Subject: Function graphics and IDL widgets
Posted by wlandsman on Mon, 08 Jun 2015 02:41:45 GMT
View Forum Message <> Reply to Message

I had written before (http://tinyurl.com/n9nmzvw )about my conceptual problems understanding the use of function graphics in a widget. Standalone function graphics (e.g. p = PLOT(/test) ) appear within a "widget" of their own, giving the user the option, for examples to write a hardcopy, zoom in and out, and add annotation. What happens when this PLOT() command is sent to a WIDGET_WINDOW?

The answer is not complicated but I still found it useful to write the following notes. For reference, I give below the notes the simple example from the IDL documentation of using PLOT() within a WIDGET_WINDOW(). --Wayne

1. By default, all the mouse capabilities in standalone function graphics (as described in http://www.exelisvis.com/docs/graphicswindowinterface.html ) are available in a WIDGET_WINDOW. You can double click on any graphic (e.g. an axis, or a plot line) to get its property sheet. You can zoom into the plot either with the scroll wheel, or by pressing SHIFT and the left mouse button. You can even click on an axis, and press the "Delete" button to erase it (though I am hard pressed to think of how this might be useful).

On Windows machines, you can instead make a "dumb" window (so that none of the mouse commands have any effect), by adding SENSITIVE = 0 to the WIDGET_WINDOW() call. But this keyword seems to have no effect on my Mac, and, in fact, I haven't figured out how to make a dumb WIDGET_WINDOW() on my Mac.

2. On the other hand, the buttons that appear in standalone function graphics do not appear in a WIDGET_WINDOW(), and if you want them, then you must duplicate their functionality using standard widget commands (e.g. WIDGET_BUTTON()). In particular, if you want to reset a zoomed plot, or direct the plot to an hardcopy format, you must write the interface yourself.

3. Resizing a widget with function graphics is somewhat simpler than with direct graphics because the content is remembered during a widget resizing. So when one resizes a window in response to a top level resizing event,

widget_control, baseplot, xsize=new_xsize, ysize=new_ysize

there is no need to have stored the content in a pixmap, or to reload the image.

Of course, in a complicated widget with multiple windows one still has to compute the new windows sizes. If Harris/Exelis ever does update their widget capabilities, one of my two main wishes is that widget resizing become transparent to the user. ( My other main wish, of course, is to able to use color with WIDGET_TEXT() )

4. Finally, zooming into a window is automatically enabled in function graphics, but sometimes you need to also get this zoom information into the widget program. For example, I am plotting telemetry data in two windows: one showing intensity as a function of time, and one plotting Y vs.

X (centroiding data). I want the user to be able to select a time interval in the time series plot, and then only plot data in this time interval in the centroiding plot. I use a SELECTION_CHANGE_HANDLER keyword
( http://www.exelisvis.com/docs/Selection_Change_Event_Handler .html ) in WIDGET_WINDOW() giving the name of function to be called whenever the user has zoomed in on the plot.

--Wayne

```
pro testwidg
; Create the widgets.
wBase = WIDGET_BASE(/COLUMN)
wDraw = WIDGET_WINDOW(wBase, XSIZE=400, ySIZE=400)
WIDGET_CONTROL, wBase, /REALIZE

; Retrieve the newly-created Window object.
WIDGET_CONTROL, wDraw, GET_VALUE=oWin

; Make sure this is the current window
oWin.Select

p = PLOT(/TEST, /CURRENT, /FILL_BACKGROUND)

return
end
```