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: Barragán Pilar Diana

Nº de Cuenta: 318147981

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 27 de agosto de 2024

CALIFICACIÓN: _____

Ejercicio de clase práctica 3: Modelado Geométrico.

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.**

Para realizar este ejercicio lo primero fue generar el piso para a partir de este colocar todas las primitivas geométricas que conformarán a la casa posicionándolas encima, para lograr esto utilice la primitiva del cubo y simplemente en vez de ser tridimensional en la línea glm::scale(model, glm::vec3(6.0f, 0.0f, 6.0f)) el vector tiene la componente en y sin escalar para que en ese eje se mantuviera plano.

Una vez lograda la colocación del piso, puse las primitivas geométricas en el espacio, modificando los vectores de escala y translación, muy parecido a lo que ejecuté en el ejercicio práctico anterior, cambiando un poco el tamaño de escalado pues en algunas figuras se debía de aumentar como en el techo. Las formas de utilice fueron diez cubos uno para la casa, otros dos para el tronco de los árboles, dos para las ventanas frontales, dos para las ventanas de la derecha, dos para las ventanas de la izquierda y uno para la puerta, de la pirámide con base cuadrada utilice dos, una para el techo y otra para la vegetación de los árboles, por último para la ventana en la parte de atrás utilice una esfera que simplemente volví plana para colocarla en la casa.

```
//***PISO***
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 1.0f);

model = glm::translate(model, glm::vec3(0.0f, -0.97f, -4.0f));
model = glm::scale(model, glm::vec3(6.0f, 0.0f, 6.0f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
//se programe cambio entre proyección ortogonal y perspectiva
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular
```

Figura 1. Creación del piso de la casa.

```
//***CUBO ROJO***
model = glm::mat4(1.0);
//Traslación inicial para posicionar en -Z a los objetos
//model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
model = glm::translate(model, glm::vec3(0.0f, -0.26f, -4.0f));
model = glm::scale(model, glm::vec3(1.5f, 1.4f, 1.5f));
//otras transformaciones para el objeto
//model = glm::scale(model, glm::vec3(0.5f,0.5f,0.5f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
//se programe cambio entre proyección ortogonal y perspectiva
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
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
```

Figura 2. Creación de la base de la casa.

```
//TECHO AZUL

model = glm::mat4(1.0f);
color=glm::vec3(0.0f,0.0f,1.0f);

model = glm::translate(model, glm::vec3(0.0f, 0.95f, -4.0f));
model = glm::scale(model, glm::vec3(2.0f, 1.0f, 2.0f));
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));  //al presionar
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[4]->RenderMeshGeometry();//dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
```

Figura 3. Creación del techo de la casa.

```
//VENTANAS DE ENFRENTE
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(-0.3f, 0.0f, -3.4f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.4f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(0.3f, 0.0f, -3.4f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.4f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular
```

Figura 3. Creación de las ventanas de enfrente.

```
//VENTANAS DERECHA
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(-0.52f, 0.0f, -3.6f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(-0.52f, 0.0f, -4.4f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular
```

Figura 4. Creación de las ventanas del lado derecho.

```
//VENTANAS IZQUIERDA
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(0.52f, 0.0f, -3.6f));
model = glm::scale(model, glm::vec3(0.52f, 0.5f, 0.5f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(0.52f, 0.0f, -4.4f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular
```

Figura 5. Creación de las ventanas del lado izquierdo.

```
//PUERTA ENFRENTE
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);

model = glm::translate(model, glm::vec3(0.0f, -0.66f, -3.4f));
model = glm::scale(model, glm::vec3(0.4f, 0.55f, 0.4f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular
```

Figura 6. Creación de la puerta de enfrente.

```
//ÁRBOL DERECHO
//CUBO CAFÉ DERECHA
model = glm::mat4(1.0f);
color = glm::vec3(0.478f, 0.255f, 0.067f);

model = glm::translate(model, glm::vec3(-2.0f, -0.66f, -4.0f));
model = glm::scale(model, glm::vec3(0.6f, 0.6f, 0.6f));
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));  //al presionar
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular

//PIRAMIDE CUADRANGULAR VERDE OSCURO
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.5f, 0.0f);

model = glm::translate(model, glm::vec3(-2.0f, 0.1f, -4.0f));
model = glm::scale(model, glm::vec3(0.9f, 0.9f, 0.9f));
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));  //al presionar
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[4]->RenderMeshGeometry();//dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
```

Figura 7. Creación del árbol derecho.

```
//CUBO CAFÉ IZQUIERDA
model = glm::mat4(1.0f);
color = glm::vec3(0.478f, 0.255f, 0.067f);

model = glm::translate(model, glm::vec3(2.0f, -0.66f, -4.0f));
model = glm::scale(model, glm::vec3(0.6f, 0.6f, 0.6f));
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));  //al presionar la
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();//dibuja cubo y pirámide triangular

//PIRAMIDE CUADRANGULAR VERDE OSCURO
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.5f, 0.0f);

model = glm::translate(model, glm::vec3(2.0f, 0.1f, -4.0f));
model = glm::scale(model, glm::vec3(0.9f, 0.9f, 0.9f));
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));  //al presionar la
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[4]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
```

Figura 8. Creación del árbol izquierdo.

```
//VENTANA CIRCULAR
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);

model = glm::translate(model, glm::vec3(0.0f, -0.2f, -4.82f));
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.0f));
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));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();
```

Figura 9. Creación de la ventana circular posterior.

EJECUCIÓN DEL PROGRAMA:

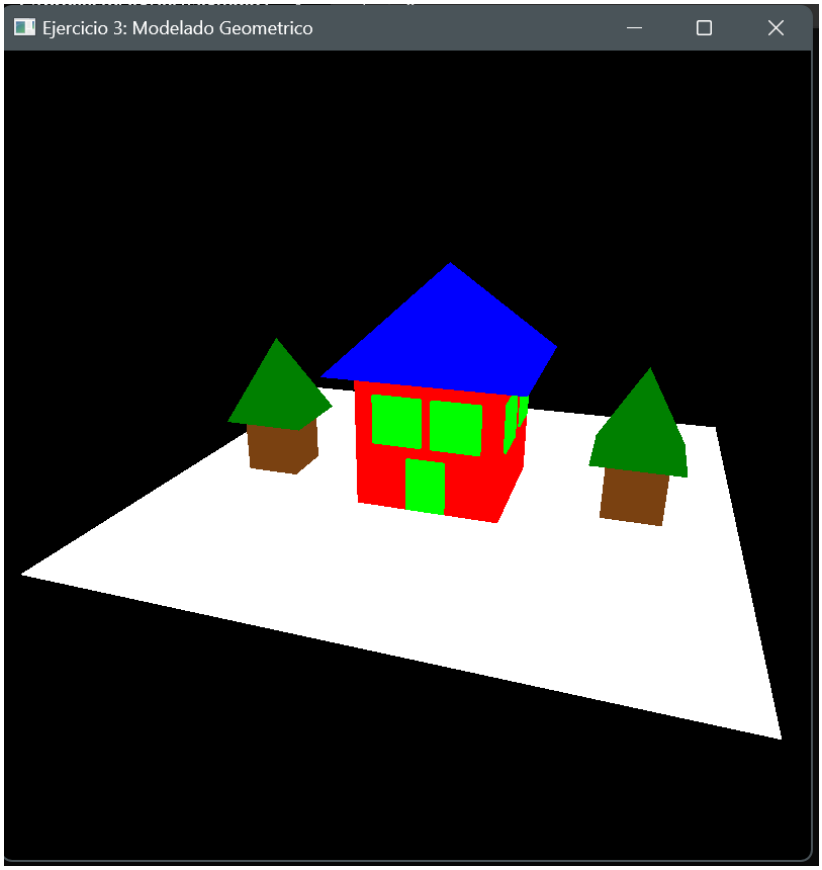

Figura 10. Casa con primitivas geométricas.

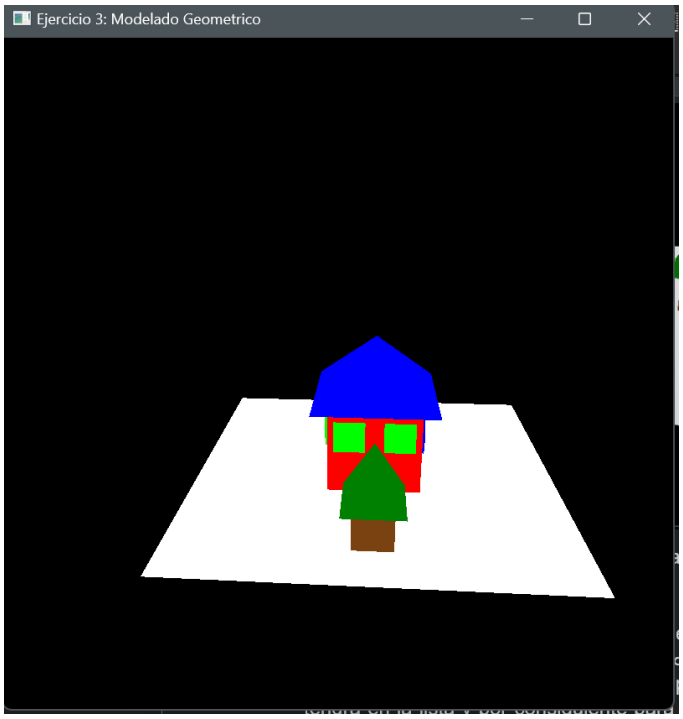

Figura 11. Casa con primitivas geométricas.

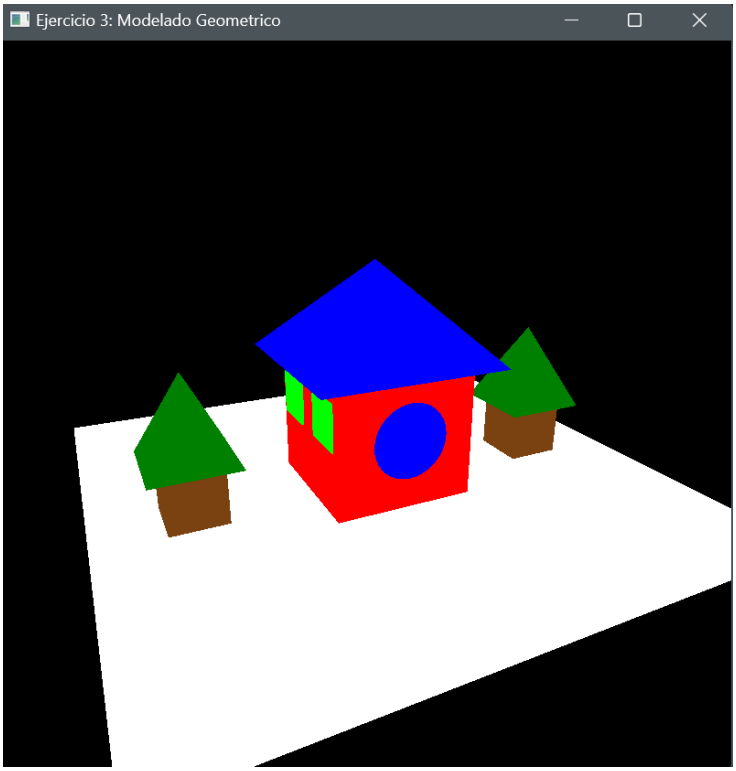

Figura 12. Casa con primitivas geométricas.

En este ejercicio el único inconveniente que encontré fue que al utilizar las teclas E, R y T para mover las figuras, se mueven cada una de ellas lo que distorsiona la imagen.

CONCLUSIONES:

El ejercicio me permitió comprender mejor todo lo que tiene que ver con la creación de las primitivas geométricas, así como su uso aplicación y modificación trabajando con el modelo tridimensional y la cámara de forma que ahora se siente cada vez más interactivo el modelo.

Finalmente el usar estas modificaciones me permitió entender mejor como es que se compone un escenario y se forman estos a partir de primitivas geométricas.