
Subject: FG-graphics polyline - moving the line and changing length/slope

Posted by [Helder Marchetto](#) on Tue, 10 Mar 2015 09:23:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I'm drawing a line with `polyline()` on an image in data space. I then handle mouse down, up and mouse motion events to allow the user to move the line (this is otherwise not possible in IDL's current version 8.4) "as it is" (without changing slope and length). To do this, I record at the mouse down event the location and use the difference when the event motion is called to translate. Below is the code to do this (maybe a bit too long, but does the job).

So here is the problem. Now I can't change the length and slope of the line by clicking near the ends of the line.

Of course there is a trick. I can get around by checking the distance between the click and the end of the line. If this is below a value of 0.03 in norm coordinates, then it skips the translation. However, I seem not to be able to get a delta that is consistent with the mouse type (cursor?) that is shown in the graphics window and this makes the whole a "bit" unprofessional...

The question:

Is there an FG-way to recognize if I'm clicking next to the line ends? Somewhere the mouse is changed and I would like to pick up that signal!

[I tried using "grep" to search for `cursor_` in the idl library without coming up with anything useful. So maybe changing the cursor shape is done in some other way in FG]

Thanks,
Helder

Here is the code for this:

```
function motionEvent, oWin, xx, yy, KeyMods
str = *(oWin.uvalue)
if str.closeToEnd then return, 1
if oWin->getSelect() ne str.In then return, 1
if str.buttonDown then begin
    pos = oWin->ConvertCoord(xx, yy, /device, /to_data)
    delta = pos[0:1]-str.lastPos
    str.In->translate, delta[0], delta[1], /data
    str.lastPos = pos[0:1]
    oWin->refresh
    *(oWin.uvalue) = str
endif
return, 1
end
```

```
function MouseDownEvent, oWin, xx, yy, Button, KeyMods, Clicks
str = *(oWin.uvalue)
```

```

pos = oWin->ConvertCoord(xx, yy, /device, /to_data)
str.lastPos = pos[0:1]
str.buttonDown = 1b
nPos = oWin->ConvertCoord(xx, yy, /device, /to_norm)
str.ln->getdata, xData, yData
nData = oWin->ConvertCoord(xData, yData, /device, /to_norm)
minDist = 0.03
str.closeToEnd = min(sqrt((nPos[0]-nData[0,0:1])^2+(nPos[1]-nData[1,0:1])^2)) lt minDist
print, str.closeToEnd
*(oWin.uvalue) = str
return, 1
end

```

```

function MouseUpEvent, oWin, xx, yy, Button, KeyMods, Clicks
str = *(oWin.uvalue)
str.buttonDown = 0b
*(oWin.uvalue) = str
return, 1
end

```

```

pro testMoveLine
img = dist(500)
tlb = widget_base(/column)
wWindow = widget_window(tlb, xsize=500, ysize=500, mouse_down_handler='MouseDownEvent',
mouse_motion_handler='motionEvent', mouse_up_handler='MouseUpEvent')
widget_control, tlb, /realize
widget_control, wWindow, get_value=oWin
io = image(img, image_dimensions=[500,500], current=oWin, margin=0)
;make some objects on top
ln = polyline([0.3,0.7]*500, [0.3,0.7]*500, '-b2', /data, target=io)
oWin.uvalue = ptr_new({ln:ln,oWin:oWin, lastPos:[0d,0d], buttonDown:0b, closeToEnd:0b})
end

```
