Subject: Re: interactive polar blobby plots
Posted by Rick Towler on Fri, 12 Jul 2002 00:41:20 GMT
View Forum Message <> Reply to Message

"tb" <tbowers0@yahoo.com> wrote
>
> My other option seems to be starting from scratch and doing something
like:
>
> ; r is 2D array of distances from [0,0] over all theta (cols) & phi (rows)
> mesh_obj, 4, vertexList, polygonList, r ; make an oddly shaped spherical
> blob
> oPolarPlot = obj_new('IDLgrPolygon', DATA=vertexList, POLY=polygonList,
> COLOR=[255,0,0], STYLE=2)
>
> then do the whole mess of trying to get axes in the center, etc., etc.,
etc.
> Also need to be able to see the axis values, so would need to add surface
> opacity stuff, etc., etc., etc.

Personally this is the way I would go. I plot point source acoustic backscatter data which is pretty much identical to your data and the "blobs" plot quite nicely. You can texmap the polygon to color the vertices as you wish (intensity in my case) and you include an alpha channel so you can modulate opacity.

There is one stumbling block. Our friend Z. The full discussion can be had by searching this group for 'Pimento Problem' but the jist is that IDL can't properly render transparent objects that can be viewed from all angles. If I understand, you will be vizualizing a single blob at a time centered about the origin? In this case the solution is fairly simple.

use mesh_obj to create your vertex and polygon arrays

use mesh_clip to create vertex and polygon arrays for the front (+Z) and back (-Z) portions of your blob.

create two polygon objects (front and back) using the vertex and polygon data you just created.

add your polygon objects to your blob model

When changing your perspective keep track of the sign of your view vector's Z value. -Z values will have you "looking from outside your screen in" which places the front half of your blob in front of the back half. In this case you want to order your blob polys in your blob model [back,front].

But, when your view vector's Z value is positive, you are "looking from inside your screen out", which places the the back blob half in front of the front blob half. In this case you need to swap the atoms in the model so they are ordered [front,back]. Use the "IsContained" method of the model along with the position keyword to get the current position of your atoms then use the "Move" method to swap the atoms as necessary.

Take a look at my camdemo_examine program for a good starting point for the viewer code (http://www.acoustics.washington.edu/~towler/). Replace the globe code with your blob code. The benefit of using my camera is that you can easily get your view vector which simplifies atom management.

In the MOTION event handler you can do something like:

```
info.camera -> getproperty, quaternion=quaternion

;check to see if viewing from front or back and flip if needed
vect = quaternion -> GetDirectionVector()
if (vect[2] ge 0) then z=1B else z=0B
if (z xor info.z) then begin
  ok = blobmodel -> IsContained(blob_back, position=back)
  ok = blobmodel -> IsContained(blob_front, position=front)

    if (front lt back) then $
      blobmodel -> Move, back, front $
    else $
      blobmodel -> Move, front, back
end
info.z = z
```

That's probably not my best chunk of code but you should get the idea...

Enjoy!

-Rick