Technote 1148

Dialog Manager Helper Functions

By C.K. Haun
Revised by Mark Cookson
Apple Worldwide Developer Technical Support

CONTENTS

This Technote discusses Dialog Manager calls available since System 7.0 which can ease the work of managing dialogs. They allow you to call on the services of the System to track the mouse cursor (i.e., change to and from the I-beam cursor) and handle the standard keystrokes for accept and cancel in your dialog.

Introduction

With the introduction of System 7.0 some new Dialog Manager calls were added to make creating standard dialog functionality simpler for the developer.

The functionality that has been simplified is:

- Setting the default button (aliases the return and enter keys and draws the default ring around it)
- Setting the cancel button (aliases to the esc and command-. keys)
- Tracking the cursor for changes to and from the cursor and I-beam

System 7 Dialog Manager Call Interfaces

The new calls are

```
/* These are copied from Universal Headers 3.1 */

EXTERN_API( Boolean )
StdFilterProc                   (DialogPtr              theDialog,
                                 EventRecord *          event,
                                 DialogItemIndex *      itemHit);

EXTERN_API( OSErr )
GetStdFilterProc                (ModalFilterUPP *       theProc);

EXTERN_API( OSErr )
SetDialogDefaultItem            (DialogPtr              theDialog,
                                 DialogItemIndex        newItem);

EXTERN_API( OSErr )
SetDialogCancelItem             (DialogPtr              theDialog,
                                 DialogItemIndex        newItem);

EXTERN_API( OSErr )
SetDialogTracksCursor           (DialogPtr              theDialog,
                                 Boolean                tracks);
```

`SetDialogDefaultItem` sets the default button, the one that responds to the Return and Enter key. It also puts the default ring around the button.

`SetDialogCancelItem` sets the cancel button, which can be the same as the default button. The cancel button responds to the esc key and the command-period (or sequence-that-produces-the-period).

`SetDialogTracksCursor` tells the Dialog Manager if it should track the cursor to change it from the arrow cursor to the I-beam cursor (or vice-versa) when it goes into (or out of) a text edit field.

The `GetStdFilterProc` and `StdFilterProc` calls work together allowing you to delegate most of the work of managing a dialog back to the Dialog Manager. You call the `StdFilterProc` function whenever you want the default behavior of the Dialog Manager.

NOTE: You must call the standard filter proc for these calls to work properly. Automatic cursor tracking, default button bordering, and keystroke aliasing for **OK** and **Cancel** will only be active if you call the standard filter procedure. Also, these calls are System 7 specific. You cannot use them in previous system versions.

Using the calls

Using these calls requires a little preparation on your part. After you create your dialog, you need to tell the Dialog Manager which items you want as the default and cancel items. The button selected as the cancel item will be toggled by the Escape key or by a Command-. keypress. The button specified as the default will be toggled by the Return or Enter key, and also will have the standard heavy black border drawn around it. The buttons will also be hilited when the correct key is hit.

`SetDialogTracksCursor` tells the Dialog Manager that it should set the cursor on behalf of your application according to what part of the dialog the mouse is hovering over. When you pass a '`true`' value to the `SetDialogTracksCursor` call the Dialog Manager will constantly check cursor position in your dialog, and change the cursor to an I-Beam when the cursor is over an edit line. Otherwise, Dialog Manager sets the cursor to the standard arrow.

Here is a snippet of code showing how to call the functions described in this Technote:

```
/* Before we go into a ModalDialog loop, do a little preparation */
    ModalFilterUPP              filterProcUPP;

    myDialogPtr = GetNewDialog (kMyDialogID, nil, (WindowPtr)-1);

    /* Tell the Dialog Manager that the OK button is the default */
    myErr = SetDialogDefaultItem (myDialogPtr, ok);

    /* Tell the Dialog Manager the cancel button is the cancel item*/
    myErr = SetDialogCancelItem (myDialogPtr, cancel);

    /* We have an edit item in our dialog, so tell the Dialog Manager
       to change the cursor to an I-Beam when it's over the edit line */
    myErr = SetDialogTracksCursor (myDialogPtr, true);

    filterProcUPP = NewModalFilterProc (ModalDialogFilter);

    do {
        ModalDialog (filterProcUPP, &hitItem);
        switch (hitItem) {
            case ...:
                break;
            case ...:
                break;
            default:
        }
    } while (hitItem != ok && hitItem != cancel);

    DisposeRoutineDescriptor (filterProcUPP);
```

Your modal dialog filter will look something like this:

```
Boolean ModalDialogFilter (DialogPtr theDialog,
     EventRecord *theEvent, short *itemHit) {
          Boolean                    result          = false;
          OSErr                      err             = noErr;
          ModalFilterUPP             standardProc;

    if ((theEvent->what == updateEvt) &&
       (WindowPtr)theEvent->message != theDialog) {
        err = DispatchWindowUpdate ((WindowPtr)theEvent->message);
    } else if ((theEvent->what == activateEvt) &&
       (WindowPtr)theEvent->message != theDialog) {
        DoActivate (theEvent, true);
    } else {
        err = GetStdFilterProc (&standardProc);
        if (err == noErr) {
            result = CallModalFilterProc (standardProc, theDialog, theEvent, itemHit);
        }
    }

    return result;
}
```

Conclusion

By using these Dialog Manager calls (even when you're not using a filter) you give your dialog a more consistent user interface and can save yourself a fair amount of work over doing it all yourself.

Further References

- Inside Macintosh:Macintosh Toolbox Essentials, Chapter 2 - Event Manager
- Inside Macintosh:Macintosh Toolbox Essentials, Chapter 4 - Window Manager
- Inside Macintosh:Macintosh Toolbox Essentials, Chapter 6 - Dialog Manager

Downloadables

 Acrobat version of this Note (how many K?)

Change History

- Originally written in October 1991, as Technote TB 37 -- Title by C.K. Haun.
- Accompanying code written and revised by C.K. Haun (1991) and Mark Cookson (1999).
- In January 1999, this Technote was updated to better organize the ideas presented.

Acknowledgments

Thanks to Pete Gontier.