```
<!DOCTYPE html>
<html lang=”en”>
<head>
    <meta charset=”UTF-8”>
    <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
    <title>Digital Decay</title>
    <style>
        body {
            margin: 0;
            overflow: hidden;
            background: #000;
            font-family: monospace;
        }
        canvas {
            display: block;
        }
    </style>
</head>
<body>
    <canvas id=”canvas”></canvas>
    <script>
        const canvas = document.getElementById(’canvas’);
```

```
const ctx = canvas.getContext('2d');

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;

// Glitch parameters

let glitchIntensity = 0;

let time = 0;

// Particle system for digital flames

const flames = [];

const codeParticles = [];

class Flame {

    constructor(x, y) {

        this.x = x;

        this.y = y;

        this.vx = (Math.random() - 0.5) * 2;

        this.vy = -Math.random() * 3 - 1;

        this.life = 1;

        this.size = Math.random() * 30 + 15;

        this.hue = Math.random() * 40 + 10; // Orange to yellow flames

    }

    update() {

        this.x += this.vx;

        this.y += this.vy;
```

```
        this.life -= 0.01;
        this.size *= 0.98;
    }
    draw() {
        ctx.save();
        ctx.globalAlpha = this.life * 0.7;
        const grad = ctx.createRadialGradient(this.x, this.y, 0, this.x, this.y, this.size);
        grad.addColorStop(0, `hsla(${this.hue + 10}, 100%, 70%, 1)`);
        grad.addColorStop(0.3, `hsla(${this.hue}, 100%, 50%, 0.8)`);
        grad.addColorStop(0.6, `hsla(${this.hue - 10}, 100%, 40%, 0.4)`);
        grad.addColorStop(1, 'transparent');
        ctx.fillStyle = grad;
        ctx.fillRect(this.x - this.size, this.y - this.size, this.size * 2, this.size * 2);
        // Add bright core
        ctx.globalAlpha = this.life;
        ctx.fillStyle = `hsla(50, 100%, 90%, ${this.life * 0.5})`;
        ctx.fillRect(this.x - this.size * 0.3, this.y - this.size * 0.3, this.size * 0.6, this.size * 0.6);
        ctx.restore();
    }
}
class CodeParticle {
    constructor(x, y) {
```

```
        this.x = x;
        this.y = y;
        this.vx = (Math.random() - 0.5) * 3;
        this.vy = Math.random() * 2 + 1;
        this.life = 1;
        this.char = ['0', '1', 'X', '#', '@', '*'][Math.floor(Math.random() * 6)];
        this.size = Math.random() * 12 + 8;
    }
    update() {
        this.x += this.vx;
        this.y += this.vy;
        this.life -= 0.008;
    }
    draw() {
        ctx.save();
        ctx.globalAlpha = this.life;
        ctx.fillStyle = `hsl(${190 + Math.random() * 40}, 100%, 60%)`;
        ctx.font = `${this.size}px monospace`;
        ctx.fillText(this.char, this.x, this.y);
        ctx.restore();
    }
}
```

```
// Robot figure
function drawRobot() {
    const centerX = canvas.width / 2;
    const centerY = canvas.height / 2;
    ctx.save();
    // Apply glitch effect
    if (Math.random() < 0.3) {
        ctx.translate(Math.random() * 10 - 5, Math.random() * 10 - 5);
    }
    // Head shape (more oval)
    ctx.strokeStyle = '#00ffff';
    ctx.lineWidth = 2;
    ctx.fillStyle = '#FFA500';
    ctx.beginPath();
    ctx.ellipse(centerX, centerY - 80, 45, 50, 0, 0, Math.PI * 2);
    ctx.fill();
    ctx.stroke();
    // Hair (distinctive Trump hair)
    ctx.fillStyle = '#FFD700';
    ctx.beginPath();
    ctx.ellipse(centerX, centerY - 115, 50, 25, -0.2, 0, Math.PI);
    ctx.fill();
```

```
// Hair swoop
ctx.beginPath();
ctx.arc(centerX + 30, centerY - 110, 20, 0, Math.PI * 1.5);
ctx.fill();
// Glitch lines across head
if (Math.random() < 0.4) {
    ctx.fillStyle = `rgba(255, 0, ${Math.random() * 255}, 0.5)`;
    ctx.fillRect(centerX - 50, centerY - 90 + Math.random() * 60, 100, 3);
}
// Eyes with flickering - narrowed, menacing
const eyeAlpha = Math.random() > 0.1 ? 1 : 0.2;
ctx.globalAlpha = eyeAlpha;
// Narrowed, intense eyes
ctx.fillStyle = '#ff0055';
ctx.beginPath();
ctx.ellipse(centerX - 15, centerY - 85, 7, 4, 0, 0, Math.PI * 2);
ctx.fill();
ctx.beginPath();
ctx.ellipse(centerX + 15, centerY - 85, 7, 4, 0, 0, Math.PI * 2);
ctx.fill();
// Dark eyebrows, furrowed and angry
ctx.globalAlpha = 1;
```

```
ctx.strokeStyle = '#8B4513';
ctx.lineWidth = 3;
ctx.beginPath();
ctx.moveTo(centerX - 25, centerY - 93);
ctx.lineTo(centerX - 8, centerY - 90);
ctx.stroke();
ctx.beginPath();
ctx.moveTo(centerX + 25, centerY - 93);
ctx.lineTo(centerX + 8, centerY - 90);
ctx.stroke();
ctx.globalAlpha = 1;
// Nose
ctx.fillStyle = '#FF8C00';
ctx.beginPath();
ctx.moveTo(centerX, centerY - 75);
ctx.lineTo(centerX - 5, centerY - 65);
ctx.lineTo(centerX + 5, centerY - 65);
ctx.closePath();
ctx.fill();
// Mouth (stern downturned scowl)
ctx.strokeStyle = '#8B4513';
ctx.lineWidth = 3;
```

```
ctx.beginPath();

ctx.moveTo(centerX - 12, centerY - 52);

ctx.quadraticCurveTo(centerX, centerY - 48, centerX + 12, centerY - 52);

ctx.stroke();

// Additional shadow under mouth for more menacing look

ctx.fillStyle = 'rgba(0, 0, 0, 0.3)';

ctx.beginPath();

ctx.ellipse(centerX, centerY - 45, 15, 5, 0, 0, Math.PI * 2);

ctx.fill();

// Neck/collar

ctx.fillStyle = '#FFA500';

ctx.fillRect(centerX - 20, centerY - 30, 40, 20);

// Suit jacket

ctx.fillStyle = '#1a1a2e';

ctx.strokeStyle = '#00ffff';

ctx.lineWidth = 2;

ctx.beginPath();

ctx.moveTo(centerX - 20, centerY - 10);

ctx.lineTo(centerX - 70, centerY - 10);

ctx.lineTo(centerX - 80, centerY + 80);

ctx.lineTo(centerX - 30, centerY + 80);

ctx.lineTo(centerX, centerY + 30);
```

```
ctx.closePath();

ctx.fill();

ctx.stroke();

ctx.beginPath();

ctx.moveTo(centerX + 20, centerY - 10);

ctx.lineTo(centerX + 70, centerY - 10);

ctx.lineTo(centerX + 80, centerY + 80);

ctx.lineTo(centerX + 30, centerY + 80);

ctx.lineTo(centerX, centerY + 30);

ctx.closePath();

ctx.fill();

ctx.stroke();

// White shirt

ctx.fillStyle = '#ffffff';

ctx.fillRect(centerX - 20, centerY - 10, 40, 40);

// Red tie

ctx.fillStyle = '#DC143C';

ctx.beginPath();

ctx.moveTo(centerX - 8, centerY - 10);

ctx.lineTo(centerX - 10, centerY + 40);

ctx.lineTo(centerX, centerY + 50);

ctx.lineTo(centerX + 10, centerY + 40);
```

```
ctx.lineTo(centerX + 8, centerY - 10);
ctx.closePath();
ctx.fill();
// Digital artifacts on body
for (let i = 0; i < 5; i++) {
    ctx.fillStyle = `rgba(0, 255, 255, ${Math.random() * 0.3})`;
    ctx.fillRect(
        centerX - 50 + Math.random() * 100,
        centerY - 20 + Math.random() * 80,
        Math.random() * 20 + 5,
        2
    );
}
// Arms holding flag
ctx.strokeStyle = '#00ffff';
ctx.lineWidth = 3;
ctx.beginPath();
ctx.moveTo(centerX - 70, centerY - 10);
ctx.lineTo(centerX - 90, centerY + 20);
ctx.stroke();
// Right arm extended to hold flag
ctx.beginPath();
```

```
    ctx.moveTo(centerX + 70, centerY - 10);
    ctx.lineTo(centerX + 95, centerY + 10);
    ctx.stroke();
    // Hand holding flagpole
    ctx.fillStyle = '#FFA500';
    ctx.beginPath();
    ctx.arc(centerX + 95, centerY + 10, 8, 0, Math.PI * 2);
    ctx.fill();
    ctx.stroke();
    ctx.restore();
}
// Flag melting into code
function drawFlag() {
    const flagX = canvas.width / 2 + 90;
    const flagY = canvas.height / 2;
    // Flag pole
    ctx.strokeStyle = '#888';
    ctx.lineWidth = 4;
    ctx.beginPath();
    ctx.moveTo(flagX, flagY + 20);
    ctx.lineTo(flagX, flagY - 120);
    ctx.stroke();
```

```
const stripeHeight = 7;
const numStripes = 13;
const flagWidth = 100;
const starFieldWidth = 40;
const starFieldHeight = stripeHeight * 7;
// Draw stripes with wave and dissolution
for (let i = 0; i < numStripes; i++) {
    const y = flagY - 110 + i * stripeHeight;
    const isRed = i % 2 === 0;
    // Draw stripe in segments for wave effect
    for (let seg = 0; seg < 20; seg++) {
        const x = flagX + (seg / 20) * flagWidth;
        const wave = Math.sin(time * 0.05 + seg * 0.3 + i * 0.1) * 4;
        const dissolve = (seg / 20) + (Math.sin(time * 0.1 + seg * 0.2) * 0.2);
        if (Math.random() > dissolve * 0.7) {
            ctx.fillStyle = isRed ? '#B22234' : '#FFFFFF';
            ctx.globalAlpha = 1 - (dissolve * 0.4);
            ctx.fillRect(x, y + wave, flagWidth / 20 + 1, stripeHeight);
        } else if (Math.random() < 0.03) {
            codeParticles.push(new CodeParticle(x, y + wave));
        }
    }
```

```
}

// Star field (blue canton)

ctx.globalAlpha = 0.9;

const cantonWave = Math.sin(time * 0.05) * 3;

ctx.fillStyle = '#3C3B6E';

ctx.fillRect(flagX, flagY - 110 + cantonWave, starFieldWidth, starFieldHeight);

// Draw stars in rows

ctx.fillStyle = '#FFFFFF';

const starSize = 3;

// 9 stars, 8 stars alternating for 5 rows (simplified)

for (let row = 0; row < 5; row++) {

    const numStarsInRow = row % 2 === 0 ? 6 : 5;

    const offsetX = row % 2 === 0 ? 4 : 7;

    for (let col = 0; col < numStarsInRow; col++) {

        const sx = flagX + offsetX + col * 6.5;

        const sy = flagY - 105 + row * 10 + cantonWave;

        // Simple star shape

        ctx.beginPath();

        for (let p = 0; p < 5; p++) {

            const angle = (p * 4 * Math.PI) / 5 - Math.PI / 2;

            const radius = p % 2 === 0 ? starSize : starSize * 0.4;

            const px = sx + Math.cos(angle) * radius;
```

```
            const py = sy + Math.sin(angle) * radius;
            if (p === 0) ctx.moveTo(px, py);
            else ctx.lineTo(px, py);
        }
        ctx.closePath();
        ctx.fill();
    }
  }
  ctx.globalAlpha = 1;
}
// Glitch overlay
function applyGlitch() {
  if (Math.random() < 0.1) {
    const sliceHeight = Math.random() * 50 + 10;
    const sliceY = Math.random() * canvas.height;
    const offset = (Math.random() - 0.5) * 50;
    const imageData = ctx.getImageData(0, sliceY, canvas.width, sliceHeight);
    ctx.putImageData(imageData, offset, sliceY);
    // RGB shift
    ctx.globalCompositeOperation = 'screen';
    ctx.fillStyle = 'rgba(255, 0, 0, 0.1)';
    ctx.fillRect(offset + 2, sliceY, canvas.width, sliceHeight);
```

```
            ctx.fillStyle = 'rgba(0, 255, 255, 0.1)';
            ctx.fillRect(offset - 2, sliceY, canvas.width, sliceHeight);
            ctx.globalCompositeOperation = 'source-over';
        }
    }
    function animate() {
        // Fade effect for trails
        ctx.fillStyle = 'rgba(0, 0, 0, 0.15)';
        ctx.fillRect(0, 0, canvas.width, canvas.height);
        time++;
        // Spawn flames at base and around flag
        if (Math.random() < 0.3) {
            flames.push(new Flame(
                canvas.width / 2 + (Math.random() - 0.5) * 200,
                canvas.height / 2 + 70
            ));
        }
        // Intense flames engulfing the flag
        if (Math.random() < 0.8) {
            const flagX = canvas.width / 2 + 90;
            const flagY = canvas.height / 2;
            flames.push(new Flame(
```

```
            flagX + Math.random() * 80 - 10,
            flagY - Math.random() * 90 - 10
        ));
    }
    // Additional flames from sides of flag
    if (Math.random() < 0.5) {
        const flagX = canvas.width / 2 + 90;
        const flagY = canvas.height / 2;
        flames.push(new Flame(
            flagX + 80,
            flagY - 40 + Math.random() * 50
        ));
        flames.push(new Flame(
            flagX - 10,
            flagY - 40 + Math.random() * 50
        ));
    }
    // Update and draw flames
    for (let i = flames.length - 1; i >= 0; i--) {
        flames[i].update();
        flames[i].draw();
        if (flames[i].life <= 0) flames.splice(i, 1);
```

```
}

// Update and draw code particles

for (let i = codeParticles.length - 1; i >= 0; i--) {

    codeParticles[i].update();

    codeParticles[i].draw();

    if (codeParticles[i].life <= 0) codeParticles.splice(i, 1);

}

// Draw main elements

drawRobot();

drawFlag();

// Apply glitch effects

applyGlitch();

// Scanline effect

ctx.globalAlpha = 0.05;

for (let i = 0; i < canvas.height; i += 4) {

    ctx.fillStyle = '#fff';

    ctx.fillRect(0, i, canvas.width, 1);

}

ctx.globalAlpha = 1;

// Digital noise

if (Math.random() < 0.1) {

    ctx.fillStyle = `rgba(0, 255, 255, ${Math.random() * 0.2})`;
```

```
            ctx.fillRect(
                Math.random() * canvas.width,
                Math.random() * canvas.height,
                Math.random() * 100 + 50,
                Math.random() * 3 + 1
            );
        }
        requestAnimationFrame(animate);
    }
    animate();
    window.addEventListener('resize', () => {
        canvas.width = window.innerWidth;
        canvas.height = window.innerHeight;
    });
  </script>
</body>
</html>
```