UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 01

NOMBRE COMPLETO: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: sábado 10 de febrero de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

- **Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa**

1. **Generar las figuras copiando los vértices de “triangulorojo” y “cuadradoverde”:**

Únicamente copiando y modificando el color de las figuras se consigue crearlas:

- triángulo azul

```
GLfloat vertices_trianguloazul[] = {
    //X         Y           Z           R       G       B
    -1.0f,  -1.0f,      0.5f,           0.0f,   0.0f,   1.0f,
    1.0f,   -1.0f,      0.5f,           0.0f,   0.0f,   1.0f,
    0.0f,   1.0f,       0.5f,           0.0f,   0.0f,   1.0f,

};

MeshColor* trianguloazul = new MeshColor();
trianguloazul->CreateMeshColor(vertices_trianguloazul, 18);
meshColorList.push_back(trianguloazul);
```

- triangulo verde (0,0.5,0)

```
GLfloat vertices_trianguloverde[] = {
    //X         Y           Z           R       G       B
    -1.0f,  -1.0f,      0.5f,           0.0f,   0.5f,   0.0f,
    1.0f,   -1.0f,      0.5f,           0.0f,   0.5f,   0.0f,
    0.0f,   1.0f,       0.5f,           0.0f,   0.5f,   0.0f,

};

MeshColor* trianguloverde = new MeshColor();
trianguloverde->CreateMeshColor(vertices_trianguloverde, 18);
meshColorList.push_back(trianguloverde);
```

- cuadrado rojo

```
GLfloat vertices_cuadradorojo[] = {
    //X         Y           Z           R       G       B
    -0.5f,  -0.5f,      0.5f,           1.0f,   0.0f,   0.0f,
    0.5f,   -0.5f,      0.5f,           1.0f,   0.0f,   0.0f,
    0.5f,   0.5f,       0.5f,           1.0f,   0.0f,   0.0f,
    -0.5f,  -0.5f,      0.5f,           1.0f,   0.0f,   0.0f,
    0.5f,   0.5f,       0.5f,           1.0f,   0.0f,   0.0f,
    -0.5f,  0.5f,       0.5f,           1.0f,   0.0f,   0.0f,

};
MeshColor* cuadradorojo = new MeshColor();
cuadradorojo->CreateMeshColor(vertices_cuadradorojo, 36);
meshColorList.push_back(cuadradorojo);
```

- cuadrado verde

```
GLfloat vertices_cuadradoverde[] = {
    //X         Y           Z           R       G       B
    -0.5f,  -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
    0.5f,   -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
    0.5f,   0.5f,       0.5f,           0.0f,   1.0f,   0.0f,
    -0.5f,  -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
    0.5f,   0.5f,       0.5f,           0.0f,   1.0f,   0.0f,
    -0.5f,  0.5f,       0.5f,           0.0f,   1.0f,   0.0f,

};
MeshColor* cuadradoverde = new MeshColor();
cuadradoverde->CreateMeshColor(vertices_cuadradoverde, 36);
meshColorList.push_back(cuadradoverde);
```

- cuadrado café (0.478, 0.255, 0.067)

```
GLfloat vertices_cuadradocafe[] = {
    //X         Y           Z           R       G       B
    -0.5f,  -0.5f,      0.5f,           0.478, 0.255, 0.067,
    0.5f,   -0.5f,      0.5f,           0.478, 0.255, 0.067,
    0.5f,   0.5f,       0.5f,           0.478, 0.255, 0.067,
    -0.5f,  -0.5f,      0.5f,           0.478, 0.255, 0.067,
    0.5f,   0.5f,       0.5f,           0.478, 0.255, 0.067,
    -0.5f,  0.5f,       0.5f,           0.478, 0.255, 0.067,

};
MeshColor* cuadradocafe = new MeshColor();
cuadradocafe->CreateMeshColor(vertices_cuadradocafe, 36);
meshColorList.push_back(cuadradocafe);
}
```

2. **Usando la proyección ortogonal generar el siguiente dibujo a partir de instancias de las figuras anteriormente creadas , recordar que todos se dibujan en el origen y por transformaciones geométricas se desplazan**

Para esta parte, las figuras instanciadas están organizadas por tipo y color:

```
//TRIANGULO AZUL
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(+0.0f, +0.6f, -4.0f));
model = glm::scale(model, glm::vec3(0.6f, 0.4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[2]->RenderMeshColor();
```

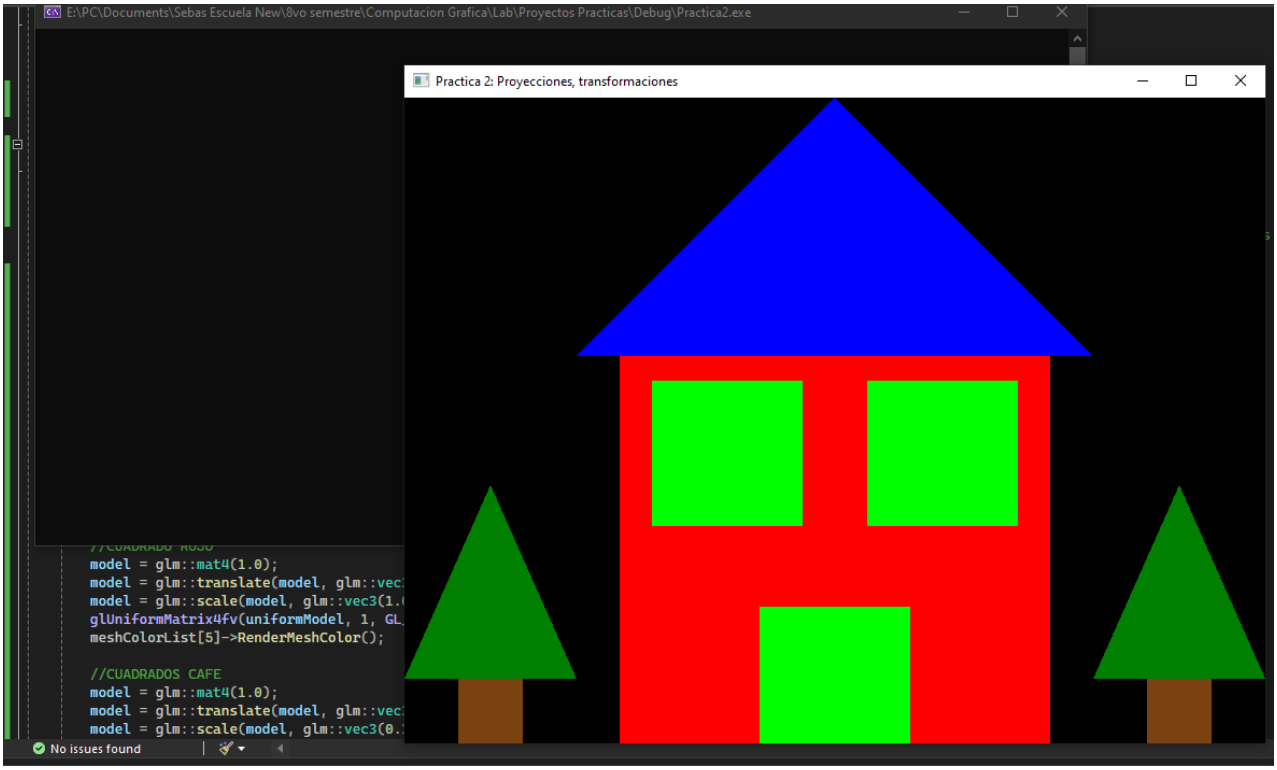
```
//TRIANGULOS VERDES
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.8f, -0.5f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.3f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[3]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(+0.8f, -0.5f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.3f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[3]->RenderMeshColor();
```

```
//CUADRADO ROJO
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.4f, -4.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.35f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[5]->RenderMeshColor();
```

```
//CUADRADOS CAFE
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(+0.8f, -0.9f, -4.0f));
model = glm::scale(model, glm::vec3(0.15f, 0.25f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[6]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.8f, -0.9f, -4.0f));
model = glm::scale(model, glm::vec3(0.15f, 0.25f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[6]->RenderMeshColor();
```

```
//CUADRADOS VERDES
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(+0.0f, -0.8f, -4.0f));
model = glm::scale(model, glm::vec3(0.35f, 0.45f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[4]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(+0.25f, -0.1f, -4.0f));
model = glm::scale(model, glm::vec3(0.35f, 0.45f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[4]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.25f, -0.1f, -4.0f));
model = glm::scale(model, glm::vec3(0.35f, 0.45f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshColorList[4]->RenderMeshColor();
```

Y la ejecución se ve así:



- **Problemas presentados**

  No hubo problemas durante la ejecución del código.

- **Conclusión:**

1. Los ejercicios de la clase: Complejidad, explicación

Fue un ejercicio entretenido, me ayudo a entender como funcionan las funciones de escalado y traslación de un entorno 2D. Además, aprendí a crear figuras y a como darles color para mostrarlas en pantalla.

2. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.

Fue una clase larga, pero también fue muy útil saber cómo funciona todo el código, sobre todo para las practicas posteriores. Aunque algo muy útil seria que hubiera mas comentarios en el código proporcionado hablando con un poquito mas de detalle de algunas cosas, como en el archivo shader.h.