Graphics Assignment 3 Report

Team Members
* Alan Babu-200050005
* Varre Suman Chaitanya-200050153

Light

As input to the fragment shader:

1) The word position is given by only applying the model transformation to the vertex positions.
2) The normal vector of the face is calculated by removing scaling and transformation aspects of the model matrix and then applying it to the model's normals.
3) Texture coordinates

The lighting is calculated by the calculateShading function in the fragment shader

Point Light

The point light structure

```
struct PointLight
{
    int on;
    vec3 position;
    float range;
};
```

Distance and direction between fragment and light is calculated

```
    float dist = length(light.position - fWorldPos);
    vec3 lightDir = (light.position - fWorldPos) / dist;
```

Attenuation with quadratic falloff till range of light is calculated

```
    float attenuation = clamp(1 - (dist * dist)/(light.range *
light.range*10), 0.0 , 1.0);
```

Diffuse component of lighting is calculated as dot product between normal and light dir.

```
    float diff = max(dot(fNormal, lightDir), 0.0);
```

Specular component is calculated using Blinn-Phong calculations

Then diffuse is added with specular component and then it is multiplied with attenuation.

This value is multiplied to color given or color from texture for final output.

Spot Light

```
struct SpotLight
{
    int on;
    vec3 position;
    vec3 direction;
    float innerCutoff;
    float outerCutoff;
    float range;
};
```

The dot product between the direction from light to fragment and actual direction of spotlight is taken

```
    float theta = dot(lightDir, -light.direction);
```

If this value is less than outerCutoff then light will not contribute to shading.

Otherwise the light intensity increases from 0 to 1 till this product is equal to innerCutoff, after which is maximum intensity of spotlight.

The result of lighting calculations similar to pointlight is then multiplied with this intensity to get the final lighting.

For the spotlight on rider attenuation is not used.
This is then multiplied with the color to get final output.

Headlight

For the headlight its parent is set as the handle of the bike. So its position and direction will be affected by the transformations applicable to the handle bar.

```
glm::vec3 HeadLight::getPosition()
{
    auto transMat = glm::mat4(1.0f);
    if (m_parent != NULL)
        transMat = m_parent->getTransformationMatrix();

    return glm::vec3(transMat* glm::vec4(position, 1));
}
glm::vec3 HeadLight::getDirection()
{
    auto trans = glm::mat4(1.0f);

    trans = glm::rotate(trans, glm::radians(rotation.x), glm::vec3(1.0f, 0.0f, 0.0f));
    trans = glm::rotate(trans, glm::radians(rotation.y), glm::vec3(0.0f, 1.0f, 0.0f));
    trans = glm::rotate(trans, glm::radians(rotation.z), glm::vec3(0.0f, 0.0f, 1.0f));

    glm::vec3 direction = glm::vec3(trans * glm::vec4(0, -1, 0, 1));

    auto transMat = glm::mat4(1.0f);
    if (m_parent != NULL)
        transMat = m_parent->getTransformationMatrix();

    return glm::normalize(glm::vec3(transMat * glm::vec4(direction, 0)));
}
```

Spotlight on rider

For the spotlight its target is set as the root node of the rider. So its direction will be given by the relative position of the rider..

```
glm::vec3 FollowLight::getDirection()
{
    auto target_pos = glm::vec3(0,0,0);
    if(m_target!=NULL)
        target_pos = glm::vec3(m_target->getTransformationMatrix() * glm::vec4(0, 0, 0, 1));
```

```
    return glm::normalize(target_pos-position);
}
```

Texturing

The texture coordinates for track, and cube were calculated manually and input to the vertex shader as an attribute.
The texture image were loaded using stb image loader(https://github.com/nothings/stb)

In the scene constructor the textures for track, rider torso, bike seat, skybox are bound to the first 4 texture slots.

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, m_trackTexture.getTextureID());
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_CUBE_MAP, m_skyboxTexture.getTextureID());
glActiveTexture(GL_TEXTURE2);
glBindTexture(GL_TEXTURE_2D, m_riderTexture.getTextureID());
glActiveTexture(GL_TEXTURE3);
glBindTexture(GL_TEXTURE_2D, m_bikeTexture.getTextureID());
```

These slot numbers are input to shaders as Sampler2D values to fragment shader during rendering.

The color is calculated using this texture and texture coordinates.

```
frag_color = texture(uColTex, fTexcoord)* lighting;
```

Script

The rider does two laps through the track.

Link to video

https://youtu.be/YqYhguLE-cY