Subject: Re: Function Graphics overlaid objects on image()
Posted by Helder Marchetto on Fri, 27 Feb 2015 15:42:18 GMT
View Forum Message <> Reply to Message

On Friday, February 27, 2015 at 9:38:58 AM UTC+1, Helder wrote:
> On Thursday, February 26, 2015 at 11:17:24 PM UTC+1, Chris Torrence wrote:
>> On Thursday, February 26, 2015 at 6:26:34 AM UTC-7, Helder wrote:
>>> Hi,
>>> I'm working with function graphics and I'm overlaying objects (lines, polygons) on images. I would like these objects to be linked to the underlying image (pinned if you wish), unless the user explicitly moves these objects with the mouse.
>>>
>>> I would like to avoid having to handle events from the object on my own (pick up event, process, send to all overlaid objects). I have the feeling that there might be an easy solution...
>>>
>>> I have so far tested three conditions (the test code is below):
>>> Data coordinates: in this case the overlays are anchored to the image (if the image is made smaller or moved, the objects are rescaled along). However, it is not possible to move the polylines. The only ways is by clicking on the end-points and changing the line length and angle. However, after this clicking on the image results in a rotation in space of the image... very inconvenient
>>>
>>> Norm or relative coordinates: in this case the objects are unfortunately not anchored to the underlying image.
>>>
>>> Is there a trivial solution to this problem that I haven't picked up?
>>>
>>> Thanks,
>>> Helder
>>>
>>>
>>> pro testFGObjects
>>> ;data coordinates
>>> w1 = window(dimensions=[500,500], window_title='Data coordinates')
>>> i1 = image(dist(500), current=w1)
>>> scale = [i1.xrange[1]-i1.xrange[0],i1.yrange[1]-i1.yrange[0]]
>>> l1 = polyline([0.25,0.75]*scale[0],[0.25,0.75]*scale[1], /data, target=i1)
>>>
>>> ;norm coordinates
>>> w2 = window(dimensions=[500,500], window_title='Norm coordinates')
>>> i2 = image(dist(500), current=w2)
>>> l2 = polyline([0.25,0.75],[0.25,0.75], /norm, target=i2)
>>>
>>> ;relative coordinates
>>> w3 = window(dimensions=[500,500], window_title='Relative coordinates')
>>> i3 = image(dist(500), current=w3)
>>> l3 = polyline([0.25,0.75],[0.25,0.75], /relative, target=i3)
>>>

>>> ;test widget interaction:
>>> ;data coordinates: it is not possible to move the line, only to change
>>> ;   its size by moving the edges. After this the image
>>> ;   becomes 3d. The line rescales/moves with the underlying image
>>> ;norm coordinates: line responds to movements with the mouse. But the line
>>> ;   does not move when rescaling/moving the undlying image
>>> ;relative coordinates: same as norm coordinates
>>> end
>>
>> Hi Helder,
>> You can see my reply to your other post. Right now, there are a couple of solutions for polylines - one is to hack your code to fix the bug. The other solution is to use norm or relative coordinates, but then override the event handler and do the scaling yourself. This is obviously more work.
>> -Chris
>
> Hi Chris,
> I think I've got the point. This now puts me in a tricky position, because I rely heavily on these "lines on images". I have to figure out how to continue with my work... Should I make a workaround until 8.5 comes out or go for the annotations (norm coords) and override the event handler? I've done the event handling in the past with direct graphics and I know I can get it working in FG. However, I would feel a bit frustrated to have to roll back (to data space) in it in the near future...
> So I have a couple of questions to help me decide (sorry for bombarding you with questions!):
>
> 1) Will some graphics (polyline) added to an image() in data space behave as following: one can move and change the size of the graphics with the mouse and then get the new coordinates with getData? This of course includes not having the image rotating...
>
> 2) When will a 8.5 be available? Don't want the minute of the release, rather something like "late 2015" or "early 2020".
>
> 3) Can the fix for these issues that will come out with 8.5 be implemented before the availability of 8.5 (!)? Would patching a FG file or two do? Would this be exportable within as .sav file?
>
> Thanks and sorry for bugging you with all these questions.
>
> Helder

Hi Chris,
I'm trying to work out the overriding of the event handler and I'm trying to do the handling myself. So I create some polygon on an image and when I click on the image and use the mouse wheel to zoom, I want the polygon to zoom along with the image (and stay on top of the same feature). I've produced the code below to demonstrate how this works. If you try this, by running it, clicking on the image and turning the mouse wheel, it will work ok.
But if you change the polygon with the mouse (rescaling it by clicking on a corner and making it bigger or smaller), two thing occur:
1) If you click on the polygon with the mouse to select it again, you get a weird result. The dots

that normally delimit the square for rescaling are not on the square!!! What?
2) Well, the square ends up where it should not be... Except that the misplaced dots are where they are supposed to be!

Any idea what's going on?

```
function WheelEvent, oWin, xPos, yPos, Delta, KeyMods
info = *(oWin.uvalue)
if info.oWin->getSelect() eq info.io then begin
   zoomFactor = (Delta gt 0) ? 1.25d : 1/1.25d
   MousePos = info.oWin->ConvertCoord(xPos, yPos, /device, /to_norm)
   info.sq->getData, xIn, yIn
   xOut = (xIn-MousePos[0])*zoomFactor+MousePos[0]
   yOut = (yIn-MousePos[1])*zoomFactor+MousePos[1]
   info.sq->setData, xOut, yOut
endif
return, 1
end

pro fgRescaleProblem
img = dist(500)
img[200:299,200:299] = max(img)-dist(100)
tlb = widget_base(/column)
wWindow = widget_window(tlb, xsize=500, ysize=500, mouse_wheel_handler='WheelEvent')
widget_control, tlb, /realize
widget_control, wWindow, get_value=oWin
io = image(img, image_dimensions=[500,500], current=oWin, margin=0)
sq = polygon([0.4,0.6,0.6,0.4],[0.4,0.4,0.6,0.6], '-r2', fill_background=0, /normal, target=io)
info = ptr_new({tlb:tlb, oWin:oWin, io:io, sq:sq})
widget_control, tlb, set_uvalue=info
oWin.uvalue = info
xmanager, 'fgRescaleProblem', tlb, /no_block
end
```