New Technical Notes
Macintosh

Developer Support

List Manager Q&As
Toolbox

Revised by: Developer Support Center October 1992
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

Avoid Macintosh List Manager for manipulating large matrices
Date Written: 7/22/91
Last reviewed: 8/30/91

Avoid using the Macintosh List Manager for manipulating large matrices like spreadsheet cells.

The Macintosh List Manager becomes ungainly when handling more than about 500 cells. With several thousand cells, such as a 100x100 matrix, it becomes downright klunky.

Using standard techniques, the List Manager can store only 32K of data in a list. For 10,000 cells, this gives you only about 3.2 characters in a cell. It's possible to switch to a system where each cell contains a pointer to text to be drawn, but with only 3.2 characters, it's not possible to store a full pointer (4 bytes). After about 500 cells, the time needed to actually set the data in the cells becomes noticeable, on the order of 10 to 15 seconds.

Finally, the List Manager has some pretty fundamental restrictions. You cannot, for instance, create variable-width rows or columns, or hide a single row or column from a list.

For manipulating large matrices, you’re much better off bypassing the List Manager entirely and writing your own spreadsheet-type mechanism. Consider caching the cells to a temporary file, depending on your memory requirements. You’ll need to use the Control Manager to set up and process scroll bars, and you should use QuickDraw to draw the contents of the cells.

Basically, writing a spreadsheet shares many of the same necessary functions and problems as writing a text editor (replacing TextEdit). Neither one is simple, but both are possible.

Creating Macintosh columns with various widths

Date Written: 2/8/91
Last reviewed: 4/10/91

How can I create a Macintosh List Manager list with different size cells for some columns and different size cells for other columns?

___

The List Manager doesn't allow variable-sized cells. One cell size fits all, or *not!* The List Manager was designed for simple lists (such the standard get file dialog), but not for spreadsheets. The source of the limitation is found in the ListRec as documented in *Inside Macintosh* Volume IV. The cellSize is a single variable (a Point). If multiple column widths were allowed, some sort of array would be required to store the width for each column, and you can see from looking at the ListRec data structure that no provision was made for this.

You can write your own List definition procedure, which can display alternative data types (PICTS, ICONS, or perhaps some data type of your own devising), but these are still limited to one cell size for the entire list. If you wish to have columns with various widths, you will need to treat each column as a separate list, or handle the lists entirely yourself as spreadsheet and database applications do.

Code for putting a list in a modal dialog box

Date Written: 1/23/92
Last reviewed: 2/6/92

How do I put a list in a Macintosh modal dialog box?

___

The sample code below puts a list in a modal dialog. There are several points to note to ensure success: You need to pass a draw procedure to the modal dialog for your list item. It's important to test for mousedown events in your modal dialog filter, because the List Manager doesn't check to see if you're kidding when you tell it you have a mousedown (that's what calling lClick does…). Also, calling Setport on your dialog in your filter procedure is not done for you automatically and is important when using GlobalToLocal.

The sample also tests lGetSelect to see that it works OK on these lists (it does). The dialog in question simply needs two items in its ditl, a button as item 1 and a user item as item 2. This should work for any sized user item.

```
pascal listDraw(WindowPtr theWindow, short itemNo)
{
   short       aItemType;
   Handle      aControl;
   Rect        aRect;
   Rect        dBounds;
   Point       aPoint;
   GetDItem((DialogPtr)theWindow,itemNo,&aItemType,&aControl,&aRect);

   PenNormal();
   InsetRect(&aRect,-1,-1);
   FrameRect(&aRect);
   LUpdate(theWindow->visRgn,(ListHandle)GetWRefCon(theWindow));
}

pascal Boolean MyFilter( DialogPtr theDialog, EventRecord *theEvent,
                        short *itemHit)
{
   short       aItemType;
   Handle      aControl;
   Rect        aRect;
   Rect        dBounds;
   Point       aPoint;

   SetPort(theDialog);
   GetDItem(theDialog,2,&aItemType,&aControl,&aRect);
   aPoint = theEvent->where;
   GlobalToLocal(&aPoint);
   if (PtInRect(aPoint,&aRect))
     if (theEvent->what==mouseDown) {
        *itemHit = 2;
       LClick(aPoint,theEvent->modifiers,
              (ListHandle)GetWRefCon((WindowPtr)theDialog));
       return true;
   };
   return false ;
}

void DialogFiddle()
{
   DialogPtr   aDLog;
   short       aItemType;
   Handle      aControl;
   Rect        aRect;
   Rect        dBounds;
   Point       aPoint;
   ListHandle  myList;
   Str255      dummy = ""Unselected Item"";
   Str255      wally = ""I am Selected!"";

   aDLog = GetNewDialog(987,(Ptr) 0, (WindowPtr) -1);
   GetDItem(aDLog,2,&aItemType,&aControl,&aRect);
   SetDItem(aDLog,2,aItemType,(Handle)&listDraw,&aRect);
   aRect.right -= 16;
   dBounds.top = 0;
   dBounds.bottom = 0;
   dBounds.left = 0;
   dBounds.right = 1;
   aPoint.h = 0;
   aPoint.v = 0;
   myList = LNew(&aRect,&dBounds,aPoint,0,(WindowPtr)aDLog,
                true,false,false,true);
   aItemType = LAddRow(5,0,myList);
   LSetCell(&dummy,15,aPoint,myList);
   aPoint.v+=1;
   LSetCell(&dummy,15,aPoint,myList);
   aPoint.v+=1;
   LSetCell(&dummy,15,aPoint,myList);
   aPoint.v+=1;
   LSetCell(&dummy,15,aPoint,myList);
   aPoint.v+=1;
   LSetCell(&dummy,15,aPoint,myList);
   (**myList).selFlags = -128;
   SetWRefCon((WindowPtr)aDLog,(long)myList);
   do {
       aPoint.h = 0;
       aPoint.v = 0;
       ModalDialog(&MyFilter,&aItemType);
       if (aItemType=2)
         if (LGetSelect(true,&aPoint,myList))
           LSetCell(&wally,14,aPoint,myList);
   } while (aItemType==2);
   DisposDialog(aDLog);
}
```