Subject: Re: How to display single orbits of satellite data in function graphics?
Posted by chris_torrence@NOSPAM on Wed, 01 May 2013 21:58:42 GMT
View Forum Message <> Reply to Message

Hi all,

Well, there is both good news and bad news. The new graphics are indeed faster in IDL 8.2.x, but for your particular problem, it doesn't make much difference. The real bottleneck is just object graphics and OpenGL. Here is a reproduce case which compares both pure object graphics and new graphics:

```
n = 793647
lon = RANDOMU(seed,n)*360
lat = RANDOMU(seed,n)*180
colour = BYTSCL(RANDOMU(seed,n))
oSym = IDLgrSymbol(24, /FILLED)
oPal = IDLgrPalette()
oPal->LoadCT, 39
tic
oPlot = IDLgrPlot(lon, lat, LINESTYLE=6, $
  PALETTE=oPal, SYMBOL=oSym, VERT_COLORS=colour)
oModel = IDLgrModel()
oModel->Add, oPlot
oView = IDLgrView(VIEWPLANE_RECT=[0,0,360,180])
oView->Add, oModel
oWin = IDLgrWindow(GRAPHICS_TREE=oView)
oWin->Draw
toc

tic
p = PLOT(lon,lat,$
  SYMBOL='circle', $
  /SYM_FILLED, $
  SYM_SIZE=0.2, $
  RGB_TABLE=39, $
  VERT_COLORS=colour, $
  LINESTYLE=6)
toc
```

On my Win7 laptop, running 64-bit IDL, with hardware rendering, this takes 10.5 seconds for object graphics, and 24.0 seconds for new graphics.

So the new graphics is only off by a factor of 2 from pure object graphics. Now we can certainly try to chip away at that difference in future releases, but we're only going to be able to get it down to 10 seconds without lifting the hood on object graphics.

I think the main problem is that there are 790,000 points, each of which is a filled circle which has 25 vertices.

Fundamentally, it comes down to the difference between direct graphics, where you are just "burning" pixels into the screen, versus object graphics, where you are maintaining an object model in both memory and in the graphics card. One is fast, the other can be modified later.

Thoughts?

-Chris
ExelisVIS
