Adding a Ctrl+arrow accelerator for moving the trackbar by just one unit, part 1: Initial plunge

devblogs.microsoft.com/oldnewthing/20181023-00

October 23, 2018

Raymond Chen

When you create a trackbar common control, you can specify how far the arrow keys will change the trackbar position (default: 1 unit) and how far the `PgUp` and `PgDn` keys will change the trackbar position (default: one fifth of the range).

If you change the default distance for the arrow keys to, say, five units, then you probably want to add a keyboard accelerator for moving by just one units, so that somebody can use `PgUp` and `PgDn` to get in the general area they want to be, then the arrow keys to get close, and then finally the `Ctrl`+arrow keys to get the exact value.

Take the scratch program and make the following changes:

```
#include <strsafe.h>

BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 g_hwndChild = CreateWindow(TRACKBAR_CLASS, TEXT(""),
    WS_CHILD | WS_VISIBLE,
    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;
}
```

We start by creating a trackbar control and setting the line size to 5 units and page size to 20 units.

```
void OnHScroll(HWND hwnd, HWND hwndCtl, UINT code, int pos)
{
 if (hwndCtl == g_hwndChild) {
  TCHAR buf[128];
  pos = (int)SendMessage(hwndCtl, TBM_GETPOS, 0, 0);
  StringCchPrintf(buf, ARRAYSIZE(buf), TEXT("pos = %d"), pos);
  SetWindowText(hwnd, buf);
 }
}

    HANDLE_MSG(hwnd, WM_HSCROLL, OnHScroll);
```

And we respond to the `WM_HSCROLL` message by displaying the trackbar's new position.

If you run this program, you'll see a happy trackbar, and you can use the keyboard to move the thumb by 20 units (with `PgUp` and `PgDn`), or by 5 units (with the left and right arrow keys). But there's no way to move the thumb by just one unit.

Let's fix that.

But how?

The first thing that comes to mind is to subclass the trackbar control and add a new keyboard accelerator. So let's do that.

```
LRESULT CALLBACK TrackbarKeyProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam,
    UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
 if (uMsg == WM_KEYDOWN && GetKeyState(VK_CONTROL) < 0) {
  int delta = 0;
  if (wParam == VK_LEFT) {
   delta = -1;
  } else if (wParam == VK_RIGHT) {
   delta = +1;
  }

  if (delta) {
   auto pos = SendMessage(hwnd, TBM_GETPOS, 0, 0);
   pos += delta;
   SendMessage(hwnd, TBM_SETPOS, TRUE, pos);
   FORWARD_WM_HSCROLL(GetParent(hwnd), hwnd,
    delta < 0 ? TB_LINEUP : TB_LINEDOWN, 0, SendMessage);
   return 0;
  }
 }
 return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}

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

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

  SetWindowSubclass(g_hwndChild, TrackbarKeyProc, 0, 0);
  return TRUE;
}

void
OnDestroy(HWND hwnd)
{
  RemoveWindowSubclass(g_hwndChild, TrackbarKeyProc, 0);
  PostQuitMessage(0);
}
```

With this version, you can hold the `Ctrl` key when pressing the left or right arrow keys, and the position will change by just one unit.

Mission accomplished?

Not quite. There's still a lot of stuff missing. You may not notice it right away, but you will eventually, probably when one of your customers reports a problem that makes you have to scramble a fix.

For example, this code doesn't manage keyboard focus indicators. Let's fix that.

```
LRESULT CALLBACK TrackbarKeyProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam,
    UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
 if (uMsg == WM_KEYDOWN && GetKeyState(VK_CONTROL) < 0) {
  ...
  if (delta) {
   ...
   FORWARD_WM_HSCROLL(GetParent(hwnd), hwnd,
    delta < 0 ? TB_LINEUP : TB_LINEDOWN, 0, SendMessage);
   SendMessage(hwnd, WM_CHANGEUISTATE,
    MAKELONG(UIS_CLEAR, UISF_HIDEFOCUS), 0);
   return 0;
  }
 }
 return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}
```

Next, this version doesn't support vertical trackbars at all. If you add the `TBS_VERT` style to the `CreateWindow` call, you'll have a vertical scroll bar, and we haven't been doing anything with the up and down arrows.

In related news, the trackbar allows you to use the up and down arrows to change the position of horizontal scroll bars. The up arrow behaves like the right arrow, and the down arrow behaves like the left arrow. Maybe you have customers who rely on this behavior, say, because that's what their accessibility tool uses.

Fortunately, one set of changes covers both of these issues.

```
LRESULT CALLBACK TrackbarKeyProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam,
    UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
 if (uMsg == WM_KEYDOWN && GetKeyState(VK_CONTROL) < 0) {
  int delta = 0;
  if (wParam == VK_LEFT || wParam == VK_UP) {
   delta = -1;
  } else if (wParam == VK_RIGHT || wParam == VK_DOWN) {
   delta = +1;
  }
  ...
 }
 return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}
```

But wait, there's also the `TBS_ DOWNISLEFT` style that changes the mapping between vertical and horizontal. If the style is set, then the up arrow acts like the left arrow, and the down arrow acts like the right arrow.

Okay, so let's fix that too.

```
WPARAM SwapKeys(WPARAM wParam, UINT vk1, UINT vk2)
{
  if (wParam == vk1) return vk2;
  if (wParam == vk2) return vk1;
  return wParam;
}

LRESULT CALLBACK TrackbarKeyProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam,
    UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
 if (uMsg == WM_KEYDOWN && GetKeyState(VK_CONTROL) < 0) {
  DWORD style = GetWindowStyle(hwnd);
  if (style & TBS_DOWNISLEFT) {
   if (style & TBS_VERT) {
    wParam = SwapKeys(wParam, VK_LEFT, VK_RIGHT);
   } else {
    wParam = SwapKeys(wParam, VK_UP, VK_DOWN);
   }
  }

  int delta = 0;
  if (wParam == VK_LEFT || wParam == VK_UP) {
   delta = -1;
  } else if (wParam == VK_RIGHT || wParam == VK_DOWN) {
   delta = +1;
  }

  if (delta) {
   auto pos = SendMessage(hwnd, TBM_GETPOS, 0, 0);
   pos += delta;
   SendMessage(hwnd, TBM_SETPOS, TRUE, pos);
   FORWARD_WM_HSCROLL(GetParent(hwnd), hwnd,
    delta < 0 ? TB_LINEUP : TB_LINEDOWN, 0, SendMessage);
   SendMessage(hwnd, WM_CHANGEUISTATE,
    MAKELONG(UIS_CLEAR, UISF_HIDEFOCUS), 0);
   return 0;
  }
 }
 return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}
```

Okay, are we done now?

Nope, you still have right-to-left languages to deal with. In those cases, we want to flip the meanings of the left and right arrows.

```
LRESULT CALLBACK TrackbarKeyProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam,
    UINT_PTR uIdSubclass, DWORD_PTR dwRefData)
{
 if (uMsg == WM_KEYDOWN && GetKeyState(VK_CONTROL) < 0) {
  if (GetWindowExStyle(hwnd) & WS_EX_LAYOUTRTL) {
   wParam = SwapKeys(wParam, VK_LEFT, VK_RIGHT);
  }

  DWORD style = GetWindowStyle(hwnd);
  if (style & TBS_DOWNISLEFT) {
   if (style & TBS_VERT) {
    wParam = SwapKeys(wParam, VK_LEFT, VK_RIGHT);
   } else {
    wParam = SwapKeys(wParam, VK_UP, VK_DOWN);
   }
  }

  int delta = 0;
  if (wParam == VK_LEFT || wParam == VK_UP) {
   delta = -1;
  } else if (wParam == VK_RIGHT || wParam == VK_DOWN) {
   delta = +1;
  }

  if (delta) {
   auto pos = SendMessage(hwnd, TBM_GETPOS, 0, 0);
   pos += delta;
   SendMessage(hwnd, TBM_SETPOS, TRUE, pos);
   FORWARD_WM_HSCROLL(GetParent(hwnd), hwnd,
    delta < 0 ? TB_LINEUP : TB_LINEDOWN, 0, SendMessage);
   SendMessage(hwnd, WM_CHANGEUISTATE,
    MAKELONG(UIS_CLEAR, UISF_HIDEFOCUS), 0);
   return 0;
  }
 }
 return DefSubclassProc(hwnd, uMsg, wParam, lParam);
}
```

Okay, *now* are we done?

Maybe. I think that covers the remaining issues, but maybe I missed something.

Y'know, this started out as a simple fix, but all the special cases turned it into a complicated mess. And maybe a future version of the trackbar control will add yet another style that introduces another special case. What we really want to do is hook into the control after it has decided what to do with the keyboard and before it changes the trackbar position.

Let’s work on that next time.

Raymond Chen

Follow