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 01

NOMBRE COMPLETO: Vargas Luna José Ángel

Nº de Cuenta: 318252333

GRUPO DE LABORATORIO: 02

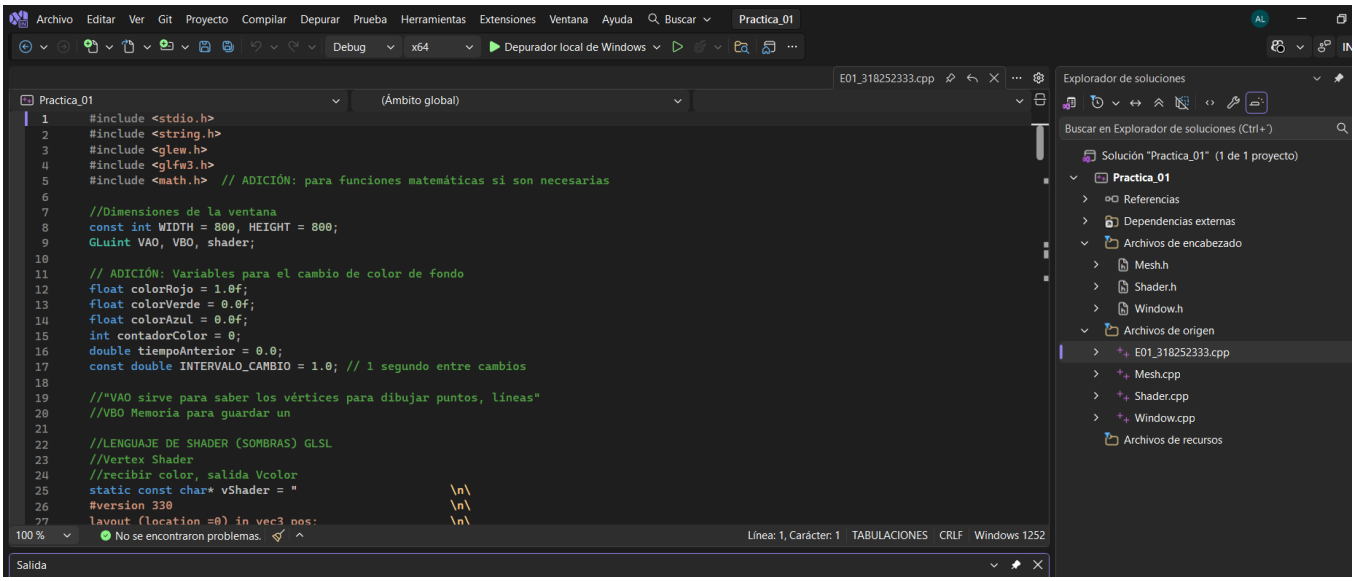
GRUPO DE TEORÍA: 07

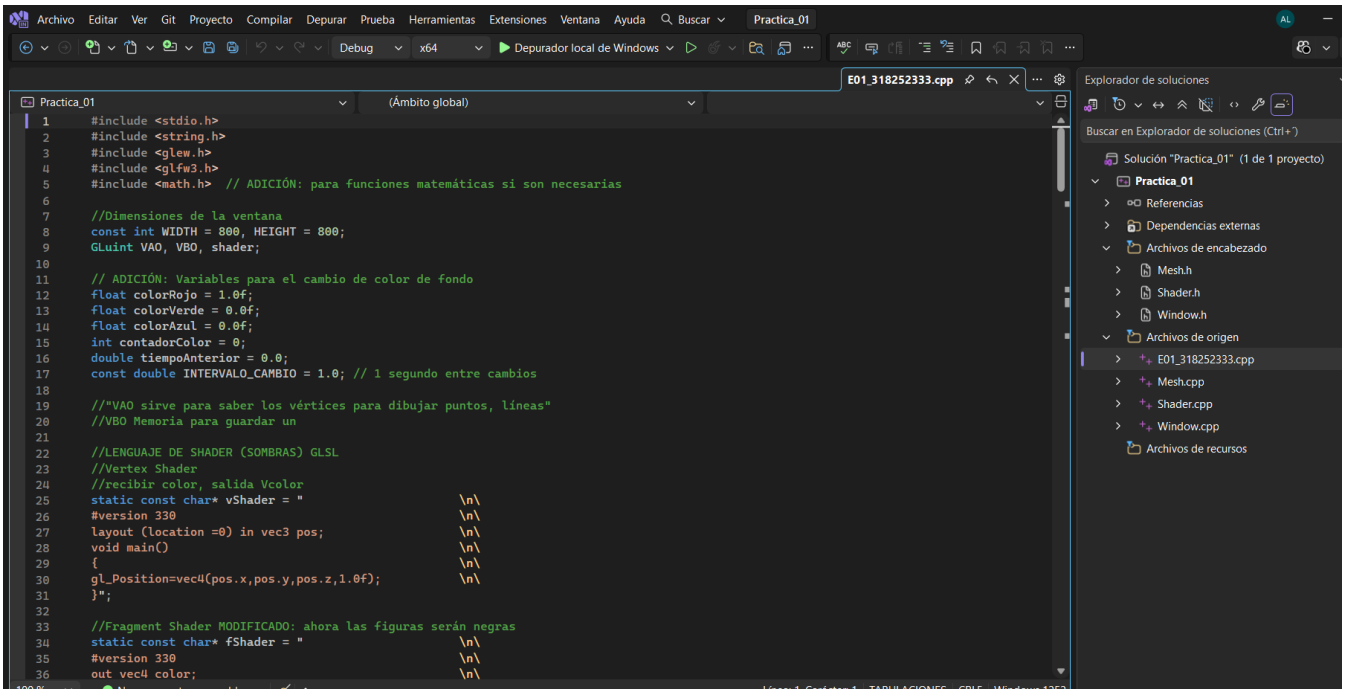
SEMESTRE 2026-2

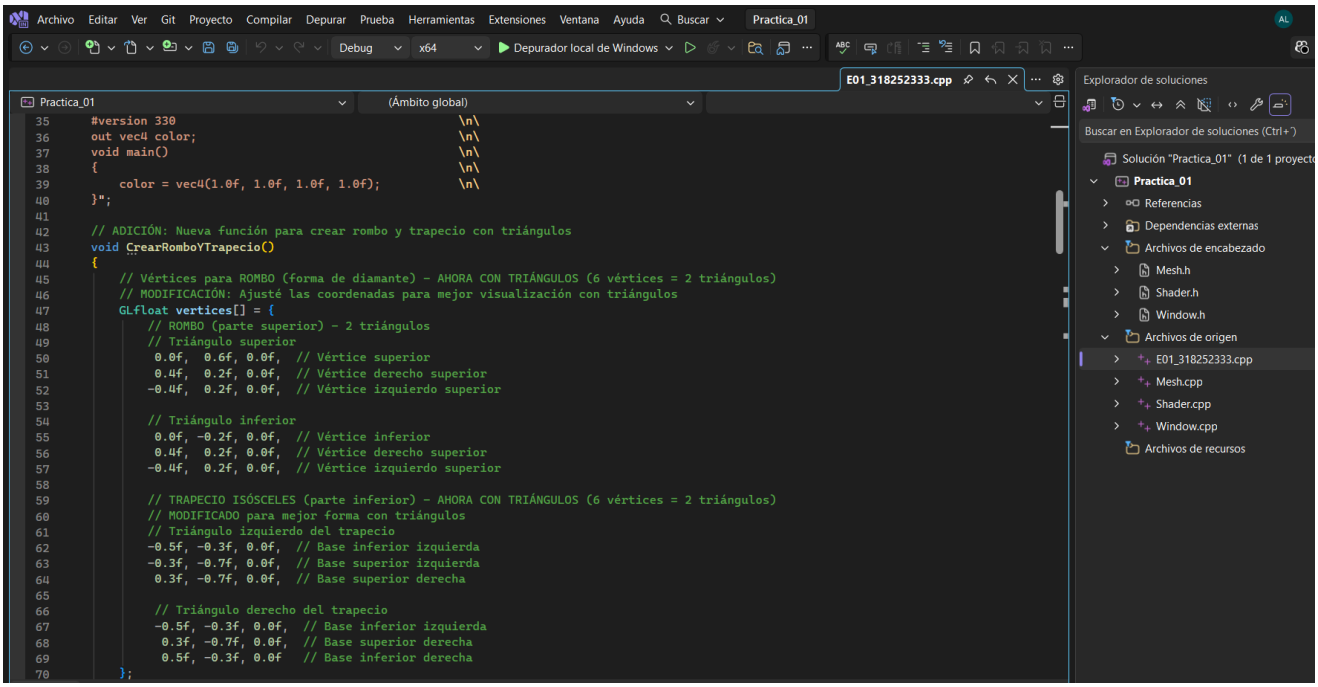
FECHA DE ENTREGA LÍMITE: 22-02-2026

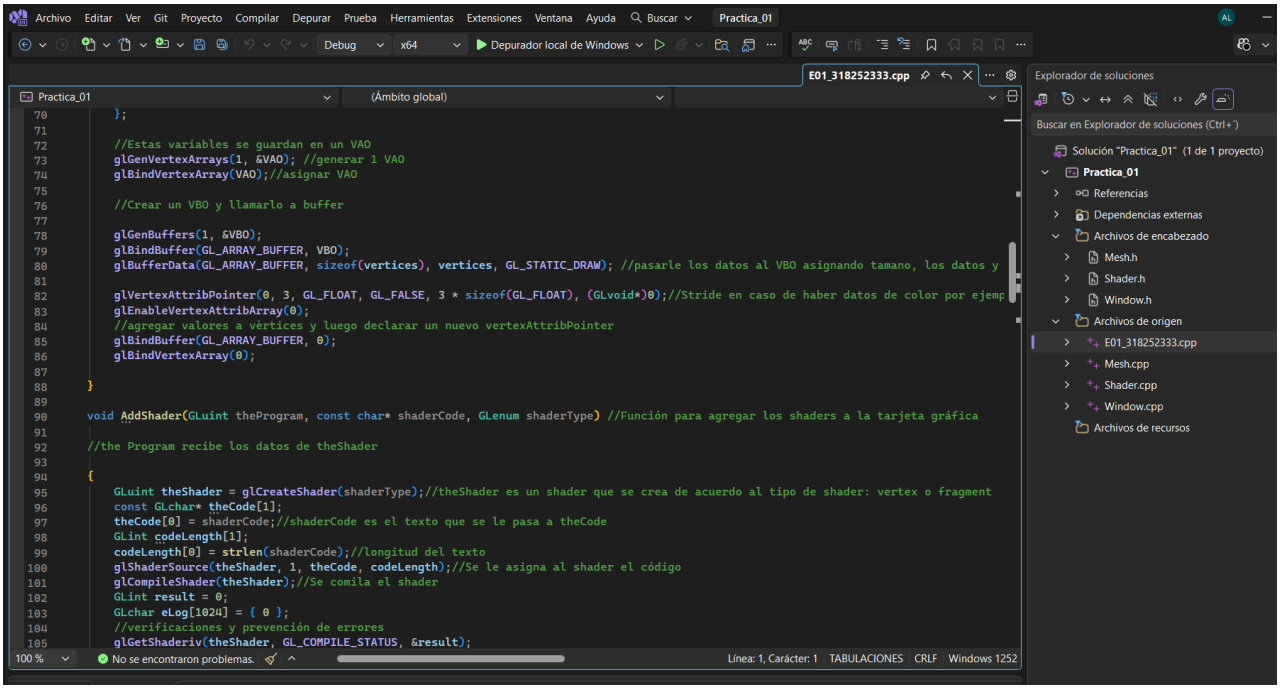
CALIFICACIÓN: _____

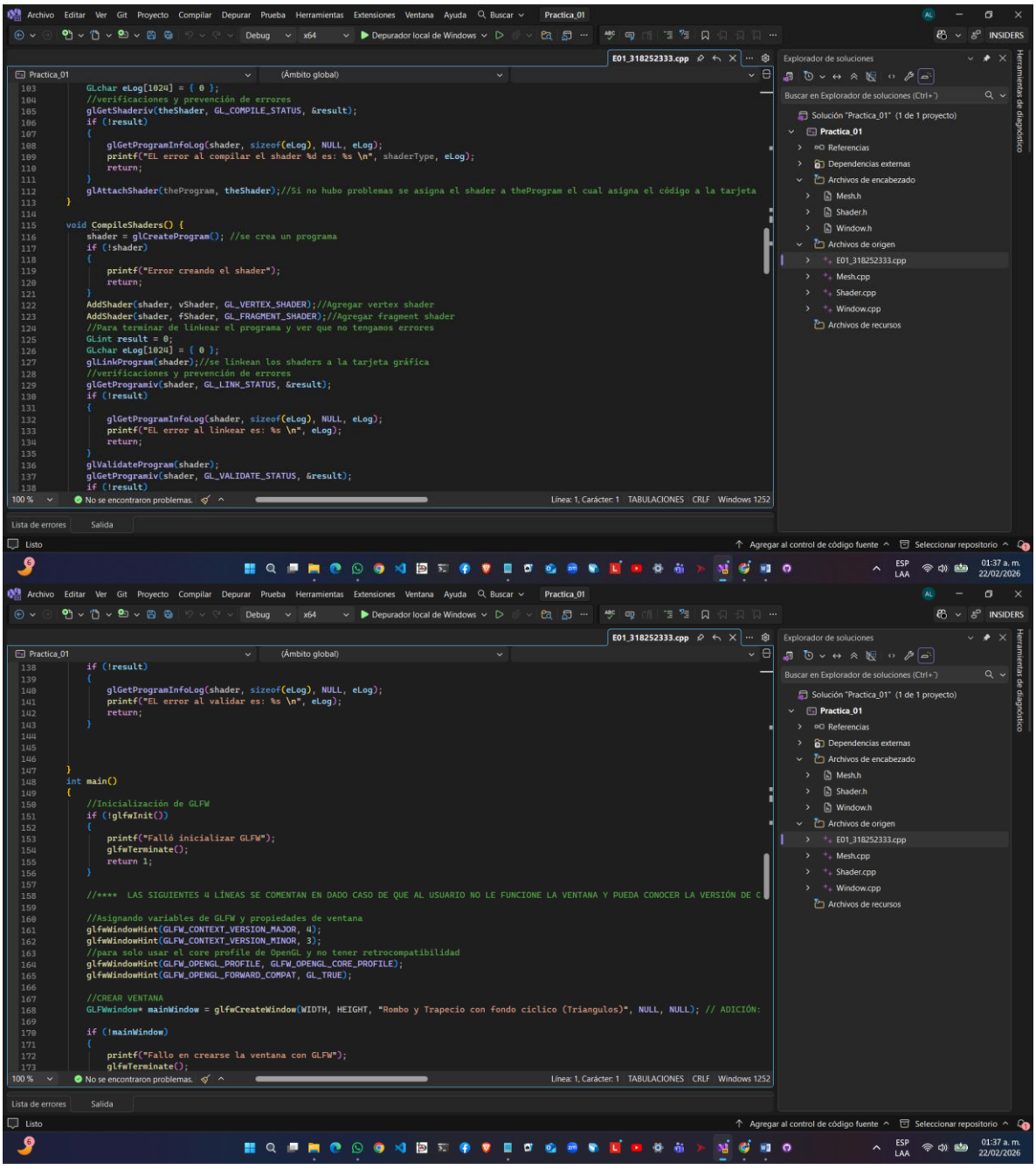
1.-Cambiar el color de fondo de la pantalla entre rojo, verde y azul de forma cíclica y solamente mostrando esos 3 colores con un periodo de lapso adecuado para el ojo humano
2.- Dibujar de forma simultánea en la ventana 1 rombo y 1 trapecio isósceles



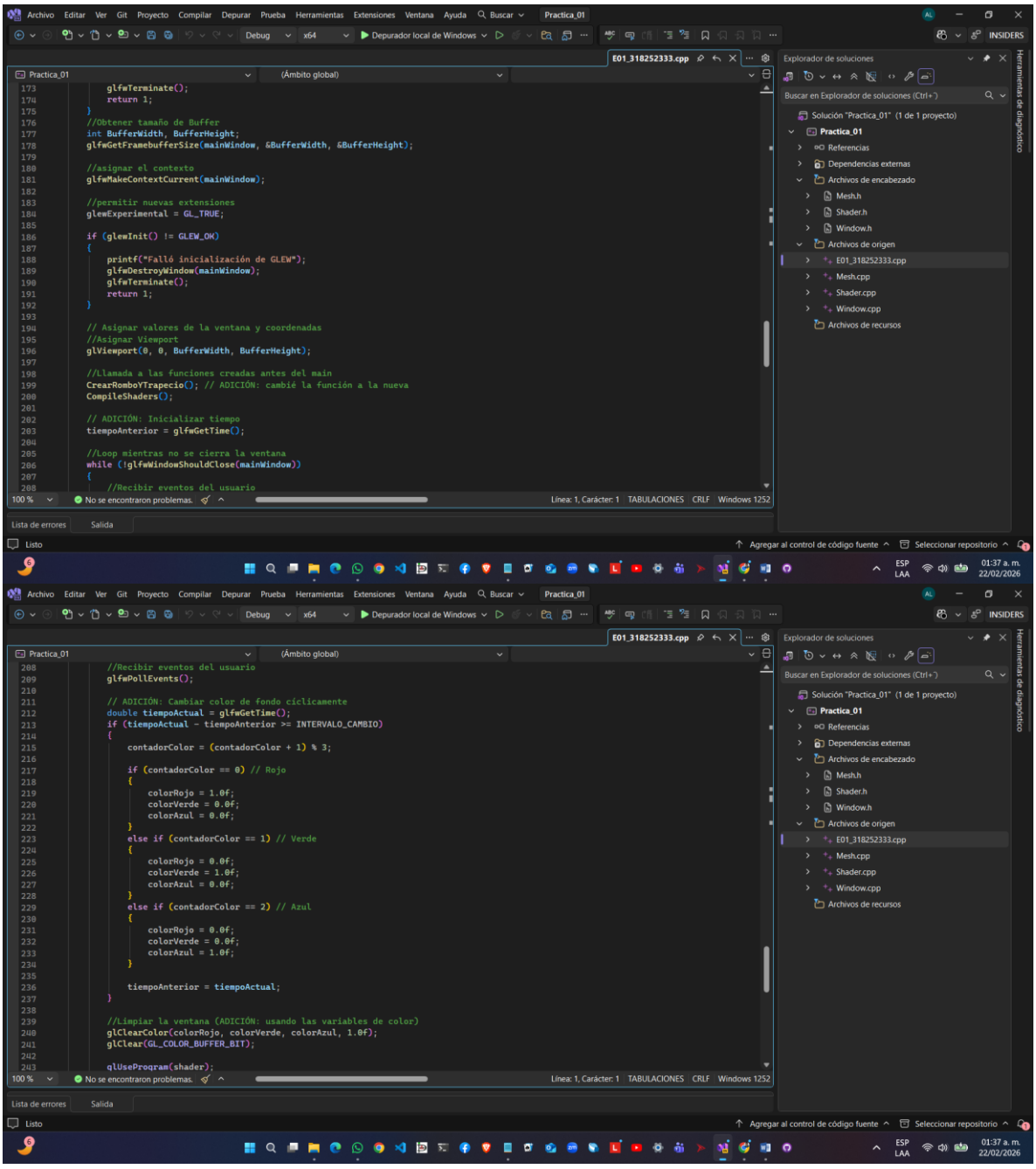



Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
E01_318252333.cpp
Practica_01 (Ámbito global)
#version 330 \n\
out vec4 color; \n\
void main() \n\
{ \n\
color = vec4(1.0f, 1.0f, 1.0f, 1.0f); \n\
}";
// ADICIÓN: Nueva función para crear rombo y trapecio con triángulos
void CrearRomboYTrapecio()
{
// Vértices para ROMBO (forma de diamante) - AHORA CON TRIÁNGULOS (6 vértices = 2 triángulos)
// MODIFICACIÓN: Ajusté las coordenadas para mejor visualización con triángulos
GLfloat vertices[] = {
// ROMBO (parte superior) - 2 triángulos
// Triángulo superior
0.0f, 0.6f, 0.0f, // Vértice superior
0.4f, 0.2f, 0.0f, // Vértice derecho superior
-0.4f, 0.2f, 0.0f, // Vértice izquierdo superior
// Triángulo inferior
0.0f, -0.2f, 0.0f, // Vértice inferior
0.4f, 0.2f, 0.0f, // Vértice derecho superior
-0.4f, 0.2f, 0.0f, // Vértice izquierdo superior
// TRAPECIO ISÓSCELES (parte inferior) - AHORA CON TRIÁNGULOS (6 vértices = 2 triángulos)
// MODIFICADO para mejor forma con triángulos
// Triángulo izquierdo del trapecio
-0.5f, -0.3f, 0.0f, // Base inferior izquierda
-0.3f, -0.7f, 0.0f, // Base superior izquierda
0.3f, -0.7f, 0.0f, // Base superior derecha
// Triángulo derecho del trapecio
-0.5f, -0.3f, 0.0f, // Base inferior izquierda
0.3f, -0.7f, 0.0f, // Base superior derecha
0.5f, -0.3f, 0.0f // Base inferior derecha
};
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+')
Solución "Practica_01" (1 de 1 proyecto
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E01_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos

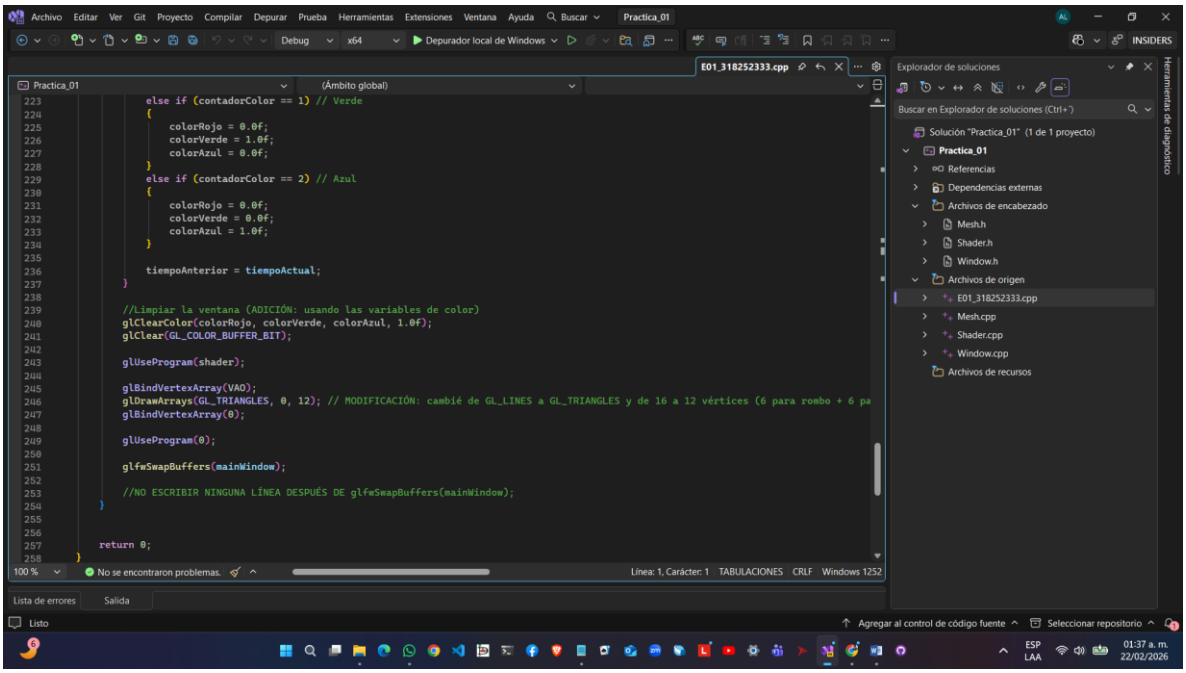

Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
E01_318252333.cpp
Practica_01 (Ámbito global)
};
//Estas variables se guardan en un VAO
glGenVertexArrays(1, &VAO); //generar 1 VAO
glBindVertexArray(VAO);//asignar VAO
//Crear un VBO y llamarlo a buffer
glGenBuffers(1, &VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW); //pasarle los datos al VBO asignando tamano, los datos y
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GL_FLOAT), (GLvoid*)0);//Stride en caso de haber datos de color por ejem
glEnableVertexAttribArray(0);
//agregar valores a vèrtices y luego declarar un nuevo vertexAttribPointer
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
}
void AddShader(GLuint theProgram, const char* shaderCode, GLenum shaderType) //Función para agregar los shaders a la tarjeta gráfica
//the Program recibe los datos de theShader
{
GLuint theShader = glCreateShader(shaderType);//theShader es un shader que se crea de acuerdo al tipo de shader: vertex o fragment
const GLchar* theCode[1];
theCode[0] = shaderCode;//shaderCode es el texto que se le pasa a theCode
GLint codeLength[1];
codeLength[0] = strlen(shaderCode);//longitud del texto
glShaderSource(theShader, 1, theCode, codeLength);//Se le asigna al shader el código
glCompileShader(theShader);//Se comila el shader
GLint result = 0;
GLchar eLog[1024] = { 0 };
//verificaciones y prevención de errores
glGetShaderiv(theShader, GL_COMPILE_STATUS, &result);
100 % No se encontraron problemas.
Línea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+')
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E01_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
INSIDERS
E01_318252333.cpp
Practica_01 (Ámbito global)
103 GLchar eLog[1024] = { 0 };
104 //verificaciones y prevención de errores
105 glGetShaderiv(theShader, GL_COMPILE_STATUS, &result);
106 if (!result)
107 {
108 glGetProgramInfoLog(shader, sizeof(eLog), NULL, eLog);
109 printf("EL error al compilar el shader %d es: %s \n", shaderType, eLog);
110 return;
111 }
112 glAttachShader(theProgram, theShader);//Si no hubo problemas se asigna el shader a theProgram el cual asigna el código a la tarjeta
113 }
114
115 void CompileShaders() {
116 shader = glCreateProgram(); //se crea un programa
117 if (!shader)
118 {
119 printf("Error creando el shader");
120 return;
121 }
122 AddShader(shader, vShader, GL_VERTEX_SHADER);//Agregar vertex shader
123 AddShader(shader, fShader, GL_FRAGMENT_SHADER);//Agregar fragment shader
124 //Para terminar de linkear el programa y ver que no tengamos errores
125 GLint result = 0;
126 GLchar eLog[1024] = { 0 };
127 glLinkProgram(shader);//se linkean los shaders a la tarjeta gráfica
128 //verificaciones y prevención de errores
129 glGetProgramiv(shader, GL_LINK_STATUS, &result);
130 if (!result)
131 {
132 glGetProgramInfoLog(shader, sizeof(eLog), NULL, eLog);
133 printf("EL error al linkear es: %s \n", eLog);
134 return;
135 }
136 glValidateProgram(shader);
137 glGetProgramiv(shader, GL_VALIDATE_STATUS, &result);
138 if (!result)
100 % No se encontraron problemas. Línea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252
Lista de errores Salida
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E01_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
Listo
Agregar al control de código fuente
Seleccionar repositorio
ESP LAA 01:37 a. m. 22/02/2026

Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
INSIDERS
E01_318252333.cpp
Practica_01 (Ámbito global)
138 if (!result)
139 {
140 glGetProgramInfoLog(shader, sizeof(eLog), NULL, eLog);
141 printf("EL error al validar es: %s \n", eLog);
142 return;
143 }
144
145
146
147 }
148 int main()
149 {
150 //Inicialización de GLFW
151 if (!glfwInit())
152 {
153 printf("Falló inicializar GLFW");
154 glfwTerminate();
155 return 1;
156 }
157
158 //**** LAS SIGUIENTES 4 LÍNEAS SE COMENTAN EN DADO CASO DE QUE AL USUARIO NO LE FUNCIONE LA VENTANA Y PUEDA CONOCER LA VERSIÓN DE
159
160 //Asignando variables de GLFW y propiedades de ventana
161 glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
162 glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
163 //para solo usar el core profile de OpenGL y no tener retrocompatibilidad
164 glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
165 glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
166
167 //CREAR VENTANA
168 GLFWwindow* mainWindow = glfwCreateWindow(WIDTH, HEIGHT, "Rombo y Trapecio con fondo ciclico (Triangulos)", NULL, NULL); // ADICIÓN:
169
170 if (!mainWindow)
171 {
172 printf("Fallo en crearse la ventana con GLFW");
173 glfwTerminate();
100 % No se encontraron problemas. Línea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252
Lista de errores Salida
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E01_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
Listo
Agregar al control de código fuente
Seleccionar repositorio
ESP LAA 01:37 a. m. 22/02/2026

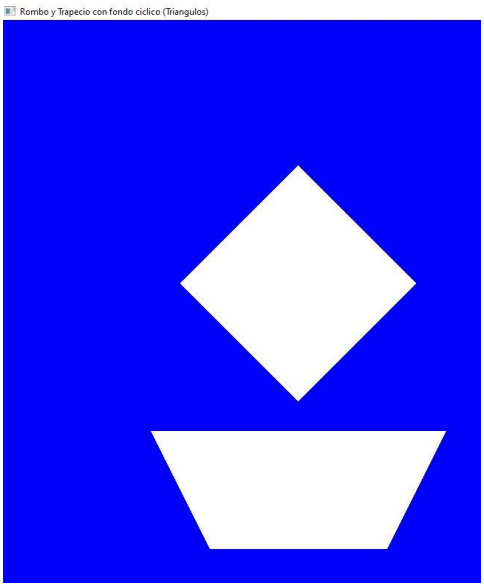

Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
INSIDERS
E01_318252333.cpp
Practica_01 (Ámbito global)
glfwTerminate();
return 1;
}
//Obtener tamaño de Buffer
int BufferWidth, BufferHeight;
glfwGetFramebufferSize(mainWindow, &BufferWidth, &BufferHeight);
//asignar el contexto
glfwMakeContextCurrent(mainWindow);
//permitir nuevas extensiones
glewExperimental = GL_TRUE;
if (glewInit() != GLEW_OK)
{
printf("Falló inicialización de GLEW");
glfwDestroyWindow(mainWindow);
glfwTerminate();
return 1;
}
// Asignar valores de la ventana y coordenadas
//Asignar Viewport
glViewport(0, 0, BufferWidth, BufferHeight);
//Llamada a las funciones creadas antes del main
CrearRomboYTrapecio(); // ADICIÓN: cambié la función a la nueva
CompileShaders();
// ADICIÓN: Inicializar tiempo
tiempoAnterior = glfwGetTime();
//Loop mientras no se cierra la ventana
while (!glfwWindowShouldClose(mainWindow))
{
//Recibir eventos del usuario
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E01_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
100 %
No se encontraron problemas.
Línea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252
Lista de errores Salida
Listo
Agregar al control de código fuente Seleccionar repositorio
ESP LAA 01:37 a. m. 22/02/2026

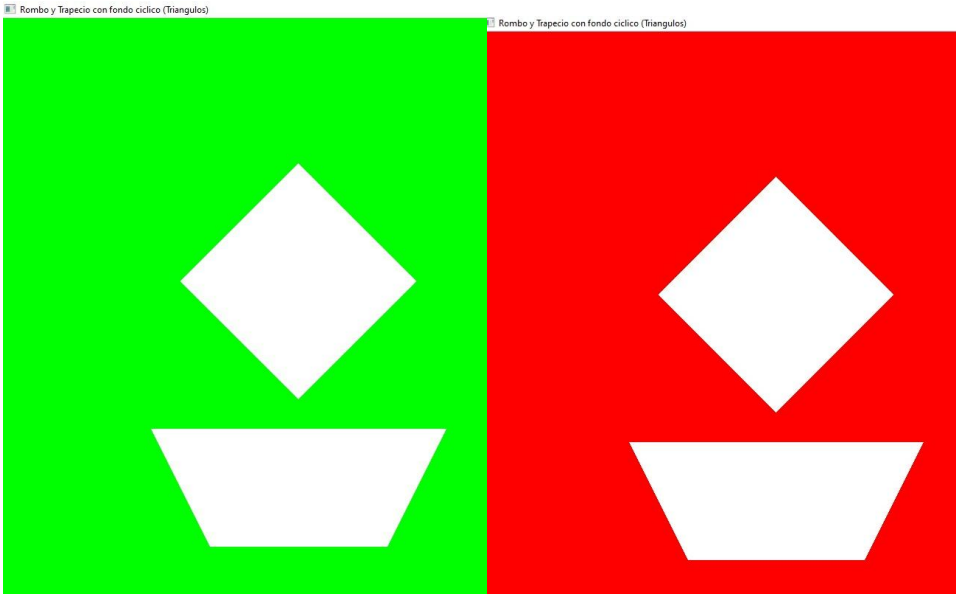
Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
INSIDERS
E01_318252333.cpp
Practica_01 (Ámbito global)
//Recibir eventos del usuario
glfwPollEvents();
// ADICIÓN: Cambiar color de fondo cíclicamente
double tiempoActual = glfwGetTime();
if (tiempoActual - tiempoAnterior >= INTERVALO_CAMBIO)
{
contadorColor = (contadorColor + 1) % 3;
if (contadorColor == 0) // Rojo
{
colorRojo = 1.0f;
colorVerde = 0.0f;
colorAzul = 0.0f;
}
else if (contadorColor == 1) // Verde
{
colorRojo = 0.0f;
colorVerde = 1.0f;
colorAzul = 0.0f;
}
else if (contadorColor == 2) // Azul
{
colorRojo = 0.0f;
colorVerde = 0.0f;
colorAzul = 1.0f;
}
tiempoAnterior = tiempoActual;
}
//Limpiar la ventana (ADICIÓN: usando las variables de color)
glClearColor(colorRojo, colorVerde, colorAzul, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);
glUseProgram(shader);
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E01_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
100 %
No se encontraron problemas.
Línea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252
Lista de errores Salida
Listo
Agregar al control de código fuente Seleccionar repositorio
ESP LAA 01:37 a. m. 22/02/2026



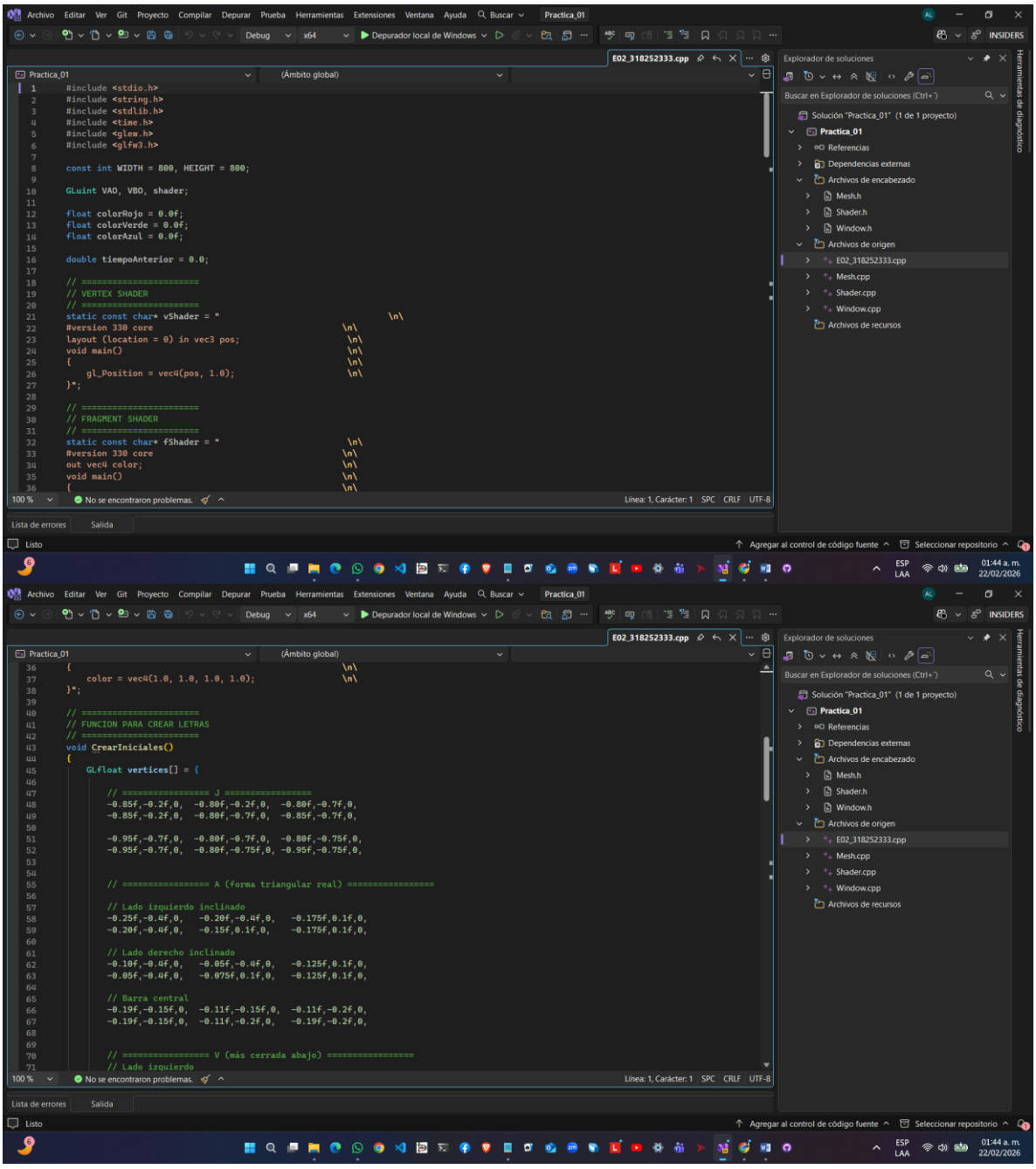
El código de este primer parte de la práctica busca mediante triángulos formar las figuras del rombo y el trapecio mediante medidas permitidas en pantalla, al igual que formar las figuras en base a triángulos (2 triángulos en un rombo y 3 o 4 en un trapecio)

Ejecución

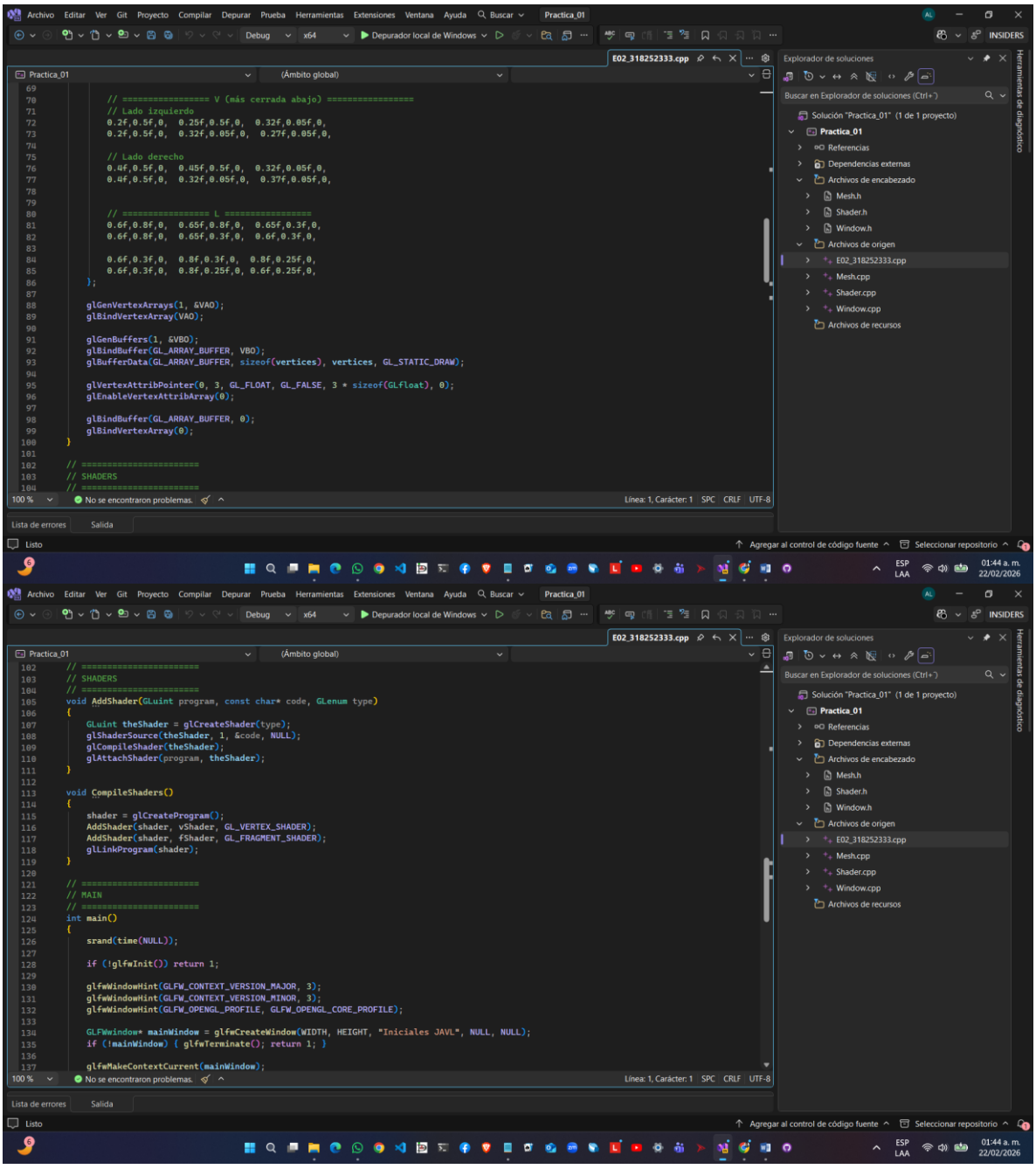




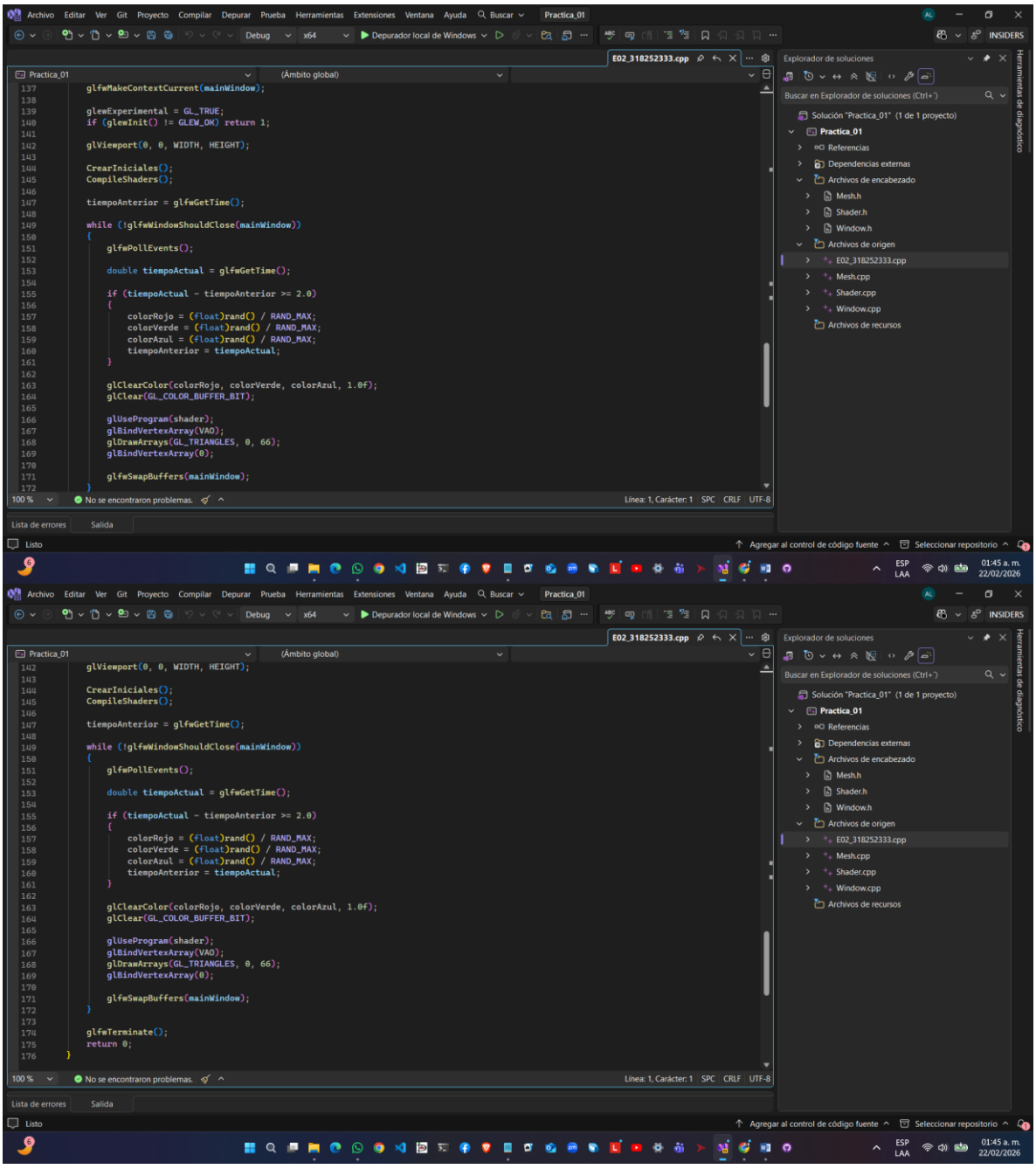
3.- Ventana cambia el color de fondo de forma random tomando rango de colores RGB y con una periodicidad de 2 segundos. (Verificar que al ejecutar el programa varias veces el orden de los colores si lo vean aleatorio y no siempre los mismos)
4.- 3 letras iniciales de sus nombres creadas a partir de triángulos, acomodadas en forma diagonal de abajo hacia arriba, todas las letras son del mismo color.


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
INSIDERS
E02_318252333.cpp
Practica_01 (Ámbito global)
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <glew.h>
#include <glfw3.h>
const int WIDTH = 800, HEIGHT = 800;
GLuint VAO, VBO, shader;
float colorRojo = 0.0f;
float colorVerde = 0.0f;
float colorAzul = 0.0f;
double tiempoAnterior = 0.0;
// ======================
// VERTEX SHADER
// ======================
static const char* vShader = "
#version 330 core
layout (location = 0) in vec3 pos;
void main()
{
gl_Position = vec4(pos, 1.0);
}";
// ======================
// FRAGMENT SHADER
// ======================
static const char* fShader = "
#version 330 core
out vec4 color;
void main()
{
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E02_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
100 %
No se encontraron problemas.
Línea: 1, Carácter: 1 SPC CRLF UTF-8
Lista de errores
Salida
Listo
Agregar al control de código fuente
Seleccionar repositorio
ESP LAA
01:44 a. m. 22/02/2026

Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Herramientas Extensiones Ventana Ayuda Buscar Practica_01
Debug x64 Depurador local de Windows
INSIDERS
E02_318252333.cpp
Practica_01 (Ámbito global)
{
color = vec4(1.0, 1.0, 1.0, 1.0);
}";
// ======================
// FUNCION PARA CREAR LETRAS
// ======================
void CrearIniciales()
{
GLfloat vertices[] = {
// ================ J ================
-0.85f,-0.2f,0, -0.80f,-0.2f,0, -0.80f,-0.7f,0,
-0.85f,-0.2f,0, -0.80f,-0.7f,0, -0.85f,-0.7f,0,
-0.95f,-0.7f,0, -0.80f,-0.7f,0, -0.80f,-0.75f,0,
-0.95f,-0.7f,0, -0.80f,-0.75f,0, -0.95f,-0.75f,0,
// ================ A (forma triangular real) ================
// Lado izquierdo inclinado
-0.25f,-0.4f,0, -0.20f,-0.4f,0, -0.175f,0.1f,0,
-0.20f,-0.4f,0, -0.15f,0.1f,0, -0.175f,0.1f,0,
// Lado derecho inclinado
-0.10f,-0.4f,0, -0.05f,-0.4f,0, -0.125f,0.1f,0,
-0.05f,-0.4f,0, -0.075f,0.1f,0, -0.125f,0.1f,0,
// Barra central
-0.19f,-0.15f,0, -0.11f,-0.15f,0, -0.11f,-0.2f,0,
-0.19f,-0.15f,0, -0.11f,-0.2f,0, -0.19f,-0.2f,0,
// ================ V (más cerrada abajo) ================
// Lado izquierdo
Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+;)
Solución "Practica_01" (1 de 1 proyecto)
Practica_01
Referencias
Dependencias externas
Archivos de encabezado
Mesh.h
Shader.h
Window.h
Archivos de origen
E02_318252333.cpp
Mesh.cpp
Shader.cpp
Window.cpp
Archivos de recursos
Herramientas de diagnóstico
100 %
No se encontraron problemas.
Línea: 1, Carácter: 1 SPC CRLF UTF-8
Lista de errores
Salida
Listo
Agregar al control de código fuente
Seleccionar repositorio
ESP LAA
01:44 a. m. 22/02/2026



```
        // ================ V (más cerrada abajo) ================
        // Lado izquierdo
        0.2f,0.5f,0,  0.25f,0.5f,0,  0.32f,0.05f,0,
        0.2f,0.5f,0,  0.32f,0.05f,0,  0.27f,0.05f,0,

        // Lado derecho
        0.4f,0.5f,0,  0.45f,0.5f,0,  0.32f,0.05f,0,
        0.4f,0.5f,0,  0.32f,0.05f,0,  0.37f,0.05f,0,


        // ================ L ================
        0.6f,0.8f,0,  0.65f,0.8f,0,  0.65f,0.3f,0,
        0.6f,0.8f,0,  0.65f,0.3f,0,  0.6f,0.3f,0,

        0.6f,0.3f,0,  0.8f,0.3f,0,  0.8f,0.25f,0,
        0.6f,0.3f,0,  0.8f,0.25f,0, 0.6f,0.25f,0,
    };

    glGenVertexArrays(1, &VAO);
    glBindVertexArray(VAO);

    glGenBuffers(1, &VBO);
    glBindBuffer(GL_ARRAY_BUFFER, VBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);

    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), 0);
    glEnableVertexAttribArray(0);

    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
}

// ======================
// SHADERS
// ======================
void AddShader(GLuint program, const char* code, GLenum type)
{
    GLuint theShader = glCreateShader(type);
    glShaderSource(theShader, 1, &code, NULL);
    glCompileShader(theShader);
    glAttachShader(program, theShader);
}

void CompileShaders()
{
    shader = glCreateProgram();
    AddShader(shader, vShader, GL_VERTEX_SHADER);
    AddShader(shader, fShader, GL_FRAGMENT_SHADER);
    glLinkProgram(shader);
}

// ======================
// MAIN
// ======================
int main()
{
    srand(time(NULL));

    if (!glfwInit()) return 1;

    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

    GLFWwindow* mainWindow = glfwCreateWindow(WIDTH, HEIGHT, "Iniciales JAVL", NULL, NULL);
    if (!mainWindow) { glfwTerminate(); return 1; }

    glfwMakeContextCurrent(mainWindow);
```



En esta parte de la práctica, ahora cambiamos de formar figuras a usar los triángulos para formar letras de forma más gruesa y notable, aunque eso cuesta trabajo, ya que letras como la J y la A, requieren un poco más de curvas o uniones, pero con esta práctica logramos acercarnos a una parte central de la misma. A su vez, ahora buscamos que los colores que se usen de fondo sean aleatorios y con una respuesta de 2 segundos lleguen a cambiar.

Ejecución

Problemas o comentarios

En un inicio, costó trabajo entender cómo funcionan las coordenadas de la pantalla al posicionar las figuras y las letras, además de ir entendiendo que ahora los colores funcionan con valores numéricos y no como anteriormente en otras programaciones (Python o C) se llamaban mediante módulos o librerías, además de entender un poco la complejidad de las matrices para los elementos a mostrar.

Conclusiones

Los ejercicios propuestos en la práctica fueron una buena idea para empezar un acercamiento al lenguaje de OpenGL, además de ir conociendo su funcionamiento entre líneas, puntos y triángulos, además de ir comprendiendo el uso y adecuación de las figuras para la programación de modelos 2D y 3D, además de que nos acercamos a una idea del modelado y su comprensión a como realmente funcionan esos programas que usan el modelado en 2D y 3D en videojuegos, animación, etc.

Referencias.

How to set the background color? (2002, 7 julio). Khronos Forums.

https://community.khronos.org/t/how-to-set-the-background-color/27016