Subject: Re: Help with moving from 8 to 24 bit colour
Posted by davidf on Wed, 22 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Surendar Jeyadev (jeyadev@wrc.xerox.com) writes:

> After some messing around, I found that the solution is as follows:
>
> device, pseudo_color=8
>
> I am not sure what this does (that is part of today's education!),

What this does is make your expensive new hardware act
like it was last year's model. :-)

But, I agree, this will solve all of your problems and
you can carry on like you have been.

Should you ever have the need or desire to see more
than 256 colors simultaneously, however, I recommend
you use the TrueColor visual class and not the
DirectColor visual class, which appears to be the
default for your machine. DirectColor confuses all
of us. :-)

In a PseudoColor visual class you specify a color by
specifying an index into a color table. Suppose, for
example, we load the color yellow into color index
10 of the color table. Yellow is full red and full
green, but no blue. We could load those values in
the current color table like this:

```
TVLCT, 255, 255, 0, 10
```

To draw a plot in that yellow color, we would do this:

```
Plot, data, Color=10
```

We have specified the color as an *index* into the color
table. If we load a new color into index 10, then the
graphic display is automatically updated, since the
display is "tied to" or "connected to" the index.

If we want a yellow color in a 24-bit environment, we
don't use an index, but we specify the color directly.
That is to say, we must specify the color triple as
a number. Now, how could that be? Well, the number is
going to be a long integer, and we are going to

"decomposed" that number into three components that will specify the red, green, and blue component of the color we want. The lowest 8 bits of the number will be used for red, the next 8 bits for green, and the next 8 bits for blue. So a yellow color will have the lowest 16 bits set, but none of the high bits set. For example,

```
yellow = 2L^16 - 1
Plot, data, Color=yellow
```

Another way to write this is as a hexadecimal number:

```
Plot, data, Color='00FFFF'xL
```

Colors that are expressed directly are not tied or connected to an index, so if we change the colors loaded at a particular index we don't affect display colors at all. The only way they can be changed is if we change them directly by specifying another color.

In IDL you can turn this color decomposition off by using the DEVICE, DECOMPOSED=0 syntax. With color decomposition off, IDL treats the color value as if it were an index and looks the color up in a color table. In other words, IDL "acts" like it was an 8-bit device. Although not to the extend that display colors are automatically updated when we change colors in the color table.

In PV-Wave, it looks to me like you actually *make* it an 8-bit device. In other words, it looks like you can't turn color decomposition off when in 24-bit mode, but you *have* to use decomposed color values.

It's hard to say which protocol is better. I would say it is probably easier for PV-Wave users to use old programs when they move to 24-bit machines, but at the total expense of not being able to take advantage of 24-bit color. (If they want 24-bit color, of course, they can close all windows and change the visual class, but then they have none of the advantages of 8-bit color.)

IDL users get to take advantage of 24-bit colors and they can still use old programs, but they have to learn new techniques for automatic update of their graphics displays when they change color

tables, since in 24-bit color the graphics display
colors are no longer tied to a color index.

I guess you take your picks and take your chances.
But that's color in a nutshell. :-)

Cheers,

David

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