APUNTES 11-08

Kevin Rodrigo Calle Leyton

CONCEPTOS 📌

- Hasta el momento de desarrollo aplicaciones de consola, que se ejecutan en codigo
  - La programacion en lotes es una ejecucion ejecutada hasta que el usuario haga una accion(de un clink o una letra)
  - Se usa bastante los metodos recursivos para que se haga algo hasta que el usuario haga algo.
  - La programacion asincrona es hacer multitareas mientras se ejecuta al mismo tiempo SIN bloquear al flujo principal
  - Los hilos, permiten programar un bloque de codigo de manera asincrona, lanzan multiples hilos que se ejecutan mientras fluye el hilo principal
- UnaAPI es una pieza de codigo que permite a 2 o mas aplicaciones comunicarse entre si, a su ves funciona como un puente de intercambio de informacion

QUE ES THREE JS

Three.js es una biblioteca de JavaScript que facilita la creación y visualización de gráficos 3D en el navegador web. Utiliza WebGL para renderizar gráficos en 3D, pero simplifica mucho su uso al proporcionar una API más fácil de manejar.

EJEMPLO DE CLASE

LINKS

- *https://threejs.org/docs/#api/en/core/BufferGeometry*
- *https://threejs.org/docs/#api/en/geometries/CircleGeometry*

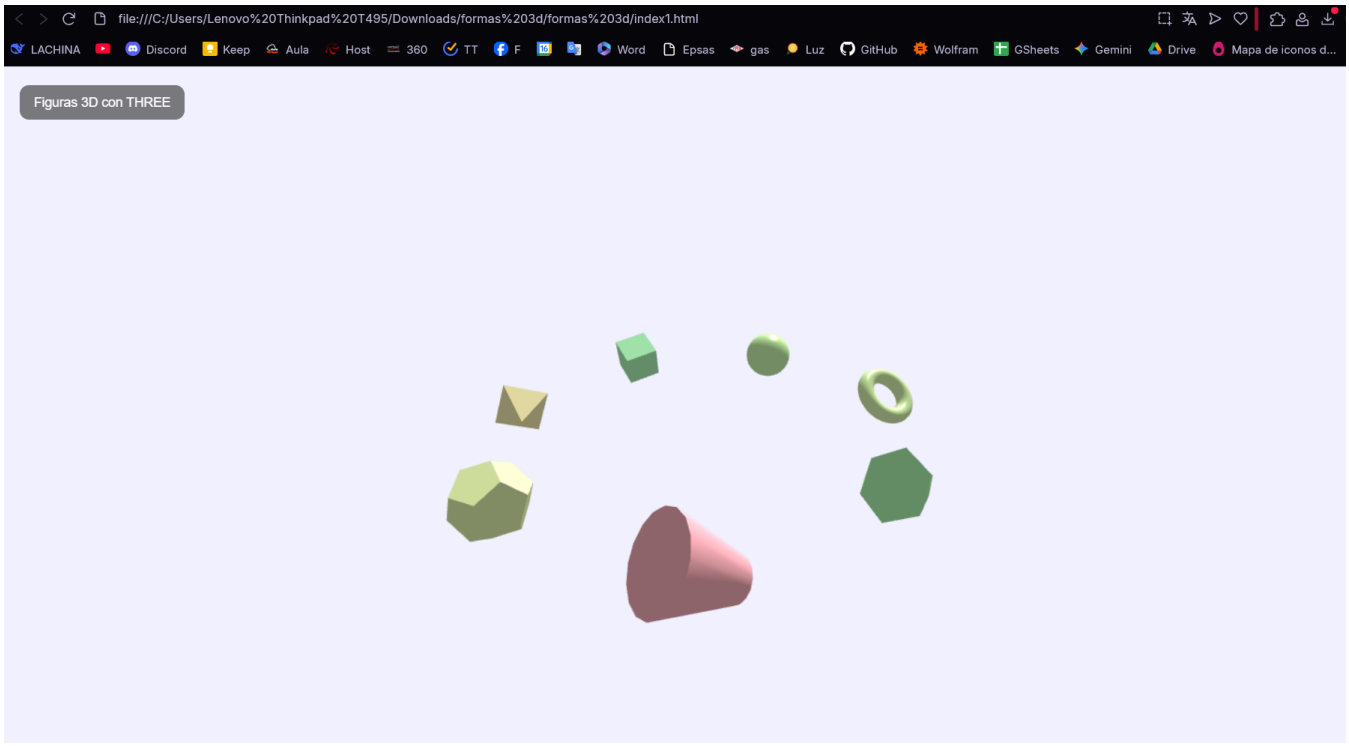
```
// Escena y renderizador

const escena = new THREE.Scene();//crear objeto escena

//se crea un renderizador

const renderizador = new THREE.WebGLRenderer();

renderizador.setSize(window.innerWidth, window.innerHeight);
```

```
document.body.appendChild(renderizador.domElement);



// Cámara ortográfica para 2D

const ancho = window.innerWidth;

const alto = window.innerHeight;

const camara = new THREE.OrthographicCamera(ancho / -2, ancho / 2, alto / 2, alto / -2, 0.1, 1000); //define los limites

camara.position.z = 10;//define la distancia



// Crear un rectángulo (plano en 2D)

const geometría = new THREE.PlaneGeometry(100, 100);  // Tamaño del rectángulo

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

const rectángulo = new THREE.Mesh(geometría, material);

escena.add(rectángulo);



//



// Animación (opcional)

function animacion() {

    requestAnimationFrame(animacion);//SE LLAMA AL OBJETOD PARA LA RECURSIVIDAD NUEVAMENTE

    rectángulo.rotation.z += 0.01;  // Rotación en 2D

    renderizador.render(escena, camara);
```

```
}

animacion();//SE UTILIZA RECURSIVIDAD



// Ajuste al redimensionar ventana

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

    const ancho = window.innerWidth;

    const alto = window.innerHeight;

    camara.left = ancho / -2;

    camara.right = ancho / 2;

    camara.top = alto / 2;

    camara.bottom = alto / -2;

    camara.updateProjectionMatrix();

    renderizador.setSize(ancho, alto);

});
```

EJERCICIOS

1. Realiza todas las figuras posibles
   Repositorio: *https://github.com/kevley123/EJERCICIOS-DE-PROGRAMACION-GRAFICA*



El HTML se ve asi:

```
<!DOCTYPE html>

<html lang="es">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Figuras 3D con THREE</title>

  <style>

    body {

      margin: 0;

      overflow: hidden;

      font-family: 'Arial', sans-serif;

    }
```

```
        .info {
            position: absolute;
            top: 20px;
            left: 20px;
            color: #fff;
            background-color: rgba(0, 0, 0, 0.5);
            padding: 10px 15px;
            border-radius: 10px;
            font-size: 14px;
            pointer-events: none;
        }
    </style>
</head>
<body>
    <div class="info">Figuras 3D con THREE</div>
    <script src="https://cdn.jsdelivr.net/npm/three@0.132.2/build/three.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/controls/OrbitControls.js"></script>
    <script>
        // camara y renderizador jsjs
        const scene = new THREE.Scene();
        const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
```

```
const renderer = new THREE.WebGLRenderer({ antialias: true });

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

document.body.appendChild(renderer.domElement);



// controles para mover y rotar

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

controls.enableDamping = true;

controls.dampingFactor = 0.05;



function getPastelColor() {

    const hue = Math.floor(Math.random() * 360);

    return new THREE.Color(`hsl(${hue}, 70%, 80%)`);

}



// figuras

const geometries = [

    new THREE.BoxGeometry(1, 1, 1),

    new THREE.SphereGeometry(0.7, 24, 24),

    new THREE.TorusGeometry(0.6, 0.2, 16, 50),

    new THREE.ConeGeometry(0.7, 1.2, 6),

    new THREE.CylinderGeometry(0.4, 0.8, 1.5, 16),
```

```
    new THREE.DodecahedronGeometry(0.8, 0),
    new THREE.OctahedronGeometry(0.8, 0)
];

const figures = geometries.map((geo, i) => {
    const material = new THREE.MeshPhongMaterial({
        color: getPastelColor(),
        shininess: 70,
        specular: 0x222222
    });
    const mesh = new THREE.Mesh(geo, material);
    const angle = (i / geometries.length) * Math.PI * 2;
    const radius = 5;
    mesh.position.set(
        Math.cos(angle) * radius,
        Math.sin(angle) * 0.5,
        Math.sin(angle) * radius
    );
    mesh.rotation.set(
        Math.random() * 0.2,
        Math.random() * 0.2,
```

```
        Math.random() * 0.2
    );
    scene.add(mesh);
    return mesh;
});

// animacion
const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);
scene.add(ambientLight);
const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);
directionalLight.position.set(5, 10, 7);
scene.add(directionalLight);

// camara (importante)
camera.position.set(0, 4, 10);
controls.update();

scene.background = new THREE.Color(0xf0f0ff);

const clock = new THREE.Clock();
function animate() {
```

```
    requestAnimationFrame(animate);

    const delta = clock.getDelta();

    const time = clock.getElapsedTime();

    figures.forEach((fig, index) => {

        fig.rotation.x += 0.005 * (index * 0.2 + 1);

        fig.rotation.y += 0.008 * (index * 0.2 + 1);

        if (Math.floor(time * 2) % 10 === index % 10) {

            fig.material.color.copy(getPastelColor());

        }

        fig.position.y += Math.sin(time + index) * 0.003;

    });

    controls.update();

    renderer.render(scene, camera);

}

animate();



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

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

    camera.updateProjectionMatrix();

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

});
```

```
    </script>

</body>

</html>
```