Subject: Re: A defense of decomposed color
Posted by JD Smith on Tue, 31 Oct 2006 20:33:38 GMT
View Forum Message <> Reply to Message

On Mon, 30 Oct 2006 21:17:23 -0700, David Fanning wrote:

> 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.
>>
> 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.

Thanks for your thoughts, David. That's the main attraction to me.
You *do* have to switch the decomposed state, but if it's handled
transparently, no one is the wiser.

> 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.

I'm a believer in the power of those tools, but was more interested in
general suggestions for people who want to leverage this
all-things-to-all-people color model in their own code.

> 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.

For those of us with our own display code, can you summarize *how*
they do the right thing? For instance, if you have an 8-bit display,
where you're forced to use color tables, how can they anticipate how
many "system" colors a program may end up needing?

>   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!
```

Does the postscript device demand color-table style colors? That to me
would be a big drawback of decomposed color (unless you consistently use
FSC_COLOR/etc.).

```
>  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! :-)
```

That's a good approach for interactive users on the command line, but
doesn't really constitute a complete solution for applications, which
may need to optimize color usage for various types of data. If the
actual code every time you want to draw a plot element on top of some
image is something like:

```
 device,get_decomposed=gd
 device,/decomposed
 plot,x,y,COLOR='fe458f'x
 device,decomposed=gd
```

not only is this inefficient (taking about 5ms extra for the decompose
fiddling -- not so bad, what if code which is called 100's of times
after a motion event?), but it's not maintainable either. Here's my
ideal scenario:

1. I can use all 256 colors for colormap drawing of images.
2. I can use any additional color as 'ffaabb'x.
3. Will degrade gracefully in 8bit color.
4. Won't require elaborate setup and takedown of the color space
   everytime I want to put something to screen.

I think that last point is what has kept me in color-table land for so
long. I know the combo of TVIMAGE/FSC_COLOR does this for you, but
I'm looking for the secret behind the sauce.

> 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! :-)

Hmmm... is that because TVIMAGE keeps track of it for you?

JD