F5. Palette Menus

QuickDraw pictures can be used as the basis for custom menus. These menus are referred to as picture palettes (but are unrelated to the "color palettes" discussed in the "Initializations" topic). If using FaceIt, then these palettes can be optionally torn off to become floating windows. If not using FaceIt, then picture palettes can only be displayed as pull-down or pop-up menus.

The following discussion assumes that FaceIt is in use, but includes initialization information for palettes that also applies when not using FaceIt (i.e., when using SetItm2 to add a picture palette menu to a non-FaceIt-based program). See the "fDemoXY" program for an example of the use of a picture palette menu in a FaceIt-based program.

Custom MDEF & WDEF

Picture palettes are supported by MDEF 1111 and WDEF 1111 which belong to the UtilIt module. If you don't use such picture palettes in a program, then the MDEF & WDEF (about 10K) are never loaded and are not needed in finished programs. In the future we will integrate floating windows with ViewIt (to support on-line editing of them), but in the meantime the only floating windows currently supported by FaceIt are those created as torn-off picture palette menus.

Palette Events

Regardless of whether a picture palette is in the form of a pull-down menu, pop-up menu, or floating window, your program will see palette choices as simple menu selections that either return control with a menu event or are handled automatically if the palette menu item is a standard item.

Creating Palettes

To create a palette, start by using a drawing or painting program to create a single picture which is to represent the palette. There are absolutely no restrictions on what types of items are drawn in the picture, although you'll typically have one or more groups of mutually exclusive items (such as tools, colors, patterns, etc.) in each picture. Copy and paste the picture as a PICT-type resource into the program file or other resource file opened by the program. Do not mark this PICT resource as purgeable.

The next step is to create MENU and DITL resources that tell UtilIt (MDEF 1111) where individual "items" are in the picture, and how these items are to be hilited, selected, and disabled. The DITL must be associated with the MENU resource by giving it either the same name as the MENU resource or the same resource ID. This choice is provided to help you avoid conflicts with other uses of DITLs.

- Association by Res ID:

MENU 1050 <-> DITL 1050

- Association by Name:

MENU "Tools" <-> DITL "Tools"

The MENU resource will have one item for each item in the palette. For example, you might have a block of 8 tools followed by 8 colors and 4 shades of gray to give a total of 20 menu items. Each palette item can be enabled/disabled, checked/unchecked, have a keyboard equivalent, a label ID, or act as parent items to hierarchical submenus (= options that the UtilIt command SetItm2 supports for palettes).

The linked DITL resource is used to define the positions of the individual palette items. This is done by using ResEdit to add the PICT followed by several user items to the DITL item list. The DITL item list should consist of,

item 1 = PICT resource used to draw the palette
item 2 = "master" user item that defines visible area
item 3 = user item defining location of first menu item
item 4 = user item defining location of second menu item
...

What you should see after building such a DITL is the PICT picture of the palette covered first by one large, "master" user item followed by a number of smaller user items that define the location of each of the palette menu items. The area within the master user item is what will be displayed in the menu or floating window, so arrange your picture and item rectangles to fit within it. Any part of the picture that

hangs outside of the master item will not be drawn, giving you a simple means of clipping away unwanted parts of a picture.

The final step in creating the resources for a palette is to return to the MENU resource and add palette-specific info. First, set the "procID" field to 1111 which is the resource ID of our custom MDEF (do this with ResEdit's "Edit Menu & MDEF ID" dialog). Then set the "Icon#" field of each item to an integer which denotes how you wish that item to appear when hilited, selected (checked), and disabled. The icon# field is most easily edited by opening the MENU resource using ResEdit's "Open Using Template" option.

All hierarchical menu items (items that open submenus) should be assigned an icon# of zero. For non-hierarchical items, setting the icon# of the item to zero (the default) will make it behave like a normal menu item, but this is probably not what you want for most palette items. The integer to place in the icon# field can be determined by adding together one number from each of the four categories in the following table. This table defines all of the hilite, selection, disable, and shape modes currently supported for palette menu items:

Hilite Modes (defined by 1st and 2nd bits of icon# byte)
- 0 = invert rectangle
- 1 = paint frame
- 2 = flash inverted frame (2 pixels thick)
- 3 = flash thin inverted frame (1 pixel thick)

Selection (Checking) Modes (3rd, 4th, and 5th bits)
- 0 = show check mark at left (requires a 12-pixel left margin and works best with Chicago 12 pt. text)
- 4 = paint insetted rectangle (like a radio button)
- 8 = paint frame
- 16 = invert rectangle
- 24 = put "X" in rectangle (like a check box)

Disable Modes (6th and 7th bits)
- 0 = black to gray (erase with gray)
- 32 = paint gray frame
- 64 = erase rectangle (item not displayed)
- 96 = no change (appearance not changed, but not hilited)

Shape Modes (8th bit)
- 0 = rectangle
- 128 = oval (draws ovals instead of rects)

where "rectangle" refers to the entire item rectangle area, "frame" refers to a two-pixel wide rectangle drawn just within the boundaries of the item rectangle, and "hiliting" refers to what happens as the user drags the cursor across items in a menu, or clicks on them in a torn-off window. For example, icon# = 3 + 16 + 0 + 0 = 19 would display a flashing inverted rectangle around the item when hiliting it (3), invert the contents of the item rectangle when checked (selected) (16), and gray the black portions of the item when it is disabled (0).

Installing Palettes

Palettes can be installed as either main (menu bar) or non-main (hierarchical or pop-up) menus. As discussed in "Initializations", the first character of the menu title informs FaceIt and UtilIt how the menu should be installed. All main menu palettes can be torn off, but you can prevent other palette menus from being torn off by using a "-" as the first character of the menu's title instead of a "+".

Managing Palettes

Palette items become selected (checked) and unselected (unchecked) via the toolbox call "CheckItem", and enabled or disabled via the calls "DisableItem" and "EnableItem". The use of these direct toolbox calls works with palettes that are operating as menus since menus get completely redrawn each time they are displayed. When torn off, however, the floating palettes have no idea that such toolbox calls are being made, and do not update their appearance. For this reason we recommend using the UtilIt command SetItm2 to reset both the enabled and checked status of palette menu items since it will ensure that the appearance of items in torn-off palette windows is properly updated.

SetItm2 takes a bit longer to execute than the toolbox calls, so you may need to be smarter about your use of it, minimizing the number of SetItm2 calls. If, for example, the status of a large number of items in a torn-off palette needs to be changed, it may be faster to use toolbox calls to reset the status of the menu items, followed by a single call to SetItm2 with parameter values that force the entire palette to be redrawn (see UtilIt help for details).

Saving Settings

The position and visible status of a palette (where it is and whether it has been torn off) will be saved when the user chooses "Save Settings" if a corresponding DLOG resource has been added to your resource file. This optional DLOG should be given the same resource ID number as the DITL resource used by the palette. Note that FaceIt uses the DLOG resource to save and restore only the top, left position and visible status of the palette, but not its size since that is determined solely by the "master" user item in the DITL.

Palette Titles

The pattern used within the title bar of a torn-off palette is determined by the pattern found in UtilIt's PAT 1111 resource. The patterns that will work properly with the current version of WDEF 1111 include white, light gray (the default), gray, and black. If white is used, parallel horizontal lines are drawn within the title bar.

In some cases you may prefer to have no text shown in the title bar of a torn-off palette. The menu's title will not be displayed in the title bar if either (a) the title is too long, (b) the title is empty, or (c) the title begins with a space character. (For non-main menus, "title" here refers to the menu title after removing the leading "+" or "-".)

Patterns & Colors

One artifact of the way in which MDEF 1111 manages events in torn-off palettes is that QuickDraw patterns within a palette's picture may not be redrawn consistently when parts of the palette are moved, hidden, and reexposed. This only occurs when drawing patterns in a QuickDraw based picture, and can be prevented by making the picture (or the part of the picture with the pattern) bitmap-based.

Although an old-style GrafPort is used to display picture palette menus, color (version 2) pictures can be used with such menus and will be displayed properly. The only thing to be aware of in this case is that the program-wide color palette is used when displaying the picture, meaning that the colors displayed in the palette menu will be limited to the colors in that color palette.

Torn-Off Behavior

Palettes that have been torn off to become floating windows will "float" above all non-palette windows except Desk Accessories in the sense that program windows being driven by FaceWare modules are never brought in front of any palette window. Clicking in the content area of a palette window that is not the frontmost palette results in a menu selection rather than bringing that palette to the front. Clicking in such a palette's title bar, however, will bring it to the front, above all other windows. These rules can be thought of as supporting two "layers" of windows: a palette layer floating above a program window layer, with Desk Accessories scattered throughout (if the DAs were opened in the program's heap). The "active" window that defines the current program context will then be the topmost, visible, non-palette window.

An interesting question we were confronted with was what exactly should happen when an item in a floating palette is clicked. Should the item be instantly "selected" (checked), or just "hilited" until the mouse is released? And if it is just hilited, should the palette behave as a menu or as a dialog when the mouse is moved to another item (in menus the next item is hilited, a dialog only unhilites the original item)? Based on feedback from users and programmers, we chose to make torn-off palette items mimic dialog items.

Errors & Memory

If MDEF 1111 detects any error in the way that you have set up a palette, it beeps twice and quits to the

Finder. This will almost always occur at launch time when program menus are being auto-installed, but may also occur when your program installs its own palette menus independently of FaceIt.

If you do dynamically load and install palette menus, note that a form of the SetItm2 command (b < 0, see UtilIt help) must be used in place of InsertMenu, DeleteMenu, and DisposMenu since the latter toolbox calls alone will not deal properly with the palette windows or other private data structures allocated by the custom MDEF and WDEF.