Creating tree view check boxes manually: Responding to clicks

devblogs.microsoft.com/oldnewthing/20171130-00

November 30, 2017

Raymond Chen

Last time, we added state images to the tree view, but we didn't provide any way for the user to click on them. Let's add the code so that the user can click on the check box to change the value. For simplicity, we will just cycle through the state images. In real life, you would probably have more complex logic for deciding what to do when the user clicks on the check box, such as ignoring clicks on disabled check boxes.

Take our program from last time and make these additions:

```
void CycleStateImage(HWND hwndTV, HTREEITEM hti)
{
  UINT oldState = TreeView_GetItemState(hwndTV, hti,
                                        TVIS_STATEIMAGEMASK);
  int stateIndex = (int)(oldState & TVIS_STATEIMAGEMASK) >> 12;

  stateIndex = stateIndex + 1;
  if (stateIndex >= ImageList_GetImageCount(
                TreeView_GetImageList(hwndTV, TVSIL_STATE)))
  {
    // We ran out of states. Wrap around to the first state.
    stateIndex = 1;
  }

  TreeView_SetItemState(hwndTV, hti,
      INDEXTOSTATEIMAGEMASK(stateIndex), TVIS_STATEIMAGEMASK);
}
```

To cycle to the next state image, we request the state image mask from the current state, isolate the state image mask, and increment it, wrapping around to 1 if we were on the last state. (Note that we wrap around to 1 rather than to 0, because we saw last time that state image zero is not used; if you ask for state image zero, that means that you want no state image at all.)

```
LRESULT OnNotifyTVClicked(HWND hwndTV)
{
  TVHITTESTINFO tvhti;
  DWORD dwPos = GetMessagePos();
  tvhti.pt.x = GET_X_LPARAM(dwPos);
  tvhti.pt.y = GET_Y_LPARAM(dwPos);
  MapWindowPoints(HWND_DESKTOP, hwndTV, &tvhti.pt, 1);
  HTREEITEM hti = TreeView_HitTest(hwndTV, &tvhti);
  if (tvhti.flags & TVHT_ONITEMSTATEICON)
  {
    CycleStateImage(hwndTV, hti);
    return TRUE; // handled
  }
  return FALSE; // not handled
}
```

Frustratingly, the tree view control notifies us when it receives a click, but it doesn't tell us where the click was. We have to fetch it ourselves by calling `GetMessagePos()`, and then converting screen coordinates to client coordinates. Once we have those coordinates, we ask the tree view what is at those coordinates, and if it says "Oh, no big deal, just the state image[1] for an item," then we get all excited and cycle the state image.

```
LRESULT OnNotifyTVKeyDown(HWND hwndTV, NMTVKEYDOWN* ptvkd)
{
  switch (ptvkd->wVKey)
  {
  case VK_SPACE:
    HTREEITEM hti = TreeView_GetNextItem(hwndTV, nullptr,
                                         TVGN_CARET);
    if (hti)
    {
      CycleStateImage(hwndTV, hti);
      return TRUE; // handled
    }
  }
  return FALSE; // not handled
}
```

For keyboard accessibility, we use the space bar as an equivalent to clicking on the state image.

```
LRESULT OnNotify(HWND hwnd, int idFrom, NMHDR* pnm)
{
  if (pnm->hwndFrom == g_hwndChild)
  {
    switch (pnm->code)
    {
    case NM_CLICK:
      return OnNotifyTVClicked(pnm->hwndFrom);

    case TVN_KEYDOWN:
      return OnNotifyTVKeyDown(pnm->hwndFrom,
         CONTAINING_RECORD(pnm, NMTVKEYDOWN, hdr));
    }
  }
  return 0; // unhandled
}

// Add to WndProc
  HANDLE_MSG(hwnd, WM_NOTIFY, OnNotify);
```

Here is our `WM_NOTIFY` handler. If the notification is coming from the tree view control, then dispatch click notifications to `OnNotifyTVClicked` and dispatch key-down notifications to `OnNotifyTVKeyDown`. The `TVN_KEYDOWN` notification comes with a custom notification structure, so we convert our generic `NMHDR` pointer to a `NMTVKEYDOWN` pointer.

And there we have it. We manually implemented tree view check boxes. This code is effectively equivalent[2] to what you get when you turn on `TVS_CHECKBOXES` and it's this code that was moved into the tree view control.

Next time, we'll bring tree view check boxes into the 21st century.

[1] Note the inconsistent terminology. Normally, the state image is called a "state image", but here, it's called a "state icon".

[2] Plus a few more quirks we'll learn about later.

Raymond Chen

Follow