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

Subject: Re: Transparency for IDLgrVolume in IDL 6.1
Posted by Mary on Tue, 19 Jul 2005 18:46:18 GMT
View Forum Message <> Reply to Message

Hello Karl,

thank you very much for your prompt response and the information you gave me! I have a better understanding now of how the volumes are rendered in IDL.

I ran your program a couple of times using different settings for rgb0, rgb1, opacity0, and opacity1 to understand how they affect the display of the image. I think I understand it now. I tried doing something similar with the IDLgrVolume by setting data0 & data1 to each of my volumes. However, I haven't been successful at seeing both volumes within each other at the same time.

I noticed that the VOLUME_SELECT property is very important when the volume object has different data. If VOLUME_SELECT = 0 then only data0 is shown. When VOLUME_SELECT = 1 then it is supposed to combine the data0 & data1 using the rgb and opacity settings for each. I've been playing with different settings (for rgb and opacity) all last night and today. But it actually seems that IDL is only using the larger volume with the rgb/opacity settings to mask the smaller volume, no matter which one I set as data0 or data1. Does this make sense? It's sort of what happens with the the sample code you sent, where vol1 is a mask to vol0; you can't actually see both vol0 and vol1 at the same time. (Did I understand your code correctly?)

However, that's not what I want: I actually need to see both volumes, so that I can see the position of the smaller volume within the larger volume. Can this be achieved with the data0/data1 properties of IDLgrVolume? Is there any other property I need to set? (I also played

with ZERO_OPACITY_SKIP but it didn't help).

Further comments and suggestions from you would be GREATLY appreciated!

Best regards,
Mary

Subject: Re: Transparency for IDLgrVolume in IDL 6.1
Posted by Karl Schultz on Tue, 19 Jul 2005 21:10:38 GMT
View Forum Message <> Reply to Message

On Tue, 19 Jul 2005 11:46:18 -0700, Mary wrote:

> Hello Karl,
>
> thank you very much for your prompt response and the information you
> gave me! I have a better understanding now of how the volumes are
> rendered in IDL.
>
> I ran your program a couple of times using different settings for rgb0,
> rgb1, opacity0, and opacity1 to understand how they affect the display
> of the image. I think I understand it now. I tried doing something
> similar with the IDLgrVolume by setting data0 & data1 to each of my
> volumes. However, I haven't been successful at seeing both volumes
> within each other at the same time.
>
> I noticed that the VOLUME_SELECT property is very important when the
> volume object has different data. If VOLUME_SELECT = 0 then only data0
> is shown. When VOLUME_SELECT = 1 then it is supposed to combine the
> data0 & data1 using the rgb and opacity settings for each. I've been
> playing with different settings (for rgb and opacity) all last night
> and today. But it actually seems that IDL is only using the larger
> volume with the rgb/opacity settings to mask the smaller volume, no
> matter which one I set as data0 or data1. Does this make sense? It's
> sort of what happens with the the sample code you sent, where vol1 is a
> mask to vol0; you can't actually see both vol0 and vol1 at the same
> time. (Did I understand your code correctly?)

I took a short-cut by using iVolume, which does its best to deal with
VOLUME_SELECT on its own.

Read on.

>
> However, that's not what I want: I actually need to see both volumes, so
> that I can see the position of the smaller volume within the larger
> volume. Can this be achieved with the data0/data1 properties of

> IDLgrVolume? Is there any other property I need to set? (I also played
> with ZERO_OPACITY_SKIP but it didn't help).

ok, this is simpler and is probably what I should have posted
in the first place.

```
vol0 = BYTARR(10,10,10)
vol1 = BYTARR(10,10,10)
vol0[1:3,1:3,1:3] = 1  ; red
vol1[6:8,6:8,6:8] = 2  ; green
ct = BYTARR(256,3) + 255
ct[1,*] = [255,100,100]
ct[2,*] = [100,255,100]
opac = BYTARR(256) + 100
opac[1] = 255
opac[2] = 255
oVol = OBJ_NEW('IDLgrVolume', vol0, vol1, VOLUME_SELECT=1, $
RGB_TABLE0=ct, RGB_TABLE1=ct, OPACITY_TABLE0=opac, OPACITY_TABLE1=opac)
xobjview, oVol
```

I may have been slightly incorrect when implying that you can simply
combine two volume datasets in dual volume mode.

What is important is how they are combined. Here are part of the docs for
VOLUME_SELECT:

+++

Render dual volume from DATA0 and DATA1.
The value at voxel [i,j,k] is modulated between the two data sets as
follows:

```
srcRed[i,j,k] = (RGB_TABLE0[DATA0[i,j,k],0] *
              RGB_TABLE1[DATA1[i,j,k],0]) / 255
srcGreen[i,j,k] = (RGB_TABLE0[DATA0[i,j,k],1] *
               RGB_TABLE1[DATA1[i,j,k],1]) / 255
srcBlue[i,j,k] = (RGB_TABLE0[DATA0[i,j,k],2] *
              RGB_TABLE1[DATA1[i,j,k],2]) / 255
srcAlpha[i,j,k] = (OPACITY_TABLE0[DATA0[i,j,k]] *
               OPACITY_TABLE1[DATA1[i,j,k]]) / 255
```

+++

It is important to understand these equations.

Note that if you try to combine volume data that is non-zero in one
dataset with volume data that is zero in the other dataset (in the same

volume location), you will end up with zero. Note that this multiplication happens after color table lookup.

That's why I filled both of my color tables with white as a "default" value and put red at index 1 and green at index 2. This keeps the results of the modulation from getting wiped out.

The downside is that you'll probably get unwanted background opacity where you don't want it, as illustrated by the grey in the above example.

You might be better off combining your volumes yourself and then giving IDLgrVolume a single combined volume. You'll have control over the combining process. I think that the two-channel mode in grVolume is really for using one volume to modulate the other and not for combining unrelated volumes.

Karl

Subject: Re: Transparency for IDLgrVolume in IDL 6.1
Posted by Mary on Tue, 19 Jul 2005 22:48:47 GMT
View Forum Message <> Reply to Message

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?

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.

So, my questions now are: will I have to convert my polygon to a volume
to be able to display them all together? Is there an easy (IDL) method
for this? Or will I have to break up my polygon into different sections
and each section separately depending on the angle of view ('pimento
problem')? Is there an easy (IDL) method for this? Or do you have any
other (easier to implement) idea/suggestion for this?

Thanks again and best regards,
Mary

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

Subject: Re: Transparency for IDLgrVolume in IDL 6.1
Posted by Mary on Wed, 20 Jul 2005 20:22:25 GMT
View Forum Message <> Reply to Message

Worked like a charm! =) As you suspected, the problem was that I was adding the combined volume object to the model before adding the polygon.

Thanks for all the help!!!
Mary
