Rearranging a listbox and repeating buttons

by Jim Brossman

The following demo shows two different programming features, the first is a way to rearrange a list box to the user's preference, not necessarily alphabetically or in ascending or descending numerical order. The second feature is to use repeating buttons to move the items in the listbox up or down.

The repeating buttons are actually two bitmaps available in the Liberty Basic 4.03 bmp folder that are placed in a graphic box to look similar to a spinner control. When the left mouse is clicked down, it is detected in the graphic box using when LeftButtonDown. Using the MouseX and MouseY commands you can tell if the cursor is on the up or down button and the selected item will move. If the left mouse button is held down, the selected item will continue to move. This eliminates the need to repeatable click an up or down arrow to move an item a long distance. When the LeftButtonUp is detected by the scan in the [moveup] or [movedown] sections, the program will go to [stopmove] and wait.

```
'Rearranging a listbox contents and Repeating buttons demo

    nomainwin
    dim array$(10)
    'This fills the listbox array, but it could be filled from a file.
    array$(1)="Pigeon"
    array$(2)="Donkey"
    array$(3)="Horse"
    array$(4)="Rabbit"
    array$(5)="Dog"
    array$(6)="Cat"
    array$(7)="Goat"
    array$(8)="Mule"

    NUMBERRECORDS=8  'Number of records in the array or the file

'This demo uses an up and down arrow bitmap button from Liberty Basic'
s bmp folder.
    'They are 20 x 20 pixels each.
    loadbmp "upbutton",
"C:\Program Files\Liberty Basic v4.03\bmp\MAXBX.bmp"
    loadbmp "dnbutton",
"C:\Program Files\Liberty Basic v4.03\bmp\MINBX.bmp"

    WindowWidth =400
    WindowHeight=400
    'This graphicbox will be used to place both buttons.
```

```
'If you use a dialog window, the graphic box should be about 4 pixels
wider
    'and 4 pixels deeper to look best.
    graphicbox #main.gb1, 345,160,20,40
    listbox #main.lb1, array$(, [selectitem], 10, 30, 305, 300
    statictext #main.st1, "", 20, 5, 300, 20
    open "Reorganize a List" for window as #main

    #main.lb1, "singleclickselect"
    #main, "trapclose [quit]"
    currec=0 'Used to contain the current element in the array.
    flag=0 'This flag is used to determine if an item has been moved.

'Put the pen down and draw the up button at the top of the graphic box
.
    #main.gb1, "down;drawbmp upbutton 0 0"
    'Draw the down button in the bottom of the graphis box.
    #main.gb1, "drawbmp dnbutton 0 20;flush"
    'When the left button is depressed in the graphic box.
    #main.gb1, "when leftButtonDown [leftdown]"
    'When the left button is released.
    #main.gb1, "when leftButtonUp [stopmove]"
    wait

    [selectitem]
'When an item is double clicked for selection in the listbox
    #main.lb1, "selection? item$" 'get the name of the item selected.
    #main.lb1, "selectionindex? currec"
'Get the current record number of the selected item.
    #main.st1, "Moving - "+item$ 'Indicates which item you will move.
    'If an item has been moved and a second item selected
    'put the first item into the array at the last spot.
    if flag=1 then
        array$(lastmove)=origitem$
        flag=0 'reset the flag
    end if
    origitem$=item$ 'Save the newly selected item's name.
    wait

    [leftdown]
'When the left mouse button is depressed while in the graphic box.

'If the left button was depressed while in the lower 1/2 of the graphi
cbox.
    if MouseY > 20 then 'Mouse in lower half of graphic box
```

```
        goto [movedown]
    else 'Mouse was in the upper half of graphic box.
        goto [moveup]
    end if

    [moveup]'up button loop
    if currec=0 then wait
'If no item was selected when left button down in graphic box.
    if currec>1 then
'The current record must be 2 or more or a move can't be done.
        a$=array$(currec-1)
'Get the item above the currently seleted item.
        array$(currec-1)=item$
'Put the currently selected item in the array element above.
        array$(currec)=a$
'Put the item from above into the current element.
        #main.lb1,"reload" 'Reload the array so it shows the move.

'Subtract one from the current record to indicate the current element
of the array.
        currec=currec-1
        'Select the new current array element.
        'This is the new position of the selected item.
        #main.lb1,"selectindex ";currec
    end if

'Set the lastmove = to the current element so that if another item is

'selected, the item being moved will be replaced with the original nam
e
    'in the [selectitem] section above.
    lastmove=currec
    'and set the flag to 1 so it will know a move has been effected.
    flag=1
    'Make this API call as a 300 millisecond delay.
    'This avoids an almost instant move to the top of the listbox.
    calldll #kernel32, "Sleep", 300 as ulong, result as void
    scan
'This scan is necessary to see when the left button is released
    'while in this loop.
    goto [moveup]
'By looping to the beginning of this section, holding the left button
    'down will continue to effect movement.

    [movedown]'down button loop
    if currec=0 then wait
```

```
'If no item was selected when left button down in graphic box.

'The current record must be less than the number of record or a move c
an't be done.
    if currec<NUMBERRECORDS then
        a$=array$(currec+1)
'Get the item below the currently seleted item.
        array$(currec+1)=item$
'Put the currently selected item in the array element below.
        array$(currec)=a$
'Put the item from below into the current element.
        #main.lb1,"reload" 'Reload the array so it shows the move.

'Add one from the current record to indicate the current element of th
e array.
        currec=currec+1
        'Select the new current array element.
        'This is the new position of the selected item.
        #main.lb1,"selectindex ";currec
    end if

'Set the lastmove = to the current element so that if another item is

'selected, the item being moved will be replaced with the original nam
e
    'in the [selectitem] section above.
    lastmove=currec
    flag=1
'and set the flag to 1 to indicate a move has been effected.
    'Make this API call as a 300 millisecond delay.
    'This avoids and almost instant move to the top of the listbox.
    calldll #kernel32, "Sleep", 300 as ulong, result as void
    scan
'This scan is necessary to see when the left button is released
    'while in this loop.
    goto [movedown]
'By looping to the beginning of this section, holding the left button
    'down will continue to effect movement.

    [stopmove]  'If the left button is released.
    'break out of the loop and wait.
    wait

    [quit]'Close or Cancel
    'At this point the array can be saved to a file in the new order.
    #main.gb1, "cls"
```

```
'Unload the bitmaps from memory.
unloadbmp "dnbutton"
unloadbmp "upbutton"
close #main
end
```