"David Fanning" <davidf@dfanning.com> wrote in message
news:MPG.1404ed3fe278116b989bd8@news.frii.com...
> ...I moved the tick marks
> up 0.001 unit in Z and all was well. Are you saying
> I could *not* do this if I didn't have the image in
> a polygon?

Ah yes...or no...I can't handle these double negatives and it's only 2 pm!

If two overlapping atoms (plots, text, polygons, etc--but not images) are at
the same level in Z, then the question of which one obscures the other does
not have a consistent answer. The IDL documentation says that it depends on
the order in which they were put into the graphics tree (which determines
the order in which they are drawn) but that's not really true. It depends on
the destination device. If the destination device is the display screen,
then it depends on whether the hardware or software renderer is selected (on
my system anyway). It may depend on the video card. Whatever it depends on
it can't be relied on.

The solution to robust overlap control is to position things in Z. The
minimum displacement required for this purpose is equal to the distance
between the front & rear clipping planes, divided by 65536. Deduction:
16-bit integers are involved somewhere! I got this number from the
IDLgrLegend code. (Actually I find it is best to double it just to be on the
safe side. I get the impression that some renderers draw text just
slightly above or below its nominal position relative to lines & polygons
etc.)

Texture-mapped polygons obey these rules but stand-alone images don't. With
your old image-based colour bar, you would see the tick marks where they
overlap the image as long as they were added to the colour bar object after
the image.

I've added an example program called mgh_example_image to my WWW site to
illustrate how drawing order & Z position affect visibility of text objects
relative to an image.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield/
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand