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: Vargas Luna José Ángel

Nº de Cuenta: 318252333

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 07

SEMESTRE 2026-2

FECHA DE ENTREGA LÍMITE: 01-03-2026

CALIFICACIÓN: _____

1.- Dibujar las iniciales de sus nombres en diagonal de abajo hacia arriba, cada letra de un color diferente

Para la realización de esta práctica, primero haremos la formación de nuestras iniciales.

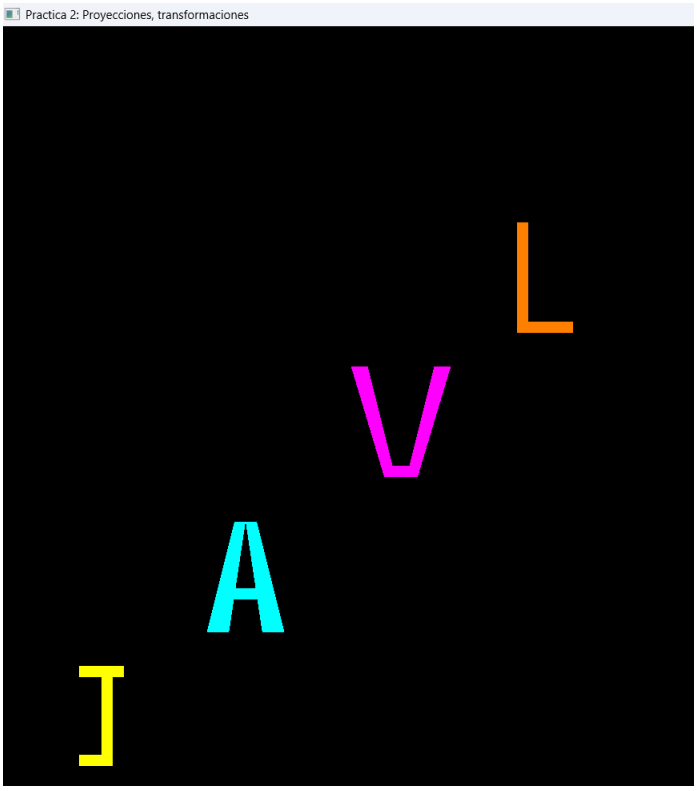
```
void CrearLetrasyFiguras() {
    // --- LETRA J (Amarilla: 1, 1, 0) ---
    GLfloat vertices_J[] = {
        // Barra superior
        -0.2f,  0.5f, 0.0f,  1.0f, 1.0f, 0.0f,   0.2f,  0.5f, 0.0f,  1.0f, 1.0f, 0.0f,   0.2f,  0.4f, 0.0f,  1.0f, 1.0f, 0.0f,
        -0.2f,  0.5f, 0.0f,  1.0f, 1.0f, 0.0f,   0.2f,  0.4f, 0.0f,  1.0f, 1.0f, 0.0f,  -0.2f,  0.4f, 0.0f,  1.0f, 1.0f, 0.0f,
        // Poste
         0.0f,  0.4f, 0.0f,  1.0f, 1.0f, 0.0f,   0.1f,  0.4f, 0.0f,  1.0f, 1.0f, 0.0f,   0.1f, -0.4f, 0.0f,  1.0f, 1.0f, 0.0f,
         0.0f,  0.4f, 0.0f,  1.0f, 1.0f, 0.0f,   0.1f, -0.4f, 0.0f,  1.0f, 1.0f, 0.0f,   0.0f, -0.4f, 0.0f,  1.0f, 1.0f, 0.0f,
         // Gancho
        -0.2f, -0.3f, 0.0f,  1.0f, 1.0f, 0.0f,   0.0f, -0.3f, 0.0f,  1.0f, 1.0f, 0.0f,   0.0f, -0.4f, 0.0f,  1.0f, 1.0f, 0.0f,
        -0.2f, -0.3f, 0.0f,  1.0f, 1.0f, 0.0f,   0.0f, -0.4f, 0.0f,  1.0f, 1.0f, 0.0f,  -0.2f, -0.4f, 0.0f,  1.0f, 1.0f, 0.0f
    };
    MeshColor* letraJ = new MeshColor();
    letraJ->CreateMeshColor(vertices_J, 108);
    meshColorList.push_back(letraJ);

    // --- LETRA A (Cian: 0, 1, 1 - Diseño Robusto) ---
    GLfloat vertices_A[] = {
        // Pata Izquierda
        -0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,  -0.15f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
        -0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,  -0.1f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
        // Pata Derecha
         0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.15f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
         0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.1f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
         // Barra Central
        -0.18f, -0.1f, 0.0f,  0.0f, 1.0f, 1.0f,   0.18f, -0.1f, 0.0f,  0.0f, 1.0f, 1.0f,   0.18f, -0.2f, 0.0f,  0.0f, 1.0f, 1.0f,
        -0.18f, -0.1f, 0.0f,  0.0f, 1.0f, 1.0f,   0.18f, -0.2f, 0.0f,  0.0f, 1.0f, 1.0f,  -0.18f, -0.2f, 0.0f,  0.0f, 1.0f, 1.0f
    };
    MeshColor* letraA = new MeshColor();
```

```
    // --- LETRA A (Cian: 0, 1, 1 - Diseño Robusto) ---
    GLfloat vertices_A[] = {
        // Pata Izquierda
        -0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,  -0.15f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
        -0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,  -0.1f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
        // Pata Derecha
         0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.15f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
         0.35f, -0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.0f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,   0.1f,  0.5f, 0.0f,  0.0f, 1.0f, 1.0f,
         // Barra Central
        -0.18f, -0.1f, 0.0f,  0.0f, 1.0f, 1.0f,   0.18f, -0.1f, 0.0f,  0.0f, 1.0f, 1.0f,   0.18f, -0.2f, 0.0f,  0.0f, 1.0f, 1.0f,
        -0.18f, -0.1f, 0.0f,  0.0f, 1.0f, 1.0f,   0.18f, -0.2f, 0.0f,  0.0f, 1.0f, 1.0f,  -0.18f, -0.2f, 0.0f,  0.0f, 1.0f, 1.0f
    };
    MeshColor* letraA = new MeshColor();
    letraA->CreateMeshColor(vertices_A, 108);
    meshColorList.push_back(letraA);

    // --- LETRA V (Magenta: 1, 0, 1 - Diseño en Bloque Cuadrado abajo) ---
    GLfloat vertices_V[] = {
        // Pata Izquierda Gruesa
        -0.45f,  0.5f, 0.0f,  1.0f, 0.0f, 1.0f,  -0.30f,  0.5f, 0.0f,  1.0f, 0.0f, 1.0f,  -0.05f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,
        -0.45f,  0.5f, 0.0f,  1.0f, 0.0f, 1.0f,  -0.05f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,  -0.15f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,
        // Pata Derecha Gruesa
         0.45f,  0.5f, 0.0f,  1.0f, 0.0f, 1.0f,   0.30f,  0.5f, 0.0f,  1.0f, 0.0f, 1.0f,   0.05f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,
         0.45f,  0.5f, 0.0f,  1.0f, 0.0f, 1.0f,   0.05f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,   0.15f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,
         // Unión Inferior (Efecto Cuadrado)
        -0.15f, -0.4f, 0.0f,  1.0f, 0.0f, 1.0f,   0.15f, -0.4f, 0.0f,  1.0f, 0.0f, 1.0f,   0.15f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,
        -0.15f, -0.4f, 0.0f,  1.0f, 0.0f, 1.0f,   0.15f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f,  -0.15f, -0.5f, 0.0f,  1.0f, 0.0f, 1.0f
    };
    MeshColor* letraV = new MeshColor();
    letraV->CreateMeshColor(vertices_V, 108);
    meshColorList.push_back(letraV);
```

```
71
72      // --- LETRA L (Naranja: 1, 0.5, 0) ---
73      GLfloat vertices_L[] = {
74          // Poste vertical
75          -0.25f,  0.5f, 0.0f,  1.0f, 0.5f, 0.0f,  -0.15f,  0.5f, 0.0f,  1.0f, 0.5f, 0.0f,  -0.15f, -0.5f, 0.0f,  1.0f, 0.5f, 0.0f,
76          -0.25f,  0.5f, 0.0f,  1.0f, 0.5f, 0.0f,  -0.15f, -0.5f, 0.0f,  1.0f, 0.5f, 0.0f,  -0.25f, -0.5f, 0.0f,  1.0f, 0.5f, 0.0f,
77          // Base horizontal
78          -0.15f, -0.4f, 0.0f,  1.0f, 0.5f, 0.0f,   0.25f, -0.4f, 0.0f,  1.0f, 0.5f, 0.0f,   0.25f, -0.5f, 0.0f,  1.0f, 0.5f, 0.0f,
79          -0.15f, -0.4f, 0.0f,  1.0f, 0.5f, 0.0f,   0.25f, -0.5f, 0.0f,  1.0f, 0.5f, 0.0f,  -0.15f, -0.5f, 0.0f,  1.0f, 0.5f, 0.0f
80      };
81      MeshColor* letraL = new MeshColor();
82      letraL->CreateMeshColor(vertices_L, 72);
83      meshColorList.push_back(letraL);
84  }
85
86  void CreateShaders() {
87      Shader* shaderColor = new Shader();
88      shaderColor->CreateFromFiles(vShaderColor, fShaderColor);
89      shaderList.push_back(*shaderColor);
90  }
91
92  int main() {
93      mainWindow = Window(800, 800);
94      mainWindow.Initialise();
95      CrearLetrasyFiguras();
96      CreateShaders();
97
98      GLuint uniformProjection = 0;
99      GLuint uniformModel = 0;
100     glm::mat4 projection = glm::perspective(glm::radians(60.0f), (GLfloat)mainWindow.getBufferWidth() / mainWindow.getBufferHeight(), 0.
101
102     while (!mainWindow.getShouldClose()) {
103         glfwPollEvents();
104         glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
105         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

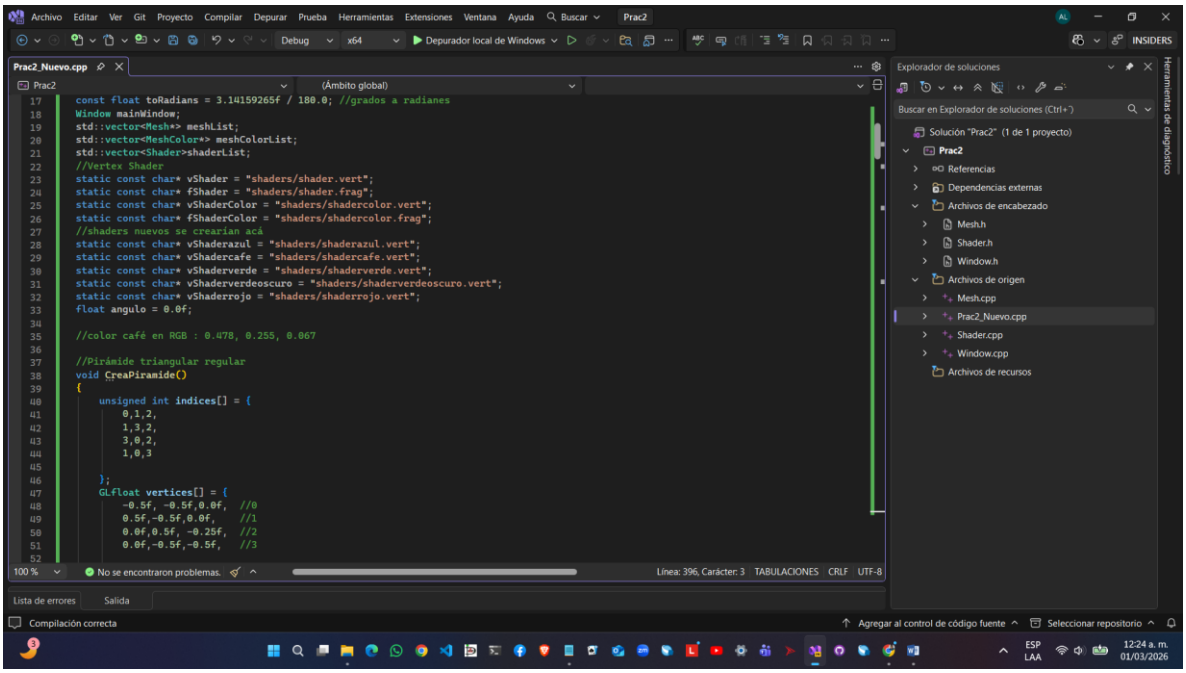
```
int main() {
    mainWindow = Window(800, 800);
    mainWindow.Initialise();
    CrearLetrasyFiguras();
    CreateShaders();

    GLuint uniformProjection = 0;
    GLuint uniformModel = 0;
    glm::mat4 projection = glm::perspective(glm::radians(60.0f), (GLfloat)mainWindow.getBufferWidth() / mainWindow.getBufferHeight(), 0.

    while (!mainWindow.getShouldClose()) {
        glfwPollEvents();
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        shaderList[0].useShader();
        uniformModel = shaderList[0].getModelLocation();
        uniformProjection = shaderList[0].getProjectLocation();
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));

        // Renderizado en diagonal ascendente (J -> A -> V -> L)
        float coords[] = { -2.0f, -0.7f, 0.7f, 2.0f };
        for (int i = 0; i < 4; i++) {
            glm::mat4 model(1.0);
            model = glm::translate(model, glm::vec3(coords[i], coords[i], -5.0f));
            glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
            meshColorList[i]->RenderMeshColor();
```

Con esa lista de configuración, logramos obtener mejor las letras, además de evitar que se vean mal configuradas con demasiados triángulos, además en la función main, ponemos las coordenadas donde quedarán nuestras iniciales de manera diagonal y ascendente.

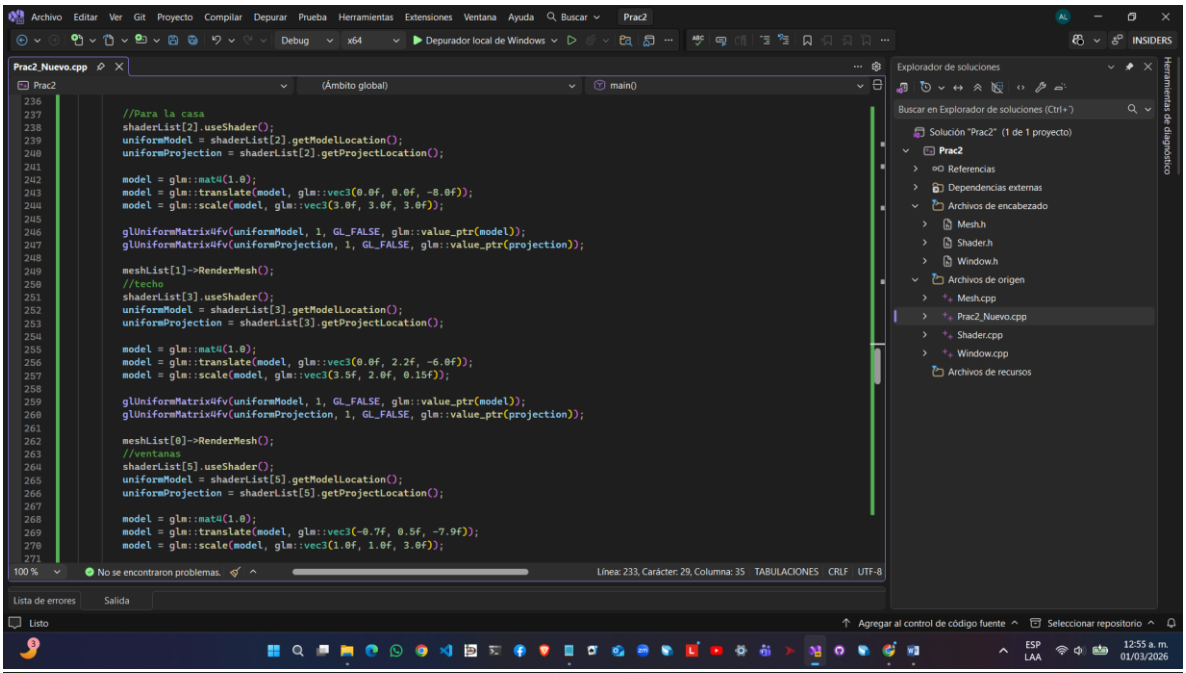


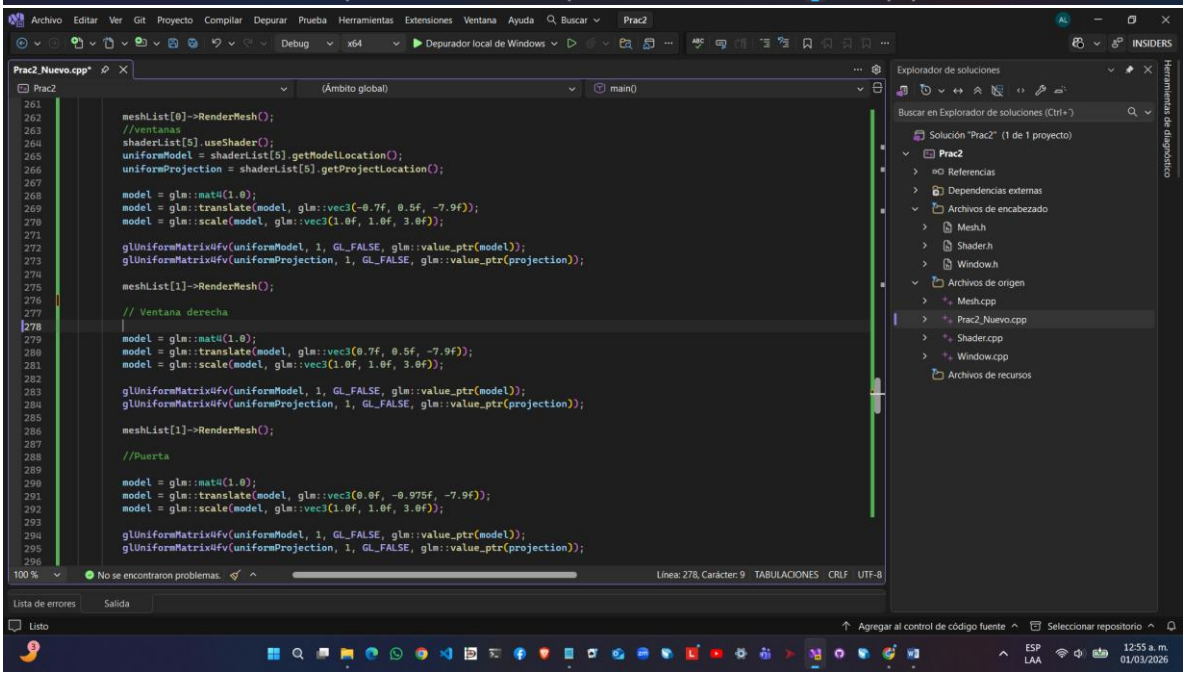
2.- Generar el dibujo de la casa de la clase, pero en lugar de instanciar triángulos y cuadrados será instanciando pirámides y cubos, para esto se requiere crear shaders diferentes de los colores: rojo, verde, azul, café y verde oscuro en lugar de usar el shader con el color clamp. Se debe de entregar los archivos de shader diferentes dentro de un zip adicional al main y documento escrito.

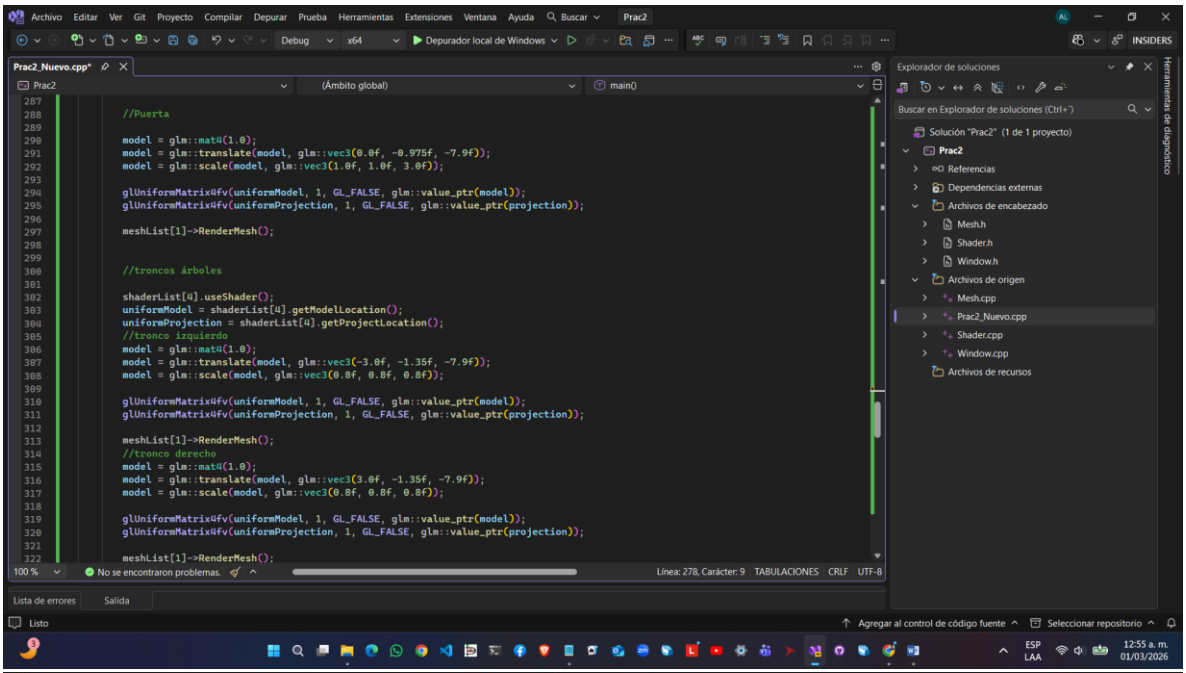
Primero crearemos nuestros shader de cada color solicitado en la práctica (rojo, azul, verde, café y verde oscuro) para después llamarlos dentro del código principal main.

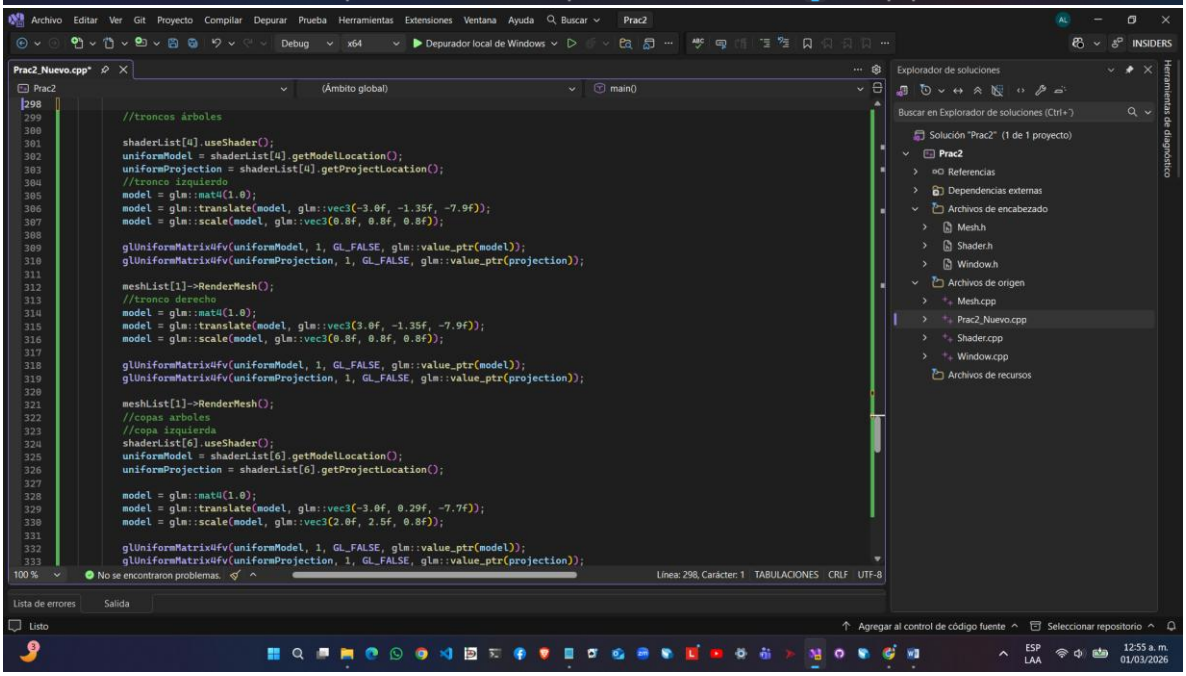


Primero llamaremos a los shaders, donde estos serán llamados y puestos como una nueva llamada en el main. Después iremos creando cada una de las partes de la casa, puerta, ventanas izquierda y derecha y los árboles.

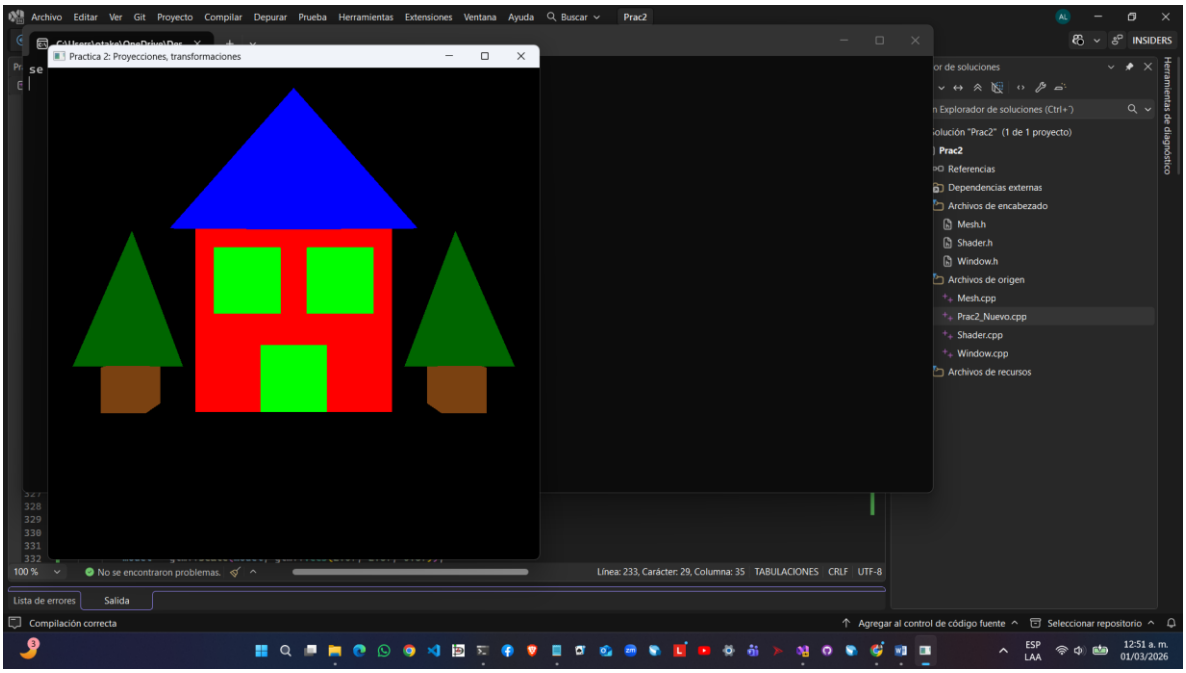

Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Prac2
Debug x64 Depurador local de Windows
INSIDERS
Prac2_Nuevo.cpp
Prac2 (Ámbito global) main()
236 //Para la casa
237 shaderList[2].useShader();
238 uniformModel = shaderList[2].getModelLocation();
239 uniformProjection = shaderList[2].getProjectLocation();
240
241 model = glm::mat4(1.0);
242 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -8.0f));
243 model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
244
245 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
246 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
247
248 meshList[1]->RenderMesh();
249 //techo
250 shaderList[3].useShader();
251 uniformModel = shaderList[3].getModelLocation();
252 uniformProjection = shaderList[3].getProjectLocation();
253
254 model = glm::mat4(1.0);
255 model = glm::translate(model, glm::vec3(0.0f, 2.2f, -6.0f));
256 model = glm::scale(model, glm::vec3(3.5f, 2.0f, 0.15f));
257
258 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
259 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
260
261 meshList[0]->RenderMesh();
262 //ventanas
263 shaderList[5].useShader();
264 uniformModel = shaderList[5].getModelLocation();
265 uniformProjection = shaderList[5].getProjectLocation();
266
267 model = glm::mat4(1.0);
268 model = glm::translate(model, glm::vec3(-0.7f, 0.5f, -7.9f));
269 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 3.0f));
270
100 % No se encontraron problemas. Línea: 233, Carácter: 29, Columna: 35 TABULACIONES CRLF UTF-8
Lista de errores Salida
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Prac2" (1 de 1 proyecto)
Prac2
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
Mesh.cpp
Prac2_Nuevo.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
Listo
Agregar al control de código fuente
Seleccionar repositorio
ESP LAA 12:55 a. m. 01/03/2026


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Prac2
Debug x64 Depurador local de Windows
INSIDERS
Prac2_Nuevo.cpp*
Prac2 (Ámbito global) main()
261 meshList[0]->RenderMesh();
262 //ventanas
263 shaderList[5].useShader();
264 uniformModel = shaderList[5].getModelLocation();
265 uniformProjection = shaderList[5].getProjectLocation();
266
267 model = glm::mat4(1.0);
268 model = glm::translate(model, glm::vec3(-0.7f, 0.5f, -7.9f));
269 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 3.0f));
270
271 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
272 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
273
274 meshList[1]->RenderMesh();
275
276 // Ventana derecha
277
278 model = glm::mat4(1.0);
279 model = glm::translate(model, glm::vec3(0.7f, 0.5f, -7.9f));
280 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 3.0f));
281
282 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
283 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
284
285 meshList[1]->RenderMesh();
286
287 //Puerta
288
289 model = glm::mat4(1.0);
290 model = glm::translate(model, glm::vec3(0.0f, -0.975f, -7.9f));
291 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 3.0f));
292
293 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
294 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
295
296
100 % No se encontraron problemas. Línea: 278, Carácter: 9 TABULACIONES CRLF UTF-8
Lista de errores Salida
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Prac2" (1 de 1 proyecto)
Prac2
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
Mesh.cpp
Prac2_Nuevo.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
Listo
Agregar al control de código fuente
Seleccionar repositorio
ESP LAA 12:55 a. m. 01/03/2026





Cuando se ejecuta correctamente la casa sale de este resultado



Problemas:

Uno de los principales problemas que tuve fue convertir figuras 2D a 3D, donde ahora tuve que entender el concepto de profundidad y vista con fondo de profundidad. Además de buscar cómo funcionan ahora figuras 3D como en este caso cubos y pirámides.

Conclusiones

Ahora que iremos avanzando con las figuras de molde 2D a 3D, entender cómo funciona el concepto de profundidad, además de lograr ir entendiendo como evitar que se traspasen o encimen las figuras una encima de la otra, también deberemos ir entendiendo como pronto funcionará la cámara en respecto a las figuras que tengamos en frente.

Referencias

Overlap and Depth Buffering. (s. f.).

https://paroj.github.io/gltut/Positioning/Tut05%20Overlap%20and%20Depth%20Buffering.html