Instituto Tecnológico y Estudios Superiores de Monterrey

Jesús Javier Hernández Delgado A01658891

Profesor: Dr. Sergio Ruiz Loza

Materia: Diseño de videojuegos 1

Fecha de entrega: 12 de agosto del 2022

Teselación

1. **Plan para continuar elaborando el octaedro en Unity:** Hasta el momento, el programa dibuja las líneas en una sola cara del triángulo, teniendo 4 triángulos en la cara que funcionan para el teselado. Se usa la geometría y topología para crear todos los triángulos en las caras de un dodecaedro, algo que por el momento solo permite tener una única cara.

El plan que se me ocurre para que se desarrolle correctamente el teselado en cada cara es que se continúe usando el mismo modelo de topología para hacer arrays de arrays y así obtener las 8 caras del dodecaedro mediante el uso de los puntos ya establecidos A, B, C, o, p y q.

- **Documentación.**

El modelado 3d se relaciona con los conceptos de geometría y topología; el icosaedro se conforma de triángulos interconectados que forman una figura. La geometría son los vértices y la topología se encarga de conectar esos vértices.

En las clases que se tuvieron estuvimos realizando el teselado de la figura; primero empezando por definir los vértices en un archivo .obj para las posiciones que iría teniendo. Luego los vectores normales que vendrían siendo las caras de los triángulos, y finalmente con la topología, definiendo el cómo conectar esos vértices. Luego de que se definió todo, en Unity se realizó un código en donde se definen los vectores A, B, C, o, p y q, se define el teselado, las conexiones que tienen los vértices, encontrar el vértice central en las conexiones, y finalmente meshear todo para formar la figura final.

Al final, para que la figura tuviera textura, simplemente se agrega que se necesitará acceder a un archivo .mtl, y definir qué parte del archivo deberá usar para que se vea en la figura.

```
# Cube.obj
# Geometry (The vertices)
mtllib Cube.mtl
v -1 -1 1
v 1 -1 1
v 1 1 1
v -1 1 1
v 1 -1 -1
v 1 1 -1
v -1 -1 -1
v -1 1 -1
# Normal vectors
n 0 0 1 # FRONT
n 1 0 0 # RIGHT
n 0 0 -1 # BACK
n -1 0 0 # LEFT
n 0 1 0 # TOP
n 0 -1 0 # BOTTOM
# vt Coordenadas de textura
vt 0 0
vt 1 0
vt 1 1
vt 0 1
# Topology (How to connect the vertices)
# FRONT
usemtl ElCubo
f 1/1/1 2/2/1 3/3/1
f 1/1/1 3/3/1 4/4/1
# RIGHT
f 2/1/2 5/2/2 6/3/2
f 2/1/2 6/3/2 3/4/2
# BACK
f 5//3 7//3 8//3
f 5//3 8//3 6//3
# LEFT
```

Figura. Archivo obj. donde se definen los vértices, vectores normales y demás. Agregando la textura.

```
7   public class Octaedro : MonoBehaviour
8   {
9       struct Shape{
10          public List<Vector3> geometry;
11          public List<int> topology;
12      };
13
14      void Tessellate(Shape input) {
15          List<int> originalT = input.topology;
16          int limit = input.topology.Count;
17          for(int t=0; t < limit; t+=3) {
18
19          Debug.Log(t);
20          Debug.Log(input.topology.Count);
21          Vector3 A = input.geometry[input.topology[t+0]];
22          Vector3 B = input.geometry[input.topology[t+1]];
23          Vector3 C = input.geometry[input.topology[t+2]];
24          int ia = FindVertex(input.geometry, A);
25          int ib = FindVertex(input.geometry, B);
26          int ic = FindVertex(input.geometry, C);
27          Vector3 o = ((A+B)/2).normalized;
28          Vector3 p = ((B+C)/2).normalized;
29          Vector3 q = ((A+C)/2).normalized;
30          int io = FindVertex(input.geometry, o);
31          int ip = FindVertex(input.geometry, p);
32          int iq = FindVertex(input.geometry, q);
```

Figura. Parte del código realizado en un script de Unity.

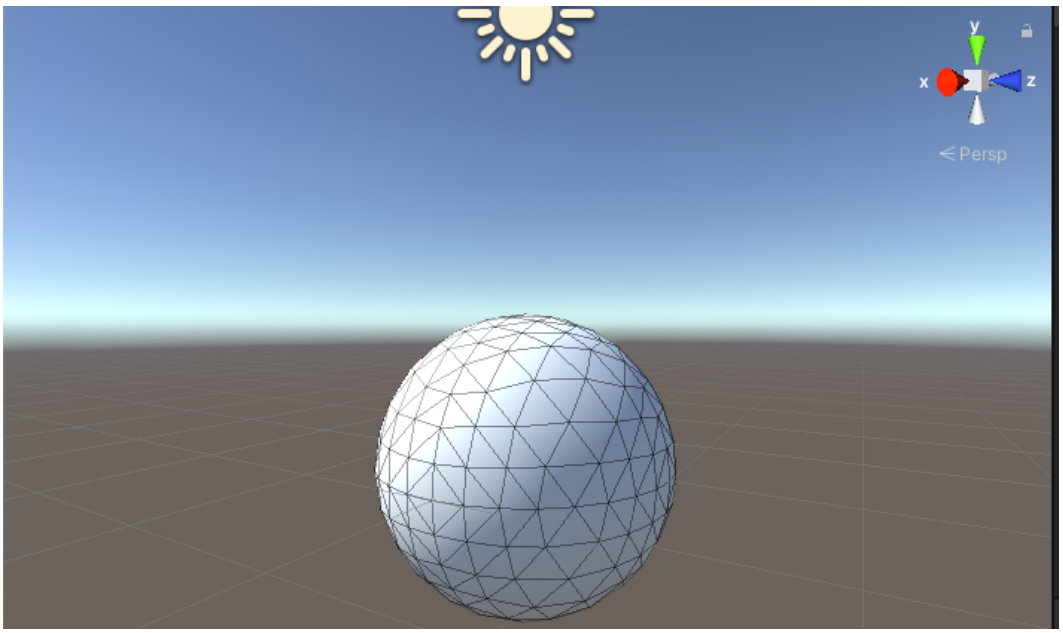

Figura. Dodecaedro final en Unity.