

Auto sizing combo box

By Petra Zamburek

This is a workaround for the shrinking of a drop down list after each screen redesign.

The problem...

You have so many objects to arrange and the screen is so small! So you do a pretty good design with many moves and resizes of your objects – and finally, everything fits. As the last step, you set the size of the drop down lists for your combo boxes. Your customers are happy.

How it continues:

During the lifetime of your application, customers usually want several changes, additional information as well as issues like “this object should be displayed on the left side and not on the right side” or “can’t you increase the size of this data field and shrink this list box?”. Their wish is your command and you do a redesign.

How it ends:

Some day you realize that the size of the combo box drop down list displays only 3 entries, though you sized them big enough to show all entries! What happened?

Well, that’s a bug (I think), but you can help yourself...

The solution...

You can calculate and set the size of the drop down list each time the position of the combo box changes. First I tried to do that on WM_MOVE or WM_SIZE, but the result was a heavy flickering when moving or resizing the window. Solution: do the size before the list is displayed: WM_LBUTTONDOWN is the right place.

Now you never have to do a resize and the list always gets calculated at the maximum possible size. If an object is located in the upper half of the screen: the list drops down. If the object is located in the lower half of the screen: the list “drops up”.

Here is a sample how to do this:

```
Library name: user32.dll
  Function: GetDC
    Description:
    Export Ordinal: 0
    Returns
      Number: HANDLE
    Parameters
      Window Handle: HWND
  Function: ReleaseDC
    Description:
    Export Ordinal: 0
    Returns
      Boolean: BOOL
    Parameters
      Window Handle: HWND
      Number: HANDLE
  Function: GetParent
    Description:
    Export Ordinal: 0
    Returns
      Window Handle: HWND
    Parameters
      Window Handle: HWND
  Function: GetWindowRect
    Description:
    Export Ordinal: 0
    Returns
      Boolean: BOOL
    Parameters
```

```

        Window Handle: HWND
        structPointer
            ! left, top, right, bottom
            Receive Number: LONG
            Receive Number: LONG
            Receive Number: LONG
            Receive Number: LONG
Library name: gdi32.dll
Function: GetDeviceCaps
Description:
Export Ordinal: 0
Returns
    Number: INT
Parameters
    Number: HANDLE
    Number: INT

Constants
    Number: VERTRES = 10
    Number: WM_LBUTTONDOWN = 0x201

Function: SalGetWindowSizeWidthX
Description:
Returns
    Number:
Parameters
    Window Handle: _hWndItem
Static Variables
Local variables
    Number: _nWidth
    Number: _nHeight
Actions
    Call SalGetWindowSize (_hWndItem, _nWidth, _nHeight)
    Return _nWidth
Function: FaGetWindowHandle
Description:
Returns
    Window Handle:
Parameters
    Window Handle: _hWnd
Static Variables
Local variables
Actions
    If (SalGetType (_hWnd) = TYPE_FormWindow OR SalGetType (_hWnd) = TYPE_DialogBox)
        AND SalGetFirstChild (_hWnd, TYPE_FormToolBar) != hWndNULL
        Return GetParent (_hWnd)
    Else
        Return _hWnd

Combo Box Class: CcmbAutosize
Class Variables
    ! handle and size of the screen
    Number: _nCvHDCScreen
    Number: _nCvScreenHeight
    ! last position on screen
    Number: _nCvItemLastRectLeft
    Number: _nCvItemLastRectTop
    ! actual position on screen
    Number: _nCvItemRectLeft
    Number: _nCvItemRectTop
    Number: _nCvItemRectRight
    Number: _nCvItemRectBottom
    ! new calculated height
    Number: _nCvNewH
Message Actions

```

```

On WM_LBUTTONDOWN
! get actual position on screen
Call GetWindowRect (hWndItem, _nCvItemRectLeft, _nCvItemRectTop,
_nCvItemRectRight, _nCvItemRectBottom)
! check if the position has changed (window move or resize)
If _nCvItemLastRectLeft != _nCvItemRectLeft OR
_nCvItemLastRectTop != _nCvItemRectTop
If _nCvScreenHeight = 0
Set _nCvHDCScreen = GetDC (FaGetWindowHandle (SalParentWindow
(hWndItem)))
Set _nCvScreenHeight = GetDeviceCaps (_nCvHDCScreen, VERTRES)
Call ReleaseDC (FaGetWindowHandle (SalParentWindow (hWndItem)),
_nCvHDCScreen)
! calculate the max. possible size, depending on the position of the object
in the upper or lower half of the screen
If _nCvItemRectTop > _nCvScreenHeight - _nCvItemRectTop
Set _nCvNewH = _nCvItemRectTop
Else
Set _nCvNewH = _nCvScreenHeight - _nCvItemRectTop
! convert the pixels to units
Set _nCvNewH = SalPixelsToFormUnits (hWndItem, _nCvNewH, TRUE)
! set size
! if you want, subtract a little bit to keep a 'saftey' distance to the
system bar
! Call SalSetWindowSize (hWndItem, SalGetWindowSizeWidthX (hWndItem),
_nCvNewH - 0.5)
Call SalSetWindowSize (hWndItem, SalGetWindowSizeWidthX (hWndItem),
_nCvNewH)
! save actual position
Set _nCvItemLastRectLeft = _nCvItemRectLeft
Set _nCvItemLastRectTop = _nCvItemRectTop

```

Petra Zamburek lives in Vienna and is working at BUWOG. Her job is CTD-programming and DB-administration (Oracle) for her own and five 3rd-party applications (two of them written with CTD). Contact her at petra.zamburek@buwog.at