



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

```
#include <stdio.h>
#include <string.h>
#include <glew.h>
#include <glfw3.h>
#include <math.h> // ADICIÓN: para funciones matemáticas si son necesarias

// Dimensiones de la ventana
const int WIDTH = 800, HEIGHT = 800;
GLuint VAO, VBO, shader;

// ADICIÓN: Variables para el cambio de color de fondo
float colorRojo = 1.0f;
float colorVerde = 0.0f;
float colorAzul = 0.0f;
int contadorColor = 0;
double tiempoAnterior = 0.0;
const double INTERVALO_CAMBIO = 1.0; // 1 segundo entre cambios

// VAO sirve para saber los vértices para dibujar puntos, líneas"
// VBO Memoria para guardar un

// LENGUAJE DE SHADER (SOMBRA) GLSL
// Vertex Shader
// recibir color, salida Vcolor
static const char* vshader = "
version 330
layout (location = 0) in vec3 pos;
layout (location = 1) in vec3 color;
out vec3 Vcolor;
void main()
{
    gl_Position = vec4(pos, 1.0);
    Vcolor = color;
}

";
// Fragment Shader
// recibir Vcolor
static const char* fshader = "
in vec3 Vcolor;
out vec4 FragColor;
void main()
{
    FragColor = vec4(Vcolor, 1.0);
}

";
```

The screenshot shows the Code::Blocks IDE interface. The top menu bar includes Archivo, Editor, Ver, Git, Proyecto, Compilar, Depurar, Prueba, Herramientas, Extensiones, Ventana, Ayuda, and Buscar. The title bar displays "Practica_01". The main code editor window shows the file "E01_318252333.cpp" with the following content:

```
#include <stdio.h>
#include <string.h>
#include <glew.h>
#include <glfw3.h>
#include <math.h> // ADICIÓN: para funciones matemáticas si son necesarias

// Dimensiones de la ventana
const int WIDTH = 800, HEIGHT = 800;
GLuint VAO, VBO, shader;

// ADICIÓN: Variables para el cambio de color de fondo
float colorRojo = 1.0f;
float colorVerde = 0.0f;
float colorAzul = 0.0f;
int contadorColor = 0;
double tiempoAnterior = 0.0;
const double INTERVALO_CAMBIO = 1.0; // 1 segundo entre cambios

// VAO sirve para saber los vértices para dibujar puntos, líneas"
// VBO Memoria para guardar un

// LENGUAJE DE SHADER (SOMBRA) GLSL
// Vertex Shader
// recibir color, salida Vcolor
static const char* vShader = "
#version 330
layout (location = 0) in vec3 pos;
void main()
{
    gl_Position=vec4(pos.x,pos.y,pos.z,1.0f);
}

// Fragment Shader MODIFICADO: ahora las figuras serán negras
static const char* fShader = "
#version 330
out vec4 color;

```

The right side of the interface features the "Explorador de soluciones" (Solution Explorer) panel, which lists the project structure:

- Solución "Practica_01" (1 de 1 proyecto)
 - 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

```
#version 330
out vec4 color;
void main()
{
    color = vec4(1.0f, 1.0f, 1.0f, 1.0f);
}
// 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 inferior
        -0.4f, -0.2f, 0.0f, // Vértice izquierdo inferior
        // 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
    };
}
```

```
};

//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 ejemplo
 glEnableVertexAttribArray(0);
 //agregar valores a vértices y luego declarar un nuevo glVertexAttribPointer
 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 compila el shader
    GLint result = 0;
    GLchar eLog[1024] = { 0 };
    //verificaciones y prevención de errores
    glGetShaderiv(theShader, GL_COMPILE_STATUS, &result);
    if (!result)
        printf("Error compiling shader: %s\n", eLog);
}
```

E01_310252333.cpp

```

163     GLenum eLog[1024] = { 0 };
164     //Verificaciones y prevención de errores
165     glGetShaderiv(theShader, GL_COMPILE_STATUS, &result);
166     if (!result)
167     {
168         glGetProgramInfoLog(shader, sizeof(eLog), NULL, eLog);
169         printf("El error al compilar el shader %d es: %s\n", shaderType, eLog);
170         return;
171     }
172     glAttachShader(theProgram, theShader); //Si no hubo problemas se asigna el shader a theProgram el cual asigna el código a la tarjeta
173
174     void CompileShaders()
175     {
176         shader = glCreateProgram(); //se crea un programa
177         if (!shader)
178         {
179             printf("Error creando el shader");
180             return;
181         }
182         AddShader(shader, vShader, GL_VERTEX_SHADER); //Agregar vertex shader
183         AddShader(shader, fShader, GL_FRAGMENT_SHADER); //Agregar fragment shader
184         //Para terminar de linkear el programa y ver que no tengamos errores
185         GLint result = 0;
186         GLsizei eLog[1024] = { 0 };
187         glLinkProgram(shader); //se linkean los shaders a la tarjeta gráfica
188         //verificaciones y prevención de errores
189         glGetProgramiv(shader, GL_LINK_STATUS, &result);
190         if (!result)
191         {
192             glGetProgramInfoLog(shader, sizeof(eLog), NULL, eLog);
193             printf("El error al linkear es: %s\n", eLog);
194             return;
195         }
196         glValidateProgram(shader);
197         glGetProgramiv(shader, GL_VALIDATE_STATUS, &result);
198         if (!result)
199         {
200             glGetProgramInfoLog(shader, sizeof(eLog), NULL, eLog);
201             printf("El error al validar es: %s\n", eLog);
202             return;
203         }
204     }
205
206     int main()
207     {
208         //Inicialización de GLFW
209         if (!glfwInit())
210         {
211             printf("Falló inicializar GLFW");
212             glfwTerminate();
213             return 1;
214         }
215
216         //***** LAS SIGUIENTES 4 LÍNEAS SE COMENTAN EN DADO CASO DE QUE AL USUARIO NO LE FUNCIONE LA VENTANA Y PUEDA CONOCER LA VERSIÓN DE C
217
218         //Asignando variables de GLFW y propiedades de ventana
219         glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
220         glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
221         //para solo usar el core profile de OpenGL y no tener retrocompatibilidad
222         glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
223         glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
224
225         //CREAR VENTANA
226         GLFWwindow* mainWindow = glfwCreateWindow(WIDTH, HEIGHT, "Rombo y Trapecio con fondo ciclico (Triangulos)", NULL, NULL); // ADICIÓN:
227
228         if (!mainWindow)
229         {
230             printf("Falló en crearse la ventana con GLFW");
231             glfwTerminate();
232         }
233     }

```

Linea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252

Lista de errores Salida

01:37 a.m. 22/02/2026

E01_310252333.cpp

```

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
149 int main()
150 {
151     //Inicialización de GLFW
152     if (!glfwInit())
153     {
154         printf("Falló inicializar GLFW");
155         glfwTerminate();
156         return 1;
157     }
158
159     //**** LAS SIGUIENTES 4 LÍNEAS SE COMENTAN EN DADO CASO DE QUE AL USUARIO NO LE FUNCIONE LA VENTANA Y PUEDA CONOCER LA VERSIÓN DE C
160
161     //Asignando variables de GLFW y propiedades de ventana
162     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
163     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
164     //para solo usar el core profile de OpenGL y no tener retrocompatibilidad
165     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
166     glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
167
168     //CREAR VENTANA
169     GLFWwindow* mainWindow = glfwCreateWindow(WIDTH, HEIGHT, "Rombo y Trapecio con fondo ciclico (Triangulos)", NULL, NULL); // ADICIÓN:
170
171     if (!mainWindow)
172     {
173         printf("Falló en crearse la ventana con GLFW");
174         glfwTerminate();
175     }

```

Linea: 1, Carácter: 1 TABULACIONES CRLF Windows 1252

Lista de errores Salida

01:37 a.m. 22/02/2026

E01_318252333.cpp

```

173     glfwTerminate();
174     return 1;
175 }
176 //Obtener tamaño de Buffer
177 int BufferWidth, BufferHeight;
178 glfwGetFramebufferSize(mainWindow, &BufferWidth, &BufferHeight);
179
180 //Asignar el contexto
181 glfwMakeContextCurrent(mainWindow);
182
183 //Permitir nuevas extensiones
184 glewExperimental = GL_TRUE;
185
186 if (glewInit() != GLEW_OK)
187 {
188     printf("Falló inicialización de GLEW");
189     glfwDestroyWindow(mainWindow);
190     glfwTerminate();
191     return 1;
192 }
193
194 // Asignar valores de la ventana y coordenadas
195 //Asignar Viewport
196 glfwViewport(0, 0, BufferWidth, BufferHeight);
197
198 //Llamada a las funciones creadas antes del main
199 CrearRomboYTrapecio(); // ADICIÓN: cambie la función a la nueva
200 CompileShaders();
201
202 // ADICIÓN: Inicializar tiempo
203 tiempoAnterior = glfwGetTime();
204
205 //Loop mientras no se cierra la ventana
206 while (!glfwWindowShouldClose(mainWindow))
207 {
208     //Recibir eventos del usuario
209     glfwPollEvents();
210
211     // ADICIÓN: Cambiar color de fondo cíclicamente
212     double tiempoActual = glfwGetTime();
213     if (tiempoActual - tiempoAnterior >= INTERVALO_CAMBIO)
214     {
215         contadorColor = (contadorColor + 1) % 3;
216
217         if (contadorColor == 0) // Rojo
218         {
219             colorRojo = 1.0f;
220             colorVerde = 0.0f;
221             colorAzul = 0.0f;
222         }
223         else if (contadorColor == 1) // Verde
224         {
225             colorRojo = 0.0f;
226             colorVerde = 1.0f;
227             colorAzul = 0.0f;
228         }
229         else if (contadorColor == 2) // Azul
230         {
231             colorRojo = 0.0f;
232             colorVerde = 0.0f;
233             colorAzul = 1.0f;
234         }
235
236         tiempoAnterior = tiempoActual;
237     }
238
239     //Limpia la ventana (ADICIÓN: usando las variables de color)
240     glClearColor(colorRojo, colorVerde, colorAzul, 1.0f);
241     glClear(GL_COLOR_BUFFER_BIT);
242
243     qUseProgram(shader);
244 }
245
246 //No se encontraron problemas.

```

Explorador de soluciones

- Solución "Práctica_01" (1 de 1 proyecto)
 - Práctica_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

Lista de errores Salida

01:37 a.m. 22/02/2026

E01_318252333.cpp

```

208     //Recibir eventos del usuario
209     glfwPollEvents();
210
211     // ADICIÓN: Cambiar color de fondo cíclicamente
212     double tiempoActual = glfwGetTime();
213     if (tiempoActual - tiempoAnterior >= INTERVALO_CAMBIO)
214     {
215         contadorColor = (contadorColor + 1) % 3;
216
217         if (contadorColor == 0) // Rojo
218         {
219             colorRojo = 1.0f;
220             colorVerde = 0.0f;
221             colorAzul = 0.0f;
222         }
223         else if (contadorColor == 1) // Verde
224         {
225             colorRojo = 0.0f;
226             colorVerde = 1.0f;
227             colorAzul = 0.0f;
228         }
229         else if (contadorColor == 2) // Azul
230         {
231             colorRojo = 0.0f;
232             colorVerde = 0.0f;
233             colorAzul = 1.0f;
234         }
235
236         tiempoAnterior = tiempoActual;
237     }
238
239     //Limpia la ventana (ADICIÓN: usando las variables de color)
240     glClearColor(colorRojo, colorVerde, colorAzul, 1.0f);
241     glClear(GL_COLOR_BUFFER_BIT);
242
243     qUseProgram(shader);
244 }
245
246 //No se encontraron problemas.

```

Explorador de soluciones

- Solución "Práctica_01" (1 de 1 proyecto)
 - Práctica_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

Lista de errores Salida

01:37 a.m. 22/02/2026

The screenshot shows the Microsoft Visual Studio interface. The code editor window displays a file named 'E01_318252333.cpp' containing C++ code for rendering a diamond and a trapezoid. The code uses OpenGL functions like glClear, glDrawArrays, and glUseProgram. The solution explorer on the right shows the project structure with files like Mesh.h, Shader.h, Window.h, Mesh.cpp, Shader.cpp, and Window.cpp.

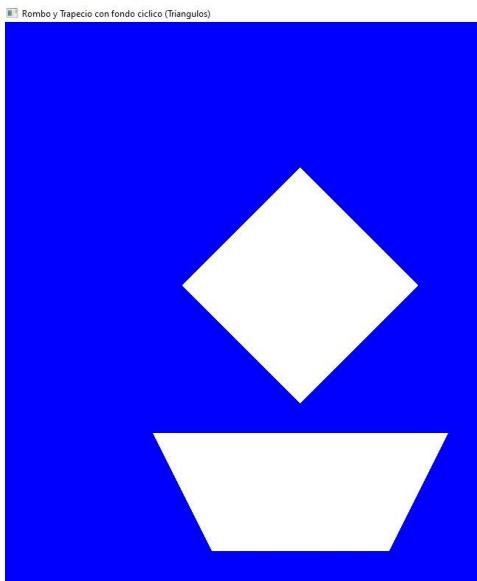
```

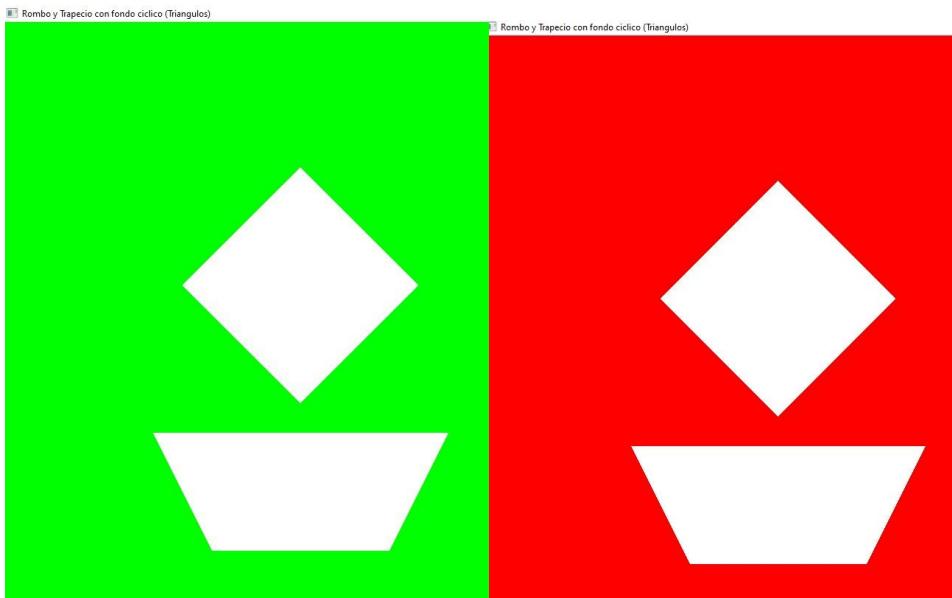
223     else if (contadorColor == 1) // Verde
224     {
225         colorRojo = 0.0f;
226         colorVerde = 1.0f;
227         colorAzul = 0.0f;
228     }
229     else if (contadorColor == 2) // Azul
230     {
231         colorRojo = 0.0f;
232         colorVerde = 0.0f;
233         colorAzul = 1.0f;
234     }
235     tiempoAnterior = tiempoActual;
236 }
237
238 //Limpiar la ventana (ADICIÓN: usando las variables de color)
239 glClearColor(colorRojo, colorVerde, colorAzul, 1.0f);
240 glClear(GL_COLOR_BUFFER_BIT);
241
242 glUseProgram(shader);
243
244 glBindVertexArray(VAO);
245 glDrawArrays(GL_TRIANGLES, 0, 12); // MODIFICACIÓN: cambié de GL_LINES a GL_TRIANGLES y de 16 a 12 vértices (6 para rombo + 6 para trapecio)
246 glBindVertexArray(0);
247
248 glUseProgram(0);
249
250 glfwSwapBuffers(mainWindow);
251
252 //NO ESCRIBIR NINGUNA LÍNEA DESPUÉS DE glfwSwapBuffers(mainWindow);
253
254
255
256
257
258 return 0;
100 %

```

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.

Practica_01

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <time.h>
5 #include <glew.h>
6 #include <glfw3.h>
7
8 const int WIDTH = 800, HEIGHT = 800;
9
10 GLuint VAO, VBO, shader;
11
12 float colorRojo = 0.0f;
13 float colorVerde = 0.0f;
14 float colorAzul = 0.0f;
15
16 double tiempoAnterior = 0.0;
17
18 // =====
19 // VERTEX SHADER
20 // =====
21 static const char* vShader = "
22 #version 330 core
23 layout (location = 0) in vec3 pos;
24 void main()
25 {
    gl_Position = vec4(pos, 1.0);
}
26
27 // =====
28 // FRAGMENT SHADER
29 // =====
30 static const char* fShader = "
31 #version 330 core
32 out vec4 color;
33 void main()
34 {
    color = vec4(1.0, 1.0, 1.0, 1.0);
}
35
36 ";
37
38 // =====
39 // FUNCION PARA CREAR LETRAS
40 // =====
41 void CrearIniciales()
42 {
    GLfloat vertices[] = {
43
        // ===== J =====
44        -0.85f,-0.2f,0, -0.80f,-0.2f,0, -0.80f,-0.7f,0,
45        -0.85f,-0.2f,0, -0.80f,-0.7f,0, -0.85f,-0.7f,0,
46
47        -0.95f,-0.7f,0, -0.80f,-0.7f,0, -0.80f,-0.75f,0,
48        -0.95f,-0.7f,0, -0.80f,-0.75f,0, -0.95f,-0.75f,0,
49
50
51        // ===== A (forma triangular real) =====
52
53        // Lado izquierdo inclinado
54        -0.25f,-0.4f,0, -0.20f,-0.4f,0, -0.175f,0.1f,0,
55        -0.20f,-0.4f,0, -0.15f,0.1f,0, -0.175f,0.1f,0,
56
57        // Lado derecho inclinado
58        -0.10f,-0.4f,0, -0.05f,-0.4f,0, -0.125f,0.1f,0,
59        -0.05f,-0.4f,0, -0.075f,0.1f,0, -0.125f,0.1f,0,
60
61
62        // Barra central
63        -0.19f,-0.15f,0, -0.11f,-0.15f,0, -0.11f,-0.2f,0,
64        -0.19f,-0.15f,0, -0.11f,-0.2f,0, -0.19f,-0.2f,0,
65
66
67        // ===== V (más cerrada abajo) =====
68
69        // Lado izquierdo
70
71    };
72
73    // No se encontraron problemas.

```

Explorador de soluciones

- Solución "Practica_01" (1 de 1 proyecto)
 - 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

Línea: 1, Carácter: 1 SPC CRLF UTF-8

Practica_01

```

36
37     color = vec4(1.0, 1.0, 1.0, 1.0);
38 }
39
40 // =====
41 // FUNCION PARA CREAR LETRAS
42 // =====
43 void CrearIniciales()
44 {
    GLfloat vertices[] = {
45
        // ===== J =====
46        -0.85f,-0.2f,0, -0.80f,-0.2f,0, -0.80f,-0.7f,0,
47        -0.85f,-0.2f,0, -0.80f,-0.7f,0, -0.85f,-0.7f,0,
48
49        -0.95f,-0.7f,0, -0.80f,-0.7f,0, -0.80f,-0.75f,0,
50        -0.95f,-0.7f,0, -0.80f,-0.75f,0, -0.95f,-0.75f,0,
51
52
53        // ===== A (forma triangular real) =====
54
54        // Lado izquierdo inclinado
55        -0.25f,-0.4f,0, -0.20f,-0.4f,0, -0.175f,0.1f,0,
56        -0.20f,-0.4f,0, -0.15f,0.1f,0, -0.175f,0.1f,0,
57
58        // Lado derecho inclinado
59        -0.10f,-0.4f,0, -0.05f,-0.4f,0, -0.125f,0.1f,0,
60        -0.05f,-0.4f,0, -0.075f,0.1f,0, -0.125f,0.1f,0,
61
62
63        // Barra central
64        -0.19f,-0.15f,0, -0.11f,-0.15f,0, -0.11f,-0.2f,0,
65        -0.19f,-0.15f,0, -0.11f,-0.2f,0, -0.19f,-0.2f,0,
66
67
68        // ===== V (más cerrada abajo) =====
69
70        // Lado izquierdo
71
72    };
73
74    // No se encontraron problemas.

```

Explorador de soluciones

- Solución "Practica_01" (1 de 1 proyecto)
 - 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

Línea: 1, Carácter: 1 SPC CRLF UTF-8

The screenshot displays two instances of Microsoft Visual Studio running side-by-side. Both instances show the same project structure and code content.

Project Structure:

- Solution Explorer:** Shows a single solution named "Practica_01" containing one project.
- File List:** Includes files like Mesh.h, Shader.h, Window.h, E02_318252333.cpp, Mesh.cpp, Shader.cpp, and Window.cpp.
- Resource List:** Shows resources like Archivos de recursos.

Code Editor Content (Left):

```
69 // ===== V (más cerrada abajo) =====
70 // Lado izquierdo
71 0.2f, 0.5f, 0, 0.25f, 0.5f, 0, 0.32f, 0, 0.65f, 0,
72 0.2f, 0.5f, 0, 0.32f, 0.05f, 0, 0.27f, 0.65f, 0,
73
74 // Lado derecho
75 0.4f, 0.5f, 0, 0.45f, 0.5f, 0, 0.32f, 0, 0.65f, 0,
76 0.4f, 0.5f, 0, 0.32f, 0.05f, 0, 0.37f, 0.65f, 0,
77
78
79 // ===== L =====
80 0.6f, 0.8f, 0, 0.65f, 0.8f, 0, 0.65f, 0.3f, 0,
81 0.6f, 0.8f, 0, 0.65f, 0.3f, 0, 0.6f, 0.3f, 0,
82
83 0.6f, 0.3f, 0, 0.8f, 0.3f, 0, 0.8f, 0.25f, 0,
84 0.6f, 0.3f, 0, 0.8f, 0.25f, 0, 0.6f, 0.25f, 0,
85
86 };
87
88 glGenVertexArrays(1, &VAO);
89 glBindVertexArray(VAO);
90
91 glGenBuffers(1, &VBO);
92 glBindBuffer(GL_ARRAY_BUFFER, VBO);
93 glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
94
95 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), 0);
96 glEnableVertexAttribArray(0);
97
98 glBindBuffer(GL_ARRAY_BUFFER, 0);
99 glBindVertexArray(0);
100
101 // ======
102 // SHADERS
103 // ======
104 // ======
```

Code Editor Content (Right):

```
102 // =====
103 // SHADERS
104 // =====
105 void AddShader(GLuint program, const char* code, GLenum type)
106 {
107     GLuint theShader = glCreateShader(type);
108     glShaderSource(theShader, 1, &code, NULL);
109     glCompileShader(theShader);
110     glAttachShader(program, theShader);
111 }
112
113 void CompileShaders()
114 {
115     shader = glCreateProgram();
116     AddShader(shader, vShader, GL_VERTEX_SHADER);
117     AddShader(shader, fShader, GL_FRAGMENT_SHADER);
118     glLinkProgram(shader);
119 }
120
121 // =====
122 // MAIN
123 // =====
124 int main()
125 {
126     srand(time(NULL));
127
128     if (!glfwInit()) return 1;
129
130     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
131     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
132     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
133
134     GLFWwindow* mainWindow = glfwCreateWindow(WIDTH, HEIGHT, "Iniciales JAVL", NULL, NULL);
135     if (!mainWindow) glfwTerminate(); return 1;
136
137     glfwMakeContextCurrent(mainWindow);
138 }
```

Common UI Elements:

- Top Bar:** Archivo, Editar, Ver, Git, Proyecto, Compilar, Depurar, Prueba, Herramientas, Extensiones, Ventana, Ayuda, Buscar.
- Status Bar:** Línea: 1, Carácter: 1, SPC, CRLF, UTF-8.
- Taskbar:** Shows various open applications like File Explorer, Task Manager, and browser tabs.
- System Tray:** Shows icons for battery, signal, and date/time (01:44 a.m., 22/02/2026).

```

137 glfwMakeContextCurrent(mainWindow);
138 glewExperimental = GL_TRUE;
139 if (glewInit() != GLEW_OK) return 1;
140
141 glfwViewport(0, 0, WIDTH, HEIGHT);
142
143 CrearIniciales();
144 CompileShaders();
145
146 tiempoAnterior = glfwGetTime();
147
148 while (!glfwWindowShouldClose(mainWindow))
149 {
150     glfwPollEvents();
151
152     double tiempoActual = glfwGetTime();
153
154     if (tiempoActual - tiempoAnterior >= 2.0)
155     {
156         colorRojo = (float)rand() / RAND_MAX;
157         colorVerde = (float)rand() / RAND_MAX;
158         colorAzul = (float)rand() / RAND_MAX;
159         tiempoAnterior = tiempoActual;
160     }
161
162     glClearColor(colorRojo, colorVerde, colorAzul, 1.0f);
163     glClear(GL_COLOR_BUFFER_BIT);
164
165     glUseProgram(shader);
166     glBindVertexArray(VAO);
167     glDrawArrays(GL_TRIANGLES, 0, 66);
168     glBindVertexArray(0);
169
170     glfwSwapBuffers(mainWindow);
171 }
172
173 glfwTerminate();
174 return 0;
175 }

```

Lista de errores Salida

01:45 a.m. 22/02/2026

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>