Chapter 3 — Using WindowBuilder Pro

This chapter describes how to perform some of the most common and powerful tasks you can undertake with WindowBuilder Pro. It is not an exhaustive reference to all of the operations WindowBuilder Pro supports; that information is included in the Reference Manual. While the Reference Manual provides you with a complete discussion of all of the menu items, commands, and options in WindowBuilder Pro, this chapter focuses on more step-by-step explanations of the use and purpose of some of the most often used, powerful, and interesting things WindowBuilder Pro enables you to do.

Specifically, this chapter looks at the following WindowBuilder Pro operations:

- aligning and sizing objects for aesthetic value
- controlling the tab order of objects in a window or dialog
- using and creating composite panes
- storing and retrieving user interface components in the WindowBuilder Pro scrapbook
- using custom panes
- rapid prototyping using LinkButtons, LinkMenus, ActionButtons, and ActionMenus
- “morphing” controls to alter the look and feel of an interface quickly

Aligning and Sizing Objects

Moving and sizing individual objects in WindowBuilder Pro is easy. You’ve probably already had some experience with this process. Just dragging an object around in the window permits you to change its position; dragging on one of its size handles (which appear when the object is selected) lets you resize the object. But when you have two or more objects whose size and/or position should be related to one another, you can find yourself doing a lot of “pixel pushing” if you try to deal with the objects individually.

Fortunately, WindowBuilder Pro provides a number of tools and shortcuts for aligning and sizing objects with respect to one another.

Aligning Multiple Objects

When you place several buttons or fields in a Smalltalk/V window, you almost always want them to line up with one another. This is one of those little touches that is often hard to apply with other user interface tools that may require you to “nudge” each element around a pixel at a time until things look neat and tidy. WindowBuilder Pro provides an extensive set of operations to support such aesthetic clean-up.

The most important principle to remember about aligning objects is to select first the object whose positioning and/or size are correct. WindowBuilder Pro uses the first object you select as the pattern to which it fits all of the other objects you select along with it.

When you have selected the object you want WindowBuilder Pro to use as the pattern for positioning or sizing, then you can hold down the Shift key while you click on the others you want to align to the first selected object. WindowBuilder Pro marks each selected object the same way, by putting selection and drag handles around them. Figure 3-1 shows several buttons in a window selected and ready for alignment.

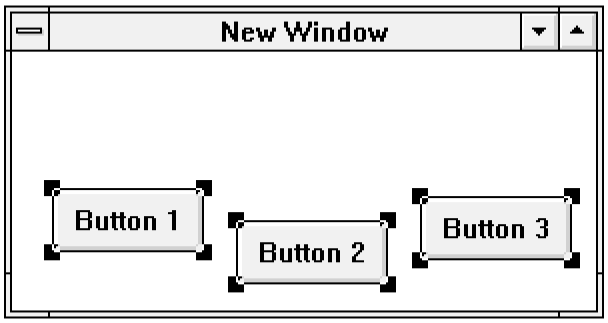


Figure 3-1. Several Selected Buttons

In Figure 3-1, we selected Button 1 first, then shift-selected Button 2 and Button 3. We are now ready to align these buttons along their top edges so that they will look right to the user. There are two ways to do this. First, you can choose the **Top** option from the **Align** menu (or use the accelerator key equivalent, Control-Shift-T). Second, you can click on the toolbar icon for top-edge alignment. It looks like Figure 3-2.

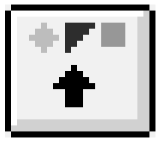


Figure 3-2. Top-Edge Alignment Icon in Toolbar

Aligning objects along their bottom, left, or right edges, or centering them with respect to one another, involves an identical process. Figure 3-3 shows all of the alignment icons in the toolbar.

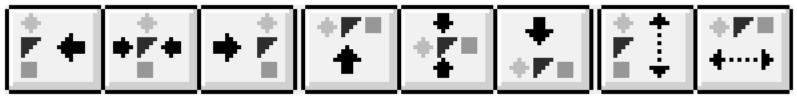


Figure 3-3. Alignment Icons in Toolbar

One of the most helpful and useful alignments WindowBuilder Pro lets you perform easily is the distribution of objects evenly over a space. You often want two or more buttons or fields to be evenly spaced along the bottom or one side of a window, for example. Look at Figure 3-4. We could clearly improve the appearance of this window by spacing the three buttons evenly vertically along the right side of the window.

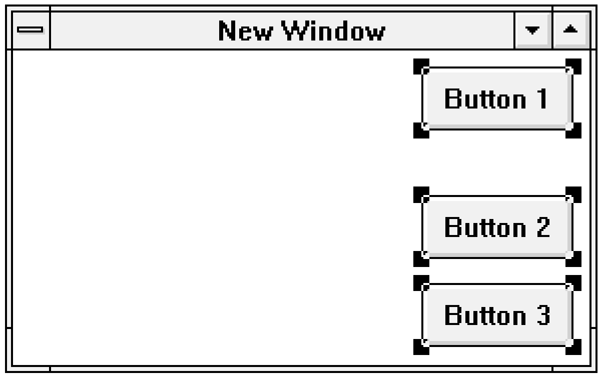


Figure 3-4. Three Badly Distributed Buttons in a Window

To do this, select all three of them (the order in which you select them in this case doesn't matter) and then distribute them vertically. You can do this by selecting **Distribute | Vertically** from the **Align** menu (or using its accelerator key equivalent, Control-Alt-Shift-D) or by choosing the **Distribute Vertically** icon from the Toolbar (see Figure 3-3). Horizontal distribution works the same way.

TIP

When you distribute objects horizontally or vertically, they space themselves evenly over the area defined by the edges of the selected buttons, not over the whole height or width of the window. WindowBuilder Pro assumes you have defined them in the right area and just want them spaced evenly. If you want the objects to be distributed over all or most of the height or width of the window, place the extreme end objects appropriately and then distribute them.

Sizing Multiple Objects

In addition to arranging objects so they line up with one another and distribute nicely over the space they occupy, you can also use WindowBuilder Pro to ensure that objects are the same size as one another.

To make sure that the height, width, or both, of one or more objects exactly matches that of a selected object, just select the object you want to use as a pattern, then extend your selection to include the other objects whose size may require adjustment. From the **Size** menu, you can then select either the **Replicate Height** option or the **Replicate Width** option (or use their keyboard equivalents, Control-Shift-H or Control-Shift-W). You can also use the toolbar icons for replicating width and height, which are located next to the **Duplicate** icon on the toolbar. (The **Duplicate** icon is nearly in the middle of the toolbar.)

Another way of sizing objects in WindowBuilder Pro is to use the auto-sizing feature. When auto-sizing is turned on — either with the **Options | Auto Size** menu choice or with the icon immediately to the left of the **Replicate Width** icon on the toolbar — objects automatically size themselves appropriately. What constitutes appropriate sizing depends on two primary factors: the type of object involved and the size of its label or contents. You can also auto-size an object with the **Size | Auto Size Selection** menu option.

Controlling Tab Order

The order in which items in a window or dialog are selected by the user who works primarily with the keyboard can be important to your user interface. Fortunately, WindowBuilder Pro provides a remarkably fast and easy way to control the order in which items in a window or dialog will be selected when the user presses the Tab key to move from item to item.

If you don’t set up a specific tab order for the items in a window or dialog, none of the items is selectable by the user pressing the Tab key. This is almost never what you want. You usually want some items (for example, button objects) to be selectable by the user and you almost always want the order in which selectable items are chosen by the user with the Tab key to follow some pre-defined sequence.

To make any objects in a window selectable with the Tab key and to define the tab order of items in a window, just click the icon in the lower left corner of the WindowBuilder Pro window that looks like Figure 3-5.

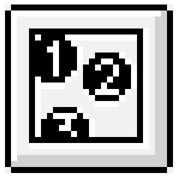


Figure 3-5. Icon for Setting Tab Order

When you click this button, WindowBuilder Pro opens a new window (Figure 3-6) with your window or dialog displayed. If you have previously set a tabbing order for this window, the items in the window that can be selected by the user with the Tab key are numbered from 1 to the number of such items that appear in the window. Otherwise, the items are all unnumbered.

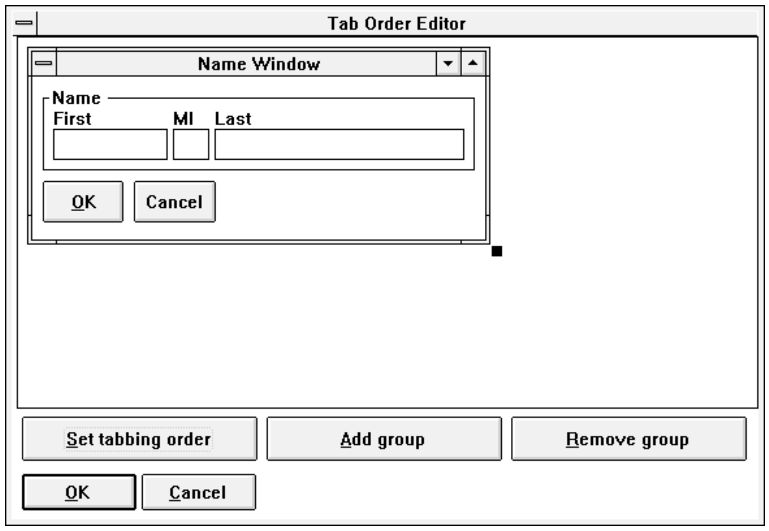


Figure 3-6. Setting the Tab Order in WindowBuilder Pro

To establish or change the Tab order of your window, click on the **Set Tab Order** button. Now just click on objects in your window that you want the user to be able to select with the Tab key, in the order in which you wish the selections to occur. When you are finished, your window will have numbered all of the selectable items (Figure 3-7) in sequence. If the sequencing matches your wishes, just click the **OK** button. Otherwise, you can click on the **Done** button and then on the **Set Tab Order** button again to start over.

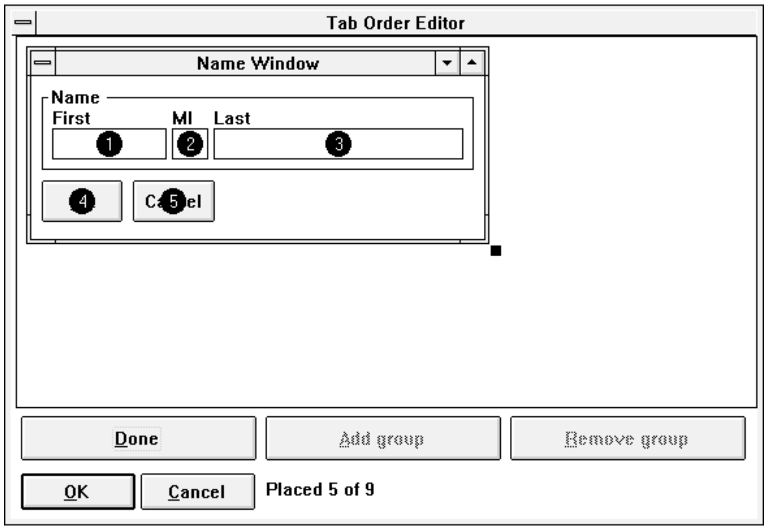


Figure 3-7. Tab Order Window After Tab Order Has Been Set

The Tab Order editor can also work with WindowBuilder Pro object groups such as the RadioButtonGroup type of CompositePane. (CompositePanes are discussed in the next section. For now, all you need to know is that a compositepane is one containing two or more subpanes treated as a single subpane.)

See the Reference Manual for a discussion of how to work with groups of objects in the Tab Order Editor.

Using and Creating CompositePanes

One of the most powerful features added to the old WindowBuilder product to create WindowBuilder Pro is that of compositepanes. This capability can enable you to create even complex interfaces extremely quickly by using pre-defined collections of related parts.

Let's explore CompositePanes in two steps. First, we'll look at how you can add such a pane to your own window and how its presence affects your Smalltalk/V programming. Then we'll see how to create your own new CompositePane and make WindowBuilder Pro aware of it.

Using an Existing CompositePane

To add an existing CompositePane to a window, all you have to do is select the Composite option from the WindowBuilder Pro Add menu. This menu displays a hierarchical submenu (see Figure 3-8) of all of the pre-defined classes of CompositePane WindowBuilder Pro knows about (Figure 3-8 assumes that the "CompositePane Examples" Add-In has been enabled).

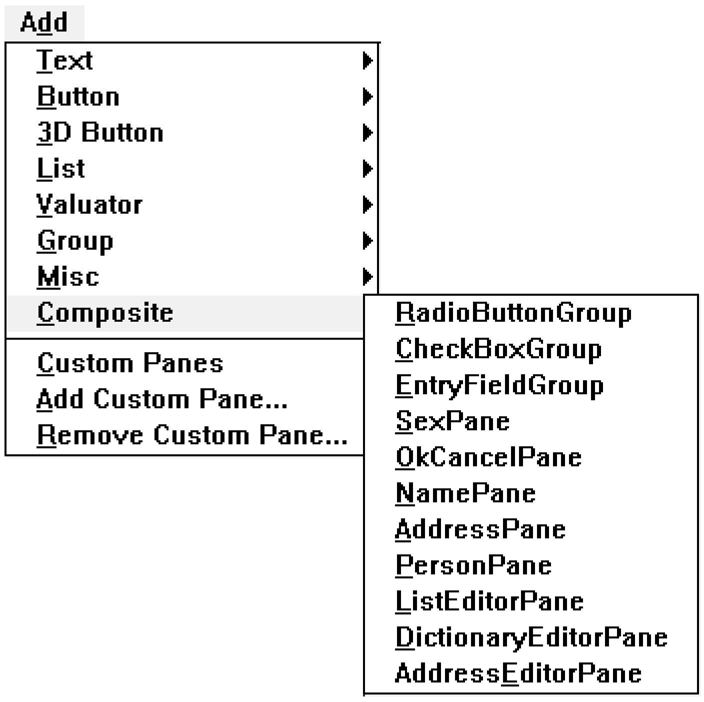


Figure 3-8. The Composite Pane Submenu

As you can see, WindowBuilder Pro comes equipped with an interesting-sounding array of CompositePane objects. Let's examine a few of them.

Create a new window by selecting **New Window** from the **File** menu. Resize it so it fills most of the available editing area in the WindowBuilder Pro window. Now from the **Add | Composite** submenu, choose the NamePane. Click near the upper left corner of your window. The result should look something like Figure 3-9.

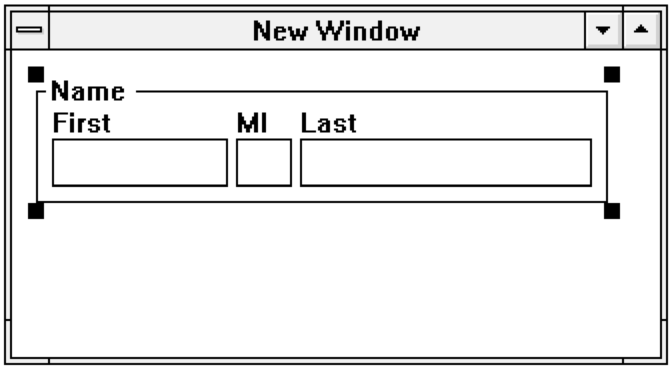


Figure 3-9. NamePane Composite Object Placed in a New Window

As you can probably guess, a NamePane Composite Object consists of three StaticText objects for the labels and three text editing objects. You can examine the individual contents of this CompositePane simply by double-clicking on it. When you do, WindowBuilder Pro opens a new editor on this subpane. Here, you can alter the Tab order or undertake any other editing you like on this CompositePane. It is, for WindowBuilder Pro's purposes, just another window.

NOTE

> Any changes you make to a CompositePane this way affects all windows that use that pane. Be sure not to make any such changes to a subpane you don't "own" if you're in a multi-programmer situation or if you really only want the change to take place for your current window or application. To make such individual changes, create a subclass of the CompositePane and use the subclass in your application window.

Now let's add another CompositePane to our window. From the **Add** menu, choose **Composite | AddressPane**. Now click just below the NamePane you just placed. The result should look something like Figure 3-10.

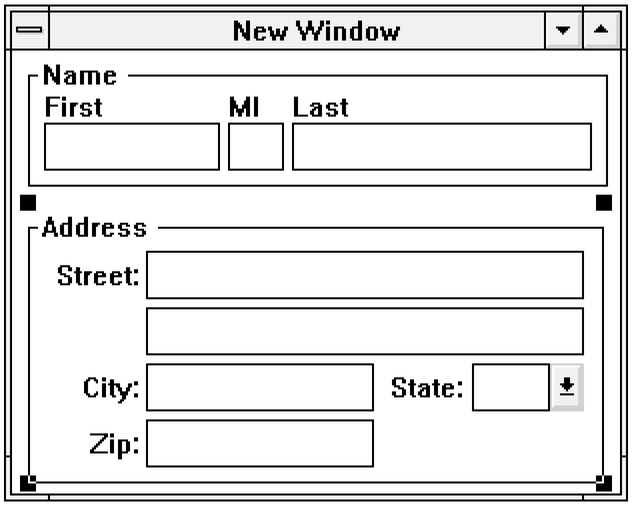


Figure 3-10. NamePane and AddressPane Placed in Window

Test this window. One thing you'll notice is that if you click in the First Name field of the NamePane and then tab a few times, the cursor will never leave that pane. The only way to get to the AddressPane's fields is to click in one of them expressly. Then using the Tab key will move you around inside that group of fields and not into the NamePane. Close the test window and select the Tab Order Editor in the WindowBuilder Pro editing window. Click on the **Set Tabbing Order** button, then click on the NamePane followed by the AddressPane. Click **OK.** Now test the window again. The Tab key now moves around correctly, both within and between the two CompositePanes.

Creating a CompositePane

These CompositePane objects are obviously pretty powerful. Let's learn how to create one of our own.

1. Create a new window in WindowBuilder Pro. When it is open, create three radio buttons like those shown in Figure 3-11.

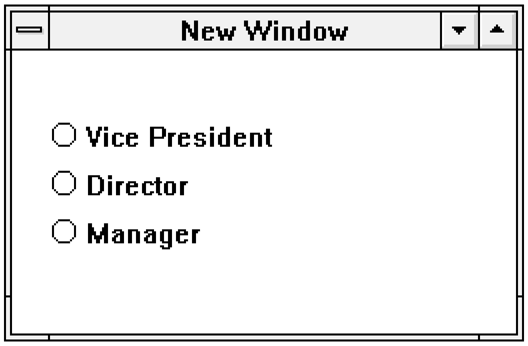
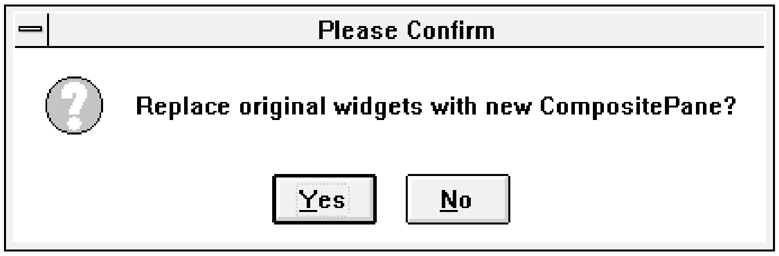


Figure 3-11. Three Radio Buttons to Start a CompositePane

2. Select all three of the radio buttons, either by Shift-clicking on them or by drawing a rectangle around them from outside their boundaries.
3. From the **File** menu, choose **Composite Panes | Create.** WindowBuilder Pro will open a new editor on the group of radio buttons, treating them as a group.
4. Save the new CompositePane object under the name TitlePane. (Or make up your own name if you prefer.) When you have done so, exit this WindowBuilder Pro editor. As you do, the dialog shown in Figure 3-12 will appear.


Please Confirm

Replace original widgets with new CompositePane?

Figure 3-12. Dialog Asking to Replace Original Widgets

As a rule, you want to answer affirmatively to this dialog. Whether you do or not, you'll be returned to the window in which you created this CompositePane object.

You can add this newly created CompositePane to the WindowBuilder Pro scrapbook or to the **Add** menu if you like. We explain how to add items to the scrapbook and how to use them in the next major section. The section following the next one discusses how to add custom panes to your WindowBuilder Pro environment.

Programming With CompositePane Objects

Before you combine elements into a new CompositePane object, you should write methods for each of the active objects.

In general, there is no real difference between writing methods for a CompositePane object and writing methods for any other kind of object. You simply write methods to respond to the events with which you've identified the pane in the WindowBuilder Pro window or dialog editor.

However, the nature of a CompositePane object dictates that you often want it to be more useful to other panes in your application and more reusable in future applications. With all of its behavior coded internally, however, this reusability is harder to attain because the CompositePane object has embedded within it panes that are not accessible to other components of your system. Fortunately, Smalltalk/V provides a straight-forward mechanism for addressing this problem.

To understand this mechanism, open a Class Hierarchy Browser on the OkCancelPane class. Check out its `cancel` method. Rather than carrying out some action, which is what you would expect given your previous experience with Smalltalk/V and with WindowBuilder Pro, this pane has a single-line method:

```
self event: #cancel
```

The `ok` method has a similar line of code. As you might be able to figure out (particularly if you've read Chapter 7 of *Smalltalk Programming for Windows,* where events are discussed), this line simply sends the event "cancel" to the owner of the OkCancelPane (most likely a window). Take a look at the class methods of OkCancelPane, and notice that a method called `supportedEvents` is one of them. Its code looks like this:

```
supportedEvents
    "Answer the events supported by this subpane."
    ^super supportedEvents
        add: #ok;
        add: #cancel;
        yourself
```

This technique of "exporting" events to the next higher level of the class hierarchy is the way you make events within CompositePanes accessible to other objects in your application and in the Smalltalk image.

Using the WindowBuilder Pro Scrapbook

You can gather commonly used controls and other panes into a scrapbook inside WindowBuilder Pro so that you can retrieve them easily later when you need them. You can have only one scrapbook open at a time, although you can merge two or more scrapbooks into one after you’ve opened your main scrapbook.

The WindowBuilder Pro scrapbook is divided into chapters. Each chapter contains one or more “pages,” with each page in turn containing a single control or group. As it comes from Objectshare Systems, the scrapbook contains chapters called:

- Buttons
- Quick Reference
- Radio Buttons
- Widget Sets

You can page through the scrapbook to find a control or group you’d like to use in your window, then just click the **OK** button in the scrapbook to load your cursor with it and close the scrapbook in one operation. You can then place this object from the scrapbook exactly as you do any other kind of pane in WindowBuilder Pro.

Controls stored in the **Quick Reference** chapter of your scrapbook are directly accessible from the **Scrapbook** menu in WindowBuilder Pro. All others require you to open the scrapbook, locate the chapter and the control you want and then click the **OK** button to retrieve them.

To save a control in your scrapbook, select the object(s) you wish to store from your current window and choose **Scrapbook | Store.** A dialog like the one shown in Figure 3-13 will appear, showing the names of all the present chapters. You can create a new chapter by clicking on the **New Chapter** button and giving the new chapter a name, or you can select a chapter from the list and then give the new object a page name. Notice that until you pick a chapter name, WindowBuilder Pro won’t let you store the new object because it doesn’t know where to put this page. A page may be stored in as many chapters as you like.

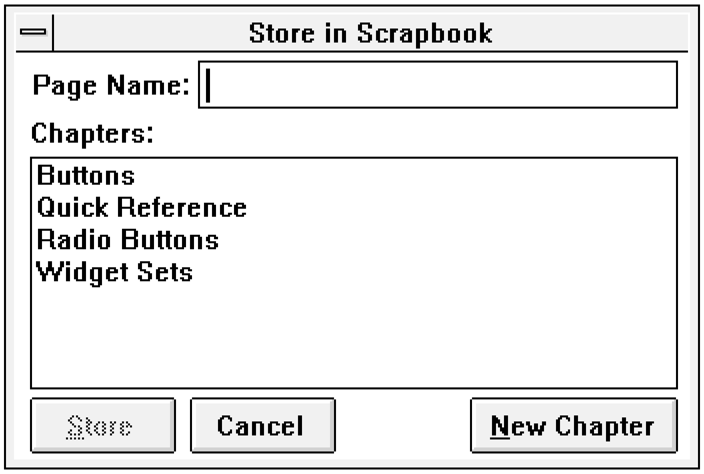


Figure 3-13. Adding a Page to a Chapter in the Scrapbook

When you end a session, WindowBuilder Pro automatically saves the scrapbook for you.

The scrapbook also includes controls for deleting pages and chapters. From the **Scrapbook** menu, you can merge a stored scrapbook with the open one, delete the current scrapbook and start over with a new one, and perform other maintenance operations.

You should be aware that the scrapbook is essentially a persistent object within your Smalltalk/V image. There is no need to save it explicitly. However, if you wish to create a Smalltalk ObjectFiler file containing the scrapbook, you can use the **Scrapbook | Save...** menu option.

Using Custom Panes

When you receive WindowBuilder Pro from Objectshare Systems, the Custom Pane menu is generally empty. You can add any pane to this sub-menu that you wish. When you choose the **Add | Add Custom Pane...** option, WindowBuilder Pro presents you with a list of all the components in the system. You can then select one of these components and it will be added to the cascading menu from the **Add | Custom Pane** menu option. You can use this approach to add your own newly created custom subclass-es of WindowBuilder Pro classes, for example.

Custom panes do not have a toolbar icon. See **Appendix A** for details on actually adding your subpane to one of the palettes.

Rapid Prototyping Tools

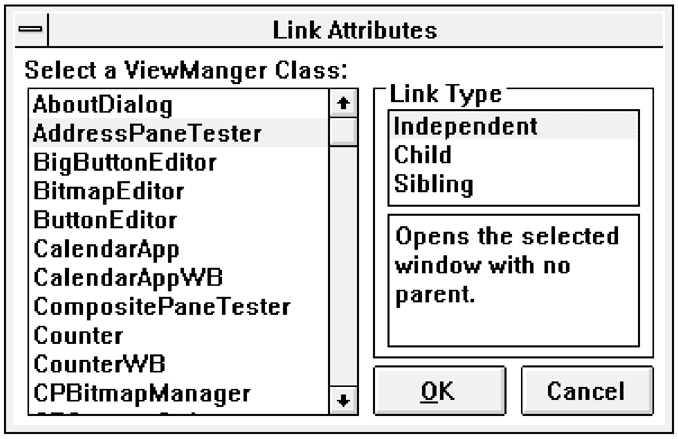
One of Smalltalk's long-understood strengths is its role as a rapid prototyping tool. WindowBuilder Pro takes that concept several steps beyond where Smalltalk begins by giving you the ability to create both buttons and menus that can automatically link to other windows or dialogs and perform simple tasks with little or no programming on your part. In fact, someone who doesn't know Smalltalk at all can build a prototype quickly using these features of WindowBuilder Pro.

Linking With Buttons and Menus

One of the most common things you probably program buttons and menu choices to do is to open another window or display a dialog. Particularly as you construct prototypes, you often want to demonstrate simply how the various windows in an application connect with one another.

WindowBuilder Pro includes a built-in feature to make programming such buttons and menus a painless task. The process is simple. Let's take a look at how you create and connect a LinkButton.

1. Add a LinkButton to your window. You can do this either from the **Add | Button** menu or by selecting the button icon labeled "Link" from the button toolbar.
2. Place the button where you want it.
3. Double-click the button. A dialog like the one shown in Figure 3-14 appears (only windows defined in your system will be displayed).



AboutDialog
AddressPaneTester
BigButtonEditor
BitmapEditor
ButtonEditor
CalendarApp
CalendarAppWB
CompositePaneTester
Counter
CounterWB
CPBitmapManager

Independent
Child
Sibling

Opens the selected window with no parent.

Figure 3-14. The LinkButton Dialog

In the dialog shown in Figure 3-14, you provide WindowBuilder Pro with two pieces of information:

- the name of the ViewManager class (i.e., window) that you wish to link to with this button
- the nature of the link (that is, whether the relationship between the window containing the LinkButton and the linked window is one of parent-child, sibling-sibling, or no relationship at all)

If you get confused about the three types of relationships, notice that if you select one of the choices, the editing rectangle immediately beneath the choice list displays a description of that relationship.

Once you select the ViewManager class to connect to and the nature of the relationship in the link, click the **OK** button in the dialog. That's all there is to it. Clicking on this button will now open the designated ViewManager window in the appropriate way.

LinkMenus work the same way, but creating them involves a different process. To create a LinkMenu, follow these steps:

1. Click on the menu icon in the lower right portion of the WindowBuilder Pro window editor. (This icon looks like Figure 3-15).

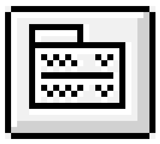


Figure 3-15. Icon for Adding Menus to Panes

2. The dialog shown in Figure 3-16 appears. This is the standard dialog for adding menus to panes. It is explained in detail in the Reference Guide.

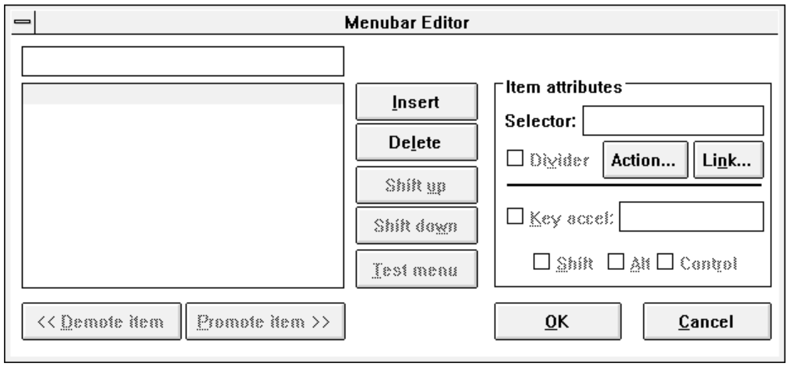


Figure 3-16. Menubar Editor

3. Create a menu and a menu item, then select the menu item. In the upper right portion of the dialog, click on the **Link...** button. In a moment, the dialog we saw earlier for LinkButtons (Figure 3-14) appears. You can now follow the same procedure as explained above.

ActionButtons and ActionMenus

When you want to create a button or menu item that activates a simple method or one that you frequently use and have therefore pre-defined, you can create an ActionButton or an ActionMenu. These objects are quite similar to their linking counterparts, discussed in the previous section. The major difference is that you can choose almost any action you wish the application to take when the user clicks on the button or chooses the menu item. They are not confined simply to linking. You could think of a LinkButton as a special case of an ActionButton in many ways.

Creating and programming an ActionButton is quite similar to the process for creating and programming a LinkButton. Here are the steps:

1. Add an ActionButton to your window. You can do this either from the **Add | Button** menu or by selecting the button icon labeled “ACT” from the button toolbar.
2. Place the button where you want it.
3. Double-click the button. A dialog like the one shown in Figure 3-17 appears.

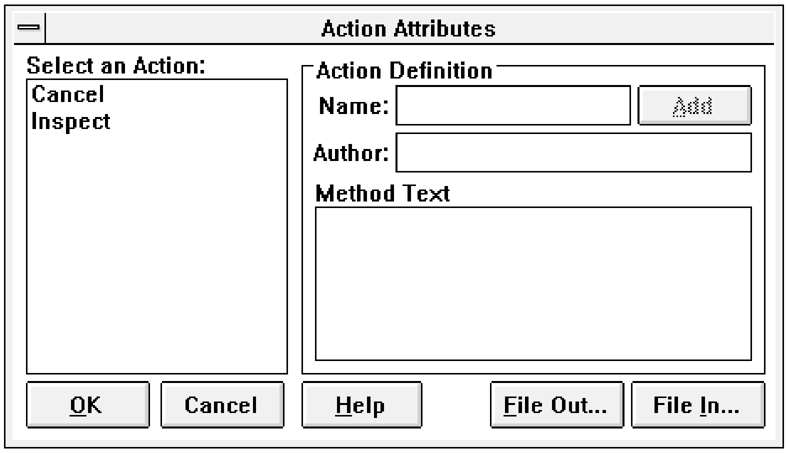
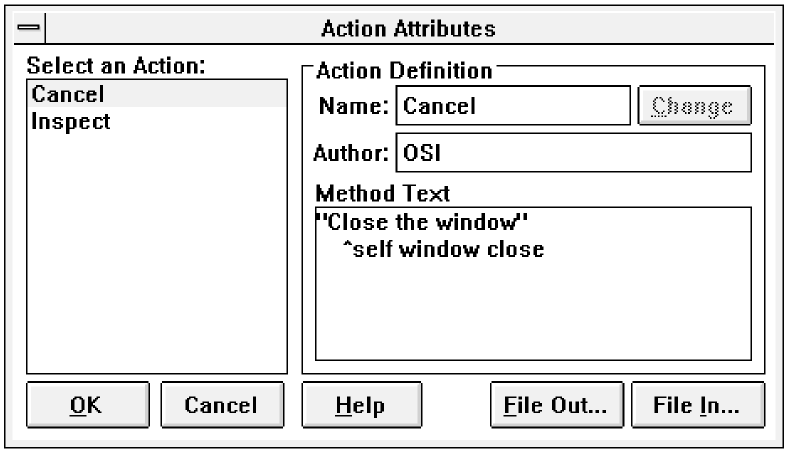


Figure 3-17. ActionButton Dialog

This dialog is more complex than the one we saw in Figure 3-14 for LinkButtons. Actions for which definitions (methods) already exist are listed in the left-hand pane of the dialog. In the right-hand pane, you can define an action to be taken when the button is clicked. This process consists of the following steps:

- giving the new action a name (use typical Smalltalk one-word syntax)
- optionally, providing the name of the author of the method
- typing or filing in the text of the method to be executed when the button is clicked

Let's take a quick look at how this process works. With the dialog displayed on your screen (if you wish), click on the "Cancel" action in the left-hand pane. Notice the contents of the rest of the dialog, as shown in Figure 3-18.



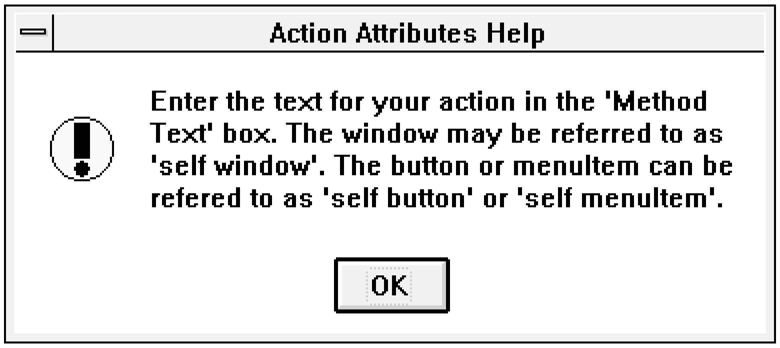
Name: Cancel
Author: OSI

```
"Close the window"
   ^self window close
```

Figure 3-18. "Cancel" Option Chosen in ActionButton Dialog

You can use the **File In...** and **File Out...** buttons in the dialog in Figure 3-18 to save a method you expect to use later or to install a method you previously filed out.

Clicking on the **Help** button displays the dialog shown in Figure 3-19, which will assist you in putting together your action methods.



Enter the text for your action in the 'Method Text' box. The window may be referred to as 'self window'. The button or menuItem can be refered to as 'self button' or 'self menuItem'.

Figure 3-19. Help Dialog for ActionButtons

Let's create our own example. We'll create a button whose role is to display a MessageBox that tells the user today's date. Follow these steps:

1. Select the ActionButton tool as described above.
2. Double-click it to open the ActionButton dialog.
3. In the **Name** field, type "ShowDate" and click the **Add** button.

4. Put your name into the **Author** field.
5. In the **Method Text** editing rectangle type the following method (comment is, of course, optional):

```
"Display a dialog box with today’s date."
MessageBox message:
    'Today is ', (Date today asString).
```

6. Click on the **OK** button.
7. Now test your window.

This is a good example of how you can create small methods directly in WindowBuilder Pro without having to leave the environment, open the Class Hierarchy Browser, create a method, save it, and then come back to WindowBuilder Pro to test it.

The Power of Morphing Controls

Sometimes when you are creating the user interface for a program, you’d like to be able to experiment with various kinds of controls in specific situations. For example, would a particular list of options be better presented to the user as a list box or as a collection of radio buttons or check boxes? In most UI construction toolkits, including earlier versions of WindowBuilder, this kind of experimentation was difficult. You essentially had to begin from scratch with each type of control.

In WindowBuilder Pro, you can tell a control to undergo a metamorphosis (thus the term “morph”) from one type of control to another. As the control modifies itself to the new type of control, it retains as much information and behavior as it can, in other words, as much as makes sense to the new control type. (Be careful here, though. Any information you have encoded with the control that isn’t appropriate to the new type of control will be lost and you’ll have to re-supply it if you decide to return to the first type of control.)

Let’s create an example.

1. Create a new window or, if you already have an experimental window open, feel free to use it.
2. Select a RadioButtonGroup object from the **Add | Composite** menu.
3. Place a radiobuttongroup pane in your window.
4. Double-click on the RadioButtonGroup. A dialog like the one shown in Figure 3-20 appears. You can use this dialog to create your radio buttons within the RadioButtonGroup.

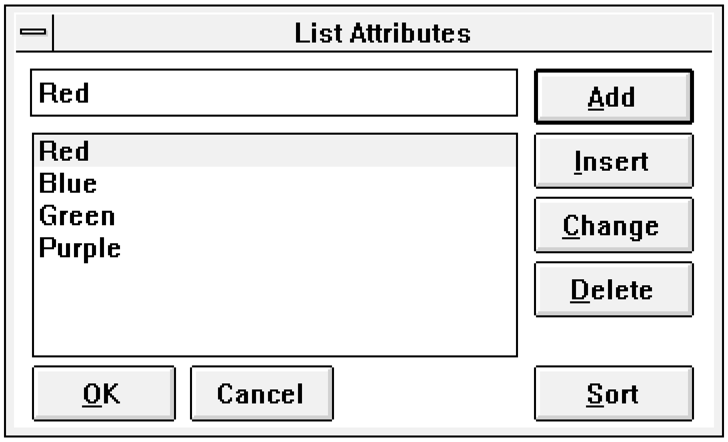


Figure 3-20. RadioButtonGroup Editing Dialog

5. Name the buttons “Red,” “Blue,” “Green,” and “Purple.” Your window should now look something like Figure 3-21.

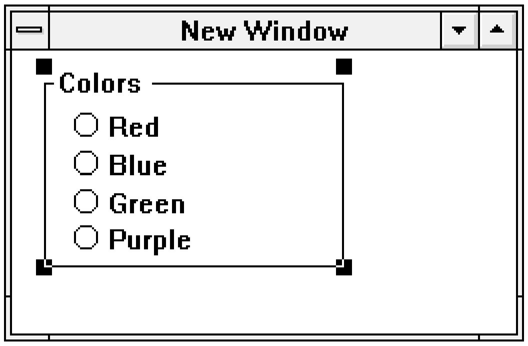


Figure 3-21. RadioButtonGroup Created in a New Window

6. Save this window and test it. When you’re satisfied that it works as expected, close it.
7. Select the RadioButtonGroup object. You can instruct WindowBuilder Pro to morph this object into another kind in one of several ways: selecting **Morph** from the **Edit** menu (or using Control-M, its accelerator option), or click in the object with the right mouse button and select **Morph** from the resulting popup menu (see Figure 3-20).

NOTE

The contents of the popup menu depend on the widgets installed in your system. The menu that you see will probably be different.

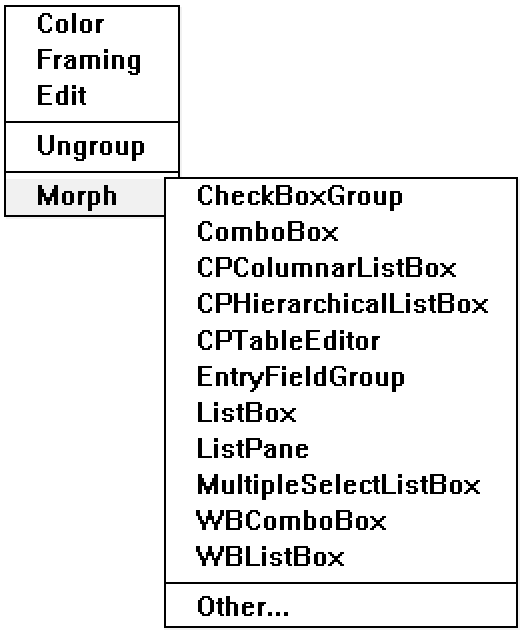


Figure 3-22. Morphing Choices for a RadioButtonGroup

NOTE

We strongly recommend you use the right-button method because the resulting popup displays the most appropriate choices of control for the type of control you are morphing. The other approach produces a dialog of all of the possible SubPane subclasses to which you could morph the selected control. Many of these are quite inappropriate, but WindowBuilder Pro doesn't intend to tell you how to design your windows, only to help you do so. From the popup, you can still choose the **Other...** option and choose a control type other than those we've pre-selected for you if you want or need to do so.

8. Select ListBox from the popup menu or from the dialog. In a few moments, the RadioButtonGroup is magically replaced by a ListBox with the same contents.
9. Test the window again to convince yourself that all still works.

You'll soon find that morphing controls is one of the most powerful concepts in WindowBuilder Pro.

Some Closing Thoughts

As we said at the beginning of this chapter, this has not been a feature-by-feature tour of WindowBuilder Pro. We leave that task to the Reference Guide, so that all of the necessary information about using this tool can be in one convenient place when you need it. Rather, our purpose here has been to give you an introduction to some of the niftier and more useful concepts in WindowBuilder Pro, the ones we have come to use most often as we've built and tested this new product for you.

Feel free to explore. The remaining chapters of the tutorial examine each of the examples from the *Smalltalk Programming for Windows* book to see how they would be approached if you were starting with WindowBuilder Pro rather than from scratch in Smalltalk/V. Along the way, we'll point out some other interesting and useful features of the product.