Subject: Re: "Color vectors" & shading
Posted by Karl Schultz on Wed, 21 Jan 2004 00:54:47 GMT
View Forum Message <> Reply to Message

"Neil" <nasalmon@onetel.net.uk> wrote in message
news:74039481.0401201339.d26efa4@posting.google.com...
> Does anyone know how i correctly implement "color vectors" in
> "setProperty" to give each polygon of an object its own predetermined
> intensity level or colour? It should be that setting "shading" = 0 in
> "setProperty" should make it so that the colour of each polygon is
> defined by the "colour vertex" of the first vertex of the polygon
> connectivity. However, generally the number of "first vertices" of a
> polygon of an object is actually less than the number of polygons.
> That means you cant use this method to give each polygon of an object
> a predetermined different colour - or have i missed something? It
> seems to me that what is really needed is a colour vector for the
> polygons not the vertices.

There's no way to do this today if you have vertices that are shared between polygons. There isn't a polygon color vector.

What you could do is make a vertex list and a connectivity list that creates a set of independent polygons and then set the color of all the vertices belonging to each polygon to the same color. Actually, you would only need to set the first vertex of a polygon to an actual color; the rest are ignored if you are doing flat shading.

Obviously, if you have many polygons that used to share vertices, this will require more space.

For example, if you have 4 vertices, A, B, C, D, that form 2 triangles with a shared edge, the triangles might be ABC, and DCB. If the vertices are in the vertex list as ABCD, the conn list would be 3 0 1 2 3 3 2 1. To make them independent triangles, your vertex list would be ABCDCB and the conn list 3 0 1 2 3 3 4 5. You would set the color of vert A (at 0) to the desired color and the color of vert D (at 3) to the desired color. The rest of the vertex colors list can be left uninitialized.

It isn't very efficient, but would work well if you don't have that many polygons.

Karl

Subject: Re: "Color vectors" & shading
Posted by nasalmon on Wed, 21 Jan 2004 08:24:38 GMT
View Forum Message <> Reply to Message

"Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message
news:<100rj8ksoii7b80@corp.supernews.com>...
> "Neil" <nasalmon@onetel.net.uk> wrote in message
> news:74039481.0401201339.d26efa4@posting.google.com...
>> Does anyone know how i correctly implement "color vectors" in
>> "setProperty" to give each polygon of an object its own predetermined
>> intensity level or colour? It should be that setting "shading" = 0 in
>> "setProperty" should make it so that the colour of each polygon is
>> defined by the "colour vertex" of the first vertex of the polygon
>> connectivity. However, generally the number of "first vertices" of a
>> polygon of an object is actually less than the number of polygons.
>> That means you cant use this method to give each polygon of an object
>> a predetermined different colour - or have i missed something? It
>> seems to me that what is really needed is a colour vector for the
>> polygons not the vertices.
>
> There's no way to do this today if you have vertices that are shared between
> polygons. There isn't a polygon color vector.
>
> What you could do is make a vertex list and a connectivity list that creates
> a set of independent polygons and then set the color of all the vertices
> belonging to each polygon to the same color. Actually, you would only need
> to set the first vertex of a polygon to an actual color; the rest are
> ignored if you are doing flat shading.
>
> Obviously, if you have many polygons that used to share vertices, this will
> require more space.
>
> For example, if you have 4 vertices, A, B, C, D, that form 2 triangles with
> a shared edge, the triangles might be ABC, and DCB. If the vertices are in
> the vertex list as ABCD, the conn list would be 3 0 1 2 3 3 2 1. To make
> them independent triangles, your vertex list would be ABCDCB and the conn
> list 3 0 1 2 3 3 4 5. You would set the color of vert A (at 0) to the
> desired color and the color of vert D (at 3) to the desired color. The rest
> of the vertex colors list can be left uninitialized.
>
> It isn't very efficient, but would work well if you don't have that many
> polygons.
>
> Karl

Karl,

many thanks for your response.

It's a shame there are no polygon colour vectors. If you think about
nature and surfaces, it is the surface that has the colour. The vertex
is just a manmade mathematical entity to descibe the surfaces. I am

sure it would be the wish of graphics programmes to mimic nature as cloesly as possible in the most efficient way.

Is the absence of polygon colour vectors a fundamental limitation of IDL? Does OpenGL have this limitation? Would there be any plans for RSI to introduce an efficient method in IDL to enable each polygon to have a predetermined and unique colour/intensity?

I can certainly create an artificial polygon_connectivity for unconnected polygons allows each polygon to have a different intensity.

best regards, Neil

Subject: Re: "Color vectors" & shading
Posted by David.Chevrier on Wed, 21 Jan 2004 16:45:28 GMT
View Forum Message <> Reply to Message

Hi,
I'm not sure if this will help or not, but I had a similar (less complex) problem. I had three arrays: long/lat/depth and wanted to create one semi-transparent polygon that was green for all points above 0 (land) and blue for all points less than 0 (water). This created a bathymetric map. Here is how I did it. I know its only two colors and uses a lot of memory (my arrays had more than 9000 numbers each.) (I also used the 'temporary' command a lot to get rid of unneeded variables, but have taken most of them out here to make it easier to read.)

```
 MESH_OBJ,0,verts,conn,TRANSPOSE([[contourlng],[contourlat],[ contourdepth]])
verts=MESH_SMOOTH(TEMPORARY(verts), conn, /FIXED_EDGE_VERTICES)
depsize=(SIZE(verts))[2]
mapimage=REPLICATE(255b,4,3,depsize)
mapimage[3,*,*]=105b  ;transparent level

oContourImage=OBJ_NEW('IDLgrImage', mapimage, BLEND_FUNCTION=[3,4],
INTERLEAVE=0, /NO_COPY)

r=MAKE_ARRAY(256,/BYTE,VALUE=0b)
g=MAKE_ARRAY(256,/BYTE,VALUE=0b)
b=MAKE_ARRAY(256,/BYTE,VALUE=255b)

depths=REFORM(verts[2,*])
lands=WHERE(depths GE 0.0, anyland)
depths=BYTSCL(TEMPORARY(depths))
IF (anyland GT 0L) THEN BEGIN
```

```
  depths[TEMPORARY(lands)]=255b
  g[255]=255b
  b[255]=0b
ENDIF

oPalette1=OBJ_NEW('IDLgrPalette',r,g,b)
oWaterMap=OBJ_NEW('IDLgrPolygon',verts, POLYGONS=conn, $
  SHADING=1, TEXTURE_MAP=oContourImage, /TEXTURE_INTERP,
HIDDEN_LINES=1, $
  XCOORD_CONV=(*stateptr).xs, VERT_COLORS=depths, PALETTE=oPalette1, $
  YCOORD_CONV=(*stateptr).ys, ZCOORD_CONV=(*stateptr).zs)
```

Subject: Re: "Color vectors" & shading
Posted by Karl Schultz on Wed, 21 Jan 2004 18:41:24 GMT
View Forum Message <> Reply to Message

"Neil" <nasalmon@onetel.net.uk> wrote in message
news:74039481.0401210024.11f5b707@posting.google.com...
> "Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message
news:<100rj8ksoii7b80@corp.supernews.com>...
>> "Neil" <nasalmon@onetel.net.uk> wrote in message
>> news:74039481.0401201339.d26efa4@posting.google.com...
>>> Does anyone know how i correctly implement "color vectors" in
>>> "setProperty" to give each polygon of an object its own predetermined
>>> intensity level or colour? It should be that setting "shading" = 0 in
>>> "setProperty" should make it so that the colour of each polygon is
>>> defined by the "colour vertex" of the first vertex of the polygon
>>> connectivity. However, generally the number of "first vertices" of a
>>> polygon of an object is actually less than the number of polygons.
>>> That means you cant use this method to give each polygon of an object
>>> a predetermined different colour - or have i missed something? It
>>> seems to me that what is really needed is a colour vector for the
>>> polygons not the vertices.
>>
>> There's no way to do this today if you have vertices that are shared
between
>> polygons. There isn't a polygon color vector.
>>
>> What you could do is make a vertex list and a connectivity list that
creates
>> a set of independent polygons and then set the color of all the vertices
>> belonging to each polygon to the same color. Actually, you would only
need
>> to set the first vertex of a polygon to an actual color; the rest are
>> ignored if you are doing flat shading.
>>
>> Obviously, if you have many polygons that used to share vertices, this

will
>> require more space.
>>
>> For example, if you have 4 vertices, A, B, C, D, that form 2 triangles
with
>> a shared edge, the triangles might be ABC, and DCB. If the vertices are
in
>> the vertex list as ABCD, the conn list would be 3 0 1 2 3 3 2 1. To
make
>> them independent triangles, your vertex list would be ABCDCB and the
conn
>> list 3 0 1 2 3 3 4 5. You would set the color of vert A (at 0) to the
>> desired color and the color of vert D (at 3) to the desired color. The
rest
>> of the vertex colors list can be left uninitialized.
>>
>> It isn't very efficient, but would work well if you don't have that many
>> polygons.
>>
>> Karl
>
> Karl,
>
> many thanks for your response.
>
> It's a shame there are no polygon colour vectors. If you think about
> nature and surfaces, it is the surface that has the colour. The vertex
> is just a manmade mathematical entity to descibe the surfaces. I am
> sure it would be the wish of graphics programmes to mimic nature as
> closely as possible in the most efficient way.

I suppose. But objects can vary in color across the extents of a single polygon. It would be cool to be able to specify the color of every pixel on an object, but that is what texture mapping is all about. So, the next best thing is to specify the color at each vertex and allow interpolation between vertices. This is a pretty effective way of handling lighting models. I guess this is all pretty debatable.

> Is the absence of polygon colour vectors a fundamental limitation of
> IDL?

Yes. It is just a limitation of the IDLgrPolygon object.

> Does OpenGL have this limitation?

No. OpenGL has the concept of "current color" as IDL passes each vertex to it. IDL would simply set the color of the first vertex before sending the first vertex to OpenGL, and then not change the color for subsequent

vertices. IDL would just have to fetch the color from the polygon color list instead of the vertex color list.

> Would there be any plans for
> RSI to introduce an efficient method in IDL to enable each polygon to
> have a predetermined and unique colour/intensity?

I can write a feature request.

The same sort of argument also applies to normals. If you are doing smooth shading, IDL uses the vertex normals you pass or computes them, based on the average geometric normal of polygons that use each vertex. If you are doing flat shading, IDL computes the geometric normal of each polygon and uses that. So, there is no way to pass in polygon normals directly. You'd have to do the same expansion trick we have discussed for the polygon colors and make all the vertex normals the same as the polygon normal you'd like to have for the polygon.

> I can certainly create an artificial polygon_connectivity for
> unconnected polygons allows each polygon to have a different
> intensity.

Yes, this should work just fine, except for the efficiency issue already discussed.

Karl