Subject: Re: DockCtrl (A dockable window widget for U++)
Posted by Oblivion on Fri, 28 Sep 2007 19:40:19 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 28 September 2007 21:59

Well, what is your need for WM_NCLBUTTONUP and what is the bug?

Ok, let me explain. Dockwindows are basically derived from "DockableCtrl" class, which is actually a derivative of TopWindow. So it is possible to use it at the same time as a ctrl, and as a window. In their "docked state" they act as a ctrl and in their "float state" they are detached (dragged/teared) from their panes and activated as tool windows. When a dockwindow is teared off/dragged in solid window dragging(with the Dockablectrl::Float()/FloatEx method), it has to receive mouse messages immediately, because it is from then on in a "dragging state". This can be done easily by sending a fake WM_NCLBUTTONDOWN message with HTCAPTION parameter; by this command, system is faked to believe that mouse is dragging a "window" from it's titlebar. Here is the sequence.

1. Drag Dock from its "dragbar"(from the title bar of the dock)
   This is done by LeftDrag(). So the left mouse is, from now on, "down"
2. Remove() dock from its pane.
3. Open it as a child toolwindow under the cursor position.
4. Immediately Send a WM_NCLBUTTONDOWN message explicitly (with HTCAPTION as wparam). (because, the Lmousebutton is still down.)
5. Move window.
6. Drop (theoretically, the window should send WM_NCLBUTTONUP so we could get the drag/drop state, but it doesnt; so we use the mouse patch we already installed when we invoke the Float() command and send the WM_NCLBUTTONUP command manually).

This is the actual tear-off/drag sequence.

(As you can see this in the executable example (0.49.7a) I've posted. This is how a solid window drag sequence can be made possible on Win32, Or maybe I don't know the way, ayn suggestions will be very useful). But unfortunately windows from XP on does not send the proper WM_NCLBUTTONUP message to the window's message pump when the left button is released on the NC area! So it couldn't "drop" at first time when mouse button is released. It is clearly stated in the MSDN that it does, but in fact it doesn't. It is a well known bug/behaviour.

So as you see, in this case the standard U++ leftup or leftdown won't work because those messages only handle AFAIK the client area. Is there something I don't know about it?

Here is the problem and the hook solution is explained:

http://www.codeguru.com/cpp/misc/misc/windowsmessaging/artic le.php/c3885/

I use a modified version of this mouse hook, So the article could give you a clear idea about the problem, solution and (hopefully) a better solution suggestion from you

Here is the modified code

```
//=====================================================
// Original code by Robert Wiejak, (c) 2001.
//
//
#ifdef PLATFORM_WIN32
LRESULT DockCtrl::Win32MessagePatch(int code, WPARAM wParam, LPARAM lParam)
{

 if(code == 0)
 {
  PMOUSEHOOKSTRUCT mousehookstruct = (PMOUSEHOOKSTRUCT) lParam;

  DockableCtrl* dock = ::__dctrlptr->ptr->FindDockWindow(mousehookstruct->hwnd);
  {

   switch(wParam)
   {
    case WM_NCLBUTTONDOWN:
      if(dock && mousehookstruct->wHitTestCode == HTCAPTION) dock->Dragging(TRUE);
     break;

    case WM_NCLBUTTONUP:

      if(dock) dock->Dragging(FALSE);
     break;

    case WM_LBUTTONUP:
      {
       DockableCtrl *target = NULL;
       if(dock && dock->IsDragged()) target = dock;
       else
       target = ::__dctrlptr->ptr->FindDraggedWindow();

       if(target)
       {
        ::PostMessage(target->GetHWND(), WM_NCLBUTTONUP, HTCAPTION,
MAKELONG(mousehookstruct->pt.x, mousehookstruct->pt.y));
        target->Dragging(FALSE);
       }
      }
     break;

    default:
     break;
   }
  }
 }
//=================================================
```
Regards.