
Subject: PNGs without X?

Posted by [Dr. Sven Geier](#) on Tue, 21 Jan 2003 19:57:19 GMT

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

Heya all

I have a routine that gathers data from various places, then plots it, then uses tvrd(/true) to read the plot into an array and then writes it out as a .png graphics file. This all works fine so far.

Since I rarely ever look at the graphs as they're plotted (the pngs go to a web-directory) I switched the plotting to a pixmap, i.e. the idl-process now opens its own window with the /pixmap parameter (and also xsize and ysize and such) and thus the whole shebang works quite nicely without ever showing anything on the screen.

Now in the process of automation I'd like to move the whole operation to a server box that does not have X running. Unfortunately it turns out that these "pixmap" are *X*-pixmap and that I can't open them without X.

So now I'm looking for some way to plot data into/onto a PNG (or GIF or JPG or whatever) without having an actual graphic server running. I played around with plotting into a PS device and using ghostscript to convert that into a JPG, but not only do the colors not come out right (the usual PS stuff) but also the fonts are all off (the !P.font=0 for the 'x' device is designed to be readable on a 400x400 window, the ps-fonts become illegible when I reduce a PS-plot to that resolution) and similar problems.

Anybody know how to do a standard run-of-the mill X-type plot into a file or into an array or into RAM or such?

Thanks in advance...

-- SG

--

--

Catch a couple Z's before responding...

Subject: Re: PNGs without X?

Posted by [R.Bauer](#) on Fri, 31 Jan 2003 19:05:01 GMT

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

Dr. Sven Geier wrote:

Dear Sven,

did you have thought about the trick printing into a ps file and then using convert or pstotimg (which I prefer) to get the ps file converted in whatever you want. (You have an unix system so you can use these routines with a spawn too)

This method if it's suitable for you has another effort too. The postscriptfonts are much better readable as the image fonts.

regards

Reimar

>
> Heya all
>
> I have a routine that gathers data from various places, then plots it,
> then uses tvrd(/true) to read the plot into an array and then writes it
> out as a .png graphics file. This all works fine so far.
>
> Since I rarely ever look at the graphs as they're plotted (the pngs go to
> a web-directory) I switched the plotting to a pixmap, i.e. the idl-process
> now opens its own window with the /pixmap parameter (and also xsize and
> ysize and such) and thus the whole shebang works quite nicely without ever
> showing anything on the screen.
>
> Now in the process of automation I'd like to move the whole operation to a
> server box that does not have X running. Unfortunately it turns out that
> these "pixmap" are *X*-pixmap and that I can't open them without X.
>
> So now I'm looking for some way to plot data into/onto a PNG (or GIF or
> JPG or whatever) without having an actual graphic server running. I played
> around with plotting into a PS device and using ghostscript to convert
> that into a JPG, but not only do the colors not come out right (the usual
> PS stuff) but also the fonts are all off (the !P.font=0 for the 'x' device
> is designed to be readable on a 400x400 window, the ps-fonts become
> illegible when I reduce a PS-plot to that resolution) and similar
> problems.
>
> Anybody know how to do a standard run-of-the mill X-type plot into a file
> or into an array or into RAM or such?
>
> Thanks in advance...
>

> -- SG
>
>

--

Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
<http://www.fz-juelich.de/icg/icg-i/>

=====

a IDL library at Forschungszentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro.html

Subject: Re: PNGs without X?

Posted by [Dr. Sven Geier](#) on Fri, 14 Feb 2003 19:21:33 GMT

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

Boy, I hope top-posting is alright...

A while ago I posted the thing appended below -- and got a few responses. I dug through the various methods and none of them really work. I had noted before that postscript isn't really an option and the same reasons really apply to plotting into the Z-buffer: neat trick that, and I may well use that in the future; but the output I get is nowhere near the output that I get when plotting to an X window. The fonts are entirely different, the colors are different, pretty much everything is different.

The color part, I suppose, is an outcropping of another problem, that I never managed to find a solution to: I find it very hard (to say the least) to use a certain given color in IDL. It seems like the simplest thing in the world to have a graph with four lines and I want one of them to be red and one green and one yellow and one blue. However there doesn't seem to be a mechanism in IDL(?) that accomplishes such a thing. Before I knew of 24-bit displays, I had a self-made colortable (with `tv!ct()`) in which I knew index 1 is red and index 5 is blue and such, but the first time I tried this on a true-color visual, I learned something.

The problem is that I need to be certain what the colors are before the plot, since there are some comments added to the data (via `xyouts`) that refer to "red line = such-n-such" and such. For the same reason I must be able to rely on a certain plot geometry (which I am consistently not getting when plotting in the Z-buffer) because the positions of things are carefully computed. The entire thing has been evolved over years to work just right and if you suddenly run it with a larger character size (like the Z-buffer seems to do) it'll all blow up.

As an example, here's some random data and label:

```
data = tan(findgen(5))
plot,data,/nodata
oplot,data,color='ff'x
oplot,data,psym=5
xyouts, 1,1.53,' <- triangle'
```

On my 24-bit display this draws a few red lines, white triangles and labels one of the triangles with "triangle". If I do this on a "window,xsize=350,ysize=350" and then I do the same thing into the Z-buffer following David's website (great site, by the way. Or terrible site, as you may look at it: Many idl programmers lose much productive time browsing through it... :) I get something that is "kinda like" the original, except that the plotting area has a different size, the fonts are different and the color is gone - pretty much as different as a plot can be while still showing the same thing.

!P.font = -1 or 0 give the same result in the Z-buffer, !p.font=1 gives a slightly prettier (but less legible) one, but all three are different from their counterparts that are plotted into the X-buffer and not one of them is as clear and legible as the p.font=0 in X-windows.

If I plot into the PS device and turn the result into an image like this:

```
gs -q -sDEVICE=ppm -r100 -sOutputFile=- idl.ps | pnmflip -r90 | pnmscale \
-xsize=350 -ysize=350 | pnmtojpeg > test.jpg
```

I do not only get no colored line (because all colors 255 and up are just "white" to postscript, apparently), but the result is transparent (i.e. white background) and the fonts are unreadable. It's easy to fix the color in this simple example by searching for 'setrgbcolor' in the postscript file, but once you have a bunch of colored lines and text, that gets pretty tricky to say the least.

I am writing this all down in a last attempt to see if someone has the magic trick "how to get consistent colors and something like an X-buffer without X" - but if there is none, I'd like to say thanks to those who posted before anyways; there were a few good ideas...

-- S

-

Catch a couple Z's before responding...

Reimar Bauer wrote:

> Dr. Sven Geier wrote:
>
>
> Dear Sven,
>
> did you have thought about the trick printing into a ps file and then using
> convert or pstotimg (which I prefer) to get the ps file converted in
> whatever you want. (You have an unix system so you can use these routines
> with a spawn too)
>
> This method if it's suitable for you has another effort too. The
> postscript fonts are much better readable as the image fonts.
>
>
> regards
>
> Reimar
>
>
>
>>
>> Heya all
>>
>> I have a routine that gathers data from various places, then plots it,
>> then uses tvrd(/true) to read the plot into an array and then writes it
>> out as a .png graphics file. This all works fine so far.
>>
>> Since I rarely ever look at the graphs as they're plotted (the pngs go to
>> a web-directory) I switched the plotting to a pixmap, i.e. the
>> idl-process now opens its own window with the /pixmap parameter (and also
>> xsize and ysize and such) and thus the whole shebang works quite nicely
>> without ever showing anything on the screen.
>>
>> Now in the process of automation I'd like to move the whole operation to
>> a server box that does not have X running. Unfortunately it turns out
>> that these "pixmap" are *X*-pixmap and that I can't open them without
>> X.
>>
>> So now I'm looking for some way to plot data into/onto a PNG (or GIF or
>> JPG or whatever) without having an actual graphic server running. I
>> played around with plotting into a PS device and using ghostscript to
>> convert that into a JPG, but not only do the colors not come out right
>> (the usual PS stuff) but also the fonts are all off (the !P.font=0 for
>> the 'x' device is designed to be readable on a 400x400 window, the
>> ps-fonts become illegible when I reduce a PS-plot to that resolution) and
>> similar problems.
>>
>> Anybody know how to do a standard run-of-the mill X-type plot into a file

>> or into an array or into RAM or such?

>>

>> Thanks in advance...

>>

>> -- SG

>>

>>

>

--

Catch a couple Z's before responding...

Subject: Re: PNGs without X?

Posted by [JD Smith](#) on Fri, 14 Feb 2003 19:41:13 GMT

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

On Fri, 14 Feb 2003 12:21:33 -0700, Dr. Sven Geier wrote:

> Boy, I hope top-posting is alright...

>

>

> A while ago I posted the thing appended below -- and got a few
> responses. I dug through the various methods and none of them really
> work. I had noted before that postscript isn't really an option and the
> same reasons really apply to plotting into the Z-buffer: neat trick
> that, and I may well use that in the future; but the output I get is
> nowhere near the output that I get when plotting to an X window. The
> fonts are entirely different, the colors are different, pretty much
> everything is different.

>

> The color part, I suppose, is an outcropping of another problem, that I
> never managed to find a solution to: I find it very hard (to say the
> least) to use a certain given color in IDL. It seems like the simplest
> thing in the world to have a graph with four lines and I want one of
> them to be red and one green and one yellow and one blue. However there
> doesn't seem to be a mechanism in IDL(?) that accomplishes such a thing.
> Before I knew of 24-bit displays, I had a self-made colortable (with
> tvlct()) in which I knew index 1 is red and index 5 is blue and such,
> but the first time I tried this on a true-color visual, I learned
> something.

>

> The problem is that I need to be certain what the colors are before the
> plot, since there are some comments added to the data (via xyouts) that
> refer to "red line = such-n-such" and such. For the same reason I must
> be able to rely on a certain plot geometry (which I am consistently not
> getting when plotting in the Z-buffer) because the positions of things
> are carefully computed. The entire thing has been evolved over years to

```

> work just right and if you suddenly run it with a larger character size
> (like the Z-buffer seems to do) it'll all blow up.
>
> As an example, here's some random data and label:
>
> data = tan(findgen(5))
> plot,data,/nodata
> oplot,data,color='ff'x
> oplot,data,psym=5
> xyouts, 1,1.53,' <- triangle'
>
> On my 24-bit display this draws a few red lines, white triangles and
> labels one of the triangles with "triangle". If I do this on a
> "window,xsize=350,ysize=350" and then I do the same thing into the
> Z-buffer following David's website (great site, by the way. Or terrible
> site, as you may look at it: Many idl programmers lose much productive
> time browsing through it... :) I get something that is "kinda like" the
> original, except that the plotting area has a different size, the fonts
> are different and the color is gone - pretty much as different as a plot
> can be while still showing the same thing.
>
> !P.font = -1 or 0 give the same result in the Z-buffer, !p.font=1 gives
> a slightly prettier (but less legible) one, but all three are different
> from their counterparts that are plotted into the X-buffer and not one
> of them is as clear and legible as the p.font=0 in X-windows.
>
> If I plot into the PS device and turn the result into an image like
> this:
>
> gs -q -sDEVICE=ppm -r100 -sOutputFile=- idl.ps | pnmflip -r90 | pnmscale
> \ -xsize=350 -ysize=350 | pnmtjpeg > test.jpg
>

```

```

device,decomposed=0 ; David's tip of the Holocene Epoch
tv!ct,255b,0b,0b,1
set_plot,'PS'
device,/color,bits_per_pixel=8,filename='test.eps',/encapsulated
data = tan(findgen(5))
plot,data,/nodata
oplot,data,color=1
oplot,data,psym=5
xyouts, 1,1.53,' <- triangle'
device,/close

```

Then use ghostscript.

JD

Subject: Re: PNGs without X?

Posted by [David Fanning](#) on Fri, 14 Feb 2003 21:46:18 GMT

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

Dr. Sven Geier (svezn@srzl.caltech.edu) (who I *believe* is using IDL) writes:

> A while ago I posted the thing appended below -- and got a few responses. I
> dug through the various methods and none of them really work. I had noted
> before that postscript isn't really an option and the same reasons really
> apply to plotting into the Z-buffer: neat trick that, and I may well use
> that in the future; but the output I get is nowhere near the output that I
> get when plotting to an X window. The fonts are entirely different, the
> colors are different, pretty much everything is different.

Remind me again why PostScript isn't an option? You seem to want beautiful output. I think I read somewhere that 95% of all scientists surveyed agreed PostScript was the way to go if this is where you thought you were headed. :-)

The fonts will be decidedly different. PostScript fonts are not the same as your machine's hardware fonts. With the demise of the Next computer, that's pretty much a fact of life for all of us.

There is no reason for colors to be different, however, unless you are letting IDL decide what colors to use, in which case you are probably getting what you deserve. :-)

The PostScript device is an 8-bit device, and it sounds like your display is a 24-bit device, so there will probably have to be a difference in how you specify color. In 8-bit devices, we have to use a color table. In 24-bit devices, we can (if we choose) specify a color directly.

> The color part, I suppose, is an outcropping of another problem, that I
> never managed to find a solution to: I find it very hard (to say the least)
> to use a certain given color in IDL.

It can be a little complicated, I guess, until you understand it a bit. The trick is to realize that the normal display device is quite a bit different from other graphics devices. Most of the time, if you are using another graphics device, you have to prepare the color table ahead of time, before you do your plotting. For example, I typically set up drawing colors like this:

```
; Set up drawing colors.
```

```
backgroundColor = FSC_Color('white', 1)
```

```
axisColor = FSC_Color('navy', 2)
dataColor_1 = FSC_Color('red', 3)
dataColor_2 = FSC_Color('green', 4)
```

Now, I can use these four drawing colors everywhere and I always see the same thing. (Naturally, I choose a white background because PostScript likes white backgrounds and it is rather more complicated to convince it that something else is more aesthetic. But see below.)

Note that FSC_COLOR is smart enough NOT to load a color table if you don't need one (e.g. you are in a 24-bit device). Then the function just returns a 24-bit value that *is* the representation of the color you want. So you get the same color on either 8-bit or 24-bit devices, With color decomposition turned on or off.

- > It seems like the simplest thing in
- > the world to have a graph with four lines and I want one of them to be red
- > and one green and one yellow and one blue. However there doesn't seem to be
- > a mechanism in IDL(?) that accomplishes such a thing. Before I knew of
- > 24-bit displays, I had a self-made colortable (with tvlct()) in which I
- > knew index 1 is red and index 5 is blue and such, but the first time I
- > tried this on a true-color visual, I learned something.

You probably learned you should have loaded your color table with FSC_Color, I hope. :-)

```
ok = FSC_COLOR(['red', 'green', 'blue', 'cyan'], Indgen(4)+1, $
  Decomposed=0)
```

- > The problem is that I need to be certain what the colors are before the
- > plot, since there are some comments added to the data (via xyouts) that
- > refer to "red line = such-n-such" and such. For the same reason I must be
- > able to rely on a certain plot geometry (which I am consistently not
- > getting when plotting in the Z-buffer) because the positions of things are
- > carefully computed. The entire thing has been evolved over years to work
- > just right and if you suddenly run it with a larger character size (like
- > the Z-buffer seems to do) it'll all blow up.

If by "just right" you mean "using device coordinates" then I am beginning to see why the PostScript device has been ruled out. :-)

But the Z-buffer should work if you make the buffer the correct size (DEVICE, SET_RESOLUTION=[?,?]), where ? and ? refer to the size of your normal

display window.

```
> As an example, here's some random data and label:
>
> data = tan(findgen(5))
> plot,data,/nodata
> oplot,data,color='ff'x
> oplot,data,psym=5
> xyouts, 1,1.53,' <- triangle'
>
> On my 24-bit display this draws a few red lines, white triangles and labels
> one of the triangles with "triangle". If I do this on a
> "window,xsize=350,ysize=350" and then I do the same thing into the Z-buffer
> following David's website (great site, by the way. Or terrible site, as you
> may look at it: Many idl programmers lose much productive time browsing
> through it... :) I get something that is "kinda like" the original, except
> that the plotting area has a different size, the fonts are different and
> the color is gone - pretty much as different as a plot can be while still
> showing the same thing.
```

Humm, well, yes. We have a few problems here.
But what about this program:

```
ok = FSC_COLOR(['red', 'olive', 'blue', 'white', 'sienna'], $
  Indgen(5)+1, ColorStructure=c)
data = tan(findgen(5))
plot,data,/nodata, Color=c.red, Background=c.white
oplot,data,color=c.blue
oplot,data,psym=5,color=c.sienna
xyouts, 1,1.53,' <- triangle',color=c.olive
```

This programs looks the same on 24-bit or 8-bit devices, with color decomposition turned on or off, in the Z-buffer, on the display, and in PostScript, at least so far as color is concerned.

Fonts will be different, of course, especially if you are using hardware fonts (!P.Font=0). And you will have to set Device, Decomposed=0 to see the Z-buffer results in color on your display (or use TVIMAGE rather than TV to display the Z-buffer image), but the color is there and is correct.

If text is positioned in data or normalized coordinates (as it is here), it should be in the same relative location in all plots, etc. All in all, a pretty workable solution, it seems to me. It even comes out correct on my printer, using the PRINTER device, if I am careful to load the

color table *BEFORE* I send it off to the printer.
(Just another device-specific nuance).

```
ok = FSC_COLOR(['red', 'olive', 'blue', 'white', 'sienna'], $
  Indgen(5)+1, ColorStructure=c, DECOMPOSED=0)
thisDevice = !D.Name
Set_Plot, 'PRINTER', /Copy
data = tan(findgen(5))
plot,data,/nodata, Color=c.red, Background=c.white
oplot,data,color=c.blue
oplot,data,psym=5,color=c.sienna
xyouts, 1,1.53,' <- triangle',color=c.olive
Device, /Close
Set_Plot, thisDevice
```

> I do not only get no colored line (because all colors 255 and up are just
> "white" to postscript, apparently),

Yes, PostScript is an 8-bit device. It's sort of like your old
computer. :-)

> but the result is transparent (i.e.
> white background) and the fonts are unreadable.

I'm not sure why the fonts are unreadable. PostScript fonts
are generally acknowledged to be as good as they get.

PostScript graphics operations are primarily vector operations.
Filling a background color is a raster operation. You can get
a background color other than white in PostScript, but you will have to
ask for it deliberately. For example, this will give you a plot
with a nice beige color as a background:

```
thisDevice = !D.Name
Set_Plot, 'PS'
Device, Color=1, Bits=8, Filename='test.ps'
ok = FSC_COLOR(['red', 'olive', 'blue', 'beige', 'sienna'], $
  Indgen(5)+1, ColorStructure=c, DECOMPOSED=0)
data = tan(findgen(5))
polyfill, [0,1,1,0,0], [0,0,1,1,0], /normal, color=c.beige
plot,data,/nodata, Color=c.red, /noerase
oplot,data,color=c.blue
oplot,data,psym=5,color=c.sienna
xyouts, 1,1.53,' <- triangle',color=c.olive
Device, /Close
Set_Plot, thisDevice
```

> I am writing this all down in a last attempt to see if someone has the magic
> trick "how to get consistent colors and something like an X-buffer without
> X"

Well, FSC_COLOR is *almost* like magic. :-)

Cheers,

David

--

David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: PNGs without X?
Posted by [R.Bauer](#) on Sat, 15 Feb 2003 13:07:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dear Sven,

we have not changed our colorsystems to 24bit because at the time it was possible we have had already developed our own plot environment based on the idl plot, contour etc.

One of the ideas was to get always nearly the same result on screen as on postscript.

In some cases it is very difficult and it doesn't work in the way we like to have it. But it is in the most cases very good.

One other idea was if you have done a script for a printout on screen it should not take more time from the user to get a quite good printout for a publication.

We are using the decomposed feature of the device command. So our platforms can have 8, 16 or 24 bit color devices. It doesn't matter.

Our plotenvironment could be completely loaded by this file,
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_bin/plotprepare.sav

As you know sav file are the same handled as pro files. By the first call it is loaded.

At the moment this is a bit newer as the rest of the library, because it

is based on the actual library we are using here. I will spent later some time in updating the published library.

The sources which are under GPL developed and a description in german is available from

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro.html

some examples how to use this tool you can find at

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl_html/idl_work_idl_work.examples.category.htm#5

For example to show a bit about the syntax:

(plotprepare.sav should be in the same path as this example)

Screen output:

```
pro test
plotprepare,plot

plotinit,plot

data = tan(findgen(5))

plot.color=plot.color_nc.red
plot.psym=0
plotxy,plot,x=findgen(5),y=data

plot.psym=5
plotxy,plot,x=findgen(5),y=data

xyouts, 1,1.53,' <- triangle'
plotend,plot
end
```

Or may be you like to give a try to following example:

```
pro test
plotprepare,plot

plotinit,plot

data = tan(findgen(5))
```

```
plot.color=plot.color_nc.red
plot.psym=0
plotxy,plot,x=findgen(5),y=data
```

```
plot.psym=5
plotxy,plot,x=findgen(5),y=data
```

```
xp_text,plot,' <- triangle'
plotend,plot
end
```

You have to reload your source,
a line similar to this one is added into your source

```
xyouts,/norm,0.23239437,0.52600000,' <-
triangle',color=plot.color_nc.blue,
align=0.00000,charsize=1.4*plot.charsize_norm_factor
; xp_text,plot,' <- triangle' ;===== automaticly replaced =====
```

postscript output:

```
pro test
plotprepare,plot
plot.psflag=1
plotinit,plot
```

```
data = tan(findgen(5))
```

```
plot.color=plot.color_nc.red
plot.psym=0
plotxy,plot,x=findgen(5),y=data
```

```
plot.psym=5
plotxy,plot,x=findgen(5),y=data
```

```
xyouts, 1,1.53,' <- triangle'
plotend,plot
end
```

conversion to png
> pstointg -flip r90 -aaliastext idl.ps idl.png

Some more infos to plotprepare.

plotprepare was compiled by "compile" from the icg library.
After loading which a call to plotprepare_info() you get some
informations about the state of the used routines

```
print,(plotprepare_info()).routines
```

plotprepare

Build Date: Sat Feb 15 13:13:25 2003

Compiled Procedures:

\$MAIN\$

AFORMAT /usr/local/icg/icg/idl_source/idl_links/aformat.pro

Tue Aug 25 21:06:52 1998

ALT_INHALT_DEF /usr/local/icg/icg/idl_source/idl_links/alt_inhalt_def.pro

Tue Jan 25 09:40:04 2000

BOXAREA2 /usr/local/icg/icg/idl_source/idl_links/boxarea2.pro

Mon May 6 09:39:06 2002

BUILD_STRUCTURE /usr/local/icg/icg/idl_source/idl_links/build_structure.pro

Fri Feb 18 11:38:10 2000

.
.
.

Some of the routines included are not developed from ICG members,
so the licensing could be different.

For licensing information please have a look at

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro.html

best regards

Reimar

Dr. Sven Geier wrote:

> Boy, I hope top-posting is alright...

>

>

> A while ago I posted the thing appended below -- and got a few responses. I

> dug through the various methods and none of them really work. I had noted

> before that postscript isn't really an option and the same reasons really

> apply to plotting into the Z-buffer: neat trick that, and I may well use

> that in the future; but the output I get is nowhere near the output that I

> get when plotting to an X window. The fonts are entirely different, the

> colors are different, pretty much everything is different.

>

> The color part, I suppose, is an outcropping of another problem, that I

> never managed to find a solution to: I find it very hard (to say the least)

> to use a certain given color in IDL. It seems like the simplest thing in

> the world to have a graph with four lines and I want one of them to be red

> and one green and one yellow and one blue. However there doesn't seem to be

> a mechanism in IDL(?) that accomplishes such a thing. Before I knew of

> 24-bit displays, I had a self-made colortable (with tvlct()) in which I

> knew index 1 is red and index 5 is blue and such, but the first time I

> tried this on a true-color visual, I learned something.

>

> The problem is that I need to be certain what the colors are before the

> plot, since there are some comments added to the data (via xyouts) that

> refer to "red line = such-n-such" and such. For the same reason I must be

> able to rely on a certain plot geometry (which I am consistently not

> getting when plotting in the Z-buffer) because the positions of things are

> carefully computed. The entire thing has been evolved over years to work

> just right and if you suddenly run it with a larger character size (like

> the Z-buffer seems to do) it'll all blow up.

>

> As an example, here's some random data and label:

>

```
> data = tan(findgen(5))
> plot,data,/nodata
> oplot,data,color='ff'x
> oplot,data,psym=5
> xyouts, 1,1.53,' <- triangle'
```

>

> On my 24-bit display this draws a few red lines, white triangles and labels

> one of the triangles with "triangle". If I do this on a

> "window,xsize=350,ysize=350" and then I do the same thing into the Z-buffer

> following David's website (great site, by the way. Or terrible site, as you

> may look at it: Many idl programmers lose much productive time browsing

> through it... :) I get something that is "kinda like" the original, except

> that the plotting area has a different size, the fonts are different and

> the color is gone - pretty much as different as a plot can be while still

> showing the same thing.

>

> !P.font = -1 or 0 give the same result in the Z-buffer, !p.font=1 gives a

> slightly prettier (but less legible) one, but all three are different from

> their counterparts that are plotted into the X-buffer and not one of them
> is as clear and legible as the p.font=0 in X-windows.
>
> If I plot into the PS device and turn the result into an image like this:
>
> gs -q -sDEVICE=ppm -r100 -sOutputFile=- idl.ps | pnmflip -r90 | pnmscale \
> -xsize=350 -ysize=350 | pnmtojpeg > test.jpg
>
> I do not only get no colored line (because all colors 255 and up are just
> "white" to postscript, apparently), but the result is transparent (i.e.
> white background) and the fonts are unreadable. It's easy to fix the color
> in this simple example by searching for 'setrgbcolor' in the postscript
> file, but once you have a bunch of colored lines and text, that gets pretty
> tricky to say the least.
>
> I am writing this all down in a last attempt to see if someone has the magic
> trick "how to get consistent colors and something like an X-buffer without
> X" - but if there is none, I'd like to say thanks to those who posted
> before anyways; there were a few good ideas...
>
>
> -- S
>
> -
> Catch a couple Z's before responding...
>
>
> Reimar Bauer wrote:
>
>
>> Dr. Sven Geier wrote:
>>
>>
>> Dear Sven,
>>
>> did you have thought about the trick printing into a ps file and then using
>> convert or pstotimg (which I prefer) to get the ps file converted in
>> whatever you want. (You have an unix system so you can use these routines
>> with a spawn too)
>>
>> This method if it's suitable for you has another effort too. The
>> postscriptfonts are much better readable as the image fonts.
>>
>>
>> regards
>>
>> Reimar
>>

```

>>
>>
>>
>>> Heya all
>>>
>>> I have a routine that gathers data from various places, then plots it,
>>> then uses tvrd(/true) to read the plot into an array and then writes it
>>> out as a .png graphics file. This all works fine so far.
>>>
>>> Since I rarely ever look at the graphs as they're plotted (the pngs go to
>>> a web-directory) I switched the plotting to a pixmap, i.e. the
>>> idl-process now opens its own window with the /pixmap parameter (and also
>>> xsize and ysize and such) and thus the whole shebang works quite nicely
>>> without ever showing anything on the screen.
>>>
>>> Now in the process of automation I'd like to move the whole operation to
>>> a server box that does not have X running. Unfortunately it turns out
>>> that these "pixmap" are *X*-pixmap and that I can't open them without
>>> X.
>>>
>>> So now I'm looking for some way to plot data into/onto a PNG (or GIF or
>>> JPG or whatever) without having an actual graphic server running. I
>>> played around with plotting into a PS device and using ghostscript to
>>> convert that into a JPG, but not only do the colors not come out right
>>> (the usual PS stuff) but also the fonts are all off (the !P.font=0 for
>>> the 'x' device is designed to be readable on a 400x400 window, the
>>> ps-fonts become illegible when I reduce a PS-plot to that resolution) and
>>> similar problems.
>>>
>>> Anybody know how to do a standard run-of-the mill X-type plot into a file
>>> or into an array or into RAM or such?
>>>
>>> Thanks in advance...
>>>
>>> -- SG
>>>
>>>
>>
>

```

--

Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
 Forschungszentrum Juelich
 email: R.Bauer@fz-juelich.de
