Subject: Re: ellipsoid 3D
Posted by Rick Towler on Thu, 03 Aug 2006 19:11:46 GMT
View Forum Message <> Reply to Message

You didn't say if you wanted to do this in Direct Graphics (DG) or Object Graphics (OG). You also don't give any details so I can't suggest one over the other. Given that:

For either OG or DG you need to create a set of points that define the ellipsoid (the vertices) and an array that specifies how the points are connected (the polygon connectivity array). Then you pass these data to the appropriate function to "plot" your 3d ellipsoid.

You could do this the hard way, by creating a function that would calculate the vertices and create the connectivity array give your major and minor axes and a position. Or you could do it the easy way :)

IDL has the 'orb' object which creates a 3d sphere. Assuming you want to do this using OG, it is a simple as creating the sphere and scaling it asymmetrically.

```
;  create the orb object
IDL> orb = obj_new('orb', color=[240,0,0], style=1)

;  since it is a subclass of IDLgrModel we can scale it.
;  stretch the sphere out 2x it's original length along the z axis
IDL> orb -> scale, 1, 1, 2

;  view the result
IDL> xobjview, orb
```

If you need to do this in DG, you can still use the orb object:

```
;  get the vertices, polygon connectivity, and transform matrix
;  from the orb object.  Even though you are looking at an ellipsoid
;  the verts will still define a sphere.  The orb's transform matrix
;  holds the key to scaling the vertices such that they define an
;  ellipsoid.
IDL> orb -> getproperty, data=verts, polygons=polys, transform=xform

;  apply the transform matrix to the spherical verts to make them
;  ellipsoidal
IDL> dgVerts = vert_t3d(verts, matrix=xform)

;  display using DG
IDL> scale3, xrange=[-2,2],yrange=[-2,2],zrange=[-2,2]
IDL> image=polyshade(dgVerts,polys, /t3d)
```

IDL> tv, image

I am aware that this DG code displays a "solid" sphere. I never do 3d in DG so this is the best I care to do. Others might offer tips for displaying 3d objects in DG if you really want to suffer thru this in DG.

HTH!

-Rick

adisn123@yahoo.com wrote:
> Hi,
>
> I'm a begginer in IDL image processing, so if someone lends me some
> help, that'd be great.
>
> I'm trying to make an ellipsoid in 3D.
>
> Not solid, but hollow ellipsoidal in 3D.
>
> Anybody help?
>

Subject: Re: ellipsoid 3D
Posted by adisn123 on Thu, 03 Aug 2006 19:20:54 GMT
View Forum Message <> Reply to Message

Hi Rick,

Thanks for the detail step by step explination.

It gave me a great help.

I think object graphic is good for now.
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.

Thanks..

Rick Towler wrote:
> You didn't say if you wanted to do this in Direct Graphics (DG) or

> Object Graphics (OG). You also don't give any details so I can't
> suggest one over the other. Given that:
>
> For either OG or DG you need to create a set of points that define the
> ellipsoid (the vertices) and an array that specifies how the points are
> connected (the polygon connectivity array). Then you pass these data to
> the appropriate function to "plot" your 3d ellipsoid.
>
> You could do this the hard way, by creating a function that would
> calculate the vertices and create the connectivity array give your major
> and minor axes and a position. Or you could do it the easy way :)
>
> IDL has the 'orb' object which creates a 3d sphere. Assuming you want
> to do this using OG, it is a simple as creating the sphere and scaling
> it asymmetrically.
>
> ; create the orb object
> IDL> orb = obj_new('orb', color=[240,0,0], style=1)
>
> ; since it is a subclass of IDLgrModel we can scale it.
> ; stretch the sphere out 2x it's original length along the z axis
> IDL> orb -> scale, 1, 1, 2
>
> ; view the result
> IDL> xobjview, orb
>
>
> If you need to do this in DG, you can still use the orb object:
>
> ; get the vertices, polygon connectivity, and transform matrix
> ; from the orb object. Even though you are looking at an ellipsoid
> ; the verts will still define a sphere. The orb's transform matrix
> ; holds the key to scaling the vertices such that they define an
> ; ellipsoid.
> IDL> orb -> getproperty, data=verts, polygons=polys, transform=xform
>
> ; apply the transform matrix to the spherical verts to make them
> ; ellipsoidal
> IDL> dgVerts = vert_t3d(verts, matrix=xform)
>
> ; display using DG
> IDL> scale3, xrange=[-2,2],yrange=[-2,2],zrange=[-2,2]
> IDL> image=polyshade(dgVerts,polys, /t3d)
> IDL> tv, image
>
> I am aware that this DG code displays a "solid" sphere. I never do 3d
> in DG so this is the best I care to do. Others might offer tips for
> displaying 3d objects in DG if you really want to suffer thru this in DG.

>
> HTH!
>
> -Rick
>
>
> adisn123@yahoo.com wrote:
>> Hi,
>>
>> I'm a begginer in IDL image processing, so if someone lends me some
>> help, that'd be great.
>>
>> I'm trying to make an ellipsoid in 3D.
>>
>> Not solid, but hollow ellipsoidal in 3D.
>>
>> Anybody help?
>>

Subject: Re: ellipsoid 3D
Posted by adisn123 on Thu, 03 Aug 2006 19:58:43 GMT
View Forum Message <> Reply to Message

Could you answer
how the following arrow expression works in IDL?
->

Rick Towler wrote:
> You didn't say if you wanted to do this in Direct Graphics (DG) or
> Object Graphics (OG). You also don't give any details so I can't
> suggest one over the other. Given that:
>
> For either OG or DG you need to create a set of points that define the
> ellipsoid (the vertices) and an array that specifies how the points are
> connected (the polygon connectivity array). Then you pass these data to
> the appropriate function to "plot" your 3d ellipsoid.
>
> You could do this the hard way, by creating a function that would
> calculate the vertices and create the connectivity array give your major
> and minor axes and a position. Or you could do it the easy way :)
>
> IDL has the 'orb' object which creates a 3d sphere. Assuming you want
> to do this using OG, it is a simple as creating the sphere and scaling
> it asymmetrically.

>
> ; create the orb object
> IDL> orb = obj_new('orb', color=[240,0,0], style=1)
>
> ; since it is a subclass of IDLgrModel we can scale it.
> ; stretch the sphere out 2x it's original length along the z axis
> IDL> orb -> scale, 1, 1, 2
>
> ; view the result
> IDL> xobjview, orb
>
>
> If you need to do this in DG, you can still use the orb object:
>
> ; get the vertices, polygon connectivity, and transform matrix
> ; from the orb object. Even though you are looking at an ellipsoid
> ; the verts will still define a sphere. The orb's transform matrix
> ; holds the key to scaling the vertices such that they define an
> ; ellipsoid.
> IDL> orb -> getproperty, data=verts, polygons=polys, transform=xform
>
> ; apply the transform matrix to the spherical verts to make them
> ; ellipsoidal
> IDL> dgVerts = vert_t3d(verts, matrix=xform)
>
> ; display using DG
> IDL> scale3, xrange=[-2,2],yrange=[-2,2],zrange=[-2,2]
> IDL> image=polyshade(dgVerts,polys, /t3d)
> IDL> tv, image
>
> I am aware that this DG code displays a "solid" sphere. I never do 3d
> in DG so this is the best I care to do. Others might offer tips for
> displaying 3d objects in DG if you really want to suffer thru this in DG.
>
> HTH!
>
> -Rick
>
>
> adisn123@yahoo.com wrote:
>> Hi,
>>
>> I'm a begginer in IDL image processing, so if someone lends me some
>> help, that'd be great.
>>
>> I'm trying to make an ellipsoid in 3D.
>>
>> Not solid, but hollow ellipsoidal in 3D.

>>
>> Anybody help?
>>

Subject: Re: ellipsoid 3D
Posted by Jean H. on Thu, 03 Aug 2006 20:38:47 GMT
View Forum Message <> Reply to Message

adisn123@yahoo.com wrote:
> Could you answer
> how the following arrow expression works in IDL?
> ->

object -> method

you apply the method on the given object..

Jean

>
>
>
>
> Rick Towler wrote:
>
>> You didn't say if you wanted to do this in Direct Graphics (DG) or
>> Object Graphics (OG). You also don't give any details so I can't
>> suggest one over the other. Given that:
>>
>> For either OG or DG you need to create a set of points that define the
>> ellipsoid (the vertices) and an array that specifies how the points are
>> connected (the polygon connectivity array). Then you pass these data to
>> the appropriate function to "plot" your 3d ellipsoid.
>>
>> You could do this the hard way, by creating a function that would
>> calculate the vertices and create the connectivity array give your major
>> and minor axes and a position. Or you could do it the easy way :)
>>
>> IDL has the 'orb' object which creates a 3d sphere. Assuming you want
>> to do this using OG, it is a simple as creating the sphere and scaling
>> it asymmetrically.
>>
>> ; create the orb object
>> IDL> orb = obj_new('orb', color=[240,0,0], style=1)
>>
>> ; since it is a subclass of IDLgrModel we can scale it.

>> ; stretch the sphere out 2x it's original length along the z axis
>> IDL> orb -> scale, 1, 1, 2
>>
>> ; view the result
>> IDL> xobjview, orb
>>
>>
>> If you need to do this in DG, you can still use the orb object:
>>
>> ; get the vertices, polygon connectivity, and transform matrix
>> ; from the orb object. Even though you are looking at an ellipsoid
>> ; the verts will still define a sphere. The orb's transform matrix
>> ; holds the key to scaling the vertices such that they define an
>> ; ellipsoid.
>> IDL> orb -> getproperty, data=verts, polygons=polys, transform=xform
>>
>> ; apply the transform matrix to the spherical verts to make them
>> ; ellipsoidal
>> IDL> dgVerts = vert_t3d(verts, matrix=xform)
>>
>> ; display using DG
>> IDL> scale3, xrange=[-2,2],yrange=[-2,2],zrange=[-2,2]
>> IDL> image=polyshade(dgVerts,polys, /t3d)
>> IDL> tv, image
>>
>> I am aware that this DG code displays a "solid" sphere. I never do 3d
>> in DG so this is the best I care to do. Others might offer tips for
>> displaying 3d objects in DG if you really want to suffer thru this in DG.
>>
>> HTH!
>>
>> -Rick
>>
>>
>> adisn123@yahoo.com wrote:
>>
>>> Hi,
>>>
>>> I'm a begginer in IDL image processing, so if someone lends me some
>>> help, that'd be great.
>>>
>>> I'm trying to make an ellipsoid in 3D.
>>>
>>> Not solid, but hollow ellipsoidal in 3D.
>>>
>>> Anybody help?
>>>
>

>

Subject: Re: ellipsoid 3D
Posted by Rick Towler on Thu, 03 Aug 2006 23:45:53 GMT
View Forum Message <> Reply to Message

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 or 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
