



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 N° 03

NOMBRE COMPLETO: Gabriel Patricio Balam Flores

N° de Cuenta: 320280324

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 07/sept

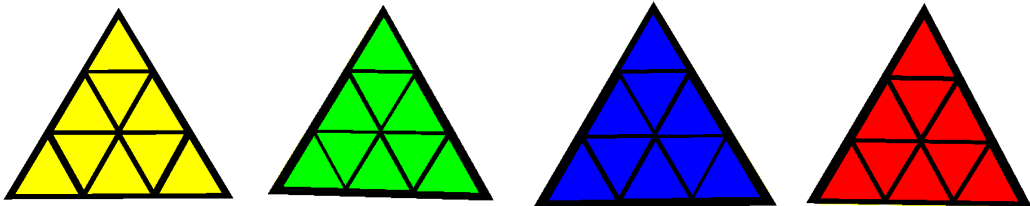
CALIFICACIÓN: _____

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

En esta práctica teníamos que modelar un Piraminx utilizando los modelos previamente creados. A primera vista, uno puede llegar a subestimar el ejercicio; sin embargo, al ponerse manos a la obra se percibe el verdadero reto que supone. Calcular las coordenadas exactas donde debe ir cada tetraedro resulta complicado debido a los ángulos y las distancias. Por ello, lo más viable es ajustar las pirámides de manera iterativa hasta lograr que encajen lo mejor posible.

Ejecución del ejercicio

Bloques de código del ejercicio 1
<pre>// BASE: piramide triangular negra ===== model = glm::mat4(1.0); model = glm::translate(model, glm::vec3(0.0f, 0.0f, 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, 0.0f, 0.0f); glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos meshList[1]->RenderMesh();</pre>

Para la cara roja

```
// LADO ROJO: =====
// Piramide 1 ( punta arriba )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, -0.01f, 0.2f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
//
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();

// Piramide 2 ( medio derecha )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.15f, -0.01f, -0.0725f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
//
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();

// Piramide 3 ( medio izquierda )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-0.15f, -0.01f, -0.0725f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
//
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();

// Piramide 4 ( abajo derecha )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.3f, -0.01f, -0.34f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
//
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();

// Piramide 5 ( abajo medio )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, -0.01f, -0.34f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
//
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();

// Piramide 6 ( abajo izquierda )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-0.3f, -0.01f, -0.34f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
//
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();

// Piramide 7 ( medio medio )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, -0.01f, -0.09f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
model = glm::rotate(model, 180 * toRadians, 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, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]-->RenderMesh();

// Piramide 8 ( Abajo medio derecha )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.15f, -0.01f, -0.36f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
model = glm::rotate(model, 180 * toRadians, 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, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]-->RenderMesh();

// Piramide 9 ( Abajo medio izquierda )
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-0.15f, -0.01f, -0.36f));
model = glm::scale(model, glm::vec3(0.27f, 0.27f, 0.27f));
model = glm::rotate(model, 180 * toRadians, 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, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[1]-->RenderMesh();
```

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.

Contratiempos

El primer contratiempo que encontré fue la correcta creación del tetraedro base. Después de un rato trabajando, me di cuenta de que, si desde el inicio se construye bien el tetraedro, se pueden facilitar muchos pasos posteriores.

El principal problema fue el tiempo que requiere posicionar adecuadamente los triángulos de cada cara. En mi caso, gracias a la correcta orientación del modelo base, pude espejear algunos triángulos y

ahorrar bastante tiempo; sin embargo, con el método que utilicé resulta casi imposible ubicar todos los triángulos con precisión.

Otro contratiempo previsible surgió con la cámara: al ejecutar el programa no se tiene claridad sobre en qué eje se está trabajando ni en qué posición exacta se encuentra uno. Para resolverlo, lo primero que hice fue crear unos ejes improvisados con rectángulos de distintos colores que me sirvieran como guía visual.

3.- Conclusión:

- a. Los ejercicios del reporte: Complejidad, Explicación.
 - b. Comentarios generales.
 - c. Conclusión
1. Bibliografía en formato APA

Conclusión

Entiendo el objetivo de la práctica al enfrentarnos directamente a problemas de rotación, traslación y escalado. Sin embargo, la verdadera complejidad no radica tanto en aplicar estas transformaciones, sino en el tiempo y la paciencia que requiere posicionar correctamente los triángulos.

En un momento de la práctica, mi TOC no me permitió continuar con el procedimiento que llevaba y empecé a intentar calcular matemáticamente las coordenadas. No obstante, debido a mi poca experiencia en esta área, hacerlo de esa manera me habría tomado más tiempo que resolverlo “al tanteo”. Una alternativa más práctica fue apoyarme en GeoGebra para obtener aproximaciones y calcular algunos ángulos.

En general, el resultado fue mejor de lo que esperaba: la forma es comprensible y logré mejorar mis habilidades con las herramientas de escalado, rotación y traslación.

Bibliografía

- Página principal. (26 de septiembre de 2018). *Wiki de OpenGL* . Recuperado el 29 de agosto de 2025 de http://www.khronos.org/opengl/wiki_opengl/index.php?title=Main_Page&oldid=14430 .