```
import * as THREE from 'three';
import { OrbitControls } from 'three/examples/jsm/controls/
OrbitControls';

document.addEventListener("DOMContentLoaded", () => {
    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);

    // Controls
    const controls = new OrbitControls(camera, renderer.domElement);
    controls.enableDamping = true;

    // Define Constants
    const PHI = (1 + Math.sqrt(5)) / 2;
    const FIELD_STRENGTH = 9.81; // Simulated gravity force
    const LEVITATION_FORCE = PHI * 7.5;
    let epoch = 0;

    // Particle Object (Levitation Test Subject)
    const geometry = new THREE.SphereGeometry(0.5, 32, 32);
    const material = new THREE.MeshStandardMaterial({ color: 0xff00ff,
emissive: 0x550055 });
    const particle = new THREE.Mesh(geometry, material);
    particle.position.y = 2;
    scene.add(particle);

    // Light Sources
    const ambientLight = new THREE.AmbientLight(0x404040);
    scene.add(ambientLight);
    const pointLight = new THREE.PointLight(0xffffff, 1, 100);
    pointLight.position.set(10, 10, 10);
    scene.add(pointLight);

    camera.position.z = 5;

    // Quantum Levitation Model
    function simulateLevitation() {
        const harmonicOscillation = Math.sin(epoch * PHI * Math.PI) *
0.1;
        const forceBalance = LEVITATION_FORCE - FIELD_STRENGTH +
harmonicOscillation;
        particle.position.y += forceBalance * 0.01;

        if (particle.position.y > 5) particle.position.y = 5;
        if (particle.position.y < 1) particle.position.y = 1;
    }
```

```
    // Render Loop
    function animate() {
        requestAnimationFrame(animate);
        simulateLevitation();
        controls.update();
        renderer.render(scene, camera);
        epoch++;
    }
    animate();

    // Resize Handler
    window.addEventListener('resize', () => {
        renderer.setSize(window.innerWidth, window.innerHeight);
        camera.aspect = window.innerWidth / window.innerHeight;
        camera.updateProjectionMatrix();
    });
});
```