Subject: idl 5.3, problem with plots and mapping environments
Posted by Vapuser on Fri, 28 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

<rant>
!version={ mipseb IRIX unix 5.3 Nov 11 1999}

I've just run into a strange problem with PLOTS and mapping environments that seems to be new with IDL 5.3. I say this because I'm not doing anything radically different from before and now I'm getting this strange ... well... effect. You'll see in a moment why I hesitate to be more specific in naming it.

Do the following

```
IDL> map_set,0,180,lim=[-10,100,60,290] ,/grid,/lab,/cont
IDL> plots,[100,200],[80,91]
```

This will produce the message

```
% PLOTS: Value of Latitude is out of allowed range.
```

Fine. I don't really have a problem with this. Yet.

But, if you put this into a routine, with 'catch,error' code to try to catch the 'error,' you'll never will. At least, you won't on my SGI. I did

```
;------------------------------
pro junk2

  catch, error
  IF error NE 0 THEN BEGIN
    Message,!error_state.msg,/cont
    stop
  ENDIF
  map_set,0,180,lim=[-10,100,60,290] ,/grid,/lab,/cont
  plots,[100,200],[80,91]

END

;------------------------------
```

The error handling code is never called, running this puts you right back at the prompt at %MAIN.

However

IDL> help,!error_state,/st

```
** Structure !ERROR_STATE, 7 tags, length=52:
   NAME            STRING    'IDL_M_BADARGVAL'
   BLOCK           STRING    'IDL_MBLK_CORE'
   CODE            LONG              -82
   SYS_CODE        LONG      Array[2]
   MSG             STRING    'PLOTS: Value of Latitude is out of allowed ra'...
   SYS_MSG         STRING    ''
   MSG_PREFIX      STRING    '% '
```

Apparently all errors are equal, but some errors are *more* equal.

I also tried 'on_error' and 'check_math().' No dice.

So we have a routine sending a message which is impossible to turn off because it never generates an error that can be caught with any error catching code! I find this *very* frustrating. You try to write code that only generates exceptions in, well, exceptional circumstances and here comes IDL creating a message which suggests that somethings wrong, but it won't let you find which vector of the hundreds or thousands is the problem!

Now, why am I worried about plotting over the poles, you may ask.

I make animations where I move vectors around in response to wind fields, static or dynamic. Occasionally I make them close to the poles and the vector which I want to draw may go over the pole. Here's the problem. I wrote the vector plotter to be independent of the environment, it doesn't know from mapping environments and, if I could possibly help it, I'd rather that it continued this way. When I call the plotter I don't know whether the vector will go over the pole. Up until now, IDL has been either smart enough to deal with the problem or blessedly unaware/silent about the existance of one. But now I have to find a way to determine that a mapping environemt is in place inside the plotter -- and a cursory, but fairly thorough examination of the !map environment variable revealed no obvious way to make this determination, or send in a keyword telling the plotter that it's in a mapping environment and please check for boundary cases like plotting over a pole. Which means I have to chase through all my code changing the calls.

More annoying is the fact that IDL can wrap longitudes without any problem. For instance, you can do this:

IDL> map_set,90,0,/stereo,/grid,/lab,/cont & plots,[90,-90],[30,30]

and this:

IDL> map_set,90,0,/stereo,/grid,/lab,/cont & plots,[90,270],[30,30]

of event this

IDL> map_set,90,0,/stereo,/grid,/lab,/cont & plots,[90,-90]+360,[30,30]

It's all the same to IDL.

All that would be necessary to fix this problem is if the core of IDL was smart enough to do what I'll have to do anyway, take any reference to a latitude greater than 90, subtract 90 and change the longitude to the antipodal longitude. (mutatis mutandis for negative latitude).

Finally, the most annoying thing is that the vectors in question weren't even *in* the scene, so IDL was complaining about vectors which would've been impossible to plot, even if it was smart enough to wrap the latitudes like it does the longitudes.

But at the very least if they're going to send a message, there should be a way to turn it off by catching an error! What good is the message if you can't catch the error!

And this is not a small thing for me. Each one of these vectors generates 1 to 8 messages, because I draw an arrow and any one of the endpoints can generate one of these errors. I draw thousands of arrows for each frame of an animation and I make animations that have between 60 and 800 frames. In one 60 frame animation, IDL produced 624 of these unstoppable error messages!

It's this kind of *passive aggressive* help that I could very well do without.

I understand that mapping is complicated, so I very well may be missing something that makes the solution to this sort of problem difficult if not impossible. If that's the case, though, I really wish someone would tell me what would make this fix hard. I've already implemented this code for the 'locations' of the vectors (defined as the midpoint), now I'm going to have to implement it for all endpoints as well. If I can do it, RSI can too.

</rant>

William
--
William Daffer: 818-354-0161: vapuser@catspaw.jpl.nasa.gov