Universidad Nacional Autónoma de Mexico

Facultad de Ingenieria

Laboratorio de Computación Grafica e Interacción Humano Computadora

Practica 1

Nombre del alumno: Roldan Landa Meir Joshua

Nº de cuenta: 316284158

Grupo de Laboratorio: 11

Semestre 2025 -2

Fecha limite de entrega: 19/02/2025

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

Para comenzar agregamos estas librerías para poder hacer que cambiaran de color de manera random

```
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <thread>
```

Después de esto inicializamos el generador de números aleatorios a través de la siguiente función std::srand(std::time(nullptr)) para poder obtener colores aleatorios diferentes cada vez que se ejecuta el programa como se muestra en la imagen

```
int main()
{

    std::srand(std::time(nullptr));
```

Una vez declarada la función nos vamos al bucle while donde se ejecuta el programa y donde se muestran los colores, pero para esto generamos números de entre 0.0 y 1.0 para los colores rojo, verde y azul en el color del fondo en la terminal

```
//Limpiar la ventana
//glClearColor(0.0f,0.0f,0.0f,1.0f);
glClearColor(red, green, blue, 1.0f);
```

```
//Loop mientras no se cierra la ventana
while (!glfwWindowShouldClose(mainWindow))
{
    //codigo
    // Variables flotante de colores RGB aleatorios
    float red = static_cast<float>(std::rand()) / RAND_MAX;
    float green = static_cast<float>(std::rand()) / RAND_MAX;
    float blue = static_cast<float>(std::rand()) / RAND_MAX;


    //Recibir eventos del usuario
    glfwPollEvents();
```

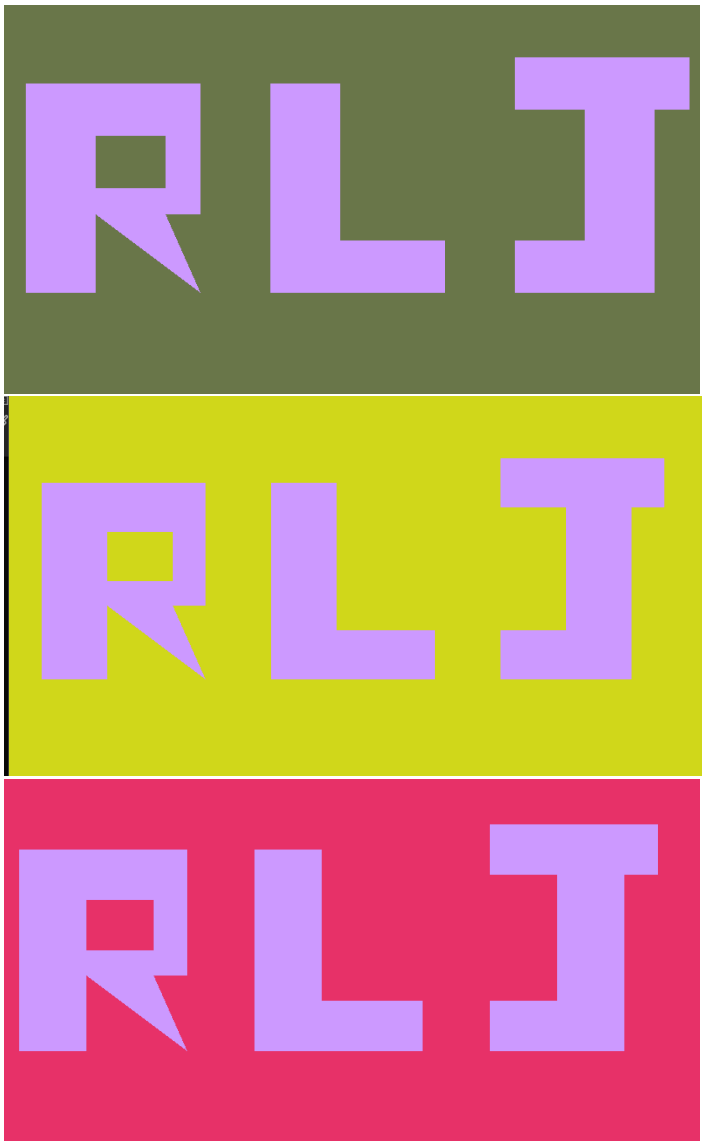
Se utiliza glClearColor(red, green, blue, 1.0f) para establecer el color de fondo con los valores aleatorios generados en donde antes claramente declaramos las variables a utilizar de tipo flotantes y utilizando rand para generar los valores aleatorios de colores

Por ultimo

```
    //Recibir eventos del usuario
    glfwPollEvents();

    //Limpiar la ventana
    //glClearColor(0.0f,0.0f,0.0f,1.0f);
    glClearColor(red, green, blue, 1.0f);

    std::this_thread::sleep_for(std::chrono::seconds(2));

    glClear(GL_COLOR_BUFFER_BIT);

    glUseProgram(shader);
```

En donde se muestra que tenemos un tiempo aproximado de 2 segundos para poder notar el cambio de color esta pausa la hace para generar un nuevo color y con esto llegaríamos a mostrar lo que se nos pide

Estos son nuestros resultados que obtuvimos, por obvias razones no fueron muchas imágenes pero en el código se aprecia perfectamente :



2.- 3 letras iniciales de sus nombres creadas a partir de triángulos, todas las letras son del mismo color.
Los dos ejercicios se muestran de forma simultánea y están en el mismo main

Para las letras solo tuvimos que agregar vértices dentro del plano de -1 a 1 en X y en Y y en Z solo fue cuestión de ir agregando valores para los vértices de todas las letras R J y L en este caso que son las iniciales de mi nombre

```
//R
-0.9f, -0.2f,0.0f,
-0.9f, 0.6f,0.0f,
-0.7f, 0.6f,0.0f,

-0.7f, 0.6f,0.0f,
-0.7f,-0.2f,0.0f,
-0.9f, -0.2f,0.0f,

-0.7f, 0.6f,0.0f,
-0.4f, 0.6f,0.0f,
-0.7f, 0.4f,0.0f,

-0.7f, 0.4f,0.0f,
-0.4f,0.4f,0.0f,
-0.4f,0.6f,0.0f,

-0.5f, 0.4f,0.0f,
-0.4f, 0.4f,0.0f,
-0.5f, 0.2f,0.0f,
```

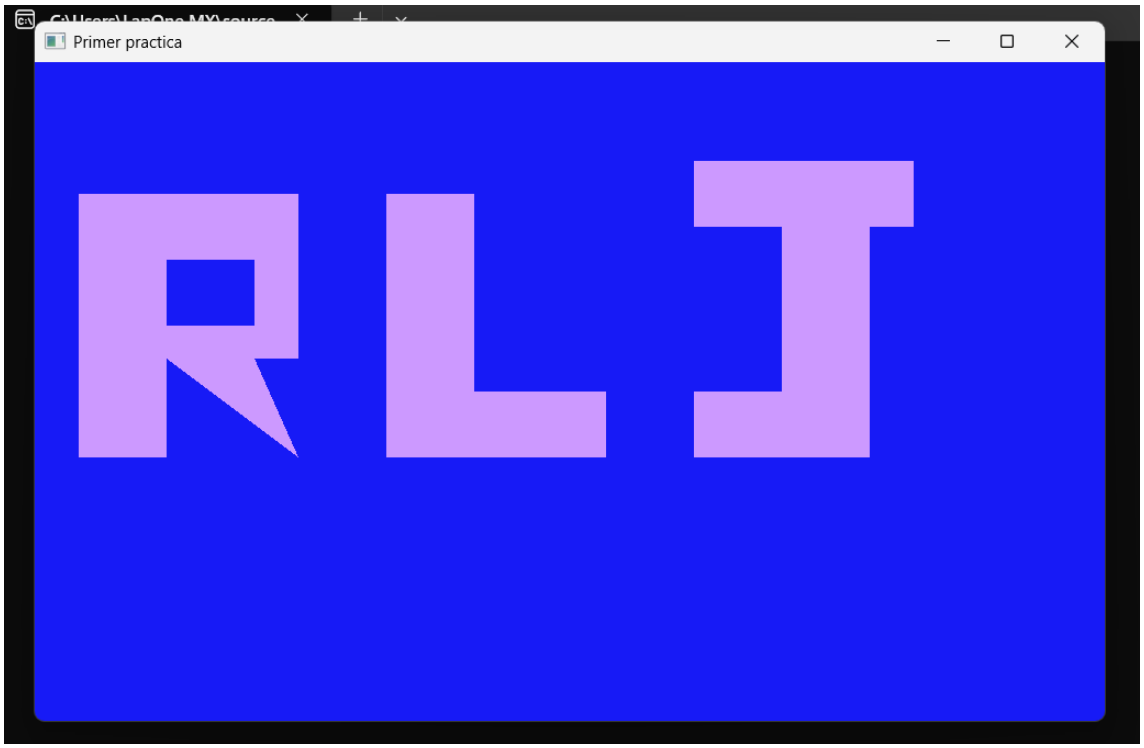
```
// L
-0.2f, -0.2f,0.0f,
-0.2f,0.6f,0.0f,
0.0f, 0.6f,0.0f,

0.0f, 0.6f,0.0f,
-0.2f, -0.2f,0.0f,
0.0f, -0.2f,0.0f,

0.0f, -0.2f,0.0f,
0.0f, 0.0f,0.0f,
0.3f, 0.0f,0.0f,

0.0f, -0.2f,0.0f,
0.3f, 0.0f,0.0f,
0.3f, -0.2f,0.0f,
```

```
// J
0.5f, 0.0f,0.0f,
0.5f,-0.2f,0.0f,
0.7f, -0.2f,0.0f,

0.7f, 0.0f,0.0f,
0.7f, -0.2f,0.0f,
0.5f, 0.0f,0.0f,

0.7f, -0.2f,0.0f,
0.7f, 0.7f,0.0f,
0.9f, -0.2f,0.0f,

0.9f, -0.2f,0.0f,
0.9f, 0.7f,0.0f,
0.7f, 0.7f,0.0f,

0.5f, 0.5f,0.0f,
0.5f, 0.7f,0.0f,
0.7f, 0.5f,0.0f,
```

Y por ultimo solo modifcamos el valor de los veetices para mostrar en pantalla

```
    glBindVertexArray(VAO);
    glDrawArrays(GL_TRIANGLES,0,80);
    glBindVertexArray(0);
    glUseProgram(0);

    glfwSwapBuffers(mainWindow);

}


return 0;
```

Donde tomaremos los valores desde 0 hasta el número de vértices que ocupamos y con esto tendríamos las letras en pantalla como se muestra en la siguiente imagen:



Conclusión:

No tuve algún inconveniente mas que el de estar agregando muchos vértices aunque considero que pude haber utilizado otra función para poder agilizar esto y no colocar tantos vértices pero no tuve mucho tiempo y no investigue mas a fondo en esto pero de ahí en fuera todo estuvo bastante sencillo además de las librerías que también me causaron un poco de confusión pero investigando salió todo a fondo.

REFERENCIAS:

- https://www.google.com/search?q=https://www.geeksforgeeks.org/random-color-generation-in-c/
- https://www.google.com/url?sa=E&source=gmail&q=https://www.glfw.org/documentation.html
- https://www.google.com/url?sa=E&source=gmail&q=http://www.opengl-tutorial.org/
- https://www.google.com/url?sa=E&source=gmail&q=https://learnopengl.com/