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: BOLAÑOS GUERRERO JULIAN

Nº de Cuenta: 319157293

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

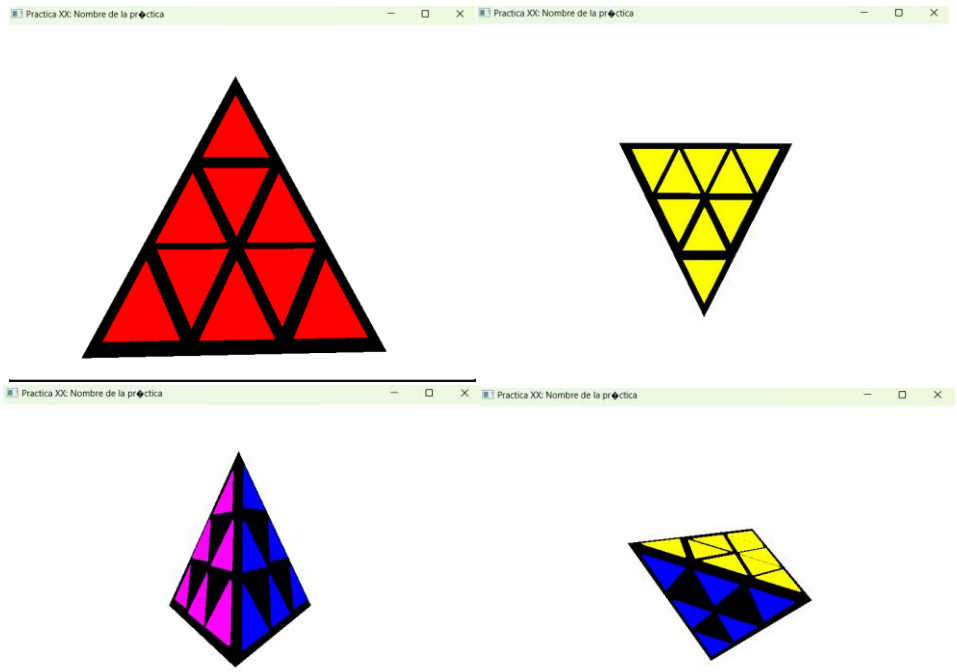
SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 05 – 03 – 25

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.



Cómo se puede observar, mi ejercicio está incompleto debido a varias razones la cuales, pues no justifican nada, pero considero que son importantes que las conozca porque realmente sí lo intenté.

Al inicio, lo primero que intente fue reutilizar pirámides, por ejemplo, la pirámide de la punta fácilmente podía ser una sola, pero estaba el reto de lograr que cada cara fuera de cierta manera independiente para que tuviera un color distinto al resto, está implementación me quito mucho tiempo lograrla, sin embargo, el siguiente reto era analizar la forma en que se iban a mostrar las divisiones. Cuando colocaba la pirámide sobre la pirámide negra, ocurrían dos cosas, si era más pequeña (la punta) no se veía, y si era más grande tapaba por completo la pirámide negra lo que ocultaba las “divisiones” u orillas. Lo siguiente que intente fue que, a cada mini pirámide le marcara sus aristas y poder controlar el grosor de estas, así solo tendría que acomodar las pirámides y ahorraría varias instancias, sin embargo no pude conseguir marcar líneas lo suficientemente gruesas para simular el espacio entre cada pirámide y esto también me consumió mucho tiempo, finalmente tuve que iniciar de nuevo pero ahora generando cada instancia de pirámide para cada cara

(esto lo evite desde el inicio porque usted nos pidió que por ejemplo, en las orillas, la figura podía compartir o ser la misma pirámide como en el caso de la punta, y que teníamos que ver cómo cambiar cada cara de un color diferente, etc.), sin embargo pues ya no logré terminar el ejercicio de esta manera. En la parte de código también pongo el código que intente y al final no resulto, igual agrego los archivos que cambie.

Código

Para el ejercicio de la práctica, en general este es el código creado, al tener ya los vértices de las pirámides, lo más importante fue crear instancias y aplicarles transformaciones para acomodarlas en la pirámide negra (la base negra para generar las líneas de división). Esto se repite a lo largo de toda la práctica. Solo varían los colores y las transformaciones.

```
// Cara roja

// Piramide de hasta arriba
model = glm::mat4(1.0f);
model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(-0.0f, 5.8f, -3.15f));
model = glm::scale(model, glm::vec3(sizeLittlePiramid, sizeLittlePiramid, sizeLittlePiramid));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.0f); // Color rojo
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

A continuación, también coloco aquí, parte del código de mis otras soluciones (donde logró poner cada cara de la pirámide de un color y donde solo dibujo las aristas).

```
// PIRAMIDE DE COLORES
shaderList[2].useShader();
uniformModel = shaderList[2].getModelLocation();
uniformProjection = shaderList[2].getProjectLocation();
uniformView = shaderList[2].getViewLocation();
uniformColor = shaderList[2].getColorLocation();

model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(15.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(5.0f, 10.0f, 5.0f));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

//meshList[4]->RenderMeshGeometry();


// ARISTAS SOLAMENTE
model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(40.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(5.0f, 10.0f, 5.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

//meshList[4]->RenderMeshGeometryAristas();
```

Para estas implementaciones tuve que modificar o crear algunos métodos de la clase Mesh, así como crear otro shader. Estos archivos los adjunto en classroom. En la parte de main, utilice parte del código para crear un cono que ya estaba, para modificarlo de modo que por cada vértice pudiera mandarle un color diferente para cada cara, un poco similar a lo que hicimos en la práctica de colocar nuestras iniciales con colores distintos, aquí muestro parte de la implementación.

```
// Para generear piramide con diferentes colores en las caras
void CrearCono(int res, float R, bool multicolor) {
    vector<GLfloat> vertices;
    vector<unsigned int> indices;

    GLfloat dt = 2 * PI / res, x, z, y = -0.5f;

    // Definir colores diferentes para cada cara
    vector<glm::vec3> faceColors = {
        {1.0f, 0.0f, 0.0f}, // Rojo
        {0.0f, 1.0f, 0.0f}, // Verde
        {0.0f, 0.0f, 1.0f}, // Azul
        {1.0f, 1.0f, 0.0f}  // Amarillo
    };

    if (!multicolor) {
        faceColors = { {0.0f,0.0f,0.0f} };
    }

    // Crear las caras laterales
    for (int n = 0; n < res; n++) {
        x = R * cos(n * dt);
        z = R * sin(n * dt);
        GLfloat xNext = R * cos((n + 1) * dt);
        GLfloat zNext = R * sin((n + 1) * dt);

        // Elegir un color único para la cara
        glm::vec3 color = faceColors[n % faceColors.size()];

        // Vértice superior (duplicado para cada cara)
        // Usamos la misma posición (0.0, 0.5, 0.0) pero como vértice único por cara
        vertices.insert(vertices.end(), { 0.0, 0.5, 0.0, color.x, color.y, color.z });

        // Vértice base 1
        vertices.insert(vertices.end(), { x, y, z, color.x, color.y, color.z });

        // Vértice base 2
        vertices.insert(vertices.end(), { xNext, y, zNext, color.x, color.y, color.z });

        // Añadir los índices de la cara
        unsigned int index = n * 3; // El vértice superior será siempre el primero
        indices.insert(indices.end(), { index, index + 1, index + 2 });
    }

    // Crear la base del cono (todos los vértices de la base tienen el mismo color)
    glm::vec3 baseColor = { 0.5f, 0.5f, 0.5f }; // Gris
    for (int n = 0; n < res; n++) {
        x = R * cos(n * dt);
        z = R * sin(n * dt);
        GLfloat xNext = R * cos((n + 1) * dt);
        GLfloat zNext = R * sin((n + 1) * dt);

        // Vértice central de la base (todos tienen el mismo color)
        vertices.insert(vertices.end(), { 0.0, y, 0.0, baseColor.x, baseColor.y, baseColor.z });
```

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.

Como lo menciono en el punto anterior, quizás el inconveniente fue haber intentado varias soluciones, al final, de haber iniciado desde un inicio con la última implementación (repetir pirámides para cada cara y ajustarlas mediante transformaciones) no hubiera tenido ningún problema de gestión de tiempo.

3.- Conclusión:

a. Los ejercicios del reporte: En general, sin contar mis otros intentos, pienso que mover polígonos 3D no es cualquier cosa, requiere de habilidades espaciales, etc. Sin embargo la solución que no complete no tenía mucha complejidad pero si consume timpo.

b. Conclusión

El modelado puede resultar no complicado pero muy laborioso ya que consume mucho esfuerzo de tiempo, sin embargo, elegir una correcta solución desde un principio pude acelerar el desarrollo.

1. Bibliografía en formato APA

Utilice varios foros de programación, perdí las ligas de los post que veía en sitios como stackoveflow.