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 07

NOMBRE COMPLETO: Miranda González José Francisco

Nº de Cuenta: 318222327

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 05/10/24

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

Reporte de Práctica 7: Iluminación 1

Instrucciones:

1.-Agregar movimiento con teclado al helicóptero hacia adelante y atrás.

2.-crear luz spotlight de helicóptero de color amarilla que apunte hacia el piso y se mueva con el helicóptero

3.- Añadir en el escenario 1 modelo de lámpara texturizada (diferente a los que usarán en su proyecto final) y crearle luz puntual blanca

Actividades realizadas

Para completar las actividades solicitadas realice lo siguiente:

En las carpetas Models y Textures coloque todo lo necesario para poder realizar la práctica sin probemas.

Lo único nuevo fue el modelo y textura de la lampara que implemente.

Dentro de main.cpp los cambios fueron los siguientes:

Cree las texturas y modelos necesarios.

```
51      // TEXTURAS DE LA PRACTICA 06
52      ////////////////////////////////
53      Texture rojo;
54      Texture negro;
55      Texture cars;
56      ////////////////////////////////
57
58      // EJERCICIO 03
59      // TEXTURAS DE LA PRACTICA 07
60      // LAMPARA
61      ////////////////////////////////
62      Texture superior;
63      Texture inferior;
64      ////////////////////////////////
65
66      Model Kitt_M;
67      Model Llanta_M;
68      Model Blackhawk_M;
69
70      // MODELOS DE LA PRACTICA 06
71      ////////////////////////////////
72      Model CocheCars;
73      Model LlantaTraseraIzquierda;
74      Model LlantaTraseraDerecha;
75      Model LlantaDelanteraIzquierda;
76      Model LlantaDelanteraDerecha;
77      ////////////////////////////////
78
79      // EJERCICIO 03
80      // MODELOS DE LA PRACTICA 07
81      // LAMPARA
82      ////////////////////////////////
83      Model LamparaSimpSons;
84      ////////////////////////////////
```

Cargue las texturas y modelos.

```
244     // TEXTURAS DE LA PRACTICA 06
245     ////////////////////////////////////////////////////////////////////////////
246     cars = Texture("Textures/Imagen-318222327GIMP.png");
247     cars.LoadTextureA();
248     negro = Texture("Textures/ColorNegroGIMP.png");
249     negro.LoadTextureA();
250     rojo = Texture("Textures/ColorRojoGIMP.png");
251     rojo.LoadTextureA();
252     ////////////////////////////////////////////////////////////////////////////
```

```
254     // EJERCICIO 03
255     // TEXTURAS DE LA PRACTICA 07
256     // LAMPARA
257     ////////////////////////////////////////////////
258     superior = Texture("Textures/lamp_shade.bmp.png");
259     inferior = Texture("Textures/char_swatches.bmp.png");
260     ////////////////////////////////////////////////
261
262     Kitt_M = Model();
263     Kitt_M.LoadModel("Models/kitt_optimizado.obj");
264     Llanta_M = Model();
265     Llanta_M.LoadModel("Models/llanta_optimizada.obj");
266     Blackhawk_M = Model();
267     Blackhawk_M.LoadModel("Models/uh60.obj");
268
269     // MODELOS DE LA PRACTICA 06
270     ////////////////////////////////////////////////////////////////
271     CocheCars = Model();
272     CocheCars.LoadModel("Models/cochePractica06.obj");
273     LlantaTraseraIzquierda = Model();
274     LlantaTraseraIzquierda.LoadModel("Models/llantaTraseraIzquierdaPractica06.obj");
275     LlantaTraseraDerecha = Model();
276     LlantaTraseraDerecha.LoadModel("Models/llantaTraseraDerechaPractica06.obj");
277     LlantaDelanteraIzquierda = Model();
278     LlantaDelanteraIzquierda.LoadModel("Models/llantaDelanteraIzquierdaPractica06.obj");
279     LlantaDelanteraDerecha = Model();
280     LlantaDelanteraDerecha.LoadModel("Models/llantaDelanteraDerechaPractica06.obj");
281     ////////////////////////////////////////////////////////////////
282
283     // EJERCICIO 03
284     // MODELOS DE LA PRACTICA 07
285     // LAMPARA
286     ////////////////////////////////////////////////
287     LamparaSimpSons = Model();
288     LamparaSimpSons.LoadModel("Models/LamparaSimpsons.obj");
289     ////////////////////////////////////////////////
```

Pointlight blanca.

```
328     // EJERCICIO 03
329     // PARA LA PRACTICA 07
330     // LUZ
331     ////////////////////////////////
332     pointLights[1] = PointLight(
333         1.0f, 1.0f, 1.0f,
334         0.8f, 1.0f,
335         -30.0f, 6.0f, -20.0f,
336         0.3f, 0.2f, 0.1f);
337
338     pointLightCount++;
339     ////////////////////////////////
```

En este caso, el archivo PointLight.cpp no contenía una función similar a SetFlash o SetPos de SpotLight.cpp.

Entonces lo que hice fue ajustar la posición de la luz al crearla (para que estuviera en la ubicación del modelo de la lampara), pero este caso no hay una jerarquía entre el modelo y la luz

Spotlight azul y amarilla.

```
// PARA EL EJERCICIO DE CLASE 07
////////////////////////////////////////////////////////////////////////////////////////////////
//Faro frontal de color azul
spotLights[2] = SpotLight(
    0.0f, 0.0f, 1.0f,
    4.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.015f, 0.0015f,
    10.0f);

spotLightCount++;// SE AUMENTA EL CONTADOR PARA NO SOBREESCRIBIR LA LUZ
////////////////////////////////////////////////////////////////////////////////////////////////

// EJERCICIO 02
// PARA LA PRACTICA 07
// SPOTLIGHT AMARILLA
////////////////////////////////////////////////////////////////////////////////////////////////
spotLights[3] = SpotLight(
    1.0f, 1.0f, 0.0f,
    3.0f, 1.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    15.0f);

spotLightCount++;// SE AUMENTA EL CONTADOR PARA NO SOBREESCRIBIR LA LUZ
////////////////////////////////////////////////////////////////////////////////////////////////
```

La ubicación y dirección de las luces están en cero porque en la función SetFlash se los añadiremos.

Matrices auxiliares para la jerarquía.

```
// PARA LA JERARQUIA
//////////////////////////
glm::mat4 modelaux(1.0);
glm::mat4 modelauxH(1.0);
//////////////////////////
```

Coche.

```
// MODIFICADO A LO QUE SE REALIZO EN LA PRACTICA 06
////////////////////////////////////////////////////////////////////////////////
//Instancia del coche
model = glm::mat4(1.0);

model = glm::translate(model, glm::vec3(0.0f, 5.4f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex(), 0.0f,0.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaeny()), glm::vec3(0.0f, 1.0f, 0.0f)); // SOLO COMO EJEMPLO (ROTAR CON P)
//EN VEZ DE PONERLE ROTACION AL COCHE Y CADA LLANTA POR SEPARADO, SE HEREDA DESDE EL COCHE

modelaux = model; // GUARDAR

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
/*color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
CocheCars.RenderModel();

model = modelaux; // A PARTIR DEL COCHE

// PARA LA LUZ AZUL CON JERARQUIA

glm::vec3 posicion = glm::vec3(model[3]); //+ glm::vec3(-12.0f, -1.0f, 0.0f); // AJUSTAR LA POSICION A PARTIR DEL COCHE
glm::vec3 direccion = glm::vec3(model[2]);

spotLights[2].SetFlash(posicion, direccion);

model = modelaux; // A PARTIR DEL COCHE

// LLANTA TRASERA IZQUIERDA

model = glm::translate(model, glm::vec3(6.28f, -3.5f, -13.2f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
/*color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
LlantaTraseraIzquierda.RenderModel();
```

```
506         model = modelaux; // A PARTIR DEL COCHE
507
508         // LLANTA TRASERA DERECHA
509
510         model = glm::translate(model, glm::vec3(-6.5f, -3.5f, -13.2f));
511         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
512         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
513         /*color = glm::vec3(0.0f, 1.0f, 1.0f);
514         glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
515         LlantaTraseraDerecha.RenderModel();
516
517         model = modelaux; // A PARTIR DEL COCHE
518
519         // LLANTA DELANTERA IZQUIERDA
520
521         model = glm::translate(model, glm::vec3(6.2f, -3.5f, 10.7f));
522         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
523         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
524         /*color = glm::vec3(1.0f, 0.0f, 1.0f);
525         glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
526         LlantaDelanteraIzquierda.RenderModel();
527
528         model = modelaux; // A PARTIR DEL COCHE
529
530         // LLANTA DELANTERA DERECHA
531
532         model = glm::translate(model, glm::vec3(-6.5f, -3.5f, 10.7f));
533         model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
534         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
535         /*color = glm::vec3(0.42f, 0.0f, 0.18f);
536         glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
537         LlantaDelanteraDerecha.RenderModel();
538         ////////////////////////////////////////////////////////////////////////////////
```

En esta parte de la práctica, para poder aplicar la jerarquía a la Spotlight realice lo siguiente:

Necesite pasar de mat 4 a un vec3.

```
//Instancia del coche
model = glm::mat4(1.0);

model = glm::translate(model, glm::vec3(0.0f, 5.4f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex(), 0.0f,0.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaeny()), glm::vec3(0.0f, 1.0f, 0.0f)); // SOLO COMO EJEMPLO (ROTAR CON P)
//EN VEZ DE PONERLE ROTACION AL COCHE Y CADA LLANTA POR SEPARADO, SE HEREDA DESDE EL COCHE

modelaux = model; // GUARDAR

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
/*color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
CocheCars.RenderModel();

model = modelaux; // A PARTIR DEL COCHE
```

En model = modelaux comienzo a partir del coche.

Por lo tanto, de model obtuve la posición y dirección para la luz.

```
// PARA LA LUZ AZUL CON JERARQUIA

glm::vec3 posicion = glm::vec3(model[3]); //+ glm::vec3(-12.0f, -1.0f, 0.0f); // AJUSTAR LA POSICION A PARTIR DEL COCHE
glm::vec3 direccion = glm::vec3(model[2]);
```

Estos valores que ya contienen la información del coche se los paso a la función SetFlash, de esta forma la jerarquía se mantiene.

```
spotLights[2].SetFlash(posicion, direccion);
```

Para el helicóptero, la idea fue la misma.

```
// EJERCICIO 01
// PARA LA PRACTICA 07
// HELICOPTERO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0);

model = glm::translate(model, glm::vec3(-20.0f, 20.0f, 20.0));
model = glm::translate(model, glm::vec3(mainWindow.getmueveHelicoptero(), 0.0f, 0.0f)); // MOVER EL HELICOPTERO EN EL EJE X (CON I y O)
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaenz()), glm::vec3(1.0f, 0.0f, 0.0f)); // SOLO COMO EJEMPLO (ROTAR CON Z)

modelauxH = model;

model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Blackhawk_M.RenderModel();

model = modelauxH; // A PARTIR DEL HELICOPTERO

// EJERCICIO 02
// PARA LA PRACTICA 07
// SPOTLIGHT AMARILLA

// PARA LA LUZ AZUL CON JERARQUIA

glm::vec3 posicionH = glm::vec3(model[3]);
glm::vec3 direccionH = glm::vec3(model[2]);

direccionH = -direccionH; // PORQUE APUNTA A Z INICIALMENTE EN EL MODELO ORIGINAL Y QUIERO QUE APUNTE A - Z EN EL MODELO ORIGINAL
// AL HACER LAS ROTACIONES ES COMO SI APUNTARA HACIA -Y

spotLights[3].SetFlash(posicionH, direccionH);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Se puede observar que se agregó una rotación por teclado tanto al coche como al helicóptero.

Esto no se solicitaba en la práctica, pero lo hice para comprobar que al rotar los modelos la luz lo hiciera con ellos.

Esto lo explicare mas detallado en problemas presentados.

Al final solo coloque la lampara.

```
// EJERCICIO 03
// PARA LA PRACTICA 07
// LAMPARA
////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0);

model = glm::translate(model, glm::vec3(-30.0f, 2.55f, -20.0f));
model = glm::scale(model, glm::vec3(10.0f, 10.0f, 10.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LamparaSimpSons.RenderModel();
////////////////////////////////////////////////////////////////////////////////
```

En window.cpp los cambios fueron los siguientes:

Para la traslación y rotación (solo como ejemplo).

```
// PARA LA PRACTICA 07
// HELICOPTERO
//////////////////////////
mueveHelicoptero = 0.0f;
//////////////////////////

// PARA COMPROBAR QUE LA JERARQUIA CON LA LUZ SE HIZO DE MANERA CORRECTA (SOLO COMO EJEMPLO)
//////////////////
rotaeny = 0.0f;
rotaenz = 0.0f;
//////////////////
```

```
    // PARA LA PRACTICA 07
    // HELICOPTERO
    ////////////////////////////////////////////////////////
    if (key == GLFW_KEY_I)
    {
        theWindow->mueveHelicoptero += 1.0;
    }
    if (key == GLFW_KEY_O)
    {
        theWindow->mueveHelicoptero -= 1.0;
    }
    ////////////////////////////////////////////////////////

    // PARA COMPROBAR QUE LA JERARQUIA CON LA LUZ SE HIZO DE MANERA CORRECTA (SOLO COMO EJEMPLO)
    ////////////////////////////////////////////
    if (key == GLFW_KEY_P)
    {
        theWindow->rotaeny -= 1.0;
    }
    if (key == GLFW_KEY_Z)
    {
        theWindow->rotaenz -= 1.0;
    }
    ////////////////////////////////////////////
```

En window.h los cambios fueron los siguientes:

```
    // PARA LA PRACTICA 07
    // HELICOPTERO
    ////////////////////////////////////////////////////////
    GLfloat getmueveHelicoptero() { return mueveHelicoptero; }
    ////////////////////////////////////////////////////////

    // PARA COMPROBAR QUE LA JERARQUIA CON LA LUZ SE HIZO DE MANERA CORRECTA (SOLO COMO EJEMPLO)
    ////////////////////////////////////////////
    GLfloat getrotaeny() { return rotaeny; }
    GLfloat getrotaenz() { return rotaenz; }
    ////////////////////////////////////////////
```

```
    // PARA LA PRACTICA 07
    // HELICOPTERO
    ////////////////////////////
    GLfloat mueveHelicoptero;
    ////////////////////////////

    // PARA COMPROBAR QUE LA JERARQUIA CON LA LUZ SE HIZO DE MANERA CORRECTA (SOLO COMO EJEMPLO)
    ////////////////////
    GLfloat rotaeny;
    GLfloat rotaenz;
    ////////////////////
```

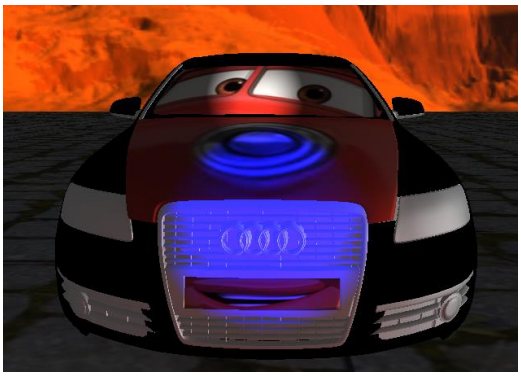
Por último, en shader_light.frag comenté la última línea y des comenté la penúltima para poder ver las luces.

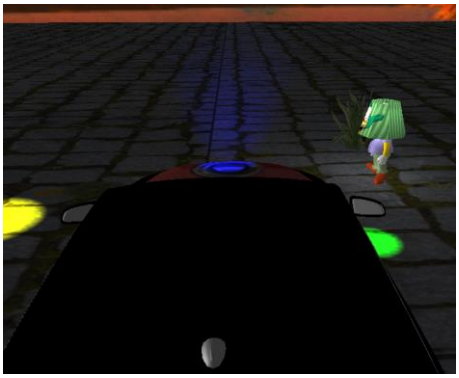
```
void main()
{
    vec4 finalcolor = CalcDirectionalLight();
    finalcolor += CalcPointLights();
    finalcolor += CalcSpotLights();
    color = texture(theTexture, TexCoord)*vColor*finalcolor; // Descomentar esta linea
    //color = texture(theTexture, TexCoord)*vColor; // Comentar esta linea para las luces
}
```

Ejecución del programa:

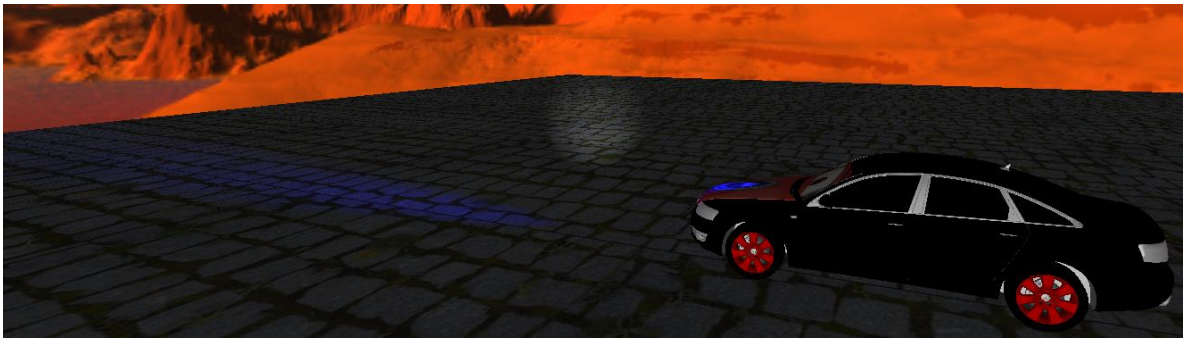
Vista inicial:

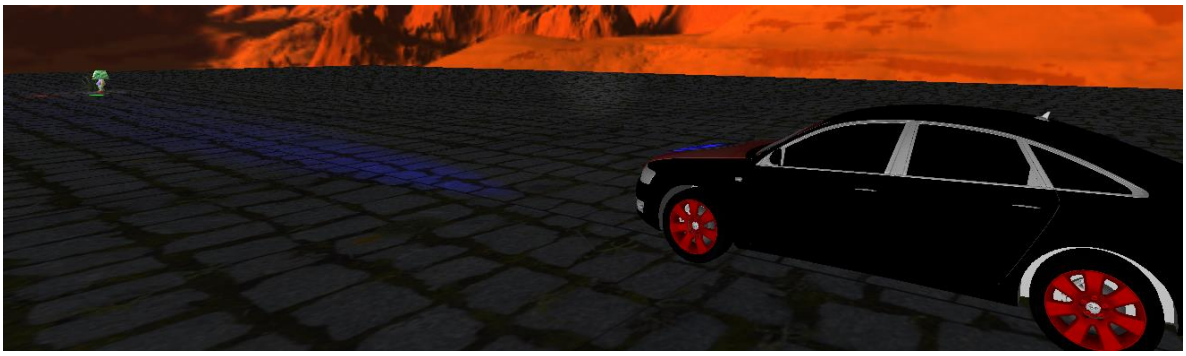
Spotlight azul:



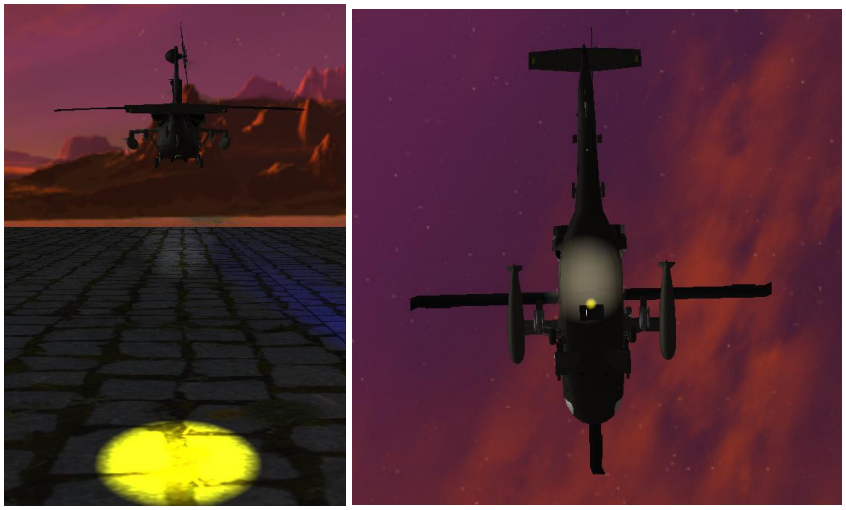


Avanzar y retroceder:

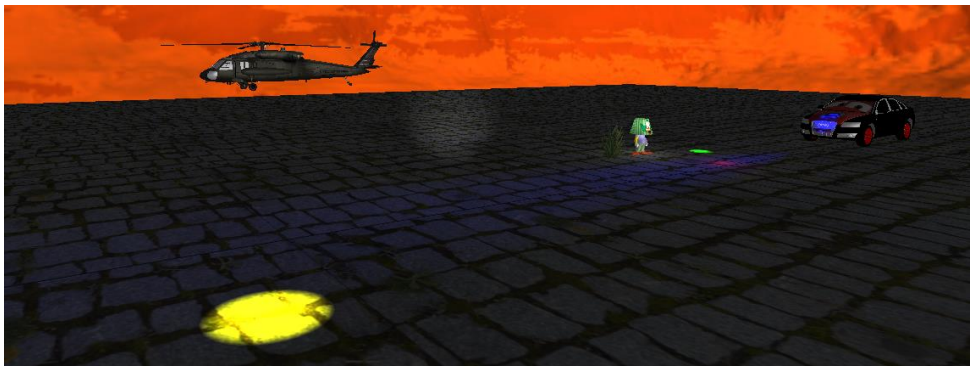


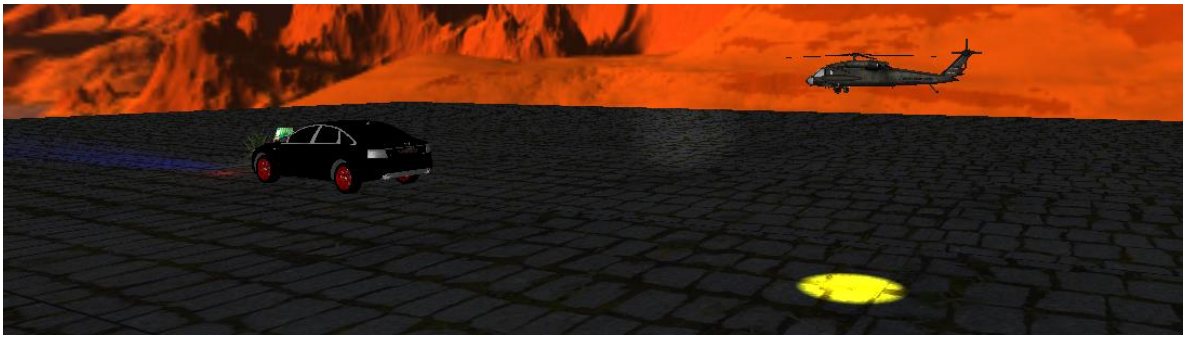


Spotlight amarilla:



Avanzar y retroceder:





Pointlight blanca y lampara:

Problemas presentados

1. Jerarquía
   Para colocar las spotlight con jerarquía fue un poco complicado.
   Pero buscando en internet encontré como obtener a partir de model la ubicación y la dirección para pasarle estos valores a la función SetFlash.

```
model = modelaux; // A PARTIR DEL COCHE

// PARA LA LUZ AZUL CON JERARQUIA

glm::vec3 posicion = glm::vec3(model[3]); //+ glm::vec3(-12.0f, -1.0f, 0.0f); // AJUSTAR LA POSICION A PARTIR DEL COCHE
glm::vec3 direccion = glm::vec3(model[2]);

spotLights[2].SetFlash(posicion, direccion);
```

2. SetFlash
   En cuestión de la posición.
   Esta tenia los valores de la posición del coche, por lo tanto, la luz era capaz de moverse junto con él.
   La parte de código que esta comentada era para ajustar la luz un poco más enfrente del coche.

Al hacer esto todo funcionaba de manera correcta y de hecho lo podía dejar así, pero no lo hice por cuestión de la dirección. Como la luz iba a estar alejada de punto inicial, al momento de rotar el coche esta iba a estar un poco desfasada, por eso decidí dejarla en la posición original.

En cuestión de dirección.
Esta tenia los valores de la dirección del coche, por lo tanto, la luz era capaz de rotar junto con el.
Esto no era necesario en la práctica, pero agregue rotación por teclado para comprobar que la luz rotara con el coche.

Si hubiera ocupado SetPos o en SetFlash hubiera puesto directamente la dirección de la luz esta no hubiera rotado con él.

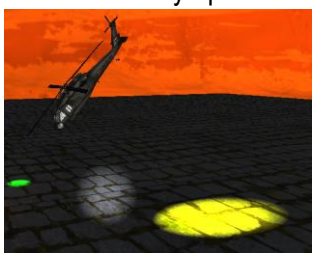
En esta parte me di cuenta de que si no le paso a SetFlash la dirección directamente o no la coloco al crear la Spotlight, esta tomaba la dirección del eje +Z.
Esto no lo note al principio porque el coche en un inicio estaba mirando hacia +Z y al rotarlo -90° en Y se posicionaba sobre el eje X, entonces la luz pareciera que se ajustó en -1.0f, 0.0f, 0.0f, pero esta originalmente hacia +Z.

Esto lo pude notar mejor con el helicóptero, si no le aplicamos ninguna rotación esta es su forma original:

La luz esta hacia +Z.
Por eso realice lo siguiente:

```
direccionH = -direccionH; // PORQUE APUNTA A Z INICIALMENTE EN EL MODELO ORIGINAL Y QUIERO QUE APUNTE A - Z EN EL MODELO ORIGINAL
// AL HACER LAS ROTACIONES ES COMO SI APUNTARA HACIA -Y
```

Para que fuera hacia -Z, que en realidad parecía -Y.
Haciendo eso ya podía rotar el helicóptero junto con la luz.



Conclusión

Me pude evitar varios problemas si solo le hubiera pasado la posición a la función SetFlash y hubiera colocado la dirección directamente.

En este caso si no le especifico la dirección de la luz parece que la toma como +Z, entonces es importante ver como se importan los modelos si es que se planea rotarlos junto con una luz.

Si solo planeamos mover la luz junto al modelo podemos ocupar SetPos y al crear la luz establecemos su dirección. Ya depende de lo que queramos realizar.

En general pienso que las actividades solicitadas se realizaron de manera correcta y lo extra solo fue con la intención de comprobar que lo implementado funcionara como se esperaba.

Bibliografía

The models resource. (s.f). Krusty Lamp. Consultado el 4 de octubre del 2024 de https://www.models-resource.com/pc_computer/simpsonshitrun/model/45951/