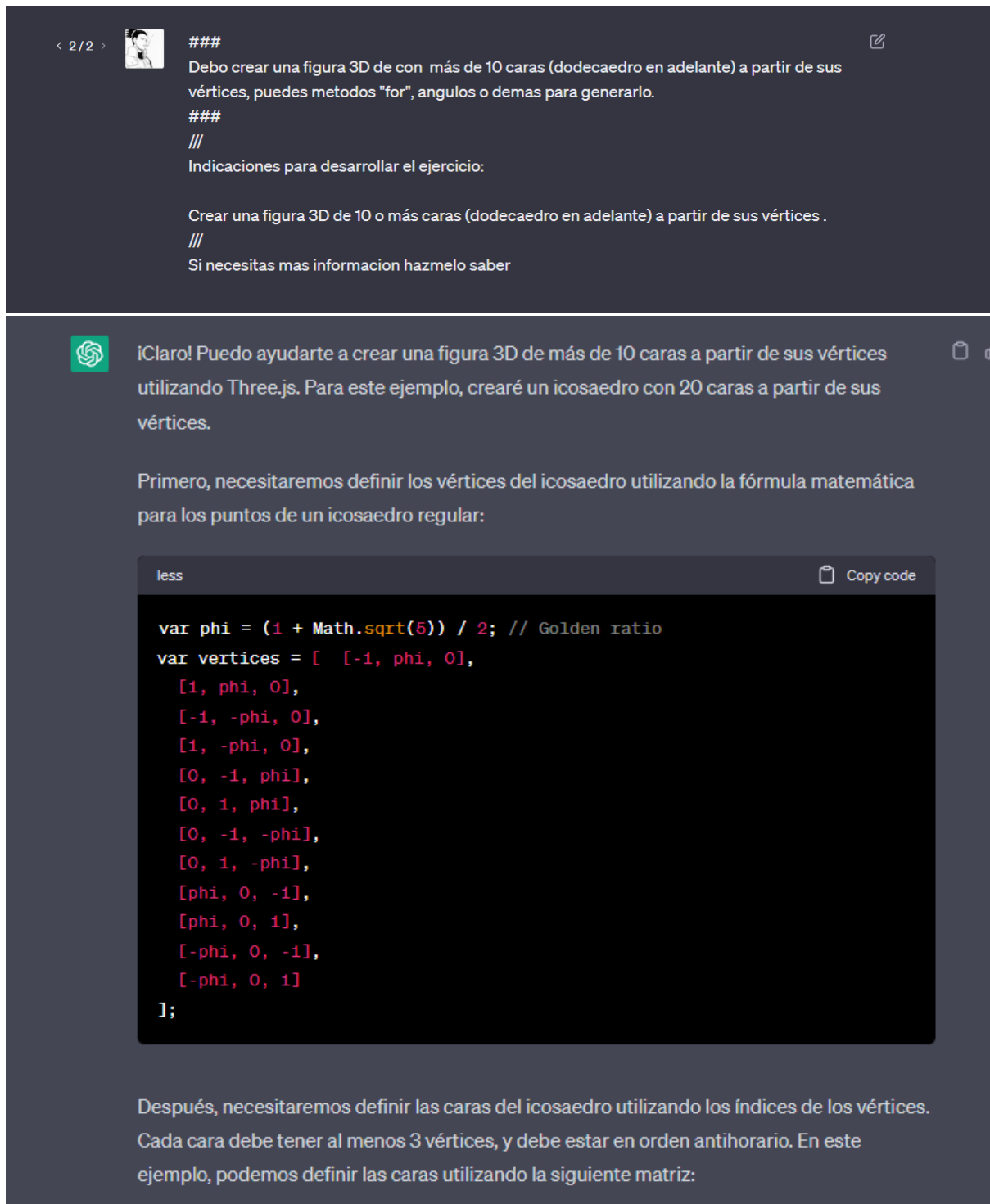


A lo largo del presente documento mostrarán las preguntas realizadas a la inteligencia artificial “ChatGPT” para generar una figura 3D de 10 o más caras (dodecaedro en adelante) a partir de sus vértices. Cabe resaltar que solo se utilizó la IA para generar la figura, la creación de la escena se realizó con base en los códigos antes subidos al github.

1. Pregunta realizada



ChatGPT conversation showing a user asking for help to create a 3D figure with more than 10 faces (dodecahedron or more) from its vertices, and the AI responding with a solution using Three.js and a list of vertices for an icosahedron.

User prompt: Debo crear una figura 3D de con más de 10 caras (dodecaedro en adelante) a partir de sus vértices, puedes metodos "for", angulos o demas para generarlo.

AI response: ¡Claro! Puedo ayudarte a crear una figura 3D de más de 10 caras a partir de sus vértices utilizando Three.js. Para este ejemplo, crearé un icosaedro con 20 caras a partir de sus vértices.

Primero, necesitaremos definir los vértices del icosaedro utilizando la fórmula matemática para los puntos de un icosaedro regular:

```
var phi = (1 + Math.sqrt(5)) / 2; // Golden ratio
var vertices = [ [-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]
];
```

Después, necesitaremos definir las caras del icosaedro utilizando los índices de los vértices. Cada cara debe tener al menos 3 vértices, y debe estar en orden antihorario. En este ejemplo, podemos definir las caras utilizando la siguiente matriz:

Después, necesitaremos definir las caras del icosaedro utilizando los índices de los vértices. Cada cara debe tener al menos 3 vértices, y debe estar en orden antihorario. En este ejemplo, podemos definir las caras utilizando la siguiente matriz:

```
csharp Copy code

var faces = [
    [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]
];
```

Ahora podemos utilizar Three.js para crear la figura 3D a partir de los vértices y caras definidos anteriormente:

```
css Copy code

var geometry = new THREE.Geometry();

// Agregar vértices
for (var i = 0; i < vertices.length; i++) {
    var vertex = new THREE.Vector3().fromArray(vertices[i]);
    geometry.vertices.push(vertex);
}

// Agregar caras
for (var i = 0; i < faces.length; i++) {
    var face = new THREE.Face3(faces[i][0], faces[i][1], faces[i][2]);
    geometry.faces.push(face);
}

// Crear malla de la figura
var material = new THREE.MeshBasicMaterial({color: 0xffffff, wireframe: true});
var mesh = new THREE.Mesh(geometry, material);

// Agregar la malla a la escena
scene.add(mesh);
```

Este código creará un icosaedro con 20 caras, utilizando los vértices y caras definidos anteriormente. La malla se agregará a la escena y se renderizará en la pantalla. Puedes ajustar el color y los materiales según tus necesidades. Esper

2. Una vez está tomada esta geometría se realizaron los siguientes cambios:

- Se reescribe la variable “geometry” cómo “`var geometry = new THREE.BufferGeometry()`” ya que el método `THREE.Geometry` es erróneo.
- Luego de eso, el objeto no se generaba correctamente, por ende se le pidió a la inteligencia artificial revisar el código con la corrección antedicha y ya con esa respuesta se construyó el código anexado en este mismo repositorio.

< 2/2 >



El objeto no se genera correctamente, por favor revisa el código y realiza las correcciones apropiadas para graficar la geometría correctamente.

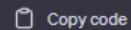


El objeto no se genera porque la geometría del objeto (`geometry`) no contiene la información necesaria para construir el objeto 3D. En este código, se ha creado una instancia de `THREE.BufferGeometry()`, pero no se ha agregado ninguna información a ella.



Para solucionar este problema, primero debes agregar los datos de la geometría de la malla, es decir, los vértices y las caras, a la geometría del buffer. Esto se puede hacer de varias maneras, por ejemplo, utilizando el método `setIndex()` y `setAttribute()` de la instancia de `THREE.BufferGeometry()`. Aquí te dejo un ejemplo que podrías utilizar:

php



```
// Crear geometría del buffer a partir de los datos de vértices y caras
geometry.setIndex(faces.flat());
geometry.setAttribute('position', new THREE.Float32BufferAttribute(vertices.

// Crear malla de la figura
var material = new THREE.MeshBasicMaterial({color: 0xffffff, wireframe: true
var mesh = new THREE.Mesh(geometry, material);
```

En este código, `faces.flat()` devuelve una matriz plana de índices de vértices para cada cara del objeto, y `vertices.flat()` devuelve una matriz plana de coordenadas de vértices en el espacio 3D. Luego, `geometry.setIndex()` y `geometry.setAttribute()` se utilizan para asignar estos datos a la geometría del buffer.

Una vez que la geometría del buffer contiene la información necesaria, puedes crear una malla a partir de ella y agregarla a la escena, como se hace en el código original. Con estos cambios, el objeto debería generarse correctamente.

3. Este fue el resultado final:

