Menu Controls

Menu controls must be of type "List or Menu", and be linked to a MENU resource. The MENU resource can be created and edited with ResEdit.

Operation

BaseCt uses Menu Manager routines to initialize and display menus. The MenuHandle associated with each menu control can be found in cHiData (after calling GetCtl), and used with Menu Manager toolbox calls.

Menu controls appear as static text items (with the typical frame and style options), but display pop-up menus when clicked. The pop-up menu is aligned with the left side of the control unless the control's text is right-justified, in which case the menu is aligned with the right side of the control. The menu can be also be popped up either above or below the control (see "Options" below).

Each menu control represents a separate instance of the associated MENU resource (unless the option to use an existing menu is set - see "Options" below). A copy of the linked MENU is made when the control is initialized, and destroyed when the control is disposed of. Also note that a control's menu is not inserted in the menu list until the control is clicked, thereby avoiding conflicts with other instances based on the same MENU resource.

A useful trick which you see us using at the top of this window is to place an untitled, transparent menu control above a static, non-text control. This creates the impression that the static icon or picture itself is a menu control.

Options

The following bit values can be added to VarCode to set various menu control options:

1 = single-selection menu. Forces BaseCt to display the first checked menu item as the control's title, and affects "auto-processing" of menus described below. (If no item is checked, the control title is shown.)

2 = display arrow in title. Displays the downward pointing arrow that Apple recommends using to show presence of a pop-up menu.

4 = display menu "above" the menu control (rather than below it). For single-selection menu controls, the menu is popped-up above the control so that the currently checked item is beneath the cursor. For multi-selection menu controls, the entire menu is shown above the menu control. If this bit is not set, then the menu appears as a "pull-down" menu.

8 = auto-process. For single-selection menus, the selected item is checked and all others in the menu are unchecked. For multi-selection menu controls, the selected item's checked state is toggled. Note that this bit does not affect the normal menu events that get posted when program, standard, or labeled menu items are selected.

16 = use an existing, already initialized, non-main menu (rather than initializing a new instance). This option is useful when a programmer wishes to use a menu in a menu control that has been initialized and is being maintained by the main program.

32 = vertically center the control's title

64 = do not hilite (invert) control when hit

Data Linking

The data linking supported by menu controls is very similar to that described for scrollable lists. The menu control's "value" corresponds to a long integer (32 bits) that can be linked to any program variable (see "Data Links" in the ViewIt Guide for more info on data linking). For single-selection menus, this value is equal to the number of the currently checked menu item. For multi-selection menus, the control's value is equal to the sum of the bit values corresponding to the checked items in the menu. For example, if the first, third, and fifth menu items are checked in a multi-selection menu, then the menu's "value" will equal 1 + 4 + 16 = 21. If no items are checked, then the menu's value = 0 (for both menu types).

Data linking works best when the "auto-process" option is in use since, in this case, it makes sense to think of the menu as having a single "value". On GetVal, for example, an integer value is returned that indicates which items in the menu are checked. On SetVal, the checked state of menu items is updated to reflect the value of the linked program variable.

When the "auto-process" option is not in use, then it makes more sense to treat menu controls as normal menus. In this case "labeled" items are often used in such menus since UtilIt's SetItm2 command makes it easy to manipulate any number of instances of labeled items in any number of menu controls (see the UtilIt Guide for more info).

Limitations

• Do not use menuIDs that are already in use by menus in the main menu bar (a main menu cannot also be used as a non-main menu).
• Any hierarchical menus attached to items in control menus do not get initialized or inserted by BaseCt (so the main program must do this or have the menus auto-initialized by FaceIt).
• Do not use FaceIt's standard "program-wide" menu items in menu controls (Quit, Delete, etc.).
• Never dispose of or release the original MENU resource associated with menu controls.
• When using GetVal or SetVal with multi-selection menus, the maximum number of menu items that can be affected is 32 since the menu's internal "value" is a 4-byte (32-bit) integer. This restriction does not apply to single-selection menus.
• If using SetItm2 to manage the state of menu items in menu controls, you will not be able to pass the res ID or menuID of the linked MENU since a new instance of the MenuHandle is created for each menu control (call GetCtl and pass the MenuHandle in cHiData).
• Miscellaneous: No support for colors other than frame, body & content. No support for hand scrolling.