```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>GRIDIRON 26: ADVANCED SIMULATION</title>
   <style>
      :root { --ncaa: #ff4500; --nfl: #013369; --bg: #0a0a0a; }
      body { background: var(--bg); color: #fff; font-family: 'Inter', sans-serif; display: flex;
flex-direction: column; align-items: center; margin: 0; overflow: hidden; }
      .hud { width: 1000px; display: grid; grid-template-columns: repeat(8, 1fr); gap: 10px;
padding: 15px; background: #111; border-bottom: 3px solid var(--ncaa); }
      .card { background: #1a1a1a; padding: 8px; border-radius: 4px; text-align: center; border:
1px solid #333; }
      .label { font-size: 9px; color: #888; text-transform: uppercase; }
      .val { font-size: 14px; font-weight: 800; }
      canvas { background: #245e1c; border: 5px solid #222; box-shadow: 0 0 50px
rgba(0,0,0,0.8); }
      #overlay { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);
background: rgba(0,0,0,0.95); padding: 30px; border-radius: 10px; border: 2px solid var(--ncaa);
text-align: center; width: 450px; z-index: 100; }
      select, button { width: 100%; padding: 12px; margin: 5px 0; border-radius: 5px; font-weight:
bold; cursor: pointer; border: none; }
      button { background: var(--ncaa); color: #fff; }
      .msg { position: absolute; top: 150px; font-size: 40px; font-weight: 900; color: yellow;
text-shadow: 2px 2px #000; display: none; }
      #log { font-size: 12px; color: #aaa; margin-top: 10px; }
   </style>
</head>
<body>

<div class="hud">
   <div class="card"><div class="label">Era</div><div class="val" id="ui-year">2025
NCAA</div></div>
   <div class="card"><div class="label">Score</div><div class="val" id="ui-score">0 -
0</div></div>
   <div class="card"><div class="label">Momentum</div><div class="val"
id="ui-phys">Stable</div></div>
   <div class="card"><div class="label">Down</div><div class="val" id="ui-down">1st &
10</div></div>
   <div class="card"><div class="label">XP Level</div><div class="val" id="ui-xp">Lv.
1</div></div>
   <div class="card"><div class="label">Health</div><div class="val"
id="ui-health">100%</div></div>
```

```
  <div class="card"><div class="label">OVR</div><div class="val" id="ui-ovr">75</div></div>
  <div class="card"><div class="label">Injuries</div><div class="val"
id="ui-injuries">0</div></div>
</div>

<div id="msg" class="msg">TOUCHDOWN!</div>
<canvas id="simCanvas" width="1000" height="500"></canvas>
<div id="log">Game Log: Ready to play.</div>

<div id="overlay">
  <h2 id="menu-title">GRIDIRON 26</h2>
  <div id="menu-content">
    <select id="select-year">
      <option value="2025">Start Year: 2025</option>
      <option value="2026">Start Year: 2026</option>
      <option value="2027">Start Year: 2027</option>
    </select>
    <select id="select-team">
      <option value="Colorado">Colorado Buffs (NCAA)</option>
      <option value="Texas">Texas Longhorns (NCAA)</option>
      <option value="Eagles">Philly Eagles (NFL)</option>
      <option value="Chiefs">KC Chiefs (NFL)</option>
    </select>
    <button onclick="Game.init()">BREAK HUDDLE</button>
  </div>
</div>

<script>
const Game = {
  canvas: document.getElementById('simCanvas'),
  ctx: document.getElementById('simCanvas').getContext('2d'),
  active: false,
  year: 2025, xp: 0, score: 0, oppScore: 0, injuries: 0,
  team: 'Colorado', mode: 'NCAA', phase: 'Offense',
  player: { x: 100, y: 250, vx: 0, vy: 0, mass: 220, speed: 5, health: 100, baseOVR: 70,
currentOVR: 70 },
  defenders: [],

  init() {
    this.year = parseInt(document.getElementById('select-year').value);
    this.team = document.getElementById('select-team').value;
    this.mode = this.team.includes('Eagles') || this.team.includes('Chiefs') ? 'NFL' : 'NCAA';
    document.getElementById('overlay').style.display = 'none';
```

```
        document.documentElement.style.setProperty('--ncaa', this.mode === 'NFL' ? '#013369' :
'#ff4500');
        this.active = true;
        this.player.currentOVR = this.player.baseOVR + Math.floor(this.xp/500);
        document.getElementById('ui-ovr').innerText = this.player.currentOVR;
        this.resetPlay();
        this.loop();
        this.logMessage(`Starting ${this.year} season as a ${this.mode} player.`);
    },

    resetPlay() {
        this.player.x = 100; this.player.y = 250;
        this.player.vx = 0; this.player.vy = 0;
        this.phase = 'Offense';
        this.defenders = Array.from({length: 5}, () => ({
            x: 500 + Math.random() * 300,
            y: 50 + Math.random() * 400,
            speed: 2.5 + (this.mode === 'NFL' ? 1 : 0) + (this.year - 2025) * 0.2 // NFL/Year difficulty
        }));
    },

    update() {
        if (!this.active) return;

        // Physics-Based Movement
        if (Keys.W) this.player.vy -= 0.6; if (Keys.S) this.player.vy += 0.6;
        if (Keys.A) this.player.vx -= 0.6; if (Keys.D) this.player.vx += 0.6;

        this.player.vx *= 0.92; this.player.vy *= 0.92;
        this.player.x += this.player.vx * (this.player.health / 100); // Health affects speed
        this.player.y += this.player.vy * (this.player.health / 100);

        // Boundary Logic (Out of Bounds)
        if (this.player.y < 50 || this.player.y > 450) this.handleWhistle("Out of Bounds");

        // Defender AI & Physics Tackling
        this.defenders.forEach(d => {
            let dx = this.player.x - d.x;
            let dy = this.player.y - d.y;
            let dist = Math.hypot(dx, dy);

            d.x += (dx/dist) * d.speed;
            d.y += (dy/dist) * d.speed;
            if (dist < 25) this.resolveCollision(d);
        });

        // Touchdown
        if (this.player.x > 920) this.handleScore();
    },

    resolveCollision(d) {
        let momentum = Math.abs(this.player.vx) * this.player.mass * (this.player.currentOVR /
100); // OVR affects trucking
        if (momentum > 450) {
            document.getElementById('ui-phys').innerText = "Stumble Recovered!";
            d.x -= 50; // Trucked!
            this.player.vx *= 0.3;
            this.logMessage("Trucked a defender with pure OVR!");
        } else {
            this.handleWhistle("Tackled!");
            this.checkForInjury();
        }
    },

    checkForInjury() {
        if (Math.random() < 0.1) { // 10% injury chance per tackle
            this.player.health -= 25;
            this.injuries += 1;
            document.getElementById('ui-health').innerText = `${this.player.health}%`;
            document.getElementById('ui-injuries').innerText = this.injuries;
            this.logMessage(`INJURY: Health reduced to ${this.player.health}%.`);
            if (this.player.health <= 0) {
                this.handleWhistle("Season-ending Injury!");
                this.xp *= 0.5; // XP Penalty for major injury
            }
        }
    },

    handleWhistle(msg) {
        this.active = false;
        this.xp += 25;
        this.updateHUD();
        this.logMessage(msg);
        setTimeout(() => {
            this.active = true;
            this.resetPlay();
        }, 1500);
    },

    handleScore() {
        this.active = false;
        this.score += 7;
        this.xp += 150;
        document.getElementById('ui-score').innerText = `${this.score} - ${this.oppScore}`;
        document.getElementById('msg').innerText = "TOUCHDOWN!";
        document.getElementById('msg').style.display = 'block';

        setTimeout(() => {
            document.getElementById('msg').style.display = 'none';
            this.updateHUD();
            this.resetPlay();
            this.active = true;
        }, 2000);
        this.logMessage("TOUCHDOWN!");
    },

    updateHUD() {
        document.getElementById('ui-xp').innerText = `Lv. ${Math.floor(this.xp/500) + 1}`;
        document.getElementById('ui-health').innerText = `${this.player.health}%`;
        this.checkForOffseason();
    },

    checkForOffseason() {
        if (this.xp > 1000 && this.mode === 'NCAA') {
            this.active = false;
            document.getElementById('overlay').style.display = 'block';
            document.getElementById('menu-title').innerText = `SEASON ${this.year} COMPLETE`;
            document.getElementById('menu-content').innerHTML = `
                <p>OVR Increased! New OVR: ${this.player.currentOVR + 1}.</p>
                <button onclick="Game.declareForDraft()">DECLARE FOR NFL DRAFT (Go
Pro)</button>
                <button onclick="Game.enterPortal()">ENTER TRANSFER PORTAL (Next NCAA
Year)</button>
            `;
            this.logMessage(`Offseason triggered! Declare for NFL Draft or hit the transfer portal.
Your OVR increased.`);
        }
    },

    declareForDraft() {
        this.year = 2026;
        this.mode = 'NFL';
        this.xp = 0;
        this.player.health = 100;
        this.player.baseOVR += 1;
        document.getElementById('ui-year').innerText = `${this.year} NFL PRO`;
        this.init();
    },

    enterPortal() {
        this.year += 1;
        this.xp = 0;
        this.player.health = 100;
        this.player.baseOVR += 1; // Slight OVR boost for another year of college
        document.getElementById('ui-year').innerText = `${this.year} NCAA`;
        this.init();
    },

    logMessage(msg) {
        document.getElementById('log').innerText = `Game Log: ${msg}`;
    },

    draw() {
        const {ctx, canvas, player} = this;
        ctx.clearRect(0,0,1000,500);

        // Field texture/lines (simplified for brevity)
        ctx.fillStyle = '#245e1c'; ctx.fillRect(0,0,1000,500);
        ctx.strokeStyle = 'rgba(255,255,255,0.3)';
        for(let i=0; i<1000; i+=50) { ctx.beginPath(); ctx.moveTo(i,50); ctx.lineTo(i,450); ctx.stroke();
}
        ctx.fillStyle = '#111'; ctx.fillRect(0,0,1000,50); ctx.fillRect(0,450,1000,50);

        // Player Shadow and Rating Display
        ctx.fillStyle = 'rgba(0,0,0,0.3)'; ctx.beginPath(); ctx.ellipse(player.x, player.y+15, 15, 5, 0, 0,
Math.PI*2); ctx.fill();
        ctx.fillStyle = (this.mode === 'NCAA') ? '#ff4500' : '#013369';
        ctx.beginPath(); ctx.arc(player.x, player.y, 15, 0, Math.PI*2); ctx.fill();
        ctx.strokeStyle = '#fff'; ctx.lineWidth = 3; ctx.stroke();
        ctx.fillStyle = '#fff'; ctx.font = '10px Inter'; ctx.textAlign = 'center';
        ctx.fillText(player.currentOVR, player.x, player.y + 3);

        // Defenders
        ctx.fillStyle = '#333';
        this.defenders.forEach(d => {
            ctx.beginPath(); ctx.arc(d.x, d.y, 15, 0, Math.PI*2); ctx.fill();
            ctx.strokeStyle = '#fff'; ctx.stroke();
        });
    },

    loop() {
        this.update();
        this.draw();
        requestAnimationFrame(this.loop.bind(this));
    }
};

const Keys = { W: false, S: false, A: false, D: false };
window.onkeydown = e => { if (['KeyW', 'KeyS', 'KeyA', 'KeyD'].includes(e.code))
Keys[e.code.replace('Key', '')] = true; };
window.onkeyup = e => { if (['KeyW', 'KeyS', 'KeyA', 'KeyD'].includes(e.code))
Keys[e.code.replace('Key', '')] = false; };
</script>
</body>
</html>
```