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: Francisco Galindo Mena

Nº de Cuenta: 320054336

GRUPO DE LABORATORIO: 13

GRUPO DE TEORÍA: 06

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 28 agosto 2025

CALIFICACIÓN: _____

Reporte de práctica 01. Introducción a OpenGL

Francisco Galindo Mena

Semestre 2026-1

1. Ejercicios de la práctica

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

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 simultanea y están en el mismo main

Utilizando la función `glfwGetTime()`, pudo saberse cuánto tiempo había pasado entre un cambio de color y el siguiente para que todo fuera constante. La ventaja que tiene utilizar una función de este tipo es que funcionará independientemente de la taza de refresco del monitor o de la tasa de cuadros por segundo de la que sea capaz la GPU en la computadora que lo ejecute. Se utiliza la función `rand()`, (con la debida llamada a `srand()`) para cambiar el color de fondo aleatoreamente.

Figura 1: Ejecución del ejercicio

El código que fue modificado para el cambio de color fue el siguiente:

```
srand(time(NULL));
// ...
double timeSinceColorChange = glfwGetTime();
float red = 1.0f, green = 0.0f, blue = 0.0f;

while (!glfwWindowShouldClose(mainWindow)) {
    //Recibir eventos del usuario
    glfwPollEvents();

    // Checa qué hora es
    double now = glfwGetTime();

    // Cada dos segundos, cambia de color
    if (now - timeSinceColorChange >= 2) {
        timeSinceColorChange = now;
        red = (float)rand()/((float)RAND_MAX);
        green = (float)rand()/((float)RAND_MAX);
        blue = (float)rand()/((float)RAND_MAX);
    }

    glClearColor(red, green, blue, 1.0f);

    // ...
    // Dibuja la cantidad de vértices que tengas en el arreglo
    glDrawArrays(GL_TRIANGLES,0,sizeof(vertices) / sizeof(GL_FLOAT));
    // ...
}
```

Los vértices de los triángulos correspondientes a mis iniciales son estos:

```
// Este arreglo ahora es una variable global, para poder acceder a
// ella en el bloque de código que mostré hace un rato
GLfloat vertices[] = {
    // F
    -1.0f, 1.0f, 0.0f,
    -0.5f, 1.0f, 0.0f,
    -1.0f, 0.875f, 0.0f,

    -1.0f, 0.875f, 0.0f,
    -0.5f, 1.0f, 0.0f,
    -0.5f, 0.875f, 0.0f,


    -1.0f, 1.0f, 0.0f,
    -0.875f, 1.0f, 0.0f,
    -1.0f, 0.25f, 0.0f,
```

```
-0.875f, 0.25f, 0.0f,
-0.875f, 1.0f, 0.0f,
-1.0f, 0.25f, 0.0f,


-1.0f, 0.75f, 0.0f,
-0.625f, 0.75f, 0.0f,
-1.0f, 0.625f, 0.0f,

-0.625f, 0.75f, 0.0f,
-1.0f, 0.625f, 0.0f,
-0.625f, 0.625f, 0.0f,

// G
-0.375f, 0.875f, 0.0f,
-0.25f, 0.875f, 0.0f,
-0.375f, 0.375f, 0.0f,

-0.25f, 0.375f, 0.0f,
-0.25f, 0.875f, 0.0f,
-0.375f, 0.375f, 0.0f,


-0.25f, 1.0f, 0.0f,
0.25f, 1.0f, 0.0f,
-0.25f, 0.875f, 0.0f,

0.25f, 0.875f, 0.0f,
0.25f, 1.0f, 0.0f,
-0.25f, 0.875f, 0.0f,


-0.25f, 0.375f, 0.0f,
0.125f, 0.375f, 0.0f,
-0.25f, 0.25f, 0.0f,

0.125f, 0.25f, 0.0f,
0.125f, 0.375f, 0.0f,
-0.25f, 0.25f, 0.0f,


0.125f, 0.75f, 0.0f,
0.25f, 0.75f, 0.0f,
0.125f, 0.375f, 0.0f,

0.25f, 0.375f, 0.0f,
0.25f, 0.75f, 0.0f,
0.125f, 0.375f, 0.0f,
```

```
0.0f, 0.75f, 0.0f,
0.125f, 0.75f, 0.0f,
0.0f, 0.625f, 0.0f,

0.125f, 0.625f, 0.0f,
0.125f, 0.75f, 0.0f,
0.0f, 0.625f, 0.0f,

// M
0.5f, 1.0f, 0.0f,
0.375f, 1.0f, 0.0f,
0.5f, 0.25f, 0.0f,

0.375f, 0.25f, 0.0f,
0.375f, 1.0f, 0.0f,
0.5f, 0.25f, 0.0f,


1.0f, 1.0f, 0.0f,
0.875f, 1.0f, 0.0f,
1.0f, 0.25f, 0.0f,

0.875f, 0.25f, 0.0f,
0.875f, 1.0f, 0.0f,
1.0f, 0.25f, 0.0f,

0.625f, 0.875f, 0.0f,
0.5f, 0.875f, 0.0f,
0.625f, 0.75f, 0.0f,

0.5f, 0.75f, 0.0f,
0.5f, 0.875f, 0.0f,
0.625f, 0.75f, 0.0f,


0.875f, 0.875f, 0.0f,
0.75f, 0.875f, 0.0f,
0.875f, 0.75f, 0.0f,

0.75f, 0.75f, 0.0f,
0.75f, 0.875f, 0.0f,
0.875f, 0.75f, 0.0f,


0.75f, 0.75f, 0.0f,
0.625f, 0.75f, 0.0f,
```

```
    0.75f, 0.625f, 0.0f,

    0.625f, 0.625f, 0.0f,
    0.625f, 0.75f, 0.0f,
    0.75f, 0.625f, 0.0f,
};
```

Para cambiar el color de las letras a gris, se modificó el *fragment shader*:

```
//Fragment Shader
//recibir Vcolor y dar de salida color
static const char* fShader = "                  \n\
#version 330                                    \n\
out vec4 color;                                 \n\
void main()                                     \n\
{                                               \n\
        color = vec4(0.5f,0.5f,0.5f,1.0f);      \n\
}";
```

2. Problemas a la hora de hacer estos ejercicios

La complicación principal de esta práctica está en tener la intuición para saber en dónde poner cada vértice de los triángulos para que las letras aparezcan donde uno quiere. En mi caso, me apoyé copiando la fuente de *Minecraft*, donde las letras están compuestas por píxeles individuales, así que cada sección rectangular podía ser dibujada con dos triángulos.

Más allá de eso, el código ya existente fue de mucha ayuda, pues lo único que hay que modificar es la cantidad y posición de los vértices a dibujar

3. Conclusión

Los ejercicios realizados en esta práctica son un buen punto de inicio para ir aprendiendo cómo funciona OpenGL, si bien el ejercicio de programación no es demasiado complicado, saber qué mover representa bastantes conocimientos previos (saber qué es un VAO, conocer la *pipeline* gráfica, etc.). De esta forma, sí puede ser algo confuso en el sentido de que hay que leer, como lo que se encuentra en ((De Vries, 2020)) mucho y poner mucha atención a la explicación del profesor para saber qué hacer.

La explicación me pareció buena, puede parecer rápida a primera vista pero la realidad es que, tras hacer el previo, sí se tienen los conocimientos necesarios para hacer la práctica y entender la explicación en el laboratorio. Quizás sería bueno que en el previo se dejara un ejercicio de programación para que la explicación en el laboratorio fuera más corta y se hicieran más ejercicios.

En conclusión, la primera práctica ha permitido tener los conocimientos básicos sobre OpenGL y el tipo de trabajo que se realizará en la materia.

Referencias

De Vries, J. (2020, junio). Learn OpenGL.