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 03

NOMBRE COMPLETO: Arroyo Ramìrez Carlos Alberto

Nº de Cuenta: 320185865

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 07/09/2025

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.

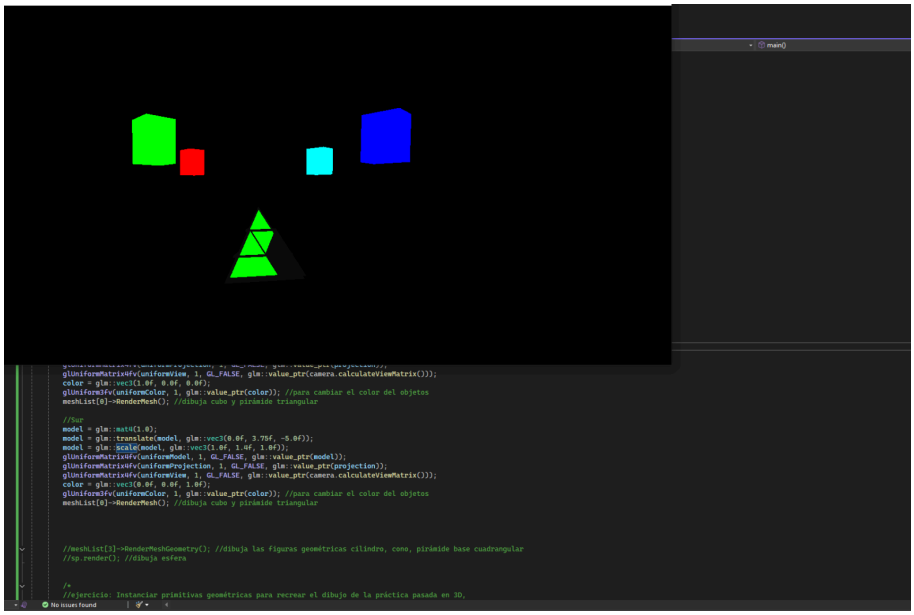
Genere los triángulos a mano, mi idea era hacerlo en blender y después copiar coordenadas y escalas en el programa, utilice el mismo código del ejercicio de clase, solo acomodando y dando escala a cada cono, el cual se le dio una resolución de 3, lo mismo para hacer las divisiones laterales, se empleo un cilindro con definición de 3.

```
//Piramide principal negra
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 3.0f, 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()));
color = glm::vec3(0.05f, 0.05f, 0.05f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[3]->RenderMeshGeometry();

//Piramide principal blanco
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.035f, 0.0f, -0.05f));
model = glm::scale(model, glm::vec3(0.901f, 2.705f, 0.901f));
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()));
color = glm::vec3(0.00f, 1.00f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[3]->RenderMeshGeometry();

//***************Frente*****************

//Negro M derecho
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.568f, -0.4f, -0.057f));
model = glm::scale(model, glm::vec3(0.647f, 1.942f, 0.647f));
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()));
color = glm::vec3(0.05f, 0.05f, 0.05f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[3]->RenderMeshGeometry();

//Blanco M derecho
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.4f, -0.4f, -0.6507f));
model = glm::scale(model, glm::vec3(0.583f, 1.751f, 0.583f));
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()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[3]->RenderMeshGeometry();

//Negro M izquierdo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.4f, -0.4f, -0.6507f));
model = glm::scale(model, glm::vec3(0.647f, 1.942f, 0.647f));
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()));
color = glm::vec3(0.05f, 0.05f, 0.05f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[3]->RenderMeshGeometry();
```

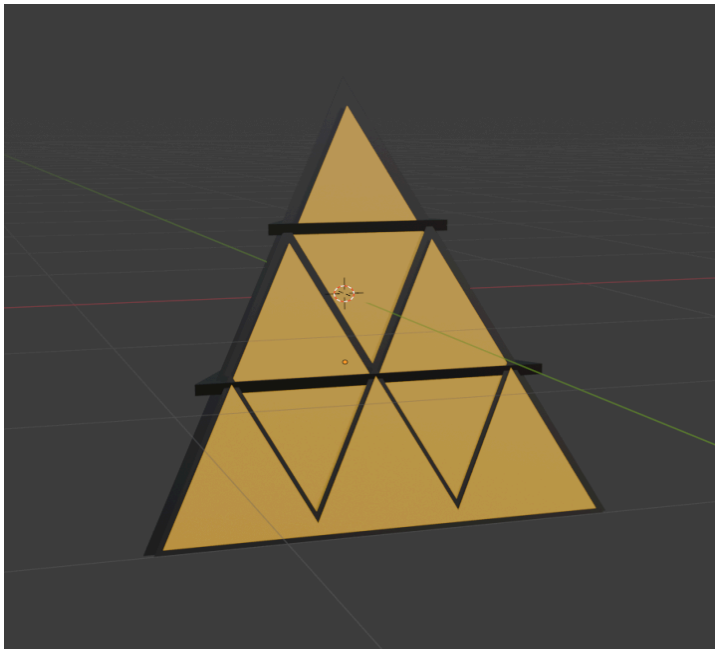
Ejemplo código por cada geometría.

```
CrearCubo();//índice 0 en MeshList
CrearPiramideTriangular();//índice 1 en MeshList
CrearCilindro(3, 1.0f);//índice 2 en MeshList
CrearCono(3, 2.0f);//índice 3 en MeshList
CrearPiramideCuadrangular();//índice 4 en MeshList
CreateShaders();
```

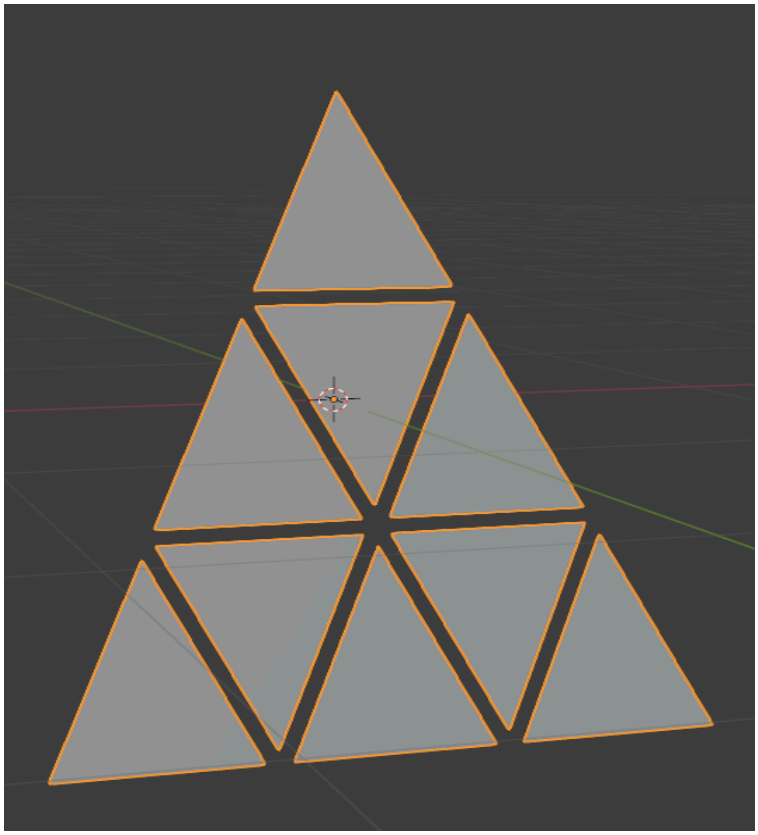
Cambio de resolución en cono y cilindro.



Pirámide en el programa.



Ejemplo hecho en Blender, emplea 11 pirámides por lado y la central



Dibujo 2d de cada triángulo en blender.

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.

Tuve varios, primero quería combinar lo que habíamos hecho con las letras para poder hacer una especie de texturizado 2d en cada cara de una pirámide simple, me tome mucho tiempo pero no logre implementarlo, después lo intente hacer en blender, copiar coordenadas de cada pirámide y ponerlas en el programa, el problema es que la pirámide está rotada, así que ya no me valieron estas coordenadas, ademas descubri quizá que el entorno 3D del programa base esta de cabeza, esto porque inserte 4 cubos ubicados en los 4 puntos cardinales, esto para ayudarme a orientarme y al menos tener una idea hacia que dirección modificar las coordenadas de cada pirámide, y al ver la relación del Norte y Oeste estaba del lado del Este, así que probablemente el sistema este de cabeza.

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
1. Bibliografía en formato APA

Como conclusión, creo que es de gran utilidad tener siempre un marco de referencia para modificar coordenadas, es por eso que me tomó mucho tiempo en organizar mis pirámides, y por lo tanto no termine.

Para hacer esta pirámide se requieren de 11 pirámides por lado, siendo un total de 34 contando la pirámide central.

Practicamos con la pista perspectiva, bastante útil para visualizar en 3D, ya no nos quedamos con una vista ortogonal fija.

LearnOpenGL. (2025). Camera. LearnOpenGL.

Recuperado de: https://learnopengl.com/Getting-started/Camera

Songho. (2024). OpenGL Cylinder, Prism & Pipe. Songho.ca.

Recuperado de: https://www.songho.ca/opengl/gl_cylinder.html

OpenGLTutorial.org. (2025). The Model, View and Projection matrices. OpenGL-Tutorial.

Recuperado de: https://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/