Description

This is version 1.0b3 of Popup CDEF. It fixes some bugs and has slightly better support for menus created by the application. This document is also new with version 1.0b3, and a new application is provided to demonstrate the different types of popup menus that can be created with this CDEF.

This CDEF implements a popup menu control. The CDEF handles display of the menu's title, the current selection, the one pixel drop shadow, and the down arrow at the end of the menu. It also handles tracking of the mouse and checking and unchecking of the current item. It is compatible with systems 6.0.5 and 7.0.

The CDEF is modeled after the popup CDEF provided by Apple in System 7.0 and described in IM-VI, p3-16 to 3-19. If you are already familiar with that CDEF then using this CDEF will be very simple. Additional support is provided for type-in popup menus and for menus created dynamically by the application.

Using the CDEF

This section describes how you can use the popup CDEF in your own applications. This section was adapted from IM-VI, p3-16 to 3-19.

Adding the CDEF to your Application

You'll need to use ResEdit, or some other resource editing utility, to copy the 'CDEF' resource with ID 129 from the file "PopupCDEF" to your application. The file "PopupCDEF.h" contains definitions of constants and types you can use to create and access the popup CDEF. The procID code to pass to NewControl is computed by multiplying the CDEF's resource ID by 16, and then adding any variation codes. The basic popup CDEF would have a procID of 16*129 = 2064. The constant kPopupProcID is defined as 2064 in the file "PopupCDEF.h". A popup control with the popupUseWFont variation code would have a procID of 16*129 + 8 = 2072. If you already have a CDEF with the same ID in your application, then you can change the ID of the popup CDEF resource, but you'll have to calculate the procID codes for your controls using the new resource ID.

The CDEF automatically handles clicks in the menu. When the user clicks in the current selection box, the Control Manager routine FindControl will report a click in the control. The application should then call TrackControl with an actionProc parameter of -1. The CDEF will hilite the control's title and will call PopupMenuSelect to display the popup menu and allow the user to select an item from the menu. Once the user has selected an item the CDEF will set the control's value to the selected item and will place a check mark next to the selected menu item.

You can use the Control Manager routines GetCtlValue to determine which menu item is currently selected, and SetCtlValue to set the currently selected menu item. You can determine the number of items in the menu using GetCtlMax. You should not call SetCtlMin or SetCtlMax, as these values are maintained by the CDEF.

The file "PopupCDEF-demo.c" contains the source code to an application that demonstrates the different types of popup menus that can be created with this CDEF. The file also contains some code snippets that illustrate how you can support type-in popup menus in your own applications.

Creating a Popup Menu Control

You can use NewControl to create a popup menu control, or you can use a 'CNTL' resource and allow the Dialog Manager to create the popup control in your dialogs. To create a control using the CDEF, you must use a procID of kPopupProcID, and then add any desired variation codes. The value, min, max, and refCon parameters to NewControl have special meanings when creating a popup menu control.

Value Parameter

The value parameter specifies the style in which to draw the popup's title. The value parameter can be any combination of the following values (defined in <Controls.h>):

popupTitleLeftJust	0x0000	left aligned text
popupTitleCenterJust	0x0001	ignored
popupTitleRightJust	0x00FF	right aligned text
popupTitleBold	0x0100	bold text
popupTitleItalic	0x0200	italic text
popupTitleUnderline	0x0400	underlined text
popupTitleOutline	0x0800	outlined text
popupTitleShadow	0x1000	shadow text
popupTitleCondense	0x2000	condensed text
popupTitleExtend	0x4000	extended text
popupTitleNoStyle	0x8000	unstyled text

If you use popupTitleRightJust, then the popup's title is drawn to the right of the popup box instead of to its left. The position of the down arrow is also reversed in the popup box. After the control has been created, the value parameter is used to determine the currently selected item. Initially, the first item in the menu is selected.

Min Parameter

The min parameter specifies the resource ID of the menu in the popup menu control. The CDEF calls GetResource('MENU', id), where id is the value of the min paremeter, to determine if the menu has already been loaded by GetMenu. If it has been loaded by GetMenu, then the handle returned by GetResource is used for the control's menu handle. Otherwise, the CDEF calls GetMenu (id)to load the menu and uses the handle returned by GetMenu for the control's menu handle. If, however, the menu is not in a resource file, then the CDEF calls GetMHandle(id), and uses the handle returned as the control's menu handle. This allows the application to dynamically create a menu for use by the control, though the application must insert the menu into the menu list before the control is created. If the CDEF used GetMenu to load the menu handle, then it uses ReleaseResource when the control is disposed of to release the memory used by the menu handle. Otherwise, it is the application's responsibility to release the memory used by the menu handle. After the control has been created, the min parameter is set to 1.

Max Parameter

The max parameter determines the width of the control's title area. Normally, you'll pass zero for the max parameter, allowing the title area to be resized dynamically to fit the title string. If you require a fixed width for the title, you can use the max parameter to set its width. If the max parameter is less than the minimum width of the title area, then the max parameter is ignored and the minimum width is used instead. The minimum width is determined by the size of the font used to display the title. After the control has been created, the max parameter is set to the number of items in the popup menu.

Variation Codes

The procID parameter should contain the resource ID of the popup CDEF multiplied by 16, plus any desired variation code. The variation code can be any combination of the following:

popupFixedWidth	0x0001	fixed-width control
popupTypeIn	0x0002	type-in style popup menu
popupUseAddResMenu	0x0004	use AddResMenu and refCon parameter
popupUseWFont	0x0008	use window's font and font size

popupFixedWidth

PopupFixedWidth makes the control fill the entire control rectangle assigned to it. Without this variation code, the control is resized dynamically to take up only as much space as is needed to display the menu and its title. PopupFixedWidth is especially useful when you want to align the left and right edges of several popup menus. When used in combination with a fixed width for the control titles (specified in the "max" parameter to NewControl), you can arrange several popup menus to be aligned at both their left and right edges and to have their titles aligned. This usually results in a more aesthetically pleasing arrangement for the controls.

popupTypeIn

Used for type-in popup menus, as described in IM-VI, 2-37. With this variation code, only the down arrow and the one pixel border and drop shadow are drawn. The control's title and current selection are not drawn. The application must hilite and unhilite the type-in field and must handle insertion and deletion of the menu item containing a value not already in the popup menu. The popupTypeIn variation code is defined in the file PopupCDEF.h.

popupUseAddResMenu

Get menu items from a resource. The CDEF interprets the refCon parameter as a value of type ResType that specifies the resource type to load into the menu using the AddResMenu procedure. For instance, to provide a popup menu listing available fonts, you would use the variation code popupUseAddResMenu and set the refCon parameter to 'FONT'. The control's refCon field is available for the application's use once the control has been created, or if the popupUseAddResMenu variation code isn't used.

popupUseWFont

Uses the font of the window containing the control instead of the system font. The control's title and current selection are drawn using the font and size of the window. The menu, when selected by the user, is also drawn in the window's font and font size.

Disabling a Menu Item

The entire control is drawn in a grayed out pattern when the control is disabled by setting its hilite value to 255 with HiliteControl. In addition, the current selection rectangle is displayed in a grayed out pattern when the menu item corresponding to the currently selected item is disabled. If you disable (or enable) the menu item after the current item has been selected, then you'll need to redraw the control by using InvalRect on the control's rectangle to generate an activate event. You can also redraw the control by using any Control Manager routine that will redraw the control.

Accessing the Menu Handle

You can access the menu handle and menu ID of the menu associated with the control by dereferencing the contrlData field of the control record. The contrlData field is a handle to a block of private information used by the CDEF. The first few bytes of this block contain the following structure:

```
/* This is the same structure as the popupPrivateData record described in IM-VI, p3-19. By making
this the first element in the popup structure, a CDEF based on this popup library is made more
compatible with Apple's CDEF. */
typedef struct {
    MenuHandle    mHandle;
    short         mID;
} PopupPrivateType, *PopupPrivatePtr, **PopupPrivateHandle;
```

This structure is defined in the file PopupCDEF.h.

Differences from the System 7.0 CDEF

This section is intended for people who are already using the popup CDEF provided with System 7.0 or who are trying to decide which popup CDEF to use. While this popup CDEF is similar to the popup CDEF provided with System 7.0, the two CDEFs are not identical. Following is a list of the most important differences between the two CDEFs.

Extensions to the System 7.0 CDEF

An additional variation code is defined, popupTypeIn. With this variation code, only the popup's down arrow is drawn, the popup's title and current selection are not drawn. You can use this variation code to support type-in popup menus. Type-in popup menus are described in IM-VI, p2-37.

To support menus created dynamically by the application, if the menu isn't in the application's resource file, then the menu is searched for in the menu list using GetMHandle. For this feature to work, the menu must not be in any resource file open at the time the popup control is created, and the menu must be inserted into the menu list before the popup control is created.

Features in the System 7.0 CDEF that are not Supported

The following features of the System 7.0 popup CDEF are not supported by this CDEF:

popupTitleCenterJust	ignored
Color QuickDraw	drawn using original QuickDraw

Other Differences

The System 7.0 popup CDEF will only display the control's title if a specific title width is specified (in the "max" parameter to NewControl) when the control is created. My popup CDEF will always show the title, unless the control has no title or the popupTypeIn variation code is used. Specifying a specific width for the title still works for my popup CDEF, and can be used when you want to align several popup menus.

The appearance of popup menus drawn with this CDEF may vary slightly from the appearance of popup menus drawn with the System 7.0 popup CDEF. Except for the differences mentioned above, the variations are very minor, and shouldn't be noticed by most users.

Things To Do

This section lists things that need to be fixed in the popup CDEF. The items are listed in approximate order of urgency. Most of these things need to be fixed in the file "PopupLib.c".

Color support needs to be added. Without a color Macintosh there's no way for me to test color code, so this popup just uses the original QuickDraw. For color support, the current selection should be drawn in the same color as the menu item; when the menu definition function is called to draw the item the color will be set correctly, but the function PopupDrawSelectionString should also set the menu's colors. The CopyBits code must be adapted to use CopyPixMap. Finally, The PopupHilite routine should set HiliteMode so that color hiliting is done correctly.

This CDEF is not as Script Manager aware as it should be. Normally, the menu definition function is used to draw the selected item. This ensures that the current selection appears as close as possible to the actual menu item. However, if the item selected from the menu is wider than the width of the current selection rectangle, then it is drawn using the function PopupDrawSelectionString. This is done so that the string will be truncated properly. The text is drawn in the correct font style, but a menu item's script code is ignored (font menus may have script codes so that they're drawn in the correct font).

Reduced icons are not drawn in the current selection box. Small icons and regular icons are drawn correctly. I'm not sure why reduced icons aren't drawn, but it is not a major problem since reduced icons are not used very often.

A minor allignment problem may occur when the text style of the popup's title results in a line height different from the line height of the plain system font (or, if the wfont attribute is true, then different from the height of the window's font). For instance, if the title's style is outline+underline+bold, its height may differ by several pixels from the height currently calculated.

Credits

Eric Bowman (bobo@reed.edu) suggested and helped me add a call to GetMHandle to allow use of menus created by the application that are not in the resource file. He also pointed out the use of a handle that had already been disposed of.

How to Contact Me

You can reach me at the following address:

Ari Halberstadt
9 Whittemore Road
Newton, MA 02158-2105
USA

ari@world.std.com