Subject: Object Graphics Vector Output
Posted by Iam A. Lurker on Wed, 15 Sep 2004 02:30:28 GMT
View Forum Message <> Reply to Message

In the trivial code below, I get two axes of different thickness and I
am not sure why. Has anyone else run into this when using the NOTEXT
keyword with IDLgrAxis?

<grumble>Looking back through my notebook (i.e. 6 months), I seem to
have an awful lot of postscript plots with multiple axes where the
first 'notext' axis ismuch thicker than the others</grumble>

Thanks,
Iam

Linux & OS X; IDL 6.0 & 5.6

```
oModel = obj_new('IDLgrModel')

; Removing NOTEXT makes the axes the same thickness?!?
oXAxis1 = obj_new('IDLgrAxis', 0, /NOTEXT, THICK=1, TICKLEN=0.02)
oYAxis1 = obj_new('IDLgrAxis', 1, /NOTEXT, THICK=1, TICKLEN=0.02)

oModel -> Add, oXAxis1
oModel -> Add, oYAxis1

oView = obj_new('IDLgrView', viewplane_rect=[-0.5,-0.5,2,2])
oView -> add, oModel

oWindow = obj_new('IDLgrWindow', GRAPHICS_TREE=oView,
DIMENSIONS=[500,500])
oWindow -> erase
oWindow -> draw

oClip = obj_new('IDLgrClipboard', GRAPHICS_TREE=oView, UNITS=2, $
DIMENSIONS=[15,15], resolution=[0.01,0.01])
oClip -> draw, filename='test.eps', postscript=1, vector=1

end
```

Subject: Re: Object Graphics Vector Output
Posted by Karl Schultz on Thu, 16 Sep 2004 16:08:05 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message

news:cib4abehg1@newsreader.mailgate.org...
> Karl Schultz wrote:
>>
>> Yes, this does seem to be fixed in IDL 6.1. We fixed a lot of issues
with
>> vector output in this release.
>>
> And introduced a bug: text appears several times too large when a view
> has its PROJECTION property set to 2 (perspective). Randall Skelton
> brought this to my attention today and I narrowed it down. I've reported
> it to support@rsinc.com, but this seemed like a good place to mention it
:-)

Another change in vector output for 6.1 is that IDL text objects are drawn
with the appropriate text primitives, instead of a set of solid triangles.
It is possible that there's a problem with using these primitives with
perspective projections. One can use the VECT_TEXT_RENDER_METHOD keyword on
the IDLgrClipboard::Draw method or IDLgrPrinter::Draw method to force the
output back to triangles. It may help.

Karl

Subject: Re: Object Graphics Vector Output
Posted by Mark Hadfield on Thu, 16 Sep 2004 23:13:39 GMT
View Forum Message <> Reply to Message

Karl Schultz wrote:
> "Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message
> news:cib4abehg1@newsreader.mailgate.org...
>
>
>> And introduced a bug: text appears several times too large when a view
>> has its PROJECTION property set to 2 (perspective). Randall Skelton
>> brought this to my attention today and I narrowed it down. I've reported
>> it to support@rsinc.com, but this seemed like a good place to mention it
>
> :-)
>
> Another change in vector output for 6.1 is that IDL text objects are drawn
> with the appropriate text primitives, instead of a set of solid triangles.
> It is possible that there's a problem with using these primitives with
> perspective projections. One can use the VECT_TEXT_RENDER_METHOD keyword on
> the IDLgrClipboard::Draw method or IDLgrPrinter::Draw method to force the
> output back to triangles. It may help.

Yes this does fix it (well, work around it).

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

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

Subject: Re: Object Graphics Vector Output
Posted by Karl Schultz on Wed, 29 Sep 2004 23:15:31 GMT
View Forum Message <> Reply to Message

"Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message
news:10lmebnk3nkjcb@corp.supernews.com...
>
snip

> 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.

I have to take part of this one back, but it is a good thing.

DEPTH_OFFSET actually works with vector output. When DEPTH_OFFSET=1 in the
testcase, the polygon appears first in the PS file, because it is
effectively deeper in depth. So, it draws first, and then the lines draw on
top of it. Effectively there were no depth buffer ties between the lines
and the polygon. When DEPTH_OFFSET=0, the polygon appears after the lines
in the PS file, so it will draw over the lines. This gives the same visual
appearance as on the screen - the polygon draws first, because it is first
in the model, but the depth buffer enforces the "tie" rule that ties go to
the first prim to draw at a given pixel, and so the depth buffer rejects the
lines.

But the other more advanced depth buffer controls such as
DEPTH_TEST_FUNCTION can't be emulated in vector output.

Karl

Subject: Re: Object Graphics Vector Output
Posted by Randall Skelton on Fri, 01 Oct 2004 22:28:53 GMT
View Forum Message <> Reply to Message

Thanks for the reply Karl. Here is another one for you... Any ideas
why models containing vector (aka hershey) text have unsightly boxes
around them when using vector postscript output?

Cheers,
Randall

-- cut --

```
;  ============================================================ ==
; Create the plot canvas (roughly one letter sized sheet of paper)

CanvasDim = [19,24.5]
oCanvasView = obj_new('IDLgrView', UNITS=2, DIMENSION=CanvasDim, $
VIEWPLANE_RECT=[0,0,1,1])

;  ============================================================ ==
; Add some vector text
Text1Dim = [CanvasDim[0], 0.75] ; cm
Text1Loc = [0, 10] ; cm

oSmFont1 = obj_new('IDLgrFont', 'Hershey', SIZE=6)

oVer = obj_new('IDLgrText', !version.release, LOCATION=[0.9,0.0], $
FONT=oSmFont1, ALIGN=1)
date = string(systime(/julian), format="(C(CYI4, '-', CMoI2.2, '-', " +
$
"CDI2.2, X, CHI2.2, ':',CMI2.2, ':', CSF07.4))")
oDate = obj_new('IDLgrText', date, LOCATION=[0.1,0.0], $
FONT=oSmFont1, ALIGN=0)

oText1Model = obj_new('IDLgrModel', NAME='Text1')
oText1Model -> add, oVer
oText1Model -> add, oDate

oText1View = obj_new('IDLgrView', UNITS=2, DIMENSION=Text1Dim, $
LOCATION=Text1Loc, /TRANSPARENT, VIEWPLANE_RECT=[0,0,1,1])
oText1View -> add, oText1Model

;  ============================================================ ==
; Add some true-type text
Text2Dim = [CanvasDim[0], 0.75] ; cm
Text2Loc = [0, 12] ; cm

oSmFont2 = obj_new('IDLgrFont', 'Times', SIZE=10)

oVer = obj_new('IDLgrText', !version.release, LOCATION=[0.9,0.0], $
FONT=oSmFont2, ALIGN=1)
date = string(systime(/julian), format="(C(CYI4, '-', CMoI2.2, '-', " +
$
"CDI2.2, X, CHI2.2, ':',CMI2.2, ':', CSF07.4))")
oDate = obj_new('IDLgrText', date, LOCATION=[0.1,0.0], $
FONT=oSmFont2, ALIGN=0)

oText2Model = obj_new('IDLgrModel', NAME='Text2')
oText2Model -> add, oVer
oText2Model -> add, oDate
```

```
oText2View = obj_new('IDLgrView', UNITS=2, DIMENSION=Text2Dim, $
LOCATION=Text2Loc, /TRANSPARENT, VIEWPLANE_RECT=[0,0,1,1])
oText2View -> add, oText2Model

;  ============================================================ ==
; Group the objects
ogroup = obj_new('IDLgrViewgroup')
ogroup -> Add, oCanvasView
ogroup -> Add, oText1View
ogroup -> Add, oText2View

owin = obj_new('IDLgrWindow', GRAPHICS_TREE=ogroup, UNITS=2, $
DIMENSIONS=CanvasDim)
owin -> draw

oclip = obj_new('IDLgrClipboard', GRAPHICS_TREE=ogroup, UNITS=2, $
DIMENSIONS=CanvasDim, RESOLUTION=[0.03,0.03])
oclip -> draw, FILENAME='test_vector.eps', /VECTOR, /POSTSCRIPT
oclip -> draw, FILENAME='test_bitmap.eps', /POSTSCRIPT

end
```