
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:** Jiménez Enriquez Rubén Pedro

**Nº de Cuenta:** 318056832

**GRUPO DE LABORATORIO:** 03

**GRUPO DE TEORÍA:** 06

**SEMESTRE** 2026-2

**FECHA DE ENTREGA LÍMITE:** 22 de febrero del 2026

**CALIFICACIÓN: _____**

## REPORTE DE PRÁCTICA:

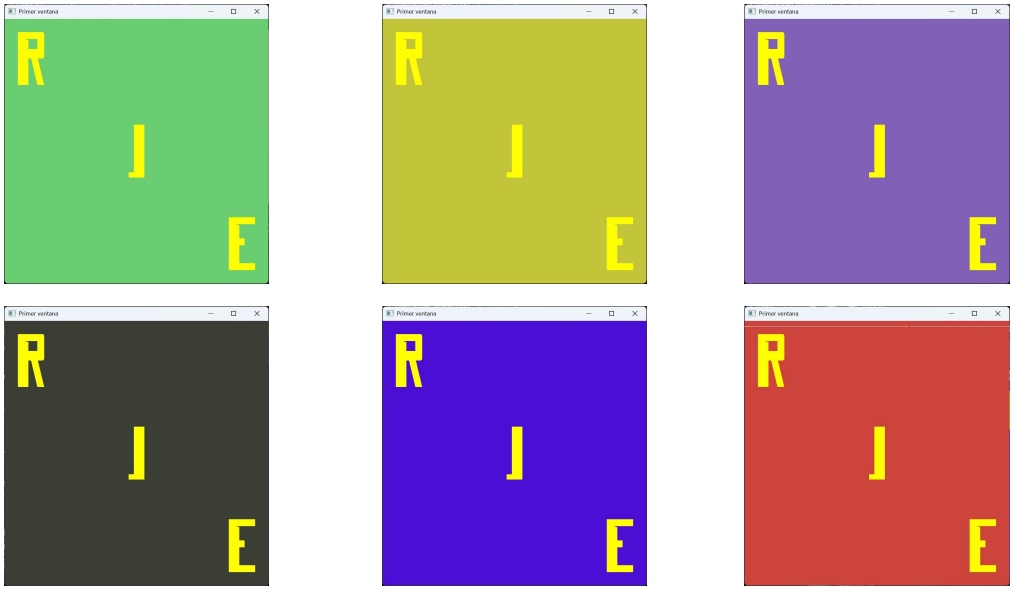
1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

I. 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).

I.I. Bloques de código generados.

```
367         //Se inicializan las variables utilizadas para cambiar el color aleatoriamente
368         float r, g, b;
369         float ultimoCambio = 0.0f;//En esta variable se guarda el tiempo en que se actualizó por última ve
370         srand(time(NULL));//Inicialización de la semilla para rand
371
372         //Se asigna un primer color aleatorio a cada variable r,g,b
373         //Se divide el valor aleatorio obtenido entre RAND_MAX para obtener un número aleatorio entre 0 y
374         r = static_cast<float>(rand()) / RAND_MAX;
375         g = static_cast<float>(rand()) / RAND_MAX;
376         b = static_cast<float>(rand()) / RAND_MAX;
377
378         //Loop mientras no se cierra la ventana
379         while (!glfwWindowShouldClose(mainWindow))
380         {
381             float tiempo = glfwGetTime();//Devuelve el tiempo en segundos desde que se inicializó GLFW
382
383             //Si la diferencia entre el tiempo en que se realizó el último cambio de color y
384             //el tiempo actual es mayor o igual a 2.0f, entonces se randomiza el color
385             if (tiempo - ultimoCambio >= 2.0f) {
386                 ultimoCambio = tiempo;
387                 r = static_cast<float>(rand()) / RAND_MAX;
388                 g = static_cast<float>(rand()) / RAND_MAX;
389                 b = static_cast<float>(rand()) / RAND_MAX;
390             }
391
392             //Actualizamos el color de fondo
393             glClearColor(r, g, b, 1.0f);
394             glClear(GL_COLOR_BUFFER_BIT);
```

I.II. Capturas de pantalla de la ejecución.



### I.III. Comentarios

Creé el repositorio para control de versiones como última actividad para el desarrollo de esta práctica, motivo por el cual en las capturas de ejecución ya aparecen las letras del siguiente ejercicio. En el reporte de la siguiente práctica corregiré esto, de forma que pueda mostrar las capturas de ejecución por ejercicio conforme los vaya desarrollando.

Con respecto al ejercicio en sí, este primero fue relativamente sencillo, ya que estoy algo familiarizado con el lenguaje C++, tuve pocas complicaciones (mismas que detallaré en la sección correspondiente más adelante), pero logré resolverlas rápidamente.

## II. 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.

### II.I. Bloques de código generados.

```
void CrearTriangulo() //Le voy a pasar datos con los que yo quiero que después dibuje un triángulo
{
    GLfloat vertices[] = {
        /*Letra R*/
        // t12 = Polygon(T, R, W1)
        -0.9f, 0.5f, 0.0f,     // T
        -0.9f, 0.9f, 0.0f,     // R
        -0.848f, 0.848f, 0.0f, // W1

        // t13 = Polygon(W1, R, J1)
        -0.848f, 0.848f, 0.0f, // W1
        -0.9f, 0.9f, 0.0f,     // R
        -0.74f, 0.9f, 0.0f,    // J1

        // t14 = Polygon(W1, I1, J1)
        -0.848f, 0.848f, 0.0f, // W1
        -0.82f, 0.848f, 0.0f,  // I1
        -0.74f, 0.9f, 0.0f,    // J1

        // t15 = Polygon(I1, J1, T1)
        -0.82f, 0.848f, 0.0f,  // I1
        -0.74f, 0.9f, 0.0f,    // J1
        -0.752f, 0.848f, 0.0f, // T1

        // t16 = Polygon(T1, J1, K1)
        -0.752f, 0.848f, 0.0f, // T1
        -0.74f, 0.9f, 0.0f,    // J1
        -0.72f, 0.9f, 0.0f,    // K1

        // t17 = Polygon(T1, L1, K1)
        -0.752f, 0.848f, 0.0f, // T1
        -0.71f, 0.899f, 0.0f,  // L1
        -0.72f, 0.9f, 0.0f,    // K1

        // t18 = Polygon(T1, L1, M1)
        -0.752f, 0.848f, 0.0f, // T1
        -0.71f, 0.899f, 0.0f,  // L1
        -0.704f, 0.896f, 0.0f, // M1

        // t19 = Polygon(T1, M1, N1)
        -0.752f, 0.848f, 0.0f, // T1
        -0.704f, 0.896f, 0.0f, // M1
        -0.701f, 0.89f, 0.0f,  // N1
```

```
        // t20 = Polygon(T1, N1, S)
        -0.752f, 0.848f, 0.0f, // T1
        -0.701f, 0.89f, 0.0f,  // N1
        -0.7f, 0.88f, 0.0f,    // S

        // t21 = Polygon(T1, S, O1)
        -0.752f, 0.848f, 0.0f, // T1
        -0.7f, 0.88f, 0.0f,    // S
        -0.7f, 0.72f, 0.0f,    // O1

        // t22 = Polygon(U1, T1, O1)
        -0.752f, 0.772f, 0.0f, // U1
        -0.752f, 0.848f, 0.0f, // T1
        -0.7f, 0.72f, 0.0f,    // O1

        // t23 = Polygon(U1, Q1, O1)
        -0.752f, 0.772f, 0.0f, // U1
        -0.701f, 0.71f, 0.0f,  // Q1
        -0.7f, 0.72f, 0.0f,    // O1

        // t24 = Polygon(U1, S1, Q1)
        -0.752f, 0.772f, 0.0f, // U1
        -0.704f, 0.704f, 0.0f, // S1
        -0.701f, 0.71f, 0.0f,  // Q1

        // t25 = Polygon(U1, R1, S1)
        -0.752f, 0.772f, 0.0f, // U1
        -0.71f, 0.701f, 0.0f,  // R1
        -0.704f, 0.704f, 0.0f, // S1

        // t26 = Polygon(U1, P1, R1)
        -0.752f, 0.772f, 0.0f, // U1
        -0.72f, 0.7f, 0.0f,    // P1
        -0.71f, 0.701f, 0.0f,  // R1

        // t27 = Polygon(U1, G1, P1)
        -0.752f, 0.772f, 0.0f, // U1
        -0.748f, 0.7f, 0.0f,   // G1
        -0.72f, 0.7f, 0.0f,    // P1

        // t28 = Polygon(H1, G1, U1)
        -0.82f, 0.772f, 0.0f,  // H1
        -0.748f, 0.7f, 0.0f,   // G1
        -0.752f, 0.772f, 0.0f, // U1
```

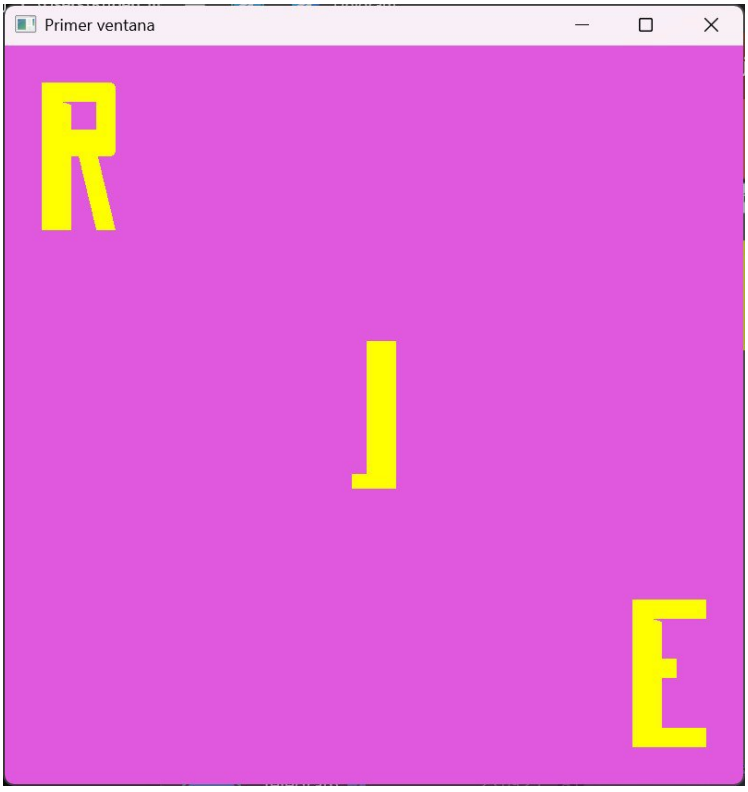
```
        // t29 = Polygon(H1, E1, G1)
        -0.82f, 0.772f, 0.0f,  // H1
        -0.8f, 0.7f, 0.0f,     // E1
        -0.748f, 0.7f, 0.0f,   // G1

        // t30 = Polygon(H1, W, E1)
        -0.82f, 0.772f, 0.0f,  // H1
        -0.82f, 0.7f, 0.0f,    // W
        -0.8f, 0.7f, 0.0f,     // E1

        // t31 = Polygon(W1, H1, T)
        -0.848f, 0.848f, 0.0f, // W1
        -0.82f, 0.772f, 0.0f,  // H1
        -0.9f, 0.5f, 0.0f,     // T

        // t32 = Polygon(H1, V, T)
        -0.82f, 0.772f, 0.0f,  // H1
        -0.82f, 0.5f, 0.0f,    // V
        -0.9f, 0.5f, 0.0f,     // T
```

```
// t33 = Polygon(E1, F1, U)
-0.8f, 0.7f, 0.0f,     // E1
-0.752f, 0.5f, 0.0f,   // F1
-0.7f, 0.5f, 0.0f,     // U

// t34 = Polygon(E1, G1, U)
-0.8f, 0.7f, 0.0f,     // E1
-0.748f, 0.7f, 0.0f,   // G1
-0.7f, 0.5f, 0.0f,     // U

// t35 = Polygon(W1, V1, H1)
-0.848f, 0.848f, 0.0f, // W1
-0.82f, 0.84f, 0.0f,   // V1
-0.82f, 0.772f, 0.0f,   // H1

/*Letra J*/
// t36 = Polygon(Z1, F2, Z)
-0.02f, 0.2f, 0.0f,    // Z1
0.06f, -0.2f, 0.0f,    // F2
0.06f, 0.2f, 0.0f,    // Z
```

```
// t37 = Polygon(H2, Z1, F2)
-0.02f, -0.16f, 0.0f,    // H2
-0.02f, 0.2f, 0.0f,    // Z1
0.06f, -0.2f, 0.0f,    // F2

// t38 = Polygon(E2, H2, F2)
-0.06f, -0.2f, 0.0f,    // E2
-0.02f, -0.16f, 0.0f,    // H2
0.06f, -0.2f, 0.0f,    // F2

// t39 = Polygon(G2, E2, H2)
-0.06f, -0.16f, 0.0f,    // G2
-0.06f, -0.2f, 0.0f,    // E2
-0.02f, -0.16f, 0.0f,     // H2

/*Letra E*/

// t1 = Polygon(G, E, J)
0.7f, -0.9f, 0.0f,      // G
0.7f, -0.5f, 0.0f,      // E
0.752f, -0.552f, 0.0f,  // J

// t2 = Polygon(E, F, J)
0.7f, -0.5f, 0.0f,      // E
0.9f, -0.5f, 0.0f,      // F
0.752f, -0.552f, 0.0f,  // J

// t3 = Polygon(J, F, I)
0.752f, -0.552f, 0.0f,  // J
0.9f, -0.5f, 0.0f,      // F
0.9f, -0.552f, 0.0f,    // I

// t4 = Polygon(J, L, O)
0.752f, -0.552f, 0.0f,  // J
0.78f, -0.66f, 0.0f,    // L
0.78f, -0.712f, 0.0f,   // O

// t5 = Polygon(J, O, G)
0.752f, -0.552f, 0.0f,  // J
0.78f, -0.712f, 0.0f,   // O
0.7f, -0.9f, 0.0f,      // G
```

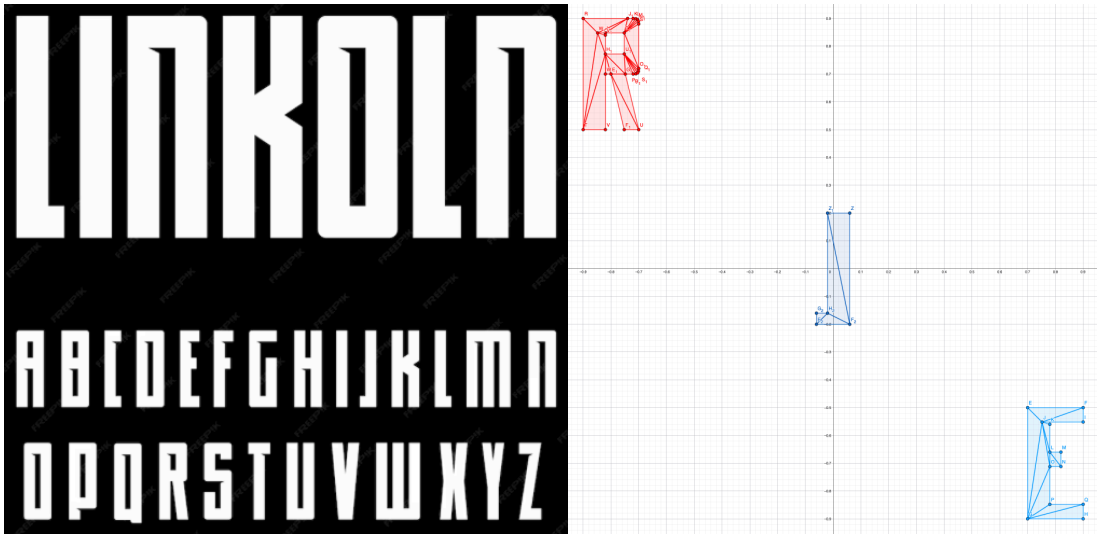
```
        // t6 = Polygon(L, N, O)
        0.78f, -0.66f, 0.0f,    // L
        0.82f, -0.712f, 0.0f,   // N
        0.78f, -0.712f, 0.0f,   // O

        // t7 = Polygon(L, M, N)
        0.78f, -0.66f, 0.0f,    // L
        0.82f, -0.66f, 0.0f,    // M
        0.82f, -0.712f, 0.0f,   // N

        // t8 = Polygon(O, P, G)
        0.78f, -0.712f, 0.0f,   // O
        0.78f, -0.848f, 0.0f,   // P
        0.7f, -0.9f, 0.0f,      // G

        // t9 = Polygon(G, P, Q)
        0.7f, -0.9f, 0.0f,      // G
        0.78f, -0.848f, 0.0f,   // P
        0.9f, -0.848f, 0.0f,    // Q

        // t10 = Polygon(G, H, Q)
        0.7f, -0.9f, 0.0f,      // G
        0.9f, -0.9f, 0.0f,      // H
        0.9f, -0.848f, 0.0f,    // Q

        // t11 = Polygon(J, K, L)
        0.752f, -0.552f, 0.0f,  // J
        0.78f, -0.56f, 0.0f,    // K
        0.78f, -0.66f, 0.0f     // L

    };//Arreglo de vértices que me van a ayudar para dibujar.
```

II.II. Capturas de pantalla de la ejecución.



II.III.Comentarios

Para esta actividad como reitero en la sección de problemas, hice uso de GeoGebra en línea para dibujar las letras de forma precisa. También tomé como base la tipografía que muestro a continuación, lado a lado con mis dibujos, misma que se encuentra referenciada en la bibliografía:




Una vez que tenía dibujada cada letra, GeoGebra permite definir el número de decimales con el que quieres que se redondeen las coordenadas. Elegí redondear a 3 decimales, lo cual me permitió hacer la lista de puntos con un formato: 0.*Xf,0.Xf,0.Xf, //etiqueta*. Muestro a continuación las listas de puntos obtenidas con GeoGebra:

Letra R

| | |
|---|---|
| $E_1 = (-0.8, 0.7)$ | $R_1 = (-0.71, 0.701)$ |
| $F_1 = (-0.752, 0.5)$ | $S = (-0.7, 0.88)$ |
| $G_1 = (-0.748, 0.7)$ | $S_1 = (-0.704, 0.704)$ |
| $H_1 = (-0.82, 0.772)$ | $T = (-0.9, 0.5)$ |
| $I_1 = (-0.82, 0.848)$ | $T_1 = (-0.752, 0.848)$ |
| $J_1 = (-0.74, 0.9)$ | $U = (-0.7, 0.5)$ |
| $K_1 = (-0.72, 0.9)$ | $U_1 = (-0.752, 0.772)$ |
| $L_1 = (-0.71, 0.899)$ | $V = (-0.82, 0.5)$ |
| $M_1 = (-0.704, 0.896)$ | $V_1 = (-0.82, 0.84)$ |
| $N_1 = (-0.701, 0.89)$ | $W = (-0.82, 0.7)$ |
| $O_1 = (-0.7, 0.72)$ | $W_1 = (-0.848, 0.848)$ |
| $P_1 = (-0.72, 0.7)$ | |
| $Q_1 = (-0.701, 0.71)$ | |
| $R = (-0.9, 0.9)$ | |

Letra J

| |
|---|
| $E_2 = (-0.06, -0.2)$ |
| $F_2 = (0.06, -0.2)$ |
| $G_2 = (-0.06, -0.16)$ |
| $H_2 = (-0.02, -0.16)$ |
| $Z = (0.06, 0.2)$ |
| $Z_1 = (-0.02, 0.2)$ |

Letra E

| |
|---|
| $E = (-0.1, 0.2)$ |
| $F = (0.1, 0.2)$ |
| $G = (-0.1, -0.2)$ |
| $H = (0.1, -0.2)$ |
| $I = (0.1, 0.148)$ |
| $J = (-0.048, 0.148)$ |
| $K = (-0.02, 0.14)$ |
| $L = (-0.02, 0.04)$ |
| $M = (0.02, 0.04)$ |
| $N = (0.02, -0.012)$ |
| $O = (-0.02, -0.012)$ |
| $P = (-0.02, -0.148)$ |
| $Q = (0.1, -0.148)$ |
| $Z = (-1, 1)$ |

Una vez que contaba con la lista de puntos en un txt, únicamente copiaba y pegaba las coordenadas de los puntos que necesitaba para formar cada triángulo (colocando un comentario antes de los tres puntos que conforman cada triángulo para que fuera más fácil hacer correcciones, en caso de que me equivocara) en base a las listas también creadas por GeoGebra que se muestran a continuación.

Letra R

| | |
|---|---|
| t12 = Polygon(T, R, $W_1$) | t24 = Polygon($U_1$, $S_1$, $Q_1$) |
| t13 = Polygon($W_1$, R, $J_1$) | t25 = Polygon($U_1$, $R_1$, $S_1$) |
| t14 = Polygon($W_1$, $I_1$, $J_1$) | t26 = Polygon($U_1$, $P_1$, $R_1$) |
| t15 = Polygon($I_1$, $J_1$, $T_1$) | t27 = Polygon($U_1$, $G_1$, $P_1$) |
| t16 = Polygon($T_1$, $J_1$, $K_1$) | t28 = Polygon($H_1$, $G_1$, $U_1$) |
| t17 = Polygon($T_1$, $L_1$, $K_1$) | t29 = Polygon($H_1$, $E_1$, $G_1$) |
| t18 = Polygon($T_1$, $L_1$, $M_1$) | t30 = Polygon($H_1$, W, $E_1$) |
| t19 = Polygon($T_1$, $M_1$, $N_1$) | t31 = Polygon($W_1$, $H_1$, T) |
| t20 = Polygon($T_1$, $N_1$, S) | t32 = Polygon($H_1$, V, T) |
| t21 = Polygon($T_1$, S, $O_1$) | t33 = Polygon($E_1$, $F_1$, U) |
| t22 = Polygon($U_1$, $T_1$, $O_1$) | t34 = Polygon($E_1$, $G_1$, U) |
| t23 = Polygon($U_1$, $Q_1$, $O_1$) | t35 = Polygon($W_1$, $V_1$, $H_1$) |

Letra J

| |
|---|
| t36 = Polygon($Z_1$, $F_2$, Z) |
| t37 = Polygon($H_2$, $Z_1$, $F_2$) |
| t38 = Polygon($E_2$, $H_2$, $F_2$) |
| t39 = Polygon($G_2$, $E_2$, $H_2$) |

Letra E

| |
|---|
| t1 = Polygon(G, E, J) |
| t2 = Polygon(E, F, J) |
| t3 = Polygon(J, F, I) |
| t4 = Polygon(J, L, O) |
| t5 = Polygon(J, O, G) |
| t6 = Polygon(L, N, O) |
| t7 = Polygon(L, M, N) |
| t8 = Polygon(O, P, G) |
| t9 = Polygon(G, P, Q) |
| t10 = Polygon(G, H, Q) |
| t11 = Polygon(J, K, L) |

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

- Me resultó bastante complicado realizar el código de los puntos que voy a utilizar para trazar los triángulos que requiero para formar las letras sin tener alguna referencia, por lo tanto, opté por hacer uso de GeoGebra. En esta página pude colocar los puntos cómodamente sobre el plano cartesiano y obtener las coordenadas redondeadas a 3 decimales para tener una buena precisión.
- Al tratar de hacer el dibujo de la primera letra (R), no había comprendido correctamente cómo es que se muestran los triángulos dibujados en la ventana. Creía que si el tamaño del dibujo era bastante pequeño, el programa hacía una especie de “zoom” para ajustar el área de la ventana a la figura. Esto derivó en que mi primer dibujo se veía bastante pequeño y en el centro, en lugar de la esquina superior izquierda, como pensaba que sucedería. Después de esto, comprendí que la ventana es como un plano que abarca de -1.0f a 1.0f en ambos ejes. Para corregirlo, únicamente ajusté mi dibujo para obtener las nuevas coordenadas.

- Cuando estaba intentando hacer que el color de fondo cambiara cada 2 segundos, ya tenía resuelta la aleatoriedad del color pero traté de ocupar un algoritmo parecido al que usé para el ejercicio de clase; este consistía en ocupar el módulo y crear dos estados (0 y 1). Mi idea era que cada que entrara en el estado 0 cambiara el color, pero lo que sucedió fue que se mantenía en el estado 0 durante un segundo completo y todo ese periodo de tiempo cambiaba de color cada ciclo. Para solucionarlo, creé una variable float (*ultimoCambio*) para guardar el tiempo en el que se realizó el último cambio de color del fondo. Entonces, cada que entra al ciclo, resta el tiempo actual menos el tiempo en el que se hizo el último cambio; si han pasado más de dos segundos, entonces hace un nuevo cambio de color y actualiza el valor de *ultimoCambio*.
- Justo en el momento de hacer la comparación para saber si ya han pasado 2 segundos a partir del último cambio, me encontré con el último problema: Si ponía la condición *tiempo - ultimoCambio == 2.0f*, nunca hacía el cambio de color. En cambio, si probé colocar *tiempo - ultimoCambio >= 2.0f, y* entonces sí funcionaba. Mi lógica fue que no es posible o es muy poco probable que la comparación se haga exactamente en el momento en el que el resultado de la comparación sea 2.0f, entonces hay que “asegurar” indicando que aunque ya hayan pasado un poco más de 2 segundos, de igual forma se realice el cambio de color.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

   No siento que los ejercicios hayan tenido mayor complicación. Una vez que encontré las herramientas adecuadas para llevarlas a cabo, la parte de la programación pude hacerla usando librerías y funciones que aprendí en asignaturas previas, como son #include <cstdlib> #include <ctime> para obtener los valores aleatorios para cada cambio de color.

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

   Como mencioné en los comentarios de la primera actividad, no consideré a tiempo configurar y llevar el control de versiones para poder explicar y mostrar con más detalle el desarrollo y resultados de cada actividad. Esto es algo que corregiré sin duda a partir de la siguiente práctica.

c. Conclusión

   Esta práctica fue de bastante ayuda para reafirmar los conocimientos adquiridos en la sesión de laboratorio. Fue una gran introducción al trabajo que vamos a estar realizando a lo largo del semestre que me permitió

también familiarizarme con el entorno de Visual Basic, la configuración del proyecto, la estructura básica del código main para trabajar con OpenGL e identificar dónde se aplican los comandos y conceptos que investigué en el previo.

Los errores que cometí, principalmente la confusión en cuanto al comportamiento del "sistema de coordenadas" de la ventana del output, fueron muy beneficiosos para mi aprendizaje. Me parece que logré comprender correctamente cómo es que funcionan y en qué parte se aplican los comandos más básicos para realizar dibujos simples haciendo uso de OpenGL.

Por otro lado, me causa curiosidad qué herramientas y/o funciones veremos más adelante para poder realizar dibujos más complejos en 2D y 3D ya que, tomando en cuenta que los dibujos que realicé son bastante sencillos a la vista, debo admitir que hacerlos fue más laborioso de lo que pensaba, principalmente las partes redondeadas. Esto puede notarse con la letra R, esta cuenta con dos curvas que no tienen siquiera un gran detalle o definición y aún así hay una diferencia notable en la cantidad de triángulos requeridas para dibujarla en comparación con las otras dos letras.

4.- Bibliografía en formato APA


- Vectorium. (s. f.). Tipografía negrita serif interior fuente mayúscula condensada alfabeto latino geométrico conjunto letras mayúsculas minúsculas serif estrecho letras blancas vectores negro [Imagen vectorial]. Freepik. https://www.freepik.es/vector-premium/tipografia-negrita-serif-interior-fuente-mayuscula-condensada-alfabeto-latino-geometrico-conjunto-letras-mayusculas-minusculas-serif-estrecho-letras-blancas-vectores-negro_6858247.htm
- GeoGebra. (2024). GeoGebra Geometría (Versión en línea) [Software de computación]. https://www.geogebra.org/geometry