

FIGURA 3D – ChatGPT

Omar Santiago Martínez Pachón

- Se realiza una figura 3D con mas de 10 caras con la ayuda de la IA – ChatGPT -, realizando la siguiente pregunta.



crear una figura 3D de 10 o más caras (dodecaedro en adelante) a partir de sus vértices.

Dándonos la siguiente respuestas (se le realizo la pregunta dos veces para analizar ambas respuestas):

Respuesta #1

```
Para crear una figura 3D de 10 o más caras a partir de sus vértices en Three.js, podemos seguir un proceso similar al utilizado en el ejemplo anterior. Aquí presento un ejemplo de cómo crear un dodecaedro (12 caras) a partir de sus vértices:
```

```
javascript Copiar código

// crear los vértices de un dodecaedro
var vertices = [
  // vertices de un icosaedro (20 caras)
  new THREE.Vector3(0, 1, PHI),
  new THREE.Vector3(0, -1, PHI),
  new THREE.Vector3(0, 1, -PHI),
  new THREE.Vector3(0, -1, -PHI),
  new THREE.Vector3(1, PHI, 0),
  new THREE.Vector3(-1, PHI, 0),
  new THREE.Vector3(1, -PHI, 0),
  new THREE.Vector3(-1, -PHI, 0),
  new THREE.Vector3(PHI, 0, 1),
  new THREE.Vector3(-PHI, 0, 1),
  new THREE.Vector3(PHI, 0, -1),
  new THREE.Vector3(-PHI, 0, -1),
];

// definir las caras del dodecaedro utilizando indices de vértices
var faces = [
  // caras de un icosaedro (20 caras)
  new THREE.Face3(0, 8, 4),
  new THREE.Face3(0, 5, 10),
  new THREE.Face3(2, 4, 9),
  new THREE.Face3(2, 11, 5),
  new THREE.Face3(1, 6, 3),
  new THREE.Face3(1, 10, 7),
  new THREE.Face3(3, 9, 6),
  new THREE.Face3(3, 7, 11),
  new THREE.Face3(0, 10, 8),
  new THREE.Face3(1, 8, 10),
  new THREE.Face3(2, 9, 11),
  new THREE.Face3(3, 6, 9),
  new THREE.Face3(4, 2, 0),
  new THREE.Face3(5, 0, 2),
  new THREE.Face3(6, 1, 3),
  new THREE.Face3(7, 3, 1),
  new THREE.Face3(8, 6, 4),
  new THREE.Face3(9, 4, 6),
  new THREE.Face3(10, 5, 7),
  new THREE.Face3(11, 7, 5),
];
```

```
// crear una geometría utilizando los vértices y las caras
var geometry = new THREE.Geometry();
geometry.vertices = vertices;
geometry.faces = faces;

// calcular las normales para las caras
geometry.computeFaceNormals();

// crear un material para la figura
var material = new THREE.MeshBasicMaterial({ color: 0xffffff, wireframe: true });

// crear una malla utilizando la geometría y el material
var mesh = new THREE.Mesh(geometry, material);

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

// renderizar la escena
renderer.render(scene, camera);
```

Respuesta #2



¡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:

```
less Copy code

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

- La IA nos dio la geometría, pero se realizaron los siguientes cambios dado que no funcionaban por lo cual se le pidió a la IA que corrigiera el error además de que se reescribió la variable “geometry” como “**var geometry = new THREE.BufferGeometry()**” ya que el método THREE.Geometry es erróneo y por esto no realizaba la construcción de la figura. Además, las IA con la corrección se agregó una parte de código.



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

Copy code

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

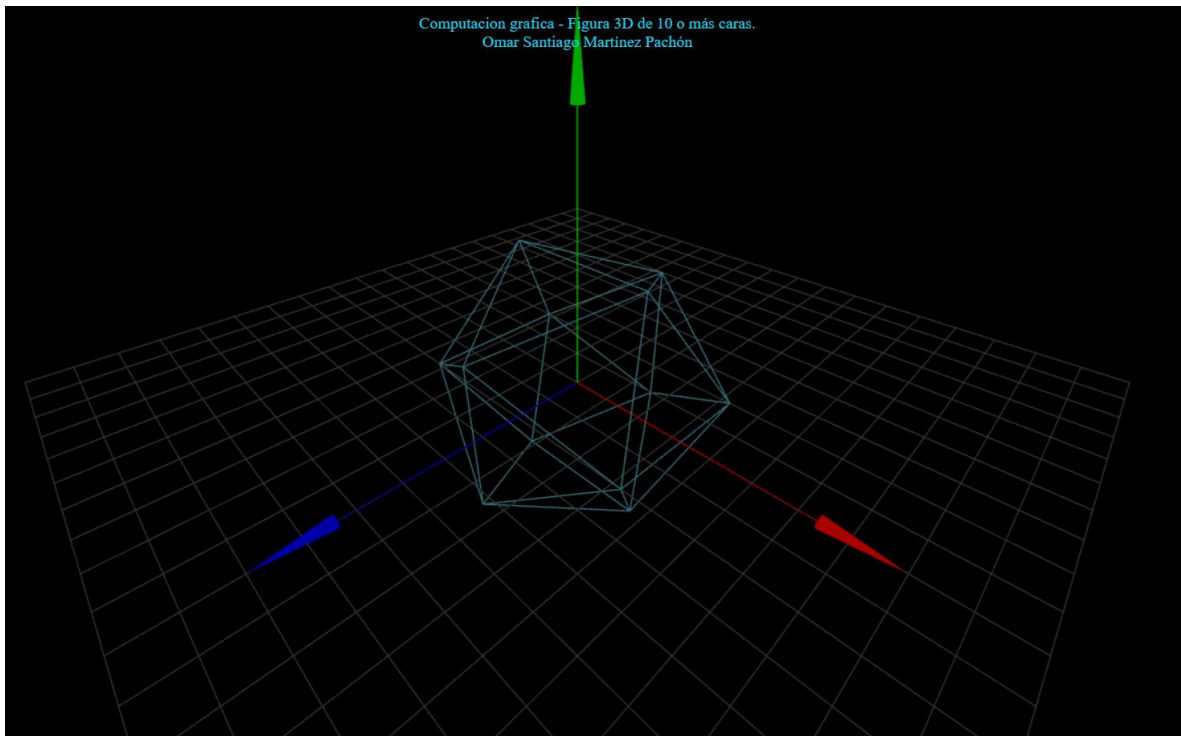
// Crear malla de la figura
var material = new THREE.MeshBasicMaterial({color: 0xffffff, wireframe: true});
var mesh = new THREE.Mesh(geometry, material);
```

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.

- En la primera respuesta nos dice que se realizara un dodecaedro (figura de 12 lados) aun asi nos da los vértices para un icosaedro (figura de 20 lado), además de que la figura no se generaba se le realizo las misma pregunta una segunda vez, lo que genero las segunda respuesta, dado esto con ambas respuestas y pedirle a la IA que corrigiera una parte se logro construir un icosaedro.

- Después todo esto, obtenemos un ICOSAEDRO (figura de 20 lados).



- Si la figura se deja solo con color como material, no se aprecia bien sus caras.

