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 02

NOMBRE COMPLETO: Camacho Ignacio Violeta

Nº de Cuenta: 319061345

GRUPO DE LABORATORIO: 13

GRUPO DE TEORÍA: 06

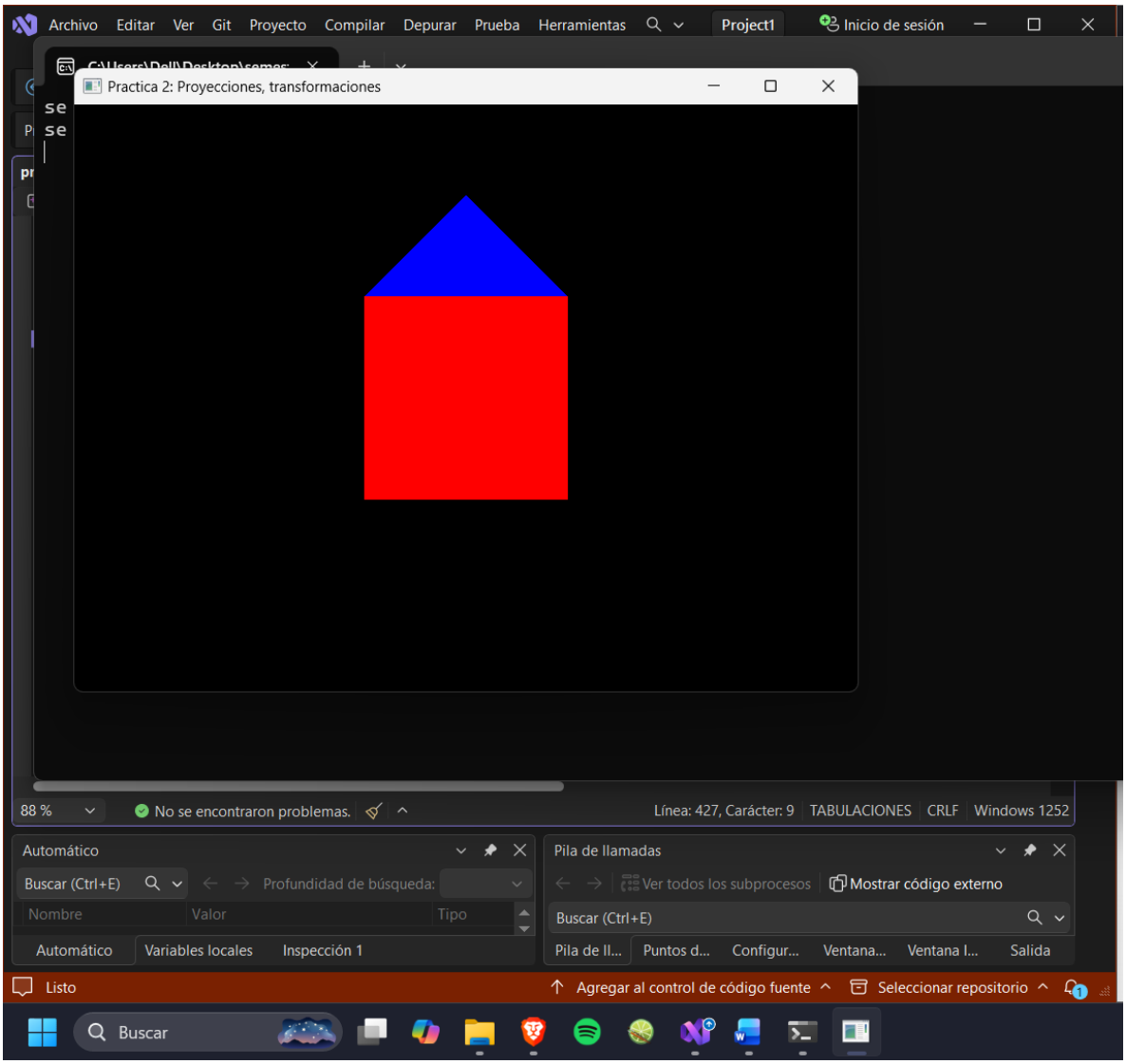
SEMESTRE 2026-2

FECHA DE ENTREGA LÍMITE: 26-02-2026

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.

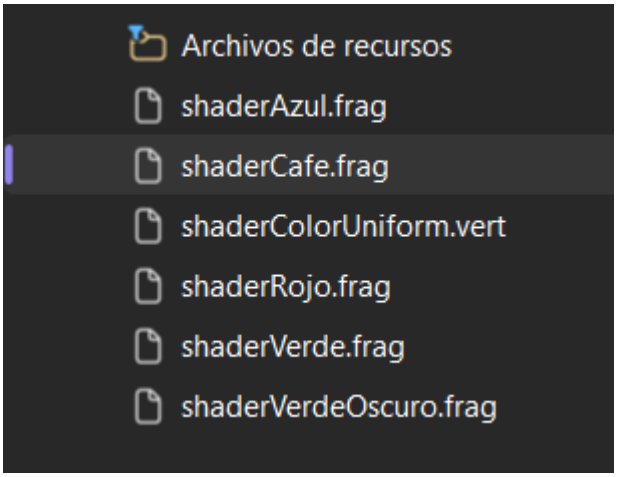


```cpp
void CrearPiramideBaseCuadrada()
{
    unsigned int indices[] = {
        // base
        0, 1, 2,
        2, 3, 0,

        // lados
        0, 1, 4,
        1, 2, 4,
        2, 3, 4,
        3, 0, 4
    };

    GLfloat vertices[] = {
        -0.5f, 0.0f,  0.5f,   // 0
         0.5f, 0.0f,  0.5f,   // 1
         0.5f, 0.0f, -0.5f,   // 2
        -0.5f, 0.0f, -0.5f,   // 3
         0.0f, 0.7f,  0.0f    // 4 (punta)
    };

    Mesh* piramide = new Mesh();
    piramide->CreateMesh(vertices, indices, 15, 18);
    meshList.push_back(piramide);
}
```

```cpp
    Shader* shaderRojo = new Shader(); // shader rojo
    shaderRojo->CreateFromFiles(
        "shaderColorUniform.vert",
        "shaderRojo.frag"
    );
    shaderList.push_back(*shaderRojo);

    Shader* shaderAzul = new Shader(); // shader azul
    shaderAzul->CreateFromFiles(
        "shaderColorUniform.vert",
        "shaderAzul.frag"
    );
    shaderList.push_back(*shaderAzul);

    Shader* shaderVerde = new Shader(); // shader Verde
    shaderVerde->CreateFromFiles(
        "shaderColorUniform.vert",
        "shaderVerde.frag"
    );
    shaderList.push_back(*shaderVerde);

    Shader* shaderVerdeOscuro = new Shader(); // shader Verde Oscuro
    shaderVerdeOscuro->CreateFromFiles(
        "shaderColorUniform.vert",
        "shaderVerdeOscuro.frag"
    );
    shaderList.push_back(*shaderVerdeOscuro);
```

```
279     CrearCubo();//índice 1 en MeshList
280     CrearPiramideBaseCuadrada();
281     CrearLetrasyFiguras(); //usa MeshColor, índices en MeshColorList
282     CreateShaders();
```

```
389     //cubo rojo para casita
390     shaderList[1].useShader();
391
392     uniformModel = shaderList[1].getModelLocation();
393     uniformProjection = shaderList[1].getProjectLocation();
394
395     model = glm::mat4(1.0f);
396
397     model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));// mover el cubo hacia atrás (profundidad)
398
399     //model = glm::rotate(model,glm::radians(angulo),glm::vec3(0.0f, 1.0f, 0.0f));// rotar sobre el eje Y (3D REAL)
400
401     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
402     glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
403
404     meshList[1]->RenderMesh();
405
406     //piramide azul techo
407     shaderList[2].useShader();
408
409     uniformModel = shaderList[2].getModelLocation();
410     uniformProjection = shaderList[2].getProjectLocation();
411
412     model = glm::mat4(1.0f);
413     model = glm::translate(model, glm::vec3(0.0f, 0.5f, -3.0f)); //moverla arriba del cubo
414     //model = glm::rotate(model, angulo, glm::vec3(0.0f, 1.0f, 0.0f));
415     //model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f)); //escalar
416     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
417     glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
418
419     // dibujar pirámide nueva
420     meshList[2]->RenderMesh();
```



Se implementaron diferentes funciones para crear figuras geométricas mediante el uso de vértices e índices:

Cubo (CrearCubo)

Se definieron 8 vértices y 36 índices para formar las 6 caras del cubo, cada una compuesta por dos triángulos.

El cubo se utiliza como la base de la casa en la escena 3D.

Pirámide triangular (CreaPiramide)

Se creó una pirámide simple usando 4 vértices y 4 caras triangulares, principalmente como ejercicio introductorio al uso de índices.

Pirámide de base cuadrada (CrearPiramideBaseCuadrada)

Esta figura se diseñó específicamente para funcionar como el techo de la casa.

Su base cuadrada tiene el mismo tamaño que el cubo, y cuenta con un vértice superior que forma las cuatro caras laterales triangulares.

Cada figura se almacena en un vector de tipo meshList, lo que permite reutilizarlas posteriormente durante el renderizado.

Un shader base para objetos 3D usando índices.

Shaders de color uniforme independientes (rojo, azul, verde, verde oscuro y café), cada uno con su propio fragment shader.

Un shader adicional para figuras 2D con color incluido en el VAO.

Esto permitió aplicar colores sólidos a los objetos sin depender del color por vértice.

Para posicionar los objetos en la escena se utilizaron matrices de modelo (model) con transformaciones como:

Traslación (glm::translate) para colocar los objetos dentro del volumen visible.

Rotación (glm::rotate) usando una variable angulo que se incrementa en cada frame para generar animación.

Escalado (glm::scale) para ajustar proporciones (por ejemplo, el techo respecto al cubo).

El cubo y la pirámide comparten la misma lógica de transformación para dar la sensación de un solo objeto 3D (una casa).

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

Durante el desarrollo de la práctica se presentaron los siguientes problemas:

Problema 1: La pirámide no tenía base cuadrada

Inicialmente, la pirámide creada era triangular, lo que no permitía usarla como techo del cubo.

Solución:

Se rediseñó la pirámide definiendo cuatro vértices para la base y un vértice superior, junto con los índices correctos para formar una pirámide de base cuadrada.

Problema 2: Diferente velocidad de rotación entre el cubo y la pirámide

La pirámide parecía girar más rápido que el cubo.

Solución:

Se corrigió el manejo de la variable angulo, asegurando que se incremente una sola vez por frame y que ambos objetos usen el mismo valor para la rotación.

Problema 3: El objeto no se mostraba al usar shaders de color uniforme

En algunos casos el objeto no aparecía en pantalla.

Solución:

Se verificó el orden y el índice correcto de los shaders dentro del vector shaderList, asegurando que cada objeto use el shader adecuado.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

Los ejercicios presentaron una dificultad media-alta, ya que requirieron comprender cómo interactúan los shaders, las mallas, los índices y las transformaciones geométricas dentro del pipeline gráfico.

El manejo correcto de matrices de transformación fue clave para lograr una escena coherente en 3D.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

Sería útil que la práctica incluyera:

Explicaciones más detalladas sobre el orden correcto de las transformaciones.

Ejemplos guiados paso a paso para el manejo de shaders personalizados.

Más tiempo dedicado a depurar errores comunes al trabajar con índices y matrices.

c. Conclusión

Esta práctica permitió reforzar conceptos fundamentales de gráficos por computadora, como el uso de mallas indexadas, proyecciones en perspectiva, shaders personalizados y transformaciones geométricas, logrando finalmente la construcción de una escena 3D funcional y animada.

1. Bibliografía en formato APA

Khronos Group. (n.d.). OpenGL 4 Reference Pages. Recuperado de https://www.khronos.org/registry/OpenGL-Refpages/gl4/

Khronos Group. (n.d.). OpenGL Shading Language (GLSL) Specification. Recuperado de https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.60.pdf

The Khronos Group Inc. (n.d.). OpenGL Programming Guide (The Red Book) — OpenGL 4.x. Recuperado de https://www.khronos.org/opengl/wiki/Getting_Started

LearnOpenGL. (n.d.). LearnOpenGL – Getting started with OpenGL. Recuperado de https://learnopengl.com

GLFW. (n.d.). GLFW – An OpenGL library for creating windows and contexts. Recuperado de https://www.glfw.org

OpenGL Mathematics (GLM). (n.d.). GLM Manual. Recuperado de https://glm.g-truc.net/0.9.9/index.html

OpenGL Tutorial. (n.d.). OpenGL Transformations Tutorial. Recuperado de https://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). OpenGL Insights (Shaders). Recuperado de https://www.oreilly.com/library/view/opengl-insights/