NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note TB27
User Items in Dialogs

CONTENTS

The Dialog Manager does not go into detail about how to manage user items in dialogs; this Technical Note describes the process.

[May 01 1985]

Introduction

To use a `userItem` with the Dialog Manager, you must define a dialog, load the dialog and install your `userItem`, and respond to events which relate to your `userItem`. If your application wants to receive mouse clicks in the `userItem`, then you must set the item to `enabled`.

Back to top

Defining a Dialog Box with a userItem

You should define the dialog box in your resource file as follows. Note that it is defined as invisible, since we have to play with the `userItem` before we can draw it.

```
resource 'DLOG' (1001) {            /* type/ID for box */
   {100,100,300,400},            /* rectangle for window */
   dBoxProc, invisible, noGoAway, 0x0,    /* note it is invisible */
   1001,
   "Test Dialog"
};

resource 'DITL' (1001) {            /* matching item list */
   {
      {160, 190, 180, 280},            /* rectangle for button */
         button { enabled, "OK" };        /* an OK button */
      {104, 144, 120, 296},            /* rectangle for item */
         userItem { enabled }        /* a user item! */
   }
```

Back to top

Loading and Preparing to Show the Dialog Box

Before we can actually show the dialog box to the user, we need two support routines. The Dialog Manager calls the first procedure whenever we need to draw our `userItem`. You should install it (as shown below) after calling `_GetNewDialog` but before calling `_ShowWindow`. This first procedure simply draws the `userItem`.

Back to top

In MPW Pascal:

```
PROCEDURE MyDraw(theDialog: DialogPtr; theItem: INTEGER);

       VAR
          iType : INTEGER;            {returned item type}
          iBox  : Rect;                {returned bounds rect}
          iHdl  : Handle;            {returned item handle}

       BEGIN
          GetDItem(theDialog,theItem,iType,iHdl,iBox); {get the box}
          FillRect(iBox,ltGray);        {fill with light gray}
          FrameRect(iBox);            {frame it}
```

Back to top

In MPW C 3.0:

```
    pascal void MyDraw(theDialog,theItem)
    DialogPtr    theDialog;
    short int    theItem;

    {
        short int    iType;            /*returned item type*/
        Rect        iBox;            /*returned bounds rect*/
        Handle        iHdl;            /*returned item handle*/

        GetDItem(theDialog,theItem,&iType,&iHdl,&iBox); /*get the box*/
        FillRect(&iBox,qd.ltGray);        /*fill with light gray*/
        FrameRect(&iBox);            /*frame it*/
```

The other necessary procedure is a filter procedure (filterProc) that the Dialog Manager calls whenever _ModalDialog receives an event (this only applies when calling _ModalDialog; modeless dialogs are covered below). The default filterProc looks for key-down and auto-key events and simulates pressing the OK button (or whatever else is item 1) if the user has pressed either the Return key or the Enter key. To support a userItem, the filterProc must handle events for any userItem items in the dialog in addition to performing the default filterProc tasks. The following short filterProc supports these types of items; when the user clicks in the userItem, the filterProc inverts it.

Back to top

In MPW Pascal:

```
    FUNCTION MyFilter(theDialog: DialogPtr; VAR theEvent: EventRecord;
                      VAR itemHit: INTEGER): BOOLEAN;
       CONST
          enterKey    = 3;
          returnKey   = 13;

       VAR
          mouseLoc : Point;            {we'll play w/ mouse}
          key      : SignedByte;        {for enter/return}
          iBox     : Rect;            {returned boundsrect}
          iHdl     : Handle;            {returned item handle}
          iType, itemHit : INTEGER;        {returned item and type}

       BEGIN
          SetPort(theDialog);
          MyFilter := FALSE;            {assume not our event}
          CASE theEvent.what OF            {which event?}
             keyDown,autoKey: BEGIN        {he hit a key}
                key := SignedByte(event.message); {get keycode}
                IF (key = enterKey) OR (key = returnKey ) THEN BEGIN
                   MyFilter := TRUE;        {we handled it}
                   itemHit := 1;        {he hit the 1st item}
                END;                {test CR or Enter}
             END;    {keydown}
             mouseDown: BEGIN            {he clicked}
                mouseLoc := theEvent.where;    {get the mouse pos'n}
                GlobalToLocal(mouseLoc);    {convert to local}
                GetDItem(theDialog,2,iType,iHdl,iBox); {get our box}
                IF PtInRect(mouseLoc,iBox) THEN BEGIN {he hit our item}
                   InvertRect(iBox);
                   MyFilter := TRUE;        {we handled it}
                   itemHit := 2;        {he hit the userItem}
                END;                {if he hit our userItem}
             END;                {mousedown}
          END;                    {event case}
```

Back to top

In MPW C 3.0:

```
pascal Boolean MyFilter(theDialog,theEvent,itemHit)
DialogPtr    theDialog;
EventRecord    *theEvent;
short int    *itemHit;

#define enterKey    3;            /*the enter key*/
#define returnKey    13;            /*the return key*/

{
    char        key;            /*for enter/return*/
    short int    iType;            /*returned item type*/
    Rect        iBox;            /*returned boundsrect*/
    Handle        iHdl;            /*returned item handle*/
    Point        mouseLoc;        /*we'll play w/ mouse*/

    SetPort(theDialog);
    switch (theEvent->what)        /*which event?*/
    {

        case keyDown:
        case autoKey:            /*he hit a key*/
            key = theEvent->message; /*get ascii code*/
            if ((key == enterKey) || (key == returnKey))
            {            /*he hit CR or Enter*/
                *itemHit = 1;    /*he hit the 1st item*/
                return(true);    /*we handled it*/
            } /*he hit CR or enter*/
            break;        /* case keydown, case autoKey */
        case mouseDown:    /*he clicked*/
            mouseLoc = theEvent->where;    /*get mouse pos'n*/
            GlobalToLocal(&mouseLoc);    /*convert to local*/

            /*get our box*/
            GetDItem(theDialog,2,&iType,&iHdl,&iBox);
            if (PtInRect(mouseLoc,&iBox))
            {            /*he hit our item*/
                InvertRect(&iBox);
                *itemHit = 2;    /*he hit the userItem*/
                return(true);    /*we handled it*/
            } /*if he hit our userItem*/
            break; /*case mouseDown */
    } /*event switch*/
    return(false); /* we're still here, so return false
                (we didn't handle the event) */
```

Back to top

Invoking the Dialog Box

When we need this dialog box, we load it into memory as follows:

Back to top

In MPW Pascal:

```
PROCEDURE DoOurDialog;

   VAR
      myDialog : DialogPtr;    {the dialog pointer}
      iType, itemHit : INTEGER;    {returned item type}
      iBox     : Rect;    {returned boundsRect}
      iHdl     : Handle;    {returned item Handle}

   BEGIN
      myDialog := GetNewDialog(1001,nil,POINTER(-1)); {get the box}
      GetDItem(myDialog,2,iType,iHdl,iBox); {2 is the item number}
      SetDItem(myDialog,2,iType,@myDraw,iBox); {install draw proc}
      ShowWindow(theDialog);    {make it visible}
      REPEAT
         ModalDialog(@MyFilter, itemHit ); {let dialog manager run it}
      UNTIL itemHit = 1;    {until he hits ok.}
      DisposDialog(myDialog);    {throw it away}
```

Back to top

In MPW C 3.0:

```
    void DoOurDialog()
    {

        DialogPtr    myDialog;    /*the dialog pointer*/
        short int    iType;        /*returned item type*/
        short int    itemHit;    /*returned from ModalDialog*/
        Rect        iBox;        /*returned boundsRect*/
        Handle        iHdl;        /*returned item Handle*/

        myDialog = GetNewDialog(1001,nil,(WindowPtr)-1); /*get the box*/

        /*2 is the item number*/
        GetDItem(myDialog,2,&iType,&iHdl,&iBox);

        /*install draw proc*/
        SetDItem(myDialog,2,iType,MyDraw,&iBox);

        ShowWindow(myDialog);    /*make it visible*/

        while (itemHit != 1) ModalDialog(MyFilter, &itemHit);
        DisposDialog(myDialog);    /*throw it away*/
```

Back to top

Using userItem Items with Modeless Dialogs

If you are using `userItem` items in modeless dialog box, the Dialog Manager will call the draw procedure when `_DialogSelect` receives an update event for the dialog box. When the user clicks on your `userItem` and it is `enabled`, `_DialogSelect` will return `TRUE`. The `itemHit` will be equal to the item number of your `userItem`. Your code can then handle this like the mouse-down event case in the example above.

Back to top

References

Inside Macintosh , The Dialog Manager

Back to top

Downloadables

Acrobat version of this Note (K). Download