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 8

NOMBRE COMPLETO: BRANDON HERNANDEZ SOLIS

Nº de Cuenta: 318263113

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 6

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 12/10/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

1. 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.

No me dio tiempo

2. 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 de acuerdo a alguna bandera asignada por ustedes.

Para esta actividad tuve que ver cómo funcionaba el shader de luces y modificarlo, así logré incluir banderas dentro del shader para que pudiera saltar luces que se iban a renderizar en caso de que se enviara un valor booleano de la luz que se estaría apagada.

La luz se agregó a la jerarquía del coche para obtener su posición e irse desplazando, aunque no estuviera encendida. Además, para el control de encendido y apagado se usó el seguimiento de la tecla Y y U para controlar las banderas del shader.

Main.cpp:

```
304     //luz Faro
305     spotLights[2] = SpotLight(0.0f, 1.0f, 1.0f,
306         10.0f, 5.0f,
307         15.0f, 2.0f, 0.0f,      //Posicion inicial
308         -5.0f, 0.0f, 0.0f,      //Direccion en -X
309         0.1f, 0.1f, 0.1f,
310         25.0f);
311     spotLightCount++;
312
313     //luz reversa
314     spotLights[3] = SpotLight(0.0f, 1.0f, 1.0f,
315         10.0f, 5.0f,
316         15.0f, 2.0f, 0.0f,      //Posicion inicial
317         5.0f, 0.0f, 0.0f,       //Direccion en -X
318         0.1f, 0.1f, 0.1f,
319         25.0f);
320     spotLightCount++;
```

```
324     //Declaro variables
325     // Declaración global
326     bool skipLight1 = false;
327     bool skipLight2 = false;
328
329     bool skipSpotLight1 = false;
330     bool skipSpotLight2 = false;
331
332     bool yKeyPressed = false;
333     bool uKeyPressed = false;
334
335     bool tKeyPressed = false;
336     bool rKeyPressed = false;
```

```
383     //Se ajsutan las luces a renderizar
384     shaderList[0].SetPointLights(pointLights, pointLightCount, skipLight1, skipLight2);
385
386     shaderList[0].SetSpotLights(spotLights, spotLightCount, skipSpotLight1, skipSpotLight2);
387
```

```
431         // Guardo la posición del coche
432         glm::vec3 PosicionFaroReversa = glm::vec3(model[3]) + glm::vec3(0.0f, -1.0f, 0.0f);
433         glm::vec3 PosicionFaro = glm::vec3(model[3]) + glm::vec3(-4.0f, -1.0f, 0.0f);
434
435         // Asigno la posición del coche a la luz para que se mueva con él
436         spotLights[3].SetPos(PosicionFaroReversa);  // Actualiza la posición de la luz
437         spotLights[2].SetPos(PosicionFaro);  // Actualiza la posición de la luz
438
439         // Renderizo el carro
440         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
441         Kitt_M.RenderModel();
```

```
493         //Alternar Luces del Carro
494
495         if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_Y) == GLFW_PRESS) {
496             // Encender la luz al presionar la tecla
497             skipSpotLight1 = false; // La luz se enciende
498         }
499         if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_Y) == GLFW_RELEASE) {
500             skipSpotLight1 = true; // Permitir que la tecla pueda alternar nuevamente al soltarse
501         }
502
503         if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_U) == GLFW_PRESS) {
504             // Encender la luz al presionar la tecla
505             skipSpotLight2 = false; // La luz se enciende
506         }
507         if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_U) == GLFW_RELEASE) {
508             skipSpotLight2 = true; // Permitir que la tecla pueda alternar nuevamente al soltarse
509         }
510
511
```

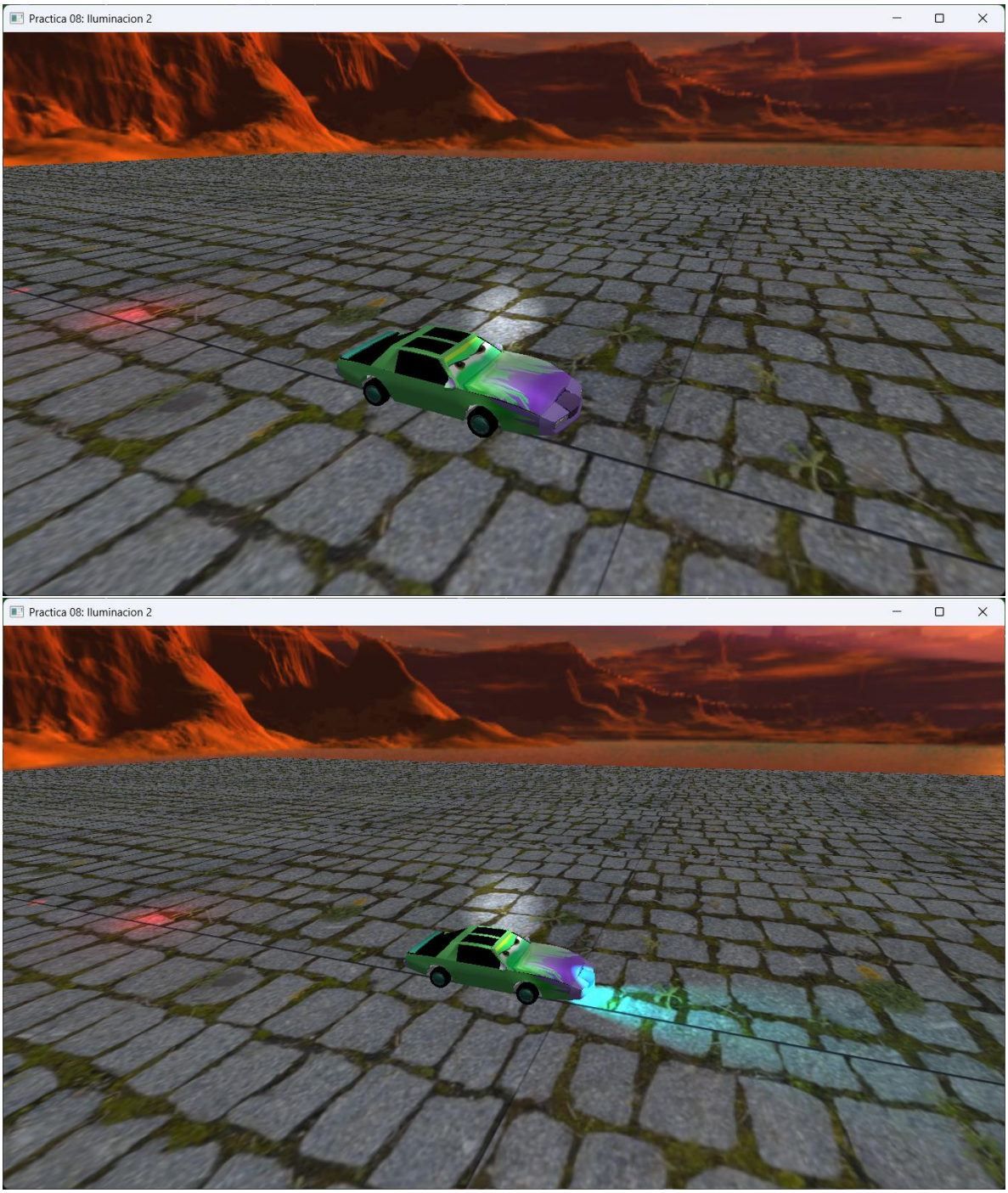
Shader_light.cpp:

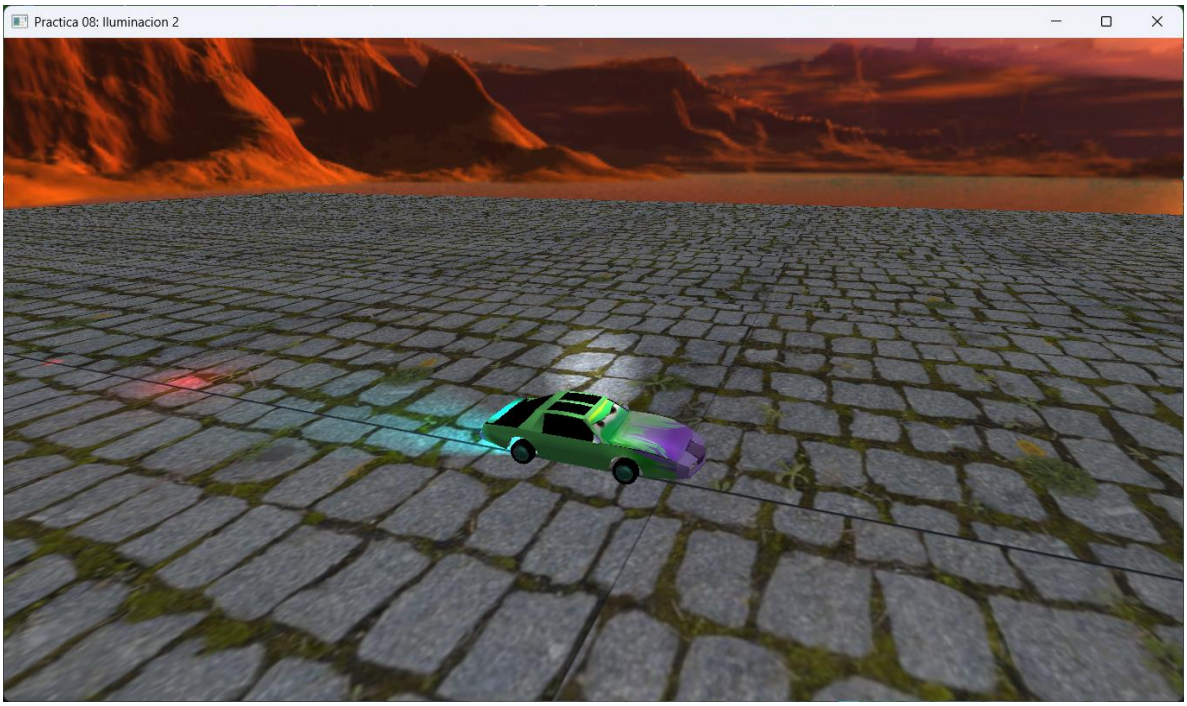
```
274 void Shader::SetSpotLights(SpotLight* sLight, unsigned int lightCount, bool skipLight1, bool skipLight2)
275 {
276     if (lightCount > MAX_SPOT_LIGHTS) lightCount = MAX_SPOT_LIGHTS;
277
278     // Ajustar la cantidad de luces que se renderizarán, excluyendo las que se van a omitir
279     unsigned int actualLightCount = lightCount;
280     if (skipLight1 == true && skipLight2 == false) actualLightCount--; // Disminuir
281     if (skipLight1 == false && skipLight2 == true) actualLightCount--; // Disminuir
282
283     if (skipLight1 == false && skipLight2 == false) actualLightCount= lightCount;
284     if (skipLight1 == true && skipLight2 == true) actualLightCount = lightCount-2;
285
286     glUniform1i(uniformSpotLightCount, actualLightCount);
287
288     unsigned int renderedLights = 0;
289
290     for (size_t i = 0; i < lightCount; i++)
291     {
292         // Saltar las luces 2 y 3 si los booleanos respectivos son verdaderos
293         if ((i == 3 && skipLight1) || (i == 2 && skipLight2)) {
294             continue;
295         }
296
297         sLight[i].UseLight(
298             uniformSpotLight[renderedLights].uniformAmbientIntensity,
299             uniformSpotLight[renderedLights].uniformColour,
```

Shader_light.h:

```
37      //Nueva Declaracion
38      void SetPointLights(PointLight* pLight, unsigned int lightCount, bool skipLight2, bool skipLight3);
39      void SetSpotLights(SpotLight* sLight, unsigned int lightCount, bool skipLight2, bool skipLight3);
40
```

Resultados:


Practica 08: Iluminacion 2
Practica 08: Iluminacion 2



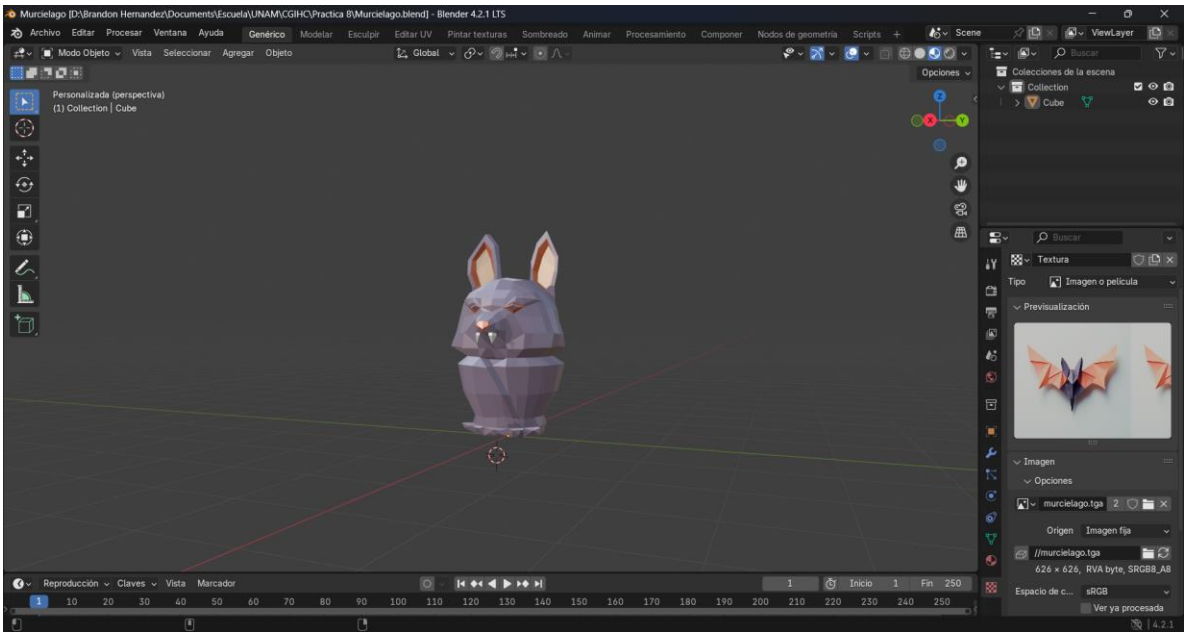
3.- Agregar otra luz de tipo puntual ligada a un modelo elegido por ustedes (no lámpara) y que puedan prender y apagar de forma independiente con teclado tanto la luz de la lámpara como la luz de este modelo (la luz de la lámpara debe de ser puntual, si la crearon spotlight en su reporte 7 tienen que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

Para esta actividad elegí una figura de un murciélago el cual edité en blender para que tuviera ojos y pudiera colocar una luz en su interior para asemejar que tendría los ojos rojos.

Como en el caso del ejercicio pasado, para tener control independiente sobre las luces a renderizar se tuvo que modificar el shader y usar banderas para controlar las luces que se omitirían.

En este caso dentro del main se usó usaron condicionales para detectar la presión de las teclas T y R, y así poder controlar cada modelo independientemente, también se agregó una variable de control para evitar que al mantener presionada la tecla se provoque que la luz parpadee repetidamente.

Blender:



Main.cpp:

```
214
215     //Modelo Murcielago
216     Murcielago = Model();
217     Murcielago.LoadModel("Models/Murcielago.obj");
218
219     //Modelo Lampara
220     Lampara = Model();
221     Lampara.LoadModel("Models/Lampara.obj");
222
```

```
267     /// EJERCICIOS
268     //Declaración de luz de mi lampara
269     pointLights[1] = PointLight(1.0f, 1.0f, 1.0f,   // Color blanco
270         1.0f, 3.0f,                 // Intensidad alta
271         40.0f, 10.0f, 0.0f,         // Posicion centrada en la lampara
272         0.1f, 0.1f, 0.02f);         // Atenuacion
273     pointLightCount++;
274
275     //Declaración de luz Estatua
276     pointLights[2] = PointLight(1.0f, 0.0f, 0.0f,   // Color Rojo
277         0.05f, 0.03f,               // Intensidad
278         0.0f, 4.0f, -20.0f,         // Posicion centrada en la lampara
279         0.05f, 0.05f, 0.02f);       // Atenuacion
280     pointLightCount++;
281
```

```cpp
//Declaro variables
// Declaración global
bool skipLight1 = false;
bool skipLight2 = false;

bool skipSpotLight1 = false;
bool skipSpotLight2 = false;

bool yKeyPressed = false;
bool uKeyPressed = false;

bool tKeyPressed = false;
bool rKeyPressed = false;
```

```cpp

//Se ajsutan las luces a renderizar
shaderList[0].SetPointLights(pointLights, pointLightCount, skipLight1, skipLight2);

shaderList[0].SetSpotLights(spotLights, spotLightCount, skipSpotLight1, skipSpotLight2);

```

```cpp
//Ejercicio 3

//Alternar luces:
if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_T) == GLFW_PRESS && !tKeyPressed) {
    // Alternar el booleano de la luz 1
    skipLight1 = !skipLight1;
    tKeyPressed = true; // Marcar que la tecla se presionó
}

if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_T) == GLFW_RELEASE) {
    tKeyPressed = false; // Permitir que la tecla pueda alternar nuevamente al soltarse
}

if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_R) == GLFW_PRESS && !rKeyPressed) {
    // Alternar el booleano de la luz 2
    skipLight2 = !skipLight2;
    rKeyPressed = true; // Marcar que la tecla se presionó
}

if (glfwGetKey(mainWindow.getMainWindow(), GLFW_KEY_R) == GLFW_RELEASE) {
    rKeyPressed = false; // Permitir que la tecla pueda alternar nuevamente al soltarse
}
```

```
// Lampara
modellamp = glm::mat4(1.0);

// Ajusto posicion
modellamp = glm::translate(modellamp, glm::vec3(40.0f, 0.0f, 0.0f));
modellamp = glm::scale(modellamp, glm::vec3(0.25f, 0.25f, 0.25f));
modellamp = glm::rotate(modellamp, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));

// Renderizo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modellamp));
Lampara.RenderModel();


//Declaro otra matriz para la estatua
glm::mat4 modelmur(1.0);

// Murcielago
modelmur = glm::mat4(1.0);

// Ajusto posicion
modelmur = glm::translate(modelmur, glm::vec3(0.0f, -1.0f, -20.0f));
modelmur = glm::scale(modelmur, glm::vec3(3.0f, 3.0f, 3.0f));
modelmur = glm::rotate(modelmur, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));

// Renderizo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelmur));
Murcielago.RenderModel();
```

Window.cpp

```
// EJERCICIO

if (key == GLFW_KEY_T && action == GLFW_PRESS)
{
    // Coloca aquí la acción que desees ejecutar al presionar 'T'
    //printf("Se presionó la tecla T\n");
}

if (key == GLFW_KEY_R && action == GLFW_PRESS)
{
    // Coloca aquí la acción que desees ejecutar al presionar 'R'
    //printf("Se presionó la tecla R\n");
}
```

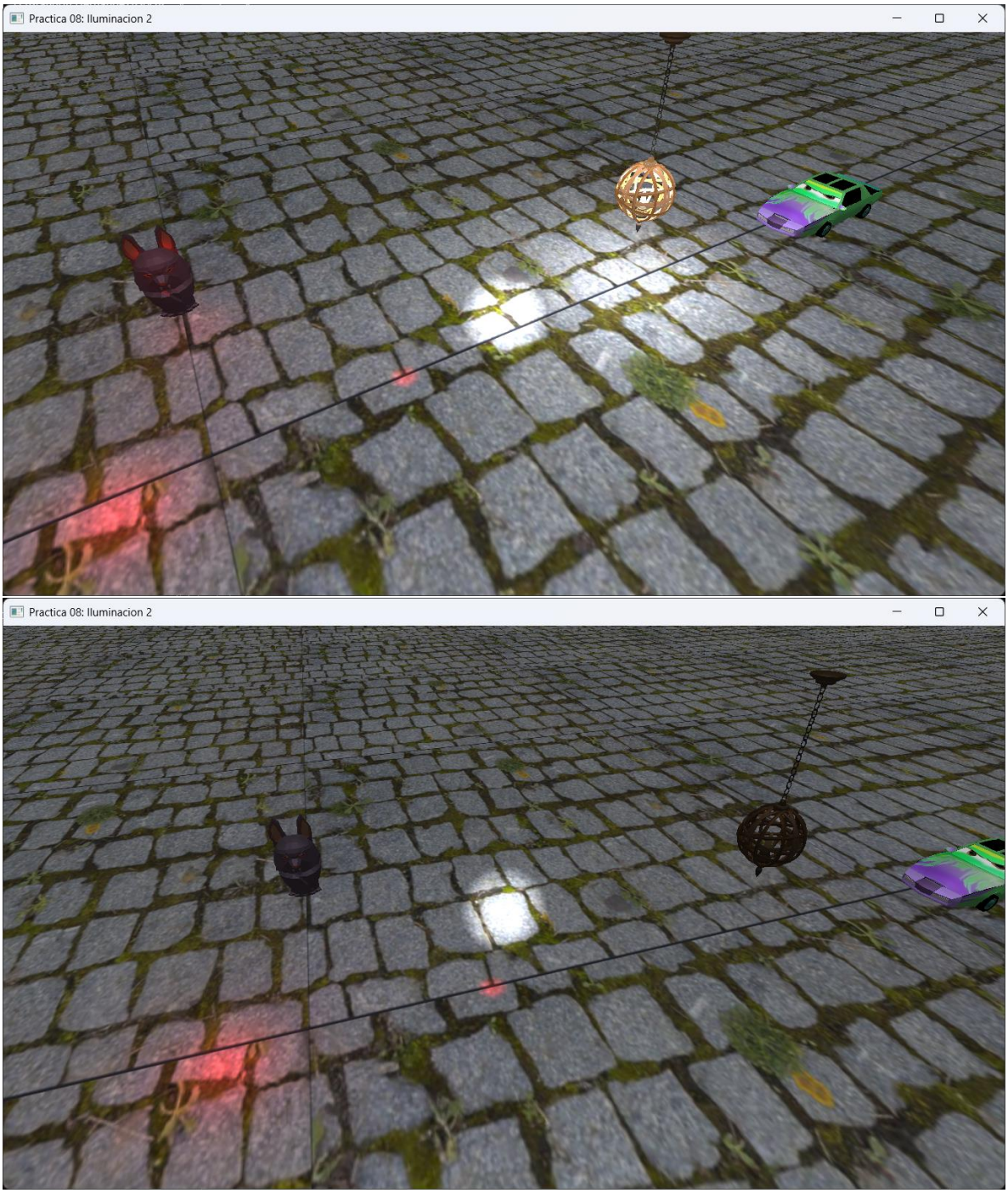
Shader_light.cpp:

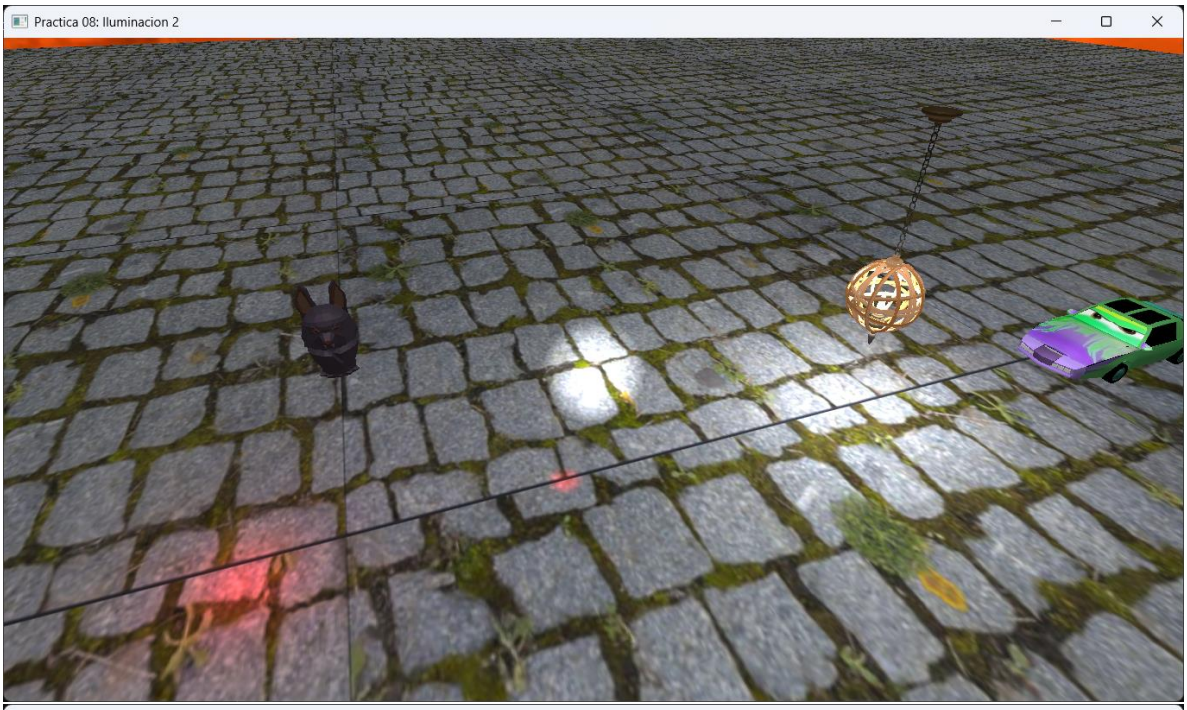
```cpp
void Shader::SetPointLights(PointLight* pLight, unsigned int lightCount, bool skipLight1, bool skipLight2)
{
    if (lightCount > MAX_POINT_LIGHTS) lightCount = MAX_POINT_LIGHTS;

    // Ajustar la cantidad de luces que se renderizarán, excluyendo las que se van a omitir
    unsigned int actualLightCount = lightCount;
    if (skipLight1 && lightCount > 1) actualLightCount--; // Disminuir si se salta la luz 2 (índice 1)
    if (skipLight2 && lightCount > 2) actualLightCount--; // Disminuir si se salta la luz 3 (índice 2)

    glUniform1i(uniformPointLightCount, actualLightCount);

    unsigned int renderedLights = 0;

    for (size_t i = 0; i < lightCount; i++)
    {
        // Saltar las luces 2 y 3 si los booleanos respectivos son verdaderos
        if ((i == 1 && skipLight1) || (i == 2 && skipLight2)) {
            continue;
        }

        // Usar la luz para renderizarla
        pLight[i].UseLight(uniformPointLight[renderedLights].uniformAmbientIntensity,
            uniformPointLight[renderedLights].uniformcolor,
            uniformPointLight[renderedLights].uniformDiffuseIntensity,
            uniformPointLight[renderedLights].uniformPosition,
            uniformPointLight[renderedLights].uniformConstant,
            uniformPointLight[renderedLights].uniformLinear,
            uniformPointLight[renderedLights].uniformExponent);

        renderedLights++;
```

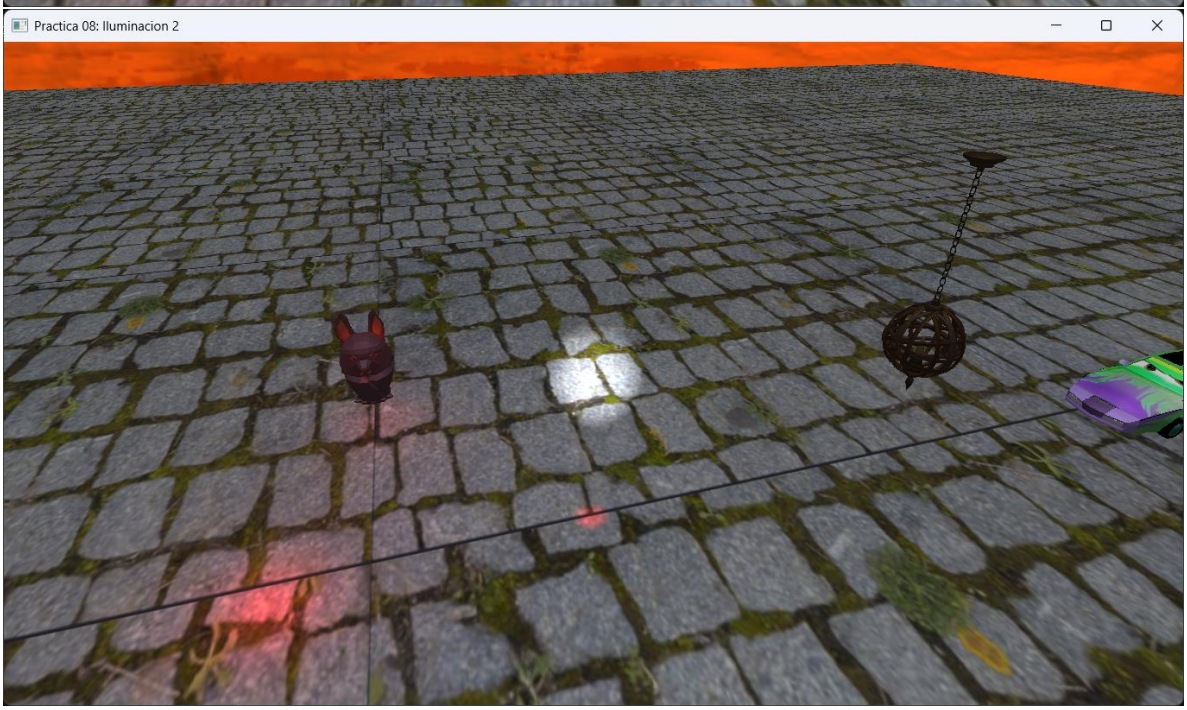
Shader_light.h:

```cpp

    //Nueva Declaracion
    void SetPointLights(PointLight* pLight, unsigned int lightCount, bool skipLight2, bool skipLight3);
    void SetSpotLights(SpotLight* sLight, unsigned int lightCount, bool skipLight2, bool skipLight3);

```

Resultados:


Practica 08: Iluminacion 2
Practica 08: Iluminacion 2


Practica 08: Iluminacion 2


Practica 08: Iluminacion 2

2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla.

No hubo problemas

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.
b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica
c. Conclusión

Los ejercicios de esta práctica estuvieron laboriosos debido a que se necesita entender el funcionamiento de los shaders y dominar los temas de las practicas anteriores sobre modelos, jerarquía, texturas y luces. Que se juntaron varias cosas y se implementó control dentro del código lo cual es útil para ir preparando lo necesario para la elaboración del proyecto final