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: Romero Dominguez Ricardo Damian

Nº de Cuenta: 319094493

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 2/15/2025

CALIFICACIÓN: _____

Índice

1. EJECUCIÓN DE EJERCICIOS

1.1. RGB Aleatorio

Para la resolución de este problema, simplemente se modificó el programa para que en el arreglo **color**, el cual tiene tres componentes que corresponden a los colores RGB respectivamente, de modo que a cada una de las componentes del arreglo, se le modifica su valor con base en la función **rand()**, la cual con base en una semilla dependiente del reloj interno de la computadora, a esta función se le aplicaron rangos arbitrarios para cada componente de modo que se vuelve ligeramente más aleatorio el color.

Código 1. RGB Aleatorio.

```
color[0] = 1.0f/(rand() % 8 + 1);
color[1] = 1.0f/(rand() % 10 + 1);
color[2] = 1.0f/(rand() % 3 + 1);

```

1.2. Periodicidad de 2 segundos

Para definir un cambio de color la pantalla del programa, simplemente se uso la funcion **Sleep()** de la librería **Windows.h**

Código 2. RGB Aleatorio.

```
Sleep(2000);

```

1.3. Letras: RDA

En cuanto a las letras refiere, primero se realizó una planificación de la distribución de las letras tomando en cuenta un rango y un dominio de $[-1, 1]$, donde primero se dividió el plano en segmentos de 0.1 [pts].

Posteriormente, se planeo usar arcos (cuartos de circulo) para algunas de las curvas de las letras, mientras que para las partes rectas solo se emplearon dos triángulos.

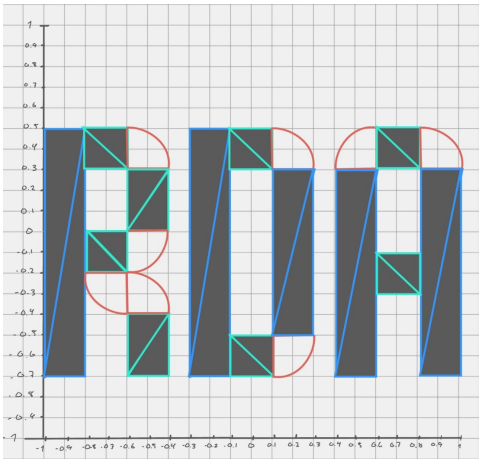

Figura 1. Planeación inicial

Para los rectángulos y cuadrados, solo se incluyeron sus coordenadas según la planificación inicial, sin embargo, para los arcos se decidió apilar triángulos uno junto al otro tomando como pivote el centro de la circunferencia, tal como en la siguiente imagen.

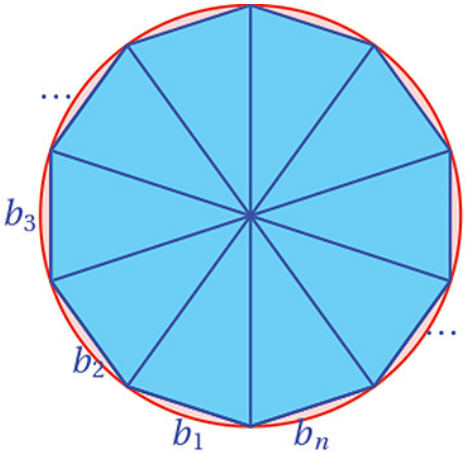

Figura 2. Círculo formado por triángulos

Para una generación decente de arco, se decidió crear a parte un programa en C++ tomando el fundamento de que un circulo es un par ordenado conformado de senos (eje Y) y cosenos (eje X), por lo que primeramente solo se generaron los puntos correspondientes al perimetro de la circunferencia y el centro de la circunferencia.

Asimismo, como requerimiento del programa, se define que como entrada se le debe de dar la posición inicial (centro del círculo), la posición final (flecha del círculo) y la resolución de la circunferencia (número de triángulos que se generan).

De este modo, mediante un ciclo for se almacenan los puntos necesarios para la formación del arco con

el uso de senos y cosenos.

Para poder modificar tanto la posición como el tamaño del arco, simplemente se multiplica cada punto calculado por un factor que data de la diferencia entre los puntos iniciales y finales, al igual que sumarles los puntos iniciales para desplazar el arco.

Ulteriormente se acomodaron los puntos generados para acomodar las coordenadas del perímetro del arco junto con el centro de la circunferencia y de esta manera ligar dichos puntos y formar los triángulos correspondientes.

Código 3. RGB Aleatorio.

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <vector>

using namespace std;

int main() {
    float xIni, yIni, xFin, yFin, resolucion;

    vector<float> puntos, vertices, ancla;

    cout << "Ingresa la coordenada x inicial: ";
    cin >> xIni;
    cout << "Ingresa la coordenada y inicial: ";
    cin >> yIni;
    cout << "Ingresa la coordenada x final: ";
    cin >> xFin;
    cout << "Ingresa la coordenada y final: ";
    cin >> yFin;
    cout << "Ingresa el numero de triangulos: ";
    cin >> resolucion;

    ancla.push_back(0.0f); // z
    ancla.push_back(yIni); // y
    ancla.push_back(xIni); // x

    float saltos = 90.0f/(resolucion),
          factorSemejanza_x = xFin-xIni,
          factorSemejanza_y = yFin-yIni;


    for (int i = 0; i <= resolucion ; i += 1) {
        puntos.push_back(0.0f);
        puntos.push_back(factorSemejanza_y*abs(sin((i*saltos*M_PI)/(180) )) + yIni);
        puntos.push_back(factorSemejanza_x*abs(cos((i*saltos*M_PI)/(180) )) + xIni);
    }

    for (int i  = puntos.size() - 1 ; i >= 3 ; i-=3) {
        vertices.push_back(puntos.at(i));
        puntos.pop_back();
        vertices.push_back(puntos.at(i - 1));
        puntos.pop_back();
        vertices.push_back(puntos.at(i - 2));
        puntos.pop_back();

        vertices.push_back(puntos.at(i - 3));
        vertices.push_back(puntos.at(i - 4));
        vertices.push_back(puntos.at(i - 5));

        vertices.push_back(ancla.at(2));
        vertices.push_back(ancla.at(1));
        vertices.push_back(ancla.at(0));

    }

    // x.xxxxf, x.xxxxf, 0.0f,
    cout << fixed << setprecision(8);
    cout << "\nCoordenadas generadas:\n";
    for (size_t i = 0; i < vertices.size(); i += 3) {
        cout << vertices[i] << "f, " << vertices[i+1] << "f, 0.0f," << endl;
```

```
62      }
63      cout << "\nNumero de vertices
    generados: " << vertices.size()
    / 3 << endl;
64
65      return 0;
66  }
67
68
```

Finalmente solo se imprimen los datos para poder copiarlos dentro del main de esta práctica.

Figura 3. Resultados de ejecución

2. PROBLEMAS PRESENTADOS

2.1. Implementación de código generador

Uno de los problemas significativamente importantes sobre la finalización de la práctica refiere a la implementación del código que genera los arcos dentro del main que emplea OpenGL, esto debido a que los vectores y los arreglos de tipo **GLfloat** no son tipos de datos compatibles, y de igual manera, dado que **GLfloat** no es un tipo de dato nativo de C++, es complicado trabajar con el.

3. CONCLUSIÓN

Dada la consumación de actividades que integran esta primer práctica de computación gráfica e interacción humano - computadora, en la cual se pueden apreciar el aprendizaje de fundamentos de programación en OpenGL, donde si bien la plataforma donde se realiza dicha programación (C++) no es totalmente desconocida, el uso de estos primeros códigos a simple vista parecen un sistema complejo debido a las múltiples funciones que contiene, y si bien dentro de la clase se explicó el código, el entendimiento recae en los detalles, por lo que espero que a lo largo del curso todas estas funciones propias de OpenGL puedan ser abordaas y aplicadas con más detalle.

En cuánto a la complejidad del ejercicio, si bien no era una tarea muy complicada debido a que solo era realizar un mapeo de un par de puntos en un espacio bidimensional, la complejidad que este ejercicio conlleva es más orientada tomando en cuenta que es un primer acercamiento a este enfoque de programación, por lo que objetivamente no fue complicado, pero si requiere un proceso de análisis.

Referencias

[C++] Funciones matemáticas. (s.f.). https://www.programacionfacil.org/cursos/c++/capitulo-19-funciones-matematicas.html#google_vignette

C++ Variables de tipo vector | Aprende Programación Competitiva. (s.f.). https://aprende.olimpiada-informatica.org/cpp-vector

Cast y conversión de tipos en C++. (2017, 11 noviembre). Trucos Informáticos. https://trucosinformaticos.wordpress.com/2017/11/11/cast-y-conversion-de-tipos-en-c/