Subject: Re: Using IDLitWindow::AddWindowEventObserver
Posted by Dick Jackson on Sat, 15 Oct 2011 02:23:32 GMT
View Forum Message <> Reply to Message

On Thursday, October 13, 2011 12:11:35 AM UTC-7, Dick Jackson wrote:
> Hi all,
>
> I'm trying to write a program where one top-level widget base will contain several Object Graphics windows, using my own window class subclassing from IDLitWindow, with my own handlers for mouse and keyboard events. I want to provide the *option* of linking the windows so that these events can be received by one window and broadcast to the others for synchronized zooming/panning/etc.
>
> I've read every word I can find about this in online help (it didn't take long, see IDLitWindow Methods) and I'm quite sure I've done this right, creating another class of observer object which is added using AddWindowEventObserver (and I'd remove them when I want to turn off linking). Now, the events are indeed coming to the window that I do the actual event on (OnWheel is called, for instance), but nothing else seems to happen. The methods on the observer objects (I have an OnWheel handler as described) never seem to be called at all. Am I wrong to expect that I'd get the original event method call on the first window and the method calls to all added observers as well?

I figured it out. The 'observer' event method calls appear to be a feature of the IDLitWindow class 'OnWheel' (etc.) handlers, so to get them to happen, you need to have the original event handler pass the call up to 'IDLitWindow::OnWheel' (etc.). It all then works as advertised, very nicely.

In case anyone's interested, one more trick to getting this 'linked window' idea working: you need to make sure that when you duplicate the event to the other windows, they don't each try to duplicate it back (... and they'll tell two friends, and so on, and so on...). I made a little class for the observer objects, and below I show how to ensure the observer events work as expected:

```
;;    This is from a handler in my class that represents a top-level
;;    window that contains several draw widgets, created like this:

self.wDraws = List()
FOR viewI=0, self.nViews-1 DO $
   self.wDraws -> Add, Widget_Draw(wRow, Graphics_Level=2, $
                              Classname='MyIDLitWindowSubclass')

;;    When the draw widgets have been realized (IDs in 'self.wDraws'):

self.oWindows = List()
self.oObservers = List()
FOREACH wDraw, self.wDraws DO BEGIN
   Widget_Control, wDraw, Get_Value=oWindow
   self.oWindows -> Add, oWindow
   ;;    Make an observer object which will send duplicate events to
   ;;    this window.
```

```
  self.oObservers -> Add, MimicObserver(oWindow)
ENDFOREACH
nWindows = N_Elements(self.oWindows)
IF nWindows GT 1 THEN $ ; Add observers which duplicate this window's
                ; events to all other windows
  FOREACH oWindow, self.oWindows, windowI DO $
    oWindow -> AddWindowEventObserver, $
      (self.oObservers)[Where(IndGen(nWindows) NE windowI)]

;---------------------------------------

PRO MimicObserver::OnWheel, Window, X, Y, Delta, Modifiers
self.window -> OnWheel, X, Y, Delta, Modifiers, /DontPropagate
END ;; MimicObserver::OnWheel

;---------------------------------------

FUNCTION MimicObserver::Init, oWindow
self.window = oWindow
RETURN, 1B
END ;; MimicObserver::Init

;---------------------------------------

PRO MimicObserver__Define
struct = {MimicObserver, $
       window: Obj_New()}
END ;; MimicObserver__Define

;---------------------------------------

PRO MyIDLitWindowSubclass::OnWheel, X, Y, Delta, Modifiers, $
  DontPropagate=dontPropagate

;;   <do the wheel-event handling and redrawing here>

IF ~Keyword_Set(dontPropagate) THEN $
  self -> IDLitWindow::OnWheel, X, Y, Delta, Modifiers

END ;; MyIDLitWindowSubclass::OnWheel

;---------------------------------------
```

I hope this helps someone save a little time figuring this out.

Cheers,
-Dick

Dick Jackson Software Consulting
Victoria, BC, Canada
www.d-jackson.com