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 1

NOMBRE COMPLETO: Araiza Valdés Diego Antonio

Nº de Cuenta: 423032833

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE: 2026-1

FECHA DE ENTREGA LÍMITE: 24/08/2025

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.
   1.1. Ventana cambia el color de fondo de forma random tomando rango de colores RGB y con una periodicidad de 2 segundos. (Verificar que al ejecutar el programa varias veces el orden de los colores si lo vean aleatorio y no siempre los mismos).

```
#include <random>
```

```
//Iniciar el RANDOM
std::random_device rd;
std::mt19937 rng(rd());
float upper_inclusive = std::nextafter(1.0f, std::numeric_limits<float>::max());
std::uniform_real_distribution<float> dist(0.0f, upper_inclusive);
//Iniciamos el valor de colores random
float RED= dist(rng), GREEN= dist(rng), BLUE= dist(rng);
```

Inicialicé un random que toma valores flotantes entre 0 y 1 inclusivo y declaré 3 variables para definir el color de la ventana, asignándoles un valor random a cada una.

```
#include <chrono>
```

```
//Iniciamos un cronómetro de 2 segundos
using clock_t = std::chrono::steady_clock;
auto last_change = clock_t::now();
constexpr auto INTERVAL = std::chrono::seconds(2);
```

Para el cambio de color, nos apoyamos de la biblioteca chrono. Primero inicializamos un reloj (asignándole un nombre corto) y creamos una variable que representa el ultimo cambio realizado (inicializándola con el tiempo actual) y otra un intervalo de 2 segundos.

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

    // Actualizar color si han pasado 2 segundos
    auto now = clock_t::now();
    if (now - last_change >= INTERVAL) {
        RED = dist(rng), GREEN = dist(rng), BLUE = dist(rng);
        last_change = now;
    }
```

Posteriormente dentro del ciclo while, declaramos la variable now para obtener el tiempo actual y comprobamos que desde el ultimo cambio realizado hayan pasado 2 segundos. De ser el caso actualizamos los valores de las variables del fondo a un nuevo valor random

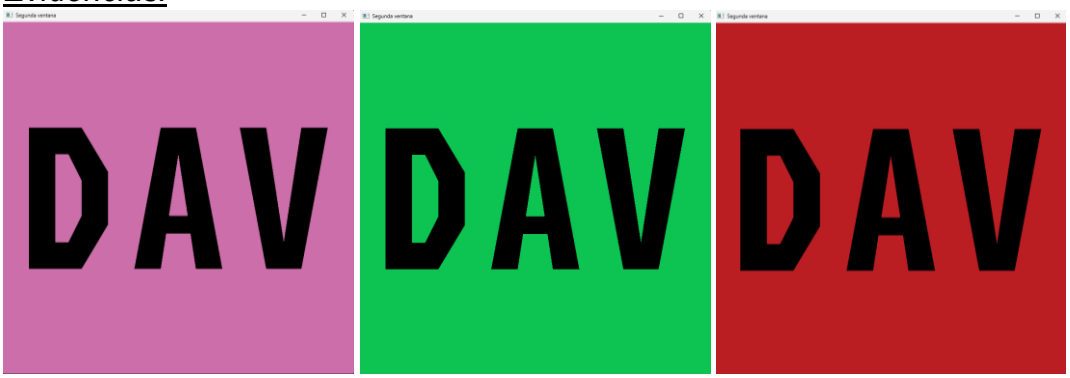
1.2.3 letras iniciales de sus nombres creadas a partir de triángulos, todas las letras son del mismo color.

```
GLfloat vertices[] = {
    //primera letra
    -0.85f, 0.4f, 0.0f,
    -0.85f,-0.4f, 0.0f,
    -0.7f,-0.4f, 0.0f,
    -0.85f, 0.4f, 0.0f,
    -0.7f,-0.4f, 0.0f,
    -0.7f, 0.4f, 0.0f,
    -0.7f, 0.4f, 0.0f,
    -0.7f, 0.25f, 0.0f,
    -0.55f, 0.25f, 0.0f,
    -0.7f, 0.4f, 0.0f,
    -0.55f, 0.25f, 0.0f,
    -0.55f, 0.4f, 0.0f,
    -0.625f, 0.25f, 0.0f,
    -0.55f, 0.1f, 0.0f,
    -0.55f, 0.25f, 0.0f,
    -0.55f, 0.4f, 0.0f,
    -0.55f, 0.15f, 0.0f,
    -0.4f, 0.15f, 0.0f,
    -0.55f, 0.15f, 0.0f,
    -0.55f,-0.15f, 0.0f,
    -0.4f,-0.15f, 0.0f,
    -0.55f, 0.15f, 0.0f,
    -0.4f,-0.15f, 0.0f,
    -0.4f, 0.15f, 0.0f,
    -0.7f,-0.25f, 0.0f,
    -0.7f,-0.4f, 0.0f,
    -0.55f,-0.4f, 0.0f,
    -0.7f,-0.25f, 0.0f,
    -0.55f,-0.4f, 0.0f,
    -0.55f,-0.25f, 0.0f,
    -0.625f,-0.25f, 0.0f,
    -0.55f,-0.25f, 0.0f,
    -0.55f,-0.1f, 0.0f,
    -0.55f,-0.15f, 0.0f,
    -0.55f,-0.4f, 0.0f,
    -0.4f,-0.15f, 0.0f,
    //segunda letra
    -0.25f,-0.4f, 0.0f,
    -0.1f,-0.4f, 0.0f,
    -0.1f, 0.4f, 0.0f,
    -0.1f, 0.25f,0.0f,
    -0.1f,-0.4f, 0.0f,
    0.00f, 0.25f, 0.0f,
    0.00f, 0.25f, 0.0f,
    0.1f,-0.4f, 0.0f,
    0.1f, 0.25f, 0.0f,
    0.1f, 0.4f, 0.0f,
    0.1f,-0.4f, 0.0f,
    0.25f,-0.4f, 0.0f,
    -0.1f, 0.4f, 0.0f,
    -0.1f, 0.25f, 0.0f,
    0.1f, 0.25f, 0.0f,
    -0.1f, 0.4f, 0.0f,
    0.1f, 0.25f, 0.0f,
    0.1f, 0.4f, 0.0f,
    -0.1f,-0.1f, 0.0f,
    -0.1f,-0.2f, 0.0f,
    0.1f,-0.2f, 0.0f,
    -0.1f,-0.1f, 0.0f,
    0.1f,-0.2f, 0.0f,
    0.1f,-0.1f, 0.0f,
    //tercera letra
    0.35f, 0.4f, 0.0f,
    0.5f,-0.4f, 0.0f,
    0.5f, 0.4f, 0.0f,
    0.5f, 0.4f, 0.0f,
    0.5f,-0.25f, 0.0f,
    0.6f,-0.25f, 0.0f,
    0.6f,-0.25f, 0.0f,
    0.7f,-0.25f, 0.0f,
    0.7f, 0.4f, 0.0f,
    0.7f, 0.4f, 0.0f,
    0.7f,-0.4f, 0.0f,
    0.85f, 0.4f, 0.0f,
    0.5f,-0.25f, 0.0f,
    0.5f,-0.4f, 0.0f,
    0.7f,-0.4f, 0.0f,
    0.5f,-0.25f, 0.0f,
    0.7f,-0.4f, 0.0f,
    0.7f,-0.25f, 0.0f,
```

```
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES,0,78);
glBindVertexArray(0);
```

Con ayuda de una hoja cuadriculada me fue posible seleccionar los vértices para realizar cada triangulo y formar así las letras D A y V, con un total de 78 vértices y 26 triángulos. Por lo que, dentro del while, se modificó el número de vértices a 78.

1.1. Evidencias:



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
   R: No hubo ningún problema a la hora de realizar los ejercicios.
3. Conclusión:
   1. Los ejercicios del reporte: Complejidad, Explicación.
      Respecto a los ejercicios realizados, creo que fueron bastante adecuados al nivel de lo visto en el laboratorio y el ejercicio realizado previamente, si bien tuve que investigar aparte por mi cuenta cómo utilizar random en C++, no fue la gran cosa y la función chrono la utilicé en el ejercicio, por lo que solo reutilicé. Con los ejercicios pude agarrar más práctica con el lenguaje C++ y acostumbrarme a 'dibujar' con OpenGL.
   2. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica.
      Habiendo realizado el previo y el ejercicio fue mucho más digerible entender el código, pero me gustaría volver a oír la explicación del mismo habiendo entendido ya conceptos como VAO, VBO y cómo funcionan los búferes.
   3. Conclusión

      Personalmente creo que se cumplieron los objetivos de la práctica. Con los ejercicios pude agarrar más práctica con el lenguaje C++ y acostumbrarme a 'dibujar' con OpenGL. Aprendí que en OpenGL se suele dibujar con triángulos y para ello se deben declarar sus vértices (una buena práctica es

hacerlo en contra de las manecillas de reloj), aprendí para que sirven los búferes y como trabajar con ellos, y tuve una pequeña introducción a OpenGL y GLSL. Además, aprendí a utilizar la librería random y chorno. Creo que fue una práctica bastante adecuada para ser la introducción a la computación gráfica.

Bibliografía en formato APA

- cppreference.com. (n.d.). std::uniform_real_distribution. Recuperado de https://en.cppreference.com/w/cpp/numeric/random/uniform_real_distribution.html
  FreeCodeCamp. (2023, diciembre 4). C++ std::chrono API. Recuperado de https://www.freecodecamp.org/news/cpp-std-chrono-api
- Stack Overflow. (2013, septiembre 12). Generate random numbers using C++11 random library. Stack Overflow. Recuperado de https://stackoverflow.com/questions/19665818/generate-random-numbers-using-c11-random-library