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: David Sánchez Gutiérrez

Nº de Cuenta: 315596397

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 04

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 20 de febrero del 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**
   - **a.** Generar las figuras copiando los vértices de triangulo rojo y cuadrado verde.
     - **i.** Código

Triángulo Azul

```
//Triangulo azul -> [3]
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

```
//Triangulo verde -> [4]
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

```
//Cuadrado rojo -> [5]
GLfloat verticesCuadradoRojo[] = {
    //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(verticesCuadradoRojo, 36);
meshColorList.push_back(cuadradoRojo);
```

Cuadrado Verde

```
//Cuadrado verde -> [2]
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

```
//Cuadrado cafe -> [6]
GLfloat verticesCuadradoCafe[] = {
    //X         Y           Z           R       G       B
    -0.5f,  -0.5f,      0.5f,           0.478f, 0.255f, 0.067f,
    0.5f,   -0.5f,      0.5f,           0.478f, 0.255f, 0.067f,
    0.5f,   0.5f,       0.5f,           0.478f, 0.255f, 0.067f,
    -0.5f,  -0.5f,      0.5f,           0.478f, 0.255f, 0.067f,
    0.5f,   0.5f,       0.5f,           0.478f, 0.255f, 0.067f,
    -0.5f,  0.5f,       0.5f,           0.478f, 0.255f, 0.067f,

};

MeshColor* cuadradoCafe = new MeshColor();
cuadradoCafe->CreateMeshColor(verticesCuadradoCafe, 36);
meshColorList.push_back(cuadradoCafe);
```

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

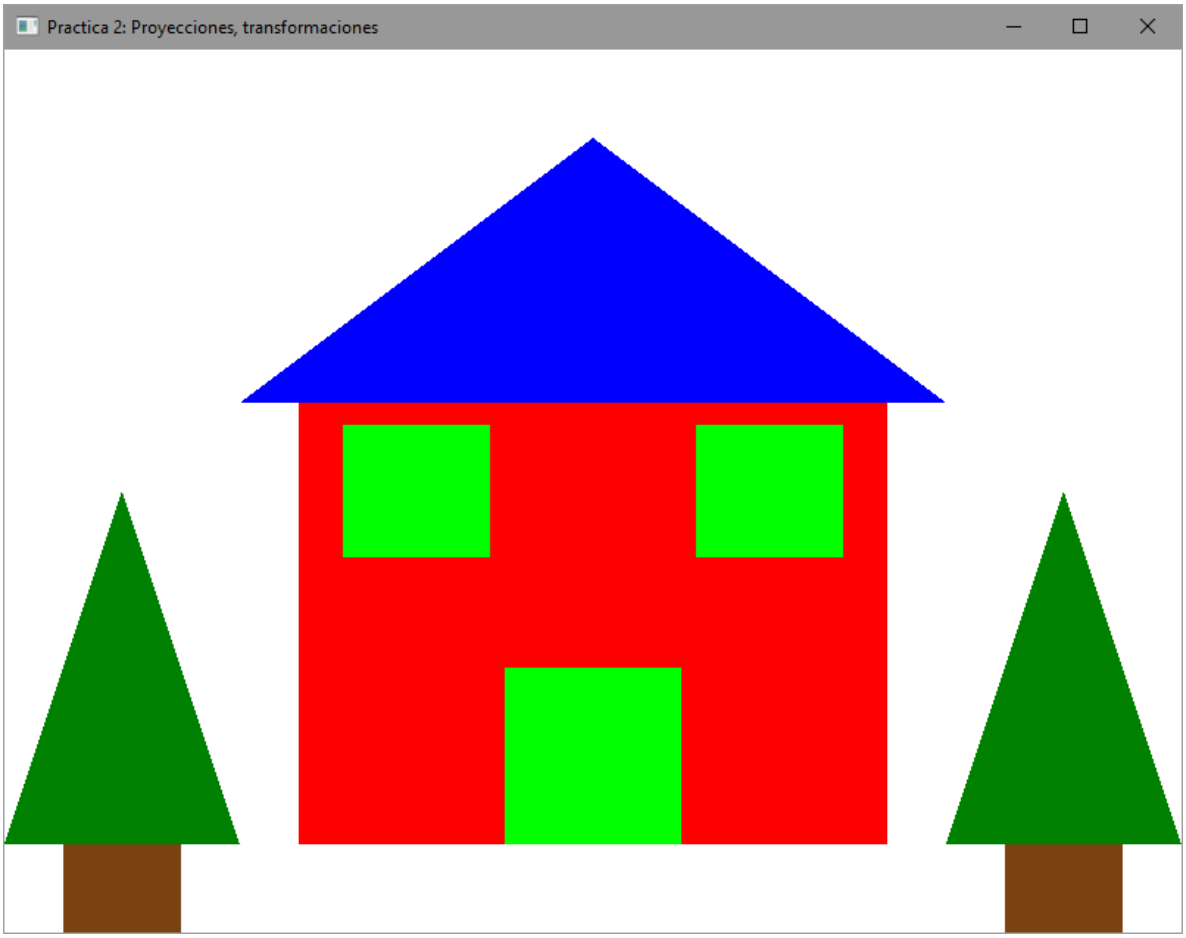
i. Código

```
//Cuadrado Rojo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.3f, -4.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y se e
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[5]->RenderMeshColor();

//Triángulo Azul
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.5f, -4.0f));
model = glm::scale(model, glm::vec3(0.6f, 0.3f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y se e
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[3]->RenderMeshColor();

//Cuadrados verdes PUERTA
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.6f, -3.9f));
model = glm::scale(model, glm::vec3(0.3f, 0.4f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y se e
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[2]->RenderMeshColor();

//Cuadrados verdes Ventanas
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.3f, 0.0f, -3.9f));
model = glm::scale(model, glm::vec3(0.25f, 0.3f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y se e
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[2]->RenderMeshColor();
```

```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.3f, 0.0f, -3.9f));
model = glm::scale(model, glm::vec3(0.25f, 0.3f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y s
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[2]->RenderMeshColor();

//Arboles
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, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y s
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[6]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.8f, -0.4f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.4f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y s
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[4]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.8f, -0.9f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y s
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[6]->RenderMeshColor();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.8f, -0.4f, -4.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.4f, 1.0f));
```

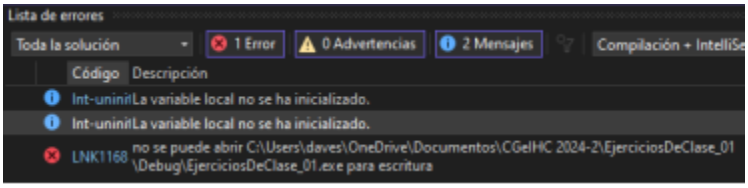
```
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y se envian
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[4]->RenderMeshColor();
```

ii. Captura



2. Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código.

Se presento el mismo error de ejecución que se me presento en la práctica 1, hizo que me tomara mas tiempo del estimado para realizar este ejercicio.



3. **Conclusión:**
    a. **Los ejercicios de la clase: Complejidad, explicación.**

Fue un ejercicio de clase bastante completo ya que esta vez usamos dos figuras: Triángulo y Cuadrado, que con ayuda de las transformaciones geométricas es más útil ya que no tenemos que programar vértice por vértice.

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

Fue una práctica acorde a lo enseñado en clase.