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: Sanchez Calvillo Saida Mayela

Nº de Cuenta: 318164481

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 15 de febrero 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.

El primer ejercicio que se dejo fue abrir una ventana que cambie el color de fondo de forma random tomando rango de colores RGB y con una periodicidad de 2 segundos.

Para esto se tomó el código realizado en clase llamado “segundo main”, el cual ya tenia declarado el color de fondo. Como aquel código solo vareaba entre los 3

colores primarios, vamos a quitar esa parte de código e introduciremos una que tenga una variación por todas las combinaciones de colores RGB de forma aleatoria en un rango de 0.0 a 1.0.

```
int main()
{
    //Semilla para generar los números aleatorios
    srand(static_cast<unsigned int>(time(0)));
```

Para generar números aleatorios se inicializo una semilla. Esto para que los colores sean diferentes cada vez que se ejecuta el programa y esto basado en el tiempo actual y lograr el intervalo de cambios que se requieren.

Después retire las variables que ya no usaremos en este código y ahora declare que cada componente RGB fuera una variable inicializada en 0.0f en lugar de condicionarlos. La variable intervalo se inicializo en 2 por los segundos que necesitamos para que cambie de color.

```
float r = 0.0f, g = 0.0f, b = 0.0f;
float tiempo_anterior = glfwGetTime();
float intervalo = 2.0f;

//Loop mientras no se cierra la ventana
while (!glfwWindowShouldClose(mainWindow))
{
    float tiempo_actual = glfwGetTime();
```

Anteriormente, dentro del condicional if solo definía que cada que transcurriera un segundo cambiaba de estado. Esta ocasión se elimino eso y se definió que cada componente RGB tuviera un valor distinto para cada ocasión.

```
// Si han pasado 2 segundos, cambia el color de fomra aleatoria
if (tiempo_actual - tiempo_anterior >= intervalo) {
    // Genera colores aleatorios en el rango de 0.0 a 1.0
    r = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
    g = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
    b = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
    tiempo_anterior = tiempo_actual;
}
```

Donde *rand()* nos genera un número entero aleatorio en el rango de 0 a *rand_max (1.0).*

static_cast<float> convierte el número entero generado por *rand()* en un número tipo flotante

Para el segundo ejercicio que hiciéramos nuestras 3 letras iniciales de nuestros nombres creadas a partir de triángulos donde todas las letras son del mismo color.

Primero dibuje las letras y se dividieron en triángulos para poder calcular cuantos vértices iban a ser en total y en donde iban a estar ubicados todos los triángulos. En total quedaron 84 vértices (28 triángulos) y solo se fueron agregando los vértices en el vector de vértices.

```
GLfloat vertices[] = {
    -0.833f,0.332f,0.0f,    // S        -0.332f, -0.332f, 0.0f,      //M        0.415f, 0.332f, 0.0f,      // C
    -0.415f,0.5f,0.0f,                 -0.166f, 0.5f, 0.0f,                    0.833f, 0.5f, 0.0f,
    -0.666f,0.5f,0.0f,                 -0.332f, 0.5f, 0.0f,                    0.583f, 0.5f, 0.0f,

    -0.833f, 0.332f,0.0f,              -0.332f, -0.332f, 0.0f,                 0.415f, 0.332f, 0.0f,
    -0.415f,0.332f, 0.0f,              -0.166f, -0.332f, 0.0f,                 0.833f, 0.332f, 0.0f,
    -0.415f,0.5f,0.0f,                 -0.166f, 0.5f, 0.0f,                    0.833f, 0.5f, 0.0f,

    -0.833f, 0.166f, 0.0f,             -0.166f, 0.249f, 0.0f,                  0.415f, -0.166f, 0.0f,
    -0.666f, 0.332f, 0.0f,             -0.0f, 0.249f, 0.0f,                    0.583f, 0.332f, 0.0f,
    -0.833f, 0.332f, 0.0f,             -0.166f, 0.5f, 0.0f,                    0.415f, 0.332f, 0.0f,

    -0.833f, 0.166f, 0.0f,             -0.166f, 0.249f, 0.0f,                  0.415f, -0.166f, 0.0f,
    -0.666f, 0.166f, 0.0f,             0.0f, 0.0f, 0.0f,                       0.583f, -0.166f, 0.0f,
    -0.666f, 0.332f, 0.0f,             0.0f, 0.249f, 0.0f,                     0.583f, 0.332f, 0.0f,

    -0.583f, 0.0f, 0.0f,               0.0f, 0.0f, 0.0f,                       0.583f, -0.332f, 0.0f,
    -0.666f, 0.166f, 0.0f,             0.166f, 0.249f, 0.0f,                   0.833f, -0.166f, 0.0f,
    -0.833f, 0.166f, 0.0f,             0.0f, 0.249f, 0.0f,                     0.415f, -0.166f, 0.0f,

    -0.583f, 0.0f, 0.0f,               0.0f, 0.249f, 0.0f,                     0.583f, -0.332f, 0.0f,
    -0.415f, 0.0f, 0.0f,               0.166f, 0.249f, 0.0f,                   0.833f, -0.332f, 0.0f,
    -0.666f, 0.166f, 0.0f,             0.166f, 0.5f, 0.0f,                     0.833f, -0.166f, 0.0f

    -0.583f, -0.166f, 0.0f,            0.166f, -0.332f, 0.0f,
    -0.415f, 0.0f, 0.0f,               0.332f, 0.5f, 0.0f,
    -0.583f, 0.0f, 0.0f,               0.166f, 0.5f, 0.0f,

    -0.583f, -0.166f, 0.0f,            0.166f, -0.332f, 0.0f,
    -0.415f, -0.166f, 0.0f,            0.332f, -0.332f, 0.0f,
    -0.415f, 0.0f, 0.0f,               0.332f, 0.5f, 0.0f,

    -0.833f, -0.332f, 0.0f,
    -0.415f, -0.166f, 0.0f,
    -0.833f, -0.166f, 0.0f,

    -0.833f, -0.332f, 0.0f,
    -0.583f, -0.332f, 0.0f,
    -0.415f, -0.166f, 0.0f,
```

```
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 84);
glBindVertexArray(0);
```

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

Hubo varios problemas donde la lógica del código no me cuadraba y quizás había una forma de hacerlo más eficiente, pero se logro usando la semilla y sin tantas líneas de código.

Un problema que me causo mayor conflicto fue distribuir las letras y los triángulos dentro de estas sobre un plano definido de 2x2 unidades. Esto lo resolví dibujando el plano cartesiano en una libreta y calculé cuanto media cada cuadrito para poder dividir bien el espacio. Por ejemplo, de 0 a 1 en “x” cada cuadrito medía 0.083, y así fui calculando las distancias entre ejes y vértices, así obtuve los pares ordenados.

3.- Conclusión:

Implementar un cambio de colores aleatorio en OpenGL utilizando *rand()* fue una solución simple para crear el efecto visual que necesitábamos. El uso del temporizador combinado con la función *glfwGetTime()* me permitió controlar el cambio de color a mi gusto y en forma de segundos en tiempo actual. Sin duda investigar sobre algunas funciones que podrían ser de ayuda resulto muy favorecedor en esta práctica.

Para formar las letras se necesitan combinar múltiples triángulos en posiciones específicas que representen los segmentos de cada letra y eso fue tardado,

Me gustaría saber algunas otras funciones o técnicas que el profesor habría utilizado para resolver estos ejercicios también.