Subject: Understanding Color in IDL
Posted by David Fanning on Wed, 11 Oct 2006 17:29:03 GMT
View Forum Message <> Reply to Message

Folks,

A quick look at my web page statistics for the keywords people use to find my pages convinces me that understanding how color works in IDL is still the number one problem for new users. (Closely followed by how to get decent PostScript output and how to calculate the log base 2 of a number, apparently.)

I've written a number of articles on color over the years, but I've never collected this material into a single coherent explanation. I've attempted to do that now with a chapter I've written for the 3rd edition of my book. (Please don't hold your breath, but I *am* working on it, off and on.)

I've made the chapter available as a PDF file for those who might be interested. If you are still confused after reading it, please let me know. I'd like to get this right before the book is published. :-)

You can find a link to the PDF file here:

http://www.dfanning.com/color_tips/colorchp3.html

Cheers,

David

P.S. Actually, the number one way people come to my web page is searching the Google image archives for "pretty girl oktoberfest", but I don't want to get into the reasons behind that!
--
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.")

Subject: Re: Understanding Color in IDL
Posted by David Fanning on Wed, 11 Oct 2006 21:40:41 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> And why wouldn't they? Decomposed TrueColor is useful when you have a
> specific set of colors in mind, approximating, as you mention, how a
> human would see it with their own eyeballs (for instance as digital
> cameras attempt to do). Most data which flows into IDL isn't obtained
> with devices which attempt any such "as it would appear in the
> real-world" approximation, but rather instruments whose data requires
> some form of visual representation to mesh with the
> evolutionarily-encoded image analysis skills of their human
> operators. Indexed color tables are the fastest route to that sort of
> visualization.

I have no problem with color tables. Couldn't live without
them in every scientific program I ever wrote. My beef is
with a dumb TV command that can't figure out for itself
whether I have a 2D array that should go though a color
table or a 24-bit image that has its color information built
in and should NEVER go through a color table.

I like decomposed color because I want to use that expensive
graphics card I bought and I want my image displayed with one
color table and I want other colors used for my beautiful
graphics display, and I want it all at once. I don't want
to have to piggy back on my image colors or (worse) sacrifice
image colors for drawing colors.

I absolutely agree with you that when a new user can't get
something as simple as a TV command to work, they aren't
exactly well-motivated to move on to something more complicated.
Say a filled contour plot with a hole in it! I wish someone
would spend a couple of days fixing these basic problems.
It's painful to have people look at you like you are nuts
when you try to explain how these things really work.

Cheers,

David
--
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.")

Subject: Re: Understanding Color in IDL
Posted by JD Smith on Thu, 12 Oct 2006 01:05:59 GMT
View Forum Message <> Reply to Message

On Wed, 11 Oct 2006 15:40:41 -0600, David Fanning wrote:

> JD Smith writes:
>
>> And why wouldn't they? Decomposed TrueColor is useful when you have a
>> specific set of colors in mind, approximating, as you mention, how a
>> human would see it with their own eyeballs (for instance as digital
>> cameras attempt to do). Most data which flows into IDL isn't obtained
>> with devices which attempt any such "as it would appear in the
>> real-world" approximation, but rather instruments whose data requires
>> some form of visual representation to mesh with the
>> evolutionarily-encoded image analysis skills of their human
>> operators. Indexed color tables are the fastest route to that sort of
>> visualization.
>
> I have no problem with color tables. Couldn't live without
> them in every scientific program I ever wrote. My beef is
> with a dumb TV command that can't figure out for itself
> whether I have a 2D array that should go though a color
> table or a 24-bit image that has its color information built
> in and should NEVER go through a color table.

No argument there ;).

> I like decomposed color because I want to use that expensive
> graphics card I bought and I want my image displayed with one
> color table and I want other colors used for my beautiful
> graphics display, and I want it all at once. I don't want
> to have to piggy back on my image colors or (worse) sacrifice
> image colors for drawing colors.

Indexed color on TrueColor devices *is* using that fancy graphics card. In fact, there was a brief period when some people really preferred 8bit cards, since "fiddling" the color map to bring out detail in an image (an activity which is thankfully waning in fashionability) operated very quickly, because it just involved writing 256 values to that on-board color table hardware, whereas for 24bit cards (which were just arriving at the time) you had to reencode and rewrite the *entire* image for each "fiddle", over and over. Now cards are fast enough that this objection no longer exists, but you could argue that using color tables on 24bit cards can be very taxing, with all the continuous redraws.

If we could keep multiple separate color tables trivially, I think you'd like color tables just as much. In fact, you could do this now, but it would require some color overseer object to keep track of and edit the entire (single) colormap. I think IDL should do this for us, at a much lower level, with multiple colormaps, deferring the decision of which to apply until the point in the code where the data is

actually run through the lookup table. Somehow it doesn't seem right
to preface every plotting/drawing command with a complete reset of the
internal color table. In that context, decomposed color is a winner.

JD

Subject: Re: Understanding Color in IDL
Posted by Foldy Lajos on Thu, 12 Oct 2006 09:09:42 GMT
View Forum Message <> Reply to Message

On Wed, 11 Oct 2006, David Fanning wrote:

> Folks,
>
> A quick look at my web page statistics for the keywords people
> use to find my pages convinces me that understanding how color
> works in IDL is still the number one problem for new users.
> (Closely followed by how to get decent PostScript output and
> how to calculate the log base 2 of a number, apparently.)
>
> I've written a number of articles on color over the years,
> but I've never collected this material into a single
> coherent explanation. I've attempted to do that now with
> a chapter I've written for the 3rd edition of my book.
> (Please don't hold your breath, but I *am* working on it,
> off and on.)
>
> I've made the chapter available as a PDF file for
> those who might be interested. If you are still confused
> after reading it, please let me know. I'd like to get
> this right before the book is published. :-)
>
> You can find a link to the PDF file here:
>
> http://www.dfanning.com/color_tips/colorchp3.html
>
> Cheers,
>
> David
>
> P.S. Actually, the number one way people come to my web page
> is searching the Google image archives for "pretty girl oktoberfest",
> but I don't want to get into the reasons behind that!
> --
> 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.")
>

Hi David,

what I miss is a summary table for quick reference. Here is what I use for
FL, on all platforms and for all devices (X, WIN, PS, PDF, Z):

```
 ============================================================
==============

   C = color (0...2^24-1)
   c = C mod 256 (0...255)

   r = red   component of C (0...255) = C mod 256
   g = green component of C (0...255) = C / 256 mod 256
   b = blue  component of C (0...255) = C / 256 / 256

   R = lookup table for red   (256 element vector)
   G = lookup table for green (256 element vector)
   B = lookup table for blue  (256 element vector)


                          R,G,B for          R,G,B for
                         decomposed=1        decomposed=0

   24 bit image              r, g, b         R[r], G[g], B[b]

    8 bit image              c, c, c         R[c], G[c], B[c]

   lines, text, etc.      R[r], G[g], B[b]    R[c], G[c], B[c]

 ============================================================
==============
```

It slightly differs from IDL behaviour. Also note the R[r], G[g], B[b]
values, which are decomposed and indexed at the same time (these values
differ from r, g, b only if a non-linear lookup table is used).

regards,
lajos

ps: probably I will change the 'lines with decomposed=1' mode to omit
table lookup, to be compatible with IDL.

Subject: Re: Understanding Color in IDL
Posted by David Fanning on Thu, 12 Oct 2006 12:30:21 GMT
View Forum Message <> Reply to Message

Lajos writes:

> ps: probably I will change the 'lines with decomposed=1' mode to omit
> table lookup, to be compatible with IDL.

I'd hold off for at least one more version. Discussions here
have triggered some interest at ITTVIS and what is
"compatible with IDL" may be changing again. :-)

Cheers,

David
--
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.")