Chapter 9 — General Reference

This section describes the various features of WindowBuilder Pro in detail. It includes descriptions of WindowBuilder Pro's subpane editors, from the menu and menubar editors to the subpane framing editor. It also describes the extensive collection of subpane manipulation tools built into WindowBuilder Pro. Finally, it describes the coding issues of which you should be aware when using WindowBuilder Pro.

Mechanics and Techniques

Selecting Multiple Subpanes

There are many operations within WindowBuilder Pro which can operate on multiple subpanes in the selection (e.g. aligning a group of subpanes, or setting the same font/color/framing for multiple subpanes). There are two methods for adding multiple subpanes to a selection: *extending* the selection, or selecting a group of subpanes, using a *rubberband selection.*

To extend the selection, simply hold the shift key down as you select another subpane. The new subpane will be added to the selection.

To do a rubberband selection, click the mouse down somewhere within the edited window, and drag across the desired subpanes while holding the mouse button down. A gray rectangle will follow the mouse, anchored at its initial position, and when you release the mouse, all subpanes at least partially falling within this rectangle will be selected. If you hold the shift key down as you do a rubberband selection, all subpanes which at least partially fall within this rectangle will be added to the existing selection.

To remove a subpane from the selection, simply select it again, holding the shift key down. The rest of the selection will remain unchanged, and this subpane will be removed.

Placing Multiple Subpanes

When you are placing the subpanes in a window, you will often find that you are placing several subpanes of the same type. It can become tedious to continually select the particular subpane from the palette, then place it.

To make this easier, WindowBuilder Pro provides a multiple subpane placement feature.

To place multiple subpanes, drop them by clicking with the right button, instead of the left. The cursor will stay loaded, and you will be able to place another subpane of the same type without reloading the cursor. The last subpane you place should be dropped with the left button.

☛ If you forget to place the last subpane with the left button, you can always unload the cursor by selecting the arrow icon from the subpane palette.

Creating a New Default Window

When WindowBuilder Pro edits a new scratch window, it uses a template as its example. For convenience, this default template can be changed to any window you design, even with subpanes in it. If you are making many windows the same size, or with many of the same components in them, you may wish to create your own default template.

To create a default template, open WindowBuilder Pro, and lay it out the way you wish. When you are finished, pull down the **File** menu, and execute the **Save as Default** method. That's all there is to it. The next time you launch WindowBuilder Pro, the starting window will appear just like this window. You can save a different default window for both dialogs and regular titled windows.

Editing Existing Windows

As promised, you can easily re-edit any viewmanagers that you've created and saved with WindowBuilder Pro. Select the **Open...** command from the **File** menu; a list of eligible viewmanagers in your image will appear. When you select one, your WindowBuilder Pro will redisplay to show the new window.

You can also launch WindowBuilder Pro on a viewmanager created with WindowBuilder Pro. To do so, select the **Edit Window…** command from the WindowBuilder Pro menu on the system Transcript. After selecting a window to edit from the dialog that pops up, WindowBuilder Pro will appear, editing this window.

By default, the list in this dialog contains only those viewmanagers that WindowBuilder Pro remembers editing. If you wish to edit another view-

manager, press the **Non WB View…** button, and another dialog will appear, containing all the viewmanagers that contain a `createViews` or `open` method. Note that some `open` methods may not have been created with WindowBuilder Pro, and will therefore only read certain information about the window (or may, in fact, yield a walkback). If a walkback occurs, WindowBuilder Pro may display incorrectly; this will not cause any problems, and can easily be rectified by editing another new or existing window.

Importing From Res Files

Many /V Windows and /V PM programmers have already created dialogs using a dialog resource editor. To provide an easy transition path, WindowBuilder Pro have provides an import command that will allow you to read in existing dialog resources, setting coordinates, styles, titles, paneNames, etc. within WindowBuilder Pro.

The **Import From Res File...** command is available from the **File** menu. If there is an associated .h file with the .res file chosen, it will be used to generate pane names.

Custom controls unknown to WindowBuilder Pro will be recognized with their coordinates and sizes intact, but will be specified as the class Subpane when the `createViews` method is generated (unless you use the Morphing utility to convert them to something else). If you wish these assigned to particular classes, you will have to edit the `createViews` method. Once you have done so, WindowBuilder Pro will correctly identify these controls in future editing.

Exporting as Res Files

In addition to importing res files, WindowBuilder Pro provides a mechanism for dialogs created with WindowBuilder Pro to be saved to disk as resource files, offering a mechanism for those developers working in multiple languages to prototype in Smalltalk, and use the same window layouts in their other environments.

To create a resource file, simply choose the **Export To Res File…** command from the **File** menu. WindowBuilder Pro will attempt to create a name for the resource file, but you can rename it to another name if you wish. Don’t worry about the suffix; WindowBuilder Pro will add it if you forget.

In addition to the .res file created, if any panes are named, these will be given matching symbolic names in an associated .h file.

Please note that .res files generated by WindowBuilder Pro are already compiled; they do not require the SDK's resource compiler.

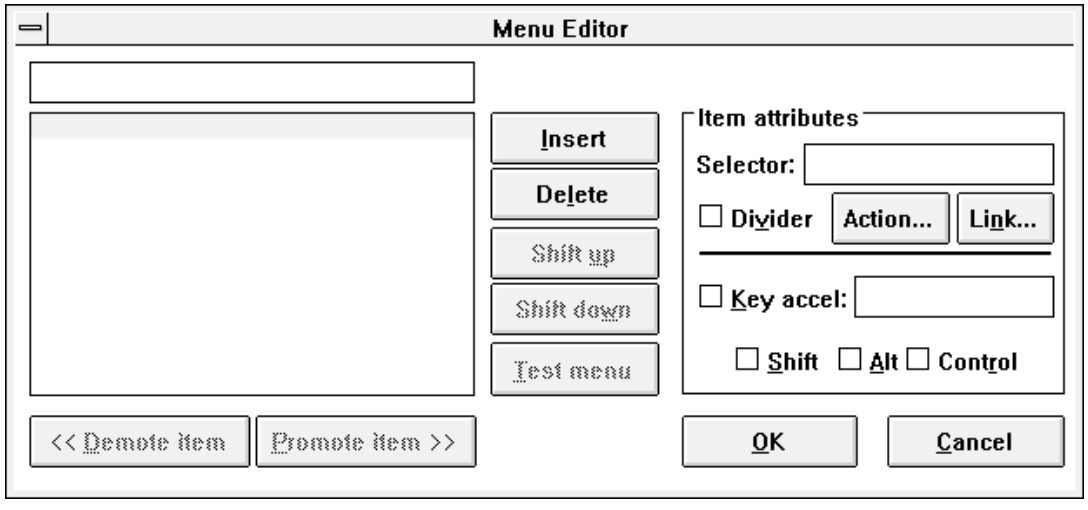
Getting an Event Summary

When building or reviewing existing windows, it often helps to get an overview of the message flow from the controls to their viewmanager. To provide a high level view of this, WindowBuilder Pro now has a command on the **Edit** menu, **Event Summary....** Executing this command assembles a summary of the controls in the window, listing the events and their associated methods.

Creating Windows and Subpanes

Adding a Popup Menu

In the Smalltalk/V environment, several types of subpanes can have popup menus associated with them. When one of these subpanes is selected in WindowBuilder Pro, the **Menu** button in the attributes pane will be enabled. To add or edit this subpane's popup menu, press the **Menu** button, and the following menu editor will appear:



The list on the left with the entry field above it is the *menu items editor.* These two controls combine to provide a special editable list box for entering the menu item names. This editor has some special properties for hierarchical menus, as we'll see below.

The pane on the right shows the attributes of the selected menu item, including the *keyboard accelerator editor* and the *selector editor.*

Entering Menu Item Names

To enter a name into the menu items editor, type it into the entry field at the top. The list below will update as you type, to show the name being added to the list. If you wish to add another name, press carriage return, and the list will add a new blank row.

Entering Hierarchical Menus

The menu items editor does not just edit a simple flat list of items; it provides explicit support for directly editing hierarchical menus as well.

If you have used the outliner in a GUI word processor, you will find this mechanism familiar. To add a submenu, enter its title, then press return to create its first menu item. Press the **Promote Item >>** button below the editor. Now when you begin typing the title of the first menu item, you will notice that it is indented, to reflect that it logically belongs to the item above it. All subsequent items you add will be placed at the same level, as descendants of the first title. When you've finished editing this submenu, press the **<< Demote Item** button below the editor. When you begin typing the next title, you will see its indentation has shifted back to match that of the submenu's title.

Of course, you needn't press the promote or demote buttons before you enter a new menu item; you can shift them around at any time and to any legitimate level you wish. Further, this indentation may be extended to any depth, so you can create hierarchical submenus as deep as you wish.

In addition to the mnemonic keys available for promote and demote (Alt+D and Alt+P), you can use the accelerator keys Ctrl+LeftArrow and Ctrl+RightArrow as well.

Rearranging Menus and Menu Items

If you wish to rearrange your menus or menu items, the menu items editor allows you to do so. Pressing the **Shift Up** or **Shift Down** button will cause the selected menu item to be moved up or down within the list.

☛ Note that submenus and their contained menu items act as a unit when rearranging. When shifting up or down, promoting or demoting, all the menu items within a submenu will move with it.

Inserting and Deleting Menu Items

Pressing the return key is one way of adding a new menu item, although it only does so after the last item in the list. If you wish to insert a new item in between two existing ones, you can press the **Insert** button, and a new empty menu item will appear after the currently selected menu item.

The **Delete** button will similarly delete the currently selected menu item. If it contains submenus, these will be removed as well. Don't worry about the accidental deletion of a large menu, however; you will be warned before such an action is taken.

☛ To completely remove a menu, delete all the menu items from it.

Selecting a Menu Item

When you wish to edit the specific attributes of a menu item, select the appropriate menu item's name in the menu items editor. You can select a new menu item by clicking it with the mouse, using the up and down arrows to reach it, or pressing carriage return. When the item you wish is selected, the attributes in the right pane will display those of the selected menu item.

The specific attributes which can be edited are described below.

Adding a Selector

Most menu items have a selector associated with them. This is the message that will be sent to your application window when the menu item is selected. To set the selector of the item, just type in its name. As with the messages associated with events in subpanes, WindowBuilder Pro will automatically generate matching skeletal methods for you.

There are two special types of menus that may be specified: **Link** menus and **Action** menus. Link menus allow you to specify a window that should be opened when the menu item is selected (Link menus act exactly like LinkButtons). For a complete description of the Link attribute editor, refer to the description of LinkButtons in the *Widget Encyclopedia.*

Action menus act just like ActionButtons and are used to perform simple actions without the need to write Smalltalk code. For example, you could attach the "Cancel" action to the "Exit" menu item. Selecting this menu

item would then automatically close the window. For a complete description of the Action attribute editor, refer to the description of ActionButtons in the *Widget Encyclopedia.*

Adding Mnemonics

In Windows and PM, all menus and menu items can be activated using a combination of **Alt** and some key in the item name.

This *mnemonic* can be set by placing a '~' character before the character you desire in the appropriate menu item's name.

Note that WindowBuilder Pro will check for any conflicts between menu items for you.

☞ Portability note: the character used to signify a mnemonic in PM is the '~' character; in MS Windows, it is the '&' character. Digitalk provides transparent support for the '~' mnemonic in Windows, so it is advisable to use this on both platforms.

Adding Keyboard Accelerators

In addition to mnemonics, Windows and PM provide a more general keyboard option for accessing menu items, called a keyboard accelerator. To edit the keyboard accelerator for the selected item, click in the keyboard accelerator editor with the mouse, then simply press the key you wish to use. The checkboxes **Control**, **Alt**, and **Shift** will toggle off or on to reflect whether you've held down the associated key combination. If you decide you wish to add or remove one of these combination keys, you can either press the new key combination, or change the values in the checkboxes with the mouse.

☞ Most applications add the text for the accelerator into the menu item's name (e.g. **Undo Shift+Backspace**). You needn't add this into the menu item's name — it will be added for you automatically by WindowBuilder Pro.

Adding a Divider

To help delineate different groups of menu items, a horizontal dividing line is often placed between the two groups. You can create a divider simply by adding a new menu item, and pressing the divider checkbox. The menu item text will fill in with dashes to reflect the change.

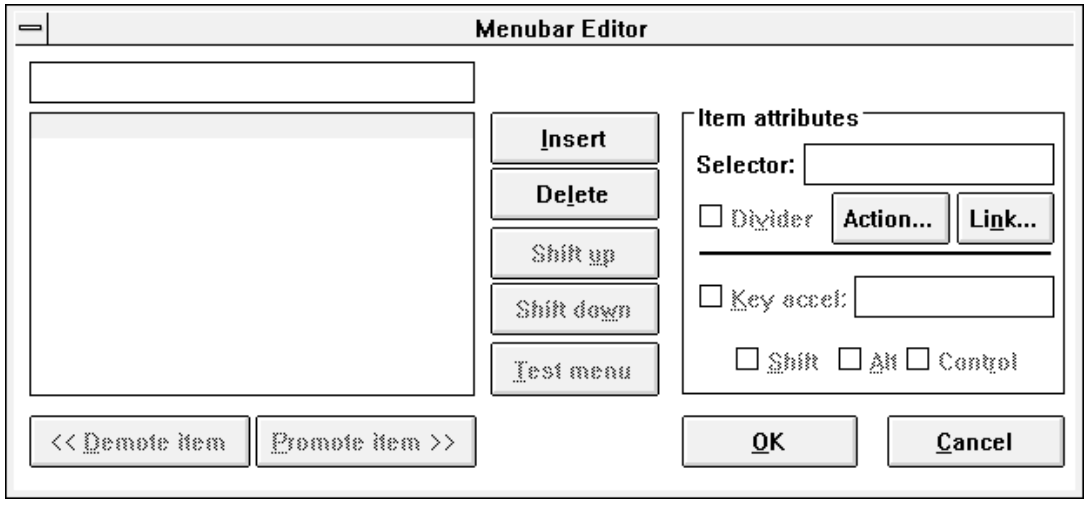
☞ A shortcut to creating dividers is to type a single '-' in a new menu item. The menu item will fill in with dashes, and the divider checkbox will automatically check itself.

Testing a Menu When you want to see a working example of your menu, simply press the **Test Menu** button. A real example of the menu you're working on will pop up in the pane. When you select an item containing a selector, the menu editor will beep to indicate it received the message.

Accepting a Menu When you're finished creating or editing your menu, just press the **OK** button, and the menu editor will be dismissed.

Creating a Menubar

Many windows have a menubar associated with them. Creating a menubar is nearly identical to creating a popup menu (described above); select the window in the layout pane by clicking in it outside any subpanes, then press the **Menu** button. The following dialog will appear:



This editor is almost identical to the popup menu editor, with two minor differences. First, the top level menu items in the menubar editor are the menu bar titles, with their submenu items indented one level.

The other difference between these editors is the behavior of the Test menu button. When pressing this button, a new window will be launched with a working example of your menubar in it.

Other than these differences, the two menu editors are identical. When finished, you need only press **OK** to set up the menubar for the window.

Adding Your Own Subpanes

If you have created your own subpanes, they can be added using WindowBuilder Pro. The **Add** menu provides a submenu called **Custom Panes** in which you can place your own subpanes.

To add a subpane to this menu, execute the **Add Custom Pane…** command from the **Add** menu. A dialog will appear from which you can choose the subpane class you want added. From then on, you will be able to add this custom pane from the **Custom Panes** menu.

If you will be adding a specific custom subpane often, you may wish to augment WindowBuilder Pro's editing capabilities to recognize this subpane class. WindowBuilder Pro has an extensive framework for adding custom functionality; this is described in *Appendix A: Customizing WindowBuilder Pro*.

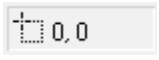
Editing the Initial Window Position

In order to provide a mechanism for easily setting the initial window position of each of the views in a viewmanager, WindowBuilder Pro has the **Set Window Position...** command available from the **Size** menu. Executing this command will cause a rectangle the size of the currently edited window to begin tracking the mouse. When the rectangle is in the position you'd like the window to start at, just click the mouse, and it will be set there.

Note that this command works independently for each top pane in a viewmanager, so it will help you in setting the positions of several windows relative to one another.

If you never set this value, WindowBuilder Pro will automatically cause your window to be centered.

☞ The tiny image inside the position indicator is a button that provides a fast path means of executing the above command.



Sizing, Aligning, and Moving Subpanes

Moving and Sizing Subpanes with the Keyboard

For those situations where you need to make fine adjustments in the layout of your window, you may find your hand too unsteady with the mouse. For such situations, WindowBuilder Pro provides single-pixel moving and sizing capabilities.

If you wish to move the selection in single-pixel increments, select the **Move By Pixel** submenu from the **Align** menu, then select the appropriate direction. Note that the keyboard accelerators for this operation use the directional arrows with the control key.

Similarly, if you wish to size the selection in single-pixel increments, the **Size By Pixel** submenu will provide this functionality. The keyboard accelerators for these commands are the same as **Move By Pixel**, with the shift key held down as well.

Constraining Sizing

When you are resizing a subpane, there are often cases where you really only want to size it in one direction, e.g. you want to make a button wider, but not taller. For such situations, we've provided a constrained dragging feature.

If you hold down the shift key before you drag a selection handle of the selected subpane, the cursor will change to reflect the direction you begin dragging in. You will then be constrained to dragging in this direction, either horizontally or vertically. When you let go of the subpane, the constraints will be removed, and subsequent resizing will not be constrained.

Aligning Subpanes

There are often cases in window building when you'd like to line up several subpanes next to each other. WindowBuilder Pro provides several alignment tools to help you here.

When you select a group of subpanes, and select one of the **Align** menu commands, the selection will be aligned accordingly. For example, if you

select several subpanes, then select the **Align Left** command, the subpanes will be moved to align their left sides.

When aligning **Left**, **Right**, **Top**, or **Bottom**, the first subpane that is selected will be chosen to align relative to. When aligning **Center**, either **Vertically** or **Horizontally**, the subpanes will be centered within the tightest rectangle enclosing them all.

Distributing Subpanes

In addition to aligning panes, there are often cases in building windows where you'd like to place an even amount of space between several subpanes. This is especially useful for setting up a grid of entry fields, or distributing a group of radio buttons. For this purpose, WindowBuilder Pro provides distribution tools.

When you select several subpanes and select one of the **Distribute** commands, the subpanes will be spaced evenly in the indicated direction using the space given between the two outermost subpanes.

Replicating Subpanes' Sizes

In addition to aligning and distributing subpanes, there are often times during window building when you'd like several subpanes to be the same width or height. WindowBuilder Pro provides a replication tool for such situations.

Simply select the subpanes you wish sized, then execute one of the **Replicate Size** commands. The first subpane selected will be used to specify the height or width to be used in replication.

Reframing Automatically

For many applications, you will want to use resizable windows. Often when a window resizes, you would like the subpanes within it to resize or move as well. In Smalltalk/V, this is usually accomplished with a framing block; a block is passed in, with the rectangle of its parent. It is then up to you in this block to calculate the subpane's rectangle based on its parent's rectangle. This is a flexible mechanism, but can be cumbersome to use.

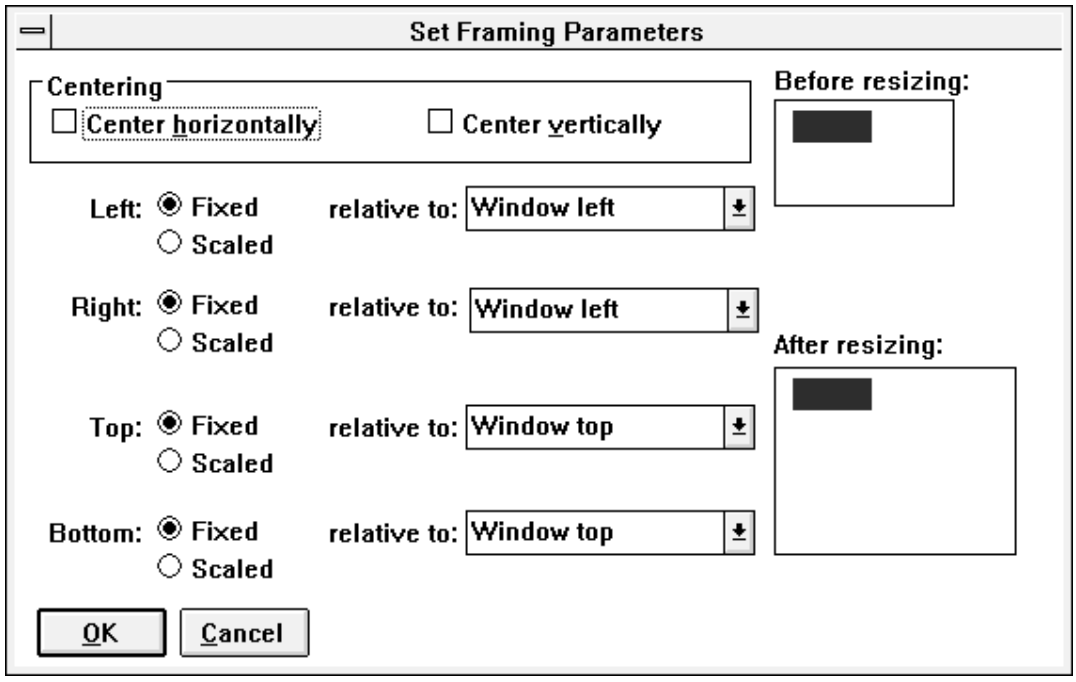
Framing Parameters

WindowBuilder Pro provides a special framing mechanism that automates most situations where you might need a framing block. This mechanism, called *framing parameters*, allows you to directly set the positions of the right, left, top, and bottom of a subpane relative to its enclosing window.

☛ Note that framing parameters are useful only for resizable windows. Since dialogs are not resizable, the **Framing** button is disabled while you are editing dialogs.

Editing Framing Parameters

To set up the framing parameters for a subpane, select it in the layout editor, and press the **Framing** button in the attributes editor. The following window will appear:



The two diagrams on the right provide a visual example of the effect the framing parameters you've set will have on the subpane when it resizes. The top diagram represents the subpane in its window before the window has sized; the bottom diagram represents the subpane in its window after the window has sized.

The top left pane provides simple centering capabilities, either vertically or horizontally.

The bottom left pane allows you to set the framing for all sides of the subpane, potentially completely independent of one another.

In general, the framing parameters of any side of the subpane can be specified independently of all others. If, however, you wish to center the subpane horizontally or vertically, you cannot specify either of the sides with the same orientation (e.g. if you center the subpane vertically, you cannot set the framing parameters for the top or the bottom of the subpane).

To clarify our description, we will describe how framing parameters work for the left side of the subpane as an example; keep in mind that framing parameters function the same for all four sides of the subpane.

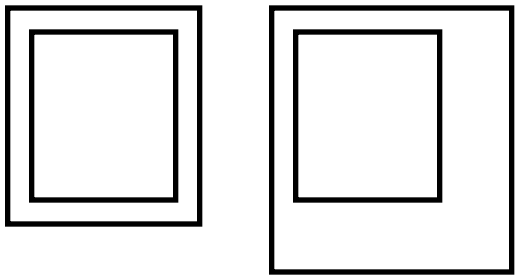
For the left side, you can set whether the coordinate should always be a fixed distance from its window's left side, its window's right side, its window's vertical center, or whether it should be fixed relative to its own right side.

☞ At least one side in both the vertical and horizontal directions must be specified. If, for example, a subpane's left side is fixed relative to its own right side, the framing parameters for the subpane's right side must be fixed somehow relative to its window; otherwise you'd have a pretty confused subpane floating around!

As an alternative to fixing the left side relative to something, you can also set it proportionally; in this case, the left side will always be in the same proportional position within the window, no matter how large or small you size it.

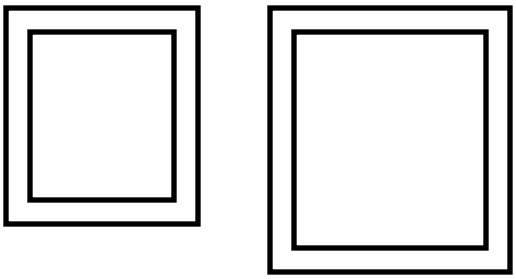
As we mentioned above, this discussion holds true for all four sides of a subpane. Since each side can be specified separately from all other sides, you can create many different possible variations, covering most common framing situations. To help illustrate the possibilities, we've provided the following examples:

Before After



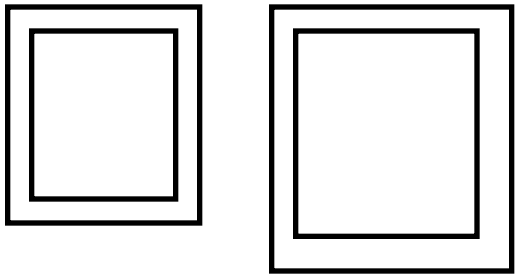
Left relative to window left, top relative to window top, right relative to pane left, bottom relative to pane top

Before After



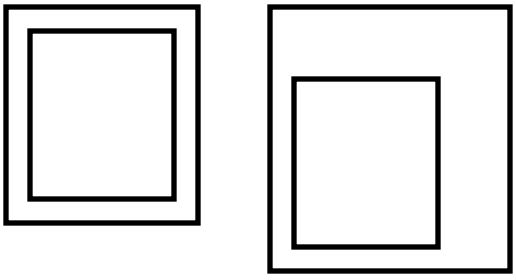
Left relative to window left, top relative to window top, right relative to window right, bottom relative to window bottom

Before After



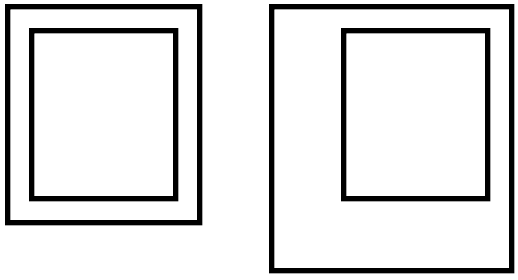
Left relative to window left, top relative to window top, right and bottom proportional

Before After



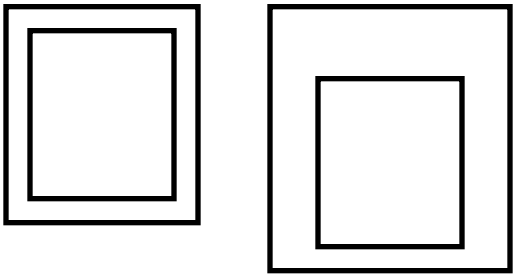
Left relative to window left, right relative to pane left, top relative to pane bottom, bottom relative to window bottom

Before After



Left relative to pane right, right relative to window right, top relative to window top, bottom relative to pane top

Before After



Horizontally centered, bottom relative to window bottom, top relative to pane

Applying Automatic Framing

When you are finished editing your framing parameters, dismiss the framing parameters editor by pressing **OK.** If the framing parameters included centering of any kind, don't be surprised when your subpane centers itself immediately! The framer takes immediate effect; if you resize the window, the subpane will automatically resize with it.

The framing command can set the framing parameters for multiple subpanes at once; add all the desired subpanes to the selection before pressing the **Framing** button. For more information on multiple selection, see *Selecting Multiple Subpanes*.

Autosizing Subpanes

Some subpanes have a "natural" size, i.e. their size doesn't need to be any larger or smaller than a calculatable value. For example, a checkbox has a well-defined size.

It is often useful for these subpanes to have a consistent size when window building. If you are aligning several text labels on the right, you will want them to consistently size to their text size.

To solve this problem, WindowBuilder Pro has an autosize feature. For those subpanes where it might help, executing the **Autosize** command on them will automatically size them to this "natural" size. For all other subpanes, the size will remain the same.

Autosizing may also be turned on or off all the time by selecting the **Auto Size** command from the **Options** menu. StaticText panes will automatically resize based on their style. If they are right justified, they will autosize to the left and vice versa.

Using the Grid

In addition to its rich set of alignment tools, WindowBuilder Pro also offers a snap-to-grid feature for aligning a group of subpanes. When subpanes are moved, sized, or placed, they will snap to the hidden grid.

The grid is always on. By default, the distance (in pixels) between any two grid lines is 4@4. You can change it to anything you like (even turn it off by setting the grid size to 0@0). WindowBuilder Pro will remember your grid settings between sessions.

If you'd like to change the grid size, select the **Grid Size** command from the **Options** menu. A prompter will appear with a point value in it. Set this to a new point, where the x value represents the distance in pixels between vertical grid lines, and the y value represents the distance between horizontal grid lines. You may also quickly set the grid size by right-button clicking on the **Grid** button at the upper right corner of the screen.

If you wish to see the grid visibly, select the **Draw Grid** command from the **Options** menu. Grid visibility may be quickly toggled on or off by clicking on the **Grid** button at the upper right corner of the screen.

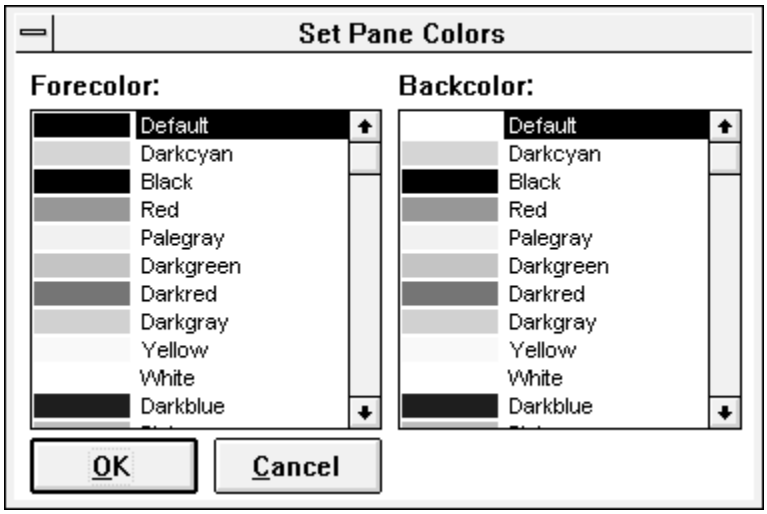
Changing Subpane Appearances

Changing a Subpane's Style

Many subpanes have multiple styles from which you can choose. Static text, for example, can be left, right, or center justified; Buttons can be regular or default buttons. The attributes editors include a **Style** combo box to select the style for the selected subpane. If the subpane in which you're interested has multiple styles, this combobox will contain several choices. Simply select the style desired, and the selected subpane will redisplay with the new style.

Changing a Subpane's Colors

If you want to provide different colors for the foreground and background of a subpane, select it, then press the **Color** button in WindowBuilder Pro. The following dialog will appear:



The set of system-defined colors is available for both the fore- and backcolors of the subpane. If you wish to use the default value for the window, select the color "Default". Press **OK** to accept the new color definition, or **Cancel** to cancel it.

This editing command can set the color for multiple subpanes at once; just add all the desired subpanes to the selection before pressing the **Color** button. For more information on multiple selection, see *Selecting Multiple Subpanes*.

Changing a Subpane's Font

Many subpanes have text associated with them. Under most circumstances, this text can be displayed using a font available in your system.

To change the font of a subpane, select it, then press the **Font** button. The Digitalk font selection dialog will appear, in which you can select the font of your choice. When you dismiss the dialog by pressing the **OK** button, the subpane will be redrawn with the new font.

This editing command can set the font for multiple subpanes at once; add all the desired subpanes to the selection before pressing the **Font** button. For more information on multiple selection, see *Selecting Multiple Subpanes*.

If the font is not a settable attribute for this subpane, the **Font** button will be disabled.

Editing Specific Attributes

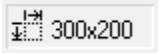
Several subpanes have more specific attributes associated with them. For example, a scrollbar has such extra editable attributes as its page increment and line increment. You can edit these specific attributes by selecting the subpane in the layout pane and pressing the **Attribute** button in the attributes pane. The attributes editor specific to this subpane will appear. Simply set the attributes you want, and press the **OK** button.

If a particular subpane has no specifically editable attributes, the **Attribute** button will be disabled. Each widget supported by WindowBuilder Pro and its associated attribute editor is described in detail in the *Widget Encyclopedia.*

Setting the Window's Size Explicitly

There are times when you may wish to set the size of a subpane or window to a particular value, e.g. to the size of a VGA screen. To do so, select the window or subpane of interest in WindowBuilder Pro, then execute the **Set Window Size…** command from the **Size** menu. A prompter will appear, in which you can enter a point representing the width and height of the selection. When you press **OK**, the selection will resize to reflect the change.

☛ The tiny image inside the size indicator is a button that provides a fast path means of executing the above command.



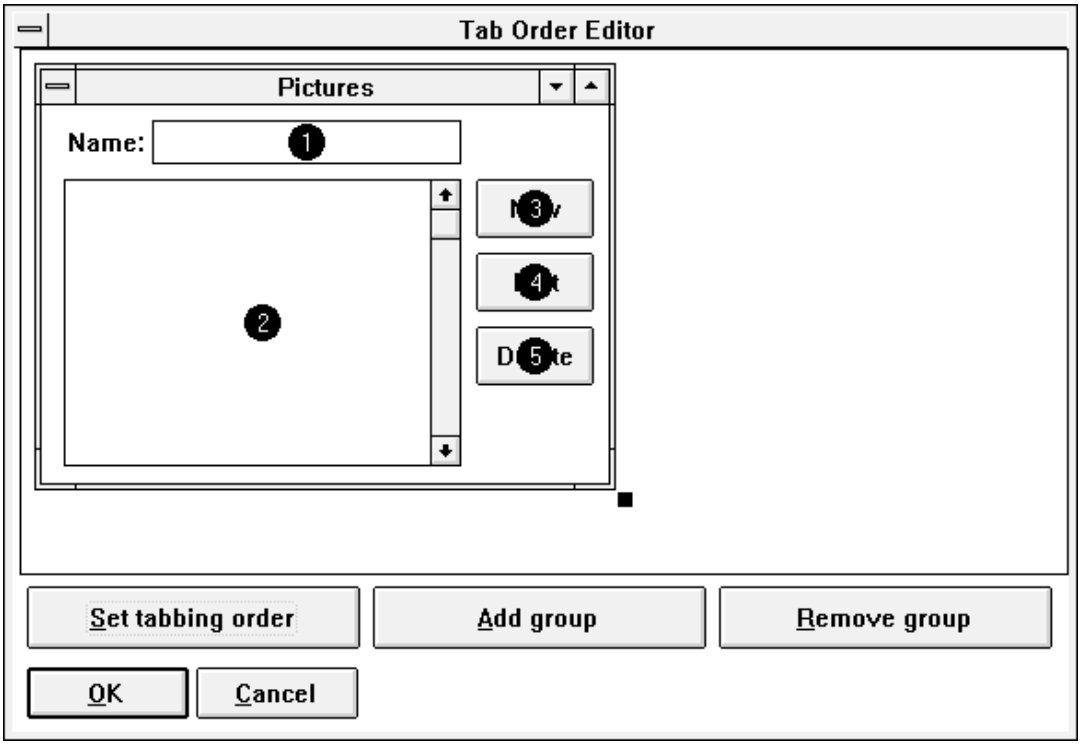
Changing Subpane Relationships & Behaviors

Editing Tabbing Order

Within dialogs, both PM and Windows provide the ability to shift the keyboard input focus from one subpane to the next. Using the tab key to shift the focus forward, and the shift-tab key to shift it back, a user can rapidly enter data without reaching for the mouse. The order in which these subpanes are tabbed to is referred to as the *tabbing order.*

WindowBuilder Pro provides a convenient editor, the Tabbing/Groups

Editor, for specifying tabbing order. To use it, pull down the **Edit** menu, and select the **Edit Tabbing/Groups…** command or click on the **Tab Editor** button in the lower left corner of the screen. The following window will appear:



The top area of this window contains any subpanes you've laid out with WindowBuilder Pro, each labeled with a circled number representing its current tabbing position. To change this order, press the **Set Tabbing Order** button (If there are no subpanes within the window, this will be dimmed. If you haven't placed any subpanes, you should dismiss this dialog, place several, then launch it again).

The buttons in the window will dim to indicate you're now in the modal state of setting the tab order. As you click on each subpane within the top area of this window, you will see a numeric label appear over it, indicating it's been set. The status line at the bottom of the window will count up as you set each subpane, and when you've completed this process, the buttons will undim themselves. Note that some subpanes do not make use of the input focus at all (e.g. static boxes); when you click on one of these, the

system will merely beep.

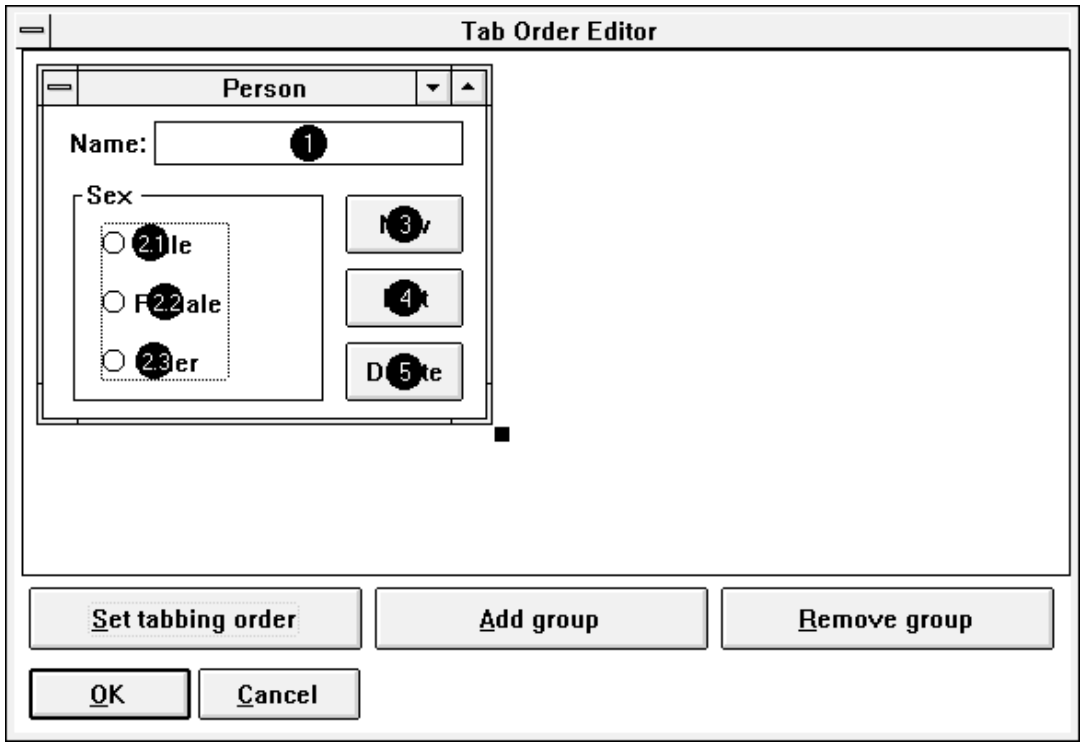
When you're satisfied with the tabbing order, just press the OK button, and the new tab order will be set.

Setting Groups

CUA supports the notion of a *group*. Most commonly used for groups of radio buttons, this construct is used to delineate a group of controls that should be grouped together for tabbing purposes. When you tab into a group, you can then use the arrow keys to move from one element to another within the group. Groups are also used to denote a collection of radio buttons that should be treated as a unit (this is especially useful for auto radio buttons, which turn themselves on and off within their group to ensure that only one button is selected at a time).

To support the grouping feature, the Tab Order/Grouping editor provides a mechanism for specifying the different groups within a window.

To use this editor, select the **Edit Tabbing/Groups…** command from the **Edit** menu. The following window will appear:



This is the same editor used for setting the tab order of a window, but it provides grouping functionality as well. To group a set of controls, press the **Add Tab Group** button. The buttons in the window will dim themselves to indicate you're now in a modal state, ready to group subpanes. To group several controls, click within the top window in an area outside them, drag the mouse across them, and release it. A rectangular box will appear around them, indicating that they're now a group. In addition, the previous tab order will be readjusted, to indicate that these components are part of a group.

If you wish to remove an existing group, press the **Remove Tab Group** button, then click on the group you wish removed. The rectangle around the previously associated subpanes will be removed to reflect the deletion.

Coding With WindowBuilder Pro

Creating applications in Smalltalk/V requires a bit more work than simply

laying them out with WindowBuilder Pro and saving them. You will also need to write some code to tie these windows' behavior to your application.

The following section describes in more detail the coding issues around a WindowBuilder Pro-generated application.

The SubPane Hierarchy

To access and manipulate the state of the subpanes in your application, you will need to send them messages. The programmatic interface of most of these subpanes is described in the *Widget Encyclopedia.* For further information on subpanes provided by Digitalk, refer to Digitalk's documentation.

Accessing Subpanes

Once you have provided a name for a subpane within WindowBuilder Pro, you can access it from within your viewmanager using the message `paneNamed:`. For example, if you have named a checkbox "myCheckBox", and wish to access it within your application, the following code fragment will do what you wish:

```
theCheckBox := self paneNamed: 'myCheckBox'.
```


The `paneNamed:` method should not be confused with Digitalk's `paneAt:` method; the Digitalk approach overloads the meaning of the name of a control (if a `getContents` event occurs, it overrides an existing name), which can yield confusing results. The `paneNamed:` method was introduced to avoid this issue.

Accessing Views

Since the ViewManager class (and WindowBuilder Pro itself) provides you with the ability to have multiple views in a window, you may wish to access a view. WindowBuilder Pro allows you to refer to a view by name, using the `viewNamed:` method. For example, if you named a view 'spellchecker' using the **Name** field of WindowBuilder Pro, the following code fragment would assign it to the variable spellChecker:

```
spellChecker := self viewNamed: 'spellchecker'.
```

Avoiding the createViews Method

The `createViews` (or `open`) method that is generated by WindowBuilder Pro should not be edited. Changes you make in this method will potentially be lost if you re-edit the window definition with WindowBuilder Pro, since WindowBuilder Pro cannot recreate all coding you add to a `createViews` or `open` method. Since there are often cases where you need to manipulate some aspect of your view manager as

it is opening (e.g. initializing subpanes, disabling menu items, adding subpanes dynamically), you may be tempted to alter this method. This is not advisable; it is almost guaranteed that you will wish to change the layout at some point, and it's very easy to lose your changes when you do so.

To help you avoid this temptation, WindowBuilder Pro has added a framework to the ViewManager class that should provide you with all the initialization support you need. In particular, two messages will automatically be sent to your viewmanager as it is being opened, allowing you to execute any code you wish at different phases of initialization. These methods are described below.

The initWindow Method

The **initWindow** message will automatically be sent to your viewmanager after all windows have been created, but before they are visible on the screen. This allows you to initialize any subpanes in ways you require before they are visible.

For example, suppose you wish to set whether a particular menu item in your viewmanager's main window is disabled or enabled, based on the state of an instance variable that was passed in when the window was opened (e.g. with an `openOn:` message). Your `openOn:` method might look like the following:

```
openOn: anObject
    theObject := anObject.
    self open.
```

This squirrels away the object passed in for future use in the instance variable theObject, and then opens the window. In the process of the open method, we take advantage of the `initWindow` method automatically being executed, as follows:

```
initWindow

    (theObject editable) ifFalse: [
        (self menuTitled: 'Object')
            disableItem: 'editObject'.
    ].
```

Here we make use of state information about theObject, disabling the appropriate menu item if necessary. This is of course a small example; your initialization needs will likely be different, but can benefit from this approach as well.

The preInitWindow Method

There are some situations, however, in which the `initWindow` method will not solve your problem, and another method, `preInitWindow`, is required. To understand where these situations might be, and when you would require preInitWindow, we must first explore the process with which windows are created in Smalltalk/V.

Smalltalk/V makes extensive use of the native operating system when it creates and manipulates windows. Subpanes are not just Smalltalk objects; they have native operating system data structures associated with them. When you first create a view with the `open` method, Smalltalk objects exist, but their corresponding Windows or PM data structures have not yet been created.

The last message executed within the open process, `openWindow` (or in some cases, `openModeless`), causes the "real" data structures to be created, after all the Smalltalk subpanes and menus have been added to your viewmanager. Any subpanes you add after these data structures have been created will not be made "real", and subsequently will never appear on the screen. Further, some alterations you make to the state of a subpane will not take effect once the subpane has been created, since Smalltalk cannot switch certain attributes after creation time. An example of this is the style of a subpane; a regular dropDown combobox cannot have its style switched to dropDownList after the combobox has been created (at least not using the Digitalk API).

The `initWindow` method described above occurs after these data structures have been created, and therefore suffers the limitations just mentioned. If you add subpanes within this method, they will never appear. Similarly, if you change the styles of subpanes, these changes will never be apparent.

The `preInitWindow` method was designed for such situations. This method will be executed after all the subpanes have been created as Smalltalk objects, but before they are built into "real" Windows or PM data structures. This allows you to add subpanes dynamically, add style information to subpanes, etc.

Note that since the `preInitWindow` method occurs before the real data structures have been created, you are limited with the sort of initialization you can perform here. Enabling/disabling menu items, for example, is not supported at this phase of the window's life cycle.

Other WindowBuilder Pro Features

Window CUA Keyboard Emulation

Smalltalk/V PM and /V Windows do not provide CUA keyboard support in regular (non dialog) windows. By default, all keyboard input needs to be handled by the programmer. WindowBuilder Pro has added this support to Smalltalk/V, so all windows now enjoy the following keyboard actions:

- the tab key moves the input focus to the next input focus control. If the new focus is a regular button, it will become the default button.
- shift tab moves the input focus to the previous input focus control. If the new focus is a regular button, it will become the default button.
- the up and left arrows select the previous control in the currently selected group.
- the down and right arrows select the next control in the currently selected group.
- if a button has the input focus, pressing the spacebar will cause the button to be pushed.
- if a button has a mnemonic associated with it, pressing the Alt key plus that character will cause the button to be pushed.
- pressing carriage return will cause the default button in the window to be pushed (if there is one).

Mnemonic Redundancy Checking

Following the CUA guidelines, when placing controls within a window, you will often want mnemonics associated with the controls to activate them via the keyboard. Determining whether a particular mnemonic is already taken or not can be a painstaking task.

WindowBuilder Pro now helps avoid this pain by automatically checking whether a particular mnemonic is in use whenever you enter a new one. If it finds one, it will alert you of the problem.

When you set the mnemonics of a control, WindowBuilder Pro will compare it against all the other controls in the window, as well as the main menus of the window's menubar if it has one.

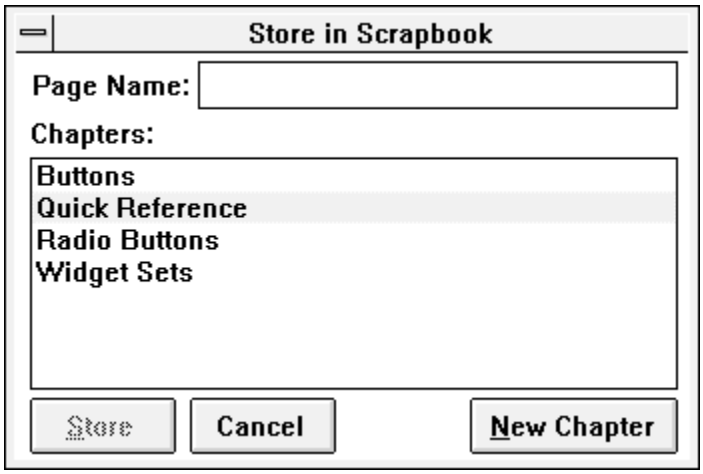
When in the menubar editor, if you set the mnemonic of two menu items within the same menu to the same value, you will be alerted as well.

Using the Scrapbook

The Scrapbook provides a convenient place to store reusable components and standard layouts. It eliminates the need to create the same layout over and over again. If every one of your dialogs requires an "OK & Cancel" button combination, you can store this pair of widgets in the Scrapbook for easy access later. Scrapbooks may be saved to disk and merged together allowing multiple developers to easily share layouts between them.

Storing to the Scrapbook

To store a subpane or a set of subpanes in the Scrapbook, select the items you want to store together and then execute the **Scrapbook** menu, **Store...** command. The following window will appear:



The Scrapbook is organized into Chapters and Pages. A page consists of one entry in the Scrapbook which has one or more items in it (e.g., the "OK & Cancel" combination would be one page). Chapters are made up of multiple pages. A chapter is basically a category. You may have as many chapters as you like and each page may be stored in multiple chapters.

The entryfield at the top of the window is used to record the name of the new page. The listbox below it shows all of the chapters that have been defined. New chapters may be easily added by clicking on the **New**

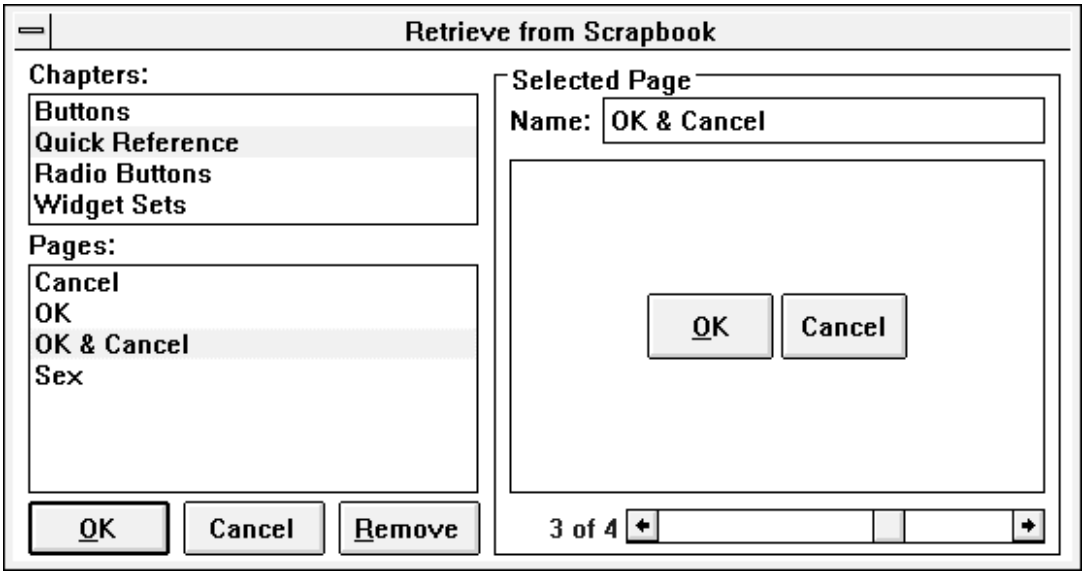
Chapter button. You may select as many chapters as you like in which to store the new page.

☞ The *Quick Reference* chapter is special. Any pages you put in it will appear as cascaded menu items on the **Scrapbook | Quick Reference** menu.

Once you have named the page and specified the chapters, you may store the page by clicking the **Store** button. The **Cancel** button will close the window without saving the page.

Retrieving from the Scrapbook

To retrieve a page from the Scrapbook, execute the **Scrapbook** menu, **Retrieve...** command. The following window will appear:



The listbox in the upper left corner of the window lists all of the defined chapters. The listbox below it shows all of the pages that make up the chapter. The right side of the window contains detail on the currently selected page. Both the name of the page and its image are displayed. In addition to selecting from the listbox, you may also use the scrollbar at the bottom of the window to scroll through each of the pages in the current chapter.

Once you have selected the page that you want to place in the editing win-

dow, click the **OK** button. After the dialog goes away, you may click the mouse button in the editing window to place the objects. Clicking the **Cancel** button in the dialog will close the dialog without loading the cursor.

The **Remove** button is used to remove individual pages or entire chapters (except the Quick Reference chapter that cannot be removed). You will asked to confirm each removal.

Widget Morphing

Morphing allows you to quickly change any widget from one type into another, allowing for powerful "what-if" style visual development. For example, a ListBox instance could be converted into a ComboBox or RadioButtonGroup instance. Common attributes are automatically translated. Attributes not needed by the target class will be lost. Attributes not provided by the source class will be defaulted.

Widgets may be morphed in two ways, either by using the **Edit** menu **Morph...** command or the subpane popup **Morph** command. The second method is more desirable as it provides a list of similar widget types from which to choose (others may be accessed through the **Other...** command).

☛ Warning: some care is needed on your part when morphing a widget into a radically different type. WindowBuilder Pro will map over any attributes the two have in common as well as event handlers for any shared events. You must be careful that the event handlers do, in fact, make sense for the new type. For example, a **getContents** event handler for a listbox would not be appropriate for an entryfield. It is recommended that morphing be limited to similar classes of objects.