
Subject: Re: ellipsoid 3D

Posted by [adisn123](#) on Fri, 04 Aug 2006 16:14:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thanks Rick.

You're more helpful than I hoped for.

Either clipping or mesh works fine with me, but I think I like mesh more since it has more capabilities.

Rick Towler wrote:

> adisn123 wrote:

>> The main purpose of this was to take a portion of ellipsoid and examine
>> it in 3D.

>>

>> I remember there is IDL routine or function that takes a part of the
>> whole volume and display it.

>>

>> I'm not sure of what it is yet, but working on it.

>

> As always, there are a couple of ways to do this. You can either use
> the MESH_CLIP function or set the CLIP_PLANES property of your orb.
> Remember that the orb is a subclass of IDLgrModel so you'll need to look
> at the documentation for IDLgrModel to find info on the CLIP_PLANES
> property.

>

> From my experience, using CLIP_PLANES is easier as you don't need to
> deal with keeping track of the clipped vertices and polygon connectivity
> data. There are a couple of limitations though. Since the clipping is
> done internally and you don't have access to the clipped vertices you
> only can make simple slices of your object. For instance you couldn't
> "slice out" or remove only a quarter of your ellipsoid by specifying two
> clipping planes orthogonal to each other which pass thru the ellipsoid's
> origin. This would in fact leave you with only 1/4 of the ellipsoid.
> The other limitation is that some graphics drivers don't use optimized
> rendering paths when clipping planes are enabled so drawing objects that
> use clipping planes can be slow.

>

> On the other hand, since MESH_CLIP returns the clipped vertices, you can
> use them to piece together more complicated objects. Using MESH_CLIP
> you could remove a quarter of your ellipsoid by slicing twice with a
> plane that passes thru the origin to return the two halves. Then slice
> one of the halves into a quarter and stick the quarter and half back
> together.

>

> Defining the clipping planes can be a bit tricky. I find it easy to
> define the plane as a set of 4 vertices and then calculate the
> coefficients of the clipping plane. Given "clip_planes" which is a
> 3x4xN array of vertices that describe N clipping planes, the following

```

> code returns a 4XN array "planes" which can be passed directly to
> IDLgrModel or (individually) to MESH_CLIP.
>
> sP = INTARR(3)
> sP[0] = SIZE(clip_planes, /DIMENSIONS)
> if (N_ELEMENTS(sP eq 2)) then sP[2] = 1
>
> planes = DBLARR(4,sP[2], /NOZERO)
>
> for r=0, sP[2] - 1 do begin
>   u = clip_planes[* ,2,r] - clip_planes[* ,0,r]
>   v = clip_planes[* ,3,r] - clip_planes[* ,0,r]
>   n = CROSSP(v,u)
>   n = n / SQRT(TOTAL(n^2))
>   planes[* ,r] = [n,TOTAL(n * (-clip_planes[* ,1,r]))]
> endfor
>
> The input verts must be wound counterclockwise when looking "from the
> outside in". What does this mean? It means that if your clipping plane
> clips the wrong half, reverse the order of the plane verts you feed this
> function.
>
>
> To put it all together:
>
>
> ; create the orb - use the DENSITY keyword to bump up the
> ; vertex count.
> orb = OBJ_NEW('orb', COLOR=[240,0,0], STYLE=1, DENSITY=3.0)
>
> ; scale asymmetrically to create the ellipsoid
> orb -> Scale,1,1,2
>
> ; define the plane to clip the ellipsoid by defining 4 verts that
> ; lie within it. Note that while not required, I lifted the
> ; plane off of the XZ plane just a bit so the verts on that plane
> ; are preserved
> clip_planes=[[-1.,-0.01,1.],[1,-0.01,1],[1,-0.01,-1],[-1,-0.01,-1]]
>
> ; calculate the plane coefficients
> sP = INTARR(3)
> sP[0] = SIZE(clip_planes, /DIMENSIONS)
> if (N_ELEMENTS(sP eq 2)) then sP[2] = 1
>
> planes = DBLARR(4,sP[2], /NOZERO)
>
> for r=0, sP[2] - 1 do begin
>   u = clip_planes[* ,2,r] - clip_planes[* ,0,r]

```

```
> v = clip_planes[*,3,r] - clip_planes[*,0,r]
> n = CROSSP(v,u)
> n = n / SQRT(TOTAL(n^2))
> planes[*,r] = [n,TOTAL(n * (-clip_planes[*,1,r]))]
> endfor
>
> ; set the CLIP_PLANES property of the orb
> orb -> SetProperty, CLIP_PLANES=planes
>
>
> ; view the result - set the BLOCK keyword so the program
> ; pauses here until we kill the XOBJVIEW window.
> XOBJVIEW, orb, /BLOCK
>
> ; destroy the orb object
> OBJ_DESTROY, orb
>
> end
>
>
> -Rick
```
