UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 03

NOMBRE COMPLETO: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: sábado 24 de febrero de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

- **Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa**

1. **Instanciar primitivas geométricas para recrear el dibujo de la práctica pasada en 3D,se requiere que exista un piso; la casa tiene una ventana azul circular justo en medio de la pared trasera, 2 ventanas verdes en cada pared lateral iguales a las de la pared frontal y solo puerta en la pared frontal.**

Primero que nada, es importante mencionar que las figuras son capaces de rotar todas juntas, como si fueran una sola, sobre un mismo punto. Esto lo logre de la siguiente manera: primero, defino el punto, desde el que rotaran las figuras, en la siguiente línea de código:

```
glm::vec3 origen(0.0f, 0.0f, -4.0f);
```

Luego, para que todas las figuras aparezcan aquí, invoco los modelos en este punto, luego, los traslado a sus coordenadas originales en las que formaran la casa. El siguiente fragmento de código lleva a cabo esta función.

```
model = glm::mat4(1.0);
model = glm::translate(model, origen);//Trasladamos la figura al punto de origen, luego la dev
model = glm::scale(model, glm::vec3(8.0f, 8.0f, 8.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
```

El código anterior sirve para definir el cuadrado que constituye al cuerpo de la casa, luego ya solo le damos color y lo dibujamos:

```
//CUERPO DE LA CASA

color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular Y pirámide base cuadrangular
```

Se repite el proceso para todas las figuras de aquí en adelante, simplemente las trasladamos a su ubicación en la casita, los escalamos, les cambiamos el color y las dibujamos:

- Para el techo de la casa y la puerta:

```
//TECHO DE LA CASA
model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[4]->RenderMesh();

//PUERTA
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, -1.3f, 0.4f));
model = glm::scale(model, glm::vec3(0.25f, 0.4f, 0.25f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

- Para las ventanas:

```
//VENTANAS LATERALES Y FRONTALES

model = glm::translate(model, glm::vec3(1.25f, 1.5f, 0.2f));
model = glm::scale(model, glm::vec3(0.75f, 0.75f, 0.75f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

model = glm::translate(model, glm::vec3(-3.2f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

model = glm::translate(model, glm::vec3(-0.8f, 0.0f, -0.7f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

model = glm::translate(model, glm::vec3(-0.1f, 0.0f, -3.25f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

model = glm::translate(model, glm::vec3(4.8f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

- Para la ventana trasera:

```
//VENTANA TRASERA
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-2.0f, 0.0f, -3.5f));
model = glm::scale(model, glm::vec3(1.0f, 0.75f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render(); //dibuja esfera
```

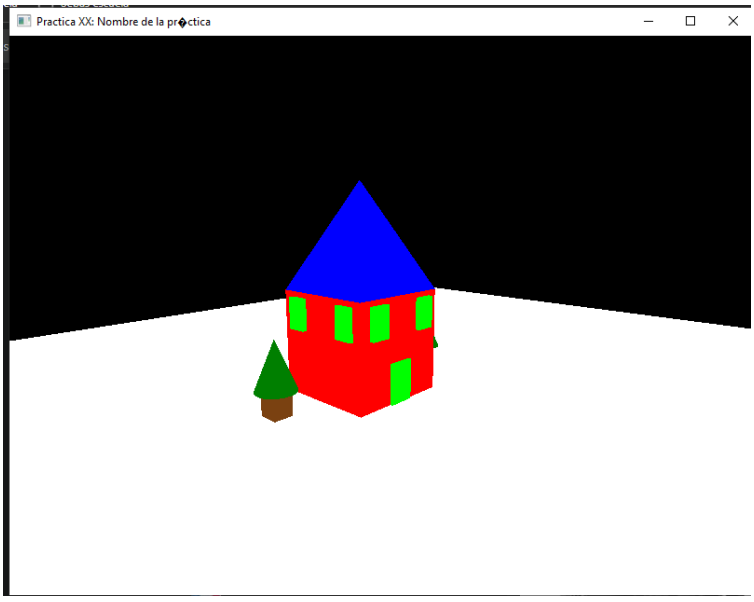
- Para los árboles:

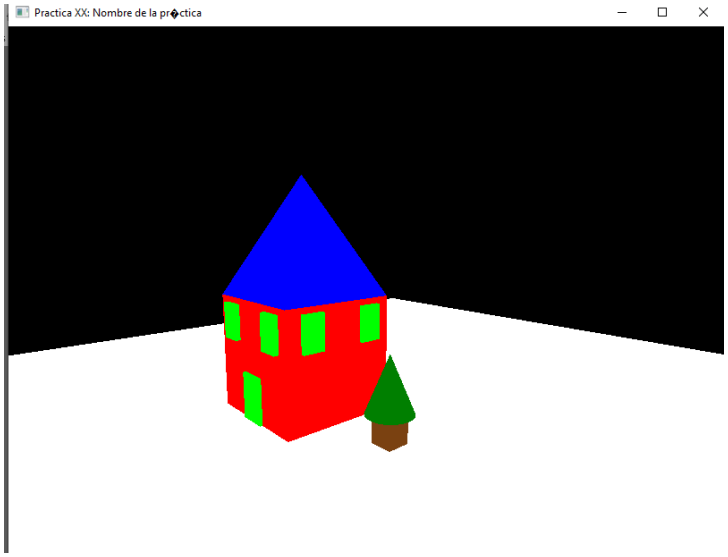
```
//TRONCOS ARBOLES

color = glm::vec3(0.478, 0.255, 0.067);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(+5.0f, -3.0f, +1.8f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

model = glm::translate(model, glm::vec3(-10.5f, 0.0f, +0.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

//HOJAS ARBOLES

color = glm::vec3(0.0, 0.5, 0.0);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(+0.0f, 1.5f, +0.0f));
model = glm::scale(model, glm::vec3(0.5f, 2.0f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono

model = glm::translate(model, glm::vec3(+21.0f, 0.0f, +0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[3]->RenderMeshGeometry();
```

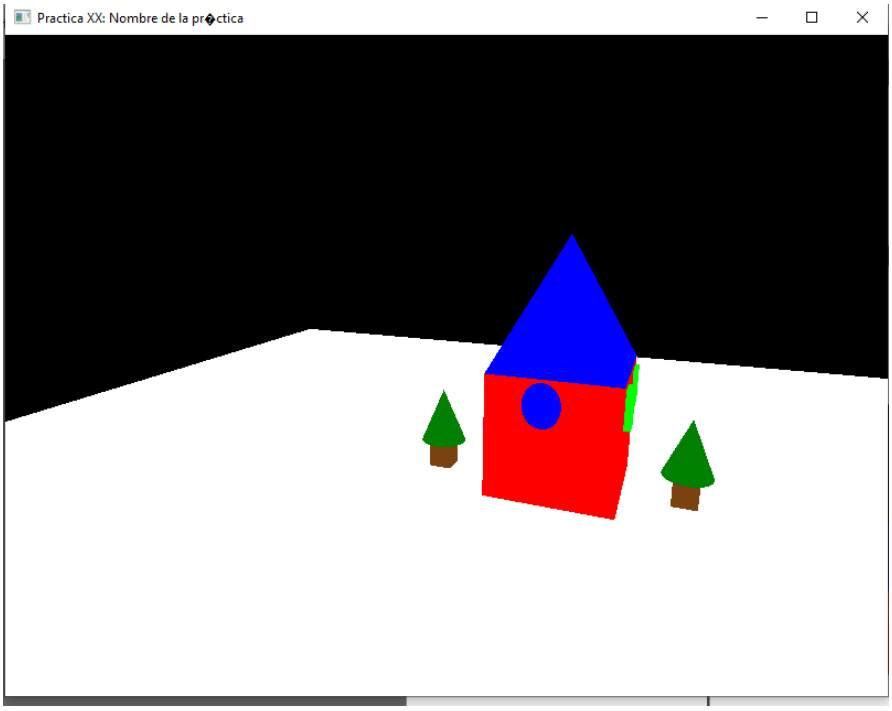
- Para el suelo:

```
//PISO

color = glm::vec3(1.0, 1.0, 1.0);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(+0.0f, -1.5f, +0.0f));
model = glm::scale(model, glm::vec3(100.0f, 1.0f, 100.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

Y la ejecución se ve así:







- **Problemas Presentados**

  En un inicio fue difícil averiguar como hacer que todas las figuras rotaran alrededor de un solo punto, para poder observarlas mediante la cámara y la rotación sin problema. Fue lo que me tomo mas tiempo de hecho. El resto fue más fácil de lograr, solamente fue llamar a las figuras y colorearlas, además de darles el tamaño con la funcion de escalado. Sin embargo, si hay un problema con el método que utilice: cada vez que creaba una nueva figura en el espacio, parecía que las coordenadas base se modificaban. Es decir, si creaba una figura en las coordenadas, por ejemplo, (2, 2, 2), estas coordenadas se convertían en las coordenadas de origen de la siguiente figura, por lo que a la hora de trasladar la siguiente figura, tenia que sumar, por ejemplo, (+15.0, 4.0, 3.0) a las

coordenadas (2,2,2). Fue una molestia deducir esto después de un rato, pero al final logre hacer la casita sin muchas trabas.

- **Conclusión:**

  En este ejercicio aprendí a trasladar distintos tipos de figuras, a rotarlas y a construir figuras en un ambiente 3D a partir de primitivas. También aprendí a crear figuras circulares y como se crean utilizando métodos específicos, como los conos, las esferas y los cilindros. Además, también aprendí a utilizar la cámara sintética de OpenGL