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: Rosas Martinez Francisco Javier

Nº de Cuenta: 320109766

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 05

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 7/Septiembre/2025

CALIFICACIÓN: ________

REPORTE DE PRÁCTICA:

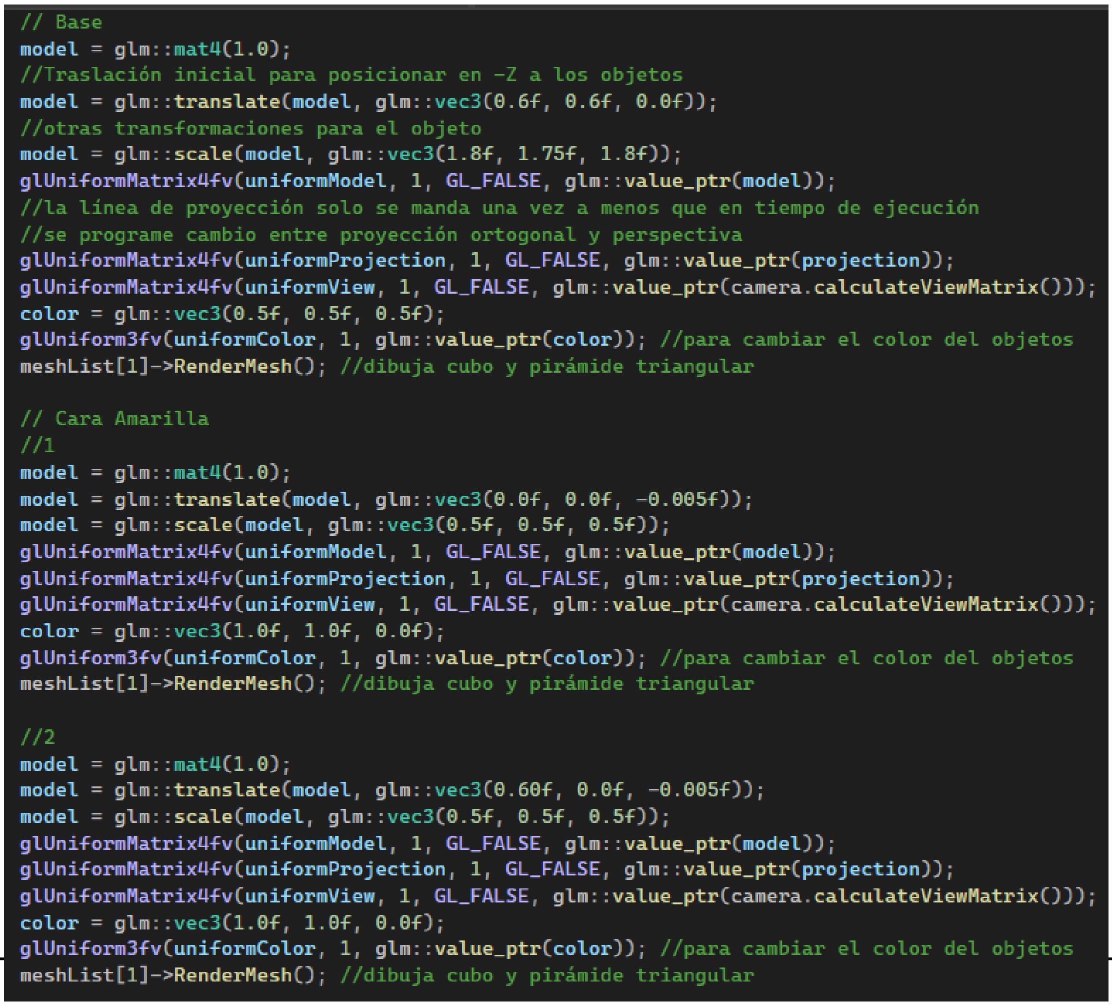
1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

Ejercicio 1:

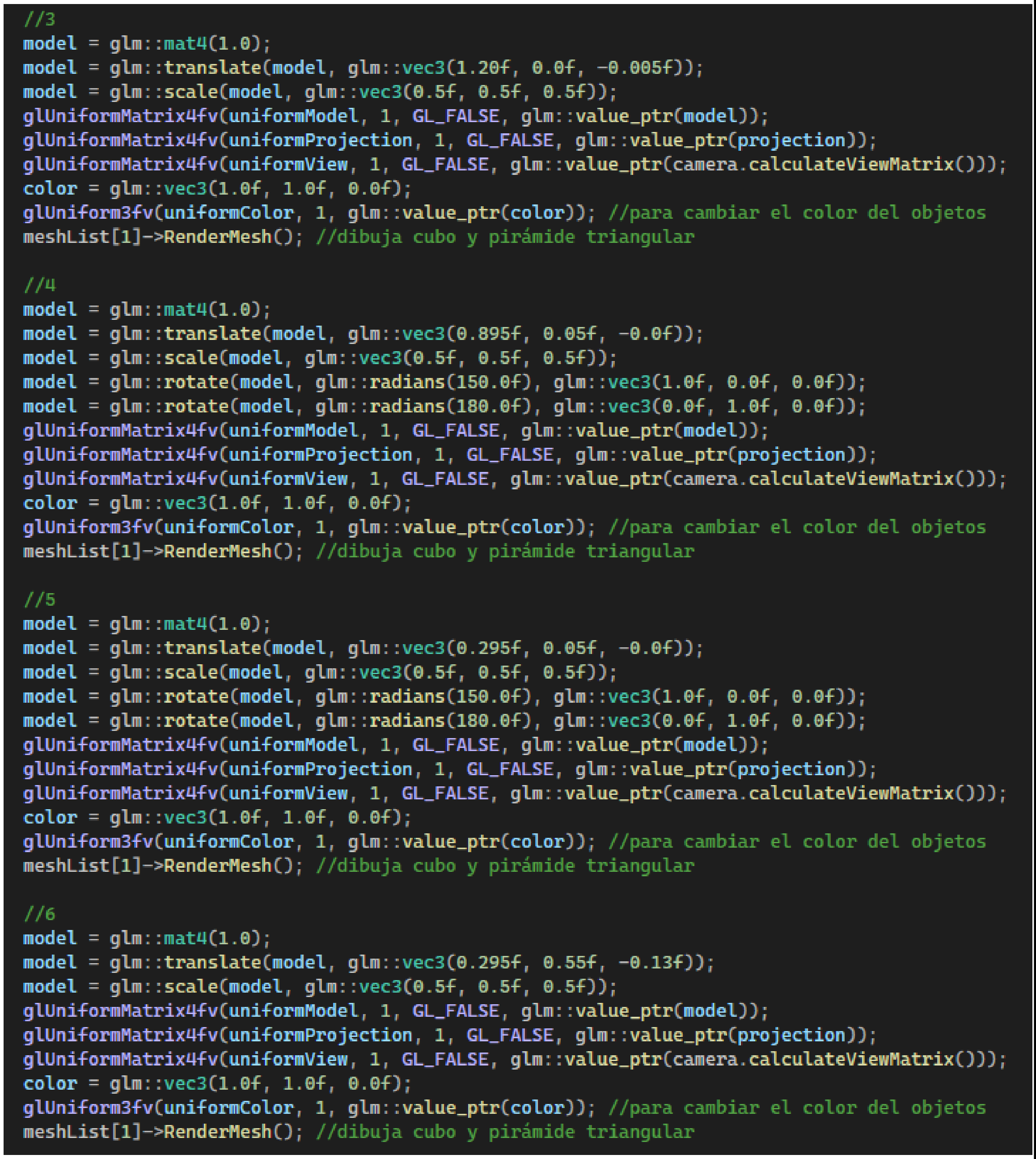
1.- Generar una pirámide rubik (pyraminx) de 9 pirámides por cara.

Para resolver el ejercicio tuve que crear nueve pirámides por cada cara de la pirámide triangular, con la diferencia de que en cada cara el color debía de ser distinto, únicamente tuve que instanciar varias veces la pirámide que nos proporcionó el profesor cambiando los atributos correspondientes.

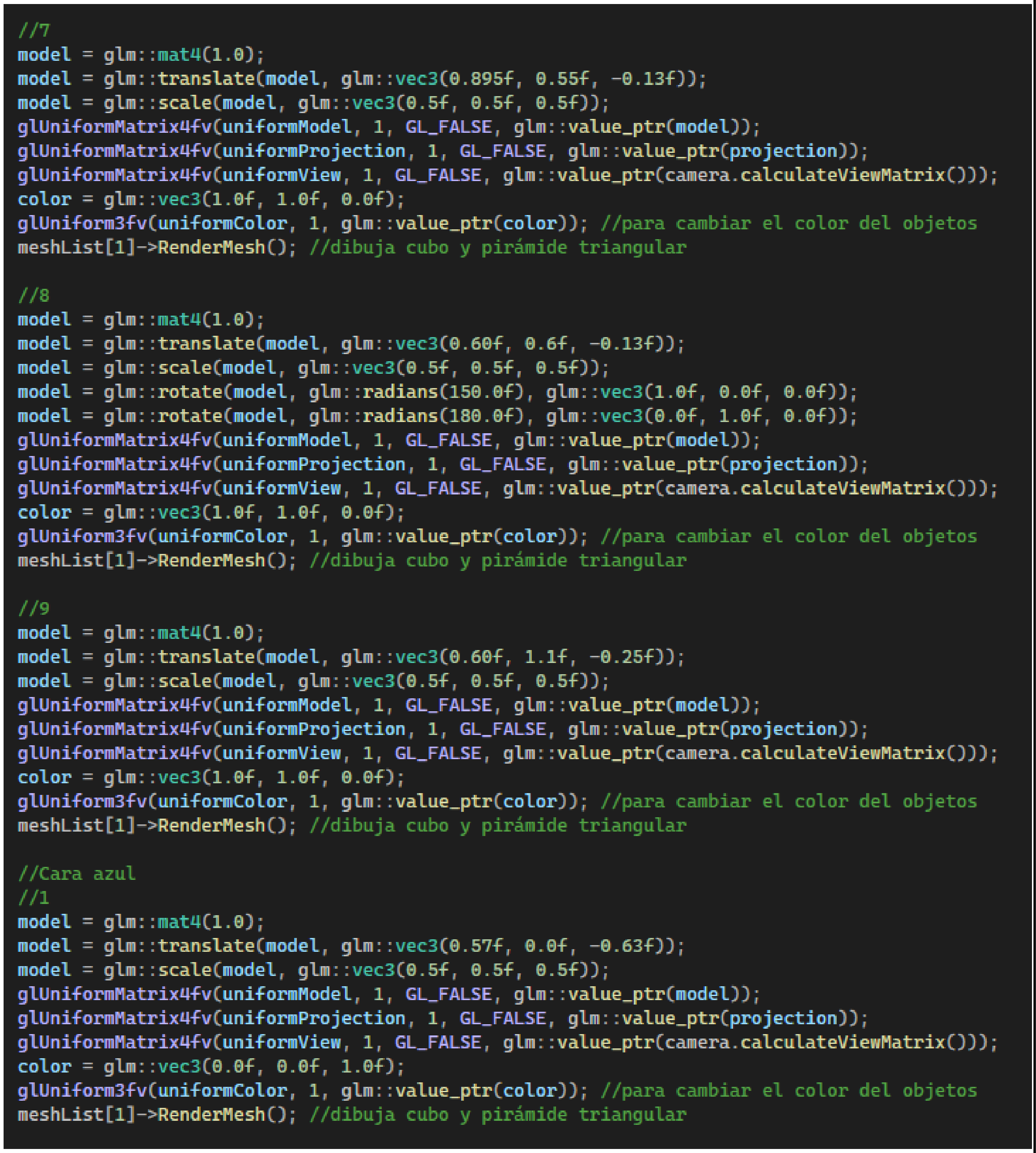
Bloques de código generado

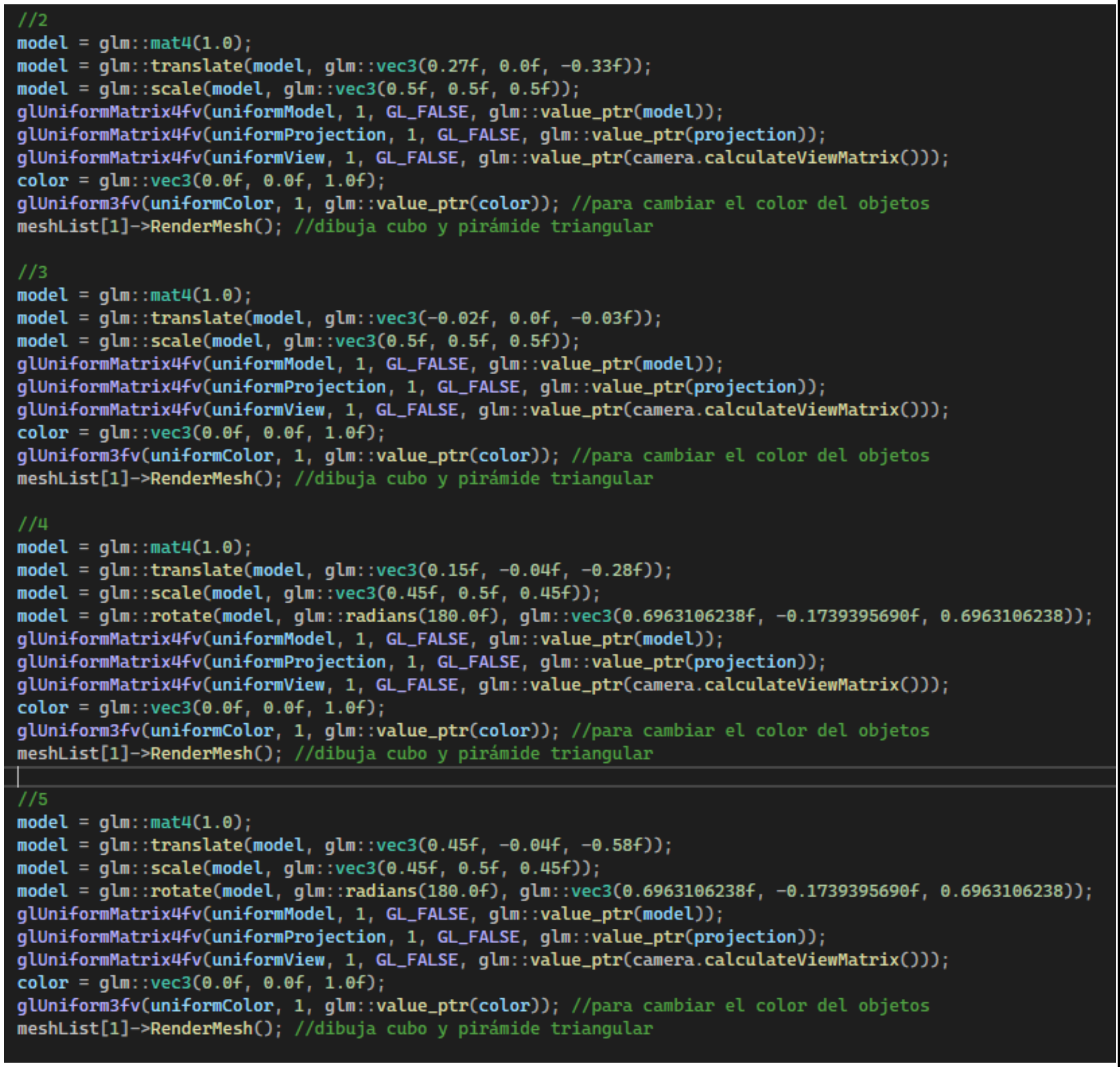


```
// Base
model = glm::mat4(1.0);
//Traslación inicial para posicionar en -Z a los objetos
model = glm::translate(model, glm::vec3(0.6f, 0.6f, 0.0f));
//otras transformaciones para el objeto
model = glm::scale(model, glm::vec3(1.8f, 1.75f, 1.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
//se programe cambio entre proyección ortogonal y perspectiva
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.5f, 0.5f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

// Cara Amarilla
//1
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -0.005f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//2
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.60f, 0.0f, -0.005f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```

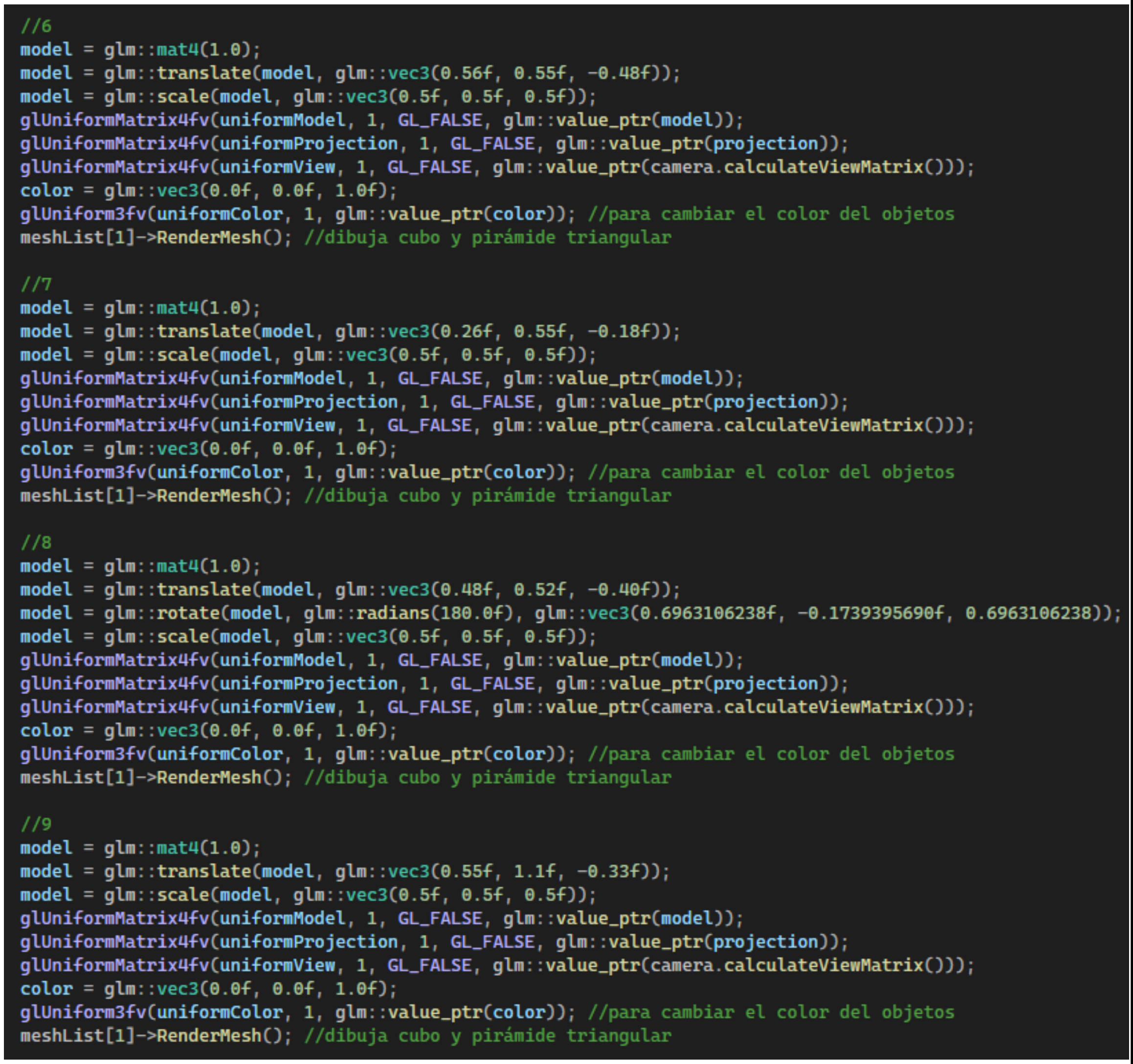


```
//3
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.20f, 0.0f, -0.005f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//4
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.895f, 0.05f, -0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(150.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//5
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.295f, 0.05f, -0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(150.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//6
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.295f, 0.55f, -0.13f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```

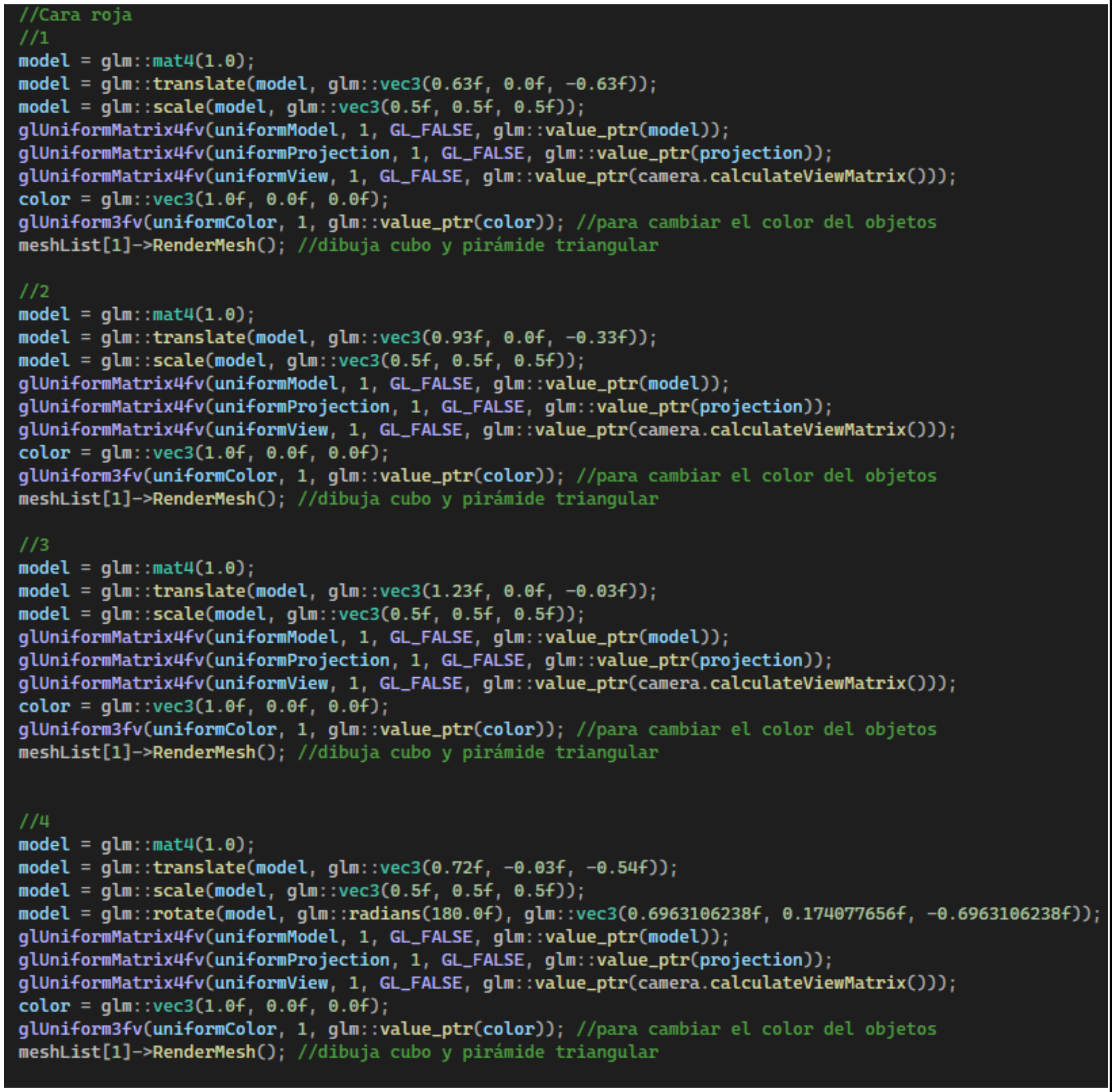

```
//7
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.895f, 0.55f, -0.13f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//8
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.60f, 0.6f, -0.13f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(150.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//9
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.60f, 1.1f, -0.25f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//Cara azul
//1
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.57f, 0.0f, -0.63f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```



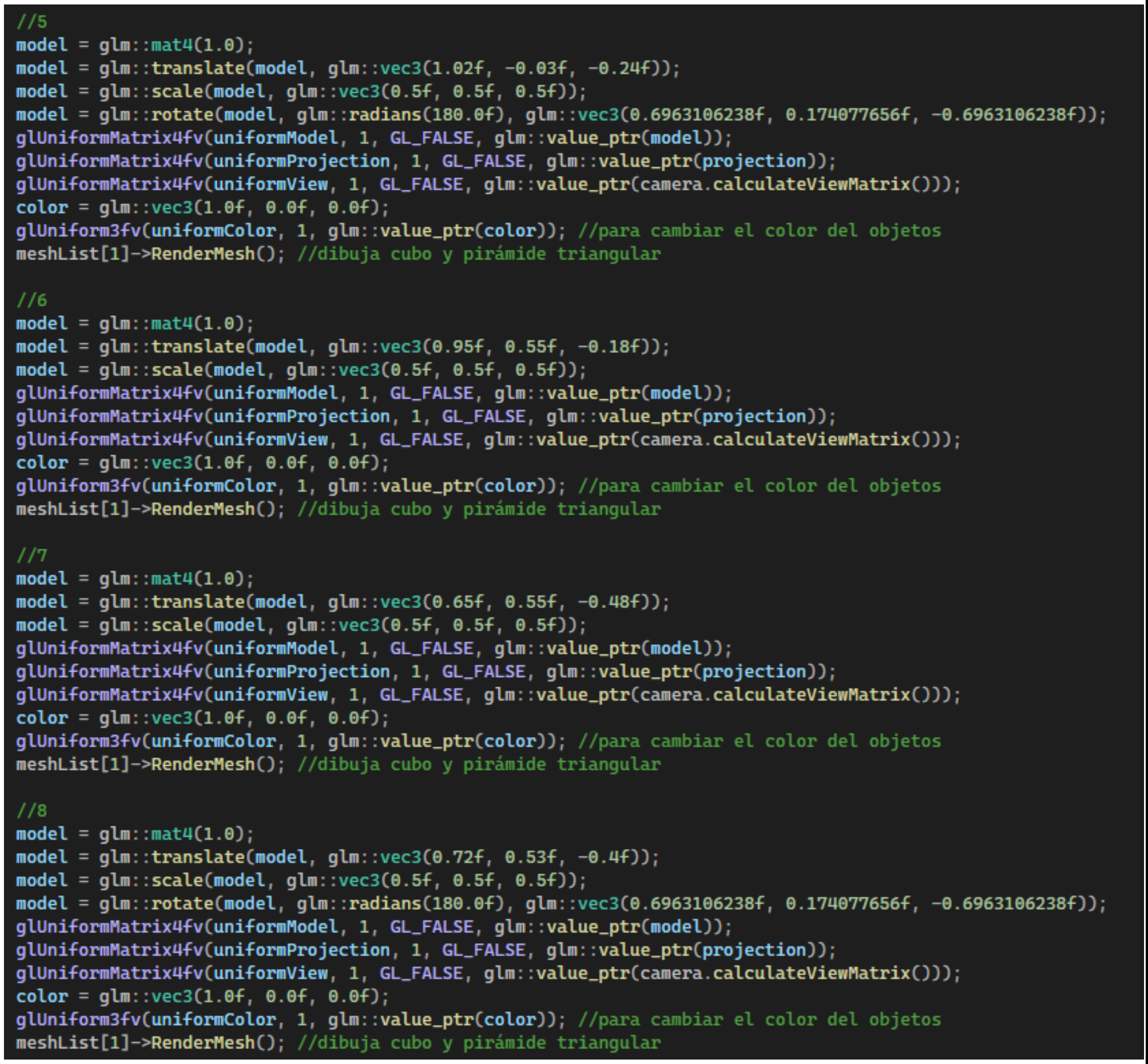
```
//2
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.27f, 0.0f, -0.33f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//3
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.02f, 0.0f, -0.03f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//4
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.15f, -0.04f, -0.28f));
model = glm::scale(model, glm::vec3(0.45f, 0.5f, 0.45f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.6963106238f, -0.1739395690f, 0.6963106238));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//5
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.45f, -0.04f, -0.58f));
model = glm::scale(model, glm::vec3(0.45f, 0.5f, 0.45f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.6963106238f, -0.1739395690f, 0.6963106238));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```



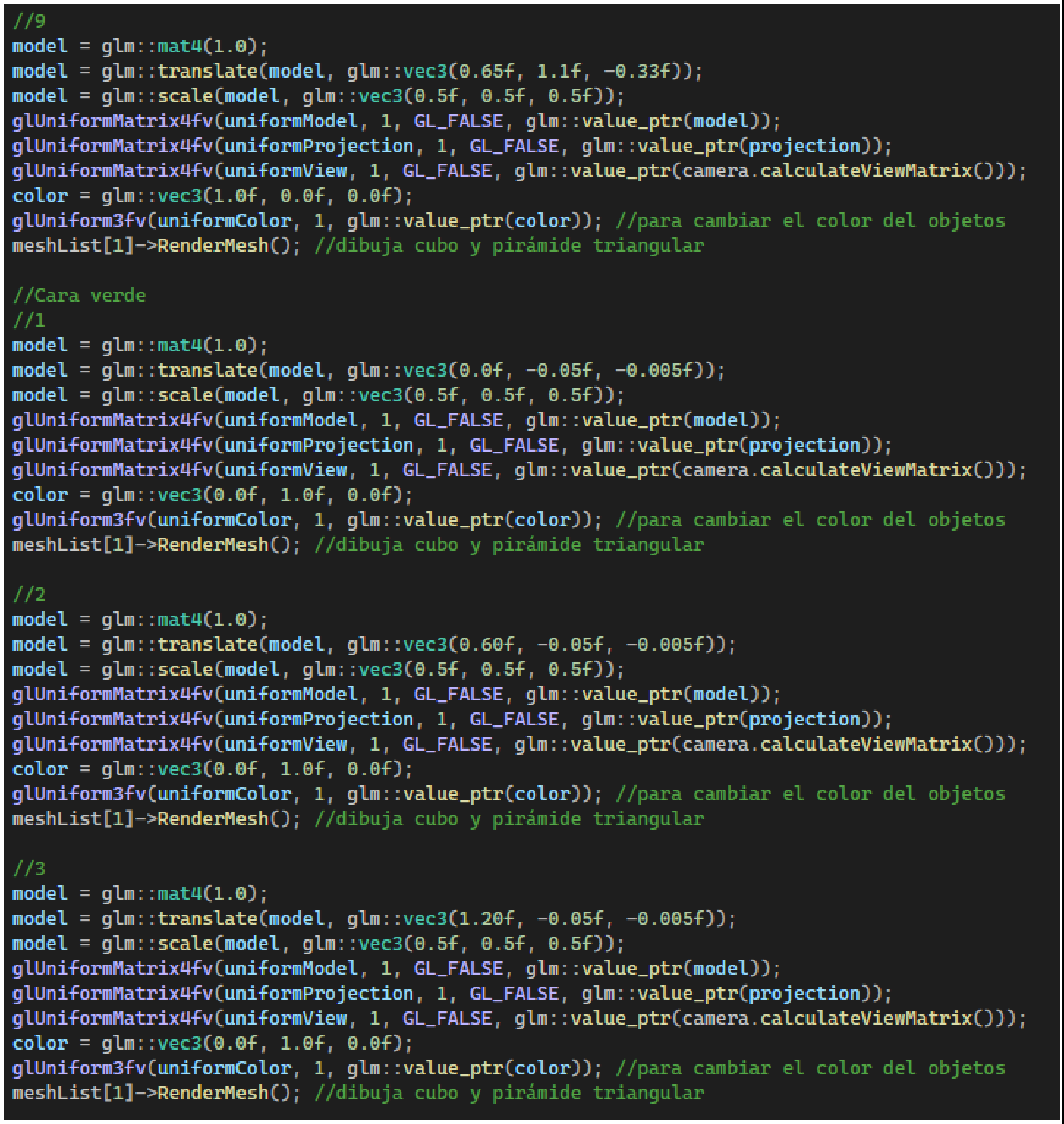
```
//6
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.56f, 0.55f, -0.48f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//7
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.26f, 0.55f, -0.18f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//8
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.48f, 0.52f, -0.40f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.6963106238f, -0.1739395690f, 0.6963106238));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//9
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.55f, 1.1f, -0.33f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```



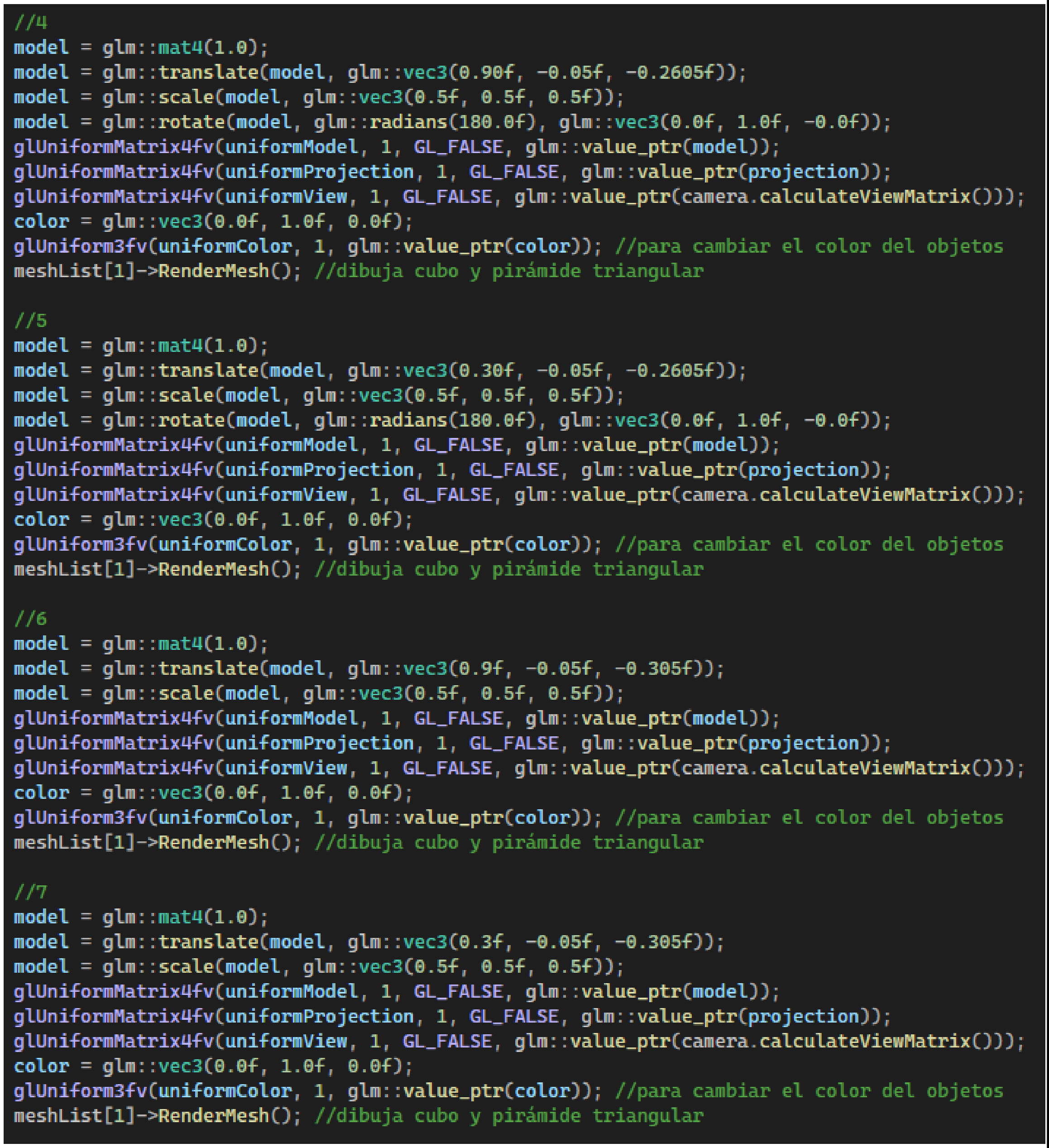
```
//Cara roja
//1
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.63f, 0.0f, -0.63f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//2
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.93f, 0.0f, -0.33f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//3
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.23f, 0.0f, -0.03f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular


//4
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.72f, -0.03f, -0.54f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.6963106238f, 0.174077656f, -0.6963106238f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```



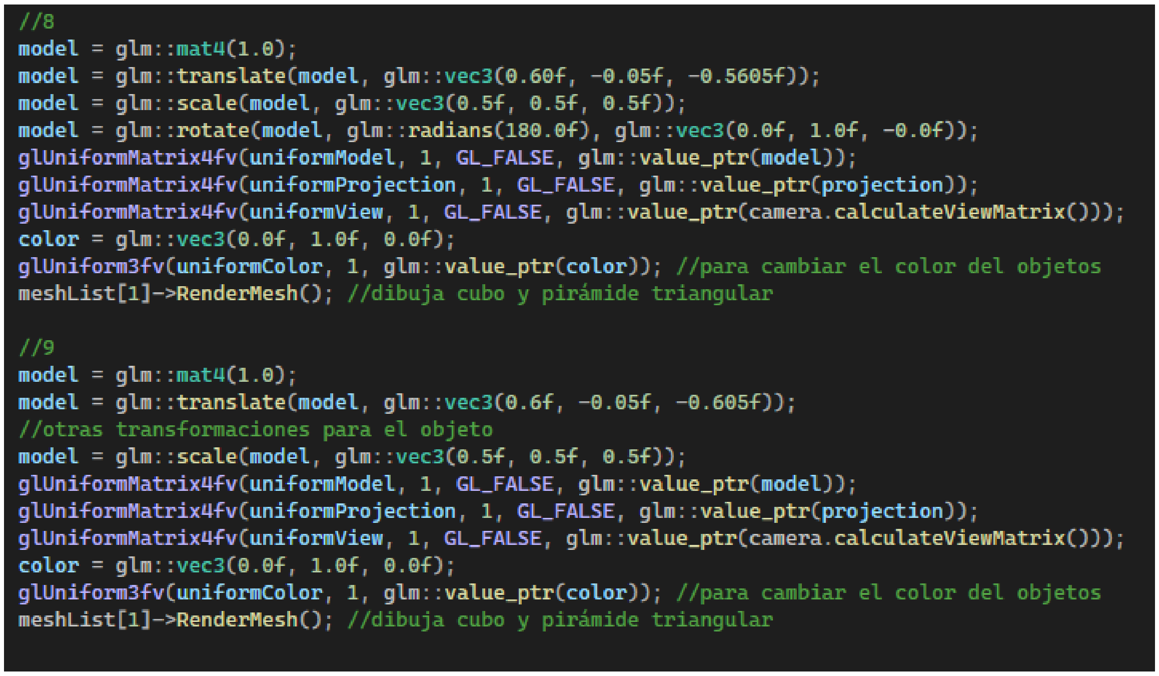
```
//5
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.02f, -0.03f, -0.24f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.6963106238f, 0.174077656f, -0.6963106238f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//6
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.95f, 0.55f, -0.18f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//7
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.65f, 0.55f, -0.48f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//8
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.72f, 0.53f, -0.4f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.6963106238f, 0.174077656f, -0.6963106238f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```



```
//9
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.65f, 1.1f, -0.33f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//Cara verde
//1
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.05f, -0.005f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//2
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.60f, -0.05f, -0.005f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//3
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.20f, -0.05f, -0.005f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```

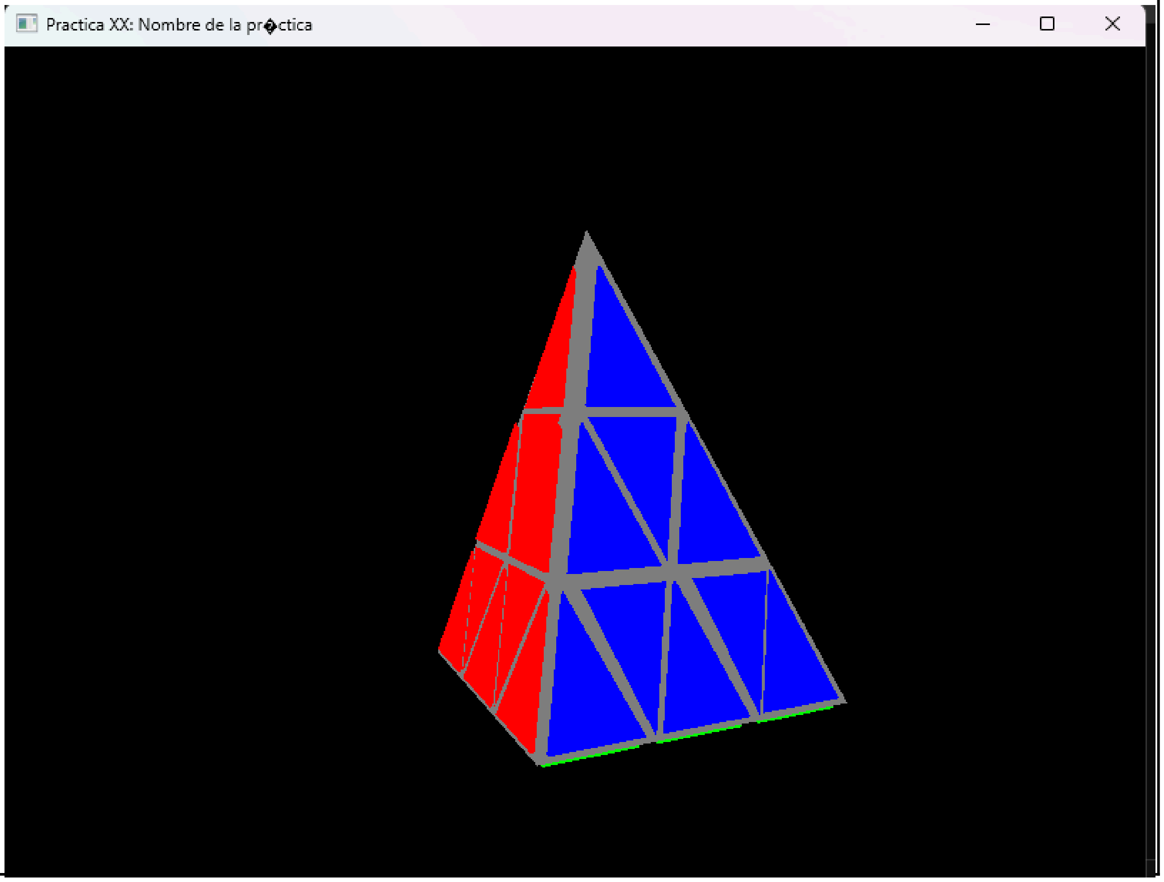


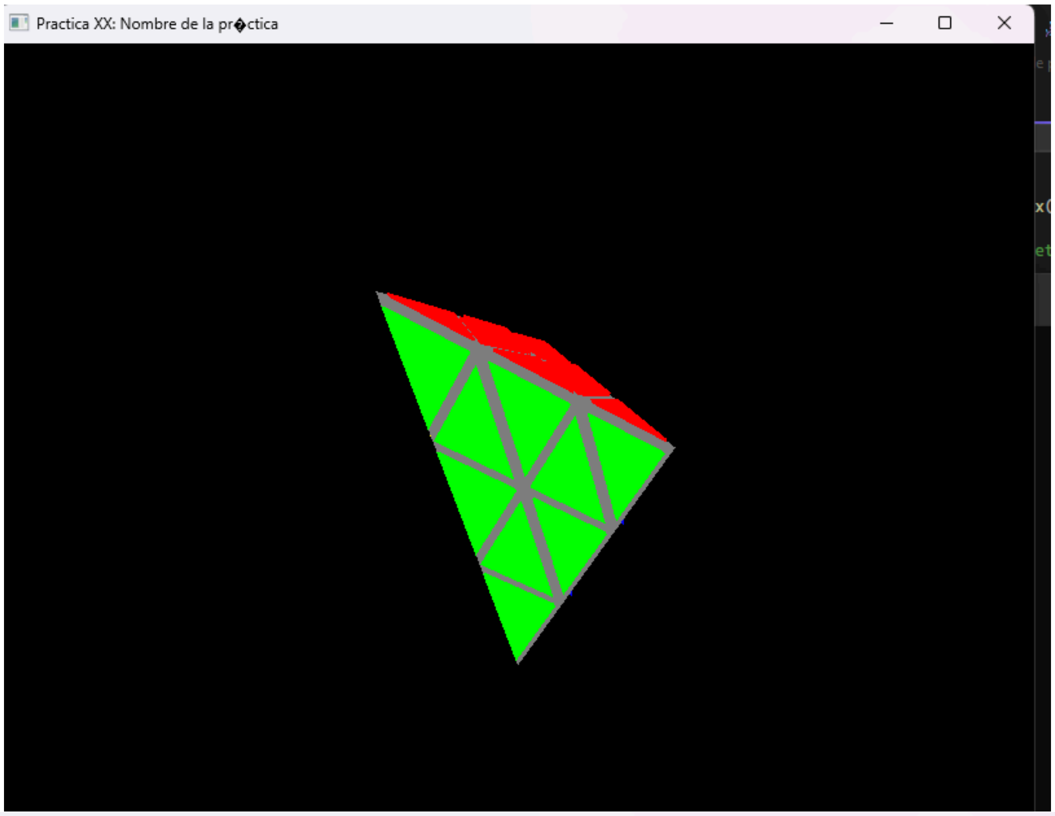
```
//4
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.90f, -0.05f, -0.2605f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, -0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//5
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.30f, -0.05f, -0.2605f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, -0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//6
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.9f, -0.05f, -0.305f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//7
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.3f, -0.05f, -0.305f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```

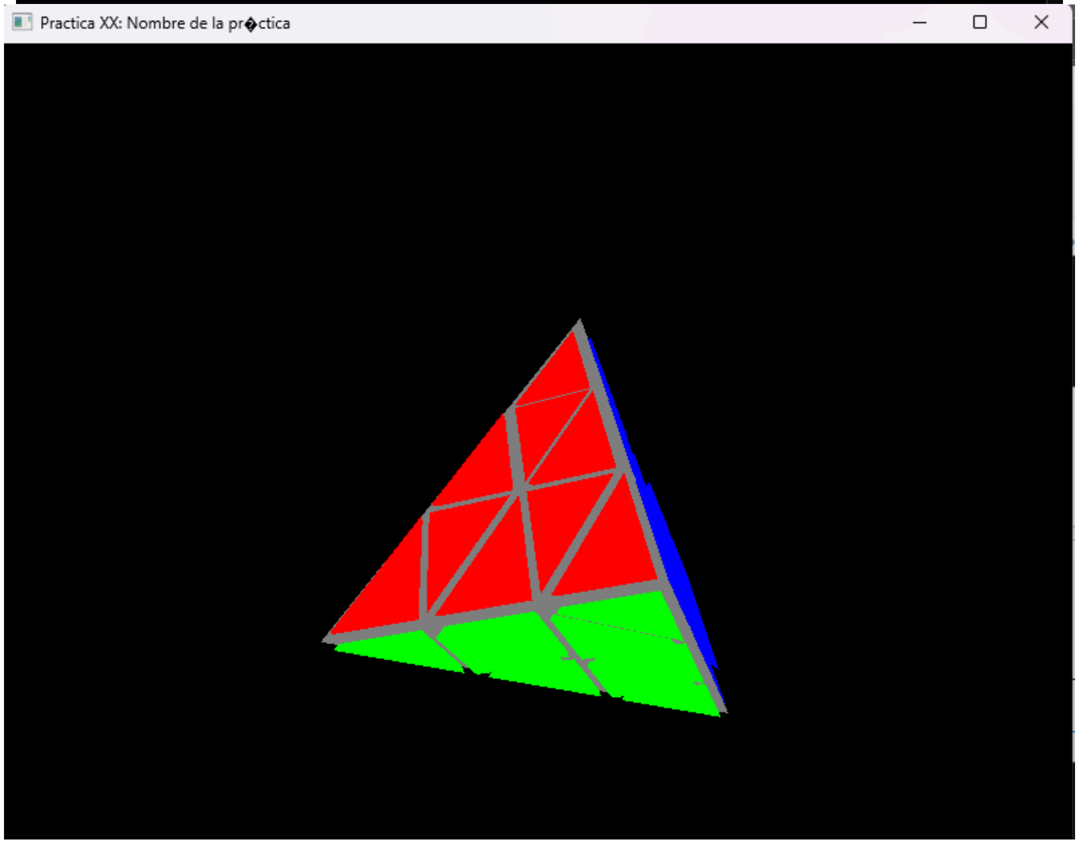


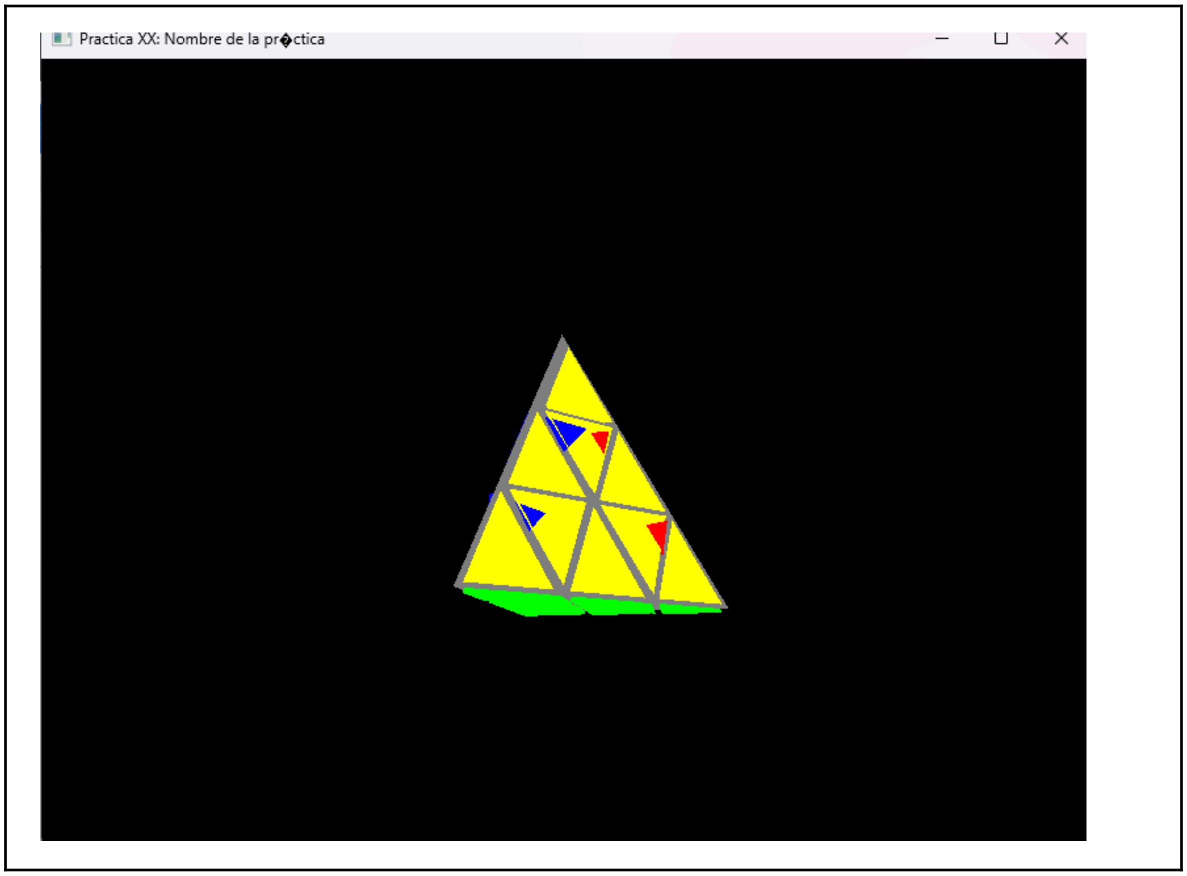
```
//8
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.60f, -0.05f, -0.5605f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, -0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular

//9
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.6f, -0.05f, -0.605f));
//otras transformaciones para el objeto
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Código en ejecución




Practica XX: Nombre de la pr�ctica


Practica XX: Nombre de la pr�ctica



2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

- Tuve demasiadas complicaciones con las pirámides que tenía que invertir, no entendía qué vector debía de mandar en la función *rotate*. Después de varios intentos me di cuenta de que la clave estaba en usar el vector normal de la cara donde quería colocar la pirámide. Ese vector me dio la orientación correcta, porque indica hacia dónde apunta la superficie, y al usarlo como eje de rotación logré alinear la pirámide de forma precisa. En pocas palabras, comprendí que el problema no era la figura en sí, sino la dirección en la que estaba girando, y que con la normal se podía resolver de manera matemática y exacta, en lugar de hacerlo a prueba y error.

- Cuando estaba terminando de colocar cada una de las pirámides me di cuenta de que cuatro de ellas sobrepasaron más área de lo debido, como se puede observar en la imagen de la cara amarilla. Esto ocurrió porque no verifiqué que la pirámide tuviera la misma dimensión en todas sus caras, lo que provocó que en esa cara amarilla sobresalieran cuatro de las pirámides. Lamentablemente me di cuenta de este problema cuando ya había terminado de colocar casi todas, por lo cual sería muy tardado volver a iniciar de cero con una pirámide que tuviera todos sus lados iguales. Para solucionarlo pensé en disminuir la escala en algunos ejes de las pirámides, pero aunque probé con distintos valores, no logré eliminar por completo el exceso que sobresalía.

Conclusión:

Con esta práctica logré reforzar mis habilidades en el uso de las funciones *scale*, *translate* y *rotate*. Gracias a ellas pude formar un pyraminx compuesto por 9 pirámides en cada cara. A mi parecer no fue una práctica complicada de resolver, pero sí resultó bastante laboriosa, ya que tuve que colocar una por una las pirámides en la posición correcta. Las que tenían su orientación invertida me tomaron mucho tiempo porque no lograba acomodarlas de manera adecuada. Sin embargo, con el apoyo de la bibliografía pude comprender cómo solucionar correctamente el problema. Aunque el resultado final no fue perfecto debido a un detalle que no consideré, logré cumplir con la realización de la actividad.

Bibliografía :

OpenGL Architecture Review Board, Shreiner, D., Woo, M., Neider, J., & Davis, T. (2010). OpenGL Programming Guide: The Official Guide to Learning OpenGL (7th ed.). Addison-Wesley.

OpenGL. (s. f.). OpenGL Documentation. Khronos Group. Recuperado de https://www.khronos.org/opengl/

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3 (8th ed.). Addison-Wesley Professional.