Creating tree view check boxes manually: A simple state image list

devblogs.microsoft.com/oldnewthing/20171129-00

November 29, 2017

Raymond Chen

Before the introduction of the `TVS_CHECKBOXES` style, you had to create your own state images which consisted of a checked and unchecked check box. To draw check boxes manually, you used the `DrawFrameControl` function. So let's do that: Build check boxes with the `DrawFrameControl` function.

As a bonus, we will add an "indeterminate" check box state and two "disabled" check box states (unchecked and checked).

Recall that the state image list for a tree view is an image list whose images correspond to the parameter passed to the `INDEXTOSTATEIMAGEMASK` macro. In other words, to get state image 1, you set the `TVIS_STATEIMAGEMASK` bits in your `stateMask` and set the `state` to include `INDEXTOSTATEIMAGEMASK(1)`.

Image zero in the image list is not used. If you set the overlay index to zero, then you get no state image at all. (Note that the nonexisting state image occupies no space. If you want to show a blank space where the state image would have been, then you need to add an explicit state image which is blank. We leave that as an exercise for the reader.)

```
HIMAGELIST CreateTreeViewCheckBoxes(HWND hwnd, int cx, int cy)
{
  const int frames = 6;

  // Get a DC for our window.
  HDC hdcScreen = GetDC(hwnd);

  // Create a 32bpp bitmap that holds the desired number of frames.
  BITMAPINFO bi = { sizeof(BITMAPINFOHEADER), cx * frames, cy, 1, 32 };
  void* p;
  HBITMAP hbmCheckboxes = CreateDIBSection(hdcScreen, &bi,
    DIB_RGB_COLORS, &p, nullptr, 0);

  // Create a compatible memory DC.
  HDC hdcMem = CreateCompatibleDC(hdcScreen);

  // Select our bitmap into it so we can draw to it.
  HBITMAP hbmOld = SelectBitmap(hdcMem, hbmCheckboxes);

  // Set up the rectangle into which we do our drawing.
  RECT rc = { 0, 0, cx, cy };

  // Frame 0 is not used. Draw nothing.
  OffsetRect(&rc, cx, 0);

  // Flags common to all of our DrawFrameControl calls:
  // Draw a flat checkbox.
  UINT baseFlags = DFCS_FLAT | DFCS_BUTTONCHECK;

  // Frame 1: Unchecked.
  DrawFrameControl(hdcMem, &rc, DFC_BUTTON,
    baseFlags);
  OffsetRect(&rc, cx, 0);

  // Frame 2: Checked.
  DrawFrameControl(hdcMem, &rc, DFC_BUTTON,
    baseFlags | DFCS_CHECKED);
  OffsetRect(&rc, cx, 0);

  // Frame 3: Indeterminate.
  DrawFrameControl(hdcMem, &rc, DFC_BUTTON,
    baseFlags | DFCS_CHECKED | DFCS_BUTTON3STATE);
  OffsetRect(&rc, cx, 0);

  // Frame 4: Disabled, unchecked.
  DrawFrameControl(hdcMem, &rc, DFC_BUTTON,
    baseFlags | DFCS_INACTIVE);
  OffsetRect(&rc, cx, 0);

  // Frame 5: Disabled, checked.
  DrawFrameControl(hdcMem, &rc, DFC_BUTTON,
    baseFlags | DFCS_INACTIVE | DFCS_CHECKED);
```

```
  // The bitmap is ready. Clean up.
  SelectBitmap(hdcMem, hbmOld);
  DeleteDC(hdcMem);
  ReleaseDC(hwnd, hdcScreen);

  // Create an imagelist from this bitmap.
  HIMAGELIST himl = ImageList_Create(cx, cy, ILC_COLOR, frames, frames);
  ImageList_Add(himl, hbmCheckboxes, nullptr);

  // Don't need the bitmap any more.
  DeleteObject(hbmCheckboxes);

  return himl;
}
```

Okay, what happened here?

We created an image list to hold $N + 1$ state images, because image zero is not used, then we draw the various images. We draw the images with the `DFCS_FLAT` style because we don't want any 3D effects. (Note also that I am ignore RTL issues.)

Let's take this for a spin. Start with our scratch program and make these changes:

```
void PopulateTreeView(HWND hwnd)
{
  TVINSERTSTRUCT tvis;
  tvis.hParent = TVI_ROOT;
  tvis.hInsertAfter = TVI_LAST;
  tvis.item.mask = TVIF_TEXT | TVIF_STATE;
  tvis.item.stateMask = TVIS_STATEIMAGEMASK | TVIS_EXPANDED;
  tvis.item.state = INDEXTOSTATEIMAGEMASK(0) | TVIS_EXPANDED;
  tvis.item.pszText = TEXT("Root");

  tvis.hParent = TreeView_InsertItem(g_hwndChild, &tvis);

  tvis.item.pszText = TEXT("0");
  tvis.item.state = INDEXTOSTATEIMAGEMASK(0);
  TreeView_InsertItem(g_hwndChild, &tvis);

  tvis.item.pszText = TEXT("1");
  tvis.item.state = INDEXTOSTATEIMAGEMASK(1);
  TreeView_InsertItem(g_hwndChild, &tvis);

  tvis.item.pszText = TEXT("2");
  tvis.item.state = INDEXTOSTATEIMAGEMASK(2);
  TreeView_InsertItem(g_hwndChild, &tvis);

  tvis.item.pszText = TEXT("3");
  tvis.item.state = INDEXTOSTATEIMAGEMASK(3);
  TreeView_InsertItem(g_hwndChild, &tvis);

  tvis.item.pszText = TEXT("4");
  tvis.item.state = INDEXTOSTATEIMAGEMASK(4);
  TreeView_InsertItem(g_hwndChild, &tvis);

  tvis.item.pszText = TEXT("5");
  tvis.item.state = INDEXTOSTATEIMAGEMASK(5);
  TreeView_InsertItem(g_hwndChild, &tvis);
}

BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  g_hwndChild = CreateWindow(WC_TREEVIEW,
    nullptr,
    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
    WS_CHILD | WS_VISIBLE,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    hwnd, nullptr, g_hinst, 0);

  HIMAGELIST himl = CreateClassicTreeViewCheckBoxes(g_hwndChild,
   16, 16);
  TreeView_SetImageList(g_hwndChild, himl, TVSIL_STATE);
```

```
  PopulateTreeView(g_hwndChild);

  return TRUE;
}

void
OnDestroy(HWND hwnd)
{
  ImageList_Destroy(TreeView_SetImageList(
    g_hwndChild, nullptr, TVSIL_STATE));
  PostQuitMessage(0);
}
```

Yes, it's anachronistic using `nullptr` in 1995. Work with me here.

Note the destruction code here is the same code we use to avoid the image list leak. It's more obvious in this formulation that the destruction is necessary, seeing as we created the image list in the first place.

We aren't responding to clicks on the check box yet. We'll add that next time. But for now, you can run the program and observe that we displayed check box state images next to items 1 through 5. Item 0 does not have a state image because a state image index of zero means "No state image."

Raymond Chen

Follow