Subject: Re: variable number of colors available on screen
Posted by David Foster on Mon, 24 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Geir Willy Rogde wrote:
>
> When starting idl I get different number of available colors each time.

> Is it possible forcing idl to use 256 colors?
> (I'm on a Unix-workstation.)

You can use an entry like the following in a user's ~/.Xdefaults file or the system Xdefaults file (/usr/openwin/lib/Xdefaults for Solaris 2.X):

```
idl.colors 256
```

The manual suggests that you can also use the following:

```
WINDOW, COLORS=256, /PIXMAP    ; Before any other windows
WDELETE                        ;  are created!
```

which explicitly tells IDL to allocate 256 colors. If enough colors aren't available, it will use a private colormap. This may be your only alternative if you _need_ 256 colors; however, you will have the problem of "color flashing" as input focus changes from IDL to other programs on the screen.

If you can live with fewer colors, you can use a lower number to avoid using the private colormap. You can also specify a negative number to tell IDL to allocate all but the specified number of colors, leaving some for other applications:

```
WINDOW, COLORS=-8, /PIXMAP     ; Leaves 8 colors unallocated
WDELETE
```

You can put these commands in your "idl_startup" file.

The documentation is not clear as to whether this same behavior is achieved using "idl.colors" in the Xdefaults file. It simply states that you can use it to set the number of colors used by IDL. You might want to give it a try. Anyone know if this works?

Having only 100 to 150 colors seems pretty low. Do you have any applications that are grabbing up colors? (One common culprit is Netscape...you can start it with "netscape -install" to avoid this.

Programs written in IDL should not count on there being a constant

number of colors available. You can use !D.N_COLORS or !D.TABLE_SIZE
(don't remember the distinction, someone please remind us) to allow
your programs to account for the number of colors available.

Hope this is useful.

Dave

--

~~~~~~ ~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2200
La Jolla, CA 92037
[UCSD Mail Code 0949]
~~~~~~ ~~~~~~

Subject: Re: variable number of colors available on screen
Posted by davidf on Mon, 24 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Geir Willy Rogde <Geir.Willy.Rogde@fi.uib.no> wrIte:

> When starting idl I get different number of available colors each time.
> The number of colors seems to depend on how many colors spent on
> other processes?

By default IDL requests all the colors that remain in the shared
color map when IDL is started. How many colors this is depends
upon what has happened before IDL starts up. The window
system has *always* allocated some colors from the shared
color map. This is why IDL will never have 256 colors on an
8-bit display if it starts in this mode. But other applications
might have requested colors from the shaded color map as
well. Your word processor, for example, probably needs a
few colors for its windows. Netscape, if it is not configured
properly, can take so many colors from the shared color map
that it will be difficult to get IDL to run properly at all.

(The new Common Desktop Environment that we are seeing on
a lot of new workstations can be a real color hog. By default
it is configured to take a huge number of colors to put up
an absolutely beautiful splash screen. But if you want to do
some work with your computer after you ogh and agh at the
splash screen, I suggest you turn some colors off. This is

a configurable item in the beautiful desktop bar at the
bottom.)

Programs that load color tables (LoadCT, XLoadCT) take color
vectors that are 256-elements in length and automatically
resample them to fit into the number of colors you are
actually using in your IDL session, so that it *looks* like
you are using the same color tables from one session to
the next. (But, of course, a red pixel in *today's* session
could indeed have a different value than the red pixel
in *yesterday's* session, depending upon how many colors
you had in your session yesterday and today. Don't allow
this to effect your image interpretations!) You can determine
how many colors IDL is using by examining the system
variable !D.N_COLORS *after* you have opened an IDL graphics
window.

> Is it possible forcing idl to use 256 colors?

You can force IDL to use any number of colors at all. Here is
how to do it.

The number of colors in the IDL session is selected when the
first IDL graphics window is opened. Once the window is opened,
the number of colors in that IDL session cannot be changed.
Most of us open that first window in what I like to call
an "unconscious" way. That is, we issue a Plot command, or
some other IDL graphics command that opens a display
window for us. If you open a window this way, IDL gets
all of the colors that remain in the shared color table.
That number is, indeed, variable.

If you open the window in a "conscious" way with the WINDOW
command, you will have more control over the number of colors
you have in your IDL session. For example, suppose you want to
have 200 colors in your IDL session. You can open that first
window like this:

```
WINDOW, Colors=200
```

If there are 200 colors available in the shared color map,
IDL will allocate 200 colors from that map. If there are
not 200 colors available in the shared color map, the
window manager will create a *private* color map for
the IDL graphics window with exactly 200 colors in them.
In either case, you will be guaranteed to have 200 colors
in your IDL session.

If you want 256 colors in your IDL session, then you can, of course, open the first graphics window like this:

```
WINDOW, Colors=256
```

But on 8-bit displays there are *never* 256 colors available in the shared color map for IDL to allocate (the window manager has already allocated a few before IDL is invoked). So this command *always* creates a private color map for the IDL graphics windows.

The problem, often, with private color maps is that they sometimes manifest themselves as a "color flashing problem". The window manager is responsible for knowing which window you are working in. (Either because your cursor is in it or because you selected it with the mouse.) It is also responsible for knowing which color table is assciated with that window. If you move your cursor from a window associated with a shared color map to one associated with a private color map, the window manager must load new colors into the hardware color table so that the colors in the new active window are correct.

If you have an IDL session with 256 colors and a private color map, then you will have to move your cursor into a graphics window (or have a graphics window as an active window) to see the colors correctly. But when you put your cursor into a graphics window quite often what will happen is that all your display windows will turn black. In fact, you may have a hard time finding them again! This is because the lower colors in most IDL color tables are dark colors. These happened to be the same colors that in the *shared* color map were used by the window manager for window colors. This "flashing" you see has to do with the way window managers work, *not* with the way IDL works.

Sometimes you want IDL to use the shared color table, but you also want to leave some colors in the shared color table for other applications to use. (Maybe you want to start up Netscape *after* you start up IDL.) You can do this with the WINDOW command by using a negative number with the Colors keyword. For example, this command will cause IDL to take all of the colors in the shared color table *except* for 20:

```
WINDOW, Colors=-20
```

Twenty colors will now be available in the color table for other applications to use.

> When plotting to postscript this works out fine, but when

> plotting to screen, the colorbar is changing.

The different number of colors on your display and in PostScript output does have implications for how you write programs that behave correctly in both environments. Most of what I know about this topic I've written in a series of articles entitled 'How to Produce Perfect PostScript Output" on my web page.

I'm sorry for the long-winded explanation. Many people ask this question and I have been meaning to write this up as an IDL programming tip for a long time. This just seemed like the time to do it. :-)

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
