Subject: Re: FG-arithmetic error and moving objects
Posted by jimuba on Thu, 16 Apr 2015 00:38:00 GMT
View Forum Message <> Reply to Message

On Thursday, March 5, 2015 at 1:50:33 PM UTC-7, Helder wrote:
> On Thursday, March 5, 2015 at 5:42:29 PM UTC+1, Chris Torrence wrote:
>> On Wednesday, March 4, 2015 at 1:44:23 AM UTC-7, Helder wrote:
>>> On Tuesday, March 3, 2015 at 9:20:20 PM UTC+1, David Fanning wrote:
>>>> Helder writes:
>>>>
>>>> > PS: looks like I got a FG-bug-magnet under my keyboard... I just keep on bumping onto these sort of stuff.
>>>> >
>>>>
>>>> A +1 for this.
>>>>
>>>> Cheers,
>>>>
>>>> David
>>>>
>>>> P.S. Let's just say this is not the most surprising thing I've heard
>>>> today. :-)
>>>>
>>>> --
>>>> David Fanning, Ph.D.
>>>> Fanning Software Consulting, Inc.
>>>> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>>>> Sepore ma de ni thue. ("Perhaps thou speakest truth.")
>>>
>>> Hi,
>>> frustrated by these errors I just found one workaround and I understood where 4-arrows mouse cursor used to move objects appears.
>>>
>>> First the (partial) workaround for not being able to move the line. Easy. Just use
>>> pg2 = polygon([0.25,0.75,0.25]*500.0,[0.25,0.75,0.25]*500.0, '-r2', /data, target=io)
>>> instead of
>>> pl = polyline([0.25,0.75]*500.0,[0.25,0.75]*500.0, '-r2', /data, target=io)
>>>
>>> This only works only partially because now you cannot make the line horizontal or vertical...
>>>
>>> As for the moving of objects, I realized (and this is obvious with the polygon line) that polygons can be moved when the mouse is on the edges of the minimum bounding box.
>>>
>>> As far as I'm concerned, point 1) in the original post is clear: it's not a bug, it's a feature. Objects movement respond to the minimum bounding box, not the edges of the polygon.
>>>
>>> [Note: I switched some months ago to FG graphics. It's easier to make nice plots/images, to save them and I like the idea of not having to handle the movements of complex objects myself.

I'm slowly realizing, at my own loss of time, that appearances have tricked me. Direct graphics has been tested for over ... well before I even knew IDL existed. And I have the feeling I'm getting to be the official bug-finder* of FG (I say this without pretenses, only frustration)].
>>>
>>> As far as the polyline to polygon fix, I'm not happy with this. I NEED lines to take any direction, including horizontal and vertical.
>>>
>>> And the arithmetic error can be hidden to the user, but it is still suspicious of something going badly wrong.
>>>
>>> Cheers,
>>> Helder
>>
>> Hi Helder,
>>
>> Don't give up!
>>
>> Regarding your issues, the main problem is that we purposely limited the translation, scaling, and rotation when annotations are in the dataspace. The assumption is that if your annotation is in "data" coordinates then you don't want it to move. Perhaps we need to re-think this limitation...
>>
>> Regarding the floating-point arithmetic errors, *sometimes* these are legitimate errors but most of the time they are just harmless noise that can be safely ignored. There are lots of places in the graphics code where a number will get rounded off to zero, or some calculation will make another number go to Infinity or NaN, and then IDL will dutifully report the exception. If you look through the graphics pro code, you will find many instances where we just quietly swallow those. So unless you actually see some visual "badness", I would just ignore them.
>>
>> Cheers,
>>
>> Chris
>> Exelis
>> p.s. keep posting your questions and feedback. We really do listen!
>
> Hi Chris,
> thanks for your answer.
> It's just half time and my favorite team is neither losing nor winning, so I take a few minutes to answer.
> I'll disregard the floating-point arithmetic errors. It's just something that does not go unnoticed. You might want to fit that in the documentation (if it's already inside... sorry, I'm a distracted reader).
>
> As far as the data space and the translations, here are my 2 cents: I've already looked quite some time ago into how to make annotations insensitive to mouse (not movable) and you've just posted the answer also elsewhere (return, 0 to handling functions).
> My suggestion: why not use keywords to lock annotations, maybe with the /lock keyword? Then one can have a consistent (identical) behavior for normal, relative, device and data coordinates.
>

> Maybe I should also say "why" I find the freedom of moving annotations in data space important. I want to place annotations (areas or lines to use to calculate intensity profiles) on images. The nice thing about FG, is that one can zoom easily with the mouse wheel and position the annotation accurately. If the annotation was defined in data space, it will "stay" where it was when zooming. This is good, but once I zoomed in, I would like to adjust the position.
>
> I hope you get the point. If not I can build some example. Of course one can find workarounds and I've been doing this since years with direct graphics, but I find that the added value of FG should just be to handle processes as describe above in a clean and direct way without having to deal with the dirty stuff...
>
> Cheers,
> Helder
>
> PS: Football is back in a few minutes. I'll IDL tomorrow again.

Hi,

It seems that the problem is that a 2-vertex polyline graphic is missing the translate and rotate manipulator handles that you normally get with a polyline that has 3 or more vertices.

With this in mind, a possible workaround to the problem is to add a 3rd vertex that duplicates the 2nd vertex to your polyline. For example:

```
img = image(dist(400,400))
p2 = polyline([100,300],[200,300],'-r2',/data,target=img)
p3 = polyline([100,300,300],[100,200,200],'-b2',/data, target=img)
```

Then when you select the blue line, the rectangular manipulator frame will appear, giving you the ability to move the line without altering the shape or slope. You can also rotate the line without changing it's shape.

Regards,
Jim
Exelis
