Subject: Colors with XOR plotting: How to do it
Posted by eaustin on Fri, 17 Jul 1992 20:47:37 GMT
View Forum Message <> Reply to Message

In article <1992Jul15.210840.10057@ll.mit.edu>, I asked
|>
|> As a further enhancement to the rubber band effect I would like to
|> be able to get a particular color with XOR set. For example, I would like to
|> be able to oplot a red X on a red line and have the X move around on the line
|> in response to the cursor position. Is this posible or does the working of
|> XOR prevent red on red like this? I would also like to put a blue mark
|> on a red line and move the blue mark around.
|>
|> I guess the bottom line is, how can I choose what color comes out with
|> XOR plotting?

I think I have an answer now so I am posting it to the net. This will require
some introductory dicussion of idl color maps so all you experts can skip
the first part. (Of course, why would an expert be reading this anyway?)

WARNING: This is all to the best of my knowledge. I am not an expert at all.

My machine, SPARC 2 running Openwindows 2, is capable of displaying
256 different colors simultaneously. There are over 16,000,000 possible colors
but only 256 can be shown at once. This is because the machine uses a
color map that has 8 bits (2^8=256) of indexing. Each index in the color map
represents one color so there can be 256 (0 to 255) colors. The color that an
index represents can be changed so you can get any color you want, but only 256
of them at a time. Each index can point to only one color but several
indicies could all point to the same color. Thus, all 256 colors could be
set to black, or red or anything else. Every pixel on the display is
assigned one of the 256 color map indicies. The display is changed by changing
the color index of the desired pixel. Thus, if index 1 points to the color
black and index 2 points to the color red then if the color index of a pixel
is changed from 1 to 2 the display will change from black to red at that point.
If index 2 pointed to green then changing the index of the pixel from 1 to 2
would change the display from black to green at that point. The index has
made the same change in both cases, from 1 to 2, but the color map is different
in the two cases so the displayed result is green instead of red.

When the machine first opens a window there are already several colors set,
the color of the background, the color of the cursor, the color of the borders,
etc. These are all stored in the color map. The number of color indicies used
depends on how many different colors you call for. If you run some application
that needs a new color, another one of the indicies is used and assigned to
that color. When idl is started it tries to share this existing color map
and adds any new colors after all the old ones. Thus, if the existing color
map only used 35 colors then the first color idl uses would be index 36.

HOWEVER, to make it easier on the user, idl calls the first idl color as index 0 and it does the translation from idl index 0 to shared color map index 36 internally without the user having to worry about it. Since there are only 256 possible colors and the shared color map has already used 35 of them before idl started, idl can only use 221 different colors before it runs out of space in the color map. If, for some reason you need more colors than the shared color map has left then idl will create a private color map with 256 colors for its very own use. There are complications to this that I do not want to go into but for a simple understanding this is fine.

When DEVICE,SET_GRAPHICS = 6 is set then idl does a bit level exclusive OR on the color map index. The trick is that the index used is the shared map index running from 0 to 255, not the idl index running from 0 to whatever. The XOR of 1 and 0 is 1 but the XOR of idl color index 1 and idl color index 0 could be anything. It depends on what the "real" indices are in the shared color map. When you do an overplot when XOR is set the color index of the new data is XORed with the color index of the existing data to get the new color index. The color of this new index is what will be plotted. It has nothing to do with what the actual colors are, only with what the color indicies are. And the color indicies used are those from the overall shared color map of 256 indicies, not the idl indicies starting at 0 (unless idl has a private color map where the idl index and the overall index are the same).

So, with this long intro over, I will show how I got the colors I wanted when using XOR. My code assigned colors as follows:

```
device,translation=t
!p.background = 128-t(0)
black = thecolor('black',index=!p.background)
white = thecolor('white',index = 1)
red = thecolor('magenta',index = 2)
green = thecolor('green',index = 3)
blue = thecolor('DeepSkyBlue',index = 4)
yellow = thecolor('yellow',index = 5)
awhite = white xor 128
ared = red xor 128
agreen = green xor 128
ablue = blue xor 128
ayellow = yellow xor 128
```

The device,translation=t
command put an array into t that showed the mapping from idl color indicies to shared color map indicies. That is, t(0) is the shared color map index of idl color 0, t(1) is the shared color map index of the idl color 1, etc. Try it yourself. I needed to set my plotting background color to shared map index 128 (10000000 in binary, notice only the top bit

is set). I did this with  !p.background = 128-t(0). This set the idl index
to the proper number so that its shared map index would be 128. I then
decided to make the background color black using the command
black = thecolor('black',index=!p.background).
"thecolor" is a locally written routine that allows you to pick colors by
name and assign them to a specific idl color index. Thw output of thecolor
is the idl index of that color. Thus, the result of
black = thecolor('black',index=!p.background) is to set black=!p.background
and to make this index point to the color black (determined by the string
'black'). I then set several other colors that I want ('white','magenta',
'green','DeepSkyBlue', and 'yellow') and call their idl color indicies
white, red, green, blue, and yellow, respectively. I do not care
what their shared color map indicies are as long as they are less than 128 so
that their high order bit is not set. Finally, I set the variables
awhite, ared, agreen, ablue, and ayellow to the XOR of their index with 128,
the background color index. Now, for example, whenever I plot an "X" using XOR
with the color index ared the result will be

ared XOR 128 (the background) = (red XOR 128) XOR 128 = red

The two "XOR 128"'s cancel each other out (that is how XOR works) leaving me
with the color index red which points to the color magenta and I see a
magenta  "X" on the display. If I now plot the same point again with ared I get

ared XOR red (left from the first time plotting) = (red XOR 128) XOR red = 128

which is my background color, black in this case, and the magenta "X" goes
away. The background could have been any color I liked and it still works.
the important part is that the background is shared color map index 128.

I am using this procedure to mark a point on a previously drawn line. I wanted
the marked point to move in response to the user moving the cursor and I
found that constantly replotting the whole thing to get rid of the old "X"
was too slow. This technique works relatively quickly.

Not a problem for me, but it could be a problem for other applications is that
where the magenta "X" intersects the magenta line the color changes to
some other undetermined color (I could determine it but it is not worth the
effort). This only occurs while the "X" is there, when I replot the "X"
to erase it the line goes back to its original color so no permanent change
has been made.

I hope this is helpful to people. Let me repeat that I am not very experienced
with idl and especially not with the real workings of windowing systems
and so on (thinking in bits is not my line of work) so I make no guarantees
as to my explanations above. This approach worked for me and I hope it can
be helpful to someone else.
--

Eric J. Austin EMAIL: eaustin@ll.mit.edu