```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>3D Solar System</title>
    <style>
        /* Minimal CSS for full-screen canvas */
        body { margin: 0; overflow: hidden; background-color: #000; }
        canvas { display: block; }
    </style>
</head>
<body>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
    <script>
        // --- 1. SETUP THE SCENE ---
        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);

        // Position the camera
        camera.position.z = 50;

        // --- 2. LIGHTING ---
        // The Sun is the main light source
        const pointLight = new THREE.PointLight(0xffffff, 2, 0, 0); // Color, Intensity, Distance, Decay
        scene.add(pointLight);

        // Add ambient light to subtly illuminate the dark side of planets
        const ambientLight = new THREE.AmbientLight(0x333333);
        scene.add(ambientLight);

        // --- 3. CREATE THE SUN (Central Object and Light Source) ---
        const sunGeometry = new THREE.SphereGeometry(5, 32, 32);
        const sunMaterial = new THREE.MeshBasicMaterial({ color: 0xFFFF00 }); // BasicMaterial doesn't respond to light
        const sun = new THREE.Mesh(sunGeometry, sunMaterial);
        scene.add(sun);

        // A list of planets with their properties
        const planetsData = [
            { name: 'Mercury', size: 0.5, color: 0xAAAAAA, distance: 8, speed: 0.048 },
            { name: 'Venus', size: 0.8, color: 0xDD7700, distance: 12, speed: 0.035 },
            { name: 'Earth', size: 1.0, color: 0x0000FF, distance: 18, speed: 0.029 },
            { name: 'Mars', size: 0.7, color: 0xFF0000, distance: 24, speed: 0.024 },
            { name: 'Jupiter', size: 3.0, color: 0xCCAA66, distance: 35, speed: 0.013 },
            { name: 'Saturn', size: 2.5, color: 0xFFAA00, distance: 48, speed: 0.009 },
    // Simplified for brevity, add rings for Saturn/Uranus for a full model
];

const planets = [];

// --- 4. CREATE PLANETS AND ORBITS ---
planetsData.forEach(data => {
    // 4a. Create the Planet Mesh
    const geometry = new THREE.SphereGeometry(data.size, 32, 32);
    // Use MeshLambertMaterial so the sun's light affects it
    const material = new THREE.MeshLambertMaterial({ color: data.color });
    const planet = new THREE.Mesh(geometry, material);

    // Add custom data for animation
    planet.distance = data.distance;
    planet.speed = data.speed;
    planet.angle = Math.random() * Math.PI * 2; // Start at a random point in orbit

    scene.add(planet);
    planets.push(planet);

    // 4b. Create the Orbit Ring (using a TorusGeometry or Line)
    const orbitGeometry = new THREE.RingGeometry(data.distance - 0.05, data.distance + 0.05, 128);
    const orbitMaterial = new THREE.MeshBasicMaterial({
        color: 0x555555,
        side: THREE.DoubleSide,
        transparent: true,
        opacity: 0.2
    });
    const orbit = new THREE.Mesh(orbitGeometry, orbitMaterial);
    // Orient the orbit flat on the X-Z plane
    orbit.rotation.x = Math.PI / 2;
    scene.add(orbit);
});

// --- 5. INTERACTIVITY: MOUSE CONTROL (Basic Orbital Camera) ---
let isDragging = false;
let previousMousePosition = { x: 0, y: 0 };
let rotationSpeed = 0.005;

// Handle mouse down to start dragging
document.addEventListener('mousedown', (e) => {
    isDragging = true;
    previousMousePosition.x = e.clientX;
    previousMousePosition.y = e.clientY;
});

// Handle mouse up to stop dragging
document.addEventListener('mouseup', () => {
    isDragging = false;
});
```

```
// Handle mouse move to rotate the camera
document.addEventListener('mousemove', (e) => {
    if (!isDragging) return;

    const deltaX = e.clientX - previousMousePosition.x;
    const deltaY = e.clientY - previousMousePosition.y;

    // Rotate the entire scene around the Y-axis (vertical drag) and X-axis (horizontal drag)
    scene.rotation.y += deltaX * rotationSpeed;
    scene.rotation.x += deltaY * rotationSpeed;

    // Clamp the X rotation to prevent the scene from flipping over
    scene.rotation.x = Math.max(-Math.PI / 2, Math.min(Math.PI / 2, scene.rotation.x));

    previousMousePosition.x = e.clientX;
    previousMousePosition.y = e.clientY;
});

// Handle mouse wheel for zooming
document.addEventListener('wheel', (e) => {
    const zoomFactor = 0.95; // Controls speed of zoom
    if (e.deltaY > 0) {
        // Zoom out (increase Z position)
        camera.position.z /= zoomFactor;
    } else {
        // Zoom in (decrease Z position)
        camera.position.z *= zoomFactor;
    }
    // Clamp zoom distance
    camera.position.z = Math.max(10, Math.min(200, camera.position.z));
});


// --- 6. ANIMATION LOOP ---
function animate() {
    requestAnimationFrame(animate);

    // 6a. Rotate the Sun (Axial Rotation)
    sun.rotation.y += 0.001;

    // 6b. Move the Planets (Orbital Revolution)
    planets.forEach(planet => {
        // Update the angle based on its orbital speed
        planet.angle += planet.speed * 0.01;

        // Calculate the new X and Z positions on the orbit plane (X-Z plane)
        planet.position.x = planet.distance * Math.cos(planet.angle);
        planet.position.z = planet.distance * Math.sin(planet.angle);

        // Rotate the planet on its own axis
```

```
                planet.rotation.y += 0.01;
            });

            renderer.render(scene, camera);
        }

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

        // Start the animation
        animate();

    </script>
</body>
</html>
```