LNextCell Query which cell is next in a list

#include <Lists.h> **List Manager Package**

Boolean	**LNextCell(***hNext*, *vNext*, *theCell*, *theList* **);**	
Boolean	*hNext* ;	look horizontally (toward the right)?
Boolean	*vNext* ;	look vertically (toward the bottom)?
Cell	***theCell* ;	starting position; receives next cell
ListHandle	*theList* ;	handle leading to a ListRec
	returns	Was a cell located?

LNextCell advances from one cell position to the next. You can advance horizontally only (across a row), vertically only (down a column), or both (horizontally until at the end of a row and wrapping from row to row).

hNext and . . .
vNext Are Booleans that identify how to look for the next cell. There are three meaningful combinations:

hNext=TRUE	**(***vNext*=FALSE) Advance horizontally to the right. If beyond the end of the row, return FALSE.
vNext=TRUE	**(***hNext*=FALSE) Advance vertically toward the bottom. If beyond the bottom of the list, return FALSE.
Both TRUE	Advance horizontally to the right. If beyond the last column, advance to first cell in the the next lower row. If beyond the bottom of the list, return FALSE.

theCell is the address of a 32-bit Cell (a.k.a. Point). On entry, it specifies where to start looking. Upon return, it contains the cell coordinates of the next cell, according to the criteria set forth in *hNext* and *vNext*. If the return value is FALSE, the value of *theCell* is undefined upon exit.

theList is a handle leading to a variable-length ListRec structure. It is a value previously obtained via **LNew**.

Returns: a Boolean identifying whether a valid 'next' cell was obtained. It is one of:

FALSE	No more cells - either at the end of a row, column, or list. The value of *theCell* is now undefined.
TRUE	A valid 'next' cell was found, its coordinates are in *theCell*.

Notes: You can use **LNextCell** in place of a set of nested loops. For instance, you could use it to loop through and deselect all cells (a function NOT provided by the List Manager). Another example: given a mouse location in local coordinates, you could determine which cell was currently pointed to via:

```
Cell        theCell;
Rect        cellRect;
Point       mousePt;
Boolean     found;

found = FALSE;  theCell.h=theCell.v=0;         /* start at top left */
```

```
do {
    LRect( &cellRect, theCell, theList );       /* get rect for this cell */
    if ( PtInRect( mousePt,  &cellRect ) )     /* if mouse is there ... */
        found = TRUE;                          /* ... we're done looking */
} while ( LNextCell( TRUE,TRUE, &theCell, theList ) && !found );
if (found) { /*. . . theCell  is the cell of interest . . . */ }
```

LNextCell is also handy in locating a series of selected cells (see **LGetSelect** for an example of usage). To locate a cell containing some specific data, use **LSearch** (i.e., there is no need to use **LNextCell** to scan the contents of a list manually).