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 02

NOMBRE COMPLETO: Oropeza Sánchez Guadalupe M

Nº de Cuenta: 317144547

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 20 de febrero de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

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

   - Se generan las figuras a partir de este código:

```
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);

GLfloat vertices_triangulorojo[] = {
    //X         Y           Z           R       G       B
    -1.0f,  -1.0f,      0.5f,           1.0f,   0.0f,   0.0f,
    1.0f,   -1.0f,      0.5f,           1.0f,   0.0f,   0.0f,
    0.0f,    1.0f,      0.5f,           1.0f,   0.0f,   0.0f
};

MeshColor* triangulorojo = new MeshColor();
triangulorojo->CreateMeshColor(vertices_triangulorojo, 18);
meshColorList.push_back(triangulorojo);

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);
```

   Se ocupan los mismos vértices para los triángulos, lo único que cambia es la combinación del RGB.

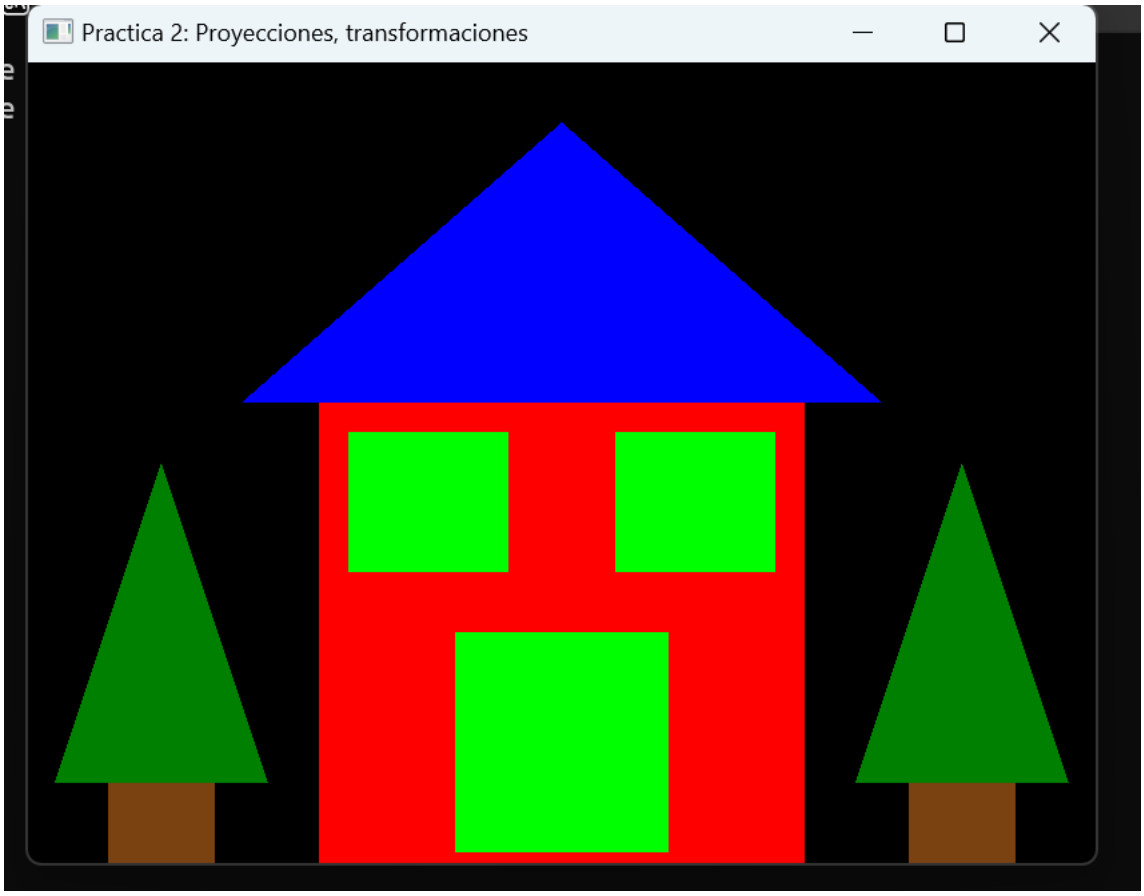
```
GLfloat vertices_cuadradoverde[] = {
    //X         Y           Z           R       G       B
-0.5f,  -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
0.5f,   -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
0.5f,   0.5f,       0.5f,           0.0f,   1.0f,   0.0f,
-0.5f,  -0.5f,      0.5f,  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


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);

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);
```

De igual manera con los cuadrados.

- Para el dibujo de la casa se ocupan dichas figuras y

```
glm::mat4 model_cuadradorojo(1.0);
model_cuadradorojo = glm::translate(model_cuadradorojo, glm::vec3(0.0f, -0.5f, -4.0f));
model_cuadradorojo = glm::rotate(model_cuadradorojo, angulo, glm::vec3(1.0f, 0.0f, 0.0f));
model_cuadradorojo = glm::scale(model_cuadradorojo, glm::vec3(.91f, 1.4f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model_cuadradorojo));//FALSE
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[5]->RenderMeshColor();
```

Con este bloque de código se configura el shader y las matrices de transformación y proyección, y después se renderiza el objeto en pantalla, con ***scale*** y ***translate*** se modifica el tamaño de las figuras y se mueven.



2. Problemas presentados. En esta practica no se presento ningún inconveniente o problema.

3. Conclusión:
    a. Los ejercicios de la clase:

    El primer ejercicio fue sencillo, ya que se tenían que crear figuras y se tomaron los vértices ya establecidos y solo se les cambio el color.

    Para el segundo ejercicio, se tenían que mover las figuras y escalar, aquí la única complejidad fue que estéticamente se viera bien (asimétrico).

    b. Comentarios generales: La explicación fue clara y completa.