Using the List Manager in a Modal Dialog

```
/*
	Using the List Manager in a Modal Dialog
	This is an example that shows how to put up a modal dialog that contains a
	user item (in this case, a List Manager list).  In this example the modal
	dialog contains 3 items: item 1 is an OK button, item 2 is the user item
	list, item 3 is a user item used to put the "default button border" around
	the OK button.  If a cell of the list is clicked, the contents of the cell
	will be displayed in a separate window.

	This is a good demonstration on how to use the List Manager, how to display
	user items in a dialog, and how to use a dialog filter proc.
*/

// Assumes inclusion of <MacHeaders>
#include <stdio.h>
#include <string.h>


// Some global constants for Items hit in this simple little dialog
#define LISTITEM	2
#define OKITEM		1
#define DUMMYITEM		3		// used so that one can set up the default item.

// Resource ID's
#define DIALOGID	128

// callback procedures used from within the dialog manager
pascal void UserProc(WindowPtr theDialog, short theItem);
pascal void ButtonProc(WindowPtr theDialog,short theItem);
pascal Boolean myDlgFilter (DialogPtr theDialog, EventRecord *theEvent,
						short *itemHit);

// Global handle to list in dialog box
ListHandle theList;

main()
{
	DialogPtr		theDialog;  // the dialog

	// Variables used in GetDItem and SetDItem
	short			type;
	Handle	theHandle;
	Rect			iRect;

	short	itemHit;		// Item hit returned from ModalDialog
	Point	theCell;		// Used to get which cell in list the user selected
	char	s[90]; // String that cell contains
	short	len;			// length of string
	WindowPtr	theWindow;	// Window to display string from cell
	Rect	aRect;

	// As always initialize the macintosh toolbox
	InitGraf(&thePort);
	InitFonts();
	InitWindows();
	InitMenus();
	TEInit();
	InitDialogs(nil);
	InitCursor();
	MaxApplZone();

	// Get Dialog and window from resource file
	theDialog = GetNewDialog(DIALOGID,nil,(WindowPtr)-1);
	SetRect(&aRect,300,60,500,120);
	theWindow = NewWindow(nil,&aRect,"\p",TRUE,2,
				nil,FALSE,0);

	// Grab the information for the LISTITEM using GetDItem,
	// then using SetDItem, set up a procedure to draw the list .
	GetDItem(theDialog,LISTITEM,&type,&theHandle,&iRect);
	SetDItem(theDialog,LISTITEM,type,(Handle)UserProc,&iRect);

	// Grab the information for a DUMMYITEM using GetDItem.
	// this will allow a proc to be called, I then will use that proc
	// to draw the ok button as the default.
	GetDItem(theDialog,DUMMYITEM,&type,&theHandle,&iRect);
	SetDItem(theDialog,DUMMYITEM,type,(Handle)ButtonProc,&iRect);

	// Now show the dialog on the screen.
	// NOTE:  This is when the above procs to draw the list item, and default
	//	button will be called.
	ShowWindow(theDialog);

	// Now enter modal dialog loop
	do {
		ModalDialog(myDlgFilter,&itemHit);
		// If the ListItem was hit, check to see if any cell was selected.
		// It it was, then simply put up a window, and draw contents to the
		// window.
		// This is where you would put your code to handle cell selections!
		if (itemHit == LISTITEM) {
			theCell.h = theCell.v = 0;
			if (LGetSelect(TRUE,&theCell,theList)) {
				LGetCell(s,&len,theCell,theList);
				s[len] = 0;
				ShowWindow(theWindow);
				SetPort(theWindow);
				EraseRect(&theWindow->portRect);
				MoveTo(10,18);
				DrawString(CtoPstr(s));
				ShowWindow(theDialog);
			}
		}
	} while (itemHit != OKITEM);
}

/*
	This is the dialog filter.  I use this so that I can use LClick to
	keep track of clicks in the List Item.
```

```
*/

pascal Boolean myDlgFilter (DialogPtr theDialog, EventRecord *theEvent,
                              short *itemHit)
{
        Rect    iRect;
        short   type;
        Handle  iHndle;
        GrafPtr         savePort;
        Point   p;
        char    theChar;

        // case on the event
        switch (theEvent->what) {
        case keyDown:
                // if key was pressed, handle return key
                theChar = (theEvent->message) & charCodeMask;
                if ( (theChar == 0x0d) || (theChar == 0x03)) {
                        *itemHit = OKITEM;
                        return TRUE;
                }
                return FALSE;
        case mouseDown:
                // Get where mouse click occured in global coordinates.
                p = theEvent->where;

                //   Save the current port.
                //   Then make sure port is set to the dialog.
                GetPort (&savePort);
                SetPort (theDialog);

                // Convert the coordinates to local to the dialog window
                GlobalToLocal(&p);

                // Since I am only concerned with mouse clicks in the user item,
                // get information for that item.
                GetDItem(theDialog,LISTITEM,&type,&iHndle,&iRect);

                // If the mouse click was not in LISTITEM, then let ModalDialog handle
                // it.
                if (!PtInRect(p,&iRect)) {
                        SetPort(savePort);
                        return FALSE;
                }
                // Mouse Click was in list item,
                // Set the itemHit to be the LISTITEM,
                // and call LClick on itemhandle (the list).
                *itemHit = LISTITEM;
                LClick(p,theEvent->modifiers,(ListHandle)iHndle);
                // Reset Port, and let ModalDialog know that we handled the event.
                SetPort(savePort);
                return TRUE;
        default :
                return FALSE;
        }
}
```

```
/*
    This is the proc that the I use to draw a border around
    the OK Button to mark it as the default item.
*/

pascal void ButtonProc(WindowPtr theDialog,short theItem)
{
    Rect    iRect;
    Handle  iHndl;
    short   iType;

    //  Grab Information for OKITEM out of the Dialog
    GetDItem(theDialog,OKITEM,&iType,&iHndl,&iRect);

    // Now frame it
    PenSize(3,3);
    InsetRect(&iRect,-4,-4);
    FrameRoundRect(&iRect,16,16);
}


/*
    This proc is used to create the list for the LISTITEM,
    and then "install" it into the dialog item list
*/

pascal void UserProc(WindowPtr theDialog, short theItem)
{

    Rect    iRect,
            rView,
            rBounds;
    short   h,
            v;
    char    s[25];
    Point   pCellSz,theCell;
    Handle  theHandle;
    short   itype;
    Rect    tempRect;

    // Grab item information
    GetDItem(theDialog,theItem,&itype,&theHandle,&iRect);

    // Set up view for the list.  Notice that there is some
    // margin left on all sides for the frame, and the verical scroll bar
    rView = iRect;
    rView.right -= 16;
    rView.left +=2;
    rView.bottom -=1;
    rView.top += 1;

    // list array is 1 colume with 25 rows
    SetRect(&rBounds,0,0,1,25);
```

```
	// force auto calculations when displaying the cells.
	pCellSz.h = 0;
	pCellSz.v = 0;

	// create list and draw it.
	theList = LNew (&rView, &rBounds, pCellSz, 0, theDialog, TRUE, TRUE, FALSE,
				TRUE);


	if (!theList) {
		DebugStr("\pUnable to allocate list");
		return;
	}

	// Now initialize the cells
	for (v=0; v < 25; v++) {
		sprintf (s,"This is cell %d ",v);
		SetPt(&theCell,0,v);
		LSetCell(s,strlen(s),theCell,theList);
	}

	// Draw a frame around the user item
	FrameRect (&iRect);

	// associate the new list handle with this user item
	SetDItem(theDialog,theItem,0,(Handle)theList,&iRect);
}

/* Rez description file for 'DLOG' and 'DITL' resources used by the example
 * above
 */
#include "Types.r"

resource 'DLOG' (128) {
	{40, 40, 240, 280},
	dBoxProc,
	visible,
	goAway,
	0x0,
	128,
	""
};

resource 'DITL' (128) {
	{	/* array DITLarray: 3 elements */
		/* [1] */
		{163, 136, 190, 222},
		Button {
			enabled,
			"OK"
		},
		/* [2] */
		{4, 13, 141, 220},
		UserItem {
			disabled
```

```
                },
                /* [3] */
                {158, 133, 195, 229},
                UserItem {
                        disabled
                }
        }
};
```