Parcial 2 Computación Grafica

Ya contaba con un código que generaba una cubo en medio de la escena que giraba en lentamente en direcciones aleatorias, trabajare sobre ese código:

**CODIGO 1:**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cubo 3D con Three.js</title>

<style>

body { margin: 0; }

canvas { display: block; }

</style>

</head>

<body>

<script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>

<script>

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

const renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

const geometry = new THREE.BoxGeometry();

const materials = [

new THREE.MeshBasicMaterial({ color: 0xff0000 }),AC-03OCT

new THREE.MeshBasicMaterial({ color: 0x00ff00 }),

new THREE.MeshBasicMaterial({ color: 0x0000ff }),

new THREE.MeshBasicMaterial({ color: 0xffff00 }),

new THREE.MeshBasicMaterial({ color: 0xff00ff }),

new THREE.MeshBasicMaterial({ color: 0x00ffff })

];

const cube = new THREE.Mesh(geometry, materials);

scene.add(cube);

camera.position.z = 5;

function animate() {

requestAnimationFrame(animate);

cube.rotation.x += 0.01;

cube.rotation.y += 0.01;

renderer.render(scene, camera);

}

animate();

</script>

</body>

</html>

**Primera pregunta:**para empezar me interesa saber cómo agregar una cámara y poner la figura en modo mesh y solo ver sus bordes.

**Respuesta:**

**Explicación del código:**

1. **Three.js**: Cargamos la biblioteca desde un CDN.
2. **Escena**: Creamos una escena donde se agregarán los objetos.
3. **Cámara**: Usamos una cámara en perspectiva para ver los objetos en 3D.
4. **Renderizador**: Inicializamos el renderizador y lo añadimos al DOM.
5. **Geometría y Material**: Creamos un cubo y le aplicamos un material básico con wireframe: true para mostrar solo los bordes.
6. **Animación**: Creamos una función que actualiza la rotación del cubo y vuelve a renderizar la escena.

**Implementación:**

**CODIGO 2:**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Ejemplo de Three.js</title>

<style>

body { margin: 0; }

canvas { display: block; }

</style>

</head>

<body>

<script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>

<script>

// Configuración de la escena, cámara y renderizador

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

const renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

// Crear una geometría y un material

const geometry = new THREE.BoxGeometry(1, 1, 1); // Cubo como ejemplo

const material = new THREE.MeshBasicMaterial({ color: 0x00ff00, wireframe: true }); // Modo wireframe

// Crear la malla y agregarla a la escena

const cube = new THREE.Mesh(geometry, material);

scene.add(cube);

// Posicionar la cámara

camera.position.z = 5;

// Función de animación

function animate() {

requestAnimationFrame(animate);

cube.rotation.x += 0.01; // Rotación en el eje X

cube.rotation.y += 0.01; // Rotación en el eje Y

renderer.render(scene, camera);

}

animate();

</script>

</body>

</html>

**Segunda pregunta:**

Ahora me interesa saber cómo hacer para que cada vez que cargo la página se generen de entre 1 a 5 cubos alineados

**Respuesta:**

**Cambios realizados:**

1. **Función createCubes**: Se agregó una función que genera entre 1 y 5 cubos. Utiliza Math.random() para obtener un número aleatorio.
2. **Color Blanco**: Los cubos ahora se crean con el color blanco (0xffffff).
3. **Alineación**: Se ajusta la posición de cada cubo en el eje X para que estén alineados con un espaciado de 2 unidades.

**Cómo funciona:**

* Cada vez que recargas la página, la función createCubes generará un número aleatorio de cubos entre 1 y 5, alineados en el eje X.

**Implementación:**

**CODIGO 3:**<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Ejemplo de Three.js</title>

<style>

body { margin: 0; }

canvas { display: block; }

</style>

</head>

<body>

<script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>

<script>

// Configuración de la escena, cámara y renderizador

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

const renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

// Array para almacenar los cubos

const cubes = [];

// Crear una función para generar cubos

function createCubes() {

const numberOfCubes = Math.floor(Math.random() \* 5) + 1; // Generar un número aleatorio entre 1 y 5

const spacing = 2; // Espaciado entre cubos

for (let i = 0; i < numberOfCubes; i++) {

const geometry = new THREE.BoxGeometry(1, 1, 1); // Cubo

const material = new THREE.LineBasicMaterial({ color: 0xffffff }); // Color blanco para el wireframe

const wireframe = new THREE.LineSegments(new THREE.EdgesGeometry(geometry), material); // Crear la malla de bordes

// Posicionar cubos en línea

wireframe.position.x = i \* spacing; // Alinear cubos

scene.add(wireframe);

cubes.push(wireframe); // Guardar el cubo en el array

}

}

// Llamar a la función para crear cubos

createCubes();

// Posicionar la cámara

camera.position.z = 5;

// Función de animación

function animate() {

requestAnimationFrame(animate);

cubes.forEach(cube => {

cube.rotation.x += 0.01; // Rotación en el eje X

cube.rotation.y += 0.01; // Rotación en el eje Y

});

renderer.render(scene, camera);

}

animate();

</script>

</body>

</html>

Ya con este podría solucionar la actividad simplemente acomodando todo el código según nuestras necesidades en esta actividad:

Desarrollar un programa utilizando las librerías *Three\_JS*y *OrbitControls*que genere de manera aleatoria entre 1 y 5 cubos alineados horizontalmente y equidistante uno del otro. Adicionalmente utilizar la librería *OrbitControls*para animar la escena.

**CODIGO FINAL:**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Ejemplo de Three.js</title>

<style>

body { margin: 0; }

canvas { display: block; }

</style>

</head>

<body>

<script src="https://cdn.jsdelivr.net/npm/three@0.128.0/build/three.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/three@0.128.0/examples/js/controls/OrbitControls.js"></script>

<script>

// Configuración de la escena, cámara y renderizador

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

const renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

// Array para almacenar los cubos

const cubes = [];

// Crear una función para generar cubos

function createCubes() {

const numberOfCubes = Math.floor(Math.random() \* 5) + 1; // Generar un número aleatorio entre 1 y 5

const spacing = 2; // Espaciado entre cubos

for (let i = 0; i < numberOfCubes; i++) {

const geometry = new THREE.BoxGeometry(1, 1, 1); // Cubo

const material = new THREE.MeshBasicMaterial({ color: 0xffffff, wireframe: true }); // Material para wireframe

const wireframe = new THREE.Mesh(geometry, material); // Crear la malla de cubo

// Calcular la posición en el centro

wireframe.position.x = (i - (numberOfCubes - 1) / 2) \* spacing; // Alinear cubos en el centro

scene.add(wireframe);

cubes.push(wireframe); // Guardar el cubo en el array

}

}

// Llamar a la función para crear cubos

createCubes();

// Posicionar la cámara

camera.position.z = 5;

// Añadir OrbitControls

const controls = new THREE.OrbitControls(camera, renderer.domElement);

controls.enableDamping = true; // Habilitar el suavizado

controls.dampingFactor = 0.25; // Factor de suavizado

// Función de animación

function animate() {

requestAnimationFrame(animate);

cubes.forEach(cube => {

cube.rotation.x += 0.01; // Rotación en el eje X

cube.rotation.y += 0.01; // Rotación en el eje Y

});

controls.update(); // Actualizar controles

renderer.render(scene, camera);

}

animate();

// Listener para resize

window.addEventListener('resize', () => {

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

renderer.setSize(window.innerWidth, window.innerHeight);

});

</script>

</body>

</html>