
Subject: Question about volume objects and opacity
Posted by [Klemens Barfus](#) on Fri, 30 Sep 2005 17:31:29 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dear list members,

trying to visualize three dimensional cloud fields, I started to work with object graphics. I created a testcase to get the differences between volume objects and polygon objects to decide which object to use.

What I do not understand is the attribute opacity of the volume objects. In the code below opacity is 255 and therefor all the volumes should be opac and have the same colour. The polygon graphic should look like the volume graphic, or not ?

But in the areas where there are more voxels in direction of the view, colour is a little bit more solid then in the areas where there is just one filled voxel.

Is there something wrong in my understanding of volume objects ?

Thanks for your help in advance !

Klemens

pro cloudfield_visualization_2

```
oModel_vol = obj_new('IDLgrModel')
```

```
; volume
```

```
voltest = bytarr(8,8,10)  
voltest(*,*,*) = 0
```

```
voltest(0:3,*,3) = 80  
voltest(4:5,0,3) = 80  
voltest(4:5,3:7,3) = 80  
voltest(6:7,*,3) = 80
```

```
voltest(1:2,0:1,4) = 80  
voltest(1,3,4) = 80  
voltest(1,6,4) = 80  
voltest(6,0,4) = 80  
voltest(6:7,1,4) = 80  
voltest(6:7,5:7,4) = 80
```

```
voltest(1,4,5) = 80
```

```
voltest(7,1,5) = 80
voltest(6:7,6:7,5) = 80
```

```
opacity = bytarr(256)
opacity[80] = 255B
opacity[0] = 0B
```

```
rgb_table0 = bytarr(256,3)
rgb_table0[80,*] = [255B,0B,0B]
rgb_table0[0,*] = [0B,255B,0B]
```

```
oVol_vol =
  obj_new('IDLgrVolume',voltest,opacity_table0=opacity,rgb_table0=rgb_table0)
```

```
oModel_vol->add,oVol_vol
```

```
; view
```

```
oView_vol =
  obj_new('IDLgrView',viewplane_rect=[-1,-1,12,12],color=[255, 255,255],zclip=[20,-20],eye=800)
```

```
oView_vol->add, oModel_vol
```

```
oModel_vol->rotate,[1,0,0],-89
```

```
oWindow_vol =
  obj_new('IDLgrWindow',dimension=[400,400],renderer=1,retain= 2,title='Volume_example')
```

```
oWindow_vol->draw, oView_vol
```

```
; surface_example
```

```
cubetest = intarr(8,8,10)
cubetest(*,*,*) = 0
```

```
cubetest(0:3,*,3) = 80
cubetest(4:5,0,3) = 80
cubetest(4:5,3:7,3) = 80
cubetest(6:7,*,3) = 80
```

```
cubetest(1:2,0:1,4) = 80
cubetest(1,3,4) = 80
```

```
cubetest(1,6,4) = 80
cubetest(6,0,4) = 80
cubetest(6:7,1,4) = 80
cubetest(6:7,5:7,4) = 80
```

```
cubetest(1,4,5) = 80
cubetest(7,1,5) = 80
cubetest(6:7,6:7,5) = 80
```

```
oModel_sur = obj_new('IDLgrModel')
```

```
for i = 0, 7 do begin
  for j = 0, 7 do begin
    for k = 0, 9 do begin
      if(cubetest(i,j,k) eq 80)then begin
        ;offset=[i,j,k-4]
        makeCube, verts, connectivity, offset=[i,j,k]
        oPoly = obj_new('IDLgrPolygon', verts,
style=1,polygons=connectivity, color=[255,0,0])
        f = [255,0,0]
        ;vc =
[[f,f,f],[f,f,f],[f,f,f],[f,f,f],[f,f,f],[f,f,f],[f,f,f],[f,f,f]]
        oPoly->setProperty, style=2, shading=0, vert_color=vc
        oModel_sur->add, oPoly
      endif
    endfor
  endfor
endfor
```

```
oModel_sur->rotate,[1,0,0],-89
```

```
oView_sur =
obj_new('IDLgrView',viewplane_rect=[-1,-1,12,12],color=[255, 255,255],zclip=[20,-20],eye=800)
```

```
oView_sur->add, oModel_sur
```

```
oWindow_sur =
obj_new('IDLgrWindow',dimension=[400,400],renderer=1,retain= 2,title='Surface_example')
```

```
oWindow_sur-> draw, oView_sur
```

```
end
```

```
pro makeCube, verts, connectivity, scale=scale, offset=offset
```

```
; creates the vertices and faces on a unit cube  
; this can be scaled and offset using the keywords
```

```
; vertices in unit coordinates
```

```
verts = $  
  [[0.0,0.0,0.0],[1.0,0.0,0.0],[1.0,1.0,0.0],[0.0,1.0,0.0],[0.  
0,0.0,1.0],[1.0,0.0,1.0],[1.0,1.0,1.0],[0.0,1.0,1.0]]
```

```
; connectivity list for the vertices
```

```
connectivity = [4,3,0,4,7,$  
               4,0,1,5,4,$  
               4,4,5,6,7,$  
               4,1,2,6,5,$  
               4,2,3,7,6,$  
               4,3,2,1,0]
```

```
; if the scale keyword is set then use it.
```

```
if keyword_set(scale) then verts=verts*scale
```

```
; if the offset keyword is present then apply it
```

```
if (n_elements(offset) ne 0) then begin
```

```
  verts(0,*)=verts(0,*) + offset(0)
```

```
  verts(1,*)=verts(1,*) + offset(1)
```

```
  verts(2,*)=verts(2,*) + offset(2)
```

```
endif
```

```
return
```

```
end
```
