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: Uriarte Ortiz Enrique Yahir

Nº de Cuenta: 318234757

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: Sábado 17 de Agosto del 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

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

    Como se pide que el color sea aleatorio, declare la librería ***"stdlib.h"*** que incluye la función *"rand()"* para poder generar los números aleatorios y la librería *"time.h"* para poder comprobar el tiempo transcurrido.

```
#include <stdlib.h>  // Para rand()
#include <time.h>    // Para time()
```

Fig. 1. Bibliotecas de programa.

Luego modifique la función main, agregue "*srand(time(NULL)):"* para inicializar el generador de números aleatorios, con esto cada ejecución del programa generara diferentes secuencias. Luego declaro la variable *"Color"* para generar el numero aleatorio entre 0 y 2, y dos variables más, una llamada *"anterior"* para almacenar el tiempo transcurrido y la otra "tiempo" para tener el tiempo actual de ejecución, ambas usando ***"glfwGetTime()"*** para obtener el tiempo desde que se inició GLFW, aunque *"tiempo"* es la única que se encuentra dentro del ciclo *"while"* ya que esta se tiene que actualizar constantemente durante la ejecución.

```
//Crear tríangulo
CrearTriangulo();
CompileShaders();

srand(time(NULL));//Generar número aleatorio
int Color = rand() % 3;//Selecciona el color aleatorio
double anterior = glfwGetTime();//Tiempo del último cambio

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

    double tiempo = glfwGetTime();// Obtener el tiempo actual

    if (tiempo - anterior >= 2.0){//Cambiar cada 2 segundos
        Color = rand()%3;//Seleccionar un color aleatorio
        anterior = tiempo;}

    switch (Color){//Establecer el color de fondo según el color actual
    case 0: glClearColor(1.0f, 0.0f, 0.0f, 1.0f); break;//Rojo
    case 1: glClearColor(0.0f, 1.0f, 0.0f, 1.0f); break;//Verde
    case 2: glClearColor(0.0f, 0.0f, 1.0f, 1.0f); break;//Azul
    }

    //Limpiar la ventana
    glClear(GL_COLOR_BUFFER_BIT);
```

Fig. 2. Función main modificada.

Después se declara un ciclo *“if”* que se ejecuta si han pasado 2 segundos, a partir de la diferencia entre el tiempo actual y el tiempo anterior, luego se genera un color aleatorio, 0 para el color rojo, 1 para el color verde o 2 para el color azul y por último actualiza *“anterior”* al tiempo actual para contar un nuevo intervalo de 2 segundos.

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

   En mi caso poseo 2 nombres por lo que el numero de iniciales que realice fueron 4, para realizar las letras utilice diversos triángulos en cada uno, tomando como ejemplo las coordenadas que utilizamos en los triángulos de los ejercicios de laboratorio, realice un dibujo en boceto de cómo se verían, e hice el cálculo aproximado de dónde colocar las coordenadas.

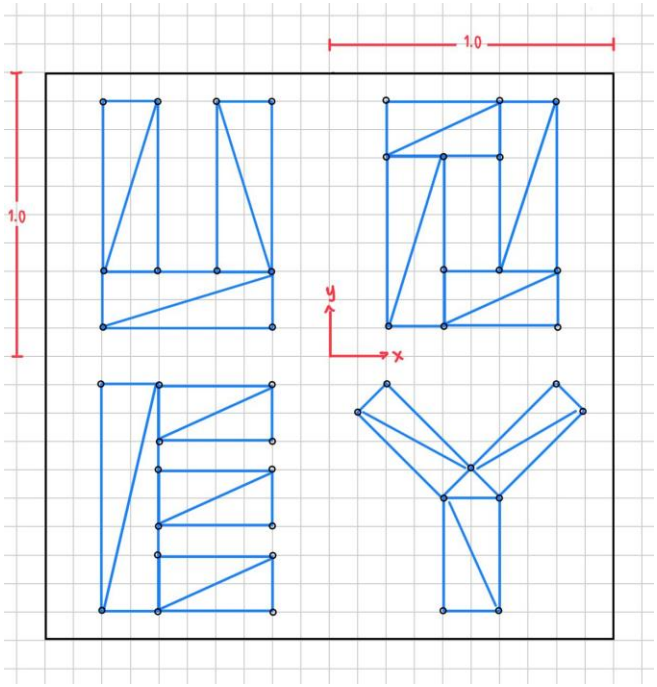


Fig. 3. Boceto para las coordenadas de los triángulos.

```
void CrearTriangulo(){
    GLfloat vertices[]={
        //Coordenadas para Letra U
        -0.2f,0.1f,0.0f,-0.8f,0.1f,0.0f,-0.2f,0.3f,0.0f,//Parte inferior 1
        -0.8f,0.1f,0.0f,-0.8f,0.3f,0.0f,-0.2f,0.3f,0.0f,//Parte inferior 2
        -0.8f,0.3f,0.0f,-0.6f,0.3f,0.0f,-0.6f,0.9f,0.0f,//Columna izquierda 1
        -0.8f,0.3f,0.0f,-0.8f,0.9f,0.0f,-0.6f,0.9f,0.0f,//Columna izquierda 2
        -0.2f,0.3f,0.0f,-0.4f,0.3f,0.0f,-0.4f,0.9f,0.0f,//Columna derecha 1
        -0.2f,0.3f,0.0f,-0.4f,0.9f,0.0f,-0.2f,0.9f,0.0f,//Columna derecha 2

        //Coordenadas para Letra O
        0.2f,0.1f,0.0f,0.4f,0.1f,0.0f,0.4f,0.7f,0.0f,//Rectangulo 1.1
        0.2f,0.1f,0.0f,0.2f,0.7f,0.0f,0.4f,0.7f,0.0f,//Rectangulo 1.2
        0.4f,0.1f,0.0f,0.8f,0.1f,0.0f,0.8f,0.3f,0.0f,//Rectangulo 2.1
        0.4f,0.1f,0.0f,0.4f,0.3f,0.0f,0.8f,0.3f,0.0f,//Rectangulo 2.2
        0.8f,0.3f,0.0f,0.8f,0.9f,0.0f,0.6f,0.3f,0.0f,//Rectangulo 3.1
        0.6f,0.3f,0.0f,0.6f,0.9f,0.0f,0.8f,0.9f,0.0f,//Rectangulo 3.2
        0.2f,0.7f,0.0f,0.6f,0.7f,0.0f,0.6f,0.9f,0.0f,//Rectangulo 4.1
        0.2f,0.7f,0.0f,0.2f,0.9f,0.0f,0.6f,0.9f,0.0f,//Rectangulo 4.2
```

Fig. 4. Coordenadas de triángulos para formar letras.

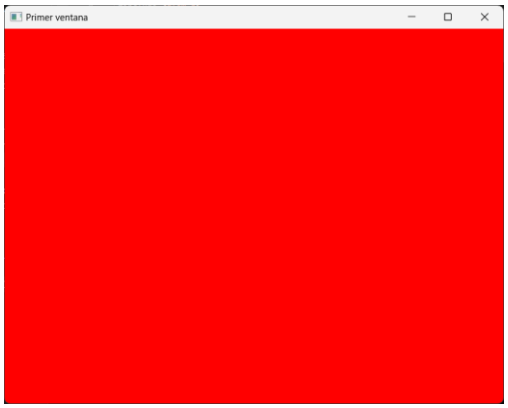
```
        //Coordenadas para Letra E
        -0.8f,-0.1f,0.0f,-0.8f,-0.9f,0.0f,-0.6f,-0.1f,0.0f,//Columna 1
        -0.8f,-0.9f,0.0f,-0.6f,-0.9f,0.0f,-0.6f,-0.1f,0.0f,//Columna 2
        -0.6f,-0.1f,0.0f,-0.2f,-0.1f,0.0f,-0.6f,-0.3f,0.0f,//Rectangulo 1.1
        -0.6f,-0.3f,0.0f,-0.2f,-0.3f,0.0f,-0.2f,-0.1f,0.0f,//Rectangulo 1.2
        -0.6f,-0.4f,0.0f,-0.2f,-0.4f,0.0f,-0.6f,-0.6f,0.0f,//Rectangulo 2.1
        -0.6f,-0.6f,0.0f,-0.2f,-0.6f,0.0f,-0.2f,-0.4f,0.0f,//Rectangulo 2.2
        -0.6f,-0.7f,0.0f,-0.6f,-0.9f,0.0f,-0.2f,-0.7f,0.0f,//Rectangulo 3.1
        0.6f,-0.9f,0.0f,-0.2f,-0.9f,0.0f,-0.2f,-0.7f,0.0f,//Rectangulo 3.2

        //Coordenadas para Letra Y
        0.4f,-0.9f,0.0f,0.6f,-0.9f,0.0f,0.4f,-0.5f,0.0f,//Columna 1
        0.4f,-0.5f,0.0f,0.6f,-0.9f,0.0f,0.6f,-0.5f,0.0f,//Columna 2
        0.4f,-0.5f,0.0f,0.1f,-0.2f,0.0f,0.2f,-0.1f,0.0f,//Rectangulo 1.1
        0.4f,-0.5f,0.0f,0.5f,-0.4f,0.0f,0.2f,-0.1f,0.0f,//Rectangulo 1.2
        0.6f,-0.5f,0.0f,0.5f,-0.4f,0.0f,0.8f,-0.1f,0.0f,//Rectangulo 2.1
        0.6f,-0.5f,0.0f,0.9f,-0.2f,0.0f,0.8f,-0.1f,0.0f,//Rectangulo 2.2
        0.4f,-0.5f,0.0f,0.6f,-0.5f,0.0f,0.5f,-0.4f,0.0f,//Triangulo central
    };
```

Fig. 5. Coordenadas de triángulos para formar letras.

Una vez declaradas las posiciones, modifico los valores de entrada de la función “glDrawArrays” de acuerdo con el número de vértices que empleare, en mi caso utilice 29 triángulos, considerando los 3 vértices q cada uno tiene, en total declaro 87 vértices en la función.

```
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 87);
glBindVertexArray(0);
```

Fig. 6. Numero de vértices para realizar los triángulos.



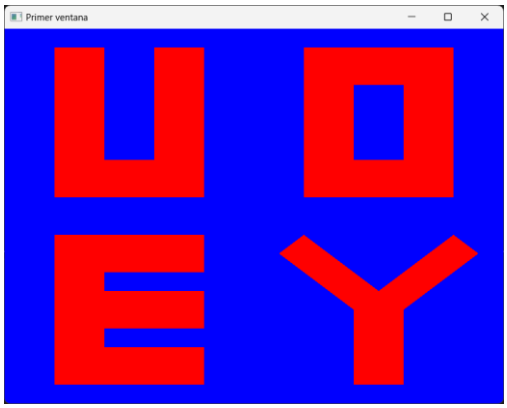


Fig. 6. y Fig. 7. Ejecución del programa.

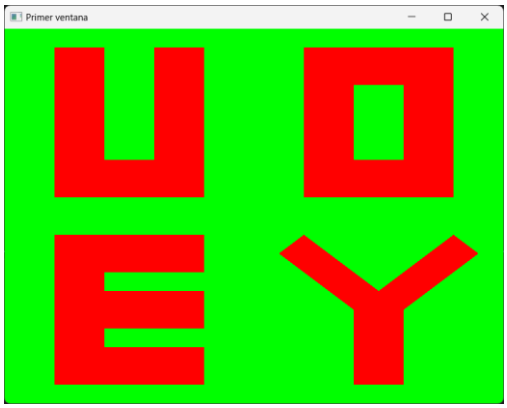


Fig. 8. Ejecución del programa.

2. Problemas presentados.

   En el primer ejercicio tuve el problema de no saber como declarar la forma en que el número aleatorio se encontrara dentro del rango de selección de las 3 opciones en el switch case, pero después de investigar un poco recordé la implementación del módulo en la función rand(), así mismo para verificar la cantidad de tiempo tuve que emplear "***glfwGetTime()"*** para ejecutar el periodo correcto de tiempo, ya que la forma en la que lo realice en los ejercicios de clase no era tan preciso para los 2 segundos. Un inconveniente también que considero tiene el programa es lo aleatorio que puede ser la selección ya que en algunos momentos pareciese que el color dura más de 2 segundos, pero en realidad lo que pasa es que vuelve a ser seleccionado ese color de manera random, después de observar la ejecución de los demás colores puedes ver que el tiempo de 2 segundos si se respeta.

   En el segundo ejercicio no tuve ningún problema de coordenadas, ya que una vez hecho el boceto en dibujo es mas sencillo encontrar donde colocar las coordenadas correctas.

3. Conclusión:
   Tras realizar los ejercicios considero que su complejidad fue un poco mayor o incluso del mismo nivel que los ejercicios de clase para esta práctica, pero con un poco mas de trabajo, como en el caso de las letras que tuve que especificar más coordenadas o en el caso del fondo de color que tuve que agregar "***glfwGetTime()"*** para hacer mejor precisión en el periodo de 2 segundos entre cada color. Considero que falto explicar de otra forma de detalle el rango de colores de la primera actividad ya que cuando la leí nuevamente en mi casa entendí que el fondo podía ser a partir de la computación de los 3 colores, por todo el espectro RGB pero después de ver unas anotaciones que realice en ese día de la practica recordé que era solo de los 3 principales. Al final considero que la practica cumplió su objetivo ya que comprendí de manera general la estructura de las instrucciones en OpenGL, además de que la práctica así como los ejercicios del laboratorio me ayudaron a recordar un poco de programación básica en C++ que no recordaba por la falta de practica.

4. Bibliografía.

- Geelnard, M., & Löwy, C. (s. f.). *glfwGetTime*. PHP-GLFW. Recuperado 16 de agosto de 2024, de https://phpgl.net/API/GLFW/glfwGetTime.html
- Sanches, A. (2020, 9 marzo). *Generar números aleatorios en C++*. Azul School. Recuperado 16 de agosto de 2024, de https://www.azulschool.net/todos-los-grupos/grupo-de-c/forum/topic/generar-numeros-aleatorios-en-c/#:~:text=C%2B%2B%20define%20la%20funci%C3%B3n%20rand,tambi%C3%A9n%20en%20la%20librer%C3%ADa%20cstdlib.&text=valor%20%3D%201%20%2B%20rand()%20%25%20100
- *Time input*. (2018, 4 noviembre). GLFW. Recuperado 16 de agosto de 2024, de https://www.glfw.org/docs/3.0/group__time.html