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: José Santiago Sánchez Medina

Nº de Cuenta: 319246881

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 31 de agosto del 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.**

a) Dibujar las iniciales de sus nombres, cada letra de un color diferente

Modificaciones al código

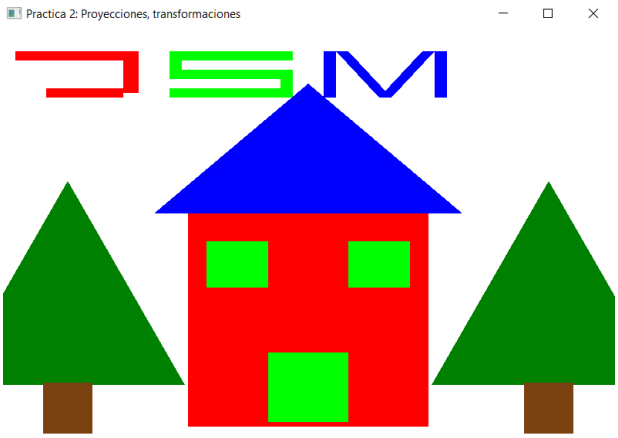
```
void CrearLetrasyFiguras()
{
    GLfloat vertices_letras[] = {
            -0.95f, 0.86f, 0.5f,  1.0f,0.0f,0.0f,   -0.55f, 0.86f, 0.5f,  1.0f,0.0f,0.0f,   -0.55f, 0.90f, 0.5f,  1.0f,0.0f,0.0f,
            -0.95f, 0.86f, 0.5f,  1.0f,0.0f,0.0f,   -0.55f, 0.90f, 0.5f,  1.0f,0.0f,0.0f,   -0.95f, 0.90f, 0.5f,  1.0f,0.0f,0.0f,
            -0.60f, 0.72f, 0.5f,  1.0f,0.0f,0.0f,   -0.55f, 0.72f, 0.5f,  1.0f,0.0f,0.0f,   -0.55f, 0.90f, 0.5f,  1.0f,0.0f,0.0f,
            -0.60f, 0.72f, 0.5f,  1.0f,0.0f,0.0f,   -0.55f, 0.90f, 0.5f,  1.0f,0.0f,0.0f,   -0.60f, 0.90f, 0.5f,  1.0f,0.0f,0.0f,
            -0.85f, 0.70f, 0.5f,  1.0f,0.0f,0.0f,   -0.60f, 0.70f, 0.5f,  1.0f,0.0f,0.0f,   -0.60f, 0.74f, 0.5f,  1.0f,0.0f,0.0f,
            -0.85f, 0.70f, 0.5f,  1.0f,0.0f,0.0f,   -0.60f, 0.74f, 0.5f,  1.0f,0.0f,0.0f,   -0.85f, 0.74f, 0.5f,  1.0f,0.0f,0.0f,

            //S (verde: 0,1,0)
            -0.45f, 0.86f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.86f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.90f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.86f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.90f, 0.5f,  0.0f,1.0f,0.0f,   -0.45f, 0.90f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.78f, 0.5f,  0.0f,1.0f,0.0f,   -0.41f, 0.78f, 0.5f,  0.0f,1.0f,0.0f,   -0.41f, 0.90f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.78f, 0.5f,  0.0f,1.0f,0.0f,   -0.41f, 0.90f, 0.5f,  0.0f,1.0f,0.0f,   -0.45f, 0.90f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.78f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.78f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.82f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.78f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.82f, 0.5f,  0.0f,1.0f,0.0f,   -0.45f, 0.82f, 0.5f,  0.0f,1.0f,0.0f,
            -0.09f, 0.70f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.70f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.82f, 0.5f,  0.0f,1.0f,0.0f,
            -0.09f, 0.70f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.82f, 0.5f,  0.0f,1.0f,0.0f,   -0.09f, 0.82f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.70f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.70f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.74f, 0.5f,  0.0f,1.0f,0.0f,
            -0.45f, 0.70f, 0.5f,  0.0f,1.0f,0.0f,   -0.05f, 0.74f, 0.5f,  0.0f,1.0f,0.0f,   -0.45f, 0.74f, 0.5f,  0.0f,1.0f,0.0f,

            //M(azul: 0,0,1)
            0.05f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.09f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.09f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,
            0.05f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.09f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.05f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,
            0.41f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.45f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.45f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,
            0.41f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.45f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.41f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,
            0.09f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.13f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.27f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,
            0.09f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.27f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.23f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,
            0.37f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.41f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.27f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,
            0.37f, 0.90f, 0.5f,   0.0f,0.0f,1.0f,   0.27f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,   0.23f, 0.70f, 0.5f,   0.0f,0.0f,1.0f,
    };
```

```
while (!mainWindow.getShouldClose())
{
    glfwPollEvents();
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    shaderList[1].useShader();
    uniformModel = shaderList[1].getModelLocation();
    uniformProjection = shaderList[1].getProjectLocation();

    //LETRAS J S M
    model = glm::mat4(1.0f);
    model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.90f));
    model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
    meshColorList[0]->RenderMeshColor();
```

Para poder dibujar mis iniciales en la ventana, lo primero que hice fue definir los vértices de cada letra dentro del arreglo vertices_letras. Ahí puse las coordenadas de los puntos que forman las letras J, S y M, y además agregué el color para cada una. La J la hice de color rojo, la S de color verde y la M de color azul. Así, cada letra queda diferenciada con un color distinto.

Después, en la parte del programa donde ya estaba creando las figuras con colores (CrearLetrasyFiguras()), agregué ese arreglo de vértices y lo pasé a CreateMeshColor, que es la función que se encarga de generar los objetos con colores usando los datos de posición y de color de cada vértice.

Ejecución únicamente del ejercicio 1, aun usando el código del ejercicio en clase sin modificar para el ejercicio 2.



b) Generar el dibujo de la casa de la clase, pero en lugar de instanciar triángulos y cuadrados será instanciando piramides y cubos, para esto se requiere crear shaders diferentes de los colores: rojo, verde, azul, café y verde oscuro en lugar de usar el shader con el color clamp

Modificaciones al código

```
// shaders nuevos: colores
static const char* fRed = "shaders/red.frag";
static const char* fGreen = "shaders/green.frag";
static const char* fBlue = "shaders/blue.frag";
static const char* fBrown = "shaders/brown.frag";
static const char* fDarkGreen = "shaders/darkgreen.frag";
```

```
//Sahders nuevos

Shader* sRed = new Shader();
sRed->CreateFromFiles(vShader, fRed);
shaderList.push_back(*sRed);

Shader* sGreen = new Shader();
sGreen->CreateFromFiles(vShader, fGreen);
shaderList.push_back(*sGreen);

Shader* sBlue = new Shader();
sBlue->CreateFromFiles(vShader, fBlue);
shaderList.push_back(*sBlue);

Shader* sBrown = new Shader();
sBrown->CreateFromFiles(vShader, fBrown);
shaderList.push_back(*sBrown);

Shader* sDarkGreen = new Shader();
sDarkGreen->CreateFromFiles(vShader, fDarkGreen);
shaderList.push_back(*sDarkGreen);
```

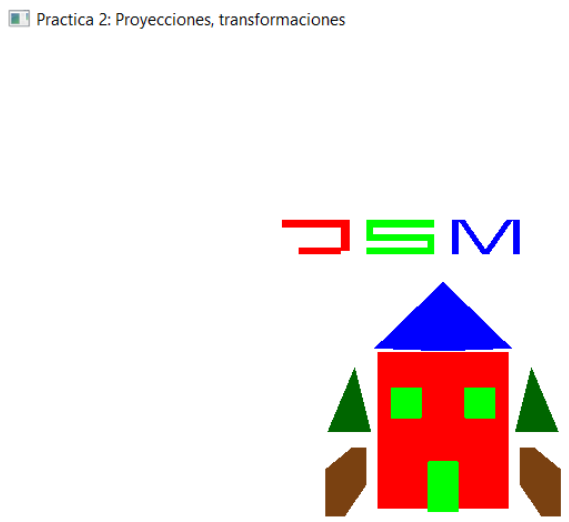
```
while (!mainWindow.getShouldClose())
{
    glfwPollEvents();
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    shaderList[1].useShader();
    uniformModel = shaderList[1].getModelLocation();
    uniformProjection = shaderList[1].getProjectLocation();

    //LETRAS J S M
    model = glm::mat4(1.0f);
    model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.90f));
    model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
    meshColorList[0]->RenderMeshColor();

    //CASA CON CUBOS Y PIRÁMIDES

    //Cuerpo de la casa (cubo rojo)
    shaderList[2].useShader(); // rojo
    uniformModel = shaderList[2].getModelLocation();
    uniformProjection = shaderList[2].getProjectLocation();
    model = glm::mat4(1.0f);
    model = glm::translate(model, glm::vec3(0.0f, -0.35f, -4.00f));
    model = glm::scale(model, glm::vec3(0.80f, 0.95f, 1.00f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
    meshList[1]->RenderMesh(); // cubo
```

```
298         // Techo (pirámide azul)
299         shaderList[4].useShader(); // azul
300         uniformModel = shaderList[4].getModelLocation();
301         uniformProjection = shaderList[4].getProjectLocation();
302         model = glm::mat4(1.0f);
303         model = glm::translate(model, glm::vec3(0.0f, 0.42f, -3.98f));
304         model = glm::scale(model, glm::vec3(0.95f, 0.50f, 1.00f));
305         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
306         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
307         meshList[0]->RenderMesh(); // pirámide
308
309         //Puerta (cubo verde)
310         shaderList[3].useShader(); // verde
311         uniformModel = shaderList[3].getModelLocation();
312         uniformProjection = shaderList[3].getProjectLocation();
313         model = glm::mat4(1.0f);
314         model = glm::translate(model, glm::vec3(0.0f, -0.68f, -3.96f));
315         model = glm::scale(model, glm::vec3(0.18f, 0.30f, 1.00f));
316         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
317         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
318         meshList[1]->RenderMesh(); // cubo
319
320         //Ventana izquierda (cubo verde)
321         shaderList[3].useShader();
322         uniformModel = shaderList[3].getModelLocation();
323         uniformProjection = shaderList[3].getProjectLocation();
324         model = glm::mat4(1.0f);
325         model = glm::translate(model, glm::vec3(-0.22f, -0.18f, -3.96f));
326         model = glm::scale(model, glm::vec3(0.18f, 0.18f, 1.00f));
327         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
328         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
329         meshList[1]->RenderMesh();
```

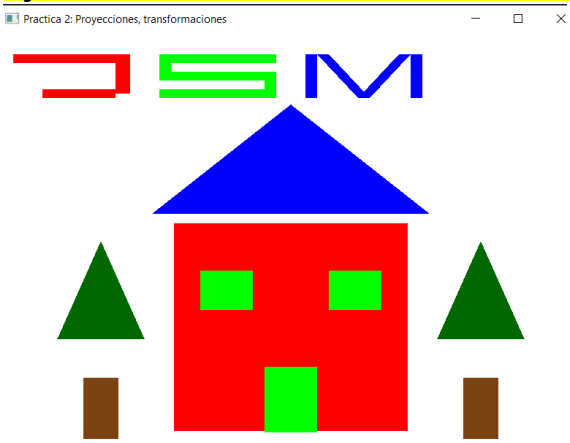
```
//Ventana derecha (cubo verde)
shaderList[3].useShader();
uniformModel = shaderList[3].getModelLocation();
uniformProjection = shaderList[3].getProjectLocation();
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.22f, -0.18f, -3.96f));
model = glm::scale(model, glm::vec3(0.18f, 0.18f, 1.00f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh();

//Árbol izquierdo

// Tronco (cubo café)
shaderList[5].useShader(); // café
uniformModel = shaderList[5].getModelLocation();
uniformProjection = shaderList[5].getProjectLocation();
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-0.65f, -0.72f, -3.97f));
model = glm::scale(model, glm::vec3(0.12f, 0.28f, 1.00f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh();

// Copa(pirámide verde oscuro)
shaderList[6].useShader(); // verde oscuro
uniformModel = shaderList[6].getModelLocation();
uniformProjection = shaderList[6].getProjectLocation();
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-0.65f, -0.18f, -4.00f));
model = glm::scale(model, glm::vec3(0.30f, 0.45f, 1.00f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh();
```

```
//Árbol derecho

shaderList[5].useShader(); // tronco café
uniformModel = shaderList[5].getModelLocation();
uniformProjection = shaderList[5].getProjectLocation();
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.65f, -0.72f, -3.97f));
model = glm::scale(model, glm::vec3(0.12f, 0.28f, 1.00f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh();

shaderList[6].useShader(); // copa verde oscuro
uniformModel = shaderList[6].getModelLocation();
uniformProjection = shaderList[6].getProjectLocation();
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.65f, -0.18f, -4.00f));
model = glm::scale(model, glm::vec3(0.30f, 0.45f, 1.00f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh();

glUseProgram(0);
mainWindow.swapBuffers();
```

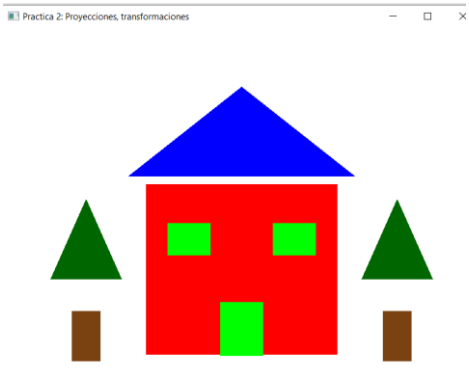
Ejecución del programa vista perspectiva



Ejecución vista ortogonal



Ejecución únicamente casa



Para la casita lo que hice fue usar cubos y pirámides en lugar de cuadrados y triángulos. El cuerpo de la casa lo armé con un cubo rojo, el techo con una pirámide

azul más grande y centrada arriba, la puerta y las ventanas con cubos verdes, y los árboles los formé con un cubo café para el tronco y una pirámide verde oscuro para la copa.

Para darle color a cada parte creé varios shaders de fragmento, cada uno con un color fijo: rojo, verde, azul, café y verde oscuro. Así, al momento de dibujar, simplemente seleccioné el shader correspondiente y la figura se pintaba con su color.

Lo que más me costó fue ajustar las coordenadas y las escalas para que las piezas quedaran bien colocadas y proporcionadas. Tuve que ir probando hasta que el techo se viera del tamaño correcto, que la puerta quedara centrada, que las ventanas estuvieran a los lados y que los árboles tuvieran el tronco debajo de la copa.

Al final logré que la casita se viera completa, con todos sus elementos y en los colores correctos, usando cubos y pirámides junto con transformaciones de traslación y escala.

2. **Conclusión:**

    a. **Los ejercicios del reporte: Complejidad, Explicación.**

En general, la parte de la casita me pareció entretenida y no tan complicada de implementar, aunque sí me costó un poco acomodar las figuras para que todas quedaran proporcionadas. Lo más difícil fue pensar bien las traslaciones y escalas para que el techo no se viera desfasado y que la puerta y las ventanas quedaran en su lugar. Una vez que entendí cómo aplicar las transformaciones, todo fluyó mejor y logré que cada parte tomara la forma que buscaba. También me gustó mucho crear los shaders de colores fijos, porque con eso pude controlar de manera sencilla qué color usar en cada figura y darle un mejor aspecto a la escena.

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

Todo bien, me gustó la práctica y no tuve mayores problemas.

    c. Conclusión

Me agradó desarrollar esta práctica porque me permitió aplicar lo aprendido de OpenGL de una manera más creativa. Aunque al principio se me complicó acomodar bien las figuras y definir las transformaciones, logré resolverlo y quedé satisfecho con el resultado. Considero que fue un buen avance porque combiné el uso de cubos y pirámides con shaders de colores fijos, lo cual hizo que la casita y los árboles se vieran mucho más completos y visualmente atractivos.

3. **Bibliografía en formato APA**