
Subject: Re: Object graphics axis

Posted by [davidf](#) on Wed, 27 Oct 1999 07:00:00 GMT

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

Karri Kaksonen (karri.kaksonen@picker.fi) writes:

> I just came back from Davids course about object graphics and it
> occurred to me that a reason why positioning the axis objects is
> so difficult is that the range depends on the location of the
> axis.
>
> The manual says that the range of the axis is set by a vector:
>
> $[-X_{min}/(X_{max}-X_{min}), 1/(X_{max}-X_{min})]$
>
> This may work if the length of the axis is 1.0 in normalized
> coordinates. In the course I chose the length to be 2.0 and in
> order to get it right I just tried out different values until
> I found it to be closer to:
>
> $[-1-X_{min}*2/(X_{max}-X_{min}), 2/(X_{max}-X_{min})]$
>
> I thought about this on my flight home last night and what I am
> afraid of is that the -1 in the first element may actually depend
> on where the axis is drawn on the screen. My location of the axis
> was at $[-1.0, -1.0]$. If this is the case then RSI should do
> something about it before version 5.3 comes out. Otherwise you have
> to update the range vector every time you reposition the axis.
>
> I run some tests at home and found out that the Location-keyword
> sets two coordinates out of three. And the coordconv sets the
> third coordinate, range and the scale.
>
> range = [-50 150]
> normalized_len = 1.6
>
> Location = [not_valid y_pos z_pos]
> scale = normalized_len/(range[1]-range[0])
> Xcoordconv = [x_pos - range[0]*scale, scale]
>
> This does not sound logical to me. I would rather see:
>
> Location = [x_pos, y_pos, z_pos]
> Scale = normalized_len/(range[1]-range[0])
> Offset = -range[0]*scale
>
> Perhaps RSI could add "scale" and "offset" to make the objects behave
> like the user expects them to work.

I agree completely that this does not work like the user expects it to work. You, Ben, me...everyone I know... has found it incredibly difficult to understand. And the IDL documentation is decidedly unhelpful. I found it frustrating too that the NORM_COORD routine supplied in the object/examples directory, and referred to in the documentation, also only works with data that starts at zero. In other words, with data created with the ever popular DIST function. :-)

After spending literally hours trying to make *real* data show up in object graphics plots I threw up my hands and threw myself on the mercy of the IDL programmers who wrote the code. They supplied me with an algorithm, which with a few slight modifications to make it actually work, reliably creates the translation and scaling factors necessary to place a real data range into a particular view.

The relevant piece of code is this:

```
scale = [((position[0]*range[1])-(position[1]*range[0])) / $  
(range[1]-range[0]), (position[1]-position[0])/(range[1]-range[0])]
```

where "range" is a two-element vector with the minimum and maximum data range, and "position" is a two-element vector with the minimum and maximum location in the arbitrary coordinate space of the view. This is the code that can be found in NORMALIZE:

<http://www.dfanning.com/programs/normalize.pro>

Why this code hasn't found its way back to RSI is beyond me, but I know for a fact that I can't write object programs without it. I've tried.

Cheers,

David

P.S. This is not to say you *can't* write object programs without NORMALIZE. I've seen plenty of object programs from RSI which seem to be working just fine. I'm only suggesting that the people who wrote those programs seem to be a hell of a lot smarter than I am. :-)

--

David Fanning, Ph.D.

Fanning Software Consulting

Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Toll-Free IDL Book Orders: 1-888-461-0155
