Subject: Re: A defense of decomposed color
Posted by David Fanning on Tue, 31 Oct 2006 04:17:23 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> I wonder if those of you using decomposed color can persuade me of its
> utility. Though color tables are perfect for image visualization,
> they are wanting for "system" colors for plot symbols, overlays, etc.
> It's frustrating to keep track of them, and different apps have
> different conventions, and can step on each other's feet, causing
> various undesirable effects.
>
> I presume the reason many of us still use undecomposed color is from
> the 8bit heritage, when there was no such thing as decomposed color.
> Do '00FFFF'x-loving people simply assume everyone has a device capable
> of interpreting 24bit, decomposed color (probably about 95% true,
> these days)? How do you handle switching back and forth from
> decomposed (for plot symbols, say) to indexed (for displaying images)?
> Do you find it really solves the headaches associated with saving a
> few colors for drawing in high indices, vs. the added juggling needed
> to switch back and forth among decomposed and non-decomposed color,
> etc.? What happens if you switch to decomposed color on an 8-bit
> display system?
>
> I'm ready to come around to embracing direct color specification with no
> color table intermediary, but I think I need a bit of persuasion.

Two words, JD: TVIMAGE and FSC_COLOR.

The point of using decomposed color is that you can load your
image color tables, use all 256 colors all the time, and *still*
use any color you like for plots and annotation, WITHOUT HAVING
TO SWITCH ANYTHING. At least you can if you use TVIMAGE (or,
alternatively, Liam's IMGDISP) and FSC_COLOR. I can't remember
the last time I switched color models, and I haven't known (or cared)
what color model I've been using for a least the last five years.

Here are just a few of the advantages of using TVIMAGE or IMGDISP:

1. Don't have to worry what color mode you are in, ever. They do
   the *right* thing to get color images correct. They can tell
   the difference between a Windows and UNIX machine.

2. Don't have to worry about how to set the TRUE keyword with
   24-bit images. They can tell the difference between an 8-bit
   image and a 24-bit image. Forget about the TRUE keyword.

3. Can set keywords that preserve the aspect ratio of your image.

4. Can "erase" the display before they draw, similar to other IDL graphics commands.

5. Images can easily be "positioned" in windows with the POSITION keyword, in the same way other IDL graphics commands are positioned, facilitating combining images with other graphics commands. (Adding axes, contour plots, etc.)

6. Displaying multiple images in windows is easy with !P.MULTI.

7. Can be used as a "smart" TV command. For example, just set the TV keyword with TVIMAGE and it acts as dumb as it used to work, except that it gets your image colors right!

8. Works correctly on your display (8-bit or 24-bit) and in all other graphics devices, too, including PostScript.

9. Image "positioning and sizing" is done the same way no matter if you are displaying your image on the display or in PostScript. No more worry about XSIZE and YSIZE keywords!

Here are a few of the advantages of using FSC_COLOR:

1. You have a palette of 104 "named" colors, including all your system colors. If you don't like the ones I provide, FSC_COLOR can read a color file with your own choices.

   Do see your color selection, type this:

   ```
   IDL> color = FSC_COLOR(/Select)
   ```

2. FSC_COLOR works in a color model independent way. If you are on an 8-bit device, FSC_COLOR loads the color in the color table (you can tell it where or it will choose a location) and it will return the location (index). If you are on a 24-bit device, FSC_COLOR will do the 'fe458f'xL thing for you, not bother to load a color, and you will get the SAME COLOR you get on an 8-bit device. No ugly code to puzzle over. The following code works on your display and in PostScript without caring what color model you are using. Plus, you can read it and have a good idea of what colors you SHOULD be seeing!

```
Plot, data, Background=FSC_Color('ivory'), Color=FSC_Color('navy')
OPlot, data, Color=FSC_Color('red')
```

(You see an obvious advantage of using decomposed color here,

since your image color tables never get corrupted by drawing
colors if you don't have to load drawing colors in the color
table.)

While I'm thinking about it, get TVREAD, too. That is the
counterpart to TVIMAGE. What TVIMAGE does to get image ON
your display, TVREAD does to get them OFF your display and
into PNG, JPEG, TIFF, and BMP files. No more having to worry
about what color model you are using and whether you are on
a Windows or UNIX machine (they handle colors differently).
TVREAD also works properly with 8-bit devices, such as the
Z-buffer.

Bottom line. Don't worry about what color model you are
using. Set yourself up in color decomposed mode, use the
proper tools to work in that mode, put your feet up, and
never think about it again. Problem solved! :-)

Cheers,

David

P.S. By the way, an image object that displays itself
with TVIMAGE and contains its own color table can
display itself correctly anywhere and anytime. You don't
even have to remember to load the colors anymore! :-)

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")