Generating tooltip text dynamically

devblogs.microsoft.com/oldnewthing/20060629-00

June 29, 2006

Raymond Chen

Our multiplexed tooltip right now is displaying the same string for all items. Let's make it display something a bit more interesting so it's more obvious that what we're doing is actually working.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 ...
 // ti.lpszText = TEXT("Placeholder tooltip");
 ti.lpszText = LPSTR_TEXTCALLBACK;
 ...
}
LRESULT
OnNotify(HWND hwnd, int idFrom, NMHDR *pnm)
{
 if (pnm->hwndFrom == g_hwndTT) {
  switch (pnm->code) {
  case TTN_GETDISPINFO:
   {
    NMTTDISPINFO *pdi = (NMTTDISPINFO *)pnm;
    if (g_iItemTip >= 0) {
     // szText is 80 characters, so %d will fit
     wsprintf(pdi->szText, TEXT("%d"), g_iItemTip);
    } else {
     pdi->szText[0] = TEXT('\0');
    }
    pdi->lpszText = pdi->szText;
   }
   break;
  }
 }
 return 0;
}
// Add to WndProc
 HANDLE_MSG(hwnd, WM_NOTIFY, OnNotify);
```

Instead of providing fixed tooltip text, we generate it on the fly by setting the text to `LPSTR_TEXTCALLBACK` and producing the text in response to the `TTN_GETDISPINFO` notification. The technique of generating tooltip text dynamically is useful in scenarios other than this. For example, the tooltip text may change based on some state that changes often ("Back to <insert name of previous page>") or the tooltip text may be slow or expensive to compute ("Number of pages: 25"). In both cases, updating the tooltip text lazily is the correct thing to do, since it falls into the "pay for play" model: Only if the user asks for a tooltip does the program go to the extra effort of producing one.

Now that you've played with the program a bit, let's tweak it every so slightly to illustrate a point I made last time: We'll make the `+` and `-` keys add and remove colored bars. This lets you see how the tooltip code updates itself when items move around.

```
void
InvalidateItems(HWND hwnd, int iItemMin, int iItemMax)
{
 RECT rc;
 SetRect(&rc, 0, g_cyItem * iItemMin,
         g_cxItem, g_cyItem * iItemMax);
 InvalidateRect(hwnd, &rc, TRUE);
}
void
UpdateTooltipFromMessagePos(HWND hwnd)
{
 DWORD dwPos = GetMessagePos();
 POINT pt = { GET_X_LPARAM(dwPos),
              GET_Y_LPARAM(dwPos) };
 ScreenToClient(hwnd, &pt);
 UpdateTooltip(pt.x, pt.y);
}
void
OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
 switch (ch) {
 case TEXT('+'):
  g_cItems += cRepeat;
  InvalidateItems(hwnd, g_cItems - cRepeat, g_cItems);
  UpdateTooltipFromMessagePos(hwnd);
  break;
 case TEXT('-'):
  if (cRepeat > g_cItems) cRepeat = g_cItems;
  g_cItems -= cRepeat;
  InvalidateItems(hwnd, g_cItems, g_cItems + cRepeat);
  UpdateTooltipFromMessagePos(hwnd);
  break;
 }
}
// Add to WndProc
 HANDLE_MSG(hwnd, WM_CHAR, OnChar);
```

We have a few new helper functions. The first invalidates the rectangle associated with a range of items. (Conforming to Hungarian convention, the term "Max" refers to the first element outside the range. In other words, "Min/Max" is endpoint-exclusive.) Controls that manage sub-elements will almost always have a function like `InvalidateItems` in order to trigger a repaint when a sub-element changes its visual appearance.

The next helper function is `UpdateTooltipFromMessagePos` which pretty much does what it says: It takes the message position and passes those coordinates (suitably converted) to `UpdateTooltip` in order to keep everything in sync. Finally, the `WM_CHAR` handler adds or removes items based on what the user typed (taking autorepeat into account). Whenever we change the number of items, we update the tooltip because one of the items that was added or removed may have been the one beneath the cursor.

There is an important subtlety to the `UpdateTooltipFromMessage` function: Remember that the message position retrieved via `GetMessagePos` applies to the most recent message retrieved from the message queue. Messages delivered via `SendMessage` bypass the message queue and therefore do not update the queue message position. Once again, we see by a different means that you can't simulate input with `SendMessage`.

Raymond Chen

Follow