Scrollbars part 12: Applying WM_NCCALCSIZE to our scrollbar sample

devblogs.microsoft.com/oldnewthing/20030917-00

September 17, 2003

Raymond Chen

Now that we have learned about the intricacies of the `WM_NCCALCSIZE` message, we can use it to get rid of the flicker in our resizing code. We just take the trick we used above and apply it to the scroll program.

First, we need to get rid of the bad flickery resize, so return the *OnWindowPosChanging* function to the version before we tried doing metaphor work:

```
BOOL OnWindowPosChanging(HWND hwnd, LPWINDOWPOS pwp)
{
    if (!(pwp->flags & SWP_NOSIZE)) {
        RECT rc = { 0, 0, pwp->cx, pwp->cy };
        AdjustSizeRectangle(hwnd, WMSZ_BOTTOM, &rc);
        pwp->cy = rc.bottom;
    }
    return 0;
}
```

Instead, our work will happen in the `WM_NCCALCSIZE` handler.

```
UINT OnNcCalcSize(HWND hwnd, BOOL fCalcValidRects,
                  NCCALCSIZE_PARAMS *pcsp)
{
    UINT uRc = (UINT)FORWARD_WM_NCCALCSIZE(hwnd,
                        fCalcValidRects, pcsp, DefWindowProc);
    if (fCalcValidRects) {
        //  Give names to these things
        RECT *prcClientNew = &pcsp->rgrc[0];
        RECT *prcValidDst  = &pcsp->rgrc[1];
        RECT *prcValidSrc  = &pcsp->rgrc[2];
        int dpos;
        int pos;
        // Did we drag the top edge enough to scroll?
        if (prcClientNew->bottom == prcValidSrc->bottom &&
            g_cyLine &&
            (dpos = (prcClientNew->top - prcValidSrc->top)
                                        / g_cyLine) != 0 &&
            (pos = ClampScrollPos(g_yOrigin + dpos)) != g_yOrigin) {
            *prcValidDst = *prcClientNew;
            ScrollTo(hwnd, pos, FALSE);
            prcValidDst->top -= dpos * g_cyLine;
            uRc = WVR_VALIDRECTS;
        }
    }
    return uRc;
}
    /* Add to WndProc */
    HANDLE_MSG(hwnd, WM_NCCALCSIZE, OnNcCalcSize);
```

This uses a new helper function which we extracted from the *ScrollTo* function. (If I had planned this better, this would have been factored out when we first wrote *ScrollTo*.)

```
int ClampScrollPos(int pos)
{
    /*
     *  Keep the value in the range 0 .. (g_cItems - g_cyPage).
     */
    pos = max(pos, 0);
    pos = min(pos, g_cItems - g_cyPage);
    return pos;
}
```

I am not entirely happy with this code, however. It is my personal opinion that the `WM_NCCALCSIZE` handler should be stateless. Notice that this one modifies state (by calling *ScrollTo*). If I had more time (sorry, I'm rushed now – you'll learn why soon), I would have put the state modification into the `WM_WINDOWPOSCHANGING` message. So I will leave that as another exercise.

Exercise: Keep the `WM_NCCALCSIZE` message stateless.

Raymond Chen

Follow