Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Computación Gráfica e Interacción Humano Computadora

Reporte de Práctica No. 6 Iluminación

Computación Gráfica

Beto Pérez Iván Alejandro

Grupo: 11

Prof. Ing. Jose Roque Roman Guadarrama

Semestre 2021-2

Gpo. Teoría: 03

No. Cuenta: 315131437

Ingeniería en Computación.

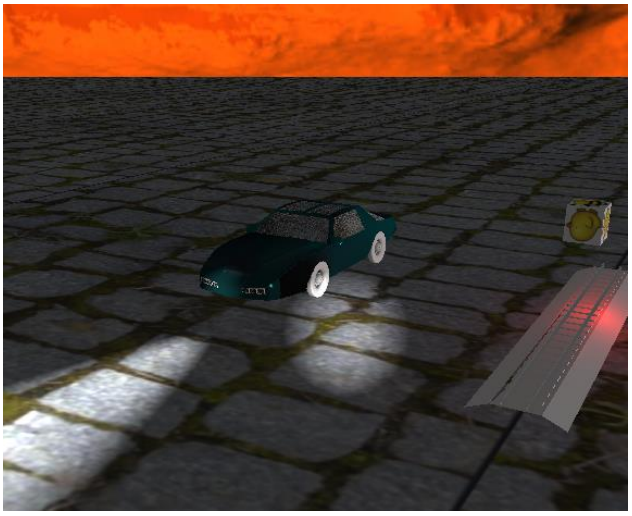
Fecha de entrega límite: 27/Junio/2021

1. **Ejecución de ejercicios**

Para el primer ejercicio, se creó la instancia de la luz spotlight para el faro del auto, justo después de las líneas del modelo para que se mantuviera la luz cuando se creara el auto y se le agregaron a los valores de posición las funciones de posición del auto para que cuando se mueva el auto, se mueva la luz con él. Y se instancia el modelo propio que se texturizó en prácticas anteriores.

```
//automovil
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f+mainWindow.getmueveauto(), 0.5f,
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//Kitt_M.RenderModel();
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
automovil.RenderModel();

//luz del coche
spotLights[2] = SpotLight(1.0f, 1.0f, 1.0f,
    1.0f, 3.0f,
    6.0f+mainWindow.getmueveauto(), 0.16f, 0.5f+mainWindow.getmueveautoz(),
    -5.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    16.0f);
spotLightCount++;
```



Y para el movimiento en el eje Z, se agregó una función en el código de Window.h, donde se declara la variable que ira incrementando y decrementando, junto con la función que devolverá el valor de dicha variable. A continuación, se muestra el código de la cabecera de window.

```
GLfloat getmueveauto() { return mueveauto; }
GLfloat getmueveautoz() { return mueveautoz; }
```

```
GLfloat mueveautoz;
GLfloat mueveauto;
```

Y después se configura el código de window.cpp, primero de declaran las variables en la clase de window para que reconozca las variables.

```
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    muevex = 2.0f;
    mueveauto = 3.0f;
    mueveautoz = 3.0f;
    muevey = 2.0f;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
}
```

Y despues, en la función de “ManejaTeclado” se agregan los casos donde se presionan, en mi caso son las teclas M para derecha y V para izquierda, para que el valor de “mueveautoz” incremente su valor o decremente según sea el caso.

```
if (key == GLFW_KEY_M) {
    theWindow->mueveautoz -= 1.0;
}
if (key == GLFW_KEY_V) {
    theWindow->mueveautoz += 1.0;
```

Y al final se agregan las funciones de movimiento en los valores de la posición del modelo del auto, en la función de translate.

```
//automovil
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f+mainWindow.getmueveauto(), 0.5f, -1.5f+mainWindow.getmu
```

Y para el segundo ejercicio, se hace algo similar como para el movimiento del auto, se declara la función y las variables en el código del window.h.

```
GLfloat getmuevex() { return muevex; }
GLfloat getmueveblackhawky() { return muevey; }
```

```
GLfloat muevex;
GLfloat muevey;
```

Y después de hacer esto, se modifica el código de window.cpp, donde primero se vuelven a declarar las variables para que sean reconocidas en el código.

```
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    muevex = 2.0f;
    mueveauto = 3.0f;
    mueveautoz = 3.0f;
    muevey = 2.0f;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
}
```

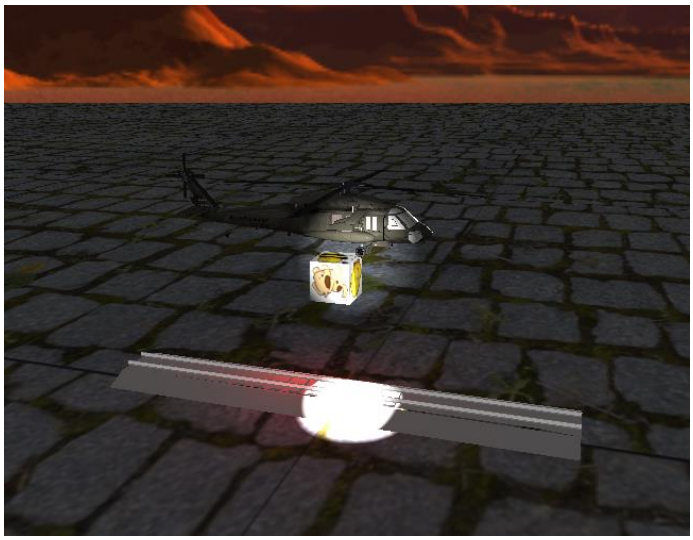
Y después se agregan los casos de movimiento en la función “ManejaTeclado”, donde para mi caso nombre a mi variable de movimiento en y “muevey”, y para el caso del movimiento hacia arriba es con la tecla T y para mover hacia abajo es con la tecla I.

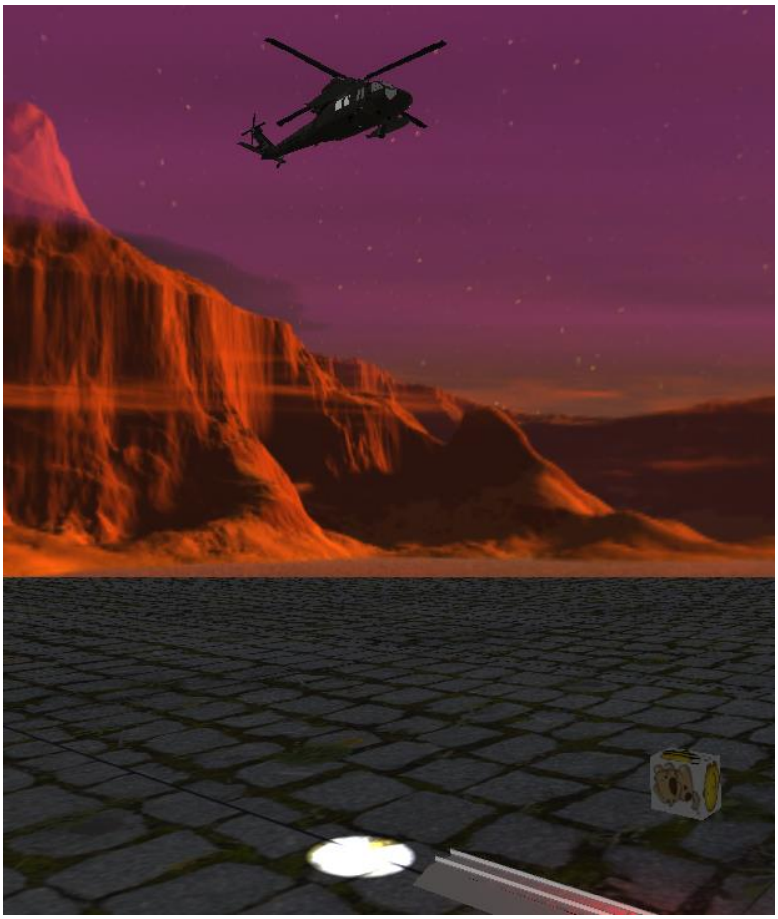
```
if (key == GLFW_KEY_I) {
    theWindow->muevey -= 1.0;
}
if (key == GLFW_KEY_T) {
    theWindow->muevey += 1.0;
}
```

Y finalmente, en el código de Iluminación, en donde tenemos nuestro modelo, se agregan las funciones en los valores de posición de la función translate del helicóptero.

```
/agregar incremento en X con teclado
odel = glm::mat4(1.0);
odel = glm::translate(model, glm::vec3(-20.0f+mainWindow.getmuevex(), 6.0f+mainWindow.getmueveblackh
```

Para la luz del helicóptero, se creó una instancia de luz spotlight y se puso justo debajo del modelo del blackhawk y para hacer que se moviera con el helicóptero, se le añadieron las funciones de movimiento del modelo a la instancia de luz para que se moviera junto con el modelo.





2. Problemas presentados

La primera vez que puse mis funciones de movimiento, me faltó declarar las variables que incrementan y decrementan en la clase de Window, ya que cuando no lo hice, no me cargaron los modelos al momento de correr el programa, aplicaba el movimiento, pero no veía nada que se moviera excepto que las luces.

3. Conclusiones

En esta práctica aprendí como es que se aplica movimiento a los modelos en distintos ejes, en donde hacer las funciones que incrementen o decrementen el valor del modelo en cierto eje, además de agregarles una luz que se podría decir que es fija al modelo para que se mueve junto con el modelo que estamos poniendo, además de volver a repasar lo de la jerarquía de modelos con el auto y las llantas.