Subject: object graphics - transparent surfaces
Posted by greg michael on Tue, 30 May 2006 07:20:30 GMT
View Forum Message <> Reply to Message

I'm trying to make some 3d red/blue anaglyphs using object graphics. I manage to blend the red surface onto the blue using the alpha channel and /depth_test_disable on the second surface, something like:

```
oSurface2 = OBJ_NEW('IDLgrSurface', data,x,y, style=2, alpha=.5, $
  color=[255,255,255],texture_map=oImage2,shading=1)
oModel2 = OBJ_NEW('IDLgrModel',/depth_test_disable)
oModel2->add,oSurface2
```

In most cases it works fine, but occasionally I get ugly pure-red bands on the tops of steep ridges. Looking more closely at the idlgrsurface alpha_channel documentation, it seems to be saying this technique is not recommended... some facets may be rendered in the wrong order for transparency. So is there a better way?

I could render one, read it back, then the other, read it back, blend them myself, and then display. But that wouldn't be a nice way to make an interactive object. By the way, I'm using 6.1, just in case anyone happens to know whether it's been changed since.

regards,
Greg

Subject: Re: object graphics - transparent surfaces
Posted by Rick Towler on Tue, 06 Jun 2006 15:49:27 GMT
View Forum Message <> Reply to Message

greg michael wrote:

> Certainly I can add my request to RSI - I suppose, from the wording of
> the documentation, they already think it's something to be fixed.

Well, not necessarily. This is a difficult issue to fix in a x-platform way. I would *really* like to see it happen, but I am not holding my breath.

> I'd be very interested to take a look at your meshing routines, and
> your camera object, too.

I'll package it up and send you the link.

-Rick

Subject: Re: object graphics - transparent surfaces
Posted by Rick Towler on Tue, 06 Jun 2006 20:22:13 GMT
View Forum Message <> Reply to Message

greg michael wrote:

> I'd be very interested to take a look at your meshing routines

Here it is... I don't have time to generalize it but it is fairly straightforward. You should be able to work up a front end that takes your data and reforms it into a 3xN array required for meshing/display.

Currently this meshes from 0,0 in rows from far to near (you can actually use the location keyword to modify this). You'll need to change the order of meshing by playing around with v1 and v2 to change the corner that meshing starts and its direction.

You could also use a combination of rotations to your data, specifying the location to the meshing routine, and transformations afterwards to get the same effect. In practice you would have 4 sets of vertices with 4 transform matrices which you would use to render the object as viewed from the 4 directions. I included just that starting bits of this for you below in the demo. The first surface renders correctly head on, the second renders correctly after you rotate it 180 degrees. What isn't apparent is that these surfaces are in different positions in world space. That is where the transformation would come in.

Hope this helps.

-Rick

depth_data (input) is a 2d array of surface elevations
data (output) is the array that will receive the vertex data
polygons (output) is the array that will receive the polygon
    connectivity data
location (optional in) defines the, ummm..., location of the far
    left corner in world space? And the surface size. Yeah,
    something like that :) [X,Z,extent in X,extent in Z]
nTriangles (optional out) returns the output of MESH_NUMTRIANGLES
use_triangles (optional in) set this keyword to create a
    mesh comprised of triangles. Default is quads (quads render
    faster on my hardware)

```
pro mesh_surface,   depth_data, $
                    data, $
```

```
            polygons, $
            location=location, $
            nTriangles=nTriangles, $
            use_triangles=useTriangles

compile_opt idl2

dims = SIZE(depth_data)

useTriangles = (N_ELEMENTS(useTriangles) eq 0) ? 0 : $
   KEYWORD_SET(useTriangles)
if (N_ELEMENTS(location) ne 4) then location=[0,0,dims[1],dims[2]]

;convert 2d data to 3xN
data = FLTARR(3,dims[4], /nozero)
for n = 0L, dims[2] - 1L do begin
   s = n * dims[1]
   e = s + dims[1] - 1L
   data[0,s:e] = FINDGEN(dims[1]) * ((location[2]-location[0]) / $
      (dims[1] - 1)) + location[0]
   data[1,s:e] = depth_data[*,n]
   data[2,s:e] = n * ((location[3]-location[1]) / $
      (dims[2] - 1)) + location[1]
endfor

;create the surface mesh
if (useTriangles) then begin
   ;create triangle strip mesh
   nconn = (dims[2] - 1L) * (8L * (dims[1] - 1L))
   nstrip = (8L * (dims[1] - 1L))
   polygons = FLTARR(nconn, /nozero)
   v1 = 0L
   v2 = LONG(dims[1])
   np = 0L
   while (np lt nconn-1) do begin
      for ns = 0L, nstrip-1, 8L do begin
         n = ns + np
         polygons[n:n+3L] = [3, v1, v1+1L, v2]
         polygons[n+4L:n+7L] = [3, v2, v1+1L, v2+1L]
         v1 = v1 + 1L
         v2 = v2 + 1L
      endfor
      np = np + nstrip
      if (np lt nconn-1) then begin
         ;mesh back row
         v1 = v1 + (2L * dims[1])
         for ns = 0L, nstrip-1, 8L do begin
            n = ns + np
```

```
                polygons[n:n+3L] = [3, v2-1L, v2, v1]
                polygons[n+4L:n+7L] = [3, v2-1L, v1, v1-1L]
                v1 = v1 - 1L
                v2 = v2 - 1L
            endfor
            v2 = v2 + (2L * dims[1])
            np = np + nstrip
        endif
    endwhile
endif else begin
    ;  Create quad strip mesh
    nconn = (dims[2] - 1) * (5 * (dims[1] -1))
    nstrip = (5L * (dims[1] - 1L))
    polygons = FLTARR(nconn, /nozero)
    v1 = 0L
    v2 = LONG(dims[1])
    np = 0L
    while (np lt nconn-1) do begin
        for ns = 0L, nstrip-1, 5L do begin
            n = ns + np
            polygons[n:n+4L] = [4, v1, v1+1L, v2+1, v2]
            v1 = v1 + 1L
            v2 = v2 + 1L
        endfor
        np = np + nstrip
        if (np lt nconn-1) then begin
            ;  Mesh back row
            v1 = v1 + (2L * dims[1])
            for ns = 0L, nstrip-1, 5L do begin
                n = ns + np
                polygons[n:n+4L] = [4, v2-1L, v2, v1, v1-1L]
                v1 = v1 - 1L
                v2 = v2 - 1L
            endfor
            v2 = v2 + (2L * dims[1])
            np = np + nstrip
        endif
    endwhile
endelse

nTriangles = MESH_NUMTRIANGLES(polygons)

end


pro mesh_example

    data = READ_IMAGE(!dir + '\examples\data\rbcells.jpg')
```

```
    data = CONGRID(data, 256,256) / 64.
    dDims = SIZE(data, /DIMENSIONS)

    mesh_surface, data, verts, polys, LOCATION=[0,0,dDims[0], dDims[1]]

    oPolygon = OBJ_NEW('IDLgrPolygon', verts, POLYGONS=polys, $
       COLOR=[200,200,200], ALPHA=0.3)
    oModel = OBJ_NEW('IDLgrModel')
    oModel -> Add, oPolygon

    XOBJVIEW, oModel, /BLOCK

    data = ROTATE(data, -90)
    mesh_surface, data, verts, polys, $
       LOCATION=[dDims[0], dDims[1], -dDims[0], -dDims[1]]
    oPolygon -> SetProperty, DATA=verts

    XOBJVIEW, oModel, /BLOCK

    OBJ_DESTROY, oModel

end
```

Subject: Re: object graphics - transparent surfaces
Posted by Rick Towler on Thu, 08 Jun 2006 19:32:45 GMT
View Forum Message <> Reply to Message

greg michael wrote:

> I'd be very interested to take a look at your meshing routines, and
> your camera object, too.

Here's a link to a package containing the code and a demo. Take it for what it is worth. The demo was not written for distribution and as written relies on IDL 6.3 (IDLsysMonitorInfo class which can be removed and you can fake it) and also a couple of my windows only dlms. I have provided pre 6.3 and 6.3 win32 dlms. All other dlms will need to be built for non-windows platforms. The code can be modified to work on these platforms but it will take a bit of work.

http://www.acoustics.washington.edu/~towler/programs/RHTgrCa mera_StereoPackage.zip

-Rick

Subject: Re: object graphics - transparent surfaces

Posted by greg michael on Fri, 16 Jun 2006 09:42:48 GMT
View Forum Message <> Reply to Message

Hi Rick,

Thanks very much indeed for all this stuff - that's very kind of you. I haven't had a chance to look into it all yet, but I'll let you know when I get stuck...

many greetings from Berlin,
Greg

Subject: Re: object graphics - transparent surfaces
Posted by Michael Galloy on Wed, 28 Jun 2006 15:42:34 GMT
View Forum Message <> Reply to Message

greg michael wrote:
> Hi Mike,
>
> Is your pref_get routine available somewhere? Couldn't find it on your
> site...
>
> kind regards,
> Greg
>

Greg,

PREF_GET was introduced in IDL 6.2. I used it as a platform independent way to get the default window size. I had some kind of difficulty in just inheriting from IDLgrWindow and getting the default size if the DIMENSION keyword wasn't passed anything. Let me see if it's possible to not use PREF_GET.

Mike
--
www.michaelgalloy.com