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: Carandia Lorenzo Brenda Fernanda

Nº de Cuenta: 319018961

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 05

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 15 de febrero de 2025

CALIFICACIÓN: _____

PRÁCTICA 01: INTRODUCCIÓN A OPENGL

ACTIVIDADES REALIZADAS

1. **Ventana cambia el color de fondo de forma random tomando rango de colores RGB y con una periodicidad de 2 segundos.**

Bloque de código

- LIBRERÍA Y VARIABLES

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <glew.h>
4   #include <glfw3.h>
5   #include <cstdlib>  // Para rand() y srand()
6   #include <ctime>    // Para time()
7   //Dimensiones de la ventana
8   const int WIDTH = 800, HEIGHT = 600;
9   GLuint VAO, VBO, shader;
10  float rojo , verde, azul;
11  double tanterior = glfwGetTime();
```

Se agregaron las bibliotecas `<cstdlib>` y `<ctime>`, para poder hacer uso de la función `rand()`, `srand()` y `time ()`, las cuales nos ayudaran a generar números aleatorios y a su vez, colores diferentes. Se declaran las variables rojo, verde y azul para controlar el color de fondo, así como la funcion `glfwGetTime()` guardada en la variable `tanterior` para medir el tiempo transcurrido.

- NÚMEROS ALEATORIOS Y PRIMER COLOR

```
274
275     // Generar numeros aleatorios con la hora actual, para el cambio de color
276     srand(static_cast<unsigned int>(time(0)));
277
278     //Inicializa el color de la primera ventana, para que no sea negro
279     rojo = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
280     verde = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
281     azul = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
```

Se hace uso de `srand` para poder hacer que cada vez que ejecutemos el programa el color de fondo de la ventana sea diferente (el valor), ya que si no se generarían los mismos colores (números) siempre que se ejecute. Esto lo hace mediante la función `time()` que obtiene la hora actual y con ayuda de `static_cast<unsigned int>` se pasa a un número entero sin signo, el cual será el valor de srand.

En la segunda parte, se inicializa el color de fondo de manera aleatoria, para que no siempre sea de color negro como estaba. Para esto a las variables rojo, verde y azul se les da un valor aleatorio dentro del rango RGB que va de [0.0,1.0]. Para esto se hace uso de `rand()` *para generar números aleatorios enteros que van de un rango de [0, RAND_MAX], pasando a un tipo float con* `static_cast<float>(rand())`*, pero como estos valores no están dentro del rango requerido se tiene que normalizar, por eso se divide entre el número máximo RAND_MAX, para que pueda quedar dentro del rango; esto pasa para las variables* `rojo, verde y azul.`

- **CAMBIO CADA DOS SEG Y COLOR**

```
291     while (!glfwWindowShouldClose(mainWindow))
292     {
293         // Recibir eventos del usuario
294         glfwPollEvents();
295 
296         double tactual = glfwGetTime();
297 
298         // Cambiar el color de fondo cada 2 segundos
299         if (tactual- tanterior >= 2.0)
300         {
301             rojo = static_cast<float>(rand()) / static_cast<float>(RAND_MAX); //Genera elvalor de rojo aleatorio
302             verde = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);//Genera el valor de verde aleatorio
303             azul = static_cast<float>(rand()) / static_cast<float>(RAND_MAX); //Genera el valor de azul aleatorio
304             tanterior = tactual;
305         }
306 
307         //Limpiar la ventana
308         glClearColor(rojo,verde, azul,1.0f);
309         glClear(GL_COLOR_BUFFER_BIT);
```

Para el cambio de color reciclamos el ejercicio de clase, en donde se declara la variable `tactual` *para obtener el tiempo actual con* `glfwGetTime()`*. Luego, se calcula la diferencia entre este tiempo y* `tanterior`*, que almacena la última actualización de color. Si han transcurrido al menos dos segundos, entonces el color cambia, esto es el valor de las variables rojo, verde y azul es diferente, esto debido a que* `rand()` *les proporciona valores random que se tiene que normalizar para estar dentro del parámetro de [0.0,1.0] ,de colores RGB. Y por último la variable* `tanterior` *se actualiza para continuar con el ciclo.*

2. **3 letras iniciales de sus nombres creadas a partir de triángulos, todas las letras son del mismo color.**

Bloque de código

```
34  void CrearTriangulo()
35  {
36      GLfloat vertices[] = {
37
38          // Letra F
39          -0.4f, 0.7f, 0.0f,  //Parte superior de la letra
40          -0.4f, 0.6f, 0.0f,  //Empezando de derecha a izquierda
41          -0.5f, 0.7f, 0.0f,
42
43          -0.5f, 0.7f, 0.0f,
44          -0.4f, 0.6f, 0.0f,
45          -0.6f, 0.6f, 0.0f,
46
47          -0.5f, 0.7f, 0.0f,
48          -0.6f, 0.6f, 0.0f,
49          -0.7f, 0.7f, 0.0f,
50
51          -0.7f, 0.7f, 0.0f,  //Empieza el tronco, la parte vertical
52          -0.6f, 0.6f, 0.0f,
53          -0.7f, 0.5f, 0.0f,
54
55          -0.6f, 0.6f, 0.0f,
56          -0.7f, 0.5f, 0.0f,
57          -0.6f, 0.5f, 0.0f,
58
59          -0.6f, 0.5f, 0.0f,  //Parte horizontal menor
60          -0.5f, 0.5f, 0.0f,
61          -0.5f, 0.4f, 0.0f,
62
63          -0.6f, 0.5f, 0.0f,
64          -0.5f, 0.4f, 0.0f,
65          -0.6f, 0.4f, 0.0f,
66
67          -0.7f, 0.5f, 0.0f,  // Termina la parte vertical
68          -0.6f, 0.4f, 0.0f,
69          -0.6f, 0.5f, 0.0f,
70
71          -0.7f, 0.5f, 0.0f,
72          -0.6f, 0.4f, 0.0f,
73          -0.7f, 0.2f, 0.0f,
74
75          -0.7f, 0.2f, 0.0f,
76          -0.6f, 0.4f, 0.0f,
77          -0.6f, 0.2f, 0.0f,
```

```
79          //Letra C
80
81          0.7f, 0.54f, 0.0f,  //Empieza la parte curva
82          0.6f, 0.54f, 0.0f,  // superior de izquierda
83          0.7f, 0.6f, 0.0f,   //a derecha de la letra
84
85          0.7f, 0.6f, 0.0f,
86          0.6f, 0.54f, 0.0f,
87          0.6f, 0.6f, 0.0f,
88
89          0.7f, 0.6f, 0.0f,
90          0.6f, 0.6f, 0.0f,
91          0.6f, 0.7f, 0.0f,
92
93          0.6f, 0.7f, 0.0f,
94          0.6f, 0.6f, 0.0f,
95          0.4f, 0.6f, 0.0f,
96
97          0.6f, 0.7f, 0.0f,
98          0.4f, 0.6f, 0.0f,
99          0.4f, 0.7f, 0.0f,
100
101         0.4f, 0.7f, 0.0f,
102         0.4f, 0.6f, 0.0f,
103         0.3f, 0.6f, 0.0f,
104
105         0.3f, 0.6f, 0.0f,   //Empieza la parte vertical
106         0.4f, 0.6f, 0.0f,   //de la letra
107         0.3f, 0.3f, 0.0f,
108
109         0.4f, 0.6f, 0.0f,
110         0.3f, 0.3f, 0.0f,
111         0.4f, 0.3f, 0.0f,

113         0.3f, 0.3f, 0.0f,
114         0.4f, 0.3f, 0.0f,
115         0.4f, 0.2f, 0.0f,
116
117         0.4f, 0.3f, 0.0f,   //Empieza la parte curva
118         0.4f, 0.2f, 0.0f,   //inferior de la
119         0.6f, 0.2f, 0.0f,   //letra
120
121         0.4f, 0.3f, 0.0f,
122         0.6f, 0.2f, 0.0f,
123         0.6f, 0.3f, 0.0f,
124
125         0.6f, 0.3f, 0.0f,
126         0.6f, 0.2f, 0.0f,
127         0.7f, 0.3f, 0.0f,
128
129         0.6f, 0.36f, 0.0f,
130         0.6f, 0.3f, 0.0f,
131         0.7f, 0.3f, 0.0f,
132
133         0.7f, 0.3f, 0.0f,
134         0.6f, 0.36f, 0.0f,
135         0.7f, 0.36f, 0.0f,
```

```
137         //letra L
138         -0.1f, -0.2f, 0.0f,  // Parte horizontal de la
139          0.0f, -0.2f, 0.0f,  //Letra
140         -0.1f, -0.7f, 0.0f,
141
142          0.0f, -0.2f, 0.0f,
143         -0.1f, -0.7f, 0.0f,
144          0.0f, -0.7f, 0.0f,
145
146          0.0f, -0.6f, 0.0f,  //Parte vertical de la
147          0.0f, -0.7f, 0.0f, // letra
148          0.2f, -0.7f, 0.0f,
149
150          0.2f, -0.7f, 0.0f,
151          0.0f, -0.6f, 0.0f,
152          0.2f, -0.6f, 0.0f,
153
154     };
```

Para la construcción de las letras, elegí la F de Fernanda la C de Carandia y la L de Lorenzo, para poder respetar las dimensiones, de los ejes (x,y) que van de (-1,1), hice uso del programa GeoGebra, para poder ubicarme en las dimensiones correspondientes sin pasarme. Simplemente se van poniendo los puntos de cada triangulo que constituye a cada letra. Para la F se usaron 10 triángulos, para la letra C 14 triángulos y para la L 4 triángulos.

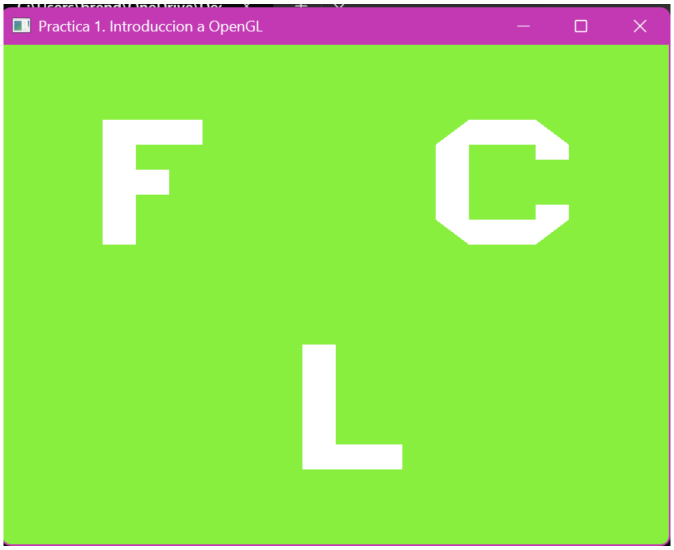
```
23      //recibir Vcolor y dar de salida color
24      static const char* fShader = "
25      #version 330
26      out vec4 color;
27      void main()
28      {
29          color = vec4(1.0f,1.0f,1.0f,1.0f);
30      }";
```

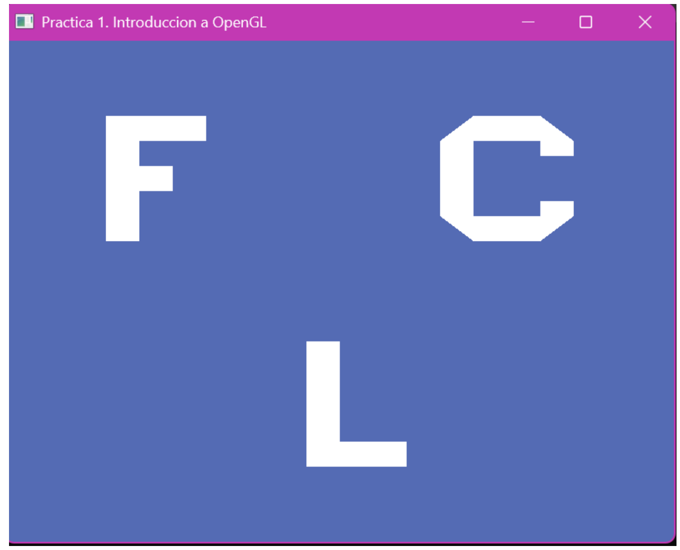
El color de fondo de todas las letras, es de color blanco, ya que es unos de los colores que puede combinar con la gama de colores RGB y evitar que en un momento dado, el fondo de la ventana y el de la letra sean iguales y se pierdan.

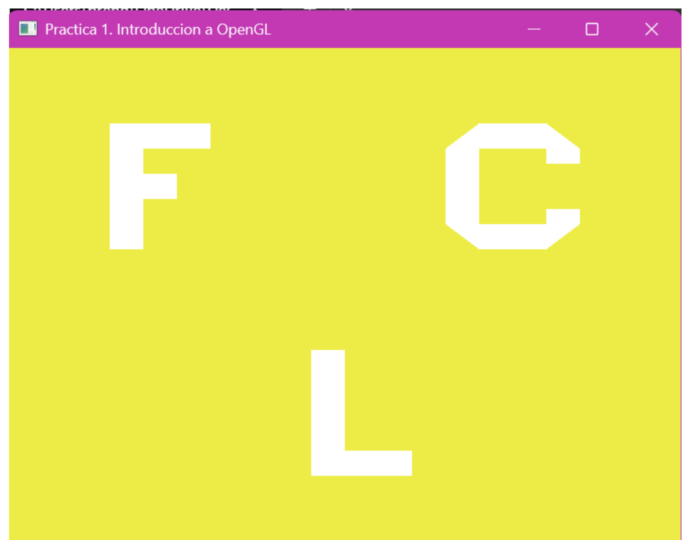
```
313             glBindVertexArray(VAO);
314             glDrawArrays(GL_TRIANGLES,0,84);
315             glBindVertexArray(0);
```

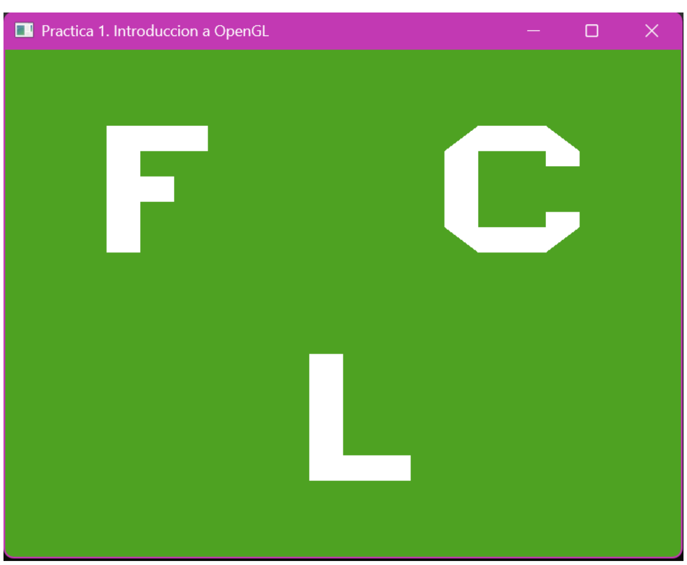
En este caso, para formar todas las letras elegidas se necesitaron 28 triángulos, los cuales son los 84 puntos de cada coordenada que se tendrá que dibujar; por eso se cambia el valor en el glDrawArrays a 84.

3. Ejecución del programa. (Cambio de color y letras)









Se puede ver las letras formadas por triángulos de color blanco, y las diferentes tonalidades de los colores del rango RGB, cada dos segundos.

4. Problemas presentados.

En general, no existieron muchos problemas, primero con la variable time que había declarado para el ejercicio de clase, tenia un problema puesto que time era una palabra reservada dentro de `<ctime>` por lo que lo cambie `tanterior`. Después tuve otro problema, puesto que el primer color de fondo en cada ejecución era de color negro, así que lo solucione haciendo que el valor de las variables de rojo, verde y azul tenga un valor random como inicio y no sea negro.

5. Conclusión

Los ejercicios de la práctica, en cuestión de complejidad estuvo bastante normal, para las letras me apoye del programa de GeoGebra para poder tener las coordenadas claras de cada triangulo. Para el transcurso del tiempo de dos segundos reutilicé el programa de ejercicio y lo que me resulto un poco más complicado fue el cambio de color RGB, puesto que tenia una idea con random, pero no sabia como acotar el intervalo para que los valores fueran de [0.0,1.0], así que tuve que investigar un poco más. En conclusión, creo que los ejercicios fueron buenos para darnos una idea de cómo empezar en OpenGL.

6. Bibliografía

- *Función RGB - Soporte técnico de Microsoft*. (s. f.). https://support.microsoft.com/es-es/topic/funci%C3%B3n-rgb-aa04db19-fb8a-4f58-9ad6-71a1f5a43e94
- GeeksforGeeks. (2025b, enero 11). *rand() and srand() in C++*. GeeksforGeeks. https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/?ref=gcse_outind
- *RAND_MAX - Cppreference.com*. (s. f.). https://en.cppreference.com/w/cpp/numeric/random/RAND_MAX