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
```

Subject: Re: Function graphics and IDL widgets
Posted by Helder Marchetto on Mon, 08 Jun 2015 11:18:55 GMT
View Forum Message <> Reply to Message

On Monday, June 8, 2015 at 4:41:47 AM UTC+2, wlandsman wrote:
> 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

Hi Wayne,
Regarding point 4) and 1). I use the GraphicsEventAdapter a lot to catch events, process them as I wish and eventually continue with the standard processing.

One way to get the "dumb" window is to use the GraphicsEventAdapter object and set all methods listed in http://www.exelisvis.com/docs/GraphicsEventHandler.html to return, 0.

It requires a bit more than a "sentive = 0", but it does the job.

Regarding the zooming in and out, I've made a short example to use the "esc" key to reset axis and position of the plot. Here it is:

```
function graphicsHandler::KeyHandler, oWin, isASCII, character, keyValue, x, y, press, release,
keyMods
if isASCII && (character eq 27b) && press then begin
   self.refObj.xRange   = self.xRange
   self.refObj.yRange   = self.yRange
   self.refObj.position = self.pos
endif
return, 0
end

function graphicsHandler::Init, refObj
self.refObj = refObj
self.xRange = refObj.xRange
self.yRange = refObj.yRange
self.pos    = refObj.position
return, 1
end

pro graphicsHandler__define
void = {graphicsHandler,$
        inherits GraphicsEventAdapter,$
        refObj:obj_new(),$
        xRange:[0d,0d],$
        yRange:[0d,0d],$
        pos:[0d,0d,0d,0d]}
end

pro testwidg, p=p
 ; 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)
 oWin.event_handler = obj_new('graphicsHandler', p)
end
```

What I did not manage to do, was make a widget_window not sensitive to anything except zooming in and out. That tricked me.

I hope it helps. I found your list of problems/info on the widget_window very useful. I depend on widget_window a lot, so this really helps me. Thanks. Keep us updated if you find workarounds...

Regards,
Helder

PS: I tried to dig in the "plot" function and look for where the window is created. This happens with the function

IDLNotify('wb_create_canvas',dimStr,'')
called in lib/graphics/graphic.pro on line 337.

Unfortunately this is not accessible so I cannot say how the buttons were created in the first place and what they exactly do so to insert them 'artifically' in the widget_window...

Subject: Re: Function graphics and IDL widgets
Posted by markb77 on Mon, 08 Jun 2015 12:07:37 GMT
View Forum Message <> Reply to Message

hi,

In my public IDL library I have a "wmb_plotwindow" object which is essentially a simple resizable plot object. It is also based on the WIDGET_WINDOW. The code is available here:

http://github.com/superchromix/wmb_lib

there is a "test_wmb_plotwin" procedure to demonstrate how to use the code.

best,
Mark
