

## Taller Semana 14 - CG

Nicolás Bossio

El taller realizado se hizo con la ayuda de la herramienta ChatGPT, donde a continuación se comentarán y explicarán todos los procesos solicitados y utilizados para realizar el código.

1. Se le pregunto a ChatGPT como realizar un Dodecaedro en java, no obstante, tras multiples intentos el codigo no se podía ejecutar, o sencillamente estaba mal hecho, debido a que sus vertices se intersectan y no crean la figura deseada.

Haz un Dodecaedro 3D generado a partir de sus vértices, utilizando BufferGeometry

¡Por supuesto! Aquí te proporciono un ejemplo de cómo crear un dodecaedro 3D en Three.js utilizando BufferGeometry. Asumiremos que ya tienes instalada la librería Three.js y que tienes una escena y una cámara configuradas.

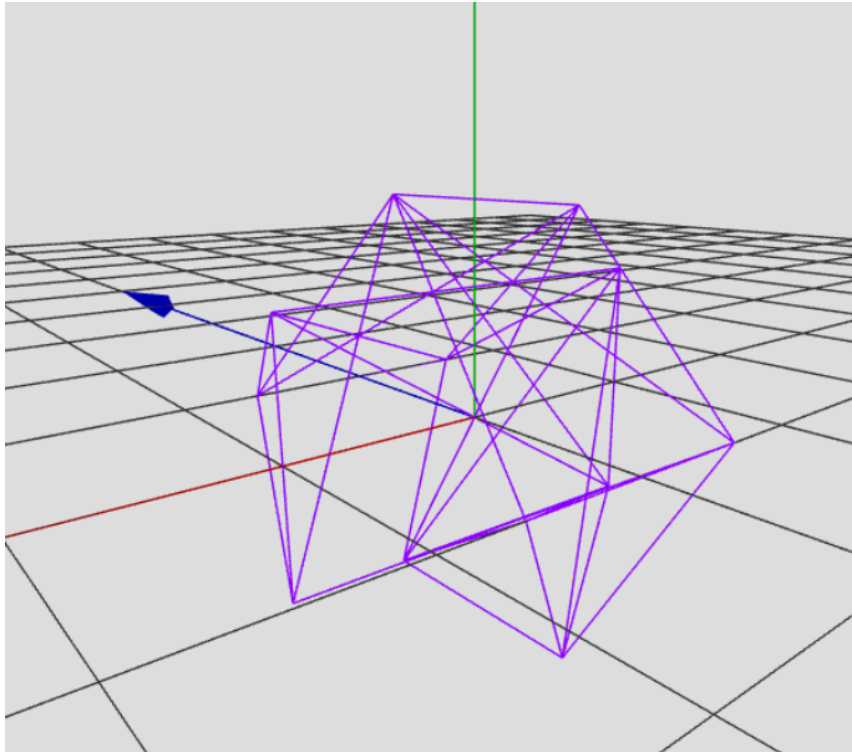
```
// Definimos la posición de los vértices del dodecaedro
const phi = (1 + Math.sqrt(5)) / 2; // Proporción dorada
const a = 1 / phi;
const b = 1;
const vertices = [
  // Grupo superior
  [0, b, a],
  [0, b, -a],
  [a, 0, b],
  [-a, 0, b],
  [b, a, 0],
  [-b, a, 0],
  // Grupo inferior
  [b, -a, 0],
  [-b, -a, 0],
  [a, 0, -b],
  [-a, 0, -b],
  [0, -b, a],
  [0, -b, -a]
];

// Definimos los índices de los triángulos que componen el dodecaedro
const indices = [
  0, 1, 2, 2, 1, 3, 2, 3, 4, 4, 3, 5, 0, 2, 6,
  0, 6, 7, 0, 7, 8, 0, 8, 9, 1, 10, 3, 3, 10, 5,
  5, 10, 11, 6, 2, 4, 9, 8, 11, 9, 11, 10, 7, 6, 4,
  7, 4, 5, 7, 5, 11, 8, 7, 10, 9, 10, 1
];

// Creamos la geometría y le pasamos los vértices e índices
const geometry = new THREE.BufferGeometry();
const vertexArray = new Float32Array(vertices.flat());
const indexArray = new Uint16Array(indices);
geometry.setAttribute('position', new THREE.BufferAttribute(vertexArray, 3));
geometry.setIndex(new THREE.BufferAttribute(indexArray, 1));

// Creamos el material y el objeto de malla
const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
const dodecahedron = new THREE.Mesh(geometry, material);

// Añadimos el objeto a la escena
scene.add(dodecahedron);
```



El primer intento tenía más o menos una similitud con el Dodecaedro que buscaba realizar, no obstante, aún le faltaba unir los puntos para que se asemejara a la verdadera figura.

Por esto, para la siguiente figura le pedí que me ordenara los vértices y los índices con el propósito de observar la figura del dodecaedro, esto se debe a que los vértices están correctos, sin embargo, están unidos de manera aleatoria.

NI

Puedes hacer que los vertices y indices esten ordenados?, el dodecaedro no se puede observar bien



¡Por supuesto! Los vértices y los índices están ordenados en el código anterior, pero es posible que necesites cambiar el orden para que el dodecaedro se vea correctamente. Aquí te proporciono una versión del código con los vértices y los índices ordenados para que el dodecaedro sea visible correctamente:

```

// Definimos la posición de los vértices del dodecaedro
const phi = (1 + Math.sqrt(5)) / 2; // Proporción dorada
const a = 1 / phi;
const b = 1;
const vertices = [
  // Grupo superior
  [a, b, 0],
  [-a, b, 0],
  [0, a, b],
  [0, a, -b],
  [b, 0, a],
  [-b, 0, a],
  // Grupo inferior
  [b, 0, -a],
  [-b, 0, -a],
  [0, -a, -b],
  [0, -a, b],
  [a, -b, 0],
  [-a, -b, 0]
];

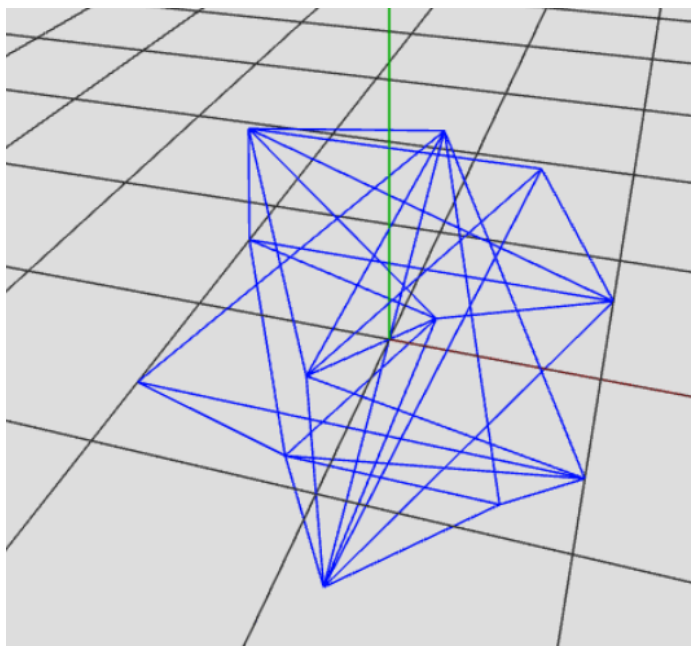
// Definimos los índices de los triángulos que componen el dodecaedro
const indices = [
  0, 1, 2, 3, 2, 1, 2, 3, 4, 5, 4, 3, 0, 6, 1,
  1, 6, 7, 1, 7, 8, 1, 8, 2, 9, 2, 8, 2, 9, 3, 10, 3, 9,
  3, 10, 4, 11, 4, 10, 4, 11, 5, 0, 2, 9, 6, 0, 9, 6, 9, 8,
  6, 8, 7, 11, 5, 4, 11, 4, 10, 11, 10, 9, 11, 9, 8, 11, 8, 7
];

// Creamos la geometría y le pasamos los vértices e índices
const geometry = new THREE.BufferGeometry();
const vertexArray = new Float32Array(vertices.flat());
const indexArray = new Uint16Array(indices);
geometry.setAttribute('position', new THREE.BufferAttribute(vertexArray, 3));
geometry.setIndex(new THREE.BufferAttribute(indexArray, 1));

// Creamos el material y el objeto de malla
const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
const dodecahedron = new THREE.Mesh(geometry, material);

// Añadimos el objeto a la escena
scene.add(dodecahedron);

```



Aunque la figura mejoró un poco, aun no se podía observar el dodecaedro que se quería obtener, siendo el segundo intento también fallido, por ende, decidí darle una última oportunidad antes de cambiar de figura.

Pasando al último intento, le pedí que hiciera los vértices uno a uno, instrucción la cual ignoro y solamente cambió algunos valores del código, observándose de la siguiente manera.

Sigue sin poder observarse, tal vez si realizas los vértices ordenado de uno en uno, se pueda observar mejor

Entiendo, podemos ordenar los vértices de uno en uno para que sea más fácil de ver. Aquí está una nueva versión del código con los vértices ordenados de uno en uno y los índices ajustados para que coincidan con el nuevo orden:

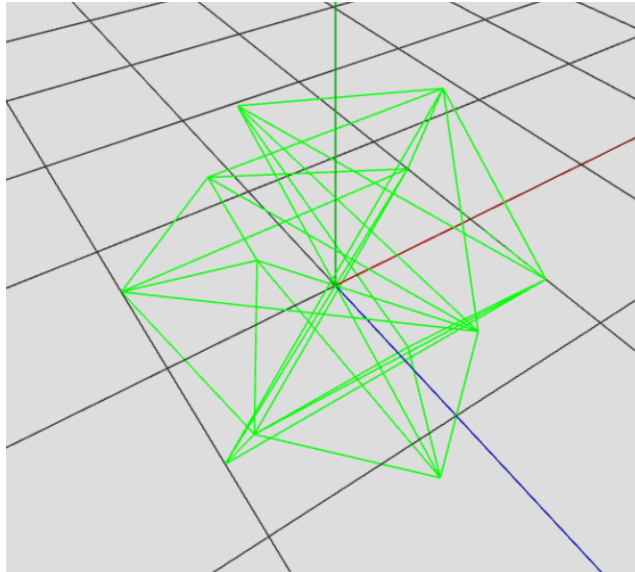
```
// Definimos la posición de los vértices del dodecaedro
const phi = (1 + Math.sqrt(5)) / 2; // Proporción dorada
const a = 1 / phi;
const b = 1;
const vertices = [
  // Grupo superior
  [a, b, 0],
  [-a, b, 0],
  [0, a, b],
  [0, a, -b],
  [b, 0, a],
  [-b, 0, a],
  // Grupo inferior
  [b, 0, -a],
  [-b, 0, -a],
  [0, -a, -b],
  [0, -a, b],
  [a, -b, 0],
  [-a, -b, 0]
].flat();

// Definimos los índices de los triángulos que componen el dodecaedro
const indices = [
  0, 1, 2, 2, 3, 0, 0, 3, 4, 4, 5, 0, 1, 6, 7, 1, 7, 8, 1, 8, 2,
  2, 8, 9, 2, 9, 3, 3, 9, 10, 3, 10, 4, 4, 10, 11, 4, 11, 5, 5, 11, 6,
  5, 6, 0, 6, 11, 7, 7, 11, 8, 8, 11, 9, 9, 11, 10, 1, 0, 6, 2, 1, 7,
  2, 7, 8, 2, 8, 9, 3, 2, 9, 4, 3, 10, 5, 4, 11, 6, 5, 0, 4, 6, 11
];

// Creamos la geometría y le pasamos los vértices e índices
const geometry = new THREE.BufferGeometry();
geometry.setAttribute('position', new THREE.BufferAttribute(new Float32Array(vertices), 1));
geometry.setIndex(new THREE.BufferAttribute(new Uint16Array(indices), 1));

// Creamos el material y el objeto de malla
const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
const dodecahedron = new THREE.Mesh(geometry, material);

// Añadimos el objeto a la escena
scene.add(dodecahedron);
```



Tras obtener este resultado fallido, decidió probar con otras figuras, aunque no lo ponga aca, intente otras 7 variantes pero el resultado no cambiaba, por ende, decidí probar con otra figura al azar que sencillamente tuviera más de 10 lados.

En este caso me dio un icosaedro, el cual está perfectamente hecho, por lo tanto sabía que era posible realizar el dodecaedro, sin embargo, no pude obtenerlo

NI

Listo, Puedes hacer otra figura en 3D superior a 10 lado con BufferGeometry



¡Claro! Aquí te dejo un ejemplo de cómo crear un icosaedro con `'BufferGeometry'`:

```

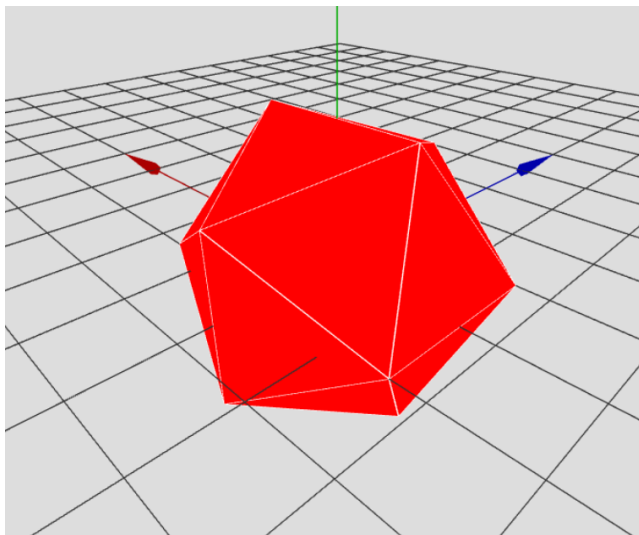
const phi = (1 + Math.sqrt(5)) / 2; // número áureo
const vertices = [
  // coordenadas de los vértices del icosaedro
  -1, phi, 0,
  1, phi, 0,
  -1, -phi, 0,
  1, -phi, 0,
  0, -1, phi,
  0, 1, phi,
  0, -1, -phi,
  0, 1, -phi,
  phi, 0, -1,
  phi, 0, 1,
  -phi, 0, -1,
  -phi, 0, 1,
];

const indices = [
  // índices de los triángulos que forman el icosaedro
  0, 11, 5,
  0, 5, 1,
  0, 1, 7,
  0, 7, 10,
  0, 10, 11,
  1, 5, 9,
  5, 11, 4,
  11, 10, 2,
  10, 7, 6,
  7, 1, 8,
  3, 9, 4,
  3, 4, 2,
  3, 2, 6,
  3, 6, 8,
  3, 8, 9,
  4, 9, 5,
  2, 4, 11,
  6, 2, 10,
  8, 6, 7,
  9, 8, 1,
];

const geometry = new THREE.BufferGeometry();
geometry.setAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));
geometry.setIndex(indices);

const material = new THREE.MeshBasicMaterial({ color: 0xff0000 });
const mesh = new THREE.Mesh(geometry, material);
scene.add(mesh);

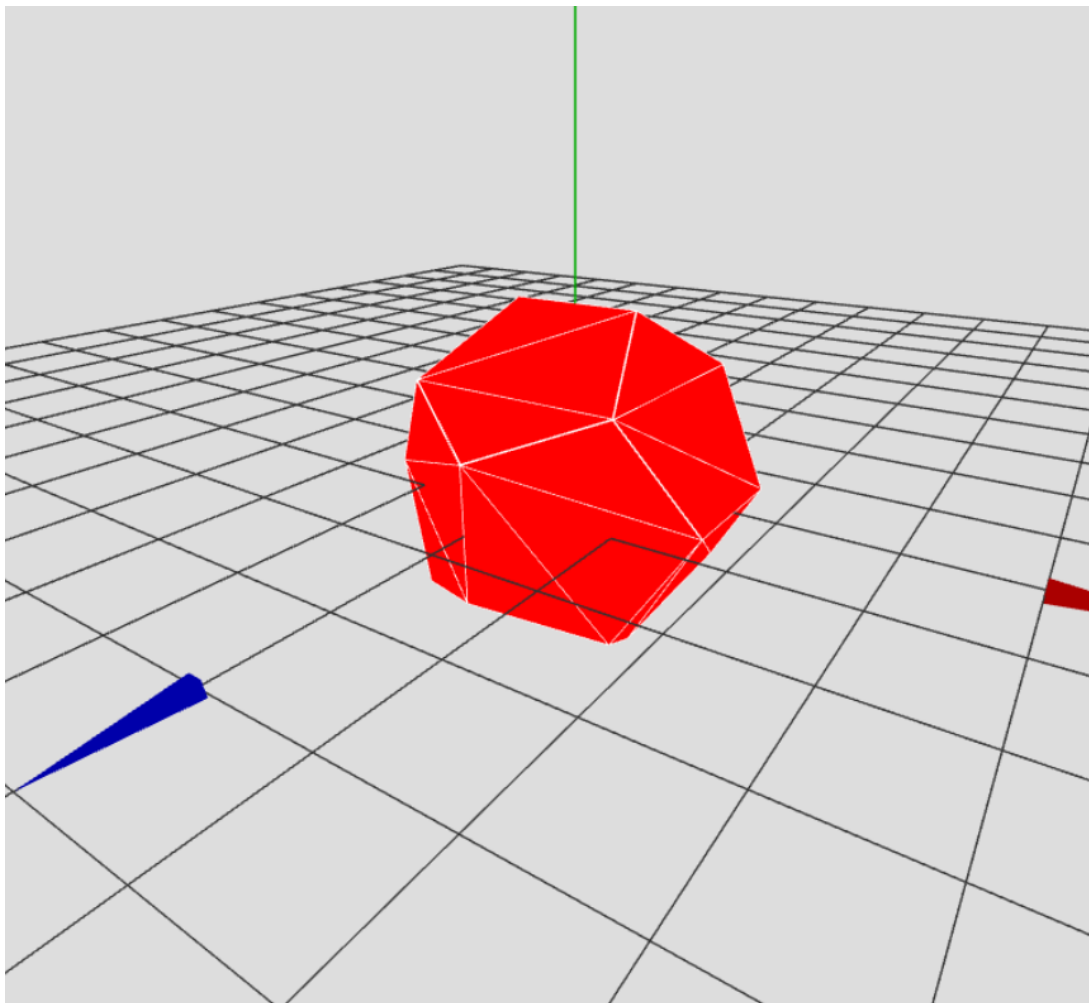
```



El Icosaedro lo hizo de manera perfecta sin problema alguno, ante mi duda seguí intentando hacer el dodecaedro sin éxito alguno, por ende, decidí ir a la página de Three.js para ver si era posible entender mejor cómo crear un dodecaedro, donde había un código, el cual tenía los vértices y las caras necesarias para realizarlo, seguido de eso, observe el código del icosaedro, el cual era exactamente igual al suministrado por ChatGPT.

Teniendo en cuenta esto, me guíé por los vértices suministrados por Three.js mientras que utilice el resto de código que me sugirió ChatGPT, de esta manera, fue posible hacer el dodecaedro, cabe resaltar que el código utilizado para los vértices estará referenciado al final del documento.

Tras realizar esos ajustes, el cubo se ve de la siguiente manera.



## Referencias

1. <https://github.com/mrdoob/three.js/blob/master/src/geometries/DodecahedronGeometry.js>
2. <https://github.com/mrdoob/three.js/blob/master/src/geometries/IcosahedronGeometry.js>

3. <https://threejs.org/docs/#api/en/geometries/DodecahedronGeometry>