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: Mino Guzmán Yara Amairani

Nº de Cuenta: 422017028

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 01 marzo 2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Actividades:

- 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 diferencias cada pirámide pequeña).

```
// Pirámide triangular regular
void CrearPiramideTriangular()
{
    unsigned int indices_piramide_triangular[] = {
            0,1,2,
            1,3,2,
            3,0,2,
            1,0,3

    };
    GLfloat vertices_piramide_triangular[] = {
        0.0f, 0.0f, 0.0f,       //0
        -0.5f, 0.0f, -0.866f,   //1
        0.5f, 0.0f, -0.866f,    //2
        0.0f,0.816f,-0.577f     //3
    };
    Mesh* obj1 = new Mesh();
    obj1->CreateMesh(vertices_piramide_triangular, indices_piramide_triangular, 12, 12);
    meshList.push_back(obj1);
}

void CreateShaders()
{
    Shader* shader1 = new Shader();
    shader1->CreateFromFiles(vShader, fShader);
    shaderList.push_back(*shader1);

    Shader* shader2 = new Shader();
    shader2->CreateFromFiles(vShaderColor, fShader);
    shaderList.push_back(*shader2);
}
```

Se utilizó la función void CrearPriamideTriangular() para poder crear el pyraminx.

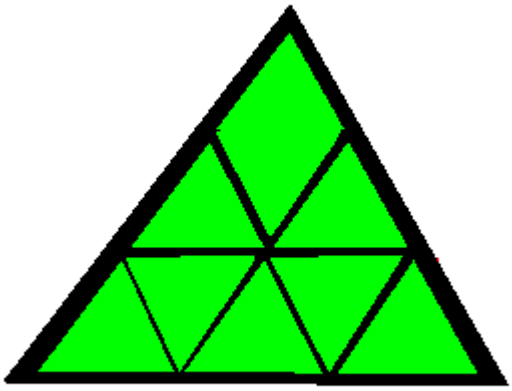
```
//Matriz global del pyraminx
glm::mat4 pyraminxModel = glm::mat4(1.0f);
pyraminxModel = glm::translate(pyraminxModel, glm::vec3(0.0f, 0.0f, -8.0f)); // Posición central del pyraminx
pyraminxModel = glm::rotate(pyraminxModel, glm::radians(rotaX), glm::vec3(1.0f, 0.0f, 0.0f));
pyraminxModel = glm::rotate(pyraminxModel, glm::radians(rotaY), glm::vec3(0.0f, 1.0f, 0.0f));
pyraminxModel = glm::rotate(pyraminxModel, glm::radians(rotaZ), glm::vec3(0.0f, 0.0f, 1.0f));
```

Cree una matriz global de mi pyraminx para que pudieran rotar en conjunto, ya que, si hacía que cada figura rotara, no parecía un Rubik.

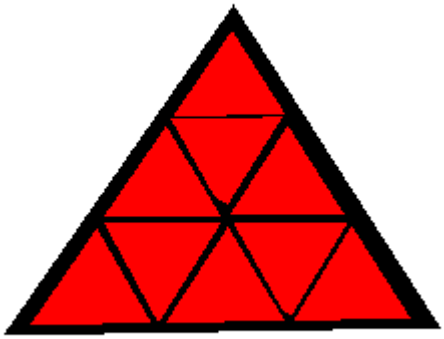
```
//---------------------------------- piramides verdes ----------------------------------------
//Abajo derecha
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::translate(model, glm::vec3(-0.3f, -1.75f, -5.6f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
model = pyraminxModel * model;
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));
meshList[0]->RenderMesh();

//Derecha abajo invertido
//Abajo derecha
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::translate(model, glm::vec3(-1.9f, -1.75f, -8.35f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
model = glm::rotate(model, glm::radians(60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(60.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(320.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(346.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = pyraminxModel * model;
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));
meshList[0]->RenderMesh();
```

Para cada cara, se crearon 9 mini pirámides, para las pirámides que no estaban invertidas, no fue necesario usar un rotare en x, y, z; en las pirámides invertidas si fue necesario para poder ir acomodando los ángulos de cada pirámide de acuerdo con la rotación que se generaba.

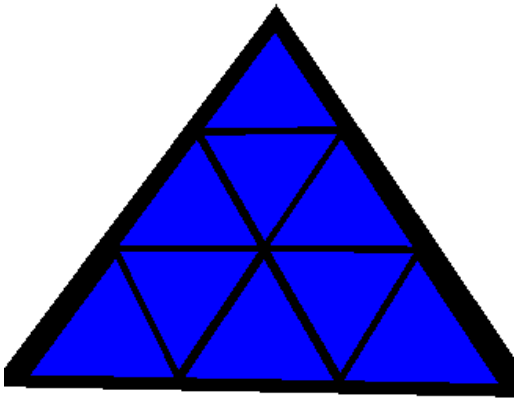
Ejecución:



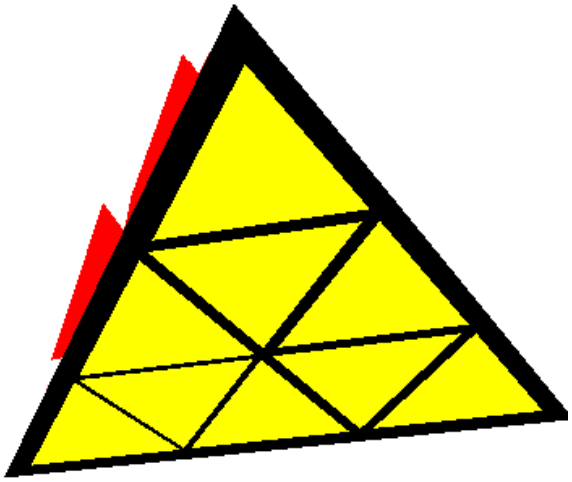
Cara color verde.



Cara color rojo.



Cara color azul.



Base de la pirámide en color amarillo.

2.- Problemas a la hora de hacer las actividades solicitadas:

- El problema presentado fue que, al momento de invertir mis triángulos en mi cara verde o roja, no quedaban parejos, si yo intentaba pegarlos, el triángulo se cortaba y ya no se veía la punta del triángulo, por lo que, por eso mismo, dichos triángulos quedaron un poco salidos.

3.- Conclusión:

a. Los ejercicios del reporte fueron adecuados para la realización de la práctica 3, ya que, a pesar de que me costó invertir mis triángulos, fue un reto nuevo para mí.
b. En conclusión, de las 3 prácticas que se han realizado, es la que más me ha costado realizar, pero también la que más me ayudó a explotar mis conocimientos de las materias de ciencias básicas para poder invertir mis triángulos al asignarles un ángulo de rotación.

Bibliografía

- Khronos Group. (2023). OpenGL 4.6 API Core Profile Specification. Recuperado el 1 de marzo de 2025, de https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf
- Bailey, M. (2024). Learn OpenGL: Extensive tutorial resource for learning Modern OpenGL. Recuperado el 1 de marzo de 2025, de https://learnopengl.com
- de Vries, J. (2024). Modern OpenGL Tutorials: Transformations and coordinate systems. Recuperado el 1 de marzo de 2025, de https://learnopengl.com/Getting-started/Transformations
- Abi-Chahla, F. (2023). Creating Hierarchical 3D Models with Modern OpenGL and GLM. Recuperado el 1 de marzo de 2025, de https://www.opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/
- Dalmau, D. S. C. (2024). Modelado geométrico y transformaciones jerárquicas con OpenGL y GLM. Revista Digital de Computación Gráfica, 18(2), 45-67. Recuperado el 1 de marzo de 2025, de https://www.revistas.unam.mx/index.php/computacion-grafica/article/view/58942