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: Barragán Pilar Diana

Nº de Cuenta: 318147981

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 17 de agosto de 2024

CALIFICACIÓN: _____

Practica 1: Introducción a OpenGL

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

En la realización de este ejercicio utilice de base el código que ya había escrito al ejecutar el ejercicio de clase, por lo tanto, nuevamente hice uso de las librerías; `time.h`, `stdlib.h`, `chrono` y `thread`, la primera de ellas para generar un número aleatorio, la segunda para el comando `rand()` que genera una sucesiones de números aleatorios distintos cada vez que se ejecuta el programa, las librerías y las emplee para el comando que me permitió detener la ejecución por los dos segundos que se estipularon para este primer ejercicio.

También agregue las variables de tipo `float` que funcionarán para asignarlas a los números aleatorios que se nos generen.

```
#include <stdio.h>
#include <string.h>
#include <glew.h>
#include <glfw3.h>
#include <time.h>
#include <stdlib.h>
#include <chrono>
#include <thread>

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

//Declaración de variables para el rgb con valores aleatorios
float R, G, B;
```

Figura 1. Agregación de librerías y variables tipo `float`.

Subsecuentemente agregue la línea de código con la que se generarían los números aleatorios en cada ejecución del programa, que como anteriormente mencione es con el uso de la función `rand()`, posterior a esto dentro del bucle `while` que se mantiene en iteración mientras no se cierre la ventana asigne a las variables `R`, `G`, `B` declaradas previamente el valor resultante al aplicar el modulo de dos, puesto que se busca que los números asociados sean 1 o 0.

Finalmente se colocan las variables en la función `glClearColor()` que nos permite asignar el código RGB que más tarde con la función `glClear(GL_COLOR_BUFFER_BIT);` se le asignará al fondo de la pantalla. Para que el cambio de color sea apreciable durante dos segundos como se especifico en el ejercicio se hace uso del Sleep Thread que detiene la ejecución del bucle.

```
//Semilla para generar los números aleatorios
srand(time(NULL));

//Loop mientras no se cierra la ventana
while (!glfwWindowShouldClose(mainWindow))
{
    //Recibir eventos del usuario
    glfwPollEvents();

    //Modulo y asignación del número aleatorio para que sea 0 o 1
    R = rand() % 2;
    G = rand() % 2;
    B = rand() % 2;

    //Limpiar la ventana
    glClearColor(R, G, B, 1.0f);

    //Detiene la ejecución del programa durante dos segundos
    std::this_thread::sleep_for(std::chrono::seconds(2));

    glClear(GL_COLOR_BUFFER_BIT);
```

Figura 2. Números aleatorios. Asignación de variables. Suspensión de la ejecución del programa.

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

Para comenzar a plantear la posición de las letras y la forma primero realice un dibujo preliminar que me permitió conocer como sería la forma de las letras al igual que el número de triángulos que tendría cada una con la respectiva posición de los vértices.

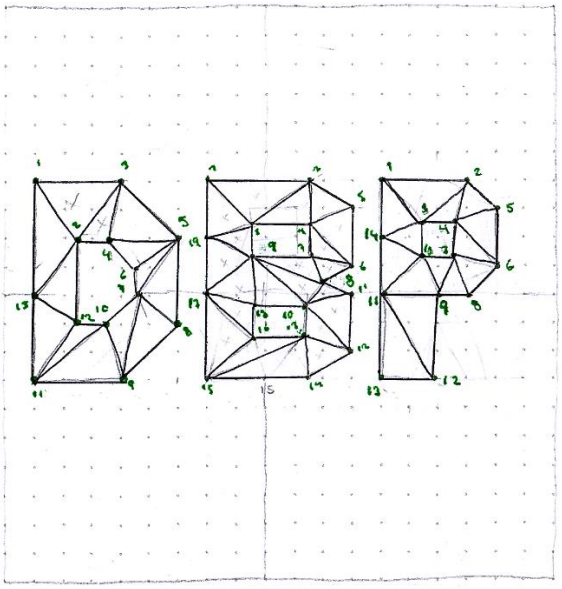

Figura 3. Dibujo preliminar de letras iniciales con triángulos.

Lo siguiente fue pasar lo efectuado en el dibujo preliminar al programa, para ello utilice la matriz de vértices, haciendo una pequeña separación simbólica cada que iniciaba una nueva letra, fui dibujando y numerando los vértices que ya había asignado en el código a partir de lo establecido en el dibujo.

```
void CrearTriangulo()
{
    GLfloat vertices[] = {
        //Letra D
        -0.9f, 0.4f,0.0f, //1
        -0.7f,0.2f, 0.0f,//2
        -0.5f,0.4f,0.0f,//3
        -0.7f,0.2f, 0.0f,//2
        -0.5f,0.4f,0.0f,//3
        -0.55f,0.2f,0.0f,//4
        -0.5f,0.4f,0.0f,//3
        -0.55f,0.2f,0.0f,//4
        -0.3f, 0.2f,0.0f,//5
        -0.55f,0.2f,0.0f,//4
        -0.3f, 0.2f,0.0f,//5
        -0.45f, 0.1f,0.0f,//6
        -0.3f, 0.2f,0.0f,//5
        -0.45f, 0.1f,0.0f,//6
        -0.45f, 0.0f,0.0f,//7
        -0.3f, 0.2f,0.0f,//5
        -0.45f, 0.0f,0.0f,//7
        -0.3f, -0.1f,0.0f,//8
        -0.45f, 0.0f,0.0f,//7
        -0.3f, -0.1f,0.0f,//8
        -0.5f, -0.3f,0.0f,//9
        -0.45f, 0.0f,0.0f,//7
        -0.5f, -0.3f,0.0f,//9
        -0.55f, -0.1f,0.0f,//10
        -0.55f, -0.1f,0.0f,//10
        -0.5f, -0.3f,0.0f,//9
        -0.9f, -0.3f,0.0f,//11
        -0.55f, -0.1f,0.0f,//10
        -0.9f, -0.3f,0.0f,//11
        -0.7f, -0.1f,0.0f,//12
        -0.9f, -0.3f,0.0f,//11
        -0.7f, -0.1f,0.0f,//12
        -0.9f, 0.0f,0.0f,//13
        -0.7f,0.2f, 0.0f,//2
        -0.7f, -0.1f,0.0f,//12
        -0.9f, 0.0f,0.0f,//13
        -0.9f, 0.0f,0.0f,//13
        -0.9f, 0.4f,0.0f, //1
        -0.7f,0.2f, 0.0f,//2   Vertices totales: 39
```

Figura 4. Vértices utilizados para formar la letra D.

```
//Letra B
-0.2f, 0.4f,0.0f, //1
0.15f,0.4f, 0.0f,//2
-0.05f,0.25f,0.0f,//3
0.15f,0.4f, 0.0f,//2
-0.05f,0.25f,0.0f,//3
0.15f,0.25f, 0.0f,//4
0.15f,0.4f, 0.0f,//2
0.15f,0.25f, 0.0f,//4
0.3f,0.3f, 0.0f,//5
0.15f,0.25f, 0.0f,//4
0.3f,0.3f, 0.0f,//5
0.3f,0.1f, 0.0f,//6
0.15f,0.25f, 0.0f,//4
0.3f,0.1f, 0.0f,//6
0.15f,0.15f, 0.0f,//7
0.3f,0.1f, 0.0f,//6
0.15f,0.15f, 0.0f,//7
0.2f,0.05f, 0.0f,//8
0.15f,0.15f, 0.0f,//7
0.2f,0.05f, 0.0f,//8
-0.05f,0.15f, 0.0f,//9
0.15f,-0.05f, 0.0f,//10
0.2f,0.05f, 0.0f,//8
-0.05f,0.15f, 0.0f,//9
0.15f,-0.05f, 0.0f,//10
0.2f,0.05f, 0.0f,//8
0.3f,0.0f, 0.0f,//11
0.15f,-0.05f, 0.0f,//10
0.3f,0.0f, 0.0f,//11
0.3f,-0.2f, 0.0f,//12
0.15f,-0.05f, 0.0f,//10
0.3f,-0.2f, 0.0f,//12
0.15f,-0.15f, 0.0f,//13
0.3f,-0.2f, 0.0f,//12
0.15f,-0.15f, 0.0f,//13
0.15f,-0.3f, 0.0f,//14
0.15f,-0.15f, 0.0f,//13
0.15f,-0.3f, 0.0f,//14
-0.2f,-0.3f, 0.0f,//15
0.15f,-0.15f, 0.0f,//13
-0.05f,-0.15f, 0.0f,//16
-0.2f,-0.3f, 0.0f,//15
-0.2f, 0.0f, 0.0f,//17
-0.05f, -0.15f, 0.0f,//16
-0.2f,-0.3f, 0.0f,//15
-0.2f, 0.0f, 0.0f,//17
-0.05f, -0.15f, 0.0f,//16
-0.05f, -0.05f, 0.0f,//18
-0.2f, 0.0f, 0.0f,//17
-0.05f, 0.15f, 0.0f,//9
-0.05f, -0.05f, 0.0f,//18
-0.05f, -0.05f, 0.0f,//18
0.15f, -0.05f, 0.0f,//10
-0.05f, 0.15f, 0.0f,//9
-0.2f, 0.0f, 0.0f,//17
-0.05f, 0.15f, 0.0f,//9
-0.2f, 0.2f, 0.0f,//19
-0.05f, 0.15f, 0.0f,//9
-0.2f, 0.2f, 0.0f,//19
-0.05f, 0.25f, 0.0f,//3
-0.2f, 0.2f, 0.0f,//19
-0.05f, 0.25f, 0.0f,//3
-0.2f, 0.4f, 0.0f, //1   Vertices totales: 63
```

Figura 5. Vértices utilizados para formar la letra B.

```
//Letra P
0.4f, 0.4f, 0.0f,//1
0.7f, 0.4f, 0.0f,//2
0.55f, 0.25f, 0.0f,//3
0.65f, 0.25f, 0.0f,//4
0.7f, 0.4f, 0.0f,//2
0.55f, 0.25f, 0.0f,//3
0.65f, 0.25f, 0.0f,//4
0.7f, 0.4f, 0.0f,//2
0.8f, 0.3f, 0.0f,//5
0.65f, 0.25f, 0.0f,//4
0.8f, 0.1f, 0.0f,//6
0.8f, 0.3f, 0.0f,//5
0.65f, 0.25f, 0.0f,//4
0.8f, 0.1f, 0.0f,//6
0.65f, 0.15f, 0.0f,//7
0.7f, 0.0f, 0.0f,//8
0.8f, 0.1f, 0.0f,//6
0.65f, 0.15f, 0.0f,//7
0.7f, 0.0f, 0.0f,//8
0.6f, 0.0f, 0.0f,//9
0.65f, 0.15f, 0.0f,//7
0.55f, 0.15f, 0.0f,//10
0.6f, 0.0f, 0.0f,//9
0.65f, 0.15f, 0.0f,//7
0.55f, 0.15f, 0.0f,//10
0.6f, 0.0f, 0.0f,//9
0.4f, 0.0f, 0.0f,//11
0.6f, -0.3f, 0.0f,//12
0.6f, 0.0f, 0.0f,//9
0.4f, 0.0f, 0.0f,//11
0.6f, -0.3f, 0.0f,//12
0.4f, -0.3f, 0.0f,//13
0.4f, 0.0f, 0.0f,//11
0.4f, 0.2f, 0.0f,//14
0.55f, 0.15f, 0.0f,//10
0.4f, 0.0f, 0.0f,//11
0.4f, 0.2f, 0.0f,//14
0.55f, 0.15f, 0.0f,//10
0.55f, 0.25f, 0.0f,//3
0.4f, 0.2f, 0.0f,//14
0.4f, 0.4f, 0.0f,//1
0.55f, 0.25f, 0.0f,//3 Vertices totales: 42
```

Figura 6. Vértices utilizados para formar la letra P.

Por ultimo para que se dibuje correctamente, otra vez vamos al ciclo `while` en donde con la función `glDrawArrays(GL_TRIANGLES,0,144);` establecemos que se dibujen nuestros triángulos en la matriz de vértices estableciendo desde el vértice 0 hasta el total final de vértices.

```
glClear(GL_COLOR_BUFFER_BIT);

glUseProgram(shader);

glBindVertexArray(VAO);

//Dibuja los triangulos desde el primer vertice hasta número total
glDrawArrays(GL_TRIANGLES,0,144);
```

Figura 7. Función para dibujar los vertices.

EJECUCIÓN DEL PROGRAMA:

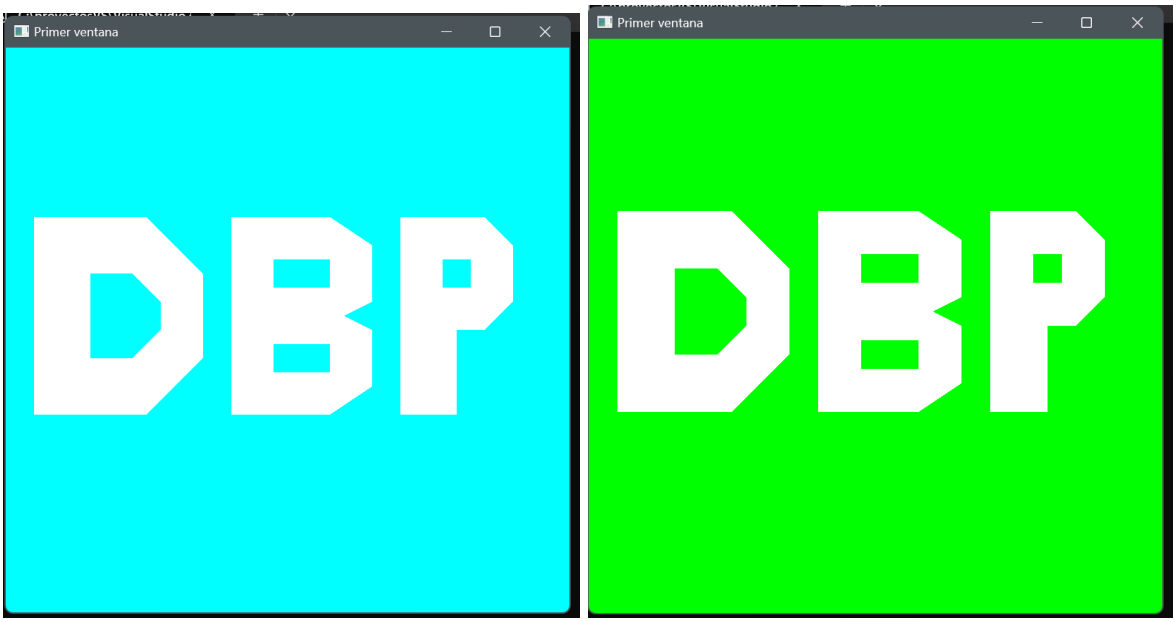

Figura 8. Ejecución del programa letras iniciales con fondo que cambia de color.

Al realizar las actividades propuestas en esta práctica no se me presento ningún problema con respecto al programa, ya que al compilarlo este se ejecuto adecuadamente. El único inconveniente fue a la hora de escribir los vértices en la matriz, pues al ser tanto tuve errores al poner incorrectamente las coordenadas lo que generaba de forma equivocada el dibujo, sin embargo, esto se solucionó fácilmente revisando los vértices y estableciendo los datos correctos.

CONCLUSIÓN:

Los ejercicios planteados en esta práctica me parecieron adecuados para conocer mejor la forma en que se debe trabajar en OpenGL, puesto que al dibujar nuestras iniciales solo con la forma geométrica de triángulos, me fue posible ver la ejecución del programa al igual que el propósito de distintas funciones, en este caso el modo de dibujar triángulos y que a partir de ellos se pueden realizar diversas figuras. Por otro lado, el agregar la funcionalidad de que el color del fondo en la pantalla cambiará me permitió modificar el código después de haber reflexionado sobre su funcionamiento en cuanto a características de una ventana.

FUENTES:

CPLUSPLUS. (2023). *std::this_thread::sleep_for*. Recuperado el 15 de agosto de 2024, de: https://cplusplus.com/reference/thread/this_thread/sleep_for/

OMIJAL. (2016). *Números Aleatorios (Random)*. Recuperado el 15 de agosto de 2024, de: https://www.omijal.org/pagina_c/random.html

OPENGL. (2020, 18 de julio). *Primitive*. Recuperado el 15 de agosto de 2024, de: https://www.khronos.org/opengl/wiki/Primitive