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 03

NOMBRE COMPLETO: Casillo Martinez Diego Leonardo

N.º de Cuenta: 319041538

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 04/09/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Desarrollo

Ejercicio 1: El objetivo de este ejercicio es generar una pirámide Rubik (Pyraminx) con 9 pequeñas pirámides en cada cara. Cada una de las caras de la Pyraminx debe mostrar un color diferente y las separaciones entre las pirámides deben ser visibles, marcadas por líneas oscuras que permiten diferenciar cada pirámide pequeña.

Para la realización de este ejercicio, se aprovecharon algunas líneas de código de la práctica anterior (Práctica 2), ya que se requería minimizar el número de pirámides generadas. Para lograr esto, se tuvo en cuenta lo siguiente:

Existen 11 pirámides que comparten vértices en la estructura de la Pyraminx. Para aprovechar este aspecto, se debe permitir que una pirámide tenga caras de diferentes colores, lo que facilita su integración en una pirámide base de color negro. Una vez hecho esto, solo será necesario generar pirámides acostadas para cada cara, con las cuales se rellenarán los huecos faltantes.

Esto se realizó alterando el shader color.vert para poder utilizar la cámara. Una vez hecho esto, generamos una función que crea una pirámide a la cual le asignamos valores RGB, de manera que cada cara de la pirámide tuviera un color diferente.

El mismo procedimiento se aplicó a otras dos estructuras, considerando los casos de una pirámide invertida y con la punta orientada hacia diferentes caras. Esto se debió a las complicaciones encontradas al intentar aplicar rotaciones en varios ejes, lo cual derivó en la creación de las siguientes líneas de código.

Para la pirámide de colores:

```
void CrearPiramideColores()
{
    GLfloat vertices_piramide[] = {
        // Posiciones          // Colores (RGB)

        // Cara 1 (Color Rojo)
        0.0f,  0.75f, 0.0f,    1.0f, 0.0f, 0.0f, // Punta más alta (Rojo)
        -0.5f, -0.3f, 0.5f,    1.0f, 0.0f, 0.0f, // Base (Rojo)
        0.5f, -0.3f, 0.5f,     1.0f, 0.0f, 0.0f, // Base (Rojo)

        // Cara 2 (Color Verde)
        0.0f,  0.75f, 0.0f,    0.0f, 1.0f, 0.0f, // Punta más alta (Verde)
        0.5f, -0.3f, 0.5f,     0.0f, 1.0f, 0.0f, // Base (Verde)
        0.0f, -0.3f, -0.7f,    0.0f, 1.0f, 0.0f, // Base (Verde)

        // Cara 3 (Color Azul)
        0.0f,  0.75f, 0.0f,    0.0f, 0.0f, 1.0f, // Punta más alta (Azul)
        0.0f, -0.3f, -0.7f,    0.0f, 0.0f, 1.0f, // Base (Azul)
        -0.5f, -0.3f, 0.5f,    0.0f, 0.0f, 1.0f, // Base (Azul)

        // Cara 4 (Base, Color Amarillo)
        0.5f, -0.3f, 0.5f,     1.0f, 1.0f, 0.0f, // Base (Amarillo)
        -0.5f, -0.3f, 0.5f,    1.0f, 1.0f, 0.0f, // Base (Amarillo)
        0.0f, -0.3f, -0.7f,    1.0f, 1.0f, 0.0f  // Base (Amarillo)
    };

    MeshColor* piramideColor = new MeshColor();
    piramideColor->CreateMeshColor(vertices_piramide, 72);
    meshColorList.push_back(piramideColor);
}
```

Para las pirámides invertidas:

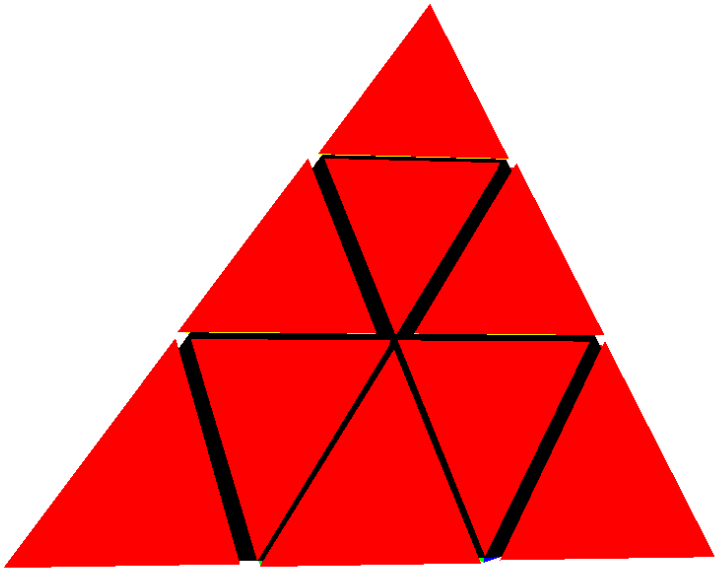
```
void CrearPiramideColoresInvertida()
{
    GLfloat vertices_piramide[] = {
        // Posiciones         // Colores (RGB)

        // Cara 1 (Color Rojo)
        -0.5f,  -0.75f, -0.2f,   1.0f, 0.0f, 0.0f, // Punta más baja (Rojo), inclinada hacia la cara azul
        -0.5f, 0.3f, 0.5f,      1.0f, 0.0f, 0.0f, // Base (Rojo)
        0.5f, 0.3f, 0.5f,       1.0f, 0.0f, 0.0f, // Base (Rojo)

        // Cara 2 (Color Verde)
        -0.5f,  -0.75f, -0.2f,   0.0f, 1.0f, 0.0f, // Punta más baja (Verde), inclinada hacia la cara azul
        0.5f, 0.3f, 0.5f,       0.0f, 1.0f, 0.0f, // Base (Verde)
        0.0f, 0.3f, -0.7f,      0.0f, 1.0f, 0.0f, // Base (Verde)

        // Cara 3 (Color Azul)
        -0.5f,  -0.75f, -0.2f,   0.0f, 0.0f, 1.0f, // Punta más baja (Azul), inclinada hacia la cara azul
        0.0f, 0.3f, -0.7f,      0.0f, 0.0f, 1.0f, // Base (Azul)
        -0.5f, 0.3f, 0.5f,      0.0f, 0.0f, 1.0f, // Base (Azul)

        // Cara 4 (Base, Color Amarillo)
        0.5f, 0.3f, 0.5f,       1.0f, 1.0f, 0.0f, // Base (Amarillo)
        -0.5f, 0.3f, 0.5f,      1.0f, 1.0f, 0.0f, // Base (Amarillo)
        0.0f, 0.3f, -0.7f,      1.0f, 1.0f, 0.0f  // Base (Amarillo)
    };

    MeshColor* piramideColor = new MeshColor();
    piramideColor->CreateMeshColor(vertices_piramide, 72);
    meshColorList.push_back(piramideColor);
}
```

```
void CrearPiramideColoresInvertida1()
{
    GLfloat vertices_piramide[] = {
        // Posiciones         // Colores (RGB)

        // Cara 1 (Color Rojo)
        0.5f,  -0.75f, -0.2f,   1.0f, 0.0f, 0.0f, // Punta centrada en Z
        -0.5f, 0.3f, 0.5f,      1.0f, 0.0f, 0.0f, // Base (Rojo)
        0.5f, 0.3f, 0.5f,       1.0f, 0.0f, 0.0f, // Base (Rojo)

        // Cara 2 (Color Verde)
        0.5f,  -0.75f, -0.2f,   0.0f, 1.0f, 0.0f, // Punta centrada en Z
        0.5f, 0.3f, 0.5f,       0.0f, 1.0f, 0.0f, // Base (Verde)
        0.0f, 0.3f, -0.7f,      0.0f, 1.0f, 0.0f, // Base (Verde)

        // Cara 3 (Color Azul)
        0.5f,  -0.75f, -0.2f,   0.0f, 0.0f, 1.0f, // Punta centrada en Z
        0.0f, 0.3f, -0.7f,      0.0f, 0.0f, 1.0f, // Base (Azul)
        -0.5f, 0.3f, 0.5f,      0.0f, 0.0f, 1.0f, // Base (Azul)

        // Cara 4 (Base, Color Amarillo)
        0.5f, 0.3f, 0.5f,       1.0f, 1.0f, 0.0f, // Base (Amarillo)
        -0.5f, 0.3f, 0.5f,      1.0f, 1.0f, 0.0f, // Base (Amarillo)
        0.0f, 0.3f, -0.7f,      1.0f, 1.0f, 0.0f  // Base (Amarillo)
    };

    MeshColor* piramideColor = new MeshColor();
    piramideColor->CreateMeshColor(vertices_piramide, 72);
```

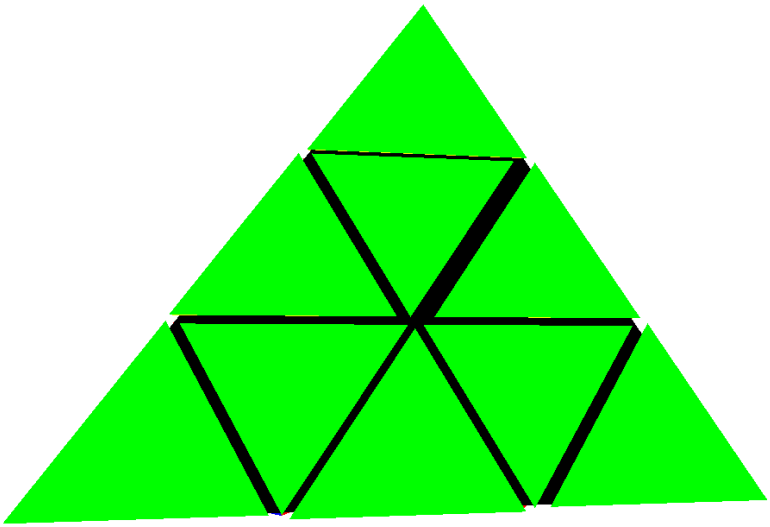
Nota: Son dos pirámides invertidas, ya que una está orientada hacia el lado azul de la cara de la Pyraminx y la otra hacia el lado verde. Esto se debe a que las rotaciones en diferentes ejes se comportaban de maneras inusuales, lo que generaba muchas complicaciones.

Gracias a ello la ejecución funciona de la siguiente forma:

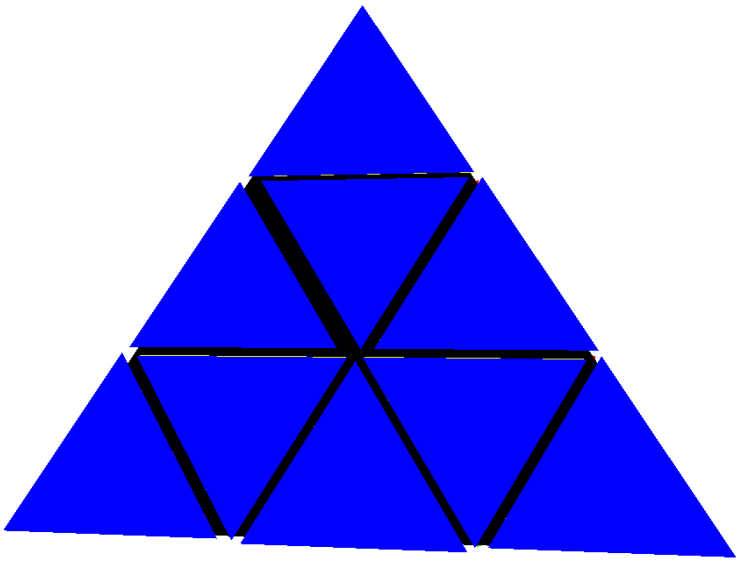
Para la cara principal (en este caso roja)



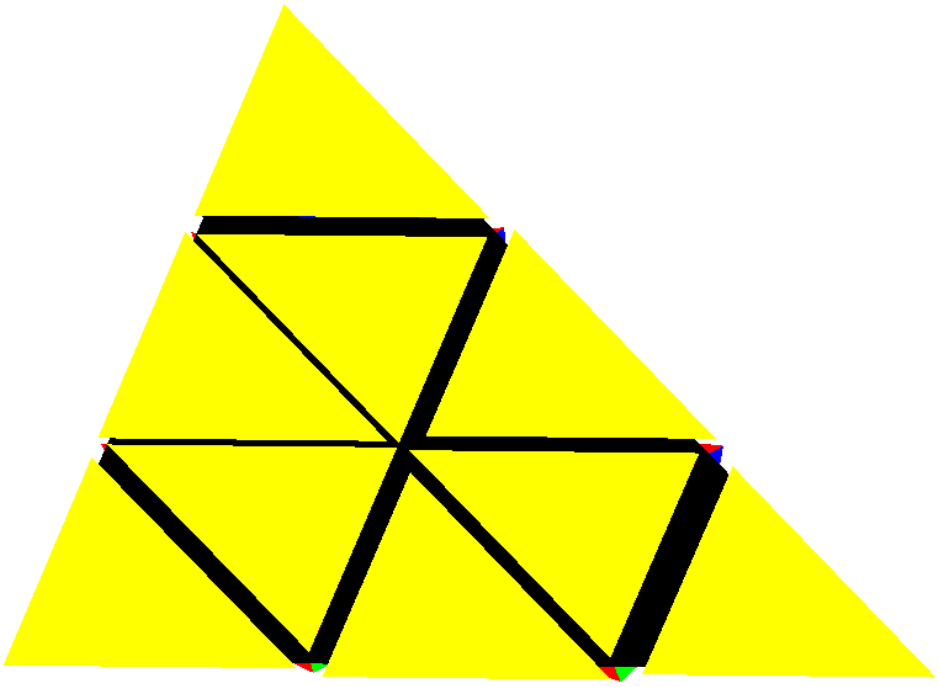
Para la cara lateral verde



Para la cara lateral azul:

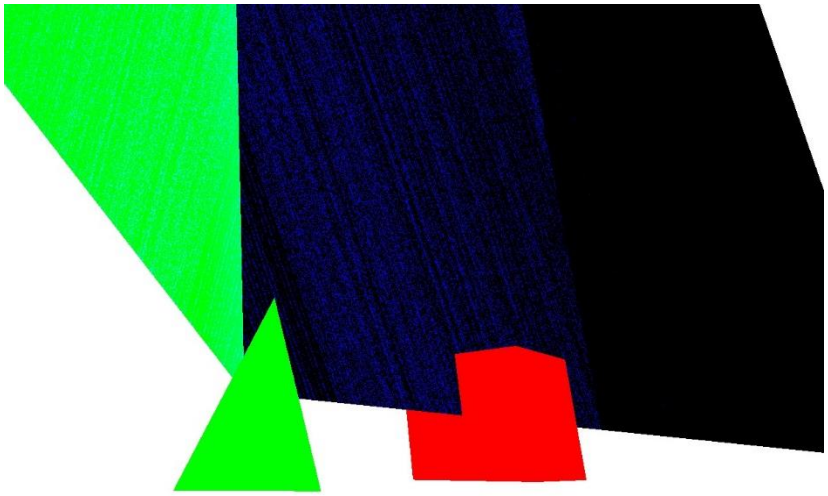


Para la tapa amarilla:



2.- problemas que surgieron:

Para esta práctica surgieron varios problemas. Uno de ellos ocurrió al intentar generar la pirámide de colores, cuando se presentó un error con el shader debido a que no se enviaba el número correcto de vértices. Esto causaba que la figura se visualizara de la siguiente manera:



Otro gran problema surgió con las rotaciones de las pirámides laterales, ya que estas se comportaban de manera extraña debido a que la pirámide creada tenía la punta en el centro. Esto generó varios inconvenientes al intentar acoplar las pirámides de manera invertida en los laterales. A pesar de aplicar rotaciones, no se obtenía el resultado deseado.

3.- Conclusión:

En esta práctica se presentaron diversas dificultades que la hicieron complicada, pero a la vez entretenida. Considero que fue un desafío que me permitió comprender mejor el funcionamiento de los shaders .frag y .vert, así como el uso de la cámara. Tanto la explicación del programa como la teoría resultaron fundamentales para lograr este entendimiento.

4.- Bibliografía:

Tutorial 3 : Matrices. (s. f.). https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/