How to host an IContextMenu, part 2 – Displaying the context menu

devblogs.microsoft.com/oldnewthing/20040922-00

September 22, 2004

Raymond Chen

Instead of invoking a fixed verb, we'll ask the user to choose from the context menu and invoke the result.

Make these changes to the OnContextMenu function:

```
#define SCRATCH_QCM_FIRST 1
#define SCRATCH_QCM_LAST  0x7FFF


#undef HANDLE_WM_CONTEXTMENU
#define HANDLE_WM_CONTEXTMENU(hwnd, wParam, lParam, fn) \
    ((fn)((hwnd), (HWND)(wParam), GET_X_LPARAM(lParam), GET_Y_LPARAM(lParam)), 0L)


// WARNING! Incomplete and buggy! See discussion
void OnContextMenu(HWND hwnd, HWND hwndContext, int xPos, int yPos)
{
  POINT pt = { xPos, yPos };
  if (pt.x == -1 && pt.y == -1) {
    pt.x = pt.y = 0;
    ClientToScreen(hwnd, &pt);
  }

  IContextMenu *pcm;
  if (SUCCEEDED(GetUIObjectOfFile(hwnd, L"C:\\Windows\\clock.avi",
                  IID_IContextMenu, (void**)&pcm))) {
    HMENU hmenu = CreatePopupMenu();
    if (hmenu) {
      if (SUCCEEDED(pcm->QueryContextMenu(hmenu, 0,
                           SCRATCH_QCM_FIRST, SCRATCH_QCM_LAST,
                           CMF_NORMAL))) {
        int iCmd = TrackPopupMenuEx(hmenu, TPM_RETURNCMD,
                                    pt.x, pt.y, hwnd, NULL);
        if (iCmd > 0) {
          CMINVOKECOMMANDINFOEX info = { 0 };
          info.cbSize = sizeof(info);
          info.fMask = CMIC_MASK_UNICODE;
          info.hwnd = hwnd;
          info.lpVerb  = MAKEINTRESOURCEA(iCmd - SCRATCH_QCM_FIRST);
          info.lpVerbW = MAKEINTRESOURCEW(iCmd - SCRATCH_QCM_FIRST);
          info.nShow = SW_SHOWNORMAL;
          pcm->InvokeCommand((LPCMINVOKECOMMANDINFO)&info);
        }
      }
      DestroyMenu(hmenu);
    }
    pcm->Release();
  }
}
```

The first change addresses the first issue brought up in the discussion of the WM_CONTEXTMENU message and fixes the HANDLE_WM_CONTEXTMENU message.

The second change addresses the second issue, and that's the special handling of keyboard-invoked context menus. When we receive a keyboard-invoked context menu, we move it to the (0, 0) position of our client area. This keeps the context menu displayed in a vaguely sane position. (If we were a container with objects, it would have been better to display the context menu over the selected sub-object.)

The third change actually does what we're talking about: Displaying the context menu to the user, collecting the result, and acting on it.

You are certainly familiar with the TrackPopupMenuEx function. Here we use the TPS_RETURNCMD flag to indicate that the item the user selected should be returned by the function instead of being posted as a WM_COMMAND to our window.

This highlights the importance of the fact that SCRATCH_QCM_FIRST is 1 and not zero. If it were zero, then we wouldn't be able to distinguish between the user selecting item zero and the user cancelling the menu.

Once we are confident that the user has selected an item from the menu, we fill out a CMINVOKECOMMANDEX structure, specifying the user's selection in the two verb fields and indicating the invocation point via the ptInvoke member.

Note that when you invoke a command by menu ID, you must specify the **offset** of the menu item relative to the starting point passed to IContextMenu::QueryContextMenu. That's why we subtracted SCRATCH_QCM_FIRST.

When you run this program, you may notice that some things don't quite work. Most obviously, the Open With and Send To submenus don't work, but there are more subtle bugs too. We'll address them over the next few days.

Raymond Chen

Follow