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: JAVIER DE JESUS INFANTE MARTINEZ

Nº de Cuenta: 422534286

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 02 / 19 / 2025

CALIFICACIÓN: _____

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

```
//determinara el color que se usara y habra un contador para ver cada cuanto se resetea el color
float contador = 1.0f;

//Variables para control del tiempo
double lastTime = 0.0;
double currentTime = 0.0;

// Agregar estas variables globales en lugar de colorActual:
float colorR = 0.0f;
float colorG = 0.0f;
float colorB = 0.0f;
```

#include <stdlib.h>: Esta biblioteca se usa para varias funciones estándar, incluyendo rand(), que genera números aleatorios.

#include <time.h>: Se usa para manejar el tiempo en C y C++. En este caso, se usa para obtener el tiempo actual con time(NULL).

```
//Aletoriedad en cada reinicio del programa
srand(time(NULL));
lastTime = glfwGetTime(); // Inicializar el tiempo
```

srand(time(NULL)): Se usa para inicializar la semilla de la función rand(), lo que permite que los números generados sean diferentes en cada ejecución del programa.

l**astTime = glfwGetTime();: glfwGetTime()** es una función de la biblioteca GLFW que devuelve el tiempo actual en segundos. Se usa para inicializar la variable

lastTime, que servirá como referencia para medir intervalos de tiempo.

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

// Verificar si han pasado 2 segundos
if (currentTime - lastTime >= 2.0) {
    contador = 1.0f;  // Hacer que contador sea 1
    lastTime = currentTime;  // Actualizar el último tiempo
}
```

currentTime = glfwGetTime();: Se obtiene el tiempo actual. Esto con referencia al inicio del programa.

if (currentTime - lastTime >= 2.0): Se verifica si han pasado al menos 2 segundos desde lastTime.

- Si la condición se cumple:

  Se asigna 1.0f al contador para indicar que se debe cambiar el color.

  Se actualiza lastTime con el nuevo tiempo para que la verificación se haga nuevamente después de otros 2 segundos. Esto hace que lastTime tenga una referencia diferente a la anterior en cada ciclo.

```
if (contador >= 1.0f) {
    contador = 0.0f;
    // Generar tres números aleatorios entre 0.0 y 1.0
    colorR = (float)rand() / RAND_MAX;
    colorG = (float)rand() / RAND_MAX;
    colorB = (float)rand() / RAND_MAX;
}

glClearColor(colorR, colorG, colorB, 1.0f);
```

Si el contador es mayor o igual a 1.0f, se ejecuta el cambio de color, reiniciando el contador a 0.0f. Luego, se generan tres valores aleatorios entre 0.0 y 1.0 para colorR, colorG y colorB, utilizando rand() / RAND_MAX. Finalmente, glClearColor(colorR, colorG, colorB, 1.0f); actualiza el color de fondo de la ventana con los nuevos valores generados.

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

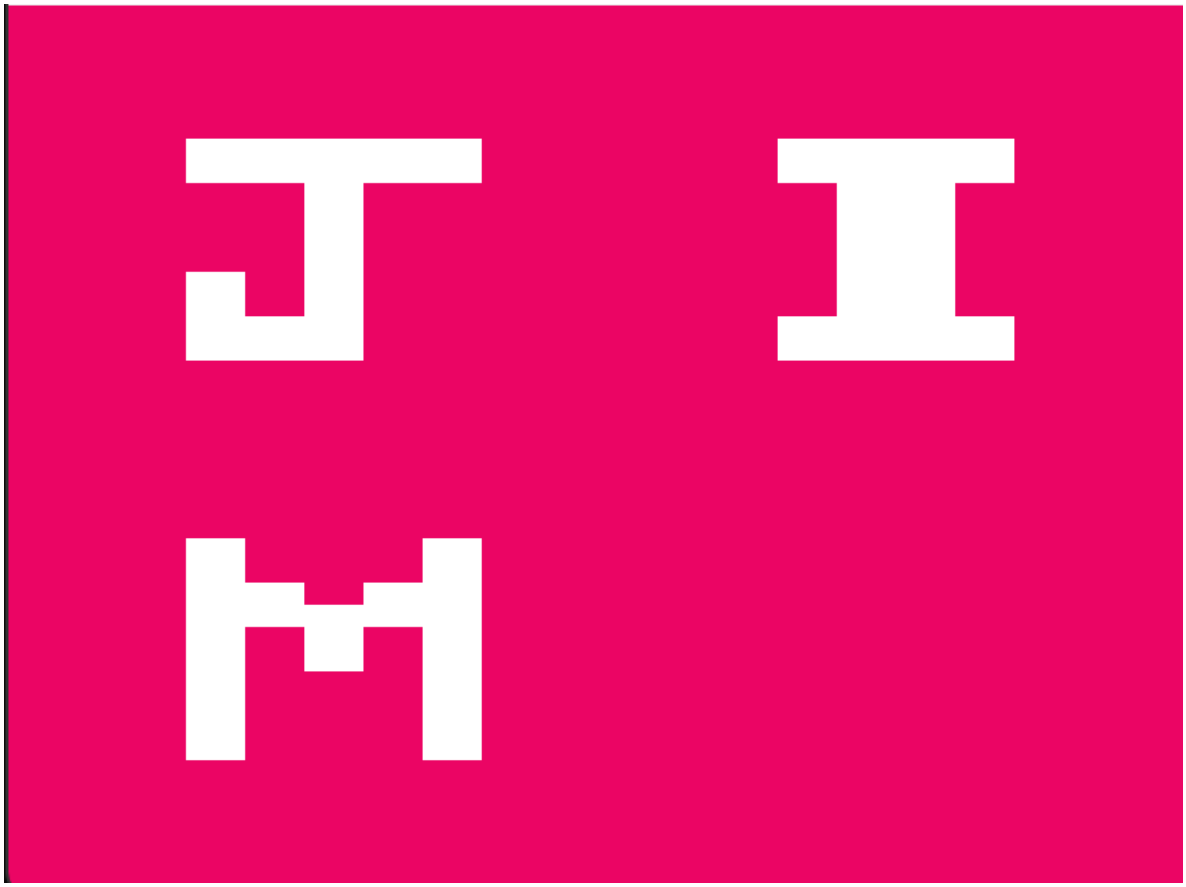
```
//Esta es la letra J

//Cola de la letra J
-0.6f, 0.4f, 0.0f,
-0.7f, 0.2f, 0.0f,
-0.6f, 0.3f, 0.0f,

-0.6f, 0.4f, 0.0f,
-0.7f, 0.2f, 0.0f,
-0.7f, 0.4f, 0.0f,

-0.7f, 0.2f, 0.0f,
-0.6f, 0.3f, 0.0f,
-0.5f, 0.3f, 0.0f,


//Parte media inferior de la letra J
-0.4f, 0.35f, 0.0f,
-0.4f, 0.2f, 0.0f,
-0.7f, 0.2f, 0.0f,

-0.5f, 0.7f, 0.0f,
-0.4f, 0.35f, 0.0f,
-0.5f, 0.3f, 0.0f,
```

Se crearon las letras "J", "I" y "M", y cada una fue seccionada en partes para facilitar la identificación de errores al posicionar un vértice en una coordenada incorrecta. Esta división permite una mejor visualización y ajuste de los elementos en la representación gráfica.

```
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES,0,93);
glBindVertexArray(0);
```

Se admiten 93 vértices según el cálculo basado en la cantidad total de triángulos utilizados, que fueron 31, multiplicados por 3 vértices por triángulo. Esto garantiza que cada triángulo tenga sus respectivos puntos definidos correctamente en la representación gráfica y evitando la falta de triángulos necesarios.

Resultado



Problemas

El principal problema inicial fue el cálculo del intervalo de 2 segundos, ya que en el ejercicio anterior no se había utilizado la biblioteca correspondiente para gestionar el tiempo. Como resultado, fue necesario modificar el código para garantizar que el cambio de color ocurriera cada 2 segundos. Además, la letra "M" originalmente tenía una representación deficiente, con solo 4 triángulos en la parte central, lo que afectaba su forma. Finalmente, se corrigió este problema aumentando la cantidad de triángulos a 7, logrando una mejor representación de la letra.

Conclusión

Esta práctica me pareció bastante buena, ya que combinó lo aprendido en clase sobre el uso de triángulos con formas que no son necesariamente fáciles de representar. Hubo muchas posibilidades para construir las letras, aunque elegí la opción que me pareció mejor, sin asegurar que fuera la más óptima.

Crear un plano cartesiano desde el inicio facilitó la detección de errores en la construcción de los triángulos y la identificación de coordenadas incorrectas. En cuanto al cambio de color, no encontré mayores dificultades más allá de incluir las bibliotecas necesarias para generar números aleatorios y asegurar que los colores cambiaran de forma impredecible en cada ejecución.

En general, la práctica está bien explicada y ofrece bastante libertad, ya que no impone restricciones de diseño, lo que permite pensar en la mejor manera de construir las letras según el criterio de cada persona. No considero que se necesite más explicación, salvo en lo relacionado con el ejercicio 1.

Podría decirse que el objetivo principal de la práctica es la visualización de las letras en el plano cartesiano, ya que, más allá de eso, el resto del trabajo es laborioso, pero no particularmente difícil.

Bibliografía

Gonzalez, T. (2024, 19 febrero). Como usar valores aleatorios en C. ACADEMIA SANROQUE. https://academiasanroque.com/como-usar-valores-aleatorios-en-c/

IBM i 7.5. (s. f.). https://www.ibm.com/docs/es/i/7.5?topic=functions-rand-rand-r-generate-random-number