Lighting issues in OpenGL

Asked 16 years, 1 month ago Modified 4 years, 10 months ago

Viewed 3k times

3

I have a triangle mesh that has no texture, but a set color (sort of blue) and alpha (0.7f). This mesh is run time generated and the normals are correct. I find that with lighting on, the color of my object changes as it moves around the level. Also, the lighting doesn't look right. When I draw this object, this is the code:

```
glEnable( GL_COLOR_MATERIAL );
float matColor[] = { cur->GetRed(), cur-
>GetGreen(), cur->GetBlue(), cur->GetAlpha() };
float white[] = { 0.3f, 0.3f, 0.3f, 1.0f };
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,
matColor);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,
white);
```

Another odd thing I noticed is that the lighting fails, when I disable `GL_FRONT_AND_BACK` and use just `GL_FRONT` or `GL_BACK`. Here is my lighting setup (done once at beginning of renderer):

```
m_lightAmbient[] = { 0.2f, 0.2f, 0.2f, 1.0f };
m_lightSpecular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
m_lightPosition[] = { 0.0f, 1200.0f, 0.0f, 1.0f };

glLightfv(GL_LIGHT0, GL_AMBIENT, m_lightAmbient);
glLightfv(GL_LIGHT0, GL_SPECULAR,
m_lightSpecular);
```

```
glLightfv(GL_LIGHT0, GL_POSITION,
m_lightPosition);
```

EDIT: I've done a lot to make the normals "more" correct (since I am generating the surface myself), but the objects color still changes depending where it is. Why is this? Does openGL have some special environment blending I don't know about?

EDIT: Turns out the color changing was because a previous texture was on the texture stack, and even though it wasn't being drawn, `glMaterialfv` was blending with it.

opengl

Share

Improve this question

Follow

edited Feb 10, 2020 at 13:58

peterh

12.6k ● 20 ● 89 ● 113

asked Nov 7, 2008 at 14:07

DavidG

1,797 ● 4 ● 21 ● 33

8 Answers

Sorted by: Highest score (default)

If your lighting fails when GL_FRONT_AND_BACK is disabled it's possible that your normals are flipped.

1

Share Improve this answer

Follow

answered Nov 7, 2008 at 14:14

mwahab

237 ● 2 ● 12

but surely then it would work with just 1 of GL_FRONT or GL_BACK... which it doesnt. – DavidG Nov 7, 2008 at 15:18

1

Could you post the code that initializes OpenGL? You're saying that all other meshes are drawn perfectly? Are you rendering them simultanously?

Share Improve this answer

Follow

answered Nov 7, 2008 at 17:12

Sebastian

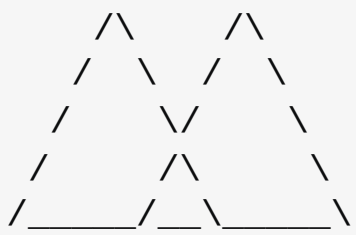
2,942 ● 4 ● 34 ● 37

1

@response to stusmith:

Z-testing won't help you with transparent triangles, you'll need per-triangle alpha sorting too. If you have an object that at any time could have overlapping triangles facing the camera (a concave object) you must draw the farthest triangles first to ensure blending is performed correctly, since Z-testing doesn't take transparency into account. Consider these two overlapping (and transparent) triangles and think about what happens when that little overlapped region is drawn, with or without Z-testing.

You'll probably reach the conclusion that the drawing order does, in fact, matter. Transparency sucks :P



I'm not convinced that this is your problem, but alpha sorting is something you need to take into account when dealing with partly transparent objects.

Share Improve this answer Follow

edited Nov 10, 2008 at 9:44

answered Nov 10, 2008 at 9:31

korona
2,229 ● 1 ● 22 ● 38

i've gone now to the lengths of disabling the transparency. still getting no lighting. :S – DavidG Nov 10, 2008 at 9:48

Yeah like I said this might not be the problem you're facing here, but it's something you need to consider at some point. I'll get back to you if I can think of what could be causing this for you... – korona Nov 10, 2008 at 11:00

1

Turns out the color changing was because a previous texture was on the texture stack, and even though it wasn't being drawn, glMaterialfv was blending with it.

Share Improve this answer

Follow

answered Nov 20, 2019 at 6:38

DavidG

1,797 ● 4 ● 21 ● 33

0

If your triangles are alpha-blended, won't you have to sort your faces by z-order from the camera? Otherwise you could be rendering a face at the back of the object on top of a face at the front.

Share Improve this answer

Follow

answered Nov 7, 2008 at 14:39

stusmith

14.1k ● 8 ● 58 ● 89

> it's 1 draw call for that object and z testing is on... works perfectly for the other meshes in the game (semi-transparent textured balls), but this particular mesh doesn't work properly. there are many differences between this and the others, been thru all options :S – DavidG Nov 7, 2008 at 14:41

0

@sebastion: multiple draw calls, each object gets a glDrawArrays. some are textured, some colored, all with normals. gl init code is: glMatrixMode(GL_MODELVIEW);

```
// Vertices!
glEnableClientState(GL_VERTEX_ARRAY);
```

```
// Depth func
glEnable(GL_DEPTH_TEST);
glDepthFunc( GL_LESS );

// Enable alpha blending
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

// Lighting
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_AMBIENT, m_lightAmbient);
glLightfv(GL_LIGHT0, GL_SPECULAR,
m_lightSpecular);
glLightfv(GL_LIGHT0, GL_POSITION,
m_lightPosition);

// Culling
glDisable( GL_CULL_FACE );
// Smooth Shading
glShadeModel(GL_SMOOTH);

m_glSetupDone = true;
```

after this i have some camera set up, but thats completely standard, projection mode, frustum, modelview, look at, translate.

Share Improve this answer

Follow

answered Nov 10, 2008 at 8:54

DavidG

1,797 ● 4 ● 21 ● 33

i'm disabling back face culling here just to test – DavidG Nov 10, 2008 at 9:52

0

Are you sure your normals are **normalized**? If not and you are specifying normals via glNormal calls, you could try to let OpenGL do the normalization for you, keep in mind that this should be avoided, but you can test it out:

```
glEnable(GL_NORMALIZE);
```

This way you are telling OpenGL to rescale all the normal vectors supplied via glNormal.

Share Improve this answer

Follow

answered Jan 5, 2009 at 14:08

Manuel

3,419 ● 1 ● 27 ● 22

0

I had a transparency issue on my terrain display, slopes would seem transparent when looked from a certain angle. It only happened when lighting was enabled in the shader. Turns out that I had not turned on depth testing, and from a certain angle the terrain was overwritten by other terrain and displaying semi-transparent.

TLDR; check if you have depth testing enabled, having it off may give transparency-like effects when lighting is involved. https://learnopengl.com/Advanced-OpenGL/Depth-testing

Share Improve this answer

Follow

answered Nov 19, 2019 at 2:12

Alex

1