Computer Science Department

Sam Houston State University™

COSC 4332 Computer Graphics

Modern OpenGL
Lightening the truck

Dr. Khaled Rabieh

Outline

1. Lightening

a) Simple Lightening → Light.fragmentshade

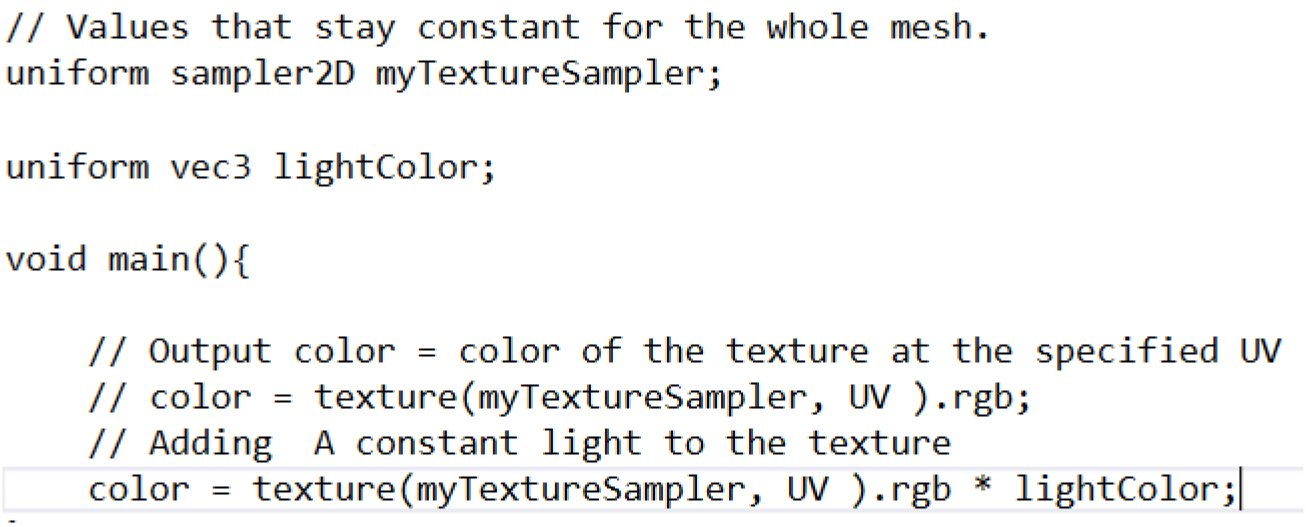
b) Ambient Lightening →Ambient_only.fragmentshader

c) Diffuse Lightening → Diffuse_Ambient.fragmentshader

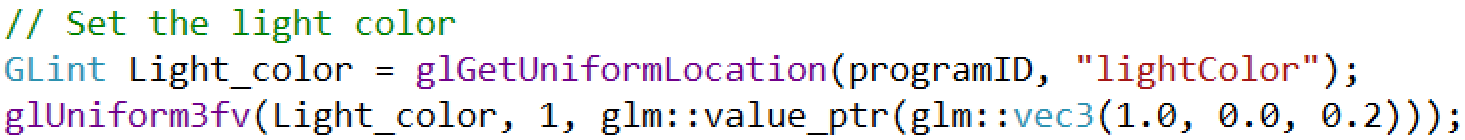
d) Specular lightening →Diffuse_Ambient_Specular.fragmentshader

Simple Lightening

Multiplying the fragments' color by any color can give the feeling that we have a light in the scene

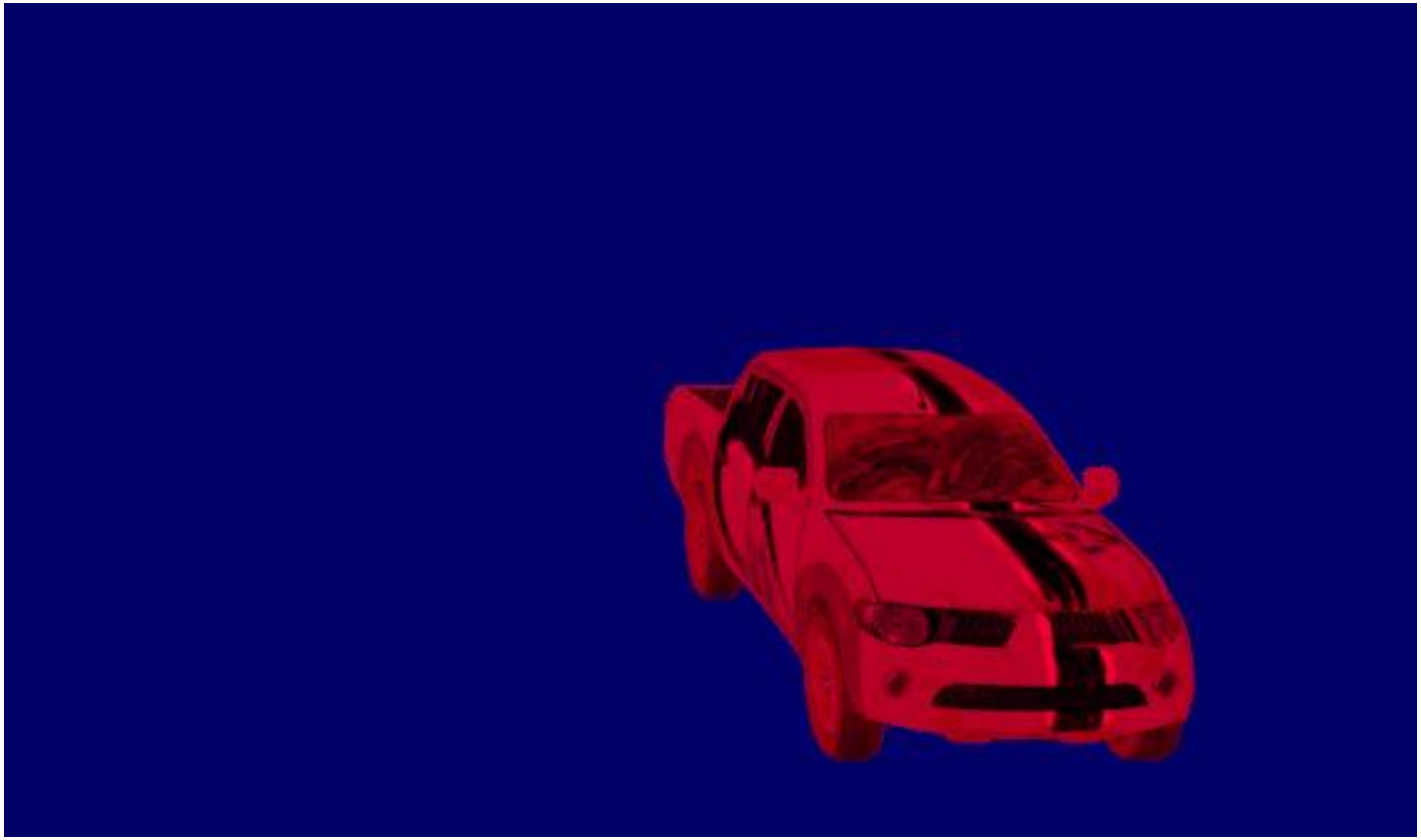


```
// Values that stay constant for the whole mesh.
uniform sampler2D myTextureSampler;

uniform vec3 lightColor;

void main(){

    // Output color = color of the texture at the specified UV
    // color = texture(myTextureSampler, UV ).rgb;
    // Adding  A constant light to the texture
    color = texture(myTextureSampler, UV ).rgb * lightColor;
```



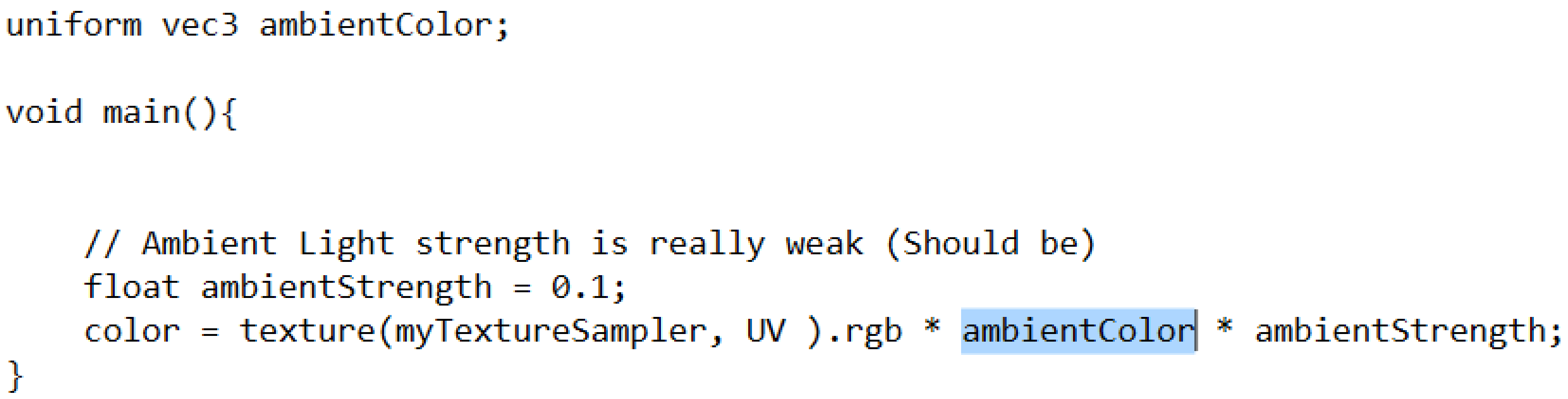
```
// Set the light color
GLint Light_color = glGetUniformLocation(programID, "lightColor");
glUniform3fv(Light_color, 1, glm::value_ptr(glm::vec3(1.0, 0.0, 0.2)));
```

Result



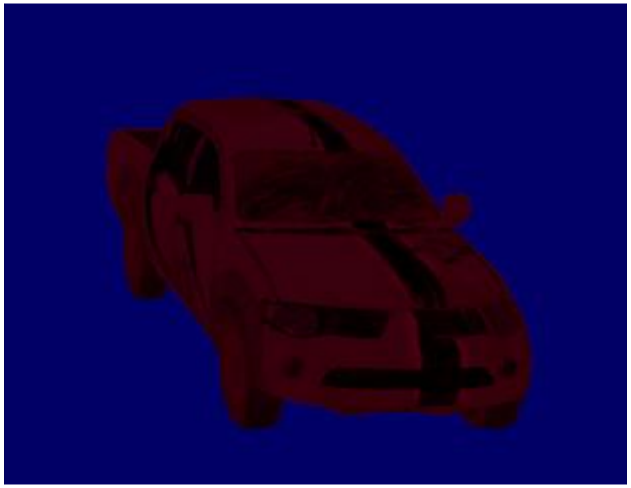
Ambient Color

- Ambient_only.fragmentshader
- Multiply by a small fraction = 0.1

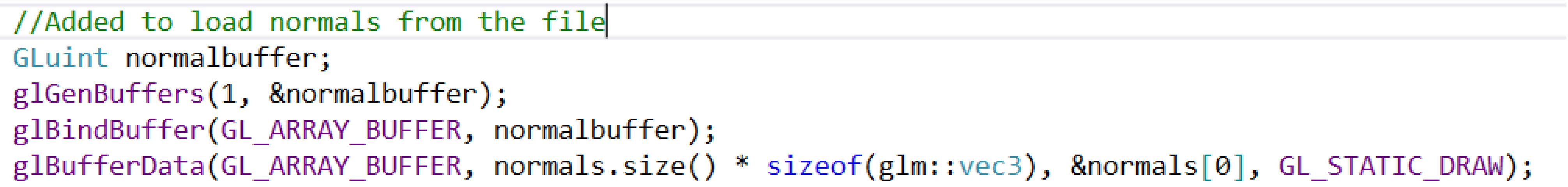


```
uniform vec3 ambientColor;

void main(){


    // Ambient Light strength is really weak (Should be)
    float ambientStrength = 0.1;
    color = texture(myTextureSampler, UV ).rgb * ambientColor * ambientStrength;
}
```

Result

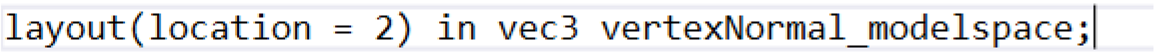


Diffuse + Ambient lightening



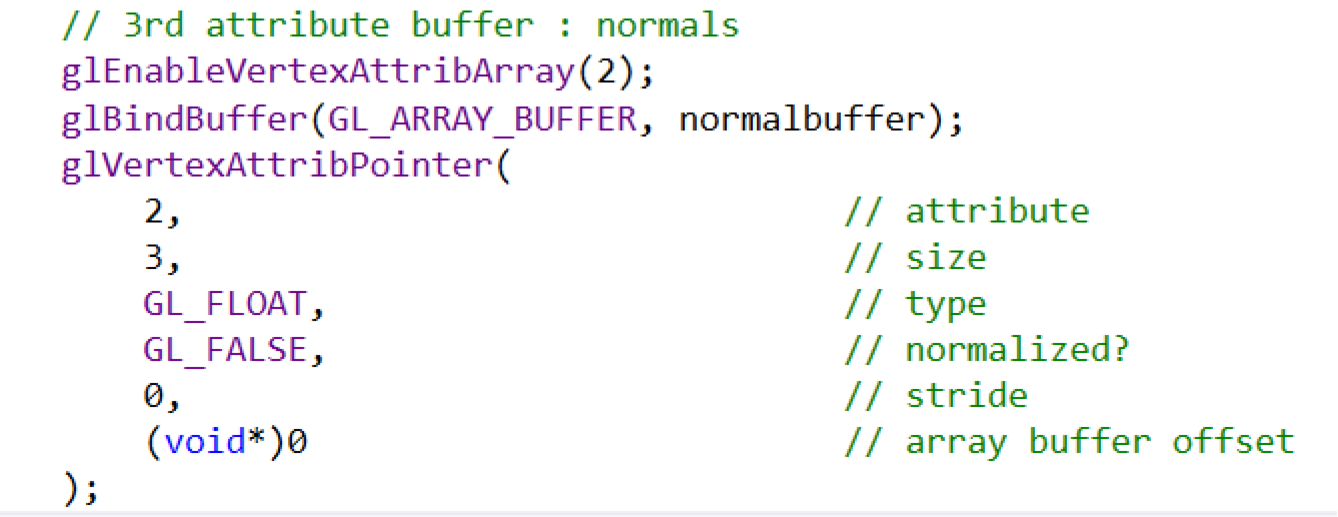
```
//Added to load normals from the file
GLuint normalbuffer;
glGenBuffers(1, &normalbuffer);
glBindBuffer(GL_ARRAY_BUFFER, normalbuffer);
glBufferData(GL_ARRAY_BUFFER, normals.size() * sizeof(glm::vec3), &normals[0], GL_STATIC_DRAW);
```

Vertex Shader



```
layout(location = 2) in vec3 vertexNormal_modelspace;
```

From the program loop

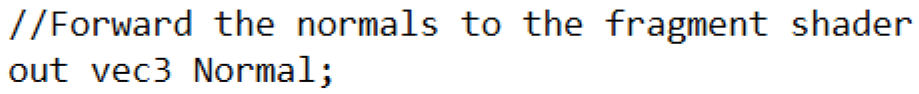


```
// 3rd attribute buffer : normals
glEnableVertexAttribArray(2);
glBindBuffer(GL_ARRAY_BUFFER, normalbuffer);
glVertexAttribPointer(
    2,                                  // attribute
    3,                                  // size
    GL_FLOAT,                           // type
    GL_FALSE,                           // normalized?
    0,                                  // stride
    (void*)0                            // array buffer offset
);
```

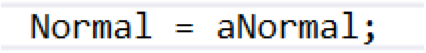
Diffuse + Ambient lightening

- All the lighting calculations are done in the fragment shader
- Forward the normal vectors from the vertex shader to the fragment shader.

vertex shader

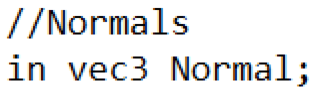


```
//Forward the normals to the fragment shader
out vec3 Normal;
```



```
Normal = aNormal;
```

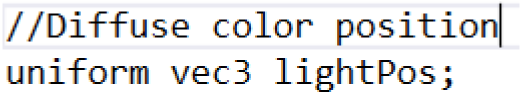
fragment shader



```
//Normals
in vec3 Normal;
```

Diffuse + Ambient lightening

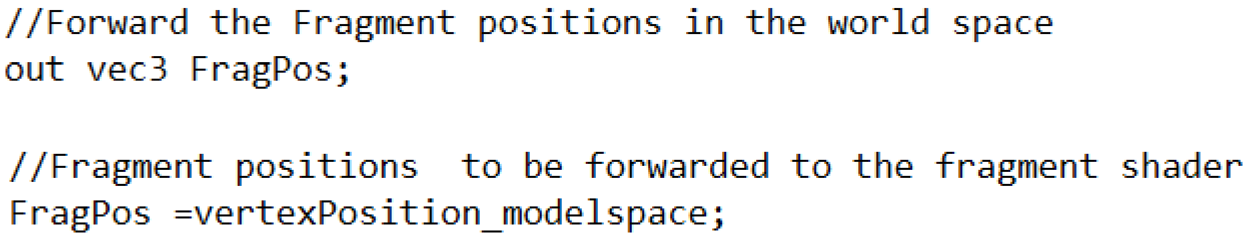
- Since the light's position is just a single static variable we can simply declare it as a uniform in the fragment shader



```
//Diffuse color position
uniform vec3 lightPos;
```

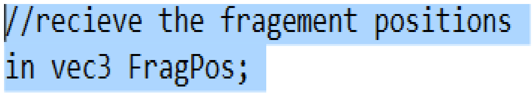
- Calculations should be done in world space
- Forward the vertex positions to the fragment shader

Vertex Shader



```
//Forward the Fragment positions in the world space
out vec3 FragPos;

//Fragment positions  to be forwarded to the fragment shader
FragPos =vertexPosition_modelspace;
```

Fragment Shader

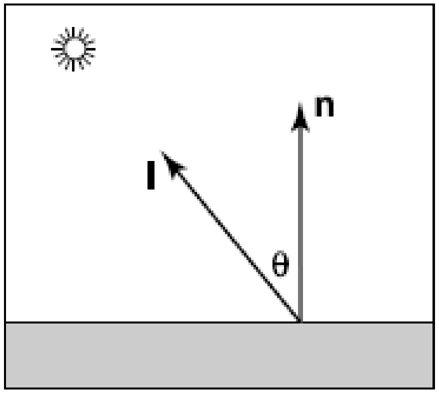


```
//recieve the fragement positions
in vec3 FragPos;
```

Diffuse + Ambient lightening

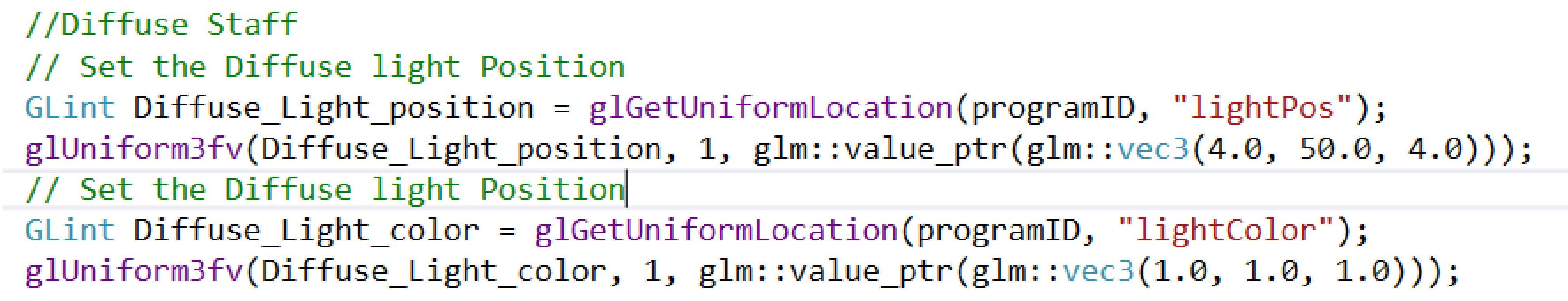
In the fragment shader, Calculate the diffuse color intensity

```
//Ambient color
uniform vec3 ambientColor;

//Diffuse color position
uniform vec3 lightPos;
//Diffuse color
uniform vec3 lightColor;

void main()
{


// Ambient Light strength is really weak (Should be)
    float ambientStrength = 0.2;
    vec3 ambient = ambientStrength * ambientColor;

    // diffuse
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    vec3 result = (ambient + diffuse) * texture(myTextureSampler, UV ).rgb;
    color = result;
}
```



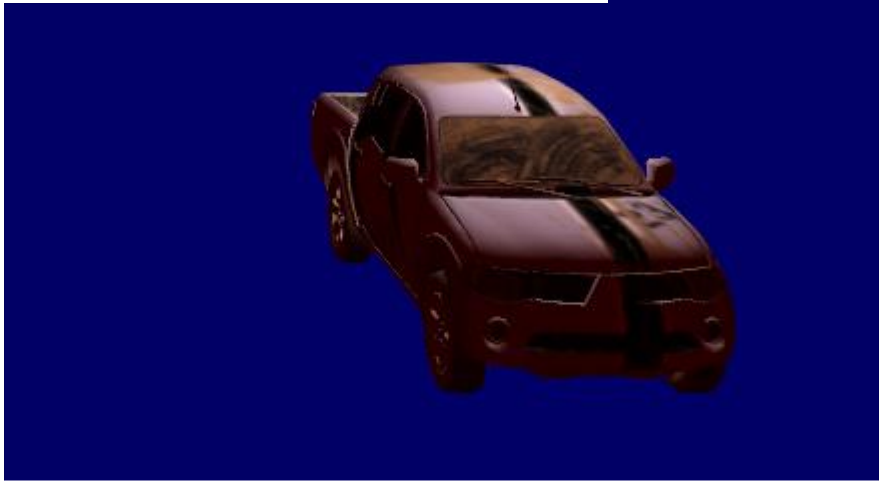
Diffuse + Ambient lightening

- Set the Ambient color to red
- Set the Diffuse color to white
- Set the Diffuse position to +50 above the vehicle



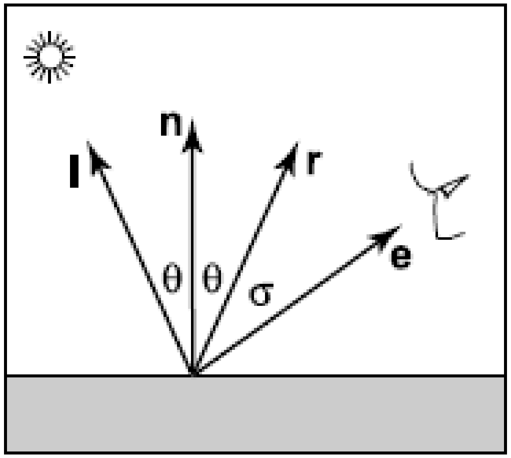
```
//Diffuse Staff
// Set the Diffuse light Position
GLint Diffuse_Light_position = glGetUniformLocation(programID, "lightPos");
glUniform3fv(Diffuse_Light_position, 1, glm::value_ptr(glm::vec3(4.0, 50.0, 4.0)));
// Set the Diffuse light Position
GLint Diffuse_Light_color = glGetUniformLocation(programID, "lightColor");
glUniform3fv(Diffuse_Light_color, 1, glm::value_ptr(glm::vec3(1.0, 1.0, 1.0)));
```

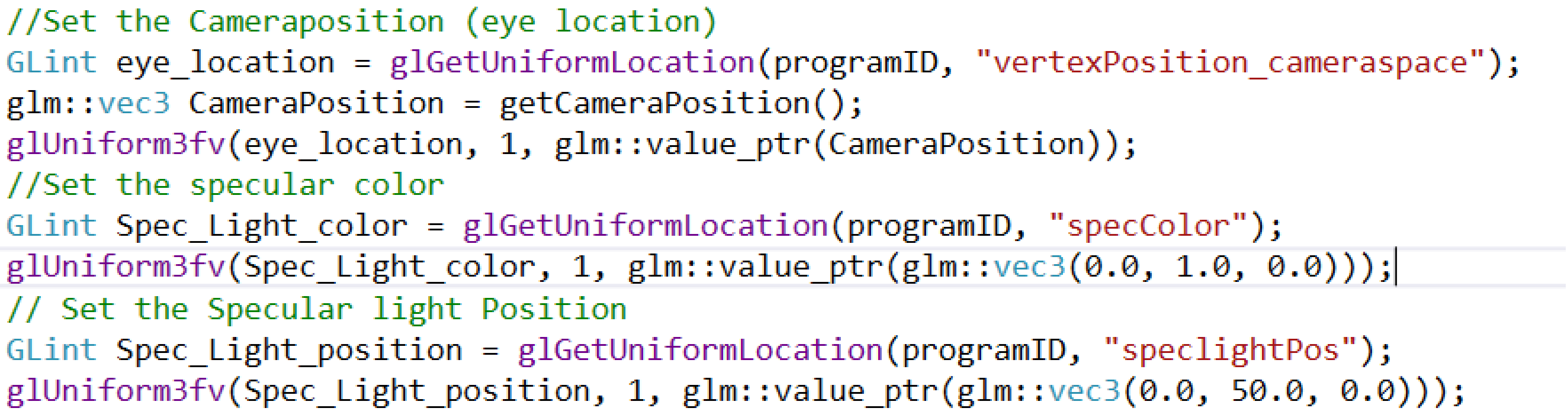
The result



Diffuse + Ambient + Specular lightening

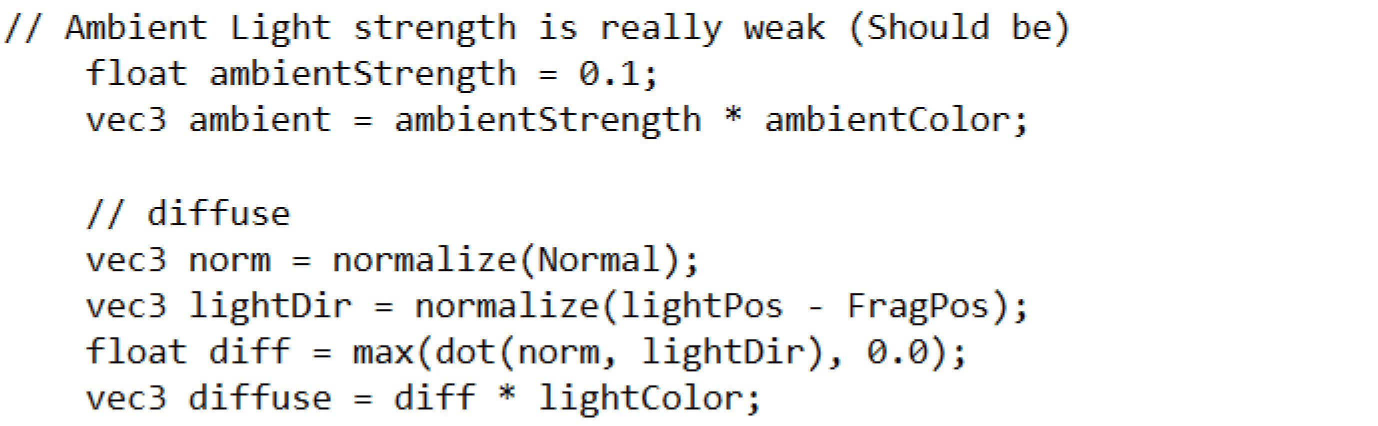
- Specular lightening are highlights
- Move with respect to the eye position
- Our eye position = Camera position



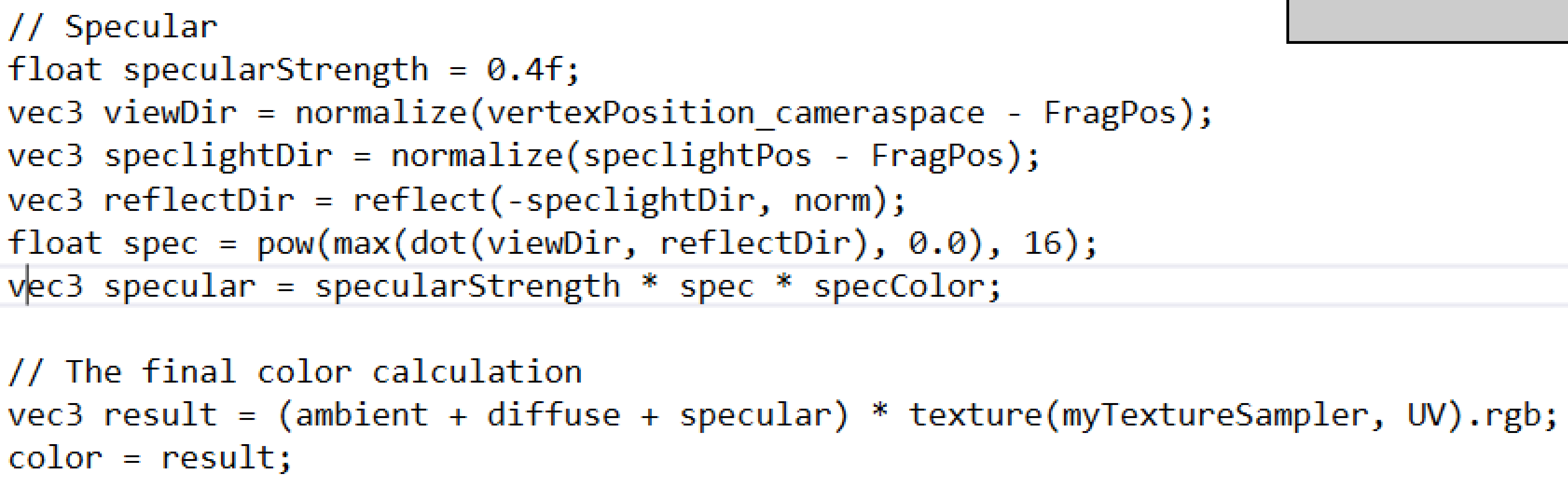


```
//Set the Cameraposition (eye location)
GLint eye_location = glGetUniformLocation(programID, "vertexPosition_cameraspace");
glm::vec3 CameraPosition = getCameraPosition();
glUniform3fv(eye_location, 1, glm::value_ptr(CameraPosition));
//Set the specular color
GLint Spec_Light_color = glGetUniformLocation(programID, "specColor");
glUniform3fv(Spec_Light_color, 1, glm::value_ptr(glm::vec3(0.0, 1.0, 0.0)));
// Set the Specular light Position
GLint Spec_Light_position = glGetUniformLocation(programID, "speclightPos");
glUniform3fv(Spec_Light_position, 1, glm::value_ptr(glm::vec3(0.0, 50.0, 0.0)));
```

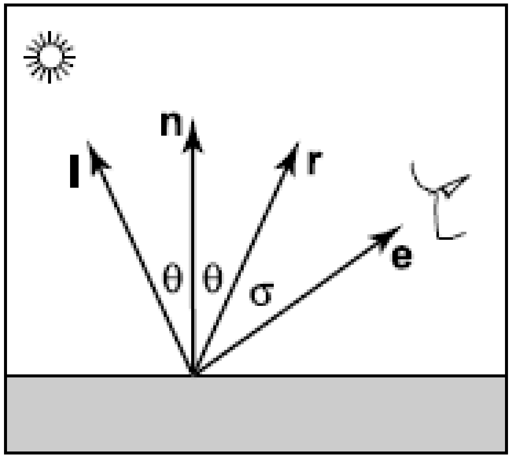
Diffuse + Ambient + Specular lightening



```
// Ambient Light strength is really weak (Should be)
    float ambientStrength = 0.1;
    vec3 ambient = ambientStrength * ambientColor;

    // diffuse
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;
```

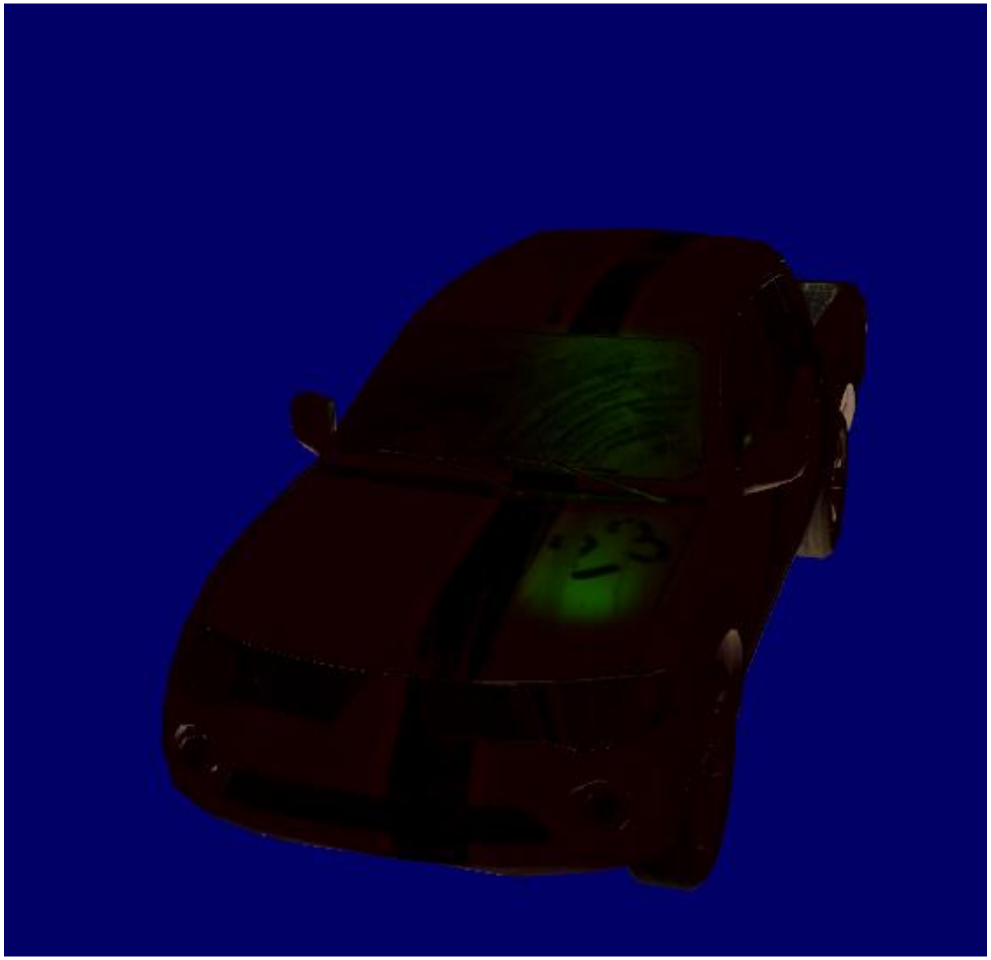


```
    // Specular
    float specularStrength = 0.4f;
    vec3 viewDir = normalize(vertexPosition_cameraspace - FragPos);
    vec3 speclightDir = normalize(speclightPos - FragPos);
    vec3 reflectDir = reflect(-speclightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 16);
    vec3 specular = specularStrength * spec * specColor;

    // The final color calculation
    vec3 result = (ambient + diffuse + specular) * texture(myTextureSampler, UV).rgb;
    color = result;
```



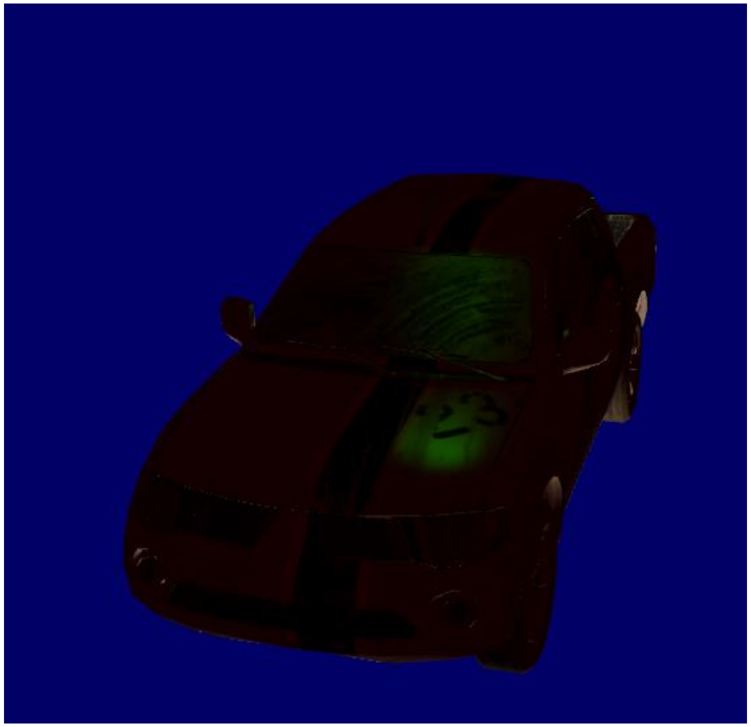
All 3 lights in one

Move around the vehicle
You should see the specular light position moves with you



Your turn

Add two diffuse light sources at the right and left of the truck
Choose any colors of your preference but should be different from other used colors in the scene



Thank You

Questions

Khaled Rabieh