Subject: Re: Color Frustration
Posted by davidf on Fri, 28 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astrosun.tn.cornell.edu> has either gone completely
bonkers or Ithaca has turned into the Boulder of the East. In
either case, his story gets curiouser and curiouser.

But...I'm pretty sure I know what is going on. :-)

JD writes:

```
>
>   tvlct,0B,255B,0B,!D.N_COLORS-1 ;load green into top color
>   tvlct, r,g,b,/get
>   r_orig=r & g_orig=g & b_orig=b & r_curr=r & g_curr=g & b_curr=b ;ughh
>
```

> I do hate that colors common block.

Right-oh. Get rid of them. Completely. Do away with them. This is
the thing that is frying your goose (er, something!).

No way you should have TVLCT, r, g, b, /Get *AND* a common
block. Stuff and nonsense. One or the other, my man. And I say,
chuck the block.

All this copying to and fro is doing you in. Putting you under, so
to speak. Casting stars in your eyes, if you catch my drift. Colors
floating about.

You want colors, go *get* them:

```
   TVLCT, r, g, b, /GET
```

You want to load them, load them:

```
   TVLCT, r, g, b
```

Or, what I prefer:

```
   LOADCT, 5, NColors=200, Bottom=10 ; or whatever,
```

But don't be messin' around with no *common* block. No, sir, you're
better than that! Steer clear of that riff-raff.

> Well, if you made it this far, it's a small miracle,and a testament to
your IDL
> fanatasism.

Do you believe I am not only reading this, but responding after a six-pack
and working 14 hours on my IDL manual today. I have to *seriously* get a life!

> And yes, David, it is a "fantastic story", but aren't all the interesting
ones?

Yes, indeed. And this one has cheered me up more than most. :-)
I'll buy the beer next time I see you , JD!

Cheers!

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438   Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Color Frustration
Posted by J.D. Smith on Fri, 28 Feb 1997 08:00:00 GMT

View Forum Message <> Reply to Message

David Fanning wrote:
>
> JD Smith <jdsmith@astrosun.tn.cornell.edu> writes:
>
> [Some fantastic story about colors, IDL, and mysterious happenings ....]
>
> This can one of only two things:
>
> (1) TGIF has started earlier than usual at Cornell this afternoon, or
>
> (2) programmer error.
>
> I personally believe (on the basis of previous posts from JD) that
> reason (1) is most likely. :-)

Alright, I did stop in for one beer, but I really don't think that put me under.
As a test of my mental accuity, I translated 'Stairway to Heaven' from English into
French, and back into English again. Only two references to the lord of the
underworld surfaced.

As for some more details, I skimped because it's one of these 20 headed beasts that makes you sigh, and wish you had only known then what you know now.

Anyway, I'll, try and fill in.

I'm not using cursor (let's not be ridiculous), but I am, as I mentioned, doing some heavy swapping in and out with the pixmap. If the pixmap maintained its own colors, or had a different sized color table, that could very well be my problem, but I really don't see how this could be the case.

First, I grab the top color for drawing the symbols, and then I ensure that calls to stretch (or my variant of it, which will leave the colormap beyond a specified top index alone) will not lose the lovely green:

```
 tvlct,0B,255B,0B,!D.N_COLORS-1 ;load green into top color
 tvlct, r,g,b,/get
 r_orig=r & g_orig=g & b_orig=b & r_curr=r & g_curr=g & b_curr=b ;ughh
```

I do hate that colors common block.

Then, I get the widget going. It has quite a few bells and whistles, but the next thing that happens of importance is when the user clicks on a pixel. Some magic happens, and it ends up either doing:

```
  ;; erase old
  wset,state.drawwin
  device,COPY=[p(0),p(1),s,s, $
             p(0),p(1),state.pixwin]
```

to erase the plotted symbol (if said symbol is there), or:

```
  plots,p(0),p(1),psym=8,symsize=float(state.zoom>3)/!D.X_CH_SIZE,/DEVICE, $
    THICK=.01,COLOR=!D.N_COLORS-1
```

to plot a new one. Note that the p's aren't really the same p's in these two calls. Another possibility is removing *all* of the plotted symbols, with a total pixmap copy. But anyway, that's about it, as far as this over-plotting feature goes.

And here's what happens. I start clicking. I get a green diamond (no purple hearts, blue diamonds, green clovers). I click again. Another green. I click for my colormap tool A nice smooth gradation of blue to white. I dismiss the tool. I click a few more greens, and then, out of thin air, a (for instance) perwinkle diamond appears. The next might be mauve, or khaki, or seaweed. Almost anything. If I bring up my colormap tool *now*, the crazy colors have taken over the place, and no loading of a new colormap, or stretching, will remove them. They sit there taunting at me. Agh. Meanwhile, If I use my zoom function to show a different piece of the image, things return to normal. The miscreant diamonds toe

the line and become green. And the pattern repeats itself -- a few more clicks and
it's a bad 70's music video again. Note that if I *leave* my color tool up and
commence to bring on the multicolor devils with more clicks, the colormap does not
go crazy too. Only if I dismiss it, and bring up another copy of the colormap tool
do I see the bizarre features. Neither does the whole image jump into electric
tripland when the event occurs. It remains normal throughout (excepting the
individual pixels possessed by minor devils). I may be missing something really
obvious, although the man who would say IDL's color management is obvious may need
to look into some therapy.

Anyway, I'm not sure if there's anything more that's relevant. Note that it
doesn't depend on the color map tool to screw up. It does so even when the tool is
nevered used. Looking at the colormap just revealed to me the severity of the
problem (and that the guache colors seemed glued in place, immovable and
inexorcisable with loadct() or stretch()).

Well, if you made it this far, it's a small miracle,and a testament to your IDL
fanatasism.

And yes, David, it is a "fantastic story", but aren't all the interesting ones?

Thanks in advance,

JD

Subject: Re: Color Frustration
Posted by J.D. Smith on Sun, 02 Mar 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Right-oh. Get rid of them. Completely. Do away with them. This is
> the thing that is frying your goose (er, something!).
>
> No way you should have TVLCT, r, g, b, /Get *AND* a common
> block. Stuff and nonsense. One or the other, my man. And I say,
> chuck the block.
>
> All this copying to and fro is doing you in. Putting you under, so
> to speak. Casting stars in your eyes, if you catch my drift. Colors
> floating about.
>
> You want colors, go *get* them:
>
>   TVLCT, r, g, b, /GET
>

> You want to load them, load them:
>
>   TVLCT, r, g, b
>
> Or, what I prefer:
>
>   LOADCT, 5, NColors=200, Bottom=10 ; or whatever,
>
> But don't be messin' around with no *common* block. No, sir, you're
> better than that! Steer clear of that riff-raft.
>
>> Well, if you made it this far, it's a small miracle,and a testament to
> your IDL
>> fanatasism.
>
> Do you believe I am not only reading this, but responding after a six-pack
> and working 14 hours on my IDL manual today. I have to *seriously* get a life!
>
>> And yes, David, it is a "fantastic story", but aren't all the interesting
> ones?
>
> Yes, indeed. And this one has cheered me up more than most. :-)
> I'll buy the beer next time I see you , JD!
>
> Cheers!
>
> David

Well, I have finally figured out what was wrong, and I should have suspected it from the beginning. It's not the use of the color common block, which, as it turns out, I actually do need, in order to set my plotting color into the "original" color vectors (though I found a cleaner way to implement it). This allows the user to pick any color table he wants,and stretch it and chop it to his hearts content, without affecting my green over-plotting color. More on this later.

The real culprit is the XOR graphics_function mode. I was temporarily thrown off the trail by my unorthodox use of the colors common block.

The scenario is this: A zoom rubber band is drawn with the XOR mode, but since I hadn't anticipated all uses of the draw widget when I wrote it, I tried to optimize things by setting XOR when the user first clicked (initiating the zoom box process), and then unsetting it only after the user released, with arbitrarily many zoom boxes drawn (with XOR) in between. The DEVICE,set_graphics call is actually pretty slow, and so this was speeding things up a good bit. The problem is, if you use this widget together with any others that do any drawing (which I am in this case), you're going to run into

troubles. It sounds obvious, but it is somewhat subtle, since what is critical
is the order in which events arrive down the pipeline. Since this is not always
apparent, I would reccommend that if you use the XOR mode, set it back to copy
immediately after every use, e.g., drawing your selection rubber band. This
does incur a performance penalty, but ensures you won't run into these
difficulties. This is the only guaranteed way to ensure that the XOR mode (or
any other mode for that matter) doesn't "leak" out into the rest of your widget
environment. And the wierd behavior of graphics_function leakage is somewhat
unpredictable, since the ordering of events is not always obvious. What seems
like a perfectly self-contained application of XOR may not remain so when widget
are teamed up!

A side note on the use of the colors common block: The difficulty is that
setting green with:

```
  tvlct,0B,255B,0B,!D.N_COLORS-1 ;load green into top color
```

doesn't affect the common block variables (only the device's color tables). Any
subsequent call to stretch, which uses the colors block original variables
(r_orig etc.) to construct the current ones (r_curr etc.), will obliterate your
green, *unless* it is set into the original variables. In addition, a call like

```
 LOADCT, 5, NColors=200, Bottom=10
```

will similarly lose the green, since it sets the variables with:

```
 r_orig(cbot) = r   ;set a portion of r_orig with loaded table
 g_orig(cbot) = g
 b_orig(cbot) = b
 r_curr = r_orig
 g_curr = g_orig
 b_curr = b_orig
 tvlct,r, g, b, cbot
```

As you can see, even if r,g, and b are only 10 elements vectors, the current
color vectors will be overwritten with the *entire* original ones. The only
option is to set the green into the original, and then reset the originals after
the widget dies, and the only way to accomplish this is to deal with the common
block directly.

Of course, the other option is to maintain a single colormap during your
widget's lifetime, but this limits the user's ability, in my application, to
explore that data.

A further application of this technique, which I am currently putting some
thought into, would allow the total colormap to be chopped up into as many
smaller colormaps as you want, each of which can be stretched and manipulated

(e.g. loading new color tables into) independent of the rest. Then you could have two (or more) images with different colormaps, and a few colors for overplotting, and a colormap tool that could use the active image to determine which sub-colormap to display and edit. Cool, eh?