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: García Soto Jean Carlo

Nº de Cuenta: 319226304

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 15/02/2025

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.

Ejercicios:

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

  En este primer ejercicio se hizo una modificación de acuerdo con un ejercicio ya hecho anteriormente que era cambiar de forma cíclica el color de fondo de rojo->verde->azul cada 2 segundos. Pero ahora se iba a cambiar el color de forma random, pero sigue siendo cada 2 segundos.

```
GeiHC                                                    (Ámbito global)
1   //ARCHIVO SEMESTRE 2024-1
2   #include <stdio.h>
3   #include <stdlib.h>     //Libreria para usar rand
4   #include <time.h>       //Libreria para usar time
5   #include <string.h>
6   #include <glew.h>
7   #include <glfw3.h>
```

  Se agregó primero las librerías <stdlib.h> y <time.h>, con el fin de poder usar las funciones **rand** para generar nuestros números random y **time** para poder controlar los tiempos que necesitemos para el requerimiento pedido.

```
9   //Dimensiones de la ventana
10  const int WIDTH = 800, HEIGHT = 800;
11  GLuint VAO, VBO, shader;
12  float rojo = 0.0f, verde = 0.0f, azul = 0.0f;
13  double lastTime = 0.0;      //Inicializar en 0.0
14  int colorIndice = 3;        // Inicia en 3 u cualquier otro que no sea 0, 1 o 2
15
```

  Declaramos e inicializamos las variables **lastTime** y **colorIndice**, la primera nos ayudara para cuando necesitemos guardar el tiempo del sistema, y la segunda en su momento, guardara cada uno de los índices el cual equivale a guardar el color del fondo escogido aleatoriamente. **colorIndice**, no se inicializa con 0, 1 o 2, dado que si empieza con alguno de estos valores, el primer color de fondo, no será el seleccionado y en vez de tres opciones tendríamos 2.

```
33
34  void cambiarColor() {       //Función para el cambio de color del fondo
35      int nuevoColor;
36      do {
37          nuevoColor = rand() % 3;            //Genera un número entre 0 y 2
38      } while (nuevoColor == colorIndice);    //Asegurando que no sea el mismo color
39
40      colorIndice = nuevoColor;               //Asigna el nuevo indice de color
41
42      //Comparar que indice es, para modifcar las variables de rojo, verde y azul
43      if (colorIndice == 0) {
44          rojo = 1.0f; verde = 0.0f; azul = 0.0f;     // Rojo
45      }
46      else if (colorIndice == 1) {
47          rojo = 0.0f; verde = 1.0f; azul = 0.0f;     // Verde
48      }
49      else if (colorIndice == 2) {
50          rojo = 0.0f; verde = 0.0f; azul = 1.0f;     // Azul
51      }
52
53  }
```

Se creó una nueva función para cuando esta se llegue a llamar, nos modifique el color de fondo. Es de tipo void, ya que nuestras variables rojo, azul y verde son globales, por lo que no necesitamos regresar algún valor.
También se implementó una condición que hace que a la hora de generar un número random, este no sea el mismo que el que se está mostrando, por lo que el do while acaba cuando se cumple esa condición. Y procedemos a guardar la variable y asignar los valores a nuestras variables rojo, verde y azul.

```
300
301     srand(time(NULL));          //Inicializar semilla aleatoria
302     lastTime = glfwGetTime();   //Obtener el tiempo inicial
303     cambiarColor();
```

La primera función **srand(time(NULL)** nos ayudará para poder generar números aleatorios nuevos cada que se pida, ya que sin este si bien **rand()** nos lo puede dar, pero cada vez que se ejecute el programa serán los mismos números aleatorios. Y ahora **lastTime** lo guardamos con el valor del tiempo inicial/actual en ese momento.
Y mandamos a llamar nuestra función de **cambiarColor()**, para que nuestro primer color en el fondo, sea aleatorio.

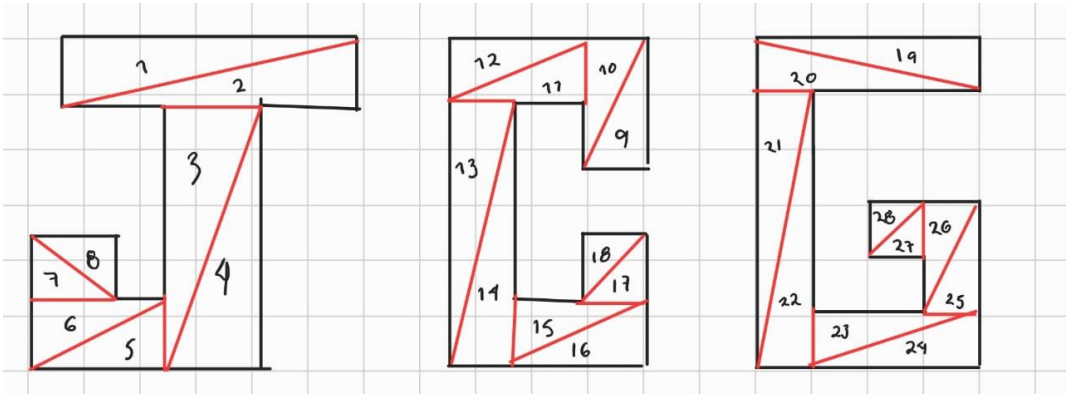
```
304
305     //Loop mientras no se cierra la ventana
306     while (!glfwWindowShouldClose(mainWindow))
307     {
308         //Recibir eventos del usuario
309         glfwPollEvents();
310
311         //Obtener el tiempo actual
312         double tiempoActual = glfwGetTime();
313
314         if (tiempoActual - lastTime >= 2.0) {
315             lastTime = tiempoActual;        //Reiniciar el tiempo
316             cambiarColor();                 //Llama a la funcion para cambiar el color
317
318         }
```

Se vuelve a declarar una variable **tiempoActual** la cual almacenará el tiempo en el que se ande ejecutando, y dado que estará en un bucle este valor estará sobrescribiéndose. Después nuestra función if nos ayudara para saber si ya pasaron los 2 segundos en dado que si, **lastTime** reiniciaría a un valor que sea igual a tiempoActual, y procedemos a llamar la función para cambiar el color del fondo.

```
319
320         //Limpiar la ventana
321         glClearColor(rojo,verde,azul,1.0f); //Asignamos las variables de rojo, verde y azul ya modificadas
322         glClear(GL_COLOR_BUFFER_BIT);
```

Si no se cumple el anterior if se mantienen el valor de las variables rojo, verde y azul. Y si se cumple el if, se le asignan y es cuando aquí cambia el color del fondo.

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



Este es un bosquejo de como están conformadas las 3 figuras, con el número de triángulos, esto con el fin de poder guiarme de una mejor manera.

```
GLfloat vertices[] = {
    //Primeras 3 iniciales de mi nombre JCG
    // Letra J
    -0.8f, 0.5f, 0.0f,      //triangulo 1
    -0.8f, 0.37f, 0.0f,
    -0.4f, 0.5f, 0.0f/**/,

    -0.8f, 0.37f, 0.0f,     //triangulo 2
    -0.4f, 0.5f, 0.0f,
    -0.4f, 0.37f, 0.0f,

    -0.67f, 0.37f, 0.0f,    //triangulo 3
    -0.53f, 0.37f, 0.0f,
    -0.67f, 0.0f, 0.0f,

    -0.53f, 0.37f, 0.0f,    //triangulo 4
    -0.53f, 0.0f, 0.0f,
    -0.67f, -0.0f, 0.0f,

    - 0.67f, 0.1f, 0.0f,    //triangulo 5
    -0.67f, 0.0f, 0.0f,
    -0.85f, 0.0f, 0.0f,

    -0.67f, 0.1f, 0.0f,     //triangulo 6
    -0.85f, 0.1f, 0.0f,
    -0.85f, 0.0f, 0.0f,

    -0.75f, 0.1f, 0.0f,     //triangulo 7
    -0.85f, 0.1f, 0.0f,
    -0.85f, 0.15f, 0.0f,

    -0.75f, 0.1f, 0.0f,     //triangulo 8
    -0.85f, 0.15f, 0.0f,
    -0.75f, 0.15f, 0.0f,
```

LabCGeiHC (Ámbito global)

```
    //letra C
    0.1f, 0.33f, 0.0f,      //triangulo 9
    0.0f, 0.33f, 0.0f,
    0.1f, 0.5f, 0.0f,

    0.0f, 0.33f, 0.0f,      //triangulo 10
    0.1f, 0.5f, 0.0f,
    0.0f, 0.5f, 0.0f,

    0.0f, 0.5f, 0.0f,       //triangulo 11
    0.0f, 0.4f, 0.0f,
    -0.2f, 0.4f, 0.0f,

    0.0f, 0.5f, 0.0f,       //triangulo 12
    -0.2f, 0.4f, 0.0f,
    -0.2f, 0.5f, 0.0f,

    -0.2f, 0.4f, 0.0f,      //triangulo 13
    -0.09f, 0.4f, 0.0f,
    -0.2f, 0.0f, 0.0f,

    -0.09f, 0.4f, 0.0f,     //triangulo 14
    -0.2f, 0.0f, 0.0f,
    -0.09f, 0.0f, 0.0f,

    -0.09f, 0.0f, 0.0f,     //triangulo 15
    -0.09f, 0.1f, 0.0f,
    0.1f, 0.1f, 0.0f,

    -0.09f, 0.0f, 0.0f,     //triangulo 16
    0.1f, 0.1f, 0.0f,
    0.1f, 0.0f, 0.0f,

    0.1f, 0.1f, 0.0f,       //triangulo 17
    0.0f, 0.1f, 0.0f,
    0.1f, 0.17f, 0.0f,

    0.0f, 0.1f, 0.0f,       //triangulo 18
    0.1f, 0.17f, 0.0f,
    0.0f, 0.17f, 0.0f,
```

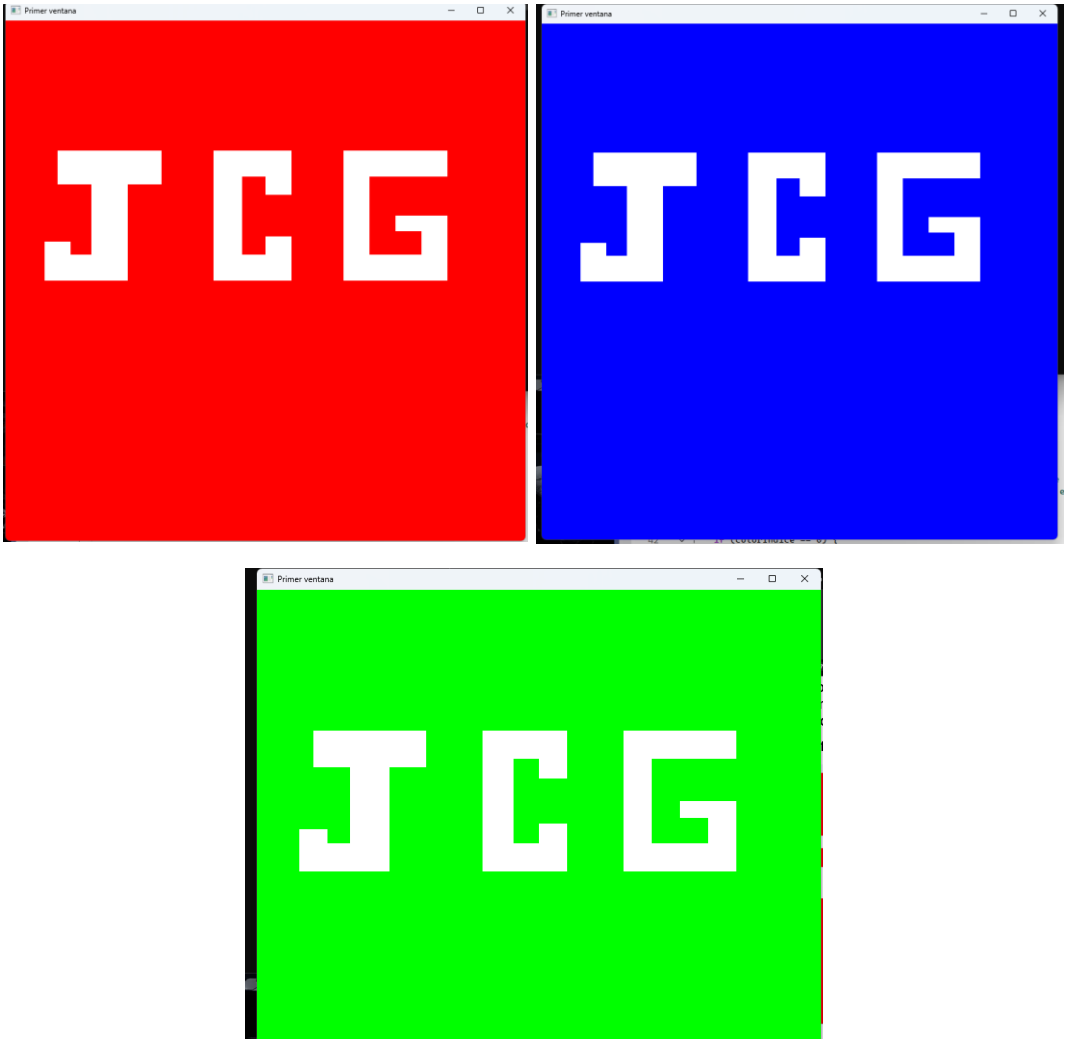
```
133         //letra G
134         0.3f, 0.5f, 0.0f,       //triangulo 19
135         0.7f, 0.5f, 0.0f,
136         0.7f, 0.4f, 0.0f,
137
138         0.7f, 0.4f, 0.0f,       //triangulo 20
139         0.3f, 0.5f, 0.0f,
140         0.3f, 0.4f, 0.0f,
141
142         0.3f, 0.4f, 0.0f,       //triangulo 21
143         0.4f, 0.4f, 0.0f,
144         0.3f, 0.0f, 0.0f,
145
146         0.4f, 0.4f, 0.0f,       //triangulo 22
147         0.3f, 0.0f, 0.0f,
148         0.4f, 0.0f, 0.0f,
149
150         0.4f, 0.0f, 0.0f,       //triangulo 23
151         0.4f, 0.1f, 0.0f,
152         0.7f, 0.1f, 0.0f,
153
154         0.4f, 0.0f, 0.0f,       //triangulo 24
155         0.7f, 0.0f, 0.0f,
156         0.7f, 0.1f, 0.0f,
157
158         0.7f, 0.1f, 0.0f,       //triangulo 25
159         0.6f, 0.1f, 0.0f,
160         0.7f, 0.25f, 0.0f,
161
162         0.6f, 0.1f, 0.0f,       //triangulo 26
163         0.6f, 0.25f, 0.0f,
164         0.7f, 0.25f, 0.0f,
165
166         0.6f, 0.25f, 0.0f,      //triangulo 27
167         0.6f, 0.19f, 0.0f,
168         0.5f, 0.19f, 0.0f,
169
170         0.6f, 0.25f, 0.0f,      //triangulo 28
171         0.5f, 0.25f, 0.0f,
172         0.5f, 0.19f, 0.0f
173
174
175     };
```

Para la J se usaron 8 triángulos, para la C se usaron 10 triángulos, y por último para la G también se usaron otros 10 triángulos.

```
326     glBindVertexArray(VAO);
327     glDrawArrays(GL_TRIANGLES,0,84); // Cambiar de acuerdo a la cantida de vertices
328     glBindVertexArray(0);
```

Dado que teníamos un total de 28 triángulos, esto lo multiplicamos por 3 (cantidad de vértices de un triángulo), y nos da como resultado 84, el cual es el mínimo que debemos tener para que se muestren todos nuestros triángulos sin error.

Los dos ejercicios se muestran de forma simultánea y están en el mismo main



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**

   Afortunadamente no hubo un error de compilación o al momento de ejecutar, lo único que llegó a ser un problema era un poco sobre la ubicación de las coordenadas y el tamaño de las letras, pero con ayuda del esquema gráfico que hice en mi Tablet, fue de mucha ayuda para poder hacerlo más sencillo.

3. **Conclusión:**
    a. **Los ejercicios del reporte: Complejidad, Explicación.**
    Considero que la complejidad de los ejercicios estuvo de acuerdo con lo visto en el laboratorio, puede que en un inicio lo haya visto algo extenso, pero eso solo se debió a que era mi primera vez haciendo esto, ya que una vez le agarras la forma de cómo hacerlo, es rápido hacerlo todo, el cual ayudó demasiado a que fueron varios triángulos los que se tenían que hacer para visualizarlo de mejor manera. Y para el cambio de color no fue tan difícil, dado que solo era pensar una forma de implementarlo y buscar si debíamos de hacer alguna adaptación de sintaxis al lenguaje C++, pero de ahí en fuera, fue sencillo esta parte

    b. **Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica**
    Me gusto el nivel de la práctica aparenta ser demasiada, pero con la primera figura se agarra confianza y sale más rápido lo demás. Por lo que se me hizo interesante dado que pude cumplir con los requerimientos de la práctica con éxito.

    c. **Conclusión**
    Se pudo cumplir con los ejercicios de la práctica, asiendo uso de programación con números random y la implementación de una pequeña función para el cambio de color. Luego se usó el previo conocimiento adquirido de construir triángulos con coordenadas, y con eso mostrar al mismo tiempo de ejecución las 3 primeras iniciales de mi nombre. Por lo que puedo concluir que los objetivos de esta práctica se cumplieron exitosamente.

4. **Bibliografía en formato APA**

- Como usar la funcion rand en C++ - En-c. (s. f.). https://www.codigazo.com/en-c/como-usar-funcion-rand-en-c
- Abellán, J. (s. f.). Obtención de Números Aleatorios en C. https://old.chuidiang.org/clinux/funciones/rand.php