Subject: Re: How to load a customized color table?
Posted by Spon on Mon, 12 Jan 2009 18:43:41 GMT
View Forum Message <> Reply to Message

On Jan 12, 9:26 am, RussellGrew <russell.g...@gmail.com> wrote:
> g[*] = 0.
>
> may be more appropriate.

Also completely unnecessary, as IDL will set all elements of a newly
created array to 0 by default unless you use the /NoZero keyword. I'm
also not sure what that common block is doing there, or the double
precision numbers that get turned into bytes anyway.

This is how I'd write such a procedure:

***

```
 ; OldRGB is an output variable that
 ; returns the previous colour table values
PRO LOADREDBLUE, OldRGB
 ; Store current colour table
TVLCT, OldRGB, /GET
 ; Generate new colour table values
R = [2B*REVERSE(BINDGEN(128)), BYTARR(128)]
G = BYTARR(256)
B = [BYTARR(128), 2B*BINDGEN(128)]
 ; And load them
TVLCT, R, G, B
PRINT, 'Loading table: Red-Black-Blue'
RETURN
END
```

***

So if you cared about turning your old colour table back on after your
call, you could use the procedure like in this example:

```
IDL> Device, Decomposed = 0
IDL> Image = Dist(256)
IDL> LOADREDBLUE, Previous ; load your custom table. The 'Previous'
variable contains the old table's values.
IDL> TVSCL, Image ; display the image
IDL> TVLCT, Previous ; reload the old table
IDL> Window, /Free & Plot, Image ; using previous colour table in a
new window
```

And if you don't care what the previous colour table looked like, just
type:

```
IDL> LOADREDBLUE
```
and you're away (assuming that, again, you've turned off decomposed colours)
```
IDL> TV, Bindgen(256,256)
```

Regards,
Chris