Subject: IDLgrVolume related
Posted by lyubo on Wed, 05 Mar 2003 17:05:19 GMT
View Forum Message <> Reply to Message

I have a grayscale volume with intensities between 0 and 255
and I use an IDLgrVolume object to display it.

I want to draw particular points inside the volume with different
color, how do I do that? Is it at all possible when I am using the
average-intensity projection (COMPOSITE_FUNCTION=3)?
Any examples or web links will be highly appreciated.

Thanks,

Lyubo

Subject: Re: IDLgrVolume related
Posted by Paul Woodford on Sun, 09 Mar 2003 05:54:09 GMT
View Forum Message <> Reply to Message

Take a look at the documentation on the volume object. The object can
hold up to four volumes, and combine them in different ways, including
an RGBA mode. The type of combination is controlled by the
volume_select keyword.

Paul

Subject: Re: IDLgrVolume related
Posted by Karl Schultz on Mon, 10 Mar 2003 15:27:08 GMT
View Forum Message <> Reply to Message

"lyubo" <lzagorch@cs.wright.edu> wrote in message
news:b4577v$8ma$1@proxy1.wright.edu...
>
> I have a grayscale volume with intensities between 0 and 255
> and I use an IDLgrVolume object to display it.
>
> I want to draw particular points inside the volume with different
> color, how do I do that? Is it at all possible when I am using the
> average-intensity projection (COMPOSITE_FUNCTION=3)?
> Any examples or web links will be highly appreciated.
>

Are your "points" defined as volumetric data or as separate geometry?

If volumetric, you could rescale your grayscale volume to use, say, 250 colors and change the color table so that the grayscale ramp runs from 0:249. Then, use the leftover color table values for your point colors. Finally, insert your points into your volume by changing the desired voxels to the appropriate color indices for your point colors. There may be interpolation problems, though.

As another poster suggested, you might be able to use a 2-channel volume and put all your point data in the second volume channel. The second volume channel has its own color table.

Or if you have separate geometry, look at the ZBUFFER keyword.

Example:

```
pro tvol
   vol = congrid(bytscl(randomu((seed=0), 4, 4, 4)), 40, 40, 20)
   oVolume = OBJ_NEW('IDLgrVolume', vol, /ZBUFFER,
OPACITY_TABLE0=BINDGEN(256) / 4)
   oOrb = OBJ_NEW('orb', POS=[20,20,10], RADIUS=15, COLOR=[255,0,0])
   XOBJVIEW, [oOrb, oVolume]
end
```

IDLgrVolume ends up creating a 2D image when it is done rendering the volume. Usually, images are just blitted to the screen without much regard for the Z buffer. But if the ZBUFFER keyword is set, the IDLgrVolume object will read the Z buffer back from the frame buffer and change the pixels in the image so that volume image pixels that are "behind" the frame buffer pixels are not drawn. It does this by setting the volume image pixel alpha to zero and telling GL to not draw pixels that have an alpha of zero. The volume object can do this because it maintains the "voxel depth" for each volume image pixel. That is, it knows how deep the first visible voxel is for each volume image pixel.

Note that it is important to draw your geometry (the orb in this case) first.

Karl
