CS3451-Fall 2014, P10 REPORT

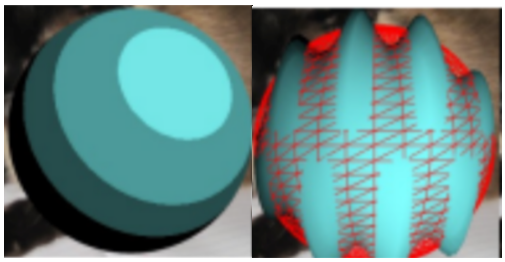
The title: CS3451 Fall 2014, Project 10 Vertex and Fragment shaders

Student: Bo Chen

Project description:

For this project, implementing two pairs of Vertex/Fragment shaders by using processing and openGL. By pressing the key “1” and “2” to switch the two different shader mode I have create in this project.

The first pair of the Vertex/Fragment shader I have combine fragment/vertex option as shown below:



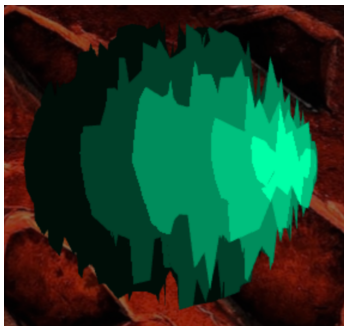
The left shading is a fragment shading, by changing the lighting from diffuse to cel-shading. It is done by thresholding the intensity of the diffuse shading at various values there are the source code in the project to make color reduce to a discrete set:

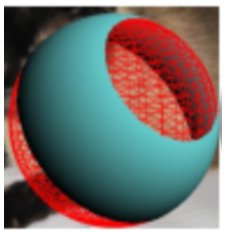
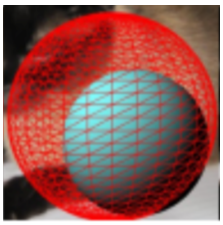
```
if (intensity > 0.95){
    color = vec4(vec3(1.0), 1.0);
    } else if (intensity > 0.85){
        color = vec4(vec3(0.75), 1.0);
    } else if (intensity > 0.55){
        color = vec4(vec3(0.55), 1.0);
    } else if (intensity > 0.25){
        color = vec4(vec3(0.25), 1.0);
    }   else {
        color = vec4(vec3(0.05), 1.0);
    }
```

The right shading effect is by modifying the vertices of the sphere so that they “warble” by moving ‘y’ each vertex with a sin function of the ‘x’ component and time. And I add a small feature to set the movement by using mouse position:

```
vec4 vert = vertex;
vert.y *= sin(vert.x/(mouseY) + time);
// think of gl_Position as a return value for vertex shaders
gl_Position = transform * vert;
vertLightDir = -lightNormal;
vertPos = vert;
```

After this two combine together I got the unnatural and science shading for a sphere.



The second mode is combine with  and 

The left fragment shading is trying to throw away a particular pixel which I don't want it, to make its shape like some kind of contact lenses:

```
if (intensity > 0.90){
    discard;
} else if (intensity == 0.0){
    discard;
} else {
    color = vec4(vec3(intensity), 1.0);
}
```

And the right vertex shading is by changing the input vertex's coordination to move the sphere, also need to make sure the function is relates with mouse position:

```
vec4 vert = vertex;
vert.x += mouseX;
vert.y += mouseY;
// think of gl_Position as a return value for vertex shaders
gl_Position = transform * vert;
vertLightDir = -lightNormal;
vertPos = vert;
```

the final effect should be look like this: