```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>GRIDIRON 26: PRO-GEN 3D ENGINE</title>
   <script src="https://cdnjs.cloudflare.com"></script>
   <style>
      :root { --primary: #00ff88; --bg: #050505; --ui-panel: rgba(10, 10, 10, 0.95); }
      body { margin: 0; background: var(--bg); color: #fff; font-family: 'Segoe UI', sans-serif;
overflow: hidden; }

      /* HUD UI */
      #hud { position: absolute; top: 0; width: 100%; display: flex; justify-content: space-between;
padding: 20px; pointer-events: none; z-index: 5; }
      .stat-box { background: var(--ui-panel); padding: 15px; border-left: 4px solid var(--primary);
min-width: 120px; }

      /* OVERLAYS */
      #menu-overlay { position: absolute; inset: 0; background: rgba(0,0,0,0.9); display: flex;
align-items: center; justify-content: center; z-index: 100; }
      .panel { background: #111; padding: 30px; border: 1px solid #333; width: 600px;
max-height: 90vh; overflow-y: auto; }

      /* BUTTONS & INPUTS */
      .btn { background: var(--primary); color: #000; border: none; padding: 12px; font-weight:
900; cursor: pointer; width: 100%; margin: 5px 0; text-transform: uppercase; }
      .btn:hover { background: #fff; }
      .skill-node { display: flex; justify-content: space-between; padding: 10px; background:
#222; margin-bottom: 5px; }

      #game-log { position: absolute; bottom: 20px; left: 20px; color: var(--primary); font-size:
12px; }
   </style>
</head>
<body>

<div id="hud">
   <div class="stat-box">
      <div style="font-size: 10px; color: #888;">QTR 1 | 10:00</div>
      <div id="ui-score">COL 0 - 0 TEX</div>
      <div id="ui-down" style="color: var(--primary); font-weight: bold;">1st & 10</div>
   </div>
   <div class="stat-box" style="text-align: right;">
      <div id="ui-player-name">PROSPECT</div>
```

```
        <div id="ui-ovr">OVR: 75</div>
        <div id="ui-xp" style="color: gold;">XP: 0</div>
    </div>
</div>

<div id="menu-overlay">
    <div class="panel" id="ui-content">
        <h1 style="margin-top:0;">GRIDIRON 26: CAREER</h1>

        <h3>1. CUSTOMIZE CHARACTER</h3>
        <input type="text" id="p-name" placeholder="Player Name" class="btn"
style="background:#222; color:#fff; text-align:left;">
        <select id="p-pos" class="btn">
            <option value="QB">QUARTERBACK (Field General)</option>
            <option value="HB">HALFBACK (Power/Speed)</option>
            <option value="WR">WIDE RECEIVER (Deep Threat)</option>
            <option value="LB">LINEBACKER (Hit Stick)</option>
        </select>

        <h3>2. UPGRADE SKILLS (XP: <span id="menu-xp">0</span>)</h3>
        <div id="skills-container">
            <div class="skill-node"><span>SPEED</span><button
onclick="Game.upgrade('spd')">+</button></div>
            <div class="skill-node"><span>STRENGTH</span><button
onclick="Game.upgrade('str')">+</button></div>
            <div class="skill-node"><span>AWARENESS</span><button
onclick="Game.upgrade('awr')">+</button></div>
        </div>

        <button class="btn" style="margin-top:20px; background:white;"
onclick="Game.startMatch()">ENTER STADIUM</button>
    </div>
</div>

<div id="game-log">System: Engine Initialized. Ready for 2026 Season.</div>

<script>
/**
 * GRIDIRON 26: CORE ENGINE OBJECT
 */
const Game = {
    // 1. Data Store
    player: {
        name: "Prospect", pos: "QB", ovr: 75, xp: 500,
        stats: { spd: 70, str: 70, awr: 70 },
        mesh: null
    },
    gameState: {
        year: 2026, down: 1, ytg: 10, score: [0,0],
        clock: 600, active: false
    },

    // 2. 3D Engine Properties
    scene: null, camera: null, renderer: null,
    entities: [], opponents: [],
    keys: {},

    initEngine() {
        this.scene = new THREE.Scene();
        this.scene.background = new THREE.Color(0x020202);
        this.camera = new THREE.PerspectiveCamera(75,
window.innerWidth/window.innerHeight, 0.1, 1000);
        this.renderer = new THREE.WebGLRenderer({ antialias: true });
        this.renderer.setSize(window.innerWidth, window.innerHeight);
        document.body.appendChild(this.renderer.domElement);

        // Lights & Stadium Floor
        const ambient = new THREE.AmbientLight(0xffffff, 0.5);
        const sun = new THREE.DirectionalLight(0xffffff, 1);
        sun.position.set(10, 50, 10);
        this.scene.add(ambient, sun);

        const fieldGeo = new THREE.PlaneGeometry(120, 53.3);
        const fieldMat = new THREE.MeshPhongMaterial({ color: 0x1a3d16 });
        const field = new THREE.Mesh(fieldGeo, fieldMat);
        field.rotation.x = -Math.PI/2;
        this.scene.add(field);

        // Draw Yard Lines
        for(let i = -50; i <= 50; i += 10) {
            const lineGeo = new THREE.PlaneGeometry(0.2, 53.3);
            const lineMat = new THREE.MeshBasicMaterial({color: 0xffffff, transparent: true, opacity:
0.3});
            const line = new THREE.Mesh(lineGeo, lineMat);
            line.rotation.x = -Math.PI/2;
            line.position.set(i, 0.01, 0);
            this.scene.add(line);
        }
        this.camera.position.set(-25, 15, 0);
        this.camera.lookAt(0,0,0);

        window.addEventListener('keydown', e => this.keys[e.code] = true);
        window.addEventListener('keyup', e => this.keys[e.code] = false);

        this.animate();
    },

    startMatch() {
        this.player.name = document.getElementById('p-name').value || "Prospect";
        this.player.pos = document.getElementById('p-pos').value;
        document.getElementById('menu-overlay').style.display = 'none';
        document.getElementById('ui-player-name').innerText = this.player.name;

        this.spawnPlayers();
        this.gameState.active = true;
    },

    spawnPlayers() {
        // Clear previous entities
        this.entities.forEach(e => this.scene.remove(e));
        this.entities = [];
        this.opponents = [];

        // Spawn User Player
        const pGeo = new THREE.CapsuleGeometry(0.5, 1, 4, 8);
        const pMat = new THREE.MeshPhongMaterial({ color: 0x00ff88 });
        this.player.mesh = new THREE.Mesh(pGeo, pMat);
        this.player.mesh.position.set(-20, 1, 0);
        this.scene.add(this.player.mesh);
        this.entities.push(this.player.mesh);

        // Spawn 11 AI Opponents (Defense)
        for(let i=0; i<11; i++) {
            const oppMat = new THREE.MeshPhongMaterial({ color: 0xff4400 });
            const opp = new THREE.Mesh(pGeo, oppMat);
            opp.position.set(5 + Math.random()*5, 1, (i-5)*4);
            this.scene.add(opp);
            this.opponents.push(opp);
        }
    },
    upgrade(stat) {
        if(this.player.xp >= 100) {
            this.player.stats[stat] += 5;
            this.player.xp -= 100;
            this.player.ovr = Math.floor((this.player.stats.spd + this.player.stats.str +
this.player.stats.awr)/3);
            document.getElementById('menu-xp').innerText = this.player.xp;
            document.getElementById('ui-xp').innerText = `XP: ${this.player.xp}`;
            document.getElementById('ui-ovr').innerText = `OVR: ${this.player.ovr}`;
        }
    },

    updatePhysics() {
        if(!this.gameState.active) return;

        // Player Movement (W/A/S/D)
        const moveSpeed = this.player.stats.spd / 500;
        if(this.keys['KeyW']) this.player.mesh.position.x += moveSpeed;
        if(this.keys['KeyS']) this.player.mesh.position.x -= moveSpeed;
        if(this.keys['KeyA']) this.player.mesh.position.z -= moveSpeed;
        if(this.keys['KeyD']) this.player.mesh.position.z += moveSpeed;

        // Camera follow
        this.camera.position.x = this.player.mesh.position.x - 15;
        this.camera.position.z = this.player.mesh.position.z;
        this.camera.lookAt(this.player.mesh.position);

        // AI Pursuit Logic (Defenders chase player)
        this.opponents.forEach(opp => {
            const dist = opp.position.distanceTo(this.player.mesh.position);
            const aiSpeed = 0.08;

            if(dist < 30) { // Aggro range
                opp.lookAt(this.player.mesh.position);
                opp.translateZ(aiSpeed);
            }

            // Tackle Collision
            if(dist < 1.2) {
                this.handleTackle();
            }
        });
    },
    handleTackle() {
        this.gameState.active = false;
        document.getElementById('game-log').innerText = "System: TACKLED! 2nd & 5.";
        this.player.xp += 20;
        setTimeout(() => {
            this.player.mesh.position.set(-20, 1, 0);
            this.gameState.active = true;
        }, 2000);
    },

    animate() {
        requestAnimationFrame(() => this.animate());
        this.updatePhysics();
        this.renderer.render(this.scene, this.camera);
    }
};

// Initialize on Load
window.onload = () => Game.initEngine();
</script>
</body>
</html>
```