Subject: Re: Object Graphics Vector Output
Posted by Karl Schultz on Wed, 29 Sep 2004 22:41:13 GMT
View Forum Message <> Reply to Message

"Randall Skelton" <randall.skelton@gmail.com> wrote in message
news:1096488546.459088.133580@h37g2000oda.googlegroups.com.. .
>> My experience is quite different (apart from the text-size bug I
>> mentioned in another posting in this thread). I find, like the
> original
>> poster, that postscript vector output from IDL 6.1 looks very poor
>> mainly due (I think) to lines having very narrow widths.
>
> Having learned the bulk of my object graphics knowledge from reading
> through Mark's code, I am either falling into the same pitfalls or I
> generally agree with his comments. In the code below, I observe the
> following:
>
> (1) The box and the vertical/horizontal lines are of slightly different
> thickness. Pay particular attention to the bottom of the box when
> previewing or printing
>
> (2) The ordering of the filled polygon objects is consistently
> incorrect. No matter what I try, I cannot get filled polygon (within
> the same model) to lie beneath polylines. I would like black lines
> around the red box!
>
> I realise these are a little pedantic compared with the original post
> where an axis itself is of different width but it is a bit of an
> annoyance nevertheless. Depending on the resolution settings, the
> lines can get quite thick and the difference becomes more apparent.

Mark and I worked out the axis problem over private e-mail. In his case, he had some mask polygons that were partly obscuring his lines, making them look very thin. It turns out that an IDL 6.1 bug caused the lines to be sorted improperly and so the polygons drew on top of the lines, when the lines should have drawn after the polygons. So, the issue there was not really related to line widths. And the bug has been fixed for the next release.

What version of IDL are you talking about here?

If 6.1, then the same line sorting bug may be to blame.

Here are some other thoughts:

1) I've seen some discussion on the net about PostScript viewers and alpha blending. The discussions varied a lot depending on what viewing software was involved. But I've noticed that a recent version of GSview has an alpha

control under the Media->Display Settings menu that controls anti-aliasing.
If I play with this, I can vary the apparent line widths quite a bit, while
displaying the same PS file. I can't be sure that this is an issue, but you
might take a look And we have to take care to distinguish between viewer
features/problems and the PostScript code itself.

2) I know we don't like to look inside PostScript files, but in this case,
doing so can shed some light. First, both IDL 6.0 and 6.1 set the line
width only once in the PS files generated by your test program. For 6.0, it
is set to 1, and in 6.1 it is set to 1.35. So, the PostScript code is
specifying the drawing of lines with a constant width. And in 6.1, they are
a bit thicker, which may help the "too thin" problem a bit. That all being
said, note that both versions issue a "0.736272 0.572656 scale" command near
the top. This is to map your square views onto your rectangular paper, I
suspect. This scale may be getting applied to the line widths as well by
the PostScript interpreter, which may explain differences you are seeing in
horizontal and vertical line widths. This might be an error on IDL's part,
but I'd have to ponder that, as well as lookup what PostScript does with
line widths in this case.

3) In your test program, you are putting the polygon first in the model and
using DEPTH_OFFSET to get it to draw "behind" the lines in the IDLgrWindow.
Depth buffer controls like DEPTH_OFFSET simply don't work in vector graphics
because there is no depth buffer. IDL tries its best to sort primitives by
depth in vector output, but it simply can't do all the things that a real
raster depth buffer can do. Prior to 6.1, this wasn't explained very well
in the docs, but in 6.1, it is explained much better. One thing you can do
is draw the polygon last (put it last in the model). The default depth
buffer "tie" rule is that ties go to whoever drew a pixel first, and vector
output follows the same rule. So, if you draw the lines first, they will
win the tie when the polygon is drawn. And of course, you can also move the
polygon back a bit in Z, but that's only an easy option if your scene is
never rotated out of the 2D plane.

Karl