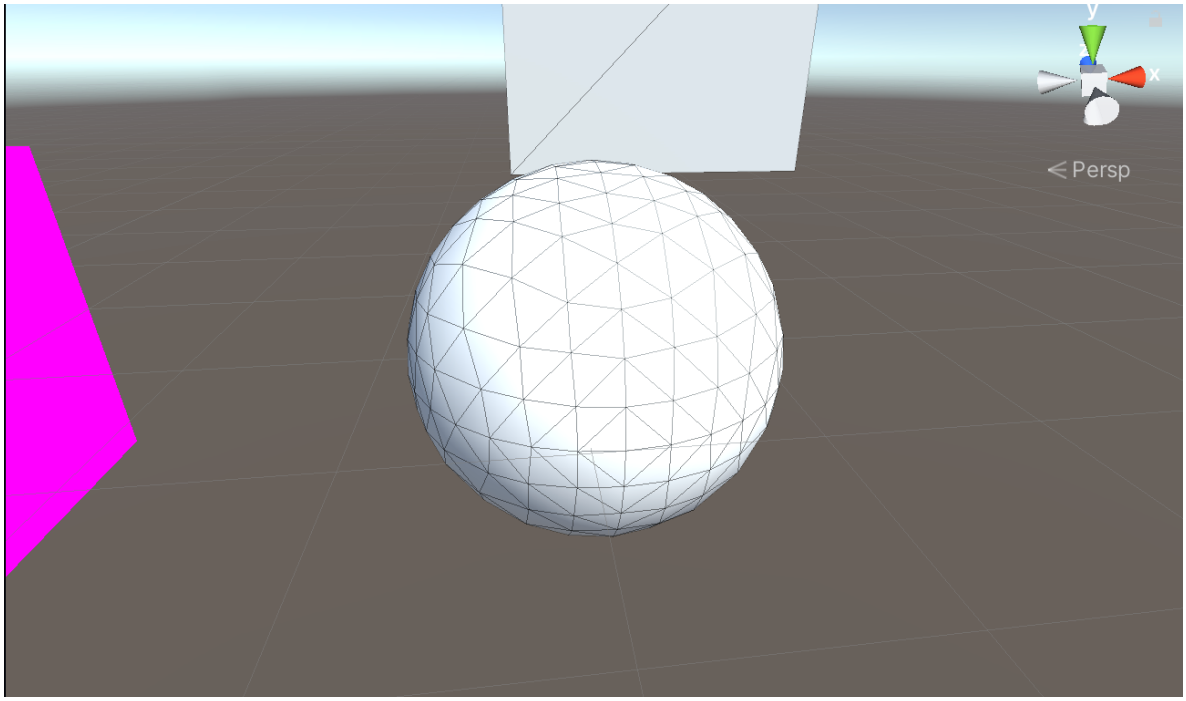
Dodecaedro

Daniel Garcia Barajas A01378688

Actualmente, en el script *Octaedro*, la función *tesellation* solo es capaz de trazar las lineas en medio de los puntos AB, BC y CA. De esta manera se traza un triangulo en una sola cara, generando cuatro triangulos en total dentro de la cara.

Para continuar con la funcion de *tesellation*, se podría iterar por la topología. En la iteración, por cada tres puntos hacer un array y así obtener una nueva topología con 8 arrays. Estos 8 arrays serían las 8 caras que después utilizaríamos la función *tesellation* para así obtner los puntos A, B, C, o, p, q.

- Resultado Final



Para obtener esta esfera, se tuvo que hacer teselación. Teselación se refiere a la división de las caras originales para así generar aún mas caras. Al inicio ya contabamos con un Octaedro, y buscamos teselarlo hasta obtener una esfera. Se creo una función *Tessellate* donde aquí itera por el numero de la topología del input de la función. En la función obtenemos sus puntos A, B, C y con estos los puntos o, p, q normalizados. De esta manera tenemos 12 caras nuevas de la cara y así una nueva topología. En la función Start, se llama la función *Tessellate* las veces necesarias hasta obtener la esfera.

```
void Tessellate(Shape input)
    {
        for(int t= 0; t < input.topology.Count; t+=12)
        {
            Vector3 A = input.geometry[input.topology[t+0]];
            Vector3 B = input.geometry[input.topology[t+1]];
            Vector3 C = input.geometry[input.topology[t+2]];
            Vector3 o = ((A+B)/2.0f).normalized;
            Vector3 p = ((B+C)/2.0f).normalized;
            Vector3 q = ((C+A)/2.0f).normalized;
            int ia = input.topology[t+0];
            int ib = input.topology[t+1];
            int ic = input.topology[t+2];
            int io = FindVertex(input.geometry, o);
            int ip = FindVertex(input.geometry, p);
            int iq = FindVertex(input.geometry, q);

            if(io == -1)
            {
                input.geometry.Add(o);
                io = input.geometry.Count-1;
            }
            if(ip == -1)
            {
                input.geometry.Add(p);
                ip = input.geometry.Count-1;
            }
            if(iq == -1)
            {
                input.geometry.Add(q);
                iq = input.geometry.Count-1;
            }

            List<int> newT = new List<int>();
            for(int i = 0; i < t; i++)
                newT.Add(input.topology[i]);
            newT.Add(ia); newT.Add(io); newT.Add(iq);
            newT.Add(io); newT.Add(ib); newT.Add(ip);
            newT.Add(iq); newT.Add(ip); newT.Add(ic);
            newT.Add(io); newT.Add(ip); newT.Add(iq);
            for(int i = t+3; i < input.topology.Count; i++)
                newT.Add(input.topology[i]);
            //Debug.Log(makeStringInt(input.topology));
            //Debug.Log(makeStringInt(newT));
            //Debug.Log(makeStringVector3(input.geometry));
            input.topology.Clear(); // Reemplazo la topología con la versión teselada.
            for(int i = 0; i < newT.Count; i++)
                input.topology.Add(newT[i]);
        }
    }
```