Subject: Re: Is it possible a transparent image in space ???
Posted by Rick Towler on Wed, 31 Mar 2004 18:52:31 GMT
View Forum Message <> Reply to Message

Hi Antonio,

Often it is helpful to break the problem down. Below I create two texture mapped polygon objects. The "near" polygon is colored orange and is located at z=0 and is textured with a 4 pixel image whose alpha values range from 255..50. The "far" polygon lt. blue in color and is located at z=-0.5 and is textured with a 4 pixel image whose alpha values are 255, 255, 255, and 0.

```
oContainer = OBJ_NEW('IDL_Container')

;  Create the near poly texture map
idata[*,0,0]=[255,150,5,255]
idata[*,0,1]=[255,150,5,225]
idata[*,1,1]=[255,150,5,125]
idata[*,1,0]=[255,150,5,50]
oTexOne = OBJ_NEW('IDLgrImage', idata)
oContainer -> Add, oTexOne

;  Create the near polygon
verts = [[-1,-1,0], $
        [1,-1,0], $
        [1,1,0], $
        [-1,1,0]]
polys = [4,0,1,2,3]
texcoords = [[0,0],[1,0],[1,1],[0,1]]
oNearPoly = OBJ_NEW('IDLgrPolygon', verts, POLYGONS=polys, $
   COLOR=[255,255,255], TEXTURE_COORD=texcoords, $
   TEXTURE_MAP=oTexOne)



;  Create the far poly texture map
idata[*,0,0]=[0,205,170,0]
idata[*,0,1]=[0,205,170,255]
idata[*,1,1]=[0,205,170,255]
idata[*,1,0]=[0,205,170,255]
oTexTwo = OBJ_NEW('IDLgrImage', idata)
oContainer -> Add, oTexTwo

;  Create the far polygon
verts = [[-1,-1,-0.5], $
        [1,-1,-0.5], $
        [1,1,-0.5], $
```

```
        [-1,1,-0.5]]
polys = [4,0,1,2,3]
texcoords = [[0,0],[1,0],[1,1],[0,1]]
oFarPoly = OBJ_NEW('IDLgrPolygon', verts, POLYGONS=polys, $
   COLOR=[255,255,255], TEXTURE_COORD=texcoords, $
   TEXTURE_MAP=oTexTwo)


;  Add the polygon objects to a model - *note the order*
oModel = OBJ_NEW('IDLgrModel')
oModel -> Add, [oNearPoly, oFarPoly]
oContainer -> Add, oModel

stop

end
```

Copy the above code and run it. When execution halts, an IDLgrModel object, oModel, will exist and contain our polygons. Use XOBJVIEW to view the model. At the IDL command prompt type:

```
XOBJVIEW, oModel
```

Before you manipulate anything, is this what we expected? Sort-of... The orange polygon does show varying levels of transparency but we see the white background instead of the light blue polygon behind it.

If you rotate the model you will see the blue polygon. And if you rotate the model 180 degrees, now viewing it from the back, things will render as we would expect. Peering thru the one transparent quadrant of the blue polygon we see the orange polygon and thru the orange polygon we see the white background.

Back at the IDL command prompt flip the order of the two polygons in our model. Whereas before the near polygon was first and the far polygon was last (based on the order we added them to the model), flipping them will move the far polygon to the first position and the near polygon to the last:

```
oModel -> Move, 0,1
```

Now go back to the XOBJVIEW window and rotate the model another 180 degrees so you are viewing it head on again. Look different?

When using transparency in IDL you must draw your objects in order from the most -Z to the most +Z. In this case, we needed to draw the far polygon first, then the near polygon. Depending on your application is can be anywhere from moderatly annoying to a real headache. The reasons for this

have been covered in depth in this group before so if you relish the details
you can search google.

I would suggest playing around with this code, adding your own textures,
until you can produce the desired results. Then go back and take a look at
your existing code.

-Rick

P.S. Don't forget to clean up:

OBJ_DESTROY, oContainer

"Antonio Santiago" wrote ...
> Hi Rick,
>
> sorry to bother you another time but i just seeing some examples found
> in google about texture maps transparencies and i think i am doing the
same.
>
> My code is a little big (is part of a more big program) so i put some
> important lines.
> (I have one IDLgrModel and put a polygon with a texture_map and a simple
> IDLgrImage.)
>
>
> -First i create two IDLgrImage ('oXImage' and 'ximage'. 'oXImage' will
> be a texture map and 'ximage' will be a simple image put on a
IDLgrModel):
>
> Note: The first data i put into image DATA is not an alpha image. I
> create later this.
>
> ;Imagen corte X
> img_x_data = data_vol[0,*,*]
> img_x_data = REFORM(img_x_data, sizes[1], sizes[2])
> oXImage = OBJ_NEW('IDLgrImage', img_x_data, $
>   XCOORD_CONV=xconv, YCOORD_CONV=yconv, ZCOORD_CONV=zconv, $
>   PALETTE=sEstado.oPalPolygon, BLEND_FUNCTION=[3,4])
> sEstado.oXSlideImage = oXImage
> sEstado.oHolderTemp->Add, oXImage
>
> ximage = OBJ_NEW('IDLgrImage', img_x_data, $

```
>    XCOORD_CONV=xconv, YCOORD_CONV=yconv, ZCOORD_CONV=zconv, $
>    PALETTE=sEstado.oPalPolygon, BLEND_FUNCTION=[3,4])
>  sEstado.oTopModelVol->Add, ximage
>  sEstado.ximage=ximage
>
>
>  -Then i create one IDLgrPolygon and assing oXImage as a texture_map:
>
>  oXSlide = OBJ_NEW('IDLgrPolygon', COLOR=[255,255,255], $
>    [[0,0,0],[0,y,0],[0,y,z],[0,0,z],[0,0,0]], $
>    THICK=2, STYLE=2, $
>    XCOORD_CONV=xconv, YCOORD_CONV=yconv, ZCOORD_CONV=zconv, $
>    TEXTURE_MAP=oXImage, TEXTURE_COORD=[[0,0], [1,0], [1,1],[0,1], [0,0]])
>  sEstado.oVolumeModelVol->Add, oXSlide
>  sEstado.oXSlide = oXSlide
>
>  -Final when i move the polygon i caught some data, convert to alpha
>  image and assign to both texture_map image and simple image:
>
>  ...
>  c[*,*,0] = red(img_x_data[*,*])
>  c[*,*,1] = green(img_x_data[*,*])
>  c[*,*,2] = blue(img_x_data[*,*])
>  c[*,*,3] = 100
>
>  sEstado.ximage->SetProperty, DATA=c, INTERLEAVE=2, $
>    XCOORD_CONV=xconv, YCOORD_CONV=yconv, ZCOORD_CONV=zconv
>  sEstado.oXSlideImage->SetProperty, DATA=c, INTERLEAVE=2, $
>    XCOORD_CONV=xconv, YCOORD_CONV=yconv, ZCOORD_CONV=zconv
>
>
>  The result is: the simple image put on the IDLgrModel is transparent but
>  the texture map not.
>
>  I think it is the same as all examples.
>
>
>  Well, if you arrive here a lot of thanks for your patient and time :)
>
>  --------
>  Antonio.
>
```