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: Sánchez Meza Ariadna Osiris

Nº de Cuenta: 316113816

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE 2026-2

FECHA DE ENTREGA LÍMITE: 22/02/2026

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.

- 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).

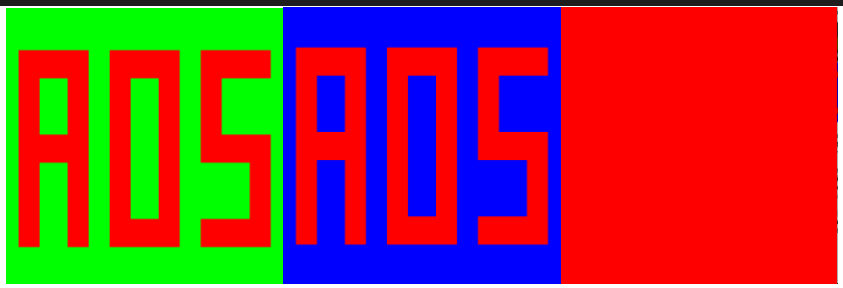
```
float tiempo = glfwGetTime();
static int ultimoSegundo = -1; // 'static' para que el valor persista entre ciclos
int segundoActual = (int)tiempo;

if (segundoActual != ultimoSegundo) {
    int colorAzar = rand() % 3; // de forma aleatoria se asigna un número entre 0 y 2 para elegir el color

    if (colorAzar == 0)      glClearColor(1.0f, 0.0f, 0.0f, 1.0f); // Red
    else if (colorAzar == 1) glClearColor(0.0f, 1.0f, 0.0f, 1.0f); // green
    else                     glClearColor(0.0f, 0.0f, 1.0f, 1.0f); // blueee

    ultimoSegundo = segundoActual;
}

glClear(GL_COLOR_BUFFER_BIT);
```



Por lo que entendí en clase el espacio de tiempo que se le da a cada cambio de color es para que no parezcan luces estridentes tipo flasheos y se puedan distinguir los colores unos de otros, en la primera modificación de esta parte del código tenía un segundo de intervalo esta vez se incremento a dos segundos y los colores fueron de forma aleatoria. Esto lo logre utilizando rand para crear la elección aleatoria entre 0 y 2 ya que tenemos 3 colores (verde, azul y rojo) y teniendo como color base las iniciales de mi nombre el color rojo.

❖ 3 letras iniciales de sus nombres creadas a partir de triángulos, acomodadas en forma diagonal de abajo hacia arriba, todas las letras son del mismo color. Los dos ejercicios se muestran de forma simultánea y están en el mismo main.

```
void CrearTriangulo() {
    GLfloat vertices[] = {
        // A Para que quede del lado izquierdo va de -0.9 a -0.4
        -0.9f, -0.7f, 0.0f,  -0.75f, -0.7f, 0.0f,  -0.75f,  0.7f, 0.0f, //Manita izquierda
        -0.9f, -0.7f, 0.0f,  -0.75f,  0.7f, 0.0f,  -0.9f,   0.7f, 0.0f,
        -0.55f, -0.7f, 0.0f, -0.4f,  -0.7f, 0.0f,  -0.4f,   0.7f, 0.0f, //Manitaa Derecha
        -0.55f, -0.7f, 0.0f, -0.4f,   0.7f, 0.0f,  -0.55f,  0.7f, 0.0f,
        -0.75f,  0.7f, 0.0f, -0.55f,  0.7f, 0.0f,  -0.55f,  0.5f, 0.0f, //techo
        -0.75f,  0.7f, 0.0f, -0.55f,  0.5f, 0.0f,  -0.75f,  0.5f, 0.0f,
        -0.75f,  0.1f, 0.0f, -0.55f,  0.1f, 0.0f,  -0.55f, -0.1f, 0.0f, //Barra de en medio
        -0.75f,  0.1f, 0.0f, -0.55f, -0.1f, 0.0f,  -0.75f, -0.1f, 0.0f,

        // O para que quede en el centro es de -0.25  a 0.25
        -0.25f, -0.7f, 0.0f, -0.1f,  -0.7f, 0.0f,  -0.1f,   0.7f, 0.0f, //barra izquierda
        -0.25f, -0.7f, 0.0f, -0.1f,   0.7f, 0.0f,  -0.25f,  0.7f, 0.0f,
         0.1f,  -0.7f, 0.0f,  0.25f, -0.7f, 0.0f,   0.25f,  0.7f, 0.0f, //barra derecha
         0.1f,  -0.7f, 0.0f,  0.25f,  0.7f, 0.0f,   0.1f,   0.7f, 0.0f,
        -0.1f,   0.7f, 0.0f,  0.1f,   0.7f, 0.0f,   0.1f,   0.5f, 0.0f, //arriba
        -0.1f,   0.7f, 0.0f,  0.1f,   0.5f, 0.0f,  -0.1f,   0.5f, 0.0f,
        -0.1f,  -0.5f, 0.0f,  0.1f,  -0.5f, 0.0f,   0.1f,  -0.7f, 0.0f, //abajo
        -0.1f,  -0.5f, 0.0f,  0.1f,  -0.7f, 0.0f,  -0.1f,  -0.7f, 0.0f,

        // S para que quede en la derecha va de 0.4 a 0.9
         0.4f,   0.7f, 0.0f,  0.9f,   0.7f, 0.0f,   0.9f,   0.5f, 0.0f, //arriba
         0.4f,   0.7f, 0.0f,  0.9f,   0.5f, 0.0f,   0.4f,   0.5f, 0.0f,
         0.4f,   0.5f, 0.0f,  0.55f,  0.5f, 0.0f,   0.55f,  0.1f, 0.0f, //vertical izquiera (cabeza)
         0.4f,   0.5f, 0.0f,  0.55f,  0.1f, 0.0f,   0.4f,   0.1f, 0.0f,
         0.4f,   0.1f, 0.0f,  0.9f,   0.1f, 0.0f,   0.9f,  -0.1f, 0.0f, //En medio
```

Para poder visualizar como iba a “cortar” cada letra primero las dibuje completas en un cuaderno y fui trazando triángulos de tal forma que respetaran la forma de un equilátero para que los vértices se de cada triangulo se pudieran unir y de forma más simple comenzar a formar la figura, así que a partir de los rectángulos o barras formadas por las partes de cada letra sume los vértices lo que dio un total de 78 vértices, que se modifica en esta función glDrawArrays(GL_TRIANGLES, 0, 78); para indicarle que serán 26 triángulos, utilizando los ejes imaginarios X y Y de (-1,1)(1,.1) respectivamente para cada eje positivo y negativo de los cuatro cuadrantes, en el caso de Z siempre fue 0 porque estamos en un plano 2D.

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 principal problema que tuve y que me llevo mucho tiempo poder comprender como funcionaba e imaginármelo fue como imaginar las coordenadas dentro de la ventana ya que al no ver mediciones me fue complicado dimensionar el tamaño de cada barra, pero después para más complicado la dimensiones de como de iba a dividir esa barra en 3 triángulos formando los 6 vértices que se necesitaban para darle forma y después de esto unirlo con el resto de la figura. Lo soluciones dibujando muchas veces al tanteo los triángulos, incluso lo que hice fue que con

una hoja más o menos construir un cuadrado tipo ventana para poder verlo en físico. Después de eso fue a prueba y error modificar las coordenadas de los vértices para ver cómo se hacen las uniones.

3.- Conclusión:

Para mi hubiese sido mejor hacer más ejercicios con otras figuras para poderlas partirlas mejor e incluso practicarlas un poco más en papel, pero fuera de eso me gusto elaborar la práctica porque a pesar de que todavía es 2D es divertido como se puede jugar con este tipo de modificaciones en el caso de colores y poder expresar cualquier dibujo en 2D en ese formato.

Bibliografía.

- Khronos Group. (2017). *The OpenGL graphics system: A specification (Version 4.6)*. https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf