Subject: Re: Adaptation of surf_track for viewing earthquakes in 3 dimensions
Posted by Rick Towler on Thu, 11 Jul 2002 15:47:06 GMT
View Forum Message <> Reply to Message

"Thomas Wright" <twright@usgs.gov> wrote

>
> 1.IDL seems peculiarly unsuited for use with 3-dimensional point data.
> I have no need of connecting earthquakes, contouring surfaces, or
> otherwise looking at them other than as points in x-y-z space. I had
> expected to be able to freely configure symbol size, symbol shape and
> color to represent different earthquake parameters such as magnitude,
> depth, time intervals or other user-defined classification parameters.
> But none of the IDLgr routines allows this.

Sheesh. You can't expect IDL to do everything out of the box...

OG atoms although useful in their own right are really meant to act as building blocks for user defined objects. You could easily create an object which is a subclass of the IDLgrModel object which contains an instance of IDLgrPolyline and IDLgrSymbol. Call it quakeSymbol__define.pro. It could take magnitude and location as parameters and pass color and symbol data to your symbol object (or you could define all sorts of symbols in your object to represent depth, time or whatnot and use your own keyword to select the symbol). Magnitude could map to the symbol's size property and location would position your single point IDLgrPolyline object in space. But then again, maybe someone has already done this.

> It would be very useful to have the rotation parameters written on the
> screen (or included in one of the "other options" buttons) while
> rotating the image using the mouse.

Surf_track uses the trackball to generate the transform matricies. The trackball (as far as I know, it has been a while) doesn't store it's orientation as pitch, yaw, and roll so there is no way to display the orientation in a intuitive form. If you are really into surf_track as your base program then your only hope is to search for the "matrix and quaternion faq" and look for some code to convert your transform matrix into PYR values. I seem to remember running into a few issues playing around with this but maybe you will have better luck.

You may want to look into my camera__define object. I do have an example program which is similar to surf_track in that the camera rotates about the origin plus it allows you to zoom in and out and it displays the orientation (it is a virtual globe). Let me know if you are interested and I will send you the files.

Also, if you are working with IDL on win32 I suggest ditching MPEG and looking into IDL2avi from Ronn Kling's website:
http://www.kilvarock.com/freesoftware/dlms/avi.htm
In conjunction with the Intel Indeo 5 codec (www.lygos.com) it makes for a powerful and simple animation export tool.

-Rick

Subject: Re: Adaptation of surf_track for viewing earthquakes in 3 dimensions
Posted by Mark Hadfield on Fri, 12 Jul 2002 04:34:21 GMT
View Forum Message <> Reply to Message

"Thomas Wright" <twright@usgs.gov> wrote in message
news:aee1db91.0207110002.b4d9e7a@posting.google.com...

> ...Animations are run using a driver program that calls surf_track
> in a loop that converts the 3-d object view to a 2-d view, then
> loads an mpeg file run by xinteranimate. Several questions have come
> up in the course of this development.

That's not the most elegant way to animate a 3D system, because all 3D information is lost when the animation is prepared.

I have played around with animations in IDL. There are all sorts of ways you can do it. I have investigated 3 of them:

- Save each frame as an image and animate the image sequence (as you describe)

- Store a sequence of atoms (or models or whatever) representing the changing part of the graphics tree in a container, Animate by adding, displaying and removing each object in this sequence.

- Store a sequence of command objects, each describing which atom (or model ...) in the view is to be modified, the method to be called on that object, and the keyword data required by that method. Animate by applying each command in turn.

All are useful in different situations, but the third is the one I use the most. I created a class called MGH_Command specifically for storing the commands. You might want to look at my library at

ftp://ftp.niwa.cri.nz/incoming/m.hadfield/MGH_MOTLEY.tar.gz

See mgh_example_animate.

> I had expected to be able to freely configure symbol size, symbol
> shape and color to represent different earthquake parameters such as
> magnitude, depth, time intervals or other user-defined
> classification parameters. But none of the IDLgr routines allows
> this. IDLgrpolyline allows symbol manipulation, but only one
> color. IDLgrpolygon accepts a color matrix using vertex_colors, but
> has no call to symbol. Idlgrplot has both symbol and color control,
> but the z parameter, if used, is set to a constant.

The normal way to display a cloud of symbols is to attach them to an IDLgrPolyline with LINESTYLE = 6 (invisible). You can have more than one symbol attached to the IDLgrPolyline, in which case the symbols are used in turn and cyclically repeated if necessary. And you can give each symbol a different COLOR property. So in principle you can display as many different symbols as you want. In practice you may find redrawing gets slow if you have too many different symbol objects, so it may require some ingenuity to limit the number. As I recall, if a symbol's colour is a single byte value then it represents an index into the color lookup table of the parent's PALETTE object. You can use this fact to store up to 256 different symbol colours relatively economically. This was discussed in a thread in March 2000 entitled "Object graphic 3d Scatterplot".

I won't comment on the feasibility of modifying SURF_TRACK except to say that I've usually found it simpler in the long run to write my own code rather than adapt code from the IDL library.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)