UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

REPORTE DE PRÁCTICA Nº 08

NOMBRE COMPLETO: Casillo Martinez Diego Leonardo

N.º de Cuenta: 319041538

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 09/10/2024

CALIFICACIÓN: _____

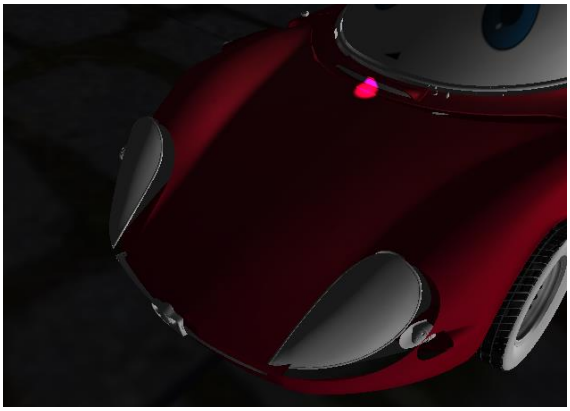
REPORTE DE PRÁCTICA:

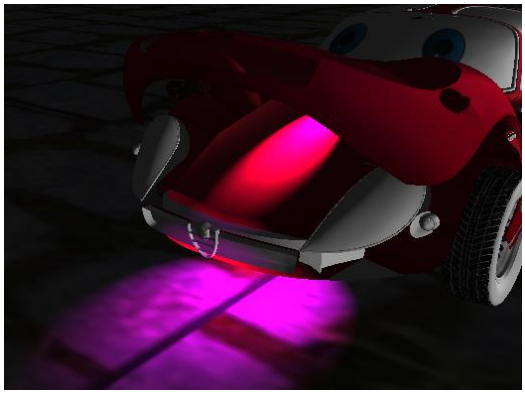
1.- Desarrollo

Ejercicio 1: este ejercicio Plantea el siguiente problema:

Agregar un spolight (que no sea luz de color blanco ni azul) que parta del cofre de su coche y al abrir y cerrar el cofre ilumine en esa dirección.

Para este ejercicio se heredó el ángulo del cofre por medio de model a este se le ligo la posición y rotación del cofre en la parte de inicio del pivote de nuestro coche de la siguiente forma:





Este ejercicio en general fue muy parecido a la practica 7 por lo que solo fue cuestión de entender que se tenia que heredar el modelo de el cofre para poder obtener el resultado deseado

Ejercicio 2: este ejercicio Plantea el siguiente problema:

Agregar luz de tipo spotlight para el coche de tal forma que al avanzar (mover con teclado hacia dirección de X negativa) ilumine con un spotlight hacia adelante y al retroceder ((mover con teclado hacia dirección de X positiva) ilumine con un spotlight hacia atrás. Son dos spotlights diferentes que se prenderán y apagarán

Para este ejercicio se definieron 2 nuevas estructuras para contener 1 luz, cada estructura representa las luces, trasera y delantera, una es blanca y la otra verde, siendo esto un trabajo similar al previo de la práctica, algo a considerar es que como el tamaño de ambas estructuras será el mismo compartirán el mismo contador:

```
// esta es la luz delantera
spotLights1[0] = SpotLight(0.0f, 1.0f, 0.0f,  // Color Azul
    2.5f, 5.0f,         // componentes ambiental y difusa
    0.0f, 0.0f, 0.0f,  // Posición inicial
    0.0f, -1.0f, 0.0f,  // Dirección hacia donde apunta la luz
    1.0f, 0.03f, 0.05f,  // atenuaciones
    25.0f);            // Ángulo de apertura
// esta es la luz trasera
spotLights2[0] = SpotLight(1.0f, 1.0f, 1.0f,  // Color Azul
    2.5f, 5.0f,         // componentes ambiental y difusa
    0.0f, 0.0f, 0.0f,  // Posición inicial
    0.0f, -1.0f, 0.0f,  // Dirección hacia donde apunta la luz
    1.0f, 0.03f, 0.05f,  // atenuaciones
    25.0f);            // Ángulo de apertura
spotLightCountLuzCoche++;
```

Luego de esto, se definieron dos banderas para controlar el cambio de cada luz y simular que se apagan y se encienden. Estas se redefinían cada vez que se aplicaba el botón de desplazamiento a la derecha o a la izquierda. Lo que hacíamos era heredar el desplazamiento para que la luz siguiera al auto. Con todo esto planeado, se tiene el siguiente código:

```
LuzDelantera = false;
LuzTrasera = false;
```

Se definen primero ambas como falsas pues ninguna de estas se prendera a menos que el coche se desplace.

Luego al detectar los botones:

```
if (key == GLFW_KEY_F)
{
    theWindow->LuzDelantera = false;
    theWindow->LuzTrasera = false;
    if (theWindow->articulacion1 > 30) {

    }
    else {
        theWindow->articulacion1 += 10.0;
    }
}


if (key == GLFW_KEY_G)
{
    theWindow->LuzDelantera = false;
    theWindow->LuzTrasera = false;
    if (theWindow->articulacion1 == 0) {

    }
    else {
        theWindow->articulacion1 -= 10.0;
    }
}
```

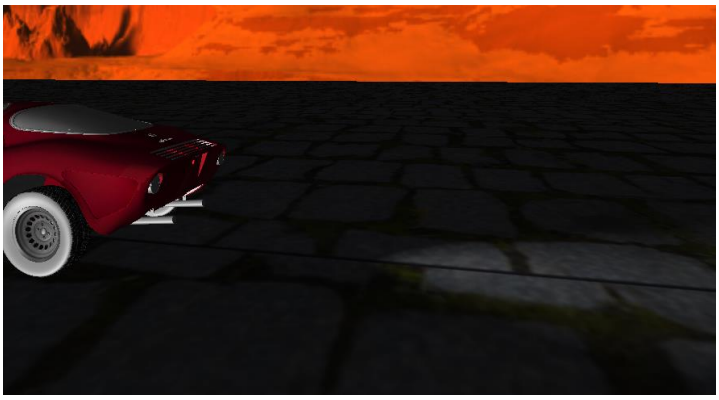
Se designa el control para poder definir que luz se va a prender y cual se va a apagar.

Y al finalizar en el main se aplica el control de la siguiente forma:

```
if (mainWindow.getLuzDelantera() == true) {


    glm::vec3 cofrePosition = glm::vec3(glm::vec3(model[3][0] - 9.0f, model[3][1] - 1.4f, model[3][2]));
    // 3. Actualizar la posición y dirección del `SpotLight`
    spotLights1[0].SetFlash(cofrePosition, glm::vec3(-1.0f, 0.0f, 0.0f));
    shaderList[0].SetSpotLights(spotLights1, spotLightCountLuzCoche);


}
else if (mainWindow.getLuzTrasera() == true) {
    glm::vec3 cofrePosition = glm::vec3(glm::vec3(model[3][0] + 9.5f, model[3][1] - .7, model[3][2]));
    spotLights2[0].SetFlash(cofrePosition, glm::vec3(1.0f, 0.0f, 0.0f));
    shaderList[0].SetSpotLights(spotLights2, spotLightCountLuzCoche);

}
else
{
    shaderList[0].SetSpotLights(spotLights, spotLightCount);
}
```

En donde se define el cambio de luz según las condiciones, si se desplaza se cambia al juego de luces traseras y delanteras si se abre el cofre se designa el juego de luz para el cofre.

Gracias a esto obtenemos el siguiente resultado:



Ejercicio 3: este ejercicio Plantea el siguiente problema:

Agregar otra luz de tipo puntual ligada a un modelo elegido por ustedes (no lámpara) y que puedan prender y apagar de forma independiente con teclado tanto la luz de la lámpara como la luz de este modelo (la luz de la lámpara debe de ser puntual, si la crearon spotlight en su reporte 7 tienen que cambiarla a luz puntual)

Para este ejercicio se declararon dos banderas, cada una indicando el valor de cada **point light**. A su vez, se definieron tres series de **point lights**: una controla la luz de un faro, otra controla la luz de una fogata y la última controla ambas. Esto se hace de la siguiente forma:

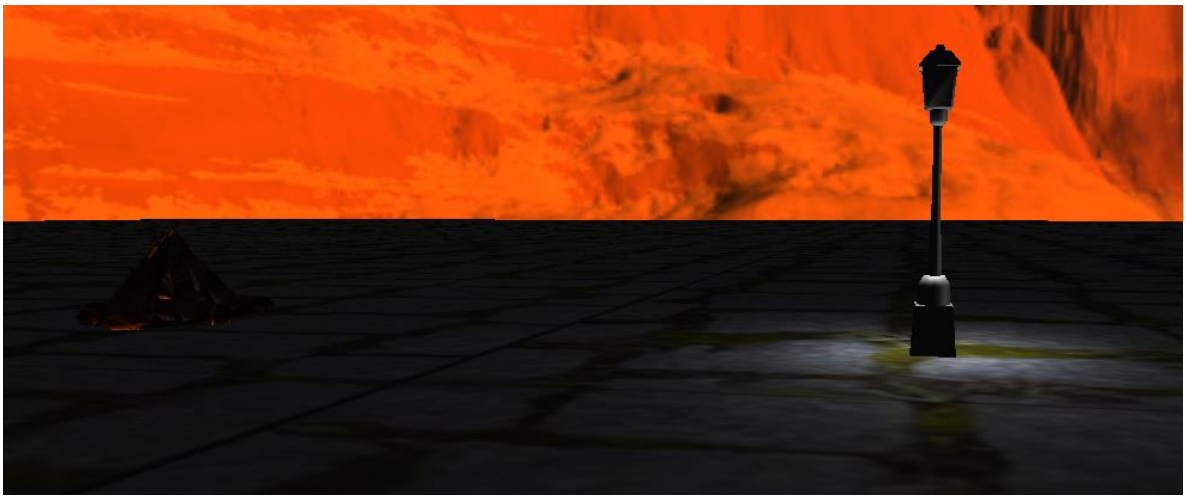
```
pointLights[0] = PointLight(1.0f, 1.0f, 1.0f,
    0.0f, 1.0f,
    6.0f, 1.5f, 12.0f,
    0.1f, 0.07f, 0.05f);

pointLights1[0] = PointLight(1.0f, 0.0f, 0.0f,
    0.0f, 1.0f,
    6.0f, .5f, -12.0f,
    0.1f, 0.07f, 0.05f);
pointLightCount++;

unsigned int pointLightCount1 = 0;
pointLights2[0] = PointLight(1.0f, 1.0f, 1.0f,
    0.0f, 1.0f,
    6.0f, 1.5f, 12.0f,
    0.1f, 0.07f, 0.05f);
pointLightCount1++;
pointLights2[1] = PointLight(1.0f, 0.0f, 0.0f,
    0.0f, 1.0f,
    6.0f, .5f, -12.0f,
    0.1f, 0.07f, 0.05f);
pointLightCount1++;
```

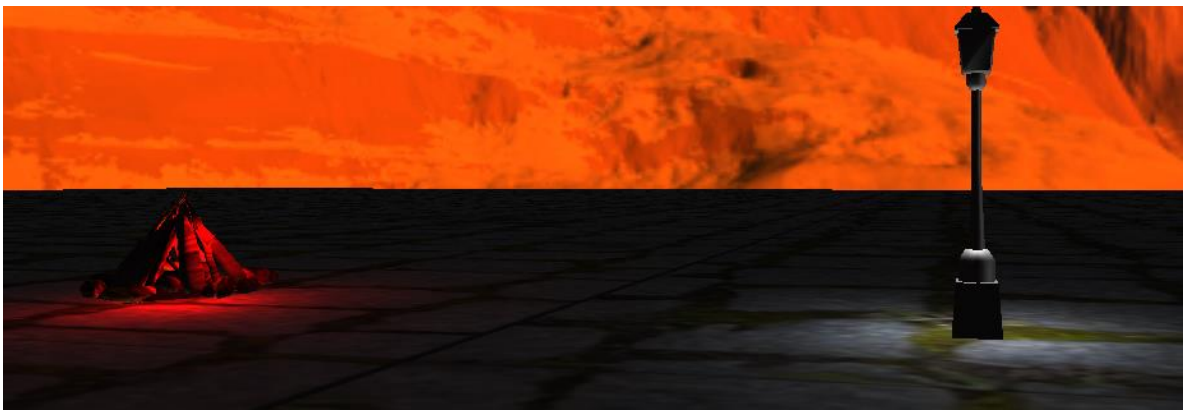
Después de esto se definió que las dos banderas iniciaran en false, y según se presione k o l se prendan o se apaguen

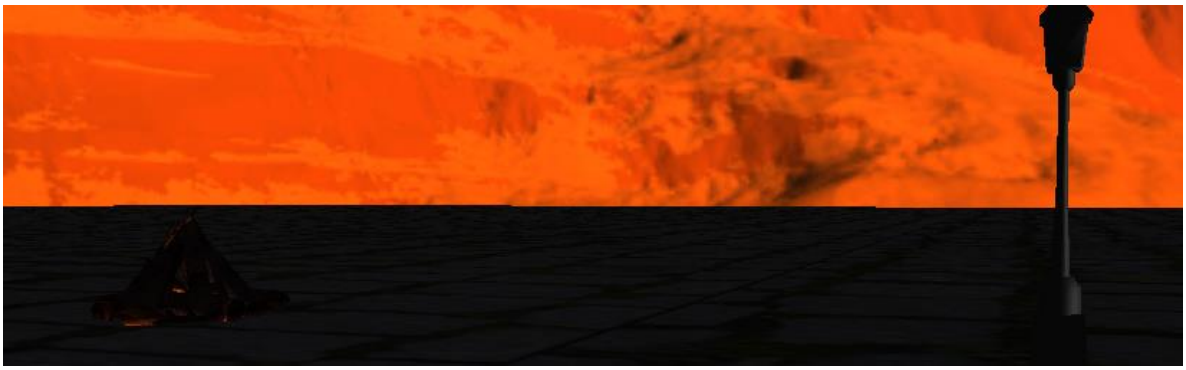
Por ultimo se genero un control en el main para poder definir cuando se apagan ambas cuando solo 1 y cuando se prenden ambas de la siguiente forma:

```
if (key == GLFW_KEY_K && action == GLFW_RELEASE )
{
    theWindow->Bandera1 = !theWindow->Bandera1;
    printf("Faro %s \n", theWindow->Bandera1 ? "Prendido" : "Apagado");
}
if (key == GLFW_KEY_L && action == GLFW_RELEASE )
{
    theWindow->Bandera2 = !theWindow->Bandera2;
    printf("Faro %s \n", theWindow->Bandera1 ? "Prendido" : "Apagado");
}
```

logrando el siguiente resultado:







2. Problemas que surgieron:

Durante esta práctica, o surgió ningún problema pues solo se aplicaron conceptos ya realizados en otras practicas anteriores

3. Conclusión:

En general, la práctica fue entretenida y, gracias a esta y a la clase teórica, me quedó mucho más claro cómo funcionan las luces. Es cuestión de práctica para entender qué representa cada posición en el arreglo de cada luz; sin embargo, con una pequeña guía, este ejercicio fue entretenido y desafiante.

4.- Bibliografía:

LearnOpenGL - Basic Lighting. (s. f.). https://learnopengl.com/Lighting/Basic-Lighting