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 3

NOMBRE COMPLETO: Sanchez Villalpando Johan

Nº de Cuenta: 422028657

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 4

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 25/02/25

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

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

   Para este ejercicio se retoma la misma casa dibujada en la práctica 2, con la adición de que ahora usaremos la cámara para visualizarla y añadiendo figuras como esferas, pirámides cuadrangulares, cilindros, entre otros.

   Además, se debe añadir un piso, para el que en mi caso se eligió el color gris, modificando un cubo extendiéndolo y quitándole altura para simular el piso, una ventana circular azul en la parte trasera y 4 ventanas nuevas, 2 en cada lateral.

   Primero se añade el primer cuadrado rojo.

```
//Cuerpo rojo
model = glm::mat4(1.0f); //piso gris
color = glm::vec3(1.0f, 0.0f, 0.0f);  //gris
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, 0.7f, -3.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.2f, 2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo piso
```

   Para el piso se usa otro cubo más delgado en Y.

```
//Piso
model = glm::mat4(1.0f); //piso gris
color = glm::vec3(0.2f, 0.2f, 0.2f);  //gris
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, -0.45f, -3.0f));
model = glm::scale(model, glm::vec3(20.0f, 0.1f, 20.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo piso
```

   Para el techo se usa una pirámide cuadrangular.

```
//Techo azul
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);  //azul
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, 2.31f, -3.0f));
model = glm::scale(model, glm::vec3(2.5f, 1.0f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[4]->RenderMeshGeometry(); //piramide cuadrangular
```

   Para las ventanas se usan cubos más delgados en un eje, ya sea Z para los frontales o en X para las ventanas de los laterales.

```
//Ventana verde front izq
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.5f, 1.2f, -2.0f));
model = glm::scale(model, glm::vec3(0.65f, 0.65f, 0.05f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo verde ventana izq

//Ventana verde front der
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-0.5f, 1.2f, -2.0f));
model = glm::scale(model, glm::vec3(0.65f, 0.65f, 0.05f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo verde ventana izq
```

```
//Ventana verde der der
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-1.0f, 1.2f, -2.5f));
model = glm::scale(model, glm::vec3(0.05f, 0.65f, 0.65f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo verde ventana izq

//Ventana verde der izq
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-1.0f, 1.2f, -3.5f));
model = glm::scale(model, glm::vec3(0.05f, 0.65f, 0.65f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo verde ventana der

//Ventana verde izq der
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(1.0f, 1.2f, -2.5f));
model = glm::scale(model, glm::vec3(0.05f, 0.65f, 0.65f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo verde ventana der
```

```
//Ventana verde izq izq
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(1.0f, 1.2f, -3.5f));
model = glm::scale(model, glm::vec3(0.05f, 0.65f, 0.65f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo verde ventana izq
```

Para la puerta es también un cubo verde delgado en Z.

```
//puerta verde
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, -0.08f, -2.0f));
model = glm::scale(model, glm::vec3(0.65f, 0.65f, 0.05f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //cubo
```

Para la ventana circular azul se usa una esfera más delgada en el eje Z.

```
//Ventana circular trasera azul
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);  //azul
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, 0.75f, -4.0f));
model = glm::scale(model, glm::vec3(0.65f, 0.65f, 0.05f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render(); //dibuja esfera
```

Para los troncos café se usan cilindros, a los cuales también se les aumentó la resolución a 10.

```
CrearCilindro(10, 1.0f);//índice 2 en MeshList
```

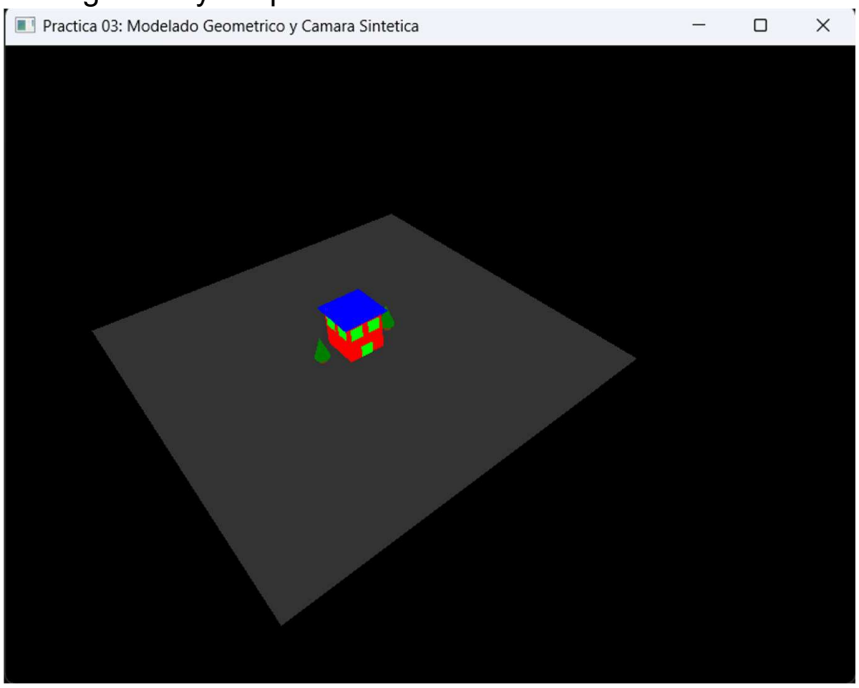
```
//tronco café der
model = glm::mat4(1.0f);
color = glm::vec3(0.478f, 0.225f, 0.067f);  //café
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(2.0f, -0.24, -3.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.3f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //cilindro

//tronco café izq
model = glm::mat4(1.0f);
color = glm::vec3(0.478f, 0.225f, 0.067f);  //café
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-2.0f, -0.24, -3.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.3f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //cilindro
```

Para los árboles usé conos.

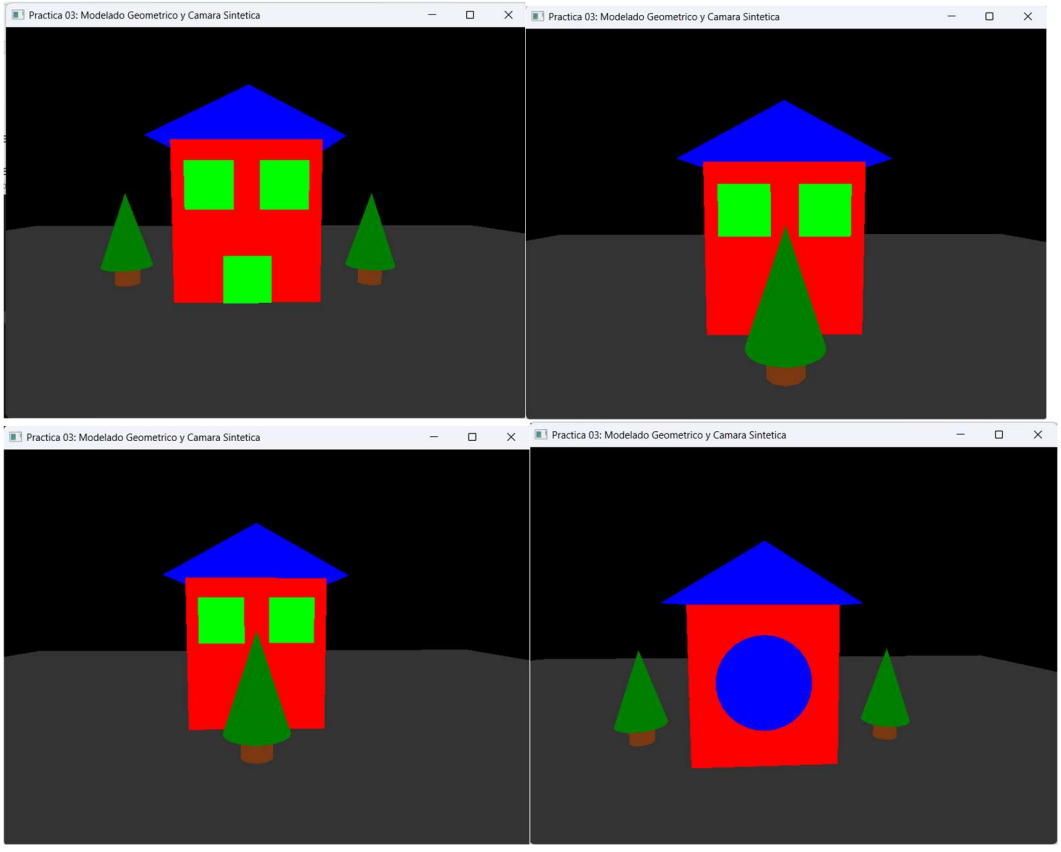
```
//cono verde izq
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.5f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-2.0f, 0.5f, -3.0f));
model = glm::scale(model, glm::vec3(0.2f, 1.2f, 0.2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[3]->RenderMeshGeometry(); //cilindro

//cono verde der
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.5f, 0.0f);  //verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(2.0f, 0.5f, -3.0f));
model = glm::scale(model, glm::vec3(0.2f, 1.2f, 0.2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[3]->RenderMeshGeometry(); //cilindro
```

Finalmente se obtiene el resultado final:

Vista general y del piso:



Vistas de la casa (frontal, lateral izq, lateral der, trasera):



2. **Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código**

   No se presentaron problemas, más que el propio acomodo, el cual es algo tedioso ya que es mover elemento por elemento en los 3 ejes, al principio es sencillo confundir los ejes y no poder ubicar correctamente sobre que eje se necesita mover la primitiva, después se vuelve más sencillo con forme se va comprendiendo el entorno.

3. **Conclusión:**
   a. **Los ejercicios de la clase: Complejidad, explicación**
   b. **Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.**

   Como conclusión considero que el ejercicio estuvo sencillo, ya que no se requiere modificar mucho código ni crear funciones o shaders, sin embargo, es algo tardado ya que se deben aplicar las transformaciones para cada primitiva, lo que puede llegar a ser tardado si se quiere un acomodo lo más correcto posible, la explicación fue buena y entendí de dónde vienen las cosas nuevas que se ven en el código, y tengo la idea de cómo se forman las primitivas y la lógica detrás de ello.

   Como comentarios es algo raro el movimiento de la cámara, ya que el mouse no se mantiene estático una vez dentro de la ventana de ejecución, por lo que al mover hay veces que el mouse sale de la ventana y es complicado hacer ciertos movimientos.