Adding a Ctrl+arrow accelerator for moving the trackbar by just one unit, part 2: Second try

devblogs.microsoft.com/oldnewthing/20181024-00

October 24, 2018

Raymond Chen

Last time, we looked at how we could add support to the trackbar so that `Ctrl`+arrow moved the thumb by one unit, even if the line size was set to a larger value. We tried doing this by subclassing the control and adding additional keyboard handling, but this turned into a bit of a mess because of all the special cases in the trackbar to accommodate various usage patterns.

What we really want to do is let the trackbar do all its keyboard processing, and step in just before it moves the thumb, so we can move it by a different amount if the `Ctrl` key is held down.

Fortunately, there's a notification for this.

Unfortunately, it requires version 6 of the common controls.

Fortunately, version 6 of the common controls is included in all versions of Windows still in support.

Take our program from last time, but stop before we added the `TrackbarKeyProc`. (Delete the `TrackbarKeyProc` and the calls to `SetWindowSubclass` and `RemoveWindow-Subclass`.)

Instead, add this code:

```
#pragma comment(linker, \
    "\"/manifestdependency:type='win32' \
    name='Microsoft.Windows.Common-Controls' \
    version='6.0.0.0' \
    processorArchitecture='*' \
    publicKeyToken='6595b64144ccf1df' \
    language='*'\"")
```

This `#pragma` is a quick way to enable version 6 of the common controls.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 g_hwndChild = CreateWindow(TRACKBAR_CLASS, TEXT(""),
    WS_CHILD | WS_VISIBLE | TBS_NOTIFYBEFOREMOVE,
   0, 0, 100, 100,
    hwnd, (HMENU)100, g_hinst, 0);

  SendMessage(g_hwndChild, TBM_SETLINESIZE, 0, 5);
  SendMessage(g_hwndChild, TBM_SETPAGESIZE, 0, 20);

  return TRUE;
}
```

The `TBS_ NOTIFYBEFOREMOVE` style enables the `TRBN_ THUMBPOSCHANGING` notification, which we will take advantage of below.

```
LRESULT OnNotify(HWND hwnd, int idCtl, NMHDR* pnm)
{
 if (pnm->hwndFrom == g_hwndChild &&
     pnm->code == TRBN_THUMBPOSCHANGING &&
     GetKeyState(VK_CONTROL) < 0) {
  auto ptpc = (NMTRBTHUMBPOSCHANGING*)pnm;
  switch (ptpc->nReason) {
  case TB_LINEUP:
  case TB_LINEDOWN:
    int pos = (int)SendMessage(pnm->hwndFrom, TBM_GETPOS, 0, 0);
    pos += (ptpc->nReason == TB_LINEUP) ? -1 : +1;
    SendMessage(pnm->hwndFrom, TBM_SETPOS, TRUE, pos);
    return TRUE; // we moved the thumb, so the control doesn't have to
  }
 }
 return 0;
}
    HANDLE_MSG(hwnd, WM_NOTIFY, OnNotify);
```

The `TRBN_ THUMBPOSCHANGING` notification is sent before the trackbar moves the thumb. and the `nReason` tells you why the trackbar wants to move the thumb.[1] If the `Ctrl` key is held down, and the reason is either a line-up or a line-down, then we fetch the current trackbar position, adjust it by one unit, and set that as the new trackbar position. We then return `TRUE` to tell the trackbar that it shouldn't move the trackbar thumb (because we moved it).

(Don't forget that if this is happening in a dialog box, you need to use `DWLP_ MSGRESULT` to make the dialog box return a nonzero value from its window procedure.)

Responding to the notification leaves the trackbar to deal with recognizing the keyboard keys and taking the various trackbar configuration settings into account in order to convert them to scroll actions. We then detect the *change position by one line* action and apply our special thumb motion if the `Ctrl` key is held down, leaving the trackbar to manage the keyboard cues and other accessibility states.

[1] There's also a `dwPos` that tells you where the thumb is moving *to*, but we are more interested in where the thumb is moving *from*.

Raymond Chen

Follow