Mouse, cursors, and keyboard

CS134

Chris Pollett

Nov. 17, 2004.

Outline

- Mouse Messages
- Cursor Tools
- The Mouse Wheel
- Focus and Autofocus
- The Keyboard

Mouse Messages

- Windows generates a number of mouse related messages: WM_MOUSEMOVE, WM_ONLBUTTONDOWN, etc.
- In MFC, these are handled by the corresponding handler functions: OnMouseMove, OnLButtonDown, etc.
- One can select which class is responsible for handling a message by going to class view, right clicking the name of the class, and selecting Properties and then clicking on the message button.
- Your class will then get a default message handler written in it which you can rewrite

Example

```
void CPopView::OnLButtonDown(UINT
   nFlags, CPoint point)
{
    SetCapture();
    pgame->onLButtonDown(this, nFlags, point);
}
```

Calling the OnDraw Method

- Your code should never call OnDraw directly.
- Instead, can do things like pDoc->UpdateAllViews(NULL);
- Or what we might do in the case of a mouse event is call Invalidate();
- Actually, what happens in Pop:
  CPopView::OnMouseView might call
  CPopDoc::UpdateAllViews which calls
  CPopView::OnUpdate, which calls
  CPopView::Invalidate which calls
  CPopView::OnDraw

Cursor Tools

- As might have multiple views, data for cursor type stored in CPopView. i.e., it has an HCURSOR _hCursor handle.

Changing the Cursor

- Most functions related to the cursor are that -- Windows functions, not MFC methods.
- For example, one can change the cursor's appearance using SetCursor(HCURSOR hCursor).
- We might change the cursor in response to menuitem events that we added. Say the menu item Viewl Pin Cursor was selected. Then in our CPopView::OnViewPinCursor method we might call SetCursor.

Making a Cursor in the Resource Editor

- To create a special looking cursor, one can do to Project| Add resource| … Then select the kind of resource, i.e., a Cursor, you want to add. You then get an Image Editing window in which you can work on your cursor.
- Be aware:
  - Cursors are only black and white
  - Want most of a cursor to be transparent, so be sure to use one of the transparent colors.
  - Cursors have an associated Hot Spot that you can change in the Resource Workshop.
  - You probably also want to change the ID of your cursor to something other than IDC_CURSOR1

Getting a Cursor Resource

- To now uses the beautiful cursor resource you've created you need to do something like: HCURSOR _hMyCursor; _hMyCursor = LoadCursor(IDC_MYCURSOR);
- This might be done at the start of CPopApp::InitInstance();
- Want to keep _hMyCursor public so CPopView has access to it. We can then use as:

```
void CPopView::OnViewMyCursor()
{ _hCursor = ((CPopApp*)::AfxGetApp())->_hMyCursor;
}
```

Using Cursor Tools

- OnMouseMove(UINT nFlags, CPoint point) -- nFlags contains info on which buttons are down, point says location of mouse.
- For example, nFlags &MK_LBUTTON checks if left button down.

The Mouse Wheel

- Generates a WM_MOUSEWHEEL event can handle for instance in CPopView::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
- In Pop only used right now to scroll through the different tool types.

Focus and Autofocus

- CPopView::OnSetCursor has code to have the view under the cursor be automatically in focus when that option is selected in the menu.

The Keyboard

- To handle you can handle either on OnChar or OnKeyDown events