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: Organista Alvarez Ricardo

Nº de Cuenta: 421018596

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 04

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 17/02/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

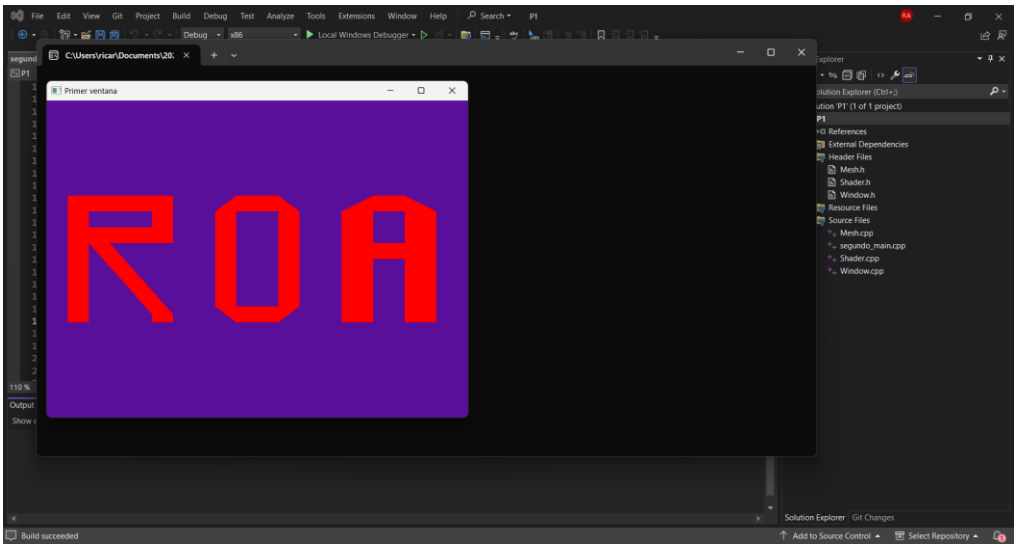
1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

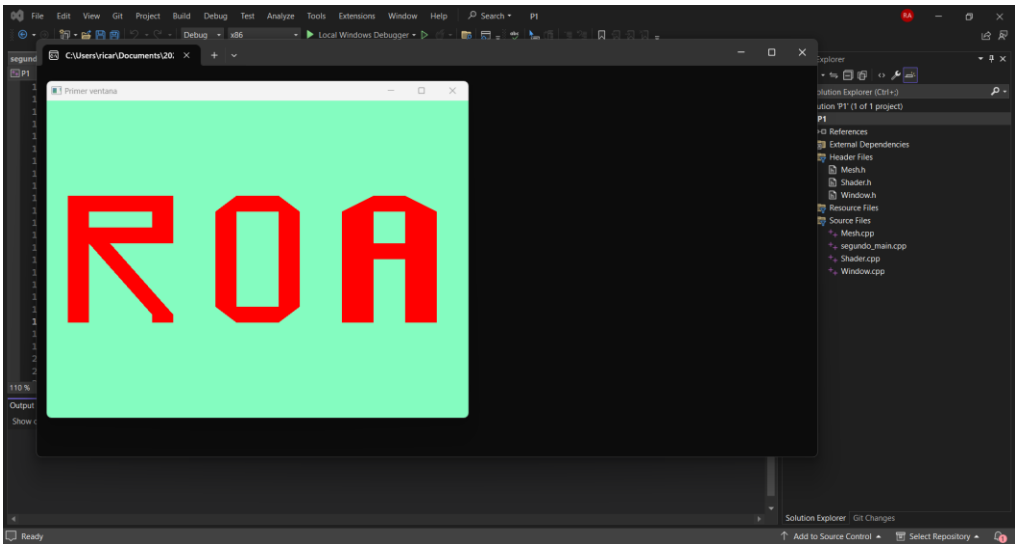
A. Ventana cambia de color de fondo de forma random tomando rango de colores RGB y con una periodicidad de dos segundos.

De manera similar a como lo hicimos en clase, ahora generamos números aleatorios flotantes entre 0 y 1 para cada valor del RGB (rojo, verde y azul). Para esto se usó la función rand, obteniendo el numero aleatorio para asignarlo a una variable que luego se le da a la función *glClearColor*, todo esto dentro del ciclo que esta activo mientras la ventana este abierta, para que en cada ejecución de esta sección del código el color cambie.

Para que el cambio de color fuera cada dos segundos use la función *Sleep* de la librería *Windows.h*, que recibe el tiempo en el que pausara el programa en milisegundos, por lo que al usar un valor de 2000 obtenemos la pausa de 2 segundos entre cada color.

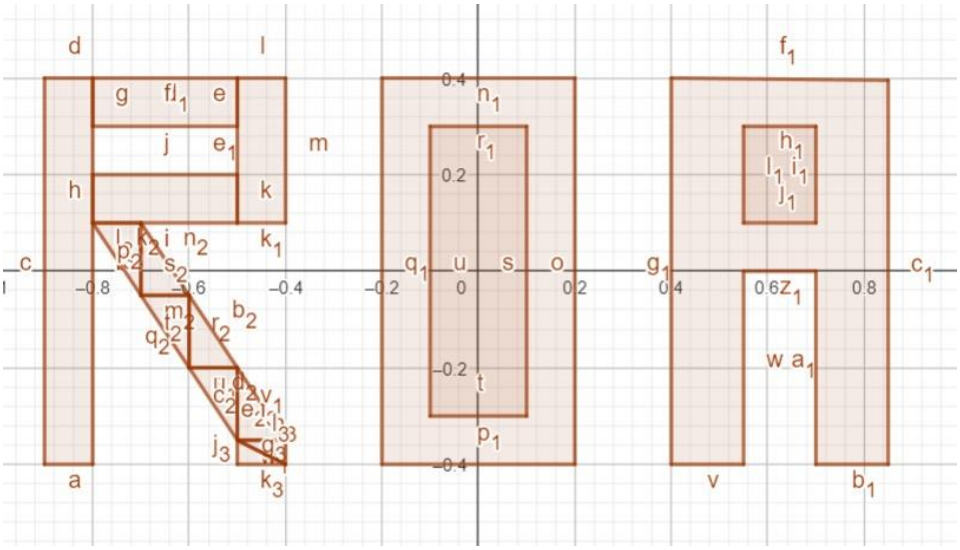
```
while (!glfwWindowShouldClose(mainWindow))
{
    //Recibir eventos del usuario
    glfwPollEvents();
    rojo = static_cast <float> (rand()) / static_cast <float> (RAND_MAX);
    verde = static_cast <float> (rand()) / static_cast <float> (RAND_MAX);
    azul = static_cast <float> (rand()) / static_cast <float> (RAND_MAX);
    Sleep(2000);
    //Limpiar la ventana
    glClearColor(rojo,verde,azul,1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    glUseProgram(shader);

    glBindVertexArray(VAO);
    glDrawArrays(GL_TRIANGLES,0,114);
    glBindVertexArray(0);

    glUseProgram(0);

    glfwSwapBuffers(mainWindow);

}
```


Primer ventana
ROA


Primer ventana
ROA

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

Para dibujar las letras hice uso de GeoGebra, dibujando las letras con polígonos para posteriormente obtener sus vértices. Dividí las letras en cuadrados ya que estos se pueden hacer juntando dos triángulos fácilmente.



Después de obtener los vértices se agregan al arreglo en el código y se especifica cuantos son a la función *glDrawArrays*.

El número de triángulos que use para cada letra es:

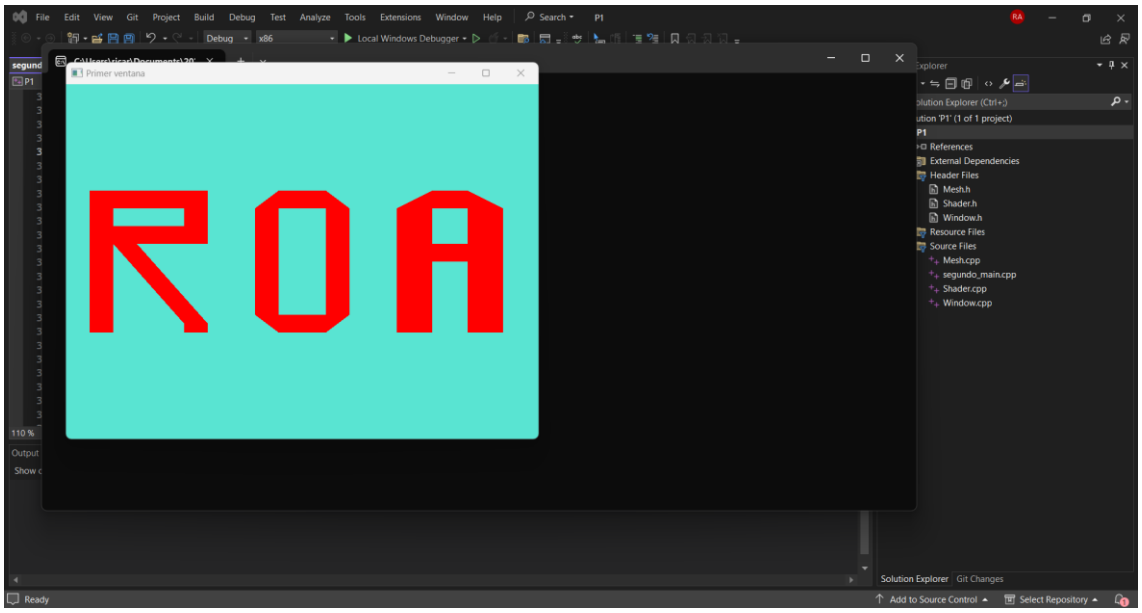
Letra R = 16 Triángulos (48 Vértices).

Letra O = 12 Triángulos (36 Vértices).

Letra A = 10 Triángulos (30 Vértices).

Usando un total de 114 vértices.

```
GLfloat vertices[] = {
    //x,y,z
    //R
    //Triangulo 1
    -0.9f,0.4f,0.0f, // C
    -0.8f,0.4f,0.0f, // D
    -0.9f,-0.4f,0.0f, // A
    //Triangulo 2
    -0.8f,0.4f,0.0f, // D
    -0.9f,-0.4f,0.0f, // A
    -0.8f,-0.4f,0.0f, // B
    //T3
    -0.8f,0.4f,0.0f, // D
    -0.8f,0.3f,0.0f, // G
    -0.5f,0.4f,0.0f, // E
    //T4
```



Los dos ejercicios se muestran en la misma pantalla.

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

El mayor problema que tuve fue generar el numero aleatorio en el rango de 0 a 1, ya que por defecto en el lenguaje C++ no existe una función para generar decimales aleatorios, por lo que fue necesario convertir el numero obtenido en un numero de punto flotante con la sintaxis *static_cast<float>(rand()),* y posteriormente normalizarlo para que estuviera en el rango de 0 a 1 dividiendo el valor obtenido entre *static_cast<float>(RAND_MAX)*.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

Estos ejercicios no tuvieron demasiada complejidad, el mayor problema sería generar los vértices para las letras, sin embargo, con la ayuda de herramientas como GeoGebra esta tarea se vuelve más sencilla limitando el problema a diseñar las letras y escribir los vértices dados por GeoGebra. Mientras que el ejercicio de cambiar de color es sencillo ya que se puede resolver utilizando librerías comunes de C++, con las que a este punto de la carrera ya hemos usado.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

La explicación fue buena ya que pude comprender el código necesario para la realización de la práctica. Aunque algunas partes del código siguen siendo difíciles de entender con la practica y el avance a lo largo del curso espero poder comprenderlo todo.

c. Conclusión

Gracias a esta práctica pude entender las partes básicas de OpenGL para el dibujo en 2D, con esto pude entender como funciona una de las formas de dibujo mediane triángulos, que es una parte básica de la computación gráfica, ya que la mayoría de los gráficos están hechos mediante polígonos, normalmente triángulos.

1. Bibliografía en formato APA

Random float number generation. (s. f.). Stack Overflow. https://stackoverflow.com/questions/686353/random-float-number-generation