Universidad Nacional

Autónoma de México

Practica No. 2

Alumno: Arroyo Llanes Miguel Alejandro.

No. Cuenta: 319218460

Materia: Lab. Computación Grafica e Interacción Humano-Computadora

Grupo Laboratorio: 3

Grupo teoría: 6

Fecha de Entrega: 22/02/2025

Nombre profesor: Ing. Jose Roque Roman Guadarrama

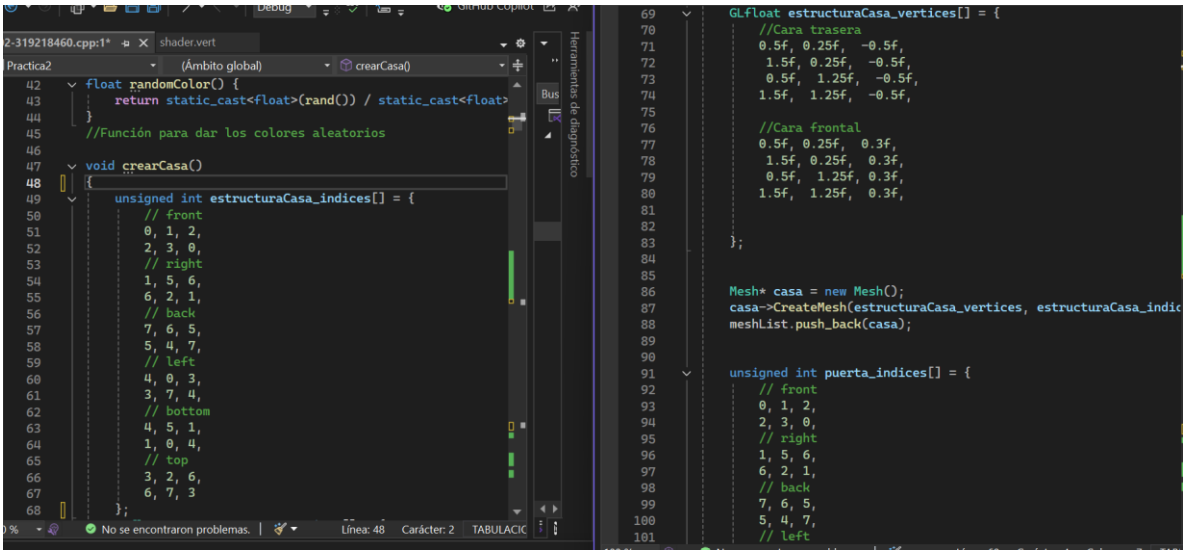
Comentario:

Primeramente, se retomo el dibujado a mano a través de una Tablet, sin embargo, esta vez aquello solo sirvió para las letras, para la casa se tuvo que realizar una investigación para entender los modelos 3d de una forma más correcta a lo que se buscaba realizar en esta práctica, una vez entendido se tuvo que modificar todos los archivos de shader para la asignación de colores de una forma controlada dejando comentarios para poder ubicar bien cada bloque de código.

Además, para no tener que editar más allá de los colores a las letras se utilizo la función de traslación aprendida en el ejercicio anterior para así solo mover un poco abajo a la izquierda y que no se encimara con la casa.

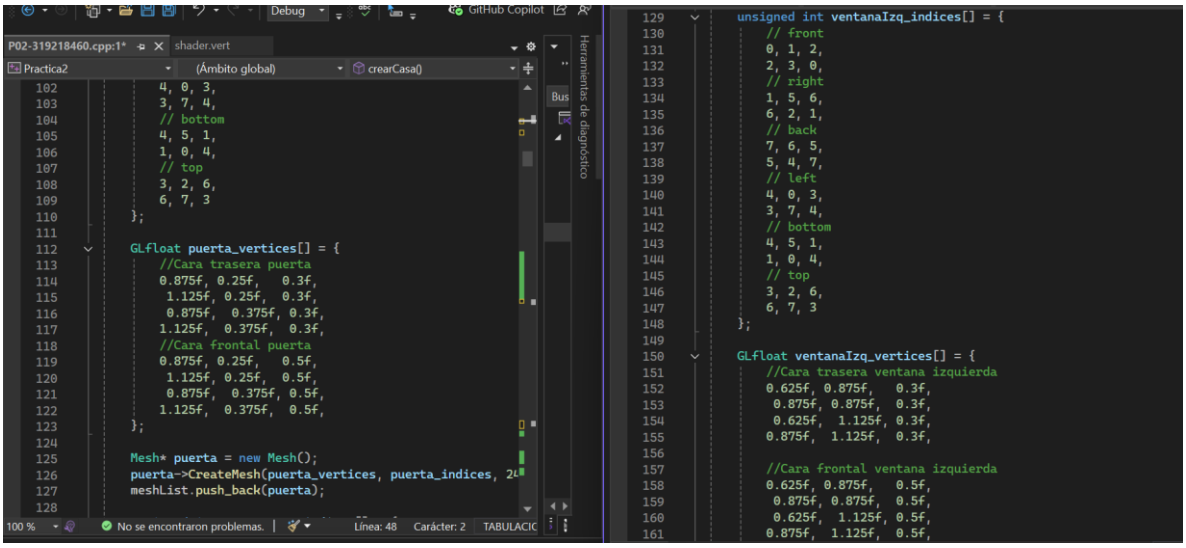
Bloque de código:

Líneas de código que muestran la función random para las letras y el dibujado de la casa.



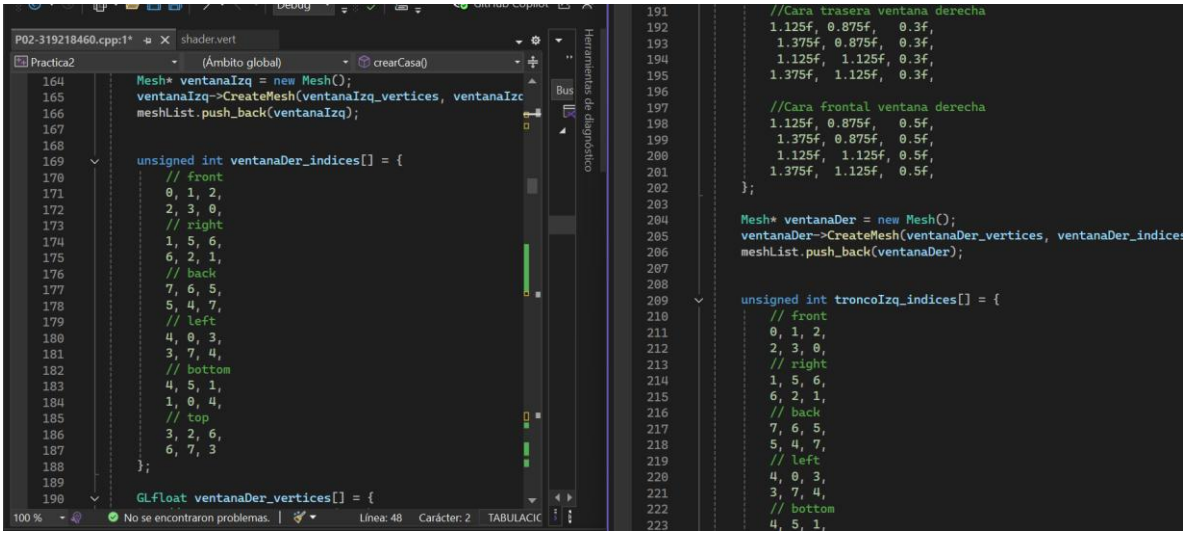
```
float randomColor() {
    return static_cast<float>(rand()) / static_cast<float>
}
//Función para dar los colores aleatorios

void crearCasa()
{
    unsigned int estructuraCasa_indices[] = {
        // front
        0, 1, 2,
        2, 3, 0,
        // right
        1, 5, 6,
        6, 2, 1,
        // back
        7, 6, 5,
        5, 4, 7,
        // left
        4, 0, 3,
        3, 7, 4,
        // bottom
        4, 5, 1,
        1, 0, 4,
        // top
        3, 2, 6,
        6, 7, 3
    };
```

```
GLfloat estructuraCasa_vertices[] = {
    //Cara trasera
    0.5f, 0.25f,  -0.5f,
     1.5f, 0.25f,  -0.5f,
     0.5f,  1.25f,  -0.5f,
    1.5f,  1.25f,  -0.5f,

    //Cara frontal
    0.5f, 0.25f,  0.3f,
     1.5f, 0.25f,  0.3f,
     0.5f,  1.25f, 0.3f,
    1.5f,  1.25f,  0.3f,


};


Mesh* casa = new Mesh();
casa->CreateMesh(estructuraCasa_vertices, estructuraCasa_indic
meshList.push_back(casa);


unsigned int puerta_indices[] = {
    // front
    0, 1, 2,
    2, 3, 0,
    // right
    1, 5, 6,
    6, 2, 1,
    // back
    7, 6, 5,
    5, 4, 7,
    // left
```



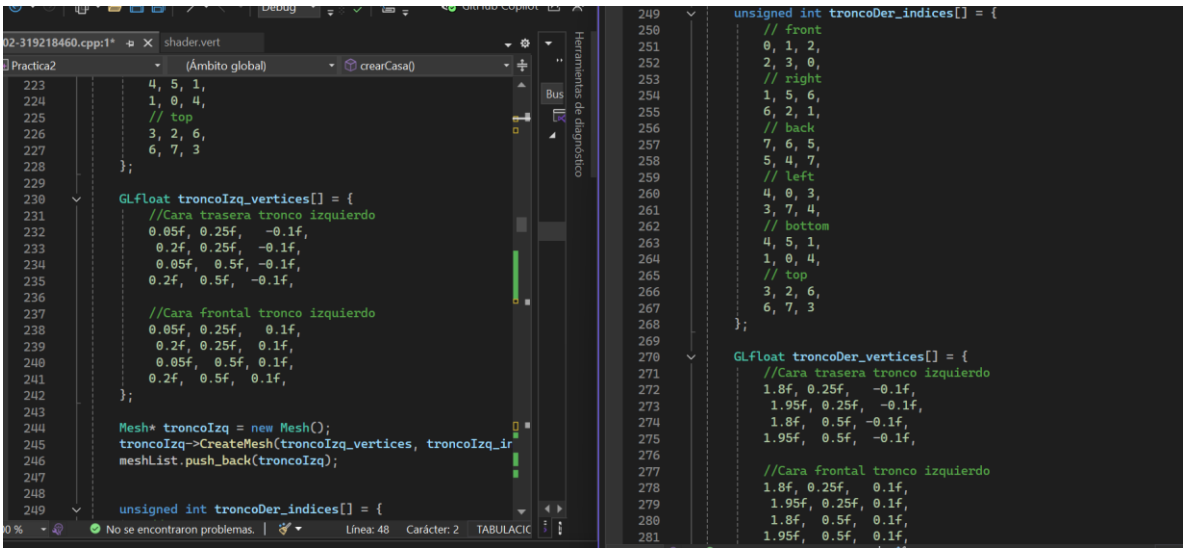
```
    4, 0, 3,
    3, 7, 4,
    // bottom
    4, 5, 1,
    1, 0, 4,
    // top
    3, 2, 6,
    6, 7, 3
};

GLfloat puerta_vertices[] = {
    //Cara trasera puerta
    0.875f, 0.25f,   0.3f,
     1.125f, 0.25f,  0.3f,
     0.875f,  0.375f, 0.3f,
    1.125f,  0.375f,  0.3f,
    //Cara frontal puerta
    0.875f, 0.25f,   0.5f,
     1.125f, 0.25f,  0.5f,
     0.875f,  0.375f, 0.5f,
    1.125f,  0.375f,  0.5f,
};

Mesh* puerta = new Mesh();
puerta->CreateMesh(puerta_vertices, puerta_indices, 2
meshList.push_back(puerta);
```

```
unsigned int ventanaIzq_indices[] = {
    // front
    0, 1, 2,
    2, 3, 0,
    // right
    1, 5, 6,
    6, 2, 1,
    // back
    7, 6, 5,
    5, 4, 7,
    // left
    4, 0, 3,
    3, 7, 4,
    // bottom
    4, 5, 1,
    1, 0, 4,
    // top
    3, 2, 6,
    6, 7, 3
};

GLfloat ventanaIzq_vertices[] = {
    //Cara trasera ventana izquierda
    0.625f, 0.875f,   0.3f,
     0.875f, 0.875f,  0.3f,
     0.625f,  1.125f, 0.3f,
    0.875f,  1.125f,  0.3f,

    //Cara frontal ventana izquierda
    0.625f, 0.875f,   0.5f,
     0.875f, 0.875f,  0.5f,
     0.625f,  1.125f, 0.5f,
    0.875f,  1.125f,  0.5f,
```



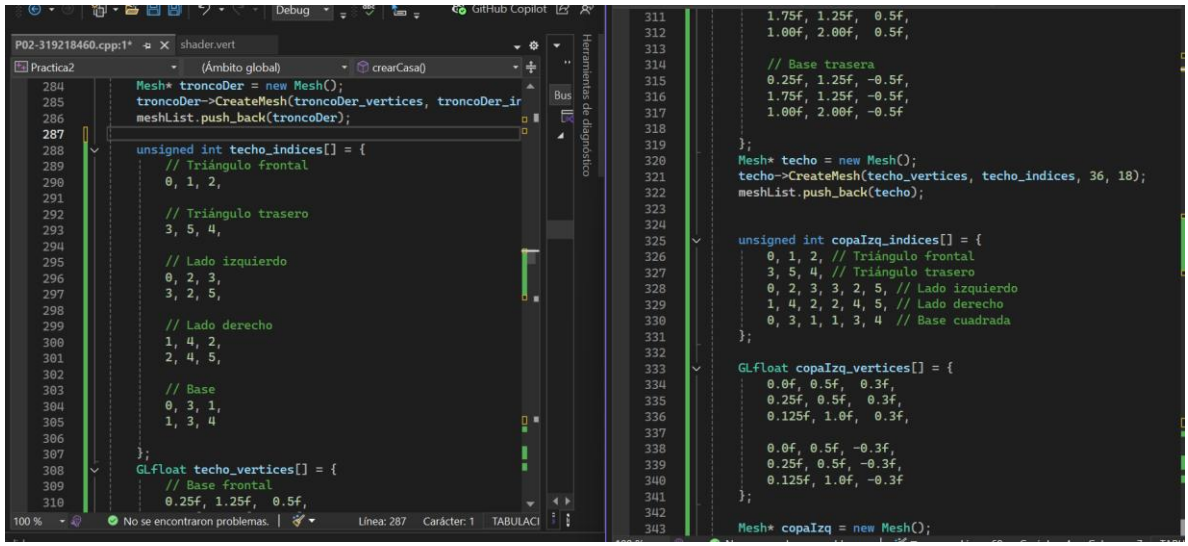
```
Mesh* ventanaIzq = new Mesh();
ventanaIzq->CreateMesh(ventanaIzq_vertices, ventanaIzc
meshList.push_back(ventanaIzq);


unsigned int ventanaDer_indices[] = {
    // front
    0, 1, 2,
    2, 3, 0,
    // right
    1, 5, 6,
    6, 2, 1,
    // back
    7, 6, 5,
    5, 4, 7,
    // left
    4, 0, 3,
    3, 7, 4,
    // bottom
    4, 5, 1,
    1, 0, 4,
    // top
    3, 2, 6,
    6, 7, 3
};

GLfloat ventanaDer_vertices[] = {
```

```
    //Cara trasera ventana derecha
    1.125f, 0.875f,   0.3f,
     1.375f, 0.875f,  0.3f,
     1.125f,  1.125f, 0.3f,
    1.375f,  1.125f,  0.3f,

    //Cara frontal ventana derecha
    1.125f, 0.875f,   0.5f,
     1.375f, 0.875f,  0.5f,
     1.125f,  1.125f, 0.5f,
    1.375f,  1.125f,  0.5f,
};

Mesh* ventanaDer = new Mesh();
ventanaDer->CreateMesh(ventanaDer_vertices, ventanaDer_indices
meshList.push_back(ventanaDer);


unsigned int troncoIzq_indices[] = {
    // front
    0, 1, 2,
    2, 3, 0,
    // right
    1, 5, 6,
    6, 2, 1,
    // back
    7, 6, 5,
    5, 4, 7,
    // left
    4, 0, 3,
    3, 7, 4,
    // bottom
    4, 5, 1,
```



```
    4, 5, 1,
    1, 0, 4,
    // top
    3, 2, 6,
    6, 7, 3
};

GLfloat troncoIzq_vertices[] = {
    //Cara trasera tronco izquierdo
    0.05f, 0.25f,   -0.1f,
     0.2f, 0.25f,  -0.1f,
     0.05f,  0.5f, -0.1f,
    0.2f,  0.5f,  -0.1f,

    //Cara frontal tronco izquierdo
    0.05f, 0.25f,   0.1f,
     0.2f, 0.25f,  0.1f,
     0.05f,  0.5f, 0.1f,
    0.2f,  0.5f,  0.1f,
};

Mesh* troncoIzq = new Mesh();
troncoIzq->CreateMesh(troncoIzq_vertices, troncoIzq_ir
meshList.push_back(troncoIzq);


unsigned int troncoDer_indices[] = {
```

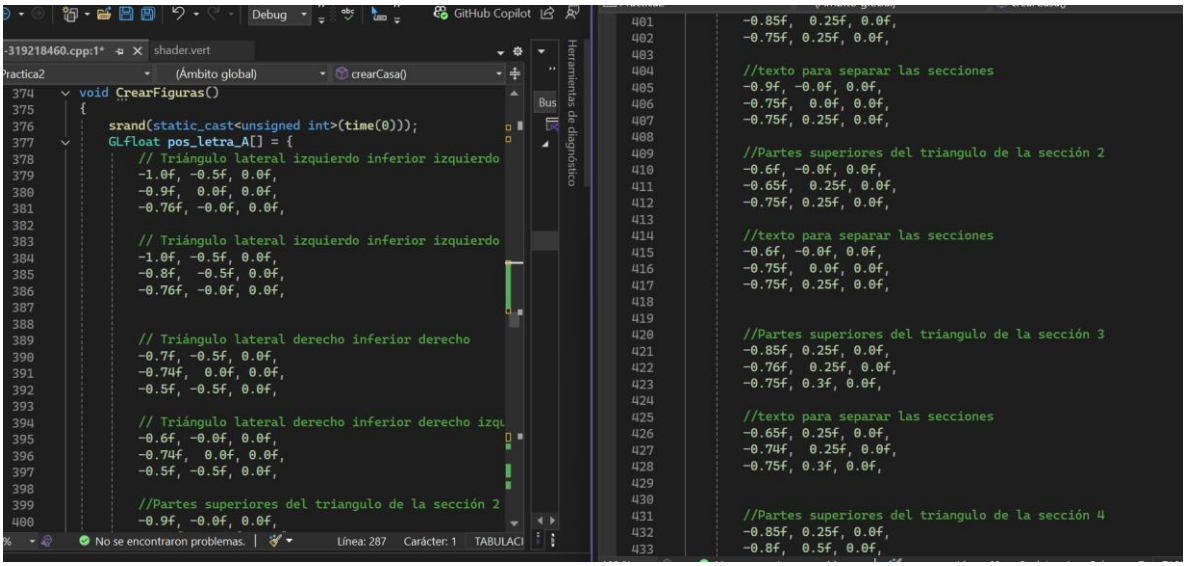
```
unsigned int troncoDer_indices[] = {
    // front
    0, 1, 2,
    2, 3, 0,
    // right
    1, 5, 6,
    6, 2, 1,
    // back
    7, 6, 5,
    5, 4, 7,
    // left
    4, 0, 3,
    3, 7, 4,
    // bottom
    4, 5, 1,
    1, 0, 4,
    // top
    3, 2, 6,
    6, 7, 3
};

GLfloat troncoDer_vertices[] = {
    //Cara trasera tronco izquierdo
    1.8f, 0.25f,   -0.1f,
     1.95f, 0.25f,  -0.1f,
     1.8f,  0.5f, -0.1f,
    1.95f,  0.5f,  -0.1f,

    //Cara frontal tronco izquierdo
    1.8f, 0.25f,   0.1f,
     1.95f, 0.25f, 0.1f,
     1.8f,  0.5f,  0.1f,
    1.95f,  0.5f,  0.1f,
```



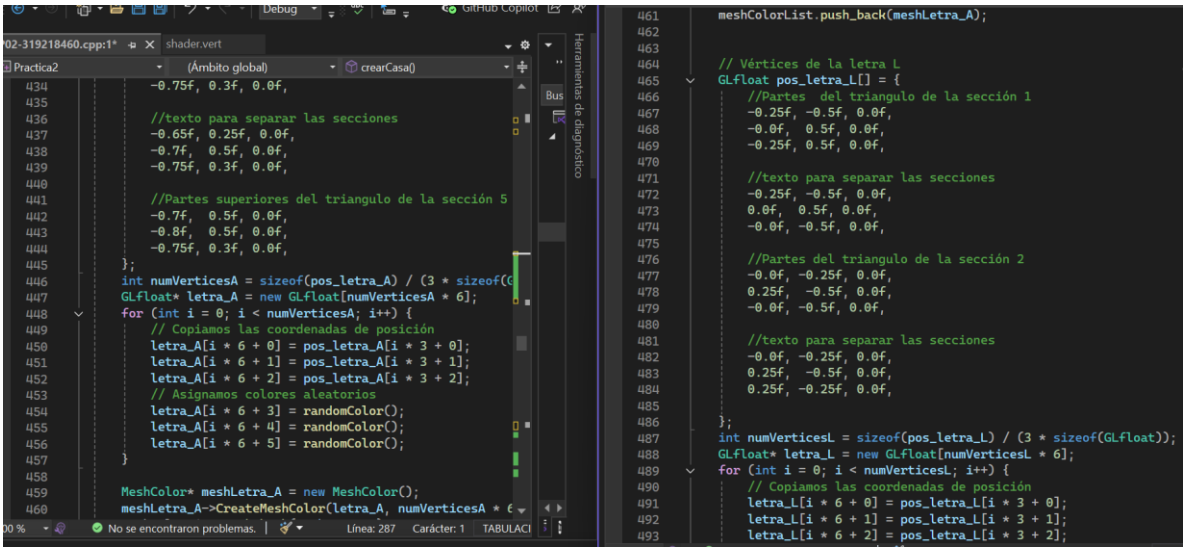
```
Mesh* troncoDer = new Mesh();
troncoDer->CreateMesh(troncoDer_vertices, troncoDer_ir
meshList.push_back(troncoDer);

unsigned int techo_indices[] = {
    // Triángulo frontal
    0, 1, 2,

    // Triángulo trasero
    3, 5, 4,

    // Lado izquierdo
    0, 2, 3,
    3, 2, 5,

    // Lado derecho
    1, 4, 2,
    2, 4, 5,

    // Base
    0, 3, 1,
    1, 3, 4

};
GLfloat techo_vertices[] = {
    // Base frontal
    0.25f, 1.25f,  0.5f,
    1.75f, 1.25f,  0.5f,
    1.00f, 2.00f,  0.5f,

    // Base trasera
    0.25f, 1.25f, -0.5f,
    1.75f, 1.25f, -0.5f,
    1.00f, 2.00f, -0.5f

};
Mesh* techo = new Mesh();
techo->CreateMesh(techo_vertices, techo_indices, 36, 18);
meshList.push_back(techo);


unsigned int copaIzq_indices[] = {
    0, 1, 2, // Triángulo frontal
    3, 5, 4, // Triángulo trasero
    0, 2, 3, 3, 2, 5, // Lado izquierdo
    1, 4, 2, 2, 4, 5, // Lado derecho
    0, 3, 1, 1, 3, 4  // Base cuadrada
};

GLfloat copaIzq_vertices[] = {
    0.0f, 0.5f,  0.3f,
    0.25f, 0.5f,  0.3f,
    0.125f, 1.0f,  0.3f,

    0.0f, 0.5f, -0.3f,
    0.25f, 0.5f, -0.3f,
    0.125f, 1.0f, -0.3f
};

Mesh* copaIzq = new Mesh();
```

```
Mesh* copaIzq = new Mesh();
copaIzq->CreateMesh(copaIzq_vertices, copaIzq_indices,
meshList.push_back(copaIzq);


unsigned int copaDer_indices[] = {
    0, 1, 2,
    3, 5, 4,
    0, 2, 3, 3, 2, 5,
    1, 4, 2, 2, 4, 5,
    0, 3, 1, 1, 3, 4
};

GLfloat copaDer_vertices[] = {
    1.75f, 0.5f,  0.3f,
    2.00f, 0.5f,  0.3f,
    1.875f, 1.0f,  0.3f,

    1.75f, 0.5f, -0.3f,
    2.00f, 0.5f, -0.3f,
    1.875f, 1.0f, -0.3f
};

Mesh* copaDer = new Mesh();
copaDer->CreateMesh(copaDer_vertices, copaDer_indices,
meshList.push_back(copaDer);
```

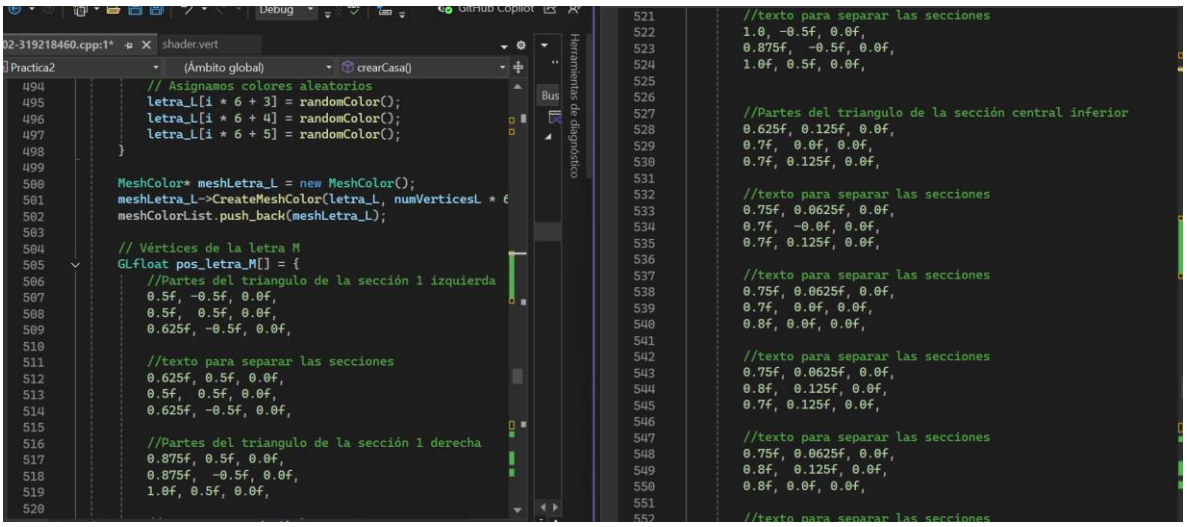
Líneas de código que muestran la función del dibujado de las letras.



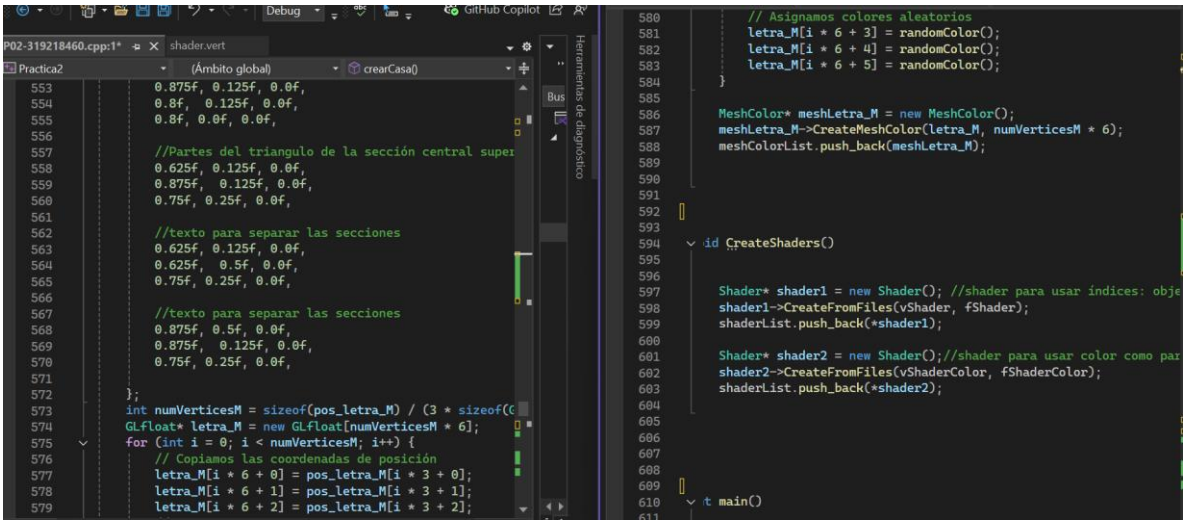
```
void CrearFiguras()
{
    srand(static_cast<unsigned int>(time(0)));
    GLfloat pos_letra_A[] = {
        // Triángulo lateral izquierdo inferior izquierdo
        -1.0f, -0.5f, 0.0f,
        -0.9f,  0.0f, 0.0f,
        -0.76f, -0.0f, 0.0f,

        // Triángulo lateral izquierdo inferior izquierdo
        -1.0f, -0.5f, 0.0f,
        -0.8f,  -0.5f, 0.0f,
        -0.76f, -0.0f, 0.0f,


        // Triángulo lateral derecho inferior derecho
        -0.7f, -0.5f, 0.0f,
        -0.74f,  0.0f, 0.0f,
        -0.5f, -0.5f, 0.0f,

        // Triángulo lateral derecho inferior derecho izqu
        -0.6f, -0.0f, 0.0f,
        -0.74f,  0.0f, 0.0f,
        -0.5f, -0.5f, 0.0f,

        //Partes superiores del triangulo de la sección 2
        -0.9f, -0.0f, 0.0f,
        -0.85f,  0.25f, 0.0f,
        -0.75f, 0.25f, 0.0f,

        //texto para separar las secciones
        -0.9f, -0.0f, 0.0f,
        -0.75f,  0.0f, 0.0f,
        -0.75f, 0.25f, 0.0f,

        //Partes superiores del triangulo de la sección 2
        -0.6f, -0.0f, 0.0f,
        -0.65f,  0.25f, 0.0f,
        -0.75f, 0.25f, 0.0f,

        //texto para separar las secciones
        -0.6f, -0.0f, 0.0f,
        -0.75f,  0.0f, 0.0f,
        -0.75f, 0.25f, 0.0f,


        //Partes superiores del triangulo de la sección 3
        -0.85f, 0.25f, 0.0f,
        -0.76f,  0.25f, 0.0f,
        -0.75f, 0.3f, 0.0f,

        //texto para separar las secciones
        -0.65f, 0.25f, 0.0f,
        -0.74f,  0.25f, 0.0f,
        -0.75f, 0.3f, 0.0f,


        //Partes superiores del triangulo de la sección 4
        -0.85f, 0.25f, 0.0f,
        -0.8f,  0.5f, 0.0f,
```



```
        -0.75f, 0.3f, 0.0f,

        //texto para separar las secciones
        -0.65f, 0.25f, 0.0f,
        -0.7f,  0.5f, 0.0f,
        -0.75f, 0.3f, 0.0f,

        //Partes superiores del triangulo de la sección 5
        -0.7f,  0.5f, 0.0f,
        -0.8f,  0.5f, 0.0f,
        -0.75f, 0.3f, 0.0f,
    };
    int numVerticesA = sizeof(pos_letra_A) / (3 * sizeof(G
    GLfloat* letra_A = new GLfloat[numVerticesA * 6];
    for (int i = 0; i < numVerticesA; i++) {
        // Copiamos las coordenadas de posición
        letra_A[i * 6 + 0] = pos_letra_A[i * 3 + 0];
        letra_A[i * 6 + 1] = pos_letra_A[i * 3 + 1];
        letra_A[i * 6 + 2] = pos_letra_A[i * 3 + 2];
        // Asignamos colores aleatorios
        letra_A[i * 6 + 3] = randomColor();
        letra_A[i * 6 + 4] = randomColor();
        letra_A[i * 6 + 5] = randomColor();
    }

    MeshColor* meshLetra_A = new MeshColor();
    meshLetra_A->CreateMeshColor(letra_A, numVerticesA * 6
    meshColorList.push_back(meshLetra_A);


    // Vértices de la letra L
    GLfloat pos_letra_L[] = {
        //Partes  del triangulo de la sección 1
        -0.25f, -0.5f, 0.0f,
        -0.0f,  0.5f, 0.0f,
        -0.25f, 0.5f, 0.0f,

        //texto para separar las secciones
        -0.25f, -0.5f, 0.0f,
        0.0f, 0.5f, 0.0f,
        -0.0f, -0.5f, 0.0f,

        //Partes del triangulo de la sección 2
        -0.0f, -0.25f, 0.0f,
        0.25f,  -0.5f, 0.0f,
        -0.0f, -0.5f, 0.0f,

        //texto para separar las secciones
        -0.0f, -0.25f, 0.0f,
        0.25f,  -0.5f, 0.0f,
        0.25f, -0.25f, 0.0f,

    };
    int numVerticesL = sizeof(pos_letra_L) / (3 * sizeof(GLfloat));
    GLfloat* letra_L = new GLfloat[numVerticesL * 6];
    for (int i = 0; i < numVerticesL; i++) {
        // Copiamos las coordenadas de posición
        letra_L[i * 6 + 0] = pos_letra_L[i * 3 + 0];
        letra_L[i * 6 + 1] = pos_letra_L[i * 3 + 1];
        letra_L[i * 6 + 2] = pos_letra_L[i * 3 + 2];
```



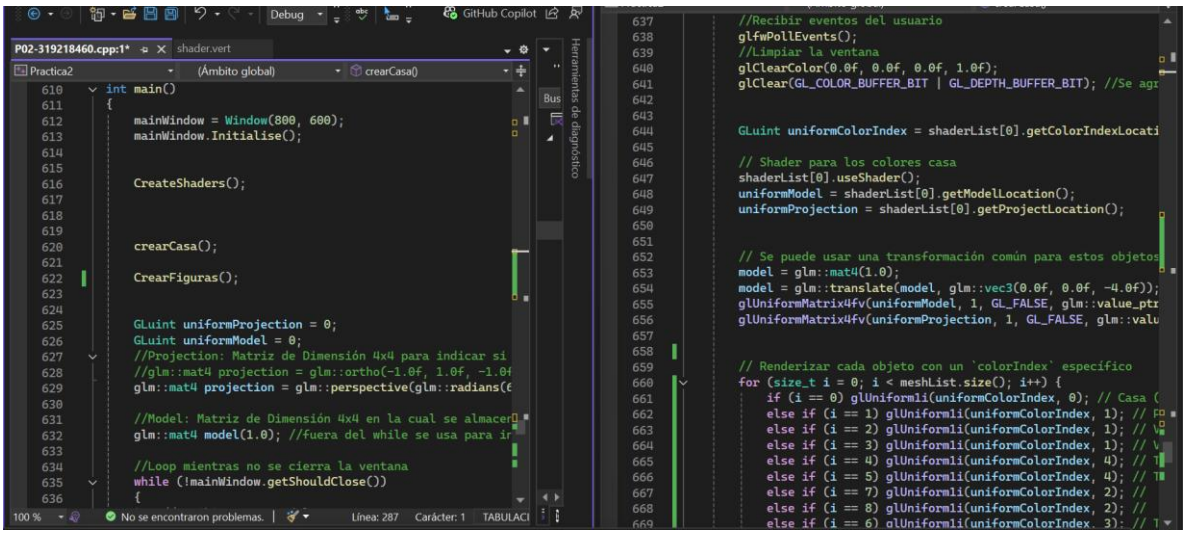
```
        // Asignamos colores aleatorios
        letra_L[i * 6 + 3] = randomColor();
        letra_L[i * 6 + 4] = randomColor();
        letra_L[i * 6 + 5] = randomColor();
    }

    MeshColor* meshLetra_L = new MeshColor();
    meshLetra_L->CreateMeshColor(letra_L, numVerticesL * 6
    meshColorList.push_back(meshLetra_L);

    // Vértices de la letra M
    GLfloat pos_letra_M[] = {
        //Partes del triangulo de la sección 1 izquierda
        0.5f, -0.5f, 0.0f,
        0.5f,  0.5f, 0.0f,
        0.625f, -0.5f, 0.0f,

        //texto para separar las secciones
        0.625f, 0.5f, 0.0f,
        0.5f,  0.5f, 0.0f,
        0.625f, -0.5f, 0.0f,

        //Partes del triangulo de la sección 1 derecha
        0.875f, 0.5f, 0.0f,
        0.875f,  -0.5f, 0.0f,
        1.0f, 0.5f, 0.0f,

        //texto para separar las secciones
        1.0, -0.5f, 0.0f,
        0.875f,  -0.5f, 0.0f,
        1.0f, 0.5f, 0.0f,


        //Partes del triangulo de la sección central inferior
        0.625f, 0.125f, 0.0f,
        0.7f,  0.0f, 0.0f,
        0.7f, 0.125f, 0.0f,

        //texto para separar las secciones
        0.75f, 0.0625f, 0.0f,
        0.7f,  -0.0f, 0.0f,
        0.7f, 0.125f, 0.0f,

        //texto para separar las secciones
        0.75f, 0.0625f, 0.0f,
        0.7f,  0.0f, 0.0f,
        0.8f, 0.0f, 0.0f,

        //texto para separar las secciones
        0.75f, 0.0625f, 0.0f,
        0.8f,  0.125f, 0.0f,
        0.7f, 0.125f, 0.0f,

        //texto para separar las secciones
        0.75f, 0.0625f, 0.0f,
        0.8f,  0.125f, 0.0f,
        0.8f, 0.0f, 0.0f,

        //texto para separar las secciones
```



```
0.875f, 0.125f, 0.0f,
0.8f,  0.125f, 0.0f,
0.8f, 0.0f, 0.0f,

//Partes del triangulo de la sección central super
0.625f, 0.125f, 0.0f,
0.875f,  0.125f, 0.0f,
0.75f, 0.25f, 0.0f,

//texto para separar las secciones
0.625f, 0.125f, 0.0f,
0.625f,  0.5f, 0.0f,
0.75f, 0.25f, 0.0f,

//texto para separar las secciones
0.875f, 0.5f, 0.0f,
0.875f,  0.125f, 0.0f,
0.75f, 0.25f, 0.0f,

};
int numVerticesM = sizeof(pos_letra_M) / (3 * sizeof(G
GLfloat* letra_M = new GLfloat[numVerticesM * 6];
for (int i = 0; i < numVerticesM; i++) {
    // Copiamos las coordenadas de posición
    letra_M[i * 6 + 0] = pos_letra_M[i * 3 + 0];
    letra_M[i * 6 + 1] = pos_letra_M[i * 3 + 1];
    letra_M[i * 6 + 2] = pos_letra_M[i * 3 + 2];
    // Asignamos colores aleatorios
    letra_M[i * 6 + 3] = randomColor();
    letra_M[i * 6 + 4] = randomColor();
    letra_M[i * 6 + 5] = randomColor();
}

MeshColor* meshLetra_M = new MeshColor();
meshLetra_M->CreateMeshColor(letra_M, numVerticesM * 6);
meshColorList.push_back(meshLetra_M);



id CreateShaders()


    Shader* shader1 = new Shader(); //shader para usar índices: obje
    shader1->CreateFromFiles(vShader, fShader);
    shaderList.push_back(*shader1);

    Shader* shader2 = new Shader();//shader para usar color como par
    shader2->CreateFromFiles(vShaderColor, fShaderColor);
    shaderList.push_back(*shader2);



t main()
```

Modificación del Main para la inclusión de el dibujado de letras y creación de la casa, así como su impresión.

En el caso de la casa se imprimió con el index de color asignado en el shader.vert.



```
int main()
{
    mainWindow = Window(800, 600);
    mainWindow.Initialise();


    CreateShaders();



    crearCasa();

    CrearFiguras();


    GLuint uniformProjection = 0;
    GLuint uniformModel = 0;
    //Projection: Matriz de Dimensión 4x4 para indicar si
    //glm::mat4 projection = glm::ortho(-1.0f, 1.0f, -1.0f
    glm::mat4 projection = glm::perspective(glm::radians(6

    //Model: Matriz de Dimensión 4x4 en la cual se almacen
    glm::mat4 model(1.0); //fuera del while se usa para ir

    //Loop mientras no se cierra la ventana
    while (!mainWindow.getShouldClose())
    {
        //Recibir eventos del usuario
        glfwPollEvents();
        //Limpiar la ventana
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Se agr


        GLuint uniformColorIndex = shaderList[0].getColorIndexLocati

        // Shader para los colores casa
        shaderList[0].useShader();
        uniformModel = shaderList[0].getModelLocation();
        uniformProjection = shaderList[0].getProjectLocation();


        // Se puede usar una transformación común para estos objetos
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::valu


        // Renderizar cada objeto con un `colorIndex` específico
        for (size_t i = 0; i < meshList.size(); i++) {
            if (i == 0) glUniform1i(uniformColorIndex, 0); // Casa (
            else if (i == 1) glUniform1i(uniformColorIndex, 1); // Fo
            else if (i == 2) glUniform1i(uniformColorIndex, 1); // V
            else if (i == 3) glUniform1i(uniformColorIndex, 1); // V
            else if (i == 4) glUniform1i(uniformColorIndex, 4); // T
            else if (i == 5) glUniform1i(uniformColorIndex, 4); // T
            else if (i == 7) glUniform1i(uniformColorIndex, 2); //
            else if (i == 8) glUniform1i(uniformColorIndex, 2); //
            else if (i == 6) glUniform1i(uniformColorIndex, 3); // T
```

```
    else if (i == 3) glUniform1i(uniformColorIndex, 1); // Ventana Der (verde claro)
    else if (i == 4) glUniform1i(uniformColorIndex, 4); // Tronco Izq (café)
    else if (i == 5) glUniform1i(uniformColorIndex, 4); // Tronco Der (café)
    else if (i == 7) glUniform1i(uniformColorIndex, 2); // Copa Izq (verde oscuro)
    else if (i == 8) glUniform1i(uniformColorIndex, 2); // Copa Der (verde oscuro)
    else if (i == 6) glUniform1i(uniformColorIndex, 3); // Techo (azul)

    meshList[i]->RenderMesh();
}



//Para las letras hay que usar el segundo set de shaders con índice 1 en ShaderList
shaderList[1].useShader();
uniformModel = shaderList[1].getModelLocation();
uniformProjection = shaderList[1].getProjectLocation();

//Inicializar matriz de dimensión 4x4 que servirá como matriz de modelo para almacenar las transformaciones geométricas
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.0f, -1.0f, -4.0f));
//
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y se envían al sh
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[0]->RenderMeshColor();
for (size_t i = 1; i < meshColorList.size(); i++) {
    meshColorList[i]->RenderMeshColor();
}
```

Ejecución:

Conclusión:

Con lo aprendido se ha adquirido el conocimiento para hacer modelado 3D, aunque se tuvo que hacer más investigación de la esperada, los conceptos aprendidos como traslación y rotación (aunque no se puso en práctica en este ejercicio) han sido conceptos útiles junto con la comprensión de figuras en 3d, aún se debe de hacer énfasis que se debe de buscar alguna alternativa para facilitar este tipo de trabajos, esto debido a que la práctica que había hecho antes en trabajos anteriores no ha sido la mejor en este caso.

Aún hay mucho que aprender del tema para encontrar una forma en la que la forma visual sea más agradable a la vista y el proceso de trabajo sea más eficiente,

Bibliografía:

1. Tutorial 4 : Un cubo con color. (s. f.). https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-4-a-colored-cube/?utm_source=chatgpt.com
2. Kurisuto. (2017, 24 noviembre). -♦-Representación de objetos 3D y texturizado en OpenGL [Vídeo]. YouTube. https://www.youtube.com/watch?v=EkXIsI-CSlc
3. Curso de introducción a OpenGL (v1.0). (1999). https://www.cs.buap.mx/~hilario_sm/slide/graficacion/Manual-opengl.pdf?utm_source=chatgpt.com