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: CARBAJAL REYES IRVIN JAIR

Nº de Cuenta: 422042084

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 5 DE MARZO DE 2025

CALIFICACIÓN: _____

Actividades

1. Generar una pirámide rubik (pyraminx) de 9 pirámides por cara.

   Cada cara de la pyraminx que se vea de un color diferente y que se vean las separaciones entre instancias (las líneas oscuras son las que permiten diferenciar cada pirámide pequeña)

   Agregar en su documento escrito las capturas de pantalla necesarias para que se vean las 4 caras de toda la

   pyraminx o un video en el cual muestra las 4 caras

Para la elaboración de este ejercicio fue necesario instanciar una piramide base

```
398         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
399         //otras transformaciones para el objeto
400         model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
401         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
402         model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
403         model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
404         model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
405         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
406         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
407         //se programe cambio entre proyección ortogonal y perspectiva
408         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
409         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
410         //color = glm::vec3(1.0f, 0.0f, 0.0f);
411         //glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
412         //meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Luego se hizo uso del MeshColorList, con el fin de poder generar una pirámide que contará con caras de colores diferentes. Es importante mencionar que se consideró el espacio que debe de existir para que los bordes sean notorios entre las caras de las pirámides

```
275 void PiramideCuadradaColor()
276 {
277     GLfloat vertices_piramide[] = {
278         //cara1 rojo
279         0.65f,-0.5f,0.5f,0.85f,0.0f,0.1f,
280         0.65f,-0.5f,-0.5f, 0.85f, 0.0f, 0.1f,
281         0.1f,0.6f,0.0f,0.85f,0.0f,0.1f,
282         //cara2 amarillo
283         -0.65f,-0.5f,0.5f,1.0f,0.9f,0.0f,
284         -0.65f,-0.5f,-0.5f,1.0f,0.9f,0.0f,
285         -0.1f,0.6f,0.0f,1.0f,0.9f,0.0f,
286         //cara3 verde
```

```
};
MeshColor* c = new MeshColor();
c->CreateMeshColor(vertices_piramide, 108);
meshColorList.push_back(c);
```

Por lo que fue necesario habilitar los dos shaders dentro del main

```
386         shaderList[0].useShader();
387         uniformModel = shaderList[0].getModelLocation();
388         uniformProjection = shaderList[0].getProjectLocation();
389         uniformView = shaderList[0].getViewLocation();
390         uniformColor = shaderList[0].getColorLocation();
391
392         shaderList[1].useShader();
393         uniformModel = shaderList[1].getModelLocation();
394         uniformProjection = shaderList[1].getProjectLocation();
```

Además para poder utilizar la cámara en la piramide de colores, también fue necesario agregar la función correspondiente en el shadercolor

```
 9   void main()
10   {
11       //gl_Position=projection*model*vec4(pos.x,pos.y,pos.z,1.0f);
12       gl_Position=projection*view*model*vec4(pos,1.0f);
13       vColor=vec4(color,1.0f);
14   }
```

Luego, se instanciar primero las pirámides que se usaban en más de una cara como la punta y las cuatro esquinas.

```
416         // R1
417         model = glm::mat4(1.0f);
418         color = glm::vec3(0.0f, 0.0f, 1.0f);
419         //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en
420         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
421         model = glm::translate(model, glm::vec3(0.0f, 0.9f, -4.0f));
422         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
423         model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
424         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PAR
425         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
426         meshColorList[0]->RenderMeshColor();
```

En las esquinas ya fue necesario trasladar las pirámides

```
//Esquinas
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pun
//model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(-0.9f, -1.0f, -3.05f));
//model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshColorList[0]->RenderMeshColor();

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pun
//model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(-0.9f, -1.0f, -4.95f));
//model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
```

Para colocar las demás pirámides sin rotar, se siguió un proceso similar en cada caso.

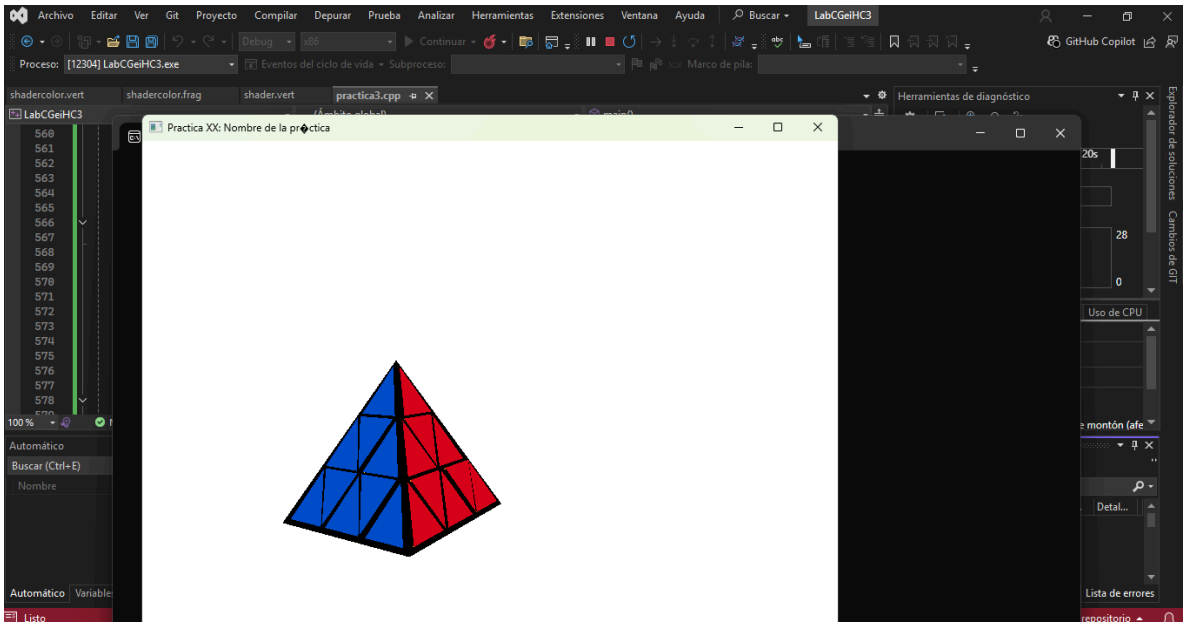
Al instanciar las pirámides que necesitan rotarse, una vez encontrado en ángulo correcto, se hacían ajustes en la traslación para ajustarlas en su debido lugar en cada cara.

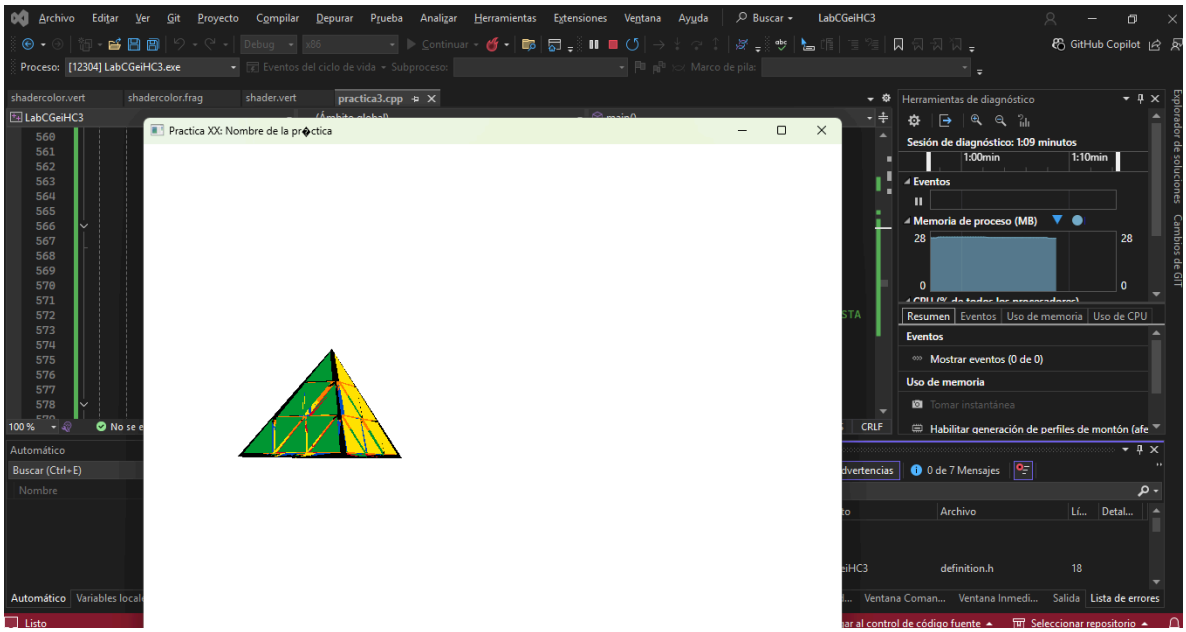
En algunos casos fue necesario rotar más de un eje

```
//Triangulos volteados R2
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los obje
//model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f, -0.2f, -3.4f));
model = glm::rotate(model, 125*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshColorList[0]->RenderMeshColor();
```

Una vez que se encontraba el ángulo para una cara, se usaba el mismo para su cara paralela pero negativo.

La ejecución queda de la siguiente manera:





Los problemas presentados durante la práctica fue determinar los vértices adecuados para la pirámide de colores, ya que había ocasiones en la que no se ajustaba correctamente a la pirámide base.

Otro problema fue encontrar el ángulo adecuado para las pirámides rotadas, ya que este es distinto para cada una de las caras, sin embargo se preserva para todos los triángulos de la misma.

Darle color a cada cara de la pirámide resultó ser un desafío, e implementar un MeshColor fue la solución más adecuada para la problemática.

Conclusiones

El desarrollo de esta práctica tuvo una complejidad elevada, ya que requirió de conocimientos adquiridos en prácticas anteriores como el uso de MeshColor, sin embargo, esto nos ayudó a reforzar esos conocimientos y cómo se pueden aplicar en entornos distintos a los vistos en sesiones pasadas. Me pareció un ejercicio desafiante, que pone en práctica el uso de proyecciones en 3D acompañado de la implementación de shaders, lo que lo convierte en una práctica muy completa pero con cierto grado de dificultad.

Bibliografía

- Joey de Vries. (s.f.). Model Loading - Mesh. LearnOpenGL. Recuperado el 5 de marzo de 2025, de https://learnopengl.com/Model-Loading/Mesh