Subject: Re: 3-D data, positional vectors, and scaling
Posted by Mark Hadfield on Wed, 23 Aug 2000 00:48:46 GMT
View Forum Message <> Reply to Message

"tbowers" <tbowers@nrlssc.navy.mil> wrote in message
news:8nv3fb$9o$1@ra.nrl.navy.mil...
> Hi,
> Could somebody help me out before I go postal on my innocent
> co-workers here?
>
> I'm trying to visualize 3-D volume data with proper lat lon
> depth xyz arrays so's I can use those as my axes AND do some
> mapping overlay stuff (I guess with map_set, etc.) Anyway,
> to test, I'm just butchering David F's XSurface (I'll get to
> Mark H's mgh_example_graph3d.pro soon enough, too) and creating
> a volume version. The problem is that IDLgrVolume doesn't
> take x,y,[z] arguments like IDLgrSurface does, so it doesn't
> store its positional info so I can use it to position stuff
> via [XYZ]Coord_Conv like David does to scale to data coords
> in his XSurface.pro, like:
> ...
> ...Looks like no matter what, the vol.
> data will plot in its "index" space (voxel plotted at array
> indices), not where I'm tryin' toi tell it to go.

IDLgrVolume is one of the few object graphics atoms (IDLgrLegend is another) that doesn't let you control its geometry, other than by changing [X,Y,Z]COORD_CONV. I don't like this, because it doesn't fit naturally into the method that both David & I use to scale data against axes.

I've only used an IDLgrVolume once, and on that occasion I adapted the coordinate system to the volume's index space rather than vice versa. But I think the thing to do is to hide the IDLgrVolume inside a wrapper object. The wrapper would be a subclass of IDLgrModel and would contain the IDLgrVolume, with maybe an intermediate models. You can give the wrapper object properties LOCATION and DIMENSIONS (like IDLgrImage, but 3D). The model tree would handle the transformation between the IDLgrVolume's index space and the desired data space in which LOCATION and DIMENSIONS are expressed.

I was going to cite my MGHgrLegend class as an example of something like this. It is very like an IDLgrLegend but has a LOCATION property. However looking at the code again I see that MGHgrLegend is a subclass of IDLgrLegend and it does its work by fiddling the COORD_CONV properties (no models involved). So perhaps my MGHgrCompositeExample class is a better example. From the documentation header:

; PURPOSE:

;   This class was written as a prototype for a typical composite graphics
;   object. It implements a grey rectangle with a black outline. It has
;   properties LOCATION and DIMENSIONS. It has no practical use.

I *think* the MGHgrCompositeExample could be converted to a volume-scaling-wrapper quite easily. One issue not anticipated in MGHgrCompositeExample is that the scaling between the "index space" and "data space" will change if the size of the volume array changes.

See

 http://katipo.niwa.cri.nz/~hadfield/gust/software/idl/mghgrl egend__define.pr
o
 http://katipo.niwa.cri.nz/~hadfield/gust/software/idl/mghgrc ompositeexample_
_define.pro
 http://katipo.niwa.cri.nz/~hadfield/gust/software/idl/mgh_ex ample_composite.
pro

As a general comment, whenever a graphics object doesn't behave the way I want, I wrap it inside a class that *makes* it behave the way I want. (Then I forget how I did it.) There are pitfalls, of course. You can inadvertently disable some of the functionality of the original or you can find yourself storing a lot of redundant intermediate data. But these days IDL makes it relatively easy to construct thin wrappers.

PS: You mentioned map projections. I don't think anybody's cracked that particular nut using object graphics yet.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield/
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand