UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

Reporte de práctica 3: modelado geométrico

NOMBRE COMPLETO: Gonzalez Villalba Bryan Jesus

Nº de Cuenta: 421530869

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 4

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 4 de septiembre del 2024

CALIFICACIÓN: _____

Introducción:

Para este reporte dibujaremos una pirámide triangular en la que dibujaremos las caras como de una piraminx en la cual una cara es roja, otra verde, otra azul y una celeste, para esto ocuparemos código de prácticas pasadas.

Desarrollo:

1. Primero dibujamos la pirámide color negra en un fondo blanco, imagen **1.0**.

```
color = glm::vec3(1.0f, 0.0f, 1.0f);
//glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
//meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
//sp.render(); //dibuja esfera



//ejercicio: Instanciar primitivas geométricas para recrear el dibujo de la práctica pasada en 3D,
//se requiere que exista piso y la casa tiene una ventana azul circular justo en medio de la pared trasera y solo 1 puerta front

// Piramide (negro)
model = glm::mat4(1.0f);
color=glm::vec3(0.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.1f, -4.0f));
model = glm::scale(model, glm::vec3(7.0f, 7.0f, 7.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 1.0

2. Dibujamos la cara color rojo, dibujando 9 pirámides triangulares de color rojo, con coordenadas y rotación diferentes, pero en todo lo demás igual, esta es la cara que más rápido me salió, imagen **2.0** a **2.8**.

```
// Triangulo 1 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f, 2.1f, -5.3f));
model = glm::rotate(model, glm::radians(-11.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.0

```
// Triangulo 2 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.8f));
model = glm::rotate(model, glm::radians(-16.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.1

```
// Triangulo 3 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(-1.1f, 0.0f, -4.8f));
model = glm::rotate(model, glm::radians(-12.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.2

```
// Triangulo 4 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(1.1f, 0.0f, -4.8f));
model = glm::rotate(model, glm::radians(-12.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.3

```
// Triangulo 5 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(-2.2f, -2.1f, -4.25f));
model = glm::rotate(model, glm::radians(-13.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.4

```
// Triangulo 6 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(-1.1f, -2.1f, -4.25f));
model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.5

```
// Triangulo 7 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(0.0f, -2.1f, -4.25f));
model = glm::rotate(model, glm::radians(-13.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.6

```
// Triangulo 8 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(1.1f, -2.1f, -4.25f));
model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.7

```
// Triangulo 9 cara (rojo)
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
//Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
model = glm::translate(model, glm::vec3(2.2f, -2.1f, -4.25f));
model = glm::rotate(model, glm::radians(-13.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

Imagen 2.8

3. Luego dibujamos 9 pirámides triangulares de color verde, fue una de las caras más difíciles de hacer y tuve que trabajar con rotación, imagen **3.0**.

```
507         // Triangulo 1 cara (verde)
508         model = glm::mat4(1.0f);
509         color = glm::vec3(0.0f, 1.0f, 0.0f);
510         //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
511         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
512         model = glm::translate(model, glm::vec3(0.45f, 2.1f, -5.84f));
513         model = glm::rotate(model, glm::radians(14.0f), glm::vec3(1.0f, 0.0f, 0.0f)); // orientacion de inclinacion
514         model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 1.0f, 0.0f)); // orientacion de profundidad
515         model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(0.0f, 0.0f, 1.0f)); // orientacion de lado
516         model = glm::scale(model, glm::vec3(1.4f, 2.0f, 0.3f));
517         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
518         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
519         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
520         meshList[1]->RenderMesh();
```

Imagen 3.0

4. Dibujamos 9 pirámides triangulares color azul, trabajar en esta cara fue difícil y ocupe rotación, imagen **4.0**.

```
642         // Triangulo 1 cara (azul)
643         model = glm::mat4(1.0f);
644         color = glm::vec3(0.0f, 0.0f, 1.0f);
645         //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
646         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
647         model = glm::translate(model, glm::vec3(-0.32f, 2.1f, -5.84f));
648         model = glm::rotate(model, glm::radians(16.0f), glm::vec3(1.0f, 0.0f, 0.0f)); // orientacion de inclinacion
649         model = glm::rotate(model, glm::radians(40.0f), glm::vec3(0.0f, 1.0f, 0.0f)); // orientacion de profundidad
650         model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(0.0f, 0.0f, 1.0f)); // orientacion de lado
651         model = glm::scale(model, glm::vec3(1.4f, 2.0f, 0.3f));
652         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
653         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
654         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
655         meshList[1]->RenderMesh();
```

Imagen 4.0

5. Dibujamos 9 pirámides triangulares color celeste, en donde su dificultad fue muy sencilla, esta fue la segunda cara que más rápido me salió, imagen **5.0**.

```
778         // Triangulo 1 cara (celeste)
779         model = glm::mat4(1.0f);
780         color = glm::vec3(0.0f, 1.0f, 1.0f);
781         //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
782         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
783         model = glm::translate(model, glm::vec3(0.0f, -3.3f, -4.0));
784         model = glm::rotate(model, glm::radians(0.0f), glm::vec3(0.0f, 1.0f, 0.0f)); // orientacion de profundidad
785         model = glm::scale(model, glm::vec3(1.4f, 0.3f, 2.0f));
786         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
787         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
788         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
789         meshList[1]->RenderMesh();
```

Imagen 5.0

6. Resultado final, no es el esperado, pero trate de que estuviera lo más presentable, imagen **6.0**, **6.1**, **6.2** y **6.3**.

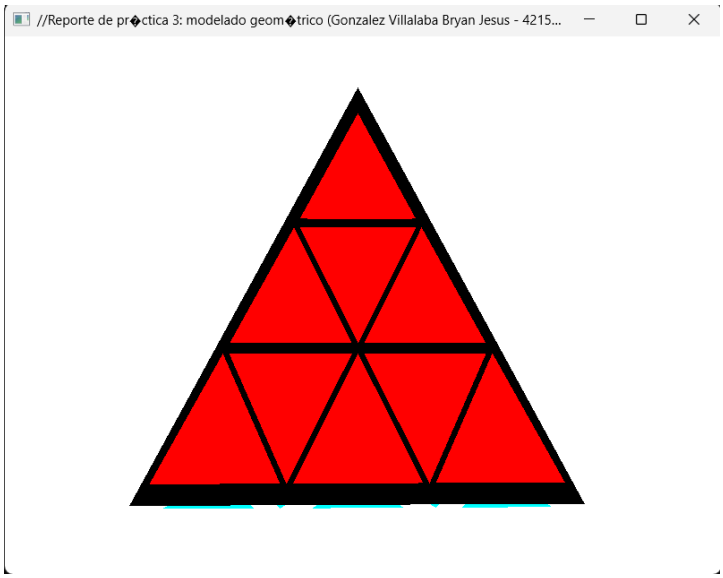

Imagen 6.0

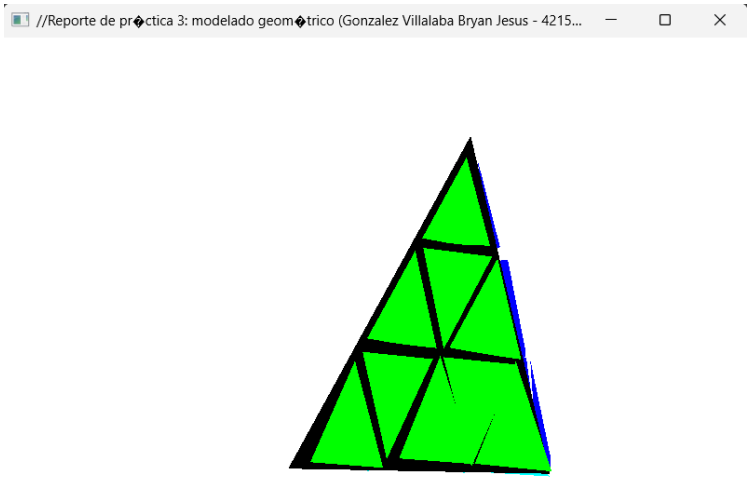

Imagen 6.1

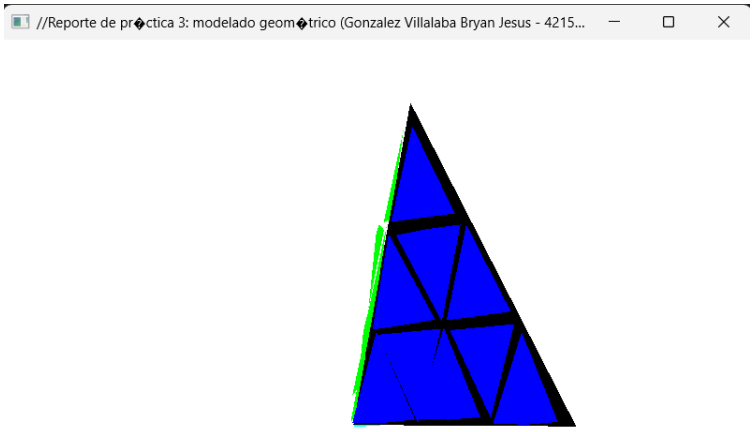

Imagen 6.2

Imagen 6.3

Conclusión:

Fue una práctica muy difícil sobre todo en las caras de color verde y color azul, ya que a mi parecer la pirámide no media lo mismo y eso me dificulto mucho el hecho de realizar bien la rotación de las piezas, en el dibujo de las caras color rojo y celeste me lleve menos de 20 minutos por cada cara, ya que fueron sencillas por la forma en la que estaban, pero en las caras que se me dificultaron me rendir, me lleve más de 6 horas tratando de dibujar las caras bien, pero no me quedaron como esperaba, el problema es que tengo que modificar el eje Z y Y de forma que el triángulo quede alineado a la pirámide, y el hecho de que es color negro me dificulto mas el trabajo, no pensé en cambiar el color, hasta ahorita que estoy haciendo el reporte, la parte más compleja de todo son las rotaciones y la escala de los lados que no veo de manera uniforme.