Subject: Re: Transparency for IDLgrVolume in IDL 6.1
Posted by Karl Schultz on Wed, 20 Jul 2005 00:08:47 GMT
View Forum Message <> Reply to Message

On Tue, 19 Jul 2005 15:48:47 -0700, Mary wrote:

> Karl,
>
> thank you so much once again. Everything I saw today while playing with
> the volumes makes sense now thanks to your explanation! =)
>
> You're right, the two channels in grVolume is not suitable for my
> application due to the unwanted background opacity that it would
> create.
>
> My original code was actually with both volumes combined into 1
> IDLgrVolume object, as you just suggested. I was setting the opacity
> table with value [255,255,255] for the smaller volume and BINDGEN(255)
> for the larger volume and similar RGB tables. This didn't look
> completely ok but I could've settled for it, except for the fact that
> then the polygon object was not displayed correctly; it was going even
> thru the smaller volume. It was as if the smaller object (which should
> be opaque) was transparent with respect to the polygon and I could see
> the polygon through the volume. Or, in other words, the polygon was
> always 'in front' of the small volume, no matter what my viewing angle.
> Would this have to do with the pimento problem?

OK, it is sort of a simplification of that. All you have to do is
remember to draw your (opaque) polygon first. See the following
discussion on the IDLgrVolume ZBUFFER property.

> Remember my problem: I have the large volume, the small volume, and a
> polygon object and all 3 belong to the same IDLgrModel. I want to
> display the small volume and the polygon within the large volume. Hence,
> I want the large volume to be semi-transparent so that I can see the
> small volume and the polygon object within it. But I also want to be
> able to display the positioning of the small volume and the polygon with
> respect to each other.

There are two problems here:

1) Combining the volumes. We've found that the two channel approach
probably won't work because it doesn't do the sort of modulation you need.
There may still be a way to do it this way, but let's move on.

You'll need to figure out a way to combine your two volume datasets into
one OUTSIDE of IDLgrVolume. If both datasets use the same color table
which is a linear ramp, you might be able to just to average the voxels

together. Or, you'll have to come up with something that works for your data.

In the example below, I am going to split the voxel space into two parts. The voxel values 0-127 are assigned to your large semi-transparent volume and 128-255 are for the small opaque volume. The color table is set up so that the first 128 entries are a gray linear ramp and the last 128 are all red. The opacity table has a linear ramp in the first half and full opacity in the last half. This isn't the most efficient arrangement, obviously, but it works ok here.

2) Displaying a polygon "with" your volume data. I really think that all you needed to do here was to set the ZBUFFER property on the volume and make sure that you draw the polygon first. Basically, this property causes the visible voxels to be clipped by objects that are closer. So, if part of your opaque polygon is in front of a visible voxel, the voxel won't draw and you will see the polygon.

I think this will do what you want:

```
vol0 = congrid(bytscl(randomu((seed=0), 4, 4, 4)), 40, 40, 20)
; make a polygonal surface that winds through the volume
ISOSURFACE, vol0, 100, v, c
oPolygon = OBJ_NEW('IDLgrPolygon', v, POLYGONS=c, COLOR=[0,255,0])
; put a grey ramp in bottom half for big volume
ct = BYTARR(256,3)
ct[0:127,0] = BINDGEN(128) * 2
ct[0:127,1] = BINDGEN(128) * 2
ct[0:127,2] = BINDGEN(128) * 2
; red in top half for small volume
ct[128:255,0] = 255 ; red
; ramp in bottom half of opacity table for big volume
opac = BYTARR(256)
opac[0:127] = BINDGEN(128) / 2
; opaque in top half for small volume
opac[128:255] = 255
; create small volume
smallVol = BYTARR(10,10,10)
smallVol[2:8,2:8,2:8] = 150
; scale large vol down to lower half
vol0 = vol0 / 2
; stuff small volume in
vol0[15:24,15:24,5:14] = smallVol
oVolume = OBJ_NEW('IDLgrVolume', vol0, RGB_TABLE0=ct, $
 OPACITY_TABLE0=opac, /ZBUFFER)
oModel = OBJ_NEW('IDLgrModel')
```

oModel->Add, oPolygon
oModel->Add, oVolume
XOBJVIEW, oModel

Karl
