Subject: Polygon Problems
Posted by Struan Gray on Wed, 29 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

This may be related to Rick Towler's problem (see "Q:how to use MESH_OBJ") but I'm not sure so I've started a new thread.

I'm playing with IDLgrPolygon objects and have hit two blank walls simultaneously. I can't decide if it's me or IDL which is being dumb.

I am making Platonic solids, and want to have each solid in a single IDLgrPolygon object, using a vertex array and polygon array to specify the vertices and faces. The first problem is that IDL only seems able to associate normal vectors with vertices, and not with faces.

Note that the default method IDL uses to compute and store the normal vectors used in the light-shading calculations takes each vertex and calculates an average of the surface normals of the polygons which share that vertex. This is an appropriate and useful way to do things for smoothly-varying mesh structures, but is totally wrong for sharp corners.

This doesn't matter for polygons like tetrahedra and cubes because they have more vertices than faces, so provided you make sure that each face as a unique first vertex in the polygon array you can calculate your own normals and the light-shading routines happily use them instead of the defaults.

However, if the polygon has more faces than vertices there is nowhere to store some of the normal vectors. Something as simple as a regular octahedron (eight faces, six vertices) becomes impossible to display correctly as a light-shaded solid 3D model. Oops.

The only way round this that I have found is to make duplicate vertices and/or split the polygon into two seperate ones. This is not only ugly, it complicates the setting of parameters, confuses programmatic ownership of the vertex data and is going to make my life dirty and slow when I start calculating things like whether arbitrary 3D points lie inside or outside the polygon.

My second problem is that I would like to display my polygons with an edge colour which is different from the face colour. I would also like to do this with IDLgrSurface objects, which seems to be a related problem.

For both types of object you can either colour the edges, or the faces, but I can find no combination of properties, texture maps or

whatever which lets me do both. I have made an ugly hack where I share data with a second IDLgrPolygon object or with an IDLgrPolyline, which draws the lines in the right places, but the plotted lines look spotty, even when I make them several pixels thick and independently of the order in which I plot the two objects.

The *really* annoying thing is that several of the object graphics demos (like the teapot, knot and shell) do this very nicely, but because they are locked up in .sav and .dat files I can't tell how they do it. This at least tells me that my problem is not the OpenGL implementation on my machines.

Below is a function which makes an octahedron. If anyone can come up with an elegant way to a) plot it correctly as a shaded solid object, and b) colour the edges differently from the faces, I will be forever in their service.

Struan

```
; ************************************************************ ***

function Octahedron

; make a unit octahedron centred on [0,0,0]

  rt2 = sqrt(2.0)

  vertex_array = [       $
    [ 0.0,  0.0,  rt2],  $
    [ rt2,  0.0,  0.0],  $
    [ 0.0,  rt2,  0.0],  $
    [-rt2,  0.0,  0.0],  $
    [ 0.0, -rt2,  0.0],  $
    [ 0.0,  0.0, -rt2]   $
  ]

  poly_array = [   $
    [3, 0, 1, 2],  $
    [3, 2, 3, 0],  $
    [3, 3, 4, 0],  $
    [3, 4, 1, 0],  $
    [3, 5, 1, 4],  $
    [3, 1, 5, 2],  $
    [3, 5, 3, 2],  $
    [3, 5, 4, 3]   $
  ]
```

```
  return, obj_new('IDLgrPolygon', $
    data=vertex_array, polygons=poly_array)

end

; ************************************************************* ***
```