Subject: Re: FSC_PLOT defaults
Posted by ben.bighair on Tue, 21 Dec 2010 01:43:01 GMT
View Forum Message <> Reply to Message

On 12/20/10 5:57 PM, David Fanning wrote:
> Wayne Landsman writes:
>
>> I am updating the plotting routines in the IDL Astronomy library (e.g.
>> AL_LEGEND, TVCIRCLE, PLOTERROR in http://idlastro.gsfc.nasa.gov/) to
>> use the new Coyote routines FSC_PLOT, and FSC_COLOR(), and for example
>> to allow colors to be specified by name. It is mostly going well
>> but I do find it jarring going from the white on black of
>>
>> IDL> plot,indgen(10)
>>
>> to the black on white of
>>
>> IDL> fsc_plot,indgen(10)
>>
>> I'm old-fashioned enough to think that black on white does not show
>> up as clearly on the screen, but I did not have a similar problem
>> with the black on white of function graphics.
>>
>> The reason is that
>>
>> IDL> o = plot(indgen(10))
>>
>> is more comparable to something like
>>
>> IDL> fsc_plot,indgen(10),thick=2,xthick=2,ythick=2,charsize = 2, xticklen = !P.ticklen*2,
yticklen=!p.ticklen*2
>>
>> to give the lines extra thickness (and clarity).
>>
>> Maybe it would be worth having a similar default display for
>> FSC_PLOT (at the expense of having different defaults than PLOT).
>> Another default I like in the function PLOT() is having exact X,Y
>> ranges (X|Y|Style= 1). --Wayne
>
> The central theme of my upcoming book is that there is
> no real need for fancy new graphics. That almost everything
> you need is already available to you via traditional graphics
> commands IF you do a few simple things. One of those things
> is write programs that work both on your display and in a
> PostScript file. Of course, it is *possible* to do this
> if one has a black background and one has a white background,
> but if you use color at all (another theme of the book), then
> this is all much harder to do.

>
> Better, I thought, to make everything work the same.
> Even old timers with schizophrenic brains catch onto
> it fairly quickly and realize how easy it is to make things
> work identically in the two different environments. :-)
>
> That said, I've gone to some trouble to make sure these
> routines work "naturally". If you set the background color
> to "black", the drawing routines should draw in white.
> I've just really changed the default background color.
>
> I've no objection to changing the default line thickness
> values, although I have tried to make these commands work
> as normally as possible, even when "normally" sometimes
> means sub-optimally. I really only tried to fix the things
> I thought were obvious defects. (I do set exact axis ranges
> on the FSC_Contour command, but personally, I don't mind
> autoscaling axes on the Plot command most of the time.)
>
> I do note that I *always* change line thicknesses when
> I make a PostScript file, so maybe changing it everywhere
> is not such a bad idea. Let me play with this a little and
> see what I can do. Anything that makes these more useful
> is what I am looking for.
>
> We still have a month or so to experiment before
> they are cast in stone. :-)

Hi,

I wonder if you might consider having a per-user default preference stored in a system variable - it's really just swiping the idea from IDL's !P, !D system variables but with improved functionality. The preferences could be stored in a FSC_Defaults.pref save file that is loaded at least once per session (somehow). I think I would make the initial setup similar to IDL's PREF_SET, PREF_GET, etc. That would allow the user to decide what they really want as defaults if different than what you provide. There could even be a default per device.

So, instead of this...

```
; Check the keywords.
    IF N_Elements(sbackground) EQ 0 THEN $
 background = 'WHITE' ELSE $
 background = sbackground
```

It might be this...

```
; Check the keywords.
    IF N_Elements(sbackground) EQ 0 THEN $
 background = FSC_Default("background") ELSE $
 background = sbackground
```

And the function FSC_Default could be simple a wrapper around
!FSC_Default object that maintains the settings.

```
Function FSC_Default, what, value, SET = set, _REF_EXTRA = extra

  IF (KEYWORD_SET(set)) THEN $
    return, !FSC_Default->Set(what, value, _EXTRA = extra) ELSE $
    return, !FSC_Default->Get(what, _EXTRA = extra)

END;
```

The object that !FSC_DEFAULT references would have two methods, Get and
Set, that redirect to "Get_PS", "Get_Win", "Set_PS", "Set_X", etc.
depending upon the current device state. Like this...

```
Function FSC_DEFAULT::Set, what, value, SAVE = save

 theMethod = "Set_" + !D.Name
 oldValue  = self->CallMethod(theMethod, what, value)
 if (save) then self->Save
 return, oldValue

END

Function FSC_DEFAULT::Get, what, COUNT = count

 theMethod = "Get_" + !D.Name
 return(self->CallMethod(theMethod, what, COUNT = count))

END
```

This might be a whacky idea and not worth the effort. But it is easy to
suggest since I wouldn't be doing the work!

On one hand it would put the default settings in the user's hands on the
other it would take a good deal of mind-numbing effort to switch all of
the keyword tests. The consolation is it only has to be done once -
forever (maybe).

Cheers,
Ben