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 1

NOMBRE COMPLETO: Gómez Enríquez Agustín

Nº de Cuenta: 317031405

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 5

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 24 de agosto del 2025

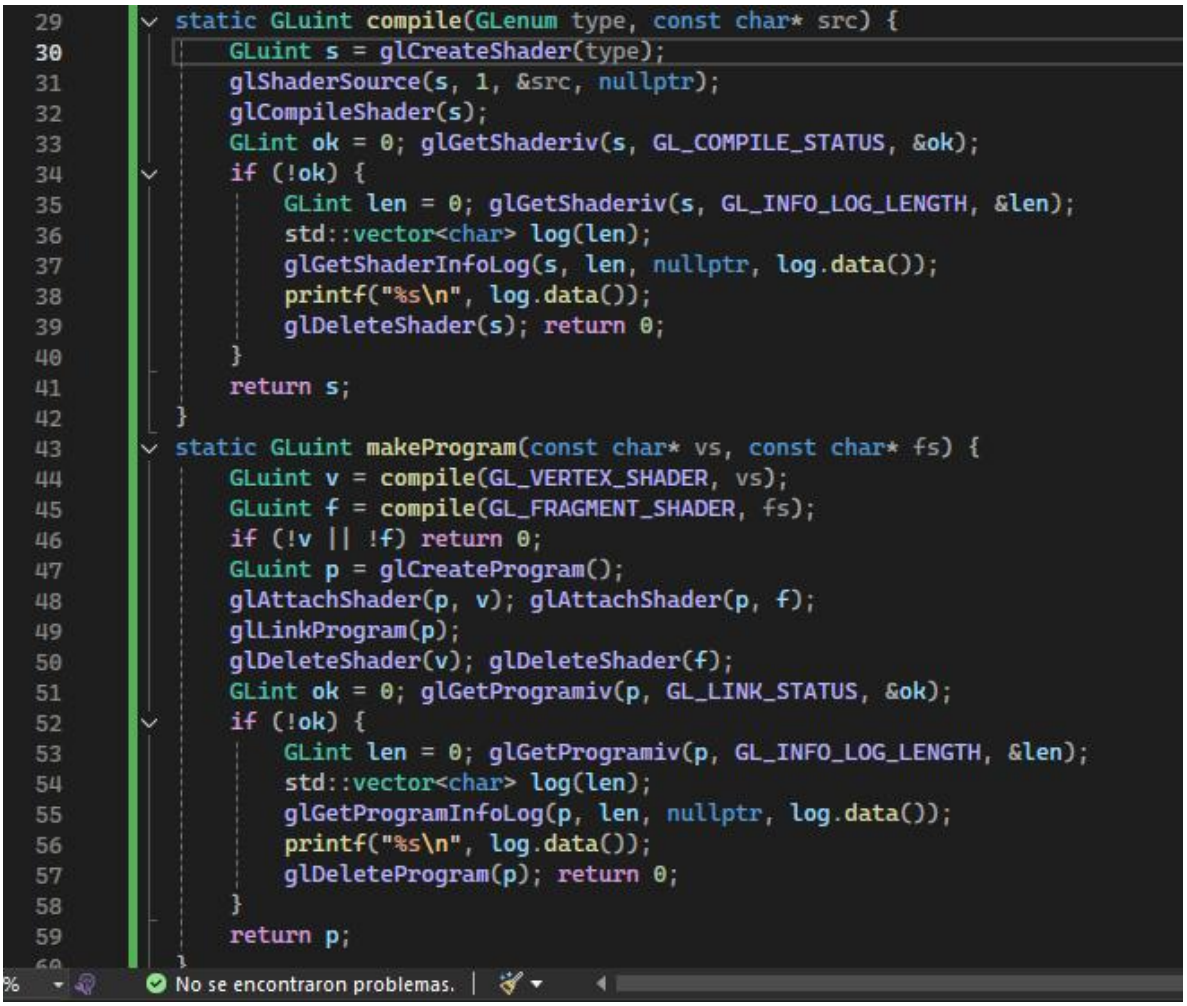
CALIFICACIÓN: _____

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.

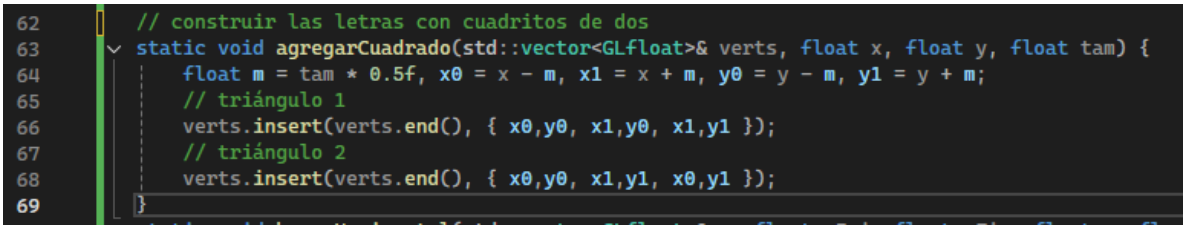
```
11  // shaders para dibujar los triángulos
12  static const char* vtxSrc = R"(
13  #version 430 core
14  layout (location = 0) in vec2 aPos;
15  void main(){
16      gl_Position = vec4(aPos, 0.0, 1.0);
17  }
18  )";
19
20  static const char* fragSrc = R"(
21  #version 430 core
22  out vec4 FragColor;
23  uniform vec3 uColor;
24  void main(){
25      FragColor = vec4(uColor, 1.0);
26  }
27  )";
```

Empezamos creando nuestro entorno de trabajo con Vertex Shader y Fragmente shader. Donde definimos que los triángulos con los que estaremos trabajando deberán aparecer en las coordenadas que tal cual se definan. Asi mismo se debe ejecutar cada pixel del triangulo que igualmente deben contener un solo color gracias al uniform uColor.



Para compilar y linkear CPU con GPU, creamos el objeto shader vacio y le pasamos el código fuente con “glShaderSource(s, 1, &src, nullptr);”, luego lo compilamos en la GPU y verificamos que se ejecutara correctamente.

Después “glLinkProgram(p);” une VS+FS en 1 ejecutable GPU.



Para no estar creando muchos vértices, creamos nuestra función “agregarCuadrado” que inserta 6 vértices centrados en los eje X y Y.

```cpp
static void barraHorizontal(std::vector<GLfloat>& v, float xIni, float xFin, float y, float tam) {
    float paso = tam * 0.9f;
    for (float x = xIni; x <= xFin; x += paso) agregarCuadrado(v, x, y, tam);
}
static void barraVertical(std::vector<GLfloat>& v, float x, float yIni, float yFin, float tam) {
    float paso = tam * 0.9f;
    for (float y = yIni; y <= yFin; y += paso) agregarCuadrado(v, x, y, tam);
}
```

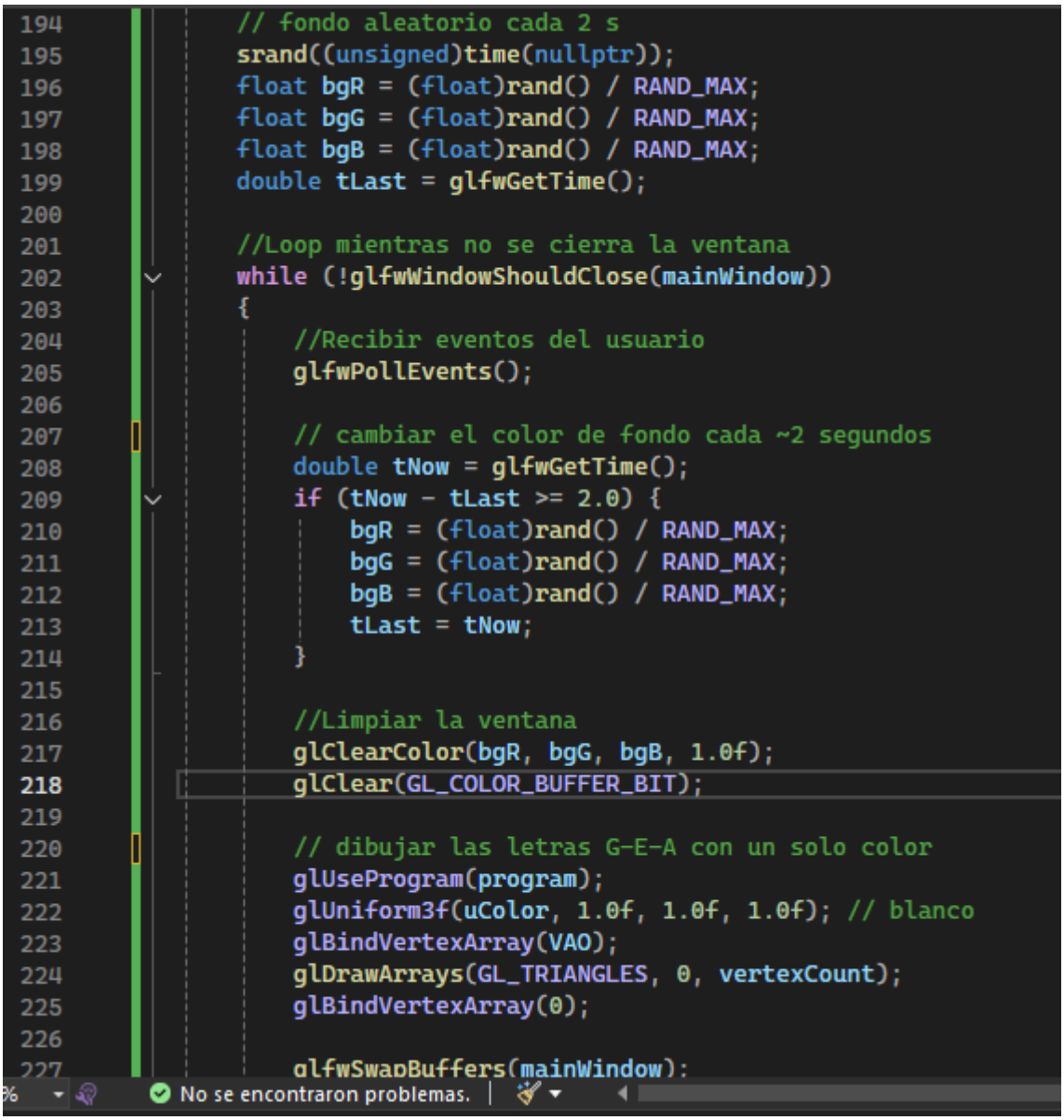
Para empezar a crear nuestras líneas (las letras), hacemos uso de unos helpers que irán colocando cuadritos de forma horizontal y vertical.

```cpp
static void crearLetrasGEA(std::vector<GLfloat>& vertices) {
    const float t = 0.055f;   // grosor de la letra
    const float yTop = 0.60f;
    const float yBot = -0.60f;

    const float letterW = 0.45f;  // ancho por letra
    const float gap = 0.14f;  // separación entre letras
    const float left = -0.95f; // margen izquierdo

    float x0G = left, x1G = x0G + letterW;
    float x0E = x1G + gap, x1E = x0E + letterW;
    float x0A = x1E + gap, x1A = x0A + letterW;

```

Aquí definimos el formato de como queremos nuestras letras, como el grosor de las letras, ancho que queremos que se vea la línea. Extra a esto agregue la separación de letras porque se llegaban a empalmar ya que no había establecido correctamente las coordenadas de las letras y con gap se crea una separación equitativa.

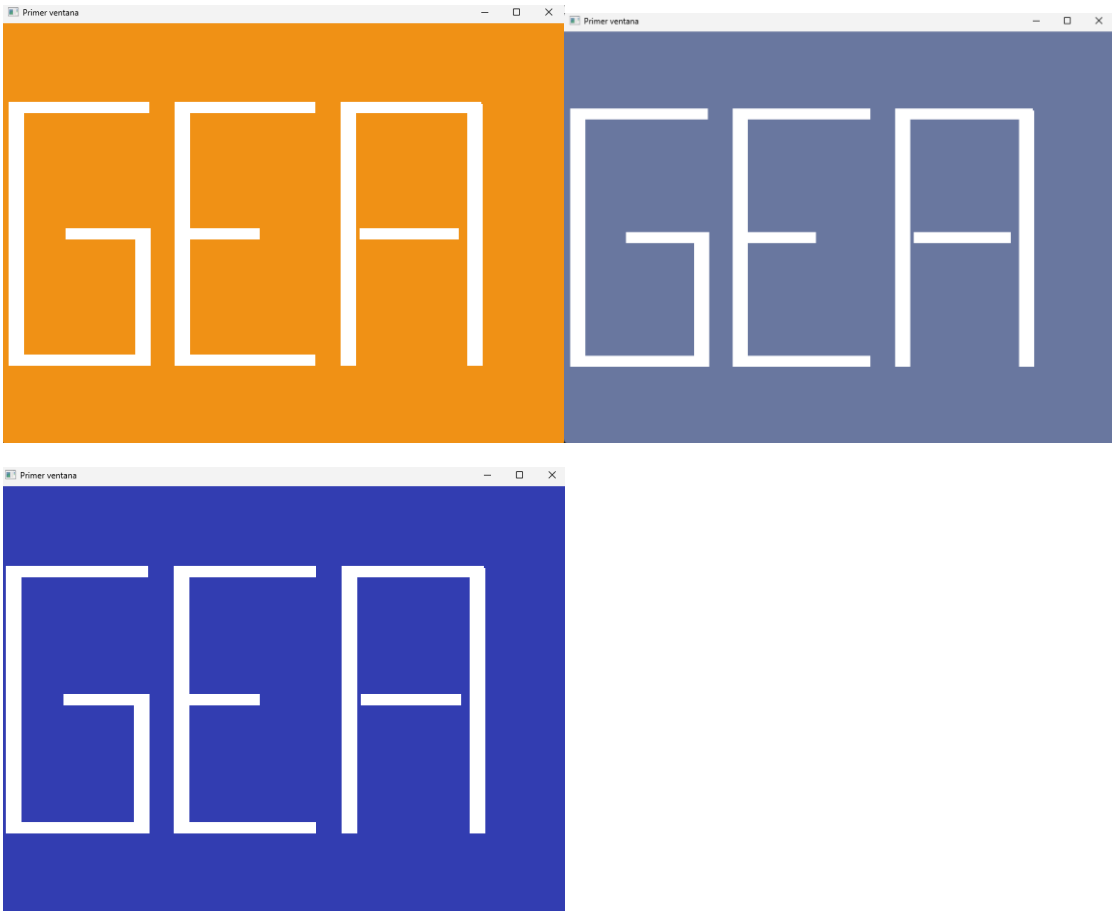
```cpp
    // G
    {
        float yMid = (yTop + yBot) * 0.5f;
        barraHorizontal(vertices, x0G, x1G, yTop, t);
        barraVertical(vertices, x0G, yBot, yTop, t);
        barraHorizontal(vertices, x0G, x1G, yBot, t);
        barraVertical(vertices, x1G, yBot, yMid - 0.02f, t);
        barraHorizontal(vertices, x0G + (x1G - x0G) * 0.45f, x1G, yMid, t);
    }
    // E
    {
        float yMid = (yTop + yBot) * 0.5f;
        barraVertical(vertices, x0E, yBot, yTop, t);
        barraHorizontal(vertices, x0E, x1E, yTop, t);
        barraHorizontal(vertices, x0E, x0E + (x1E - x0E) * 0.65f, yMid, t);
        barraHorizontal(vertices, x0E, x1E, yBot, t);
    }
    // A
    {
        float yMid = (yTop + yBot) * 0.5f;
        barraVertical(vertices, x0A, yBot, yTop, t);
        barraVertical(vertices, x1A, yBot, yTop, t);
        barraHorizontal(vertices, x0A, x1A, yTop, t);
        barraHorizontal(vertices,
            x0A + (x1A - x0A) * 0.15f,
            x1A - (x1A - x0A) * 0.15f, yMid, t);
    }
}
```

Ya por último creamos nuestros 3 bloques de código para dibujar las letras G, E y A que son las iniciales de mi nombre y apellido.



Dentro del main vamos a configurar que el color de la ventana se cambie cada 2 segundos (para que el usuario pueda apreciar el cambio de colores), esto lo manejamos con un loop para que mientras la ventana se encuentre abierta se estará repitiendo. Con “glClearColor(bgR, bgG, bgB, 1.0f);” fijamos el color del fondo y con “glClear(GL_COLOR_BUFFER_BIT);” lo vamos a ir coloreando. Para hacer que los colores del fondo sean aleatorios usamos rand para cada color de RGB, de esta manera nos estará dando una combinación de estos aleatorio.

Ejecución



De esta manera se muestra como cada2 segundos el color del fondo se va actualizando de forma aleatoria.

2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

De las primeras versiones que funcionaron tuve problemas con el acomodo de las letras ya que me estaba familiarizando con las coordenadas, además de que me faltaban corregir detalles de las letras. Esto se corrigió al parametrizar un **layout**: letterW (ancho), gap (separación), left (margen) y yTop/yBot (alto), y reconstruir las letras con esos parámetros. Para la cuestión estética solo se agregó la línea vertical faltante.

3.- Conclusión:

Al realizar la practica de forma individual sin los conocimientos previos del laboratorio fue complicado comprender el uso de visual studio y OpenGL. Asi mismo es complicado el uso del entorno de trabajo mas que nada porque no suele existir explicaciones de esta programación en español. Sin embargo, fue posible realizar esta practica por los conocimientos previos de c++. Y la búsqueda de tutoriales.

Agregando una opinión al curso, considero que estaría bien para esta primera practica agregar parte del código sobre el trazado de triángulos (cuadrados), y la actualización de pantalla de los colores.

Para concluir, esta practica fue un desafío al volver a trabajar con C++, y que se tuvieron que investigar conceptos nuevos de grafica y el descubrimiento de funciones nuevas como las que se incluyen en la librería de GL. Sin embargo, esta primera demostración de los gráficos en 2D es un buen acercamiento a la cuestión de animación y uso de la tarjeta gráfica.

Bibliografía en formato APA

- Programación, I. y. [@informaticayprogramacion2169]. (n.d.). *2 - Visual Studio 2022. Primera aplicación y conceptos básicos* [Video]. Youtube. Retrieved Agosto 29, 2025, from https://www.youtube.com/watch?v=uzhfIFeLctY
- Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1995). *Computer graphics: Principles and practice in C* (2nd ed.). Addison-Wesley.
- Cabanes, N. [@Nacho_Cabanes]. (n.d.). *Visual Studio 2022 para principiantes en C#* [Video]. Youtube. Retrieved Agosto 29, 2025, from https://www.youtube.com/watch?v=7zs1OQ_KA7I