Subject: Re: Transparency for IDLgrVolume in IDL 6.1
Posted by Karl Schultz on Mon, 18 Jul 2005 21:02:17 GMT
View Forum Message <> Reply to Message

On Mon, 18 Jul 2005 12:26:21 -0700, Danmary Sanchez wrote:

> Hello,
>
> I've read thru this group for the 'pimento problem' and 'alpha
> blending' information. It's been very interesting and educational, but
> I still haven't been able to correct the problem I'm having with
> transparencies using IDLgrVolumes. I thought upgrading to IDL 6.1 would
> make my life easier but no change so far...
>
> I have 2 separate IDLgrVolume objects (1 large volume and 1 small
> volume) and 1 IDLgrPolygon object. 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. However, I want the small volume and the polygon to
> be opaque so that I cannot see them through each other. The small
> volume and the polygon object may overlap in the 3D space, but I only
> want to see the sections that don't overlap. I.e. I would like to be
> able to visualize the depth of the small volume and the polygon within
> the large volume and their position with respect to each other.
>
> I'm trying to use IDL's 6.1 new capability for alpha channels with
> IDLgrVolume for the larger volume, but with no success. The larger
> volume is still opaque and I can't see through it. As another option,
> I've tried to create & use the RGB_TABLE & OPACITY_TABLE properties for
> the volumes but the transparency still dosn't look right.
>
> Any suggestions on how to use the new ALPHA_CHANNEL property in IDL 6.1
> correctly? Do I also need to use the other properties: RGB_TABLE,
> OPACITY_TABLE, DEPTH_CUE, DEPTH_TEST_FUNCTION, COMPOSITE_FUNCTION,
> ZBUFFER? How am I supposed to combine them?

Volumes are rendered a bit differently, and so the pimento discussion
doesn't apply in exactly the same way.

The IDL volume renderer essentially performs a ray-casting rendering of
the volume onto a 2D image. In fact, IDLgrVolume creates a sort of
light-weight IDLgrImage object that stores and renders this 2D image.
IDLgrVolume also computes some depth information based on the volume data
and writes it to the depth buffer when the grVolume is rendered. This
information is useful for rendering geometric (e.g. polygons) primitives
in such a way that the volume and polygons can block each other.

The problem is that you end up trying to depth sort the two "images" that the volume renderer creates. Since the two volumes are rendered independently, are resolved down to flat 2D images, and the images are rendered without taking into account the depth buffer, it won't work as you would like.

There might be a solution: IDLgrVolume can accept two "channels" of volume information. You can place your volume data in separate channels and IDLgrVolume will render them together according to a specific rule determined by the COMPOSITE_FUNCTION property.

The following example isn't exactly what you are trying to do, but it illustrates one way to use two channels. In this example, we use one channel pretty normally, and the other channel as a 3D "mask" to sort of take a chunk out of the volume. The second volume channel can have values of 0 or 1, to index opacity table values that are either fully opaque or fully transparent.

```
OPENR, 1, FILEPATH('head.dat', SUBDIR=['examples', 'data'])
head = BYTARR(80,100,57)
READU, 1, head
CLOSE, 1

vol0 = head
vol1 = BYTARR(80,100,57) + 1
vol1[30:50, 0:50, 26:*] = 0
oPal = OBJ_NEW('IDLgrPalette')
oPal->LoadCT, 1
oPal->GetProperty, RED_VALUES=red, GREEN_VALUES=green, BLUE_VALUES=blue
OBJ_DESTROY, oPal
rgb_table0 = [[red], [green], [blue]]
rgb_table1 = [[BYTARR(256)], [BYTARR(256)], [BYTARR(256)]]
rgb_table1[1,*] = [255,255,255]
opacity_table0 = BINDGEN(256)
opacity_table1 = BYTARR(256)
opacity_table1[1] = 255

ivolume, vol0, vol1, $
   rgb_table0=rgb_table0, $
   rgb_table1=rgb_table1, $
   OPACITY_TABLE0=opacity_table0, $
   OPACITY_TABLE1=opacity_table1
```

Notice the "gap" or "notch" cut into the skull so that you can see the detail inside.

I'm pretty sure that you can do something like this with your two sets of volume data and your polygons to get what you want. You may have to play around a bit with the opacity tables, but I think that you will get pretty far along if you just put your two volumes in, load both color tables with the same thing, and leave the opacity tables at their default settings. You may have to shift the opacity table "ramps" around to get the relative transparencies the way you want.

If the volumes are different sizes, you may have/want to pad the smaller one up to match the larger one and fill the pad with a voxel value that maps to a transparent opacity value.

Karl
