item. Note, a disabled menu item is still visible, but it is grayed out and cannot be selected.

■ **Note** For a menu item to perform an action, you must associate program code with it. Until you do so, the menu item remains grayed out and the user is unable to select it.

If a menu item should not be enabled when the application first starts, you should deselect its AutoEnable property. Later, at the appropriate time, you can programmatically enable the menu item, using a statement similar to the following:

```
MenuHandlerName.AutoEnable = True
```

In this statement, MenuHandlerName is a placeholder representing the name of the menu item you want to enable. The remainder of the statement simply instructs REALbasic to enable the menu item, enabling the user to select it.

■ **Tip** When deselected (or set to False), a menu remains disabled until the user clicks the menu that contains it, at which time you can programmatically determine if it is appropriate to enable the menu item. For example, the Cut and Copy menu items on the Edit menu are only enabled when a selection has been made within an application. Enabling these menu items when nothing has been selected does not make sense.

Reconfiguring Menu Organization

REALbasic is extremely flexible in the manner in which it lets you build menus, and create submenus and menu items. REALbasic adds new menus just to the right of the currently selected menu and it adds new menu items just beneath the currently selected menu item. At times, though, you might decide you want to change the order in which your menus or menu items are presented. REALbasic makes it easy to move things around the want you want them.

Moving Menus and Menu Items

If, after you add a new menu, you decide you don't like its current location, you can move it. One way to accomplish this is to delete the menu and add it back at the desired location. You can also drag-and-drop the menu to a new location without having to delete and re-create it by using the following procedure.

1. Open the appropriate menu bar in the MenuBar Editor.
2. Select the menu you want to move and drag it to a different location.
3. A vertical bar appears, indicating the current insertion point. Drop the menu at the desired location.

REALbasic also enables you to move menu items to different locations within the menu system. For example, the following procedure outlines the steps involved in moving a menu item from one location to another within the same menu.

1. Open the appropriate menu bar into the MenuBar Editor.
2. Select the menu where the menu item to be moved resides.
3. Select the menu item you want to move and drag it to a different location.
4. A horizontal bar appears, indicating the current insertion point. Drop the menu at the desired location.

Converting Menu Items to Menus

REALbasic also lets you convert a menu item into a new menu. This can come in handy when you make an enhancement to an application where you plan to expand on the capabilities provided by a given command. The following procedure outlines the steps involved in converting a menu item to a menu.

1. Open the appropriate menu bar in the MenuBar Editor.
2. Select the menu where the menu item currently resides.
3. Select the menu item, and then click the Convert to Menu button located on the Menubar Editor toolbar or click Project ➤ Modify ➤ Convert to Menu.
4. The menu item disappears and reappears as a new menu. Drag-and-drop the menu to the appropriate location.

Removing Menu and Menu Items

If a menu or menu item is no longer needed, you can remove it from your application by selecting it and pressing Delete or by clicking Edit ➤ Delete. Take care when deleting a menu because you also delete any submenus and menu items defined under it. Also, REALbasic does not delete any menu handler associated with a deleted menu item, so it's up to you to remember both to select and delete it.

Customizing the Apple and Macintosh Menus

As you already saw, application menus work a little differently on the Macintosh than they do on Windows and Linux. For starters, menus are not displayed on application windows on the Macintosh. Instead, menus are displayed at the top of the display area for the currently selected application.

Differences also exist between the ways that menus look and operate on Macintosh Classic and Mac OS X. Specifically, on Macintosh Classic, an Apple menu is added to the left of the File menu. On Mac OS X, though, an application menu is added to the left of the File menu, but just to the right of the Apple menu.

■ **Note** REALbasic displays the name of the application as the text value for the Application menu (for example, the name you specified as the application's name in the Mac Settings section of the APP item on the Project Editor.

To add an menu item in the Application menu on Mac OS X, you begin by placing the menu item in the menu where you want it to appear on Windows, Linux, and Macintosh Classic (if you're going to compile the application for these OSs). You then change the Super property from the default value of MenuItem to PrefsMenuItem. The result is this: the new menu item appears in the menu where it was defined on Windows, Linux, and Macintosh Classic, but it is displayed on the Application menu on Mac OS X.

You cannot add a menu item to the Apple menu on Mac OS X, but you can add it to the Macintosh Classic Apple menu. To do so, add the menu item where you want it to appear on Windows and Linux, and then change the Super property from the default value of MenuItem to AppleMenuItem. This results in the display of the menu item on the Macintosh Classic menu and the display of the menu item on the Application menu for Mac OS X. The menu item displays on the menu where you added it on Windows and Linux.

Creating a StickyPad Application

To help solidify your understanding of how to create menu systems for your REALbasic applications, this section demonstrates how to create a new application called RBQuickNote. *RBQuickNote* is a small application that enables you to create and save small, sticky note-like files on your computer.

For demonstration purposes, this application is created using the Linux version of REALbasic. However, you can use its source code to create a functional version of the application on both Macintosh and Windows. All the functionality of the RBQuickNote application is provided via its menu system, which consists of menu items located on File, Edit, and Help menus, as Figure 4-22 shows.

Figure 4-22. *The RBQuickNote application provides users with the capability to create and retrieve reminder notes and text documents.*

■ **Note** Starting in the next chapter and continuing throughout the rest of this book, you will begin learning the ins and outs of writing REALbasic code. For now, continue to key in code statements as instructed and take note of the brief explanations provided with each batch of code statements. As you progress through the remainder of this book, you may want to return and reexamine the program code included in this application.

Designing the User Interface

The RBQuickNote application is a little different from the applications you previously created in that other than the menu bar, it has no interface controls except for a single EditField control, which provides the application with a multiline entry field for collecting user input.

For the purpose of this application, the default window size of the Window1 window is fine and needn't be changed. Begin by adding an EditField control to Window1 and resize it, so it takes up the entire window. Next, change the following properties for the window and the EditField controls, as Table 4-3 shows.

Table 4-3. *Property Modifications for the RBQuickNote Application*

Object	Property	Value
Window1	Title	RBQuickNote
	Resizable	Enabled
	MaximizeButton	Enabled
EditField1	LockLeft	Enabled
	LockTop	Enabled
	LockRight	Enabled
	LockBottom	Enabled
	Multiline	Enabled
	TextColor	&c0000FF

The next step in creating the RBQuickNote is to create its menu system. Begin by double-clicking the menu bar item located on the Projects screen to open the Menubar Editor. At this point, you should see File and View menu. Modify the menu system for Menubar1 by adding the menu items you see in Table 4-4.

Table 4-4. *Menus and Menu Items for the RBQuickNote Application*

Menu	Menu Item	Text Property
File	FileOpen	&Open
	FileSave	&Save
	FileClear	C&lear
	FileQuit	E&xit
Edit	EditUndo	&Undo
	EditCut	Cu&t
	EditCopy	&Copy
	EditPaste	&Paste
	EditClear	&Delete

Menu	Menu Item	Text Property
	EditSelectAll	Select &All
Help	HelpAbout	&About

You do not have to add the FileQuit menu item on the File menu or any of the menu items for the Edit menu. REALbasic already added these menu items for you. In addition, you do not have to add any program code for these menu items as REALbasic has already provided everything these menu items need to do their job.

At this point, the menu system for the RBQuickNote application is complete and should look like the example Figure 4-23 shows.

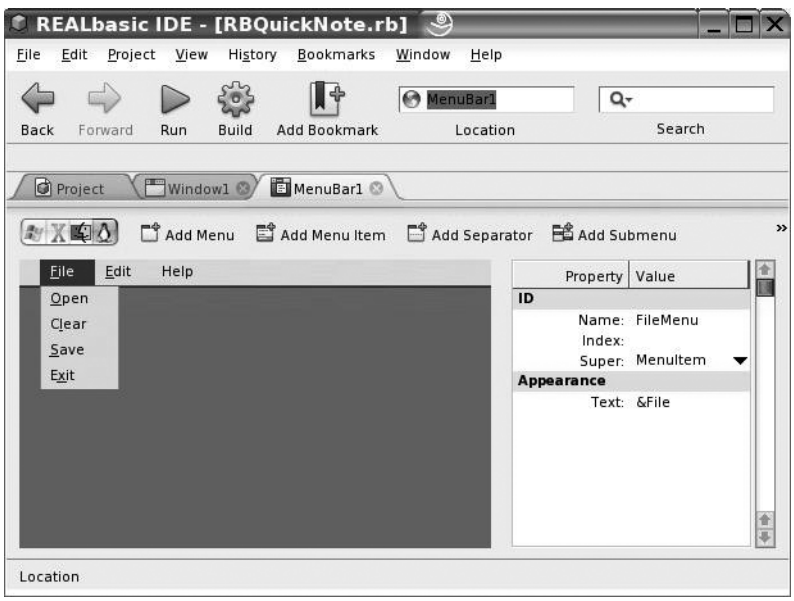

Figure 4-23. *The menu system for the RBQuickNote application*

Supplying Application Code

All code for the RBQuickNote application is placed in four event handlers, one for each menu item you added to the File and Help menus. Begin by opening the MenuBar1 in the REALbasic MenuBar Editor and switching over to Code view. Next, click the Add a Menu Event Handler button located on the MenuBar Editor toolbar. A Menu Handler's entry is added to the left-hand browser area in the Menubar Editor; a single entry is displayed beneath it with a text property of untitled. By default, this entry is selected and the right-hand side of the Windows Editor is set up to accept the code statements you want to associate with this menu item.

Select the entry for the FileClear menu item from the drop-down list for the MenuItem Name field and enter the following program statements.

```
Window1.EditField1.Text = ""
```

When executed, this statement tells REALbasic to clear out any currently displayed text in the application EditField control. Next, click the Add Menu Handler button again to add a new menu handler. Select the entry for the FileOpen menu item from the drop-down list for the MenuItem Name field and enter the following program statements.

```
Dim f as FolderItem
Dim TextInput as TextInputStream

Window1.EditField1.Text= ""

f = GetOpenFolderItem("Application/Text")

If f <> nil Then

  TextInput = f.OpenAsTextFile

  Window1.EditField1.Text = TextInput.ReadAll

End If
```

■ **Note** To learn more about the basic file operations implemented in the RBQuickNote applications, see Chapter 9.

These statements are responsible for creating a new empty file the user can use to create a new note. Next, click the Add Menu Handler button again to add another new menu handler. This time, select the entry for the FileSave menu item from the drop-down list for the MenuItem Name field and enter the following program statements.

```
Dim f as FolderItem
Dim TextStream As TextOutputStream

f=GetSaveFolderItem("Application/text","TextFile")

If f <> nil Then

  TextStream=f.CreateTextFile
  TextStream.WriteLine Window1.EditField1.Text
  TextStream.Close
End If
```

These statements are responsible for saving the note created by the user as a text file in whatever folder the user chooses to save them. Finally, click the Add Menu Handler button to add one last new menu handler. This time, select the entry for the HelpAbout menu item from the drop-down list for the MenuItem Name field and enter the following program statements.

```
MsgBox"RBQuickNote Version 1.0 - 2006"
```

This statement uses a built-in REALbasic function called MsgBox to display a text message in a preformatted pop-up dialog window.

Figure 4-24 demonstrates how the RBQuickNote looks when it's running. As you can see, it functions much like a simple text-entry application. Any text entered into it is displayed in blue and it can be minimized or maximized. By default, any files created by the application are assigned a file extension of .1st, which is a default file extension assigned by REALbasic.

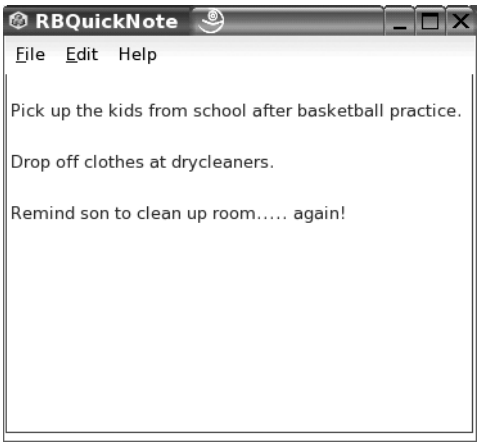

Figure 4-24. *A demonstraton of the RBQuickNote application in action*

Testing RBQuickNote

If you have not done so, go ahead and save your application. Name your application RBQuickNote. The application is now ready to compile. If any errors are flagged during compilation, then you probably made one or more typos entering code statements. Take a few moments to review the code statements and fix any typos you find.

Summary

In this chapter, you learned how to complete the development of your REALbasic application's user interface by adding menus, submenus, and menu items. You learned how to configure menus, submenus, and menu items by defining shortcuts. In addition, you learned how to define accelerator keys for applications that run on Windows and Linux. On top of all this, you learned how to visually organize menu items using separator bars, and how to reorganize menus and menu items using drag-and-drop. You also learned how to change menu items into menus. Finally, you learned how to trigger the execution of program code based on the selection of menu items.