Subject: Re: Colormaps (a favorite subject!)
Posted by Liam E. Gumley on Tue, 28 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Craig Hamilton wrote:
> David:
> Thanks for the code, but my problem is not with grey-levels vs.
> number of colormap entries. My problem is as originally stated
> it: I want the user to be able to select the number of colors
> in the colormap, but I can't popup a window to get that information
> without having the number of colormap entries already set
> in the act of popping up the window. I must not be explaining
> myself very well. I've done loads of image processing programming
> in other languages on Unix and Windoze, so I'm familiar
> with colormaps and scaling of image data for display.
> I'm working with 16-bit data on 8-bit displays right now and I want to
> avoid color-flashing, so I want to use from 32 to 128 colormap
> entries for this display program. I want the user to be
> able to select how many entries are used.
>
> Maybe a little more simply:
> I have been initializing the size of the colormap used with:
> window,0,colors=numcolors,/pixmap,xsize=10,ysize=10
> wdelete,0
> But now I just want to get the number 'numcolors' from the user first.
> I think I am realizing that I cannot get that number with a widget.
> Right?

Craig,

Here's an alternative approach.

First, let IDL decide how many colors are available. You can do this via a startup file. I'll assume that you wish to support 8-bit graphics only. Here is the startup file I would use.

```
if !version.os_family eq 'unix' then device, pseudo=8
window, /free, /pixmap, colors=-10
wdelete, !d.window
device, decomposed=0, retain=2, set_character_size=[10, 12]
device, get_visual_depth=depth
print, 'Display depth: ', depth
print, 'Color table size: ', !d.table_size
```

This causes a graphics window to be opened when IDL starts up, thus allowing IDL to determine the size of the color table. This approach is more flexible than selecting a pre-set number of colors.

Then in your application,let the user choose which part of the color table will be used. This is typically done via the BOTTOM and NCOLORS keywords. BOTTOM refers to the bottom entry in the color table, and NCOLORS refers to the number of entries in the color table that should be used. The default values would be

```
bottom = 0
ncolors = !d.table_size - bottom
```

To scale and display an image:

```
image = dist(256)
loadct, 13, bottom=bottom, ncolors=ncolors
tv, bytscl(image, top=(ncolors - 1)) + byte(bottom)
```

Using this method, you can display multiple images with different color tables. For example:

```
image = rebin(dist(32), 256, 256, /sample)
window, /free
bottom = 0
ncolors = 64
loadct, 13, bottom=bottom, ncolors=ncolors
tv, bytscl(image, top=(ncolors - 1)) + byte(bottom)

image = dist(256)
window, /free
bottom = 64
ncolors = 64
loadct, 3, bottom=bottom, ncolors=ncolors
tv, bytscl(image, top=(ncolors - 1)) + byte(bottom)
```

This technique is called 'color table splitting', and it can be very useful when IDL is running in 8-bit graphics mode. I would also encourage you to start thinking about designing you application to work in 24-bit graphics mode, where color flashing problems do not exist, and you always have 256 color table entries available.

Cheers,
Liam.
http://cimss.ssec.wisc.edu/~gumley