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 02

NOMBRE COMPLETO: Barragán Pilar Diana

Nº de Cuenta: 318147981

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 24 de agosto de 2024

CALIFICACIÓN: _____

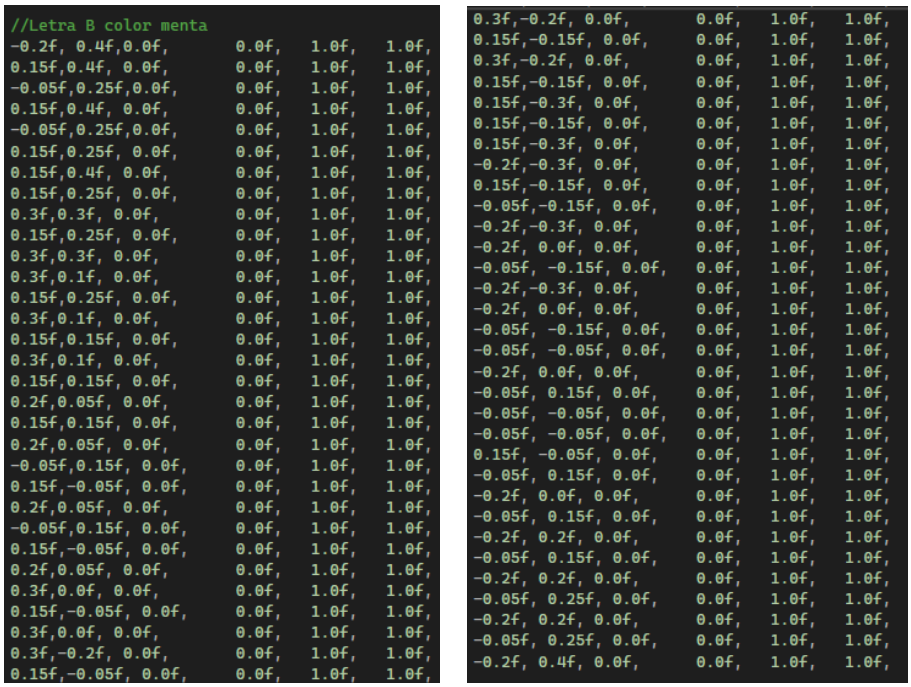
Práctica 2: Proyecciones y puertos de vista. Transformaciones Geométricas

1. Dibujar las iniciales de sus nombres, cada letra de un color diferente.

Para llevar a cabo este ejercicio utilice partes del código visto en la practica número uno, puesto que todo lo referente a las letras iniciales de mi nombre así como sus respectivos vértices ya lo había establecido previamente, por lo que para editar el código de esta práctica número dos, en la función llamada `CrearLetrasyFiguras()`, donde también se encuentra la matriz que como se vio antes los primeros tres valores corresponden a las coordenadas en “x”, “y” y “z” respectivamente y lo diferente son las siguientes tres coordenadas que corresponden al color en RGB. Modificando los valores de los últimos tres apartados fue que cambie el color de cada letra, en la letra D asigne el color amarillo que en RGB sería 1, 1, 0, para la letra B utilice el color menta con el código 0, 1, 1 y finalmente para la letra P utilicé el color rosa con el valor de 1, 0, 1.

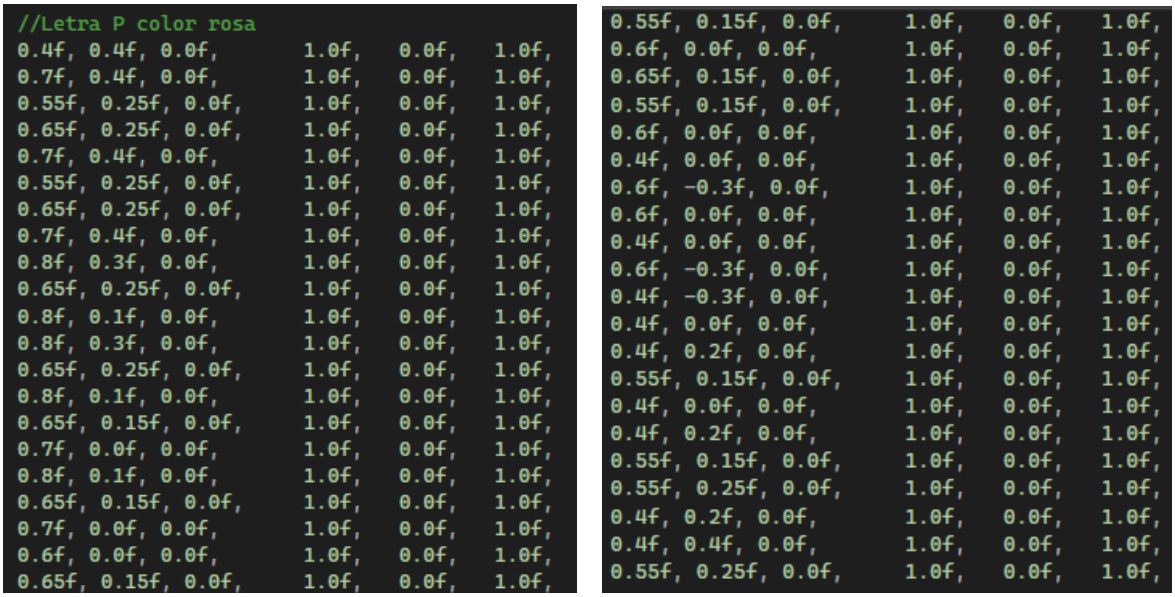
```
GLfloat vertices_letras[] = {

    //Letra D color amarillo
    //X    Y     Z          R       G       B
    -0.9f, 0.4f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.7f,0.2f, 0.0f,       1.0f,   1.0f,   0.0f,
    -0.5f,0.4f,0.0f,        1.0f,   1.0f,   0.0f,
    -0.7f,0.2f, 0.0f,       1.0f,   1.0f,   0.0f,
    -0.5f,0.4f,0.0f,        1.0f,   1.0f,   0.0f,
    -0.55f,0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.5f,0.4f,0.0f,        1.0f,   1.0f,   0.0f,
    -0.55f,0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.3f, 0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.55f,0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.3f, 0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.45f, 0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.3f, 0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.45f, 0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.45f, 0.0f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.3f, 0.2f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.45f, 0.0f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.3f, -0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.45f, 0.0f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.3f, -0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.5f, -0.3f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.45f, 0.0f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.5f, -0.3f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.55f, -0.1f,0.0f,     1.0f,   1.0f,   0.0f,
    -0.55f, -0.1f,0.0f,     1.0f,   1.0f,   0.0f,
    -0.5f, -0.3f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.9f, -0.3f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.55f, -0.1f,0.0f,     1.0f,   1.0f,   0.0f,
    -0.9f, -0.3f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.7f, -0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.9f, -0.3f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.7f, -0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.9f, 0.0f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.7f,0.2f, 0.0f,       1.0f,   1.0f,   0.0f,
    -0.7f, -0.1f,0.0f,      1.0f,   1.0f,   0.0f,
    -0.9f, 0.0f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.9f, 0.0f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.9f, 0.4f,0.0f,       1.0f,   1.0f,   0.0f,
    -0.7f,0.2f, 0.0f,       1.0f,   1.0f,   0.0f,
```

Figura 1. Matriz para las letras. Letra D con el RGB amarillo.



```
//Letra B color menta
-0.2f, 0.4f,0.0f,       0.0f,   1.0f,   1.0f,
0.15f,0.4f, 0.0f,       0.0f,   1.0f,   1.0f,
-0.05f,0.25f,0.0f,      0.0f,   1.0f,   1.0f,
0.15f,0.4f, 0.0f,       0.0f,   1.0f,   1.0f,
-0.05f,0.25f,0.0f,      0.0f,   1.0f,   1.0f,
0.15f,0.25f, 0.0f,      0.0f,   1.0f,   1.0f,
0.15f,0.4f, 0.0f,       0.0f,   1.0f,   1.0f,
0.15f,0.25f, 0.0f,      0.0f,   1.0f,   1.0f,
0.3f,0.3f, 0.0f,        0.0f,   1.0f,   1.0f,
0.15f,0.25f, 0.0f,      0.0f,   1.0f,   1.0f,
0.3f,0.3f, 0.0f,        0.0f,   1.0f,   1.0f,
0.3f,0.1f, 0.0f,        0.0f,   1.0f,   1.0f,
0.15f,0.25f, 0.0f,      0.0f,   1.0f,   1.0f,
0.3f,0.1f, 0.0f,        0.0f,   1.0f,   1.0f,
0.15f,0.15f, 0.0f,      0.0f,   1.0f,   1.0f,
0.3f,0.1f, 0.0f,        0.0f,   1.0f,   1.0f,
0.15f,0.15f, 0.0f,      0.0f,   1.0f,   1.0f,
0.2f,0.05f, 0.0f,       0.0f,   1.0f,   1.0f,
0.15f,0.15f, 0.0f,      0.0f,   1.0f,   1.0f,
0.2f,0.05f, 0.0f,       0.0f,   1.0f,   1.0f,
-0.05f,0.15f, 0.0f,     0.0f,   1.0f,   1.0f,
0.15f,-0.05f, 0.0f,     0.0f,   1.0f,   1.0f,
0.2f,0.05f, 0.0f,       0.0f,   1.0f,   1.0f,
-0.05f,0.15f, 0.0f,     0.0f,   1.0f,   1.0f,
0.15f,-0.05f, 0.0f,     0.0f,   1.0f,   1.0f,
0.2f,0.05f, 0.0f,       0.0f,   1.0f,   1.0f,
0.3f,0.0f, 0.0f,        0.0f,   1.0f,   1.0f,
0.15f,-0.05f, 0.0f,     0.0f,   1.0f,   1.0f,
0.3f,0.0f, 0.0f,        0.0f,   1.0f,   1.0f,
0.3f,-0.2f, 0.0f,       0.0f,   1.0f,   1.0f,
0.15f,-0.05f, 0.0f,     0.0f,   1.0f,   1.0f,
0.3f,-0.2f, 0.0f,       0.0f,   1.0f,   1.0f,
0.15f,-0.15f, 0.0f,     0.0f,   1.0f,   1.0f,
0.3f,-0.2f, 0.0f,       0.0f,   1.0f,   1.0f,
0.15f,-0.15f, 0.0f,     0.0f,   1.0f,   1.0f,
0.15f,-0.3f, 0.0f,      0.0f,   1.0f,   1.0f,
0.15f,-0.15f, 0.0f,     0.0f,   1.0f,   1.0f,
0.15f,-0.3f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.2f,-0.3f, 0.0f,      0.0f,   1.0f,   1.0f,
0.15f,-0.15f, 0.0f,     0.0f,   1.0f,   1.0f,
-0.05f,-0.15f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.2f,-0.3f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.2f, 0.0f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, -0.15f, 0.0f,   0.0f,   1.0f,   1.0f,
-0.2f,-0.3f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.2f, 0.0f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, -0.15f, 0.0f,   0.0f,   1.0f,   1.0f,
-0.05f, -0.05f, 0.0f,   0.0f,   1.0f,   1.0f,
-0.2f, 0.0f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, 0.15f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.05f, -0.05f, 0.0f,   0.0f,   1.0f,   1.0f,
-0.05f, -0.05f, 0.0f,   0.0f,   1.0f,   1.0f,
0.15f, -0.05f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.05f, 0.15f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.2f, 0.0f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, 0.15f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.2f, 0.2f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, 0.15f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.2f, 0.2f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, 0.25f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.2f, 0.2f, 0.0f,      0.0f,   1.0f,   1.0f,
-0.05f, 0.25f, 0.0f,    0.0f,   1.0f,   1.0f,
-0.2f, 0.4f, 0.0f,      0.0f,   1.0f,   1.0f,
```

Figura 2. Matriz para las letras. Letra B con el RGB menta.



```
//Letra P color rosa
0.4f, 0.4f, 0.0f,       1.0f,   0.0f,   1.0f,
0.7f, 0.4f, 0.0f,       1.0f,   0.0f,   1.0f,
0.55f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.65f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.7f, 0.4f, 0.0f,       1.0f,   0.0f,   1.0f,
0.55f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.65f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.7f, 0.4f, 0.0f,       1.0f,   0.0f,   1.0f,
0.8f, 0.3f, 0.0f,       1.0f,   0.0f,   1.0f,
0.65f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.8f, 0.1f, 0.0f,       1.0f,   0.0f,   1.0f,
0.8f, 0.3f, 0.0f,       1.0f,   0.0f,   1.0f,
0.65f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.8f, 0.1f, 0.0f,       1.0f,   0.0f,   1.0f,
0.65f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.7f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.8f, 0.1f, 0.0f,       1.0f,   0.0f,   1.0f,
0.65f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.7f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.6f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.65f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.55f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.6f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.65f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.55f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.6f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.4f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.6f, -0.3f, 0.0f,      1.0f,   0.0f,   1.0f,
0.6f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.4f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.6f, -0.3f, 0.0f,      1.0f,   0.0f,   1.0f,
0.4f, -0.3f, 0.0f,      1.0f,   0.0f,   1.0f,
0.4f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.4f, 0.2f, 0.0f,       1.0f,   0.0f,   1.0f,
0.55f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.4f, 0.0f, 0.0f,       1.0f,   0.0f,   1.0f,
0.4f, 0.2f, 0.0f,       1.0f,   0.0f,   1.0f,
0.55f, 0.15f, 0.0f,     1.0f,   0.0f,   1.0f,
0.55f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
0.4f, 0.2f, 0.0f,       1.0f,   0.0f,   1.0f,
0.4f, 0.4f, 0.0f,       1.0f,   0.0f,   1.0f,
0.55f, 0.25f, 0.0f,     1.0f,   0.0f,   1.0f,
```

Figura 3. Matriz para las letras. Letra P con el RGB rosa.

Finalmente, para que aparezcan adecuadamente las letras cambiamos en la línea de `CreateMeshColor` el número de vértices actualizándolo a los vértices que agregamos junto con el RGB.

```
MeshColor *letras = new MeshColor();
letras->CreateMeshColor(vertices_letras,900);
//Actualizamos el número de vertices para las letras del nombre
meshColorList.push_back(letras);
```

Figura 3. Actualización de los vértices en `CreateMeshColor`.

EJECUCIÓN DEL PROGRAMA:

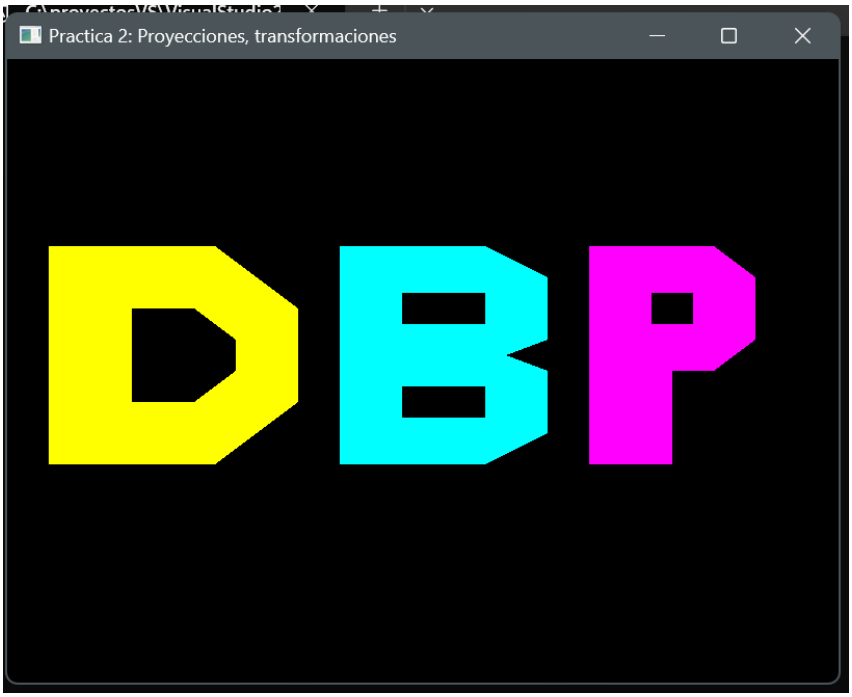


Figura 4. Ejecución de iniciales del nombre con color diferente.

2. **Generar el dibujo de la casa de la clase, pero en lugar de instanciar triángulos y cuadrados será instanciando pirámides y cubos, para esto se requiere crear shaders diferentes de los colores: rojo, verde, azul, café y verde oscuro en lugar de usar el shader con el color clamp.**

En la ejecución de este ejercicio lo primero que realice fue crear los nuevos shaders en la carpeta donde estos se encontraban, nombrándolos de acuerdo al color que correspondían: shaderR (rojo), shaderV (verde), shaderA (azul), shaderC (café) y shaderVO (verde oscuro). Los cambios que realice dentro del código fue simplemente cambiar la línea `vColor=vec4(clamp(pos,0.0f,1.0f),1.0f);` por la línea `vColor = vec4(R, G, B, 1.0f);`

Figura 5. Carpeta con los nuevos shaders.

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    vColor = vec4(1.0f, 0.0f, 0.0f, 1.0f);
}
```

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    vColor = vec4(0.0f, 1.0f, 0.0f, 1.0f);
}
```

Figura 6. Archivo shaderR.vert y shaderV.vert.

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    vColor = vec4(0.0f, 0.0f, 1.0f, 1.0f);
}
```

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    vColor = vec4(0.478f, 0.255f, 0.067f, 1.0f);
}
```

Figura 7. Archivo shaderA.vert y shaderC.vert.

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    vColor = vec4(0.0f, 0.5f, 0.0f, 1.0f);
}
```

Figura 8. Archivo shaderVO.vert.

Una vez realizadas las modificaciones de los archivos de los shaders, para poder utilizarlos en el código primero declare los nuevos shaders junto con los anteriores haciendo uso de la siguiente nomenclatura `static const char* vShaderR = "shaders/nombre_shader.vert"`. Posterior a ello en la función `CreateShaders()` cree los nuevos shaders y agregue estos objetos a la lista de shaders con su respectivo índice el cual es importante recordar puesto que más adelante en el código se hace uso de él.

```
//Creación de los nuevos colores de shaders
Shader* shader3 = new Shader(); //shader rojo indice[2]
shader3->CreateFromFiles(vShaderR, fShader);
shaderList.push_back(*shader3);

Shader* shader4 = new Shader(); //shader verde indice[3]
shader4->CreateFromFiles(vShaderV, fShader);
shaderList.push_back(*shader4);

Shader* shader5 = new Shader(); //shader azul indice[4]
shader5->CreateFromFiles(vShaderA, fShader);
shaderList.push_back(*shader5);

Shader* shader6 = new Shader(); //shader cafe indice[5]
shader6->CreateFromFiles(vShaderC, fShader);
shaderList.push_back(*shader6);

Shader* shader7 = new Shader(); //shader rojo indice[6]
shader7->CreateFromFiles(vShaderVO, fShader);
shaderList.push_back(*shader7);
```

Figura 9. Creación de los nuevos shaders de los colores.

Finalmente, en el main para lograr recrear la figura de la casa primero se inicializo la ventana y su tamaño, luego se llamaron a las funciones para crear el cubo y la pirámide a partir de sus vértices e índices, después dentro del ciclo while activamos y establecemos los valores del shader respectivo en la lista de shaders. Inicializamos la matriz de 4x4 que va a aguardar las transformaciones geométricas la cual es una matriz identidad llamada model a la que le vamos a aplicar las transformaciones de translación para mover la figura a otro punto y rotación para que rote sobre un eje.

En este caso utilice los mismos valores de vector en la función de translación que en el ejercicio con cuadrados y triángulos y agregue la matriz de rotación para poder ver mejor que las figuras eran cubos y pirámides, por ultimo para que se dibuje el cubo o la pirámide ocupe `meshList[0]->RenderMesh();` cambiando el índice dependiendo de si era un cubo o una pirámide.

```
//***PIRAMIDE AZUL***

//Para la piramide azul ocupamos el shader con indice 4
shaderList[4].useShader();
uniformModel = shaderList[4].getModelLocation();
uniformProjection = shaderList[4].getProjectLocation();
angulo += 0.01;
//Inicializar matriz de dimensión 4x4 que servirá como matriz de modelo para almacenar las
// transformaciones geométricas
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.65f, -4.0f));
model = glm::scale(model, glm::vec3(1.3f, 0.6f, 1.3f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh(); //Ya que es la piramide en el mesh list debe ser el indice 0
```

Figura 10. Código para dibujado de la pirámide azul.

```
//***CUBO ROJO***
shaderList[2].useShader();
uniformModel = shaderList[2].getModelLocation();
uniformProjection = shaderList[2].getProjectLocation();
angulo += 0.01;

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.35f, -4.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.4f, 1.0f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh(); //Ya que es el cubo en el mesh list debe ser el indice 1
```

Figura 11. Código para dibujado del cubo rojo.

```
//***PIRAMIDES VERDE OSCURO***
shaderList[6].useShader();
uniformModel = shaderList[6].getModelLocation();
uniformProjection = shaderList[6].getProjectLocation();
angulo += 0.01;

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.8f, -0.55f, -4.0f));
model = glm::scale(model, glm::vec3(0.4f, 0.5f, 0.4f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh(); //Ya que es la piramide en el mesh list debe ser el indice 0

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.8f, -0.55f, -4.0f));
model = glm::scale(model, glm::vec3(0.4f, 0.5f, 0.4f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh(); //Ya que es la piramide en el mesh list debe ser el indice 0
```

Figura 12. Código para dibujado de las pirámides verde oscuro.

```
//***CUBOS CAFÉ***
shaderList[5].useShader();
uniformModel = shaderList[5].getModelLocation();
uniformProjection = shaderList[5].getProjectLocation();
angulo += 0.01;

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.8f, -0.9f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh(); //Ya que es el cubo en el mesh list debe ser el indice 1

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.8f, -0.9f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh(); //Ya que es el cubo en el mesh list debe ser el indice 1
```

Figura 13. Código para dibujado de los cubos café.

```
//***CUBOS VERDES***
shaderList[3].useShader();
uniformModel = shaderList[3].getModelLocation();
uniformProjection = shaderList[3].getProjectLocation();
angulo += 0.01;

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.25f, 0.0f, -2.0f));
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh(); //Ya que es el cubo en el mesh list debe ser el indice 1

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.25f, 0.0f, -2.0f));
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh(); //Ya que es el cubo en el mesh list debe ser el indice 1

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.8f, -2.0f));
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh(); //Ya que es el cubo en el mesh list debe ser el indice 1
```

Figura 14. Código para dibujado de los cubos verde.

EJECUCIÓN DEL PROGRAMA:

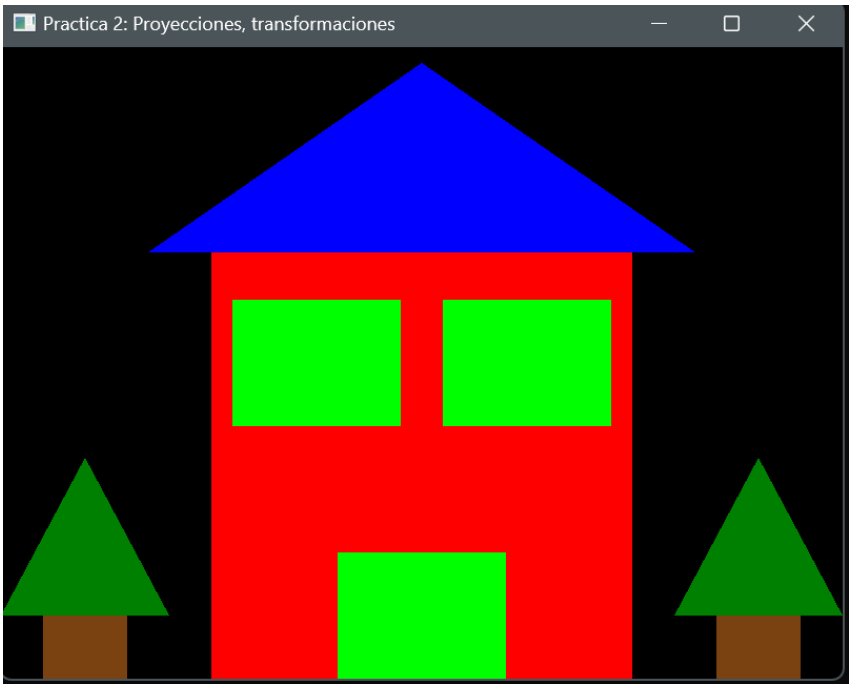


Figura 15. Ejecución código sin rotar las figuras.

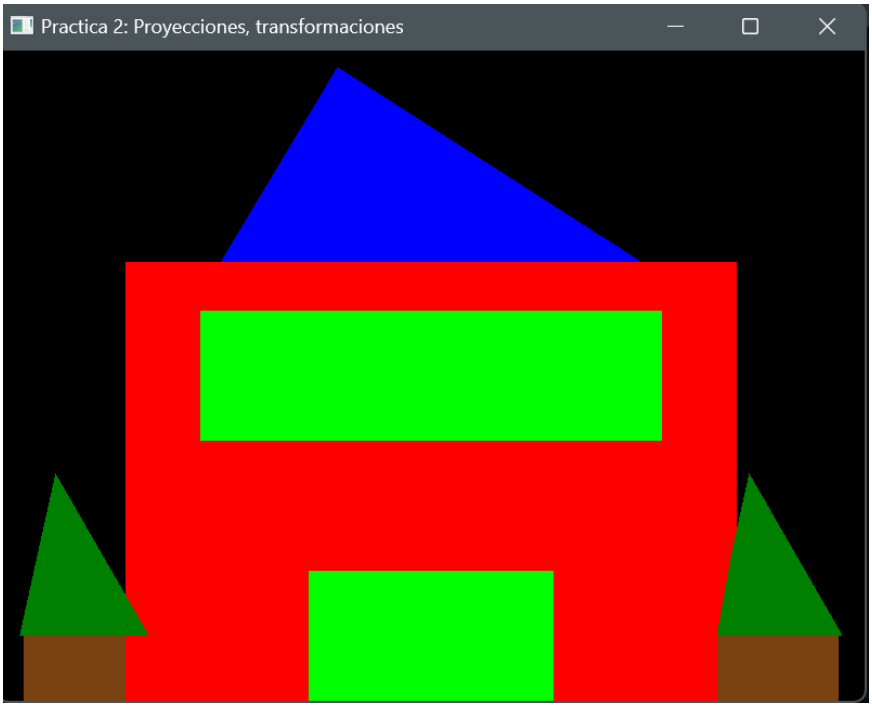


Figura 16. Ejecución código al rotar las figuras geométricas.

Al compilar el código de esta práctica no se me presento ningún inconveniente respecto a la programación de este, puesto que todo funciono con normalidad.

CONCLUSIÓN:

El realizar los ejercicios que se nos asignaron en esta práctica sin duda es gran apoyo el editar y modificar el código base que se nos brinda en clase para comprender como se manejan los modelos que no solo son en dos dimensiones o planas, sino que ahora utilizando formas geométricas como cubos o pirámides que contienen volumen, además de asimilar mejor el funcionamiento de las transformaciones geométricas como translate que mueve la figura de un punto de la pantalla a otro, scale que da la visión de que una figura es mas grande o pequeña y rotate que gira la figura en torno a un eje.

Finalmente, el manejar archivos de shader me sirvió para tener una mejor idea de como es que funcionan este tipo de archivos al igual que su implementación y uso en el código para poder dibujar las figuras.

FUENTES:

OPENGL. (s.f.). *Uniform (GLSL)*. Recuperado el 22 de agosto de 2024, de: https://www.khronos.org/opengl/wiki/Uniform_(GLSL)

OPENGL-TUTORIAL. (s.f.). *Tutorial 3: Matrices*. Recuperado el 22 de agosto de 2024, de: http://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/