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: Roberto Aburto López

Nº de Cuenta: 319131996

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 06

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 6 de septiembre de 2025

CALIFICACIÓN: ________

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.

Al inicio, tuve un problema, la pirámide que nos habían dado en el codigo no era simétrica y era difícil de trabajar lo cual me hizo perder mucho tiempo. Entonces lo que hice fue cambiar la función que crea la pirámide para que dibujara una pirámide completamente simétrica.

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

    };
    GLfloat vertices_piramide_triangular[] = {
        /*
        -0.5f, -0.5f,0.0f,  //0
        0.5f,-0.5f,0.0f,    //1
        0.0f,0.5f, -0.25f,  //2
        0.0f,-0.5f,-0.5f,   //3
        */
        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);

}
```

Una vez que la pirámide estuvo simétrica, trabajar fue mucho más fácil. Como extra en la práctica y por aprendizaje personal, noté que al girar las figuras se veía mal y la pirámide se trababa, porque cada pieza giraba sobre su propio eje y eso la deformaba. Por eso hice que diera la impresión de ser un solo objeto. Para lograrlo, agregué un bloque que funciona como "control general" de toda la pirámide: primero coloca el conjunto un poco más lejos para verlo completo y luego aplica el mismo giro en las tres direcciones; así, todas las piezas se mueven y rotan juntas, sin trabarse ni deformarse.

```
glm::mat4 piramide = glm::mat4(1.0f);
piramide = glm::translate(piramide, glm::vec3(0.0f, 0.0f, -8.0f));
piramide = glm::rotate(piramide, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
piramide = glm::rotate(piramide, glm::radians(mainWindow.getrotax()), glm::vec3(0.0f, 1.0f, 0.0f));
piramide = glm::rotate(piramide, glm::radians(mainWindow.getrotax()), glm::vec3(0.0f, 0.0f, 1.0f));
```

Después creé la base: una pirámide negra en la que incrusté las demás pirámides, cada una con su color y cara correspondientes. Esta fase me costó muchísimo, sobre todo con las pirámides invertidas, porque no lograba hacerlas coincidir con la forma de la pirámide negra principal. Para conseguirlo, me apoyé en objetos reales de mi casa para visualizar cómo debían rotar las piezas hasta encajar. Con la práctica, fui automatizando este proceso.

A continuación, se muestra el código.

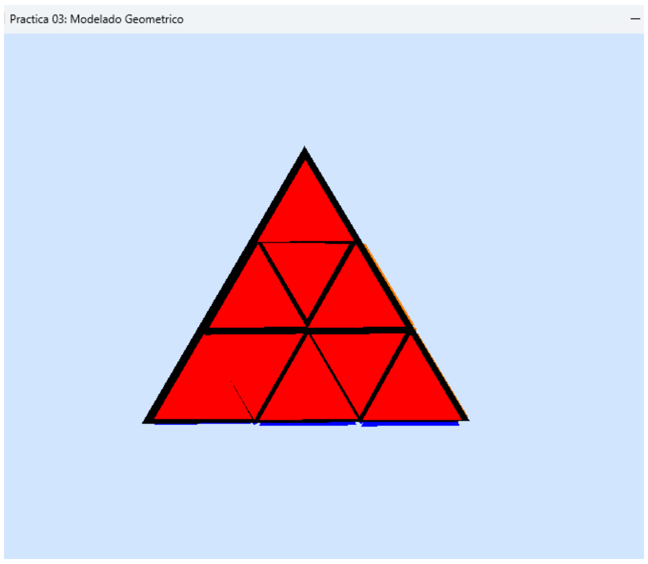
```
//Piramide principal (Negra)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.0f, -2.0f, -5.0f));
        model = glm::scale(model, glm::vec3(15.0f, 15.0f, 15.0f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * model;
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();


        //------------------------------ CARA VERDE ------------------------------------------------
-


        // 3 Triangulos parados (base)
        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-0.3f, -1.75f, -5.4f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-2.7f, -1.75f, -9.5f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-5.2f, -1.75f, -13.6f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 2 Triangulos rotados (base)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-5.1f, -1.75f, -13.5f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(55.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(56.0f), glm::vec3(0.0f, 0.0f, 1.0f));
        model = glm::rotate(model, glm::radians(316.0f), 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 = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-2.7f, -1.75f, -9.4f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(55.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(56.0f), glm::vec3(0.0f, 0.0f, 1.0f));
        model = glm::rotate(model, glm::radians(316.0f), 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 = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 2 triangulos parados (medio)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-2.9f, 2.1f, -12.2f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-0.3f, 2.1f, -8.1f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 1 triangulos invertido (medio)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-2.8f, 2.1f, -12.1f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(55.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(56.0f), glm::vec3(0.0f, 0.0f, 1.0f));
        model = glm::rotate(model, glm::radians(316.0f), 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 = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // Pico verde

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-0.5f, 6.0f, -10.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // ------------------------------------ CARA ROJA  ---------------------------------------
--


        // Pico Rojo
        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.3f, 6.0f, -10.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 3 Triangulos parados (base)
        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.35f, -1.75f, -5.5f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.7f, -1.75f, -9.7f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(5.0f, -1.75f, -13.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 2 Triangulos invertidos (base)
        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.6f, -1.75f, -9.5f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(45.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(15.0f), glm::vec3(0.0f, 0.0f, 1.0f));
        model = glm::rotate(model, glm::radians(28.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(11.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();


        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(4.9f, -1.75f, -13.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(45.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(15.0f), glm::vec3(0.0f, 0.0f, 1.0f));
        model = glm::rotate(model, glm::radians(28.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(11.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();


        // 2 Triangulo parado (medio)

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.3f, 2.2f, -8.3f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.5f, 2.2f, -12.3f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 1 triangulo medio invertida

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.5f, 2.2f, -12.5f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(45.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(15.0f), glm::vec3(0.0f, 0.0f, 1.0f));
        model = glm::rotate(model, glm::radians(28.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(11.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // ------------------------ CARA AZUL ------------------------------------------

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(4.9f, -2.1f, -13.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.15f, -2.1f, -5.5f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.6f, -2.1f, -9.7f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 2 triangulos (medio)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-2.5f, -2.1f, -9.7f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-5.0f, -2.1f, -13.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.0f, -2.1f, -13.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 1 Triangulos invertidos (medio)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.1f, -2.1f, -13.4f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // 2 Triangulos invertidos (base)

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.4f, -2.1f, -17.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 0.0f, 1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-2.5f, -2.1f, -17.8f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        // ----------------------------  Cara Naranja ------------------------------------------

        // 3 triangulos parados (base)
        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.5f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-5.0f, -1.75f, -14.2f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.5f, 0.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.0f, -1.75f, -14.2f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();

        model = glm::mat4(1.0f);
        color = glm::vec3(1.0f, 0.5f, 0.0f);
        model = glm::translate(model, glm::vec3(-0.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(5.0f, -1.75f, -14.2f));
        model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
        model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
        model = piramide * 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()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();
```

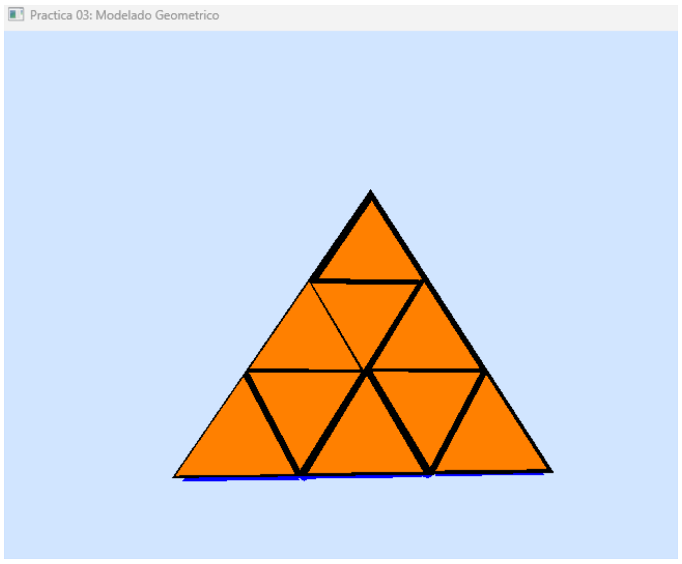
```
// 2 triangulo parado (medio)

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(2.5f, 2.2f, -12.8f));
model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
model = piramide * 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()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-2.5f, 2.2f, -12.8f));
model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
model = piramide * 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()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();

// Pico Naranja

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-0.2f, 6.0f, -11.4f));
model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
model = piramide * 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()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();

// 1 triangulos invertidos (medio)
```

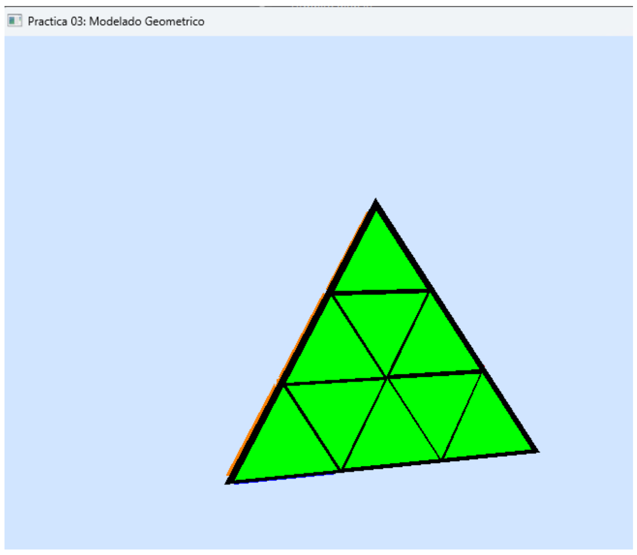
```
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.1f, 3.1f, -12.7f));
model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, glm::radians(43.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(0.0f), 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 = piramide * 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()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();

// 2 triangulos invertidos (base)

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-2.5f, -0.6f, -14.0f));
model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, glm::radians(43.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(0.0f), 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 = piramide * 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()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(2.5f, -0.6f, -14.0f));
model = glm::scale(model, glm::vec3(4.5f, 4.5f, 4.5f));
model = glm::rotate(model, glm::radians(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, glm::radians(43.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(0.0f), 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 = piramide * model;
```

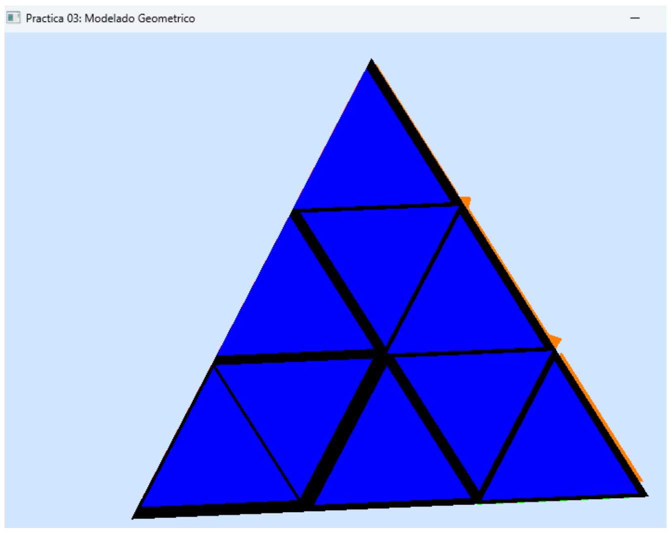
```
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
        glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        meshList[1]->RenderMesh();
```

Resultados:









Video, donde se muestra que gira como si fuera una sola pieza. Similar al video presentado en la práctica.

https://drive.google.com/file/d/1C6sEbK_T89Bbq5A8InLjPja0gLE30pCO/view?usp=drive_link

Conclusiones:

Fue un verdadero reto esta práctica, lo más pesado fue cuadrar las pirámides invertidas: estuve girando y probando muchas veces, hasta con cosas de mi casa para imaginar los giros. Ya con la práctica se me fue haciendo mecánico.