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: Medrano Miranda Daniel Ulises

Nº de Cuenta: 318045351

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 01/Septiembre/2024

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.

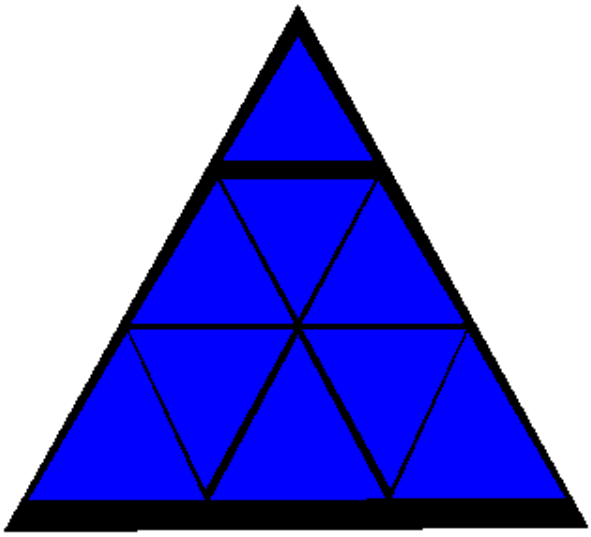
1.- Generar una pirámide rubik (pyraminx) de 9 pirámides por cara. Cada cara de la pyraminx que se vea de un color diferente y que se vean las separaciones entre instancias (las líneas oscuras son las que permiten diferenciar cada pirámide pequeña).

Para la realización de esta práctica se utilizaron pirámides triangulares y las ecuaciones de transformación.

```
575         //          - - - EJERCICIO DE PRÁCTICA - - -
576                 // PIRAMIDE NEGRO BASE
577                 model = glm::mat4(1.0);
578                 //Traslación inicial para posicionar en -Z a los objetos
579                 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -2.0f));
580                 //otras transformaciones para el objeto
581                 //model = glm::scale(model, glm::vec3(0.0f,0.0f,0.0f));
582                 //model = glm::scale(model, glm::vec3(1.5f,1.5f,1.5f));
583                 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
584                 //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
585                 //se programe cambio entre proyección ortogonal y perspectiva
586                 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
587                 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
588                 color = glm::vec3(0.0f, 0.0f, 0.0f);
589                 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
590                 meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
591                 //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
592                 //sp.render(); //dibuja esfera
593
594                 //PIRAMIDE AZUL CARA 1 - 1
595                 model = glm::mat4(1.0f);
596                 color = glm::vec3(0.0f, 0.0f, 1.0f);
597                 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
598                 model = glm::translate(model, glm::vec3(0.0f, 0.3f, -2.34f));
599                 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
600                 model = glm::rotate(model, 10 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
601                 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
602                 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
603                 meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
604                 //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
605                 //sp.render(); //dibuja esfera
606
607                 //PIRAMIDE AZUL CARA 1 - 2
608                 model = glm::mat4(1.0f);
609                 color = glm::vec3(0.0f, 0.0f, 1.0f);
610                 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
611                 model = glm::translate(model, glm::vec3(0.0f, 0.05f, -2.2f));
612                 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
613                 model = glm::rotate(model, 65 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
614                 model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
615                 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
616                 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
617                 meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
618                 //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
619                 //sp.render(); //dibuja esfera
620
```

```
621             //PIRAMIDE AZUL CARA 1 - 3
622             model = glm::mat4(1.0f);
623             color = glm::vec3(0.0f, 0.0f, 1.0f);
624             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
625             model = glm::translate(model, glm::vec3(0.0f, -0.3f, -2.02f));
626             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
627             model = glm::rotate(model, 10 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
628             //model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
629             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
630             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
631             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
632             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
633             //sp.render(); //dibuja esfera
634
635             //PIRAMIDE AZUL CARA 1 - 4
636             model = glm::mat4(1.0f);
637             color = glm::vec3(0.0f, 0.0f, 1.0f);
638             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
639             model = glm::translate(model, glm::vec3(-0.31f, -0.3f, -2.02f));
640             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
641             model = glm::rotate(model, 8 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
642             //model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
643             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
644             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
645             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
646             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
647             //sp.render(); //dibuja esfera
648
649             //PIRAMIDE AZUL CARA 1 - 5
650             model = glm::mat4(1.0f);
651             color = glm::vec3(0.0f, 0.0f, 1.0f);
652             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
653             model = glm::translate(model, glm::vec3(0.31f, -0.3f, -2.02f));
654             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
655             model = glm::rotate(model, 8 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
656             //model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
657             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
658             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
659             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
660             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
661             //sp.render(); //dibuja esfera
662
663             //PIRAMIDE AZUL CARA 1 - 6
664             model = glm::mat4(1.0f);
665             color = glm::vec3(0.0f, 0.0f, 1.0f);
666             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
667             model = glm::translate(model, glm::vec3(0.1575f, -0.01f, -2.18f));
668             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
669             model = glm::rotate(model, 10 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
670             //model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
671             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
672             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
673             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
674             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
675             //sp.render(); //dibuja esfera
676
```

```
677             //PIRAMIDE AZUL CARA 1 - 7
678             model = glm::mat4(1.0f);
679             color = glm::vec3(0.0f, 0.0f, 1.0f);
680             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
681             model = glm::translate(model, glm::vec3(-0.1575f, -0.01f, -2.18f));
682             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
683             model = glm::rotate(model, 10 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
684             //model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
685             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
686             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
687             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
688             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
689             //sp.render(); //dibuja esfera
690
691             //PIRAMIDE AZUL CARA 1 - 8
692             model = glm::mat4(1.0f);
693             color = glm::vec3(0.0f, 0.0f, 1.0f);
694             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
695             model = glm::translate(model, glm::vec3(-0.1575f, -0.23f, -2.04f));
696             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
697             model = glm::rotate(model, 65 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
698             model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
699             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
700             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
701             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
702             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
703             //sp.render(); //dibuja esfera
704
705             //PIRAMIDE AZUL CARA 1 - 9
706             model = glm::mat4(1.0f);
707             color = glm::vec3(0.0f, 0.0f, 1.0f);
708             //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo pu
709             model = glm::translate(model, glm::vec3(0.1575f, -0.23f, -2.04f));
710             model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.2f));
711             model = glm::rotate(model, 65 * toRadians, glm::vec3(-1.0f, 0.0f, 0.0f));
712             model = glm::rotate(model, 180 * toRadians, glm::vec3(-0.0f, 0.0f, 1.0f));
713             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SE
714             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
715             meshList[1]->RenderMesh(); //dibuja cubo y pirámide triangular
716             //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
717             //sp.render(); //dibuja esfera
718
```



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

En esta ocasión no logré finalizar la práctica como se esperaba, esto porque supe utilizar las funciones de transformación para las demás caras del Piraminx, Cada que intentaba hacer alguna traslación y rotación las pirámides pequeñas no se veían bien, por lo tanto, decidí dejar la práctica incompleta.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

En esta ocasión la complejidad ya demasiada para mí, antes se utilizaban figuras más sencillas y las rotaciones eran de múltiplos de 90°, en esta ocasión utilizar pirámides me pareció muy complicado.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

El profesor explicó correctamente que se debía hacer, el ejercicio de clase pude lograrlo, pero al usar pirámides mi “mundo se volvió abajo” porque no pude.

c. Conclusión

En esta ocasión no me siento satisfecho con mi desempeño debido a que no logré la realización de la práctica, el utilizar una figura como la pirámide nos obliga a analizar las transformaciones de diferente manera, aún cuando trataba de utilizar un análisis geométrico al tratar de plasmarlo en OpenGL no encontraba la forma.