```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
   <title>Virus Blaster</title>
   <style>
      /* Base Styles */
      * {
         margin: 0;
         padding: 0;
         box-sizing: border-box;
         touch-action: manipulation;
         user-select: none;
         -webkit-user-select: none;
      }

      body {
         overflow: hidden;
         font-family: 'Arial', sans-serif;
         position: fixed;
         width: 100%;
         height: 100%;
         background-color: #0a0a1a;
         color: #00ffaa;
         transition: background-color 0.5s ease;
      }

      body.light-mode {
         background-color: #f0f0f5;
         color: #0066ff;
      }

      /* Game Canvas */
      #game-container {
         position: relative;
         width: 100%;
         height: 100%;
         overflow: hidden;
      }

      #game-canvas {
         position: absolute;
         top: 0;
         left: 0;
         width: 100%;
         height: 100%;
```

```
    z-index: 1;
}

/* UI Elements */
#ui-container {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    pointer-events: none;
    z-index: 2;
    display: flex;
    flex-direction: column;
    padding: 15px;
}

.top-bar {
    display: flex;
    justify-content: space-between;
    width: 100%;
}

.stat-box {
    background-color: rgba(0, 0, 0, 0.7);
    border: 2px solid #00ffaa;
    border-radius: 10px;
    padding: 8px 12px;
    font-size: 14px;
    font-weight: bold;
    box-shadow: 0 0 10px #00ffaa;
    margin-bottom: 10px;
    pointer-events: auto;
    transition: all 0.3s ease;
}

.light-mode .stat-box {
    border-color: #0066ff;
    box-shadow: 0 0 10px #0066ff;
    background-color: rgba(255, 255, 255, 0.7);
}

#health-bar-container {
    width: 100%;
    height: 20px;
    background-color: rgba(0, 0, 0, 0.5);
    border: 2px solid #00ffaa;
    border-radius: 10px;
    overflow: hidden;
    margin-bottom: 10px;
    box-shadow: 0 0 10px #00ffaa;
    transition: all 0.3s ease;
}

.light-mode #health-bar-container {
    border-color: #0066ff;
    box-shadow: 0 0 10px #0066ff;
}

#health-bar {
    height: 100%;
    width: 100%;
    background-color: #00ffaa;
    transition: width 0.3s ease, background-color 0.5s ease;
}

.light-mode #health-bar {
    background-color: #0066ff;
}

#theme-toggle {
    position: absolute;
    bottom: 20px;
    right: 20px;
    width: 50px;
    height: 50px;
    border-radius: 50%;
    background-color: rgba(0, 0, 0, 0.7);
    border: 2px solid #00ffaa;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 24px;
    cursor: pointer;
    pointer-events: auto;
    box-shadow: 0 0 10px #00ffaa;
    transition: all 0.3s ease;
    z-index: 3;
}

.light-mode #theme-toggle {
    border-color: #0066ff;
    box-shadow: 0 0 10px #0066ff;
}

/* Game Over Screen */
```

```
#game-over {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.8);
    display: none;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    z-index: 10;
    pointer-events: auto;
}

.game-over-box {
    background-color: rgba(0, 0, 0, 0.9);
    border: 3px solid #ff0033;
    border-radius: 15px;
    padding: 20px;
    text-align: center;
    max-width: 80%;
    box-shadow: 0 0 20px #ff0033;
}

.light-mode .game-over-box {
    border-color: #ff3300;
    box-shadow: 0 0 20px #ff3300;
}

.game-over-title {
    font-size: 28px;
    margin-bottom: 15px;
    color: #ff0033;
    text-shadow: 0 0 10px #ff0033;
}

.light-mode .game-over-title {
    color: #ff3300;
    text-shadow: 0 0 10px #ff3300;
}

.final-stats {
    margin: 15px 0;
    font-size: 18px;
}

.restart-btn {
    background-color: #ff0033;
    border: none;
    border-radius: 10px;
    padding: 10px 20px;
    font-size: 18px;
    font-weight: bold;
    color: white;
    cursor: pointer;
    margin-top: 15px;
    transition: all 0.3s ease;
    box-shadow: 0 0 10px #ff0033;
}

.light-mode .restart-btn {
    background-color: #ff3300;
    box-shadow: 0 0 10px #ff3300;
}

.restart-btn:active {
    transform: scale(0.95);
}

/* Level Up Notification */
#level-up {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(0, 0, 0, 0.8);
    border: 3px solid #00ffaa;
    border-radius: 15px;
    padding: 15px 25px;
    font-size: 24px;
    font-weight: bold;
    text-align: center;
    display: none;
    z-index: 5;
    box-shadow: 0 0 20px #00ffaa;
    animation: pulse 1.5s infinite;
}

.light-mode #level-up {
    border-color: #0066ff;
    box-shadow: 0 0 20px #0066ff;
}

@keyframes pulse {
    0% { transform: translate(-50%, -50%) scale(1); }
    50% { transform: translate(-50%, -50%) scale(1.1); }
    100% { transform: translate(-50%, -50%) scale(1); }
}

/* Core Element */
.core {
    position: absolute;
    width: 40px;
    height: 40px;
    border-radius: 50%;
    background-color: #00ffaa;
    box-shadow: 0 0 20px #00ffaa;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 1;
    transition: all 0.5s ease;
}

.light-mode .core {
    background-color: #0066ff;
    box-shadow: 0 0 20px #0066ff;
}

.core-infected {
    animation: coreInfected 0.5s;
}

@keyframes coreInfected {
    0% { transform: translate(-50%, -50%) scale(1); }
    50% { transform: translate(-50%, -50%) scale(1.3); }
    100% { transform: translate(-50%, -50%) scale(1); }
}

/* Title Screen */
#title-screen {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.9);
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    z-index: 20;
    pointer-events: none;
```

```
}

.title-text {
   font-size: 48px;
   font-weight: bold;
   color: #00ffaa;
   text-shadow: 0 0 15px #00ffaa;
   margin-bottom: 30px;
   text-align: center;
   opacity: 0;
   transform: scale(0.5);
   transition: all 1s ease;
}

.light-mode .title-text {
   color: #0066ff;
   text-shadow: 0 0 15px #0066ff;
}

.title-text.show {
   opacity: 1;
   transform: scale(1);
}

.title-text.secondary {
   font-size: 24px;
   color: #ffffff;
   text-shadow: 0 0 10px #ffffff;
}

.light-mode .title-text.secondary {
   color: #ffffff;
   text-shadow: 0 0 10px #ffffff;
}

.start-btn {
   background-color: #00ffaa;
   border: none;
   border-radius: 10px;
   padding: 15px 30px;
   font-size: 20px;
   font-weight: bold;
   color: #000;
   cursor: pointer;
   margin-top: 30px;
   transition: all 0.3s ease;
   box-shadow: 0 0 15px #00ffaa;
   opacity: 0;
```

```
            transform: translateY(20px);
            transition: all 0.5s ease 2s;
            pointer-events: auto;
        }

        .light-mode .start-btn {
            background-color: #0066ff;
            box-shadow: 0 0 15px #0066ff;
            color: #fff;
        }

        .start-btn.show {
            opacity: 1;
            transform: translateY(0);
        }

        .start-btn:active {
            transform: scale(0.95);
        }
    </style>
</head>
<body>
    <div id="game-container">
        <canvas id="game-canvas"></canvas>
        <div class="core" id="core"></div>

        <div id="ui-container">
            <div class="top-bar">
                <div class="stat-box" id="score-box">Score: 0</div>
                <div class="stat-box" id="level-box">Level: 1</div>
            </div>
            <div id="health-bar-container">
                <div id="health-bar"></div>
            </div>
        </div>

        <div id="theme-toggle">🌓</div>

        <div id="level-up">Level Up!</div>

        <div id="game-over">
            <div class="game-over-box">
                <h2 class="game-over-title">SYSTEM INFECTED!</h2>
                <div class="final-stats">
                    <p>Final Score: <span id="final-score">0</span></p>
                    <p>Level Reached: <span id="final-level">1</span></p>
                </div>
                <button class="restart-btn" id="restart-btn">Reboot System</button>
            </div>
        </div>

        <div id="title-screen">
            <div class="title-text" id="main-title">VIRUS BLASTER</div>
            <div class="title-text secondary" id="sub-title">DEFEND THE CORE</div>
            <button class="start-btn" id="start-btn">START GAME</button>
        </div>
    </div>

    <script>
        // Game Constants
        const GAME_WIDTH = window.innerWidth;
        const GAME_HEIGHT = window.innerHeight;
        const CORE_RADIUS = 20;
        const INITIAL_HEALTH = 100;
        const HEALTH_LOSS_PER_VIRUS = 10;
        const VIRUS_SCORE_VALUE = 10;
        const LEVEL_UP_SCORE = 240;
        const INITIAL_VIRUS_SPEED = 1;
        const VIRUS_SPEED_INCREMENT = 0.1;
        const MAX_VIRUSES_ON_SCREEN = 10;
        const VIRUS_SPAWN_RATE = 1000;
        const PROJECTILE_SPEED = 8;
        const PROJECTILE_RADIUS = 5;
        const GUN_LENGTH = 40;
        const GUN_WIDTH = 10;

        // Game Variables
        let canvas, ctx;
        let score = 0;
        let level = 1;
        let health = INITIAL_HEALTH;
        let viruses = [];
        let projectiles = [];
        let virusSpeed = INITIAL_VIRUS_SPEED;
        let gameOver = false;
        let lastVirusSpawnTime = 0;
        let touchX = 0;
        let touchY = 0;
        let gunAngle = 0;
        let themeToggle = false;
        let gameStarted = false;

        // DOM Elements
        let scoreBox, levelBox, healthBar, gameOverScreen, finalScore, finalLevel;
        let restartBtn, themeToggleBtn, levelUpNotification, coreElement;
        let titleScreen, mainTitle, subTitle, startBtn;
```

```
// Initialize the game
function init() {
    // Get DOM elements
    canvas = document.getElementById('game-canvas');
    ctx = canvas.getContext('2d');
    scoreBox = document.getElementById('score-box');
    levelBox = document.getElementById('level-box');
    healthBar = document.getElementById('health-bar');
    gameOverScreen = document.getElementById('game-over');
    finalScore = document.getElementById('final-score');
    finalLevel = document.getElementById('final-level');
    restartBtn = document.getElementById('restart-btn');
    themeToggleBtn = document.getElementById('theme-toggle');
    levelUpNotification = document.getElementById('level-up');
    coreElement = document.getElementById('core');
    titleScreen = document.getElementById('title-screen');
    mainTitle = document.getElementById('main-title');
    subTitle = document.getElementById('sub-title');
    startBtn = document.getElementById('start-btn');

    // Set canvas size
    resizeCanvas();

    // Event listeners
    window.addEventListener('resize', resizeCanvas);
    canvas.addEventListener('touchstart', handleTouch);
    canvas.addEventListener('touchmove', handleTouch);
    canvas.addEventListener('mousedown', handleMouse);
    canvas.addEventListener('mousemove', handleMouse);
    restartBtn.addEventListener('click', restartGame);
    themeToggleBtn.addEventListener('click', toggleTheme);
    startBtn.addEventListener('click', startGame);

    // Show title animation
    showTitleAnimation();
}

// Show title animation
function showTitleAnimation() {
    // Show main title after short delay
    setTimeout(() => {
        mainTitle.classList.add('show');
    }, 500);

    // Show subtitle after main title appears
    setTimeout(() => {
        subTitle.classList.add('show');
    }, 1500);

    // Show start button last
    setTimeout(() => {
        startBtn.classList.add('show');
    }, 3000);
}

// Start the game
function startGame() {
    gameStarted = true;
    titleScreen.style.display = 'none';

    // Start game loop
    requestAnimationFrame(gameLoop);

    // Start virus spawn interval
    setInterval(spawnVirus, VIRUS_SPAWN_RATE);
}

// Resize canvas to fit window
function resizeCanvas() {
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
}

// Main game loop
function gameLoop(timestamp) {
    if (gameOver || !gameStarted) return;

    // Clear canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Draw background grid
    drawGrid();

    // Update and draw gun
    updateGun();
    drawGun();

    // Update and draw viruses
    updateViruses();
    drawViruses();

    // Update and draw projectiles
    updateProjectiles();
    drawProjectiles();
    // Continue game loop
    requestAnimationFrame(gameLoop);
}

// Draw background grid
function drawGrid() {
    const gridSize = 40;
    const lineWidth = 1;
    const color = themeToggle ? 'rgba(0, 102, 255, 0.1)' : 'rgba(0, 255, 170, 0.1)';

    ctx.strokeStyle = color;
    ctx.lineWidth = lineWidth;

    // Vertical lines
    for (let x = 0; x < canvas.width; x += gridSize) {
        ctx.beginPath();
        ctx.moveTo(x, 0);
        ctx.lineTo(x, canvas.height);
        ctx.stroke();
    }

    // Horizontal lines
    for (let y = 0; y < canvas.height; y += gridSize) {
        ctx.beginPath();
        ctx.moveTo(0, y);
        ctx.lineTo(canvas.width, y);
        ctx.stroke();
    }
}

// Update gun position based on touch/mouse
function updateGun() {
    const centerX = canvas.width / 2;
    const centerY = canvas.height / 2;
    gunAngle = Math.atan2(touchY - centerY, touchX - centerX);
}

// Draw the gun
function drawGun() {
    const centerX = canvas.width / 2;
    const centerY = canvas.height / 2;

    ctx.save();
    ctx.translate(centerX, centerY);
    ctx.rotate(gunAngle);

    // Gun barrel
    ctx.fillStyle = themeToggle ? '#0066ff' : '#00ffaa';
```

```
        ctx.shadowColor = themeToggle ? '#0066ff' : '#00ffaa';
        ctx.shadowBlur = 10;
        ctx.fillRect(0, -GUN_WIDTH / 2, GUN_LENGTH, GUN_WIDTH);

        // Gun base (circular)
        ctx.beginPath();
        ctx.arc(0, 0, GUN_WIDTH * 1.5, 0, Math.PI * 2);
        ctx.fill();

        ctx.restore();
      }

      // Spawn a new virus
      function spawnVirus() {
        if (gameOver || viruses.length >= MAX_VIRUSES_ON_SCREEN || !gameStarted)
return;

        // Determine spawn edge (0: top, 1: right, 2: bottom, 3: left)
        const edge = Math.floor(Math.random() * 4);
        let x, y;

        switch (edge) {
          case 0: // top
            x = Math.random() * canvas.width;
            y = -30;
            break;
          case 1: // right
            x = canvas.width + 30;
            y = Math.random() * canvas.height;
            break;
          case 2: // bottom
            x = Math.random() * canvas.width;
            y = canvas.height + 30;
            break;
          case 3: // left
            x = -30;
            y = Math.random() * canvas.height;
            break;
        }

        // Create virus with random type (shape)
        const virusType = Math.floor(Math.random() * 3);
        const size = 20 + Math.random() * 10;

        viruses.push({
          x: x,
          y: y,
          size: size,
        type: virusType,
        speed: virusSpeed,
        angle: Math.atan2(canvas.height / 2 - y, canvas.width / 2 - x),
        rotation: Math.random() * Math.PI * 2,
        rotationSpeed: (Math.random() - 0.5) * 0.1,
        pulsePhase: Math.random() * Math.PI * 2,
        pulseSpeed: 0.05 + Math.random() * 0.05
    });
}

// Update all viruses
function updateViruses() {
    const centerX = canvas.width / 2;
    const centerY = canvas.height / 2;

    for (let i = viruses.length - 1; i >= 0; i--) {
        const virus = viruses[i];

        // Move virus toward center
        virus.x += Math.cos(virus.angle) * virus.speed;
        virus.y += Math.sin(virus.angle) * virus.speed;
        virus.rotation += virus.rotationSpeed;
        virus.pulsePhase += virus.pulseSpeed;

        // Check if virus reached core
        const dx = virus.x - centerX;
        const dy = virus.y - centerY;
        const distance = Math.sqrt(dx * dx + dy * dy);

        if (distance < CORE_RADIUS + virus.size / 2) {
            // Virus infected the core
            viruses.splice(i, 1);
            decreaseHealth();
            coreElement.classList.add('core-infected');
            setTimeout(() => {
                coreElement.classList.remove('core-infected');
            }, 500);
            continue;
        }

        // Check if virus is out of bounds (shouldn't happen normally)
        if (virus.x < -100 || virus.x > canvas.width + 100 ||
            virus.y < -100 || virus.y > canvas.height + 100) {
            viruses.splice(i, 1);
        }
    }
}
```

```
// Draw all viruses
function drawViruses() {
    viruses.forEach(virus => {
        ctx.save();
        ctx.translate(virus.x, virus.y);
        ctx.rotate(virus.rotation);

        // Pulse effect
        const pulseScale = 1 + Math.sin(virus.pulsePhase) * 0.1;
        ctx.scale(pulseScale, pulseScale);

        // Virus glow
        ctx.shadowColor = themeToggle ? '#0066ff' : '#00ffaa';
        ctx.shadowBlur = 15;

        // Draw different virus types
        switch (virus.type) {
            case 0: // Spiky virus
                drawSpikyVirus(virus.size);
                break;
            case 1: // Round virus
                drawRoundVirus(virus.size);
                break;
            case 2: // Square virus
                drawSquareVirus(virus.size);
                break;
        }

        ctx.restore();
    });
}

// Draw spiky virus shape
function drawSpikyVirus(size) {
    const spikes = 8;
    const spikeLength = size * 0.4;

    ctx.fillStyle = themeToggle ? 'rgba(0, 102, 255, 0.8)' : 'rgba(0, 255, 170, 0.8)';

    ctx.beginPath();
    for (let i = 0; i < spikes; i++) {
        const angle = (i / spikes) * Math.PI * 2;
        const outerX = Math.cos(angle) * (size / 2 + spikeLength);
        const outerY = Math.sin(angle) * (size / 2 + spikeLength);
        const innerX = Math.cos(angle) * (size / 2);
        const innerY = Math.sin(angle) * (size / 2);

        if (i === 0) {
            ctx.moveTo(outerX, outerY);
        } else {
            ctx.lineTo(outerX, outerY);
        }

        ctx.lineTo(innerX, innerY);
    }
    ctx.closePath();
    ctx.fill();

    // Center circle
    ctx.beginPath();
    ctx.arc(0, 0, size / 3, 0, Math.PI * 2);
    ctx.fill();
}

// Draw round virus shape
function drawRoundVirus(size) {
    const dots = 6;
    const dotSize = size * 0.1;

    ctx.fillStyle = themeToggle ? 'rgba(0, 102, 255, 0.8)' : 'rgba(0, 255, 170, 0.8)';

    // Main circle
    ctx.beginPath();
    ctx.arc(0, 0, size / 2, 0, Math.PI * 2);
    ctx.fill();

    // Dots
    for (let i = 0; i < dots; i++) {
        const angle = (i / dots) * Math.PI * 2;
        const distance = size / 3;
        const x = Math.cos(angle) * distance;
        const y = Math.sin(angle) * distance;

        ctx.beginPath();
        ctx.arc(x, y, dotSize, 0, Math.PI * 2);
        ctx.fill();
    }
}

// Draw square virus shape
function drawSquareVirus(size) {
    const corners = 4;
    const cornerSize = size * 0.3;

    ctx.fillStyle = themeToggle ? 'rgba(0, 102, 255, 0.8)' : 'rgba(0, 255, 170, 0.8)';
    // Main square
    ctx.beginPath();
    ctx.rect(-size / 2, -size / 2, size, size);
    ctx.fill();

    // Corners
    for (let i = 0; i < corners; i++) {
        const angle = (i / corners) * Math.PI * 2;
        const distance = size / 2 + cornerSize / 2;
        const x = Math.cos(angle) * distance;
        const y = Math.sin(angle) * distance;

        ctx.save();
        ctx.translate(x, y);
        ctx.rotate(angle + Math.PI / 4);
        ctx.beginPath();
        ctx.rect(-cornerSize / 2, -cornerSize / 2, cornerSize, cornerSize);
        ctx.fill();
        ctx.restore();
    }
}

// Handle touch input
function handleTouch(e) {
    e.preventDefault();
    const touch = e.touches[0] || e.changedTouches[0];
    touchX = touch.clientX;
    touchY = touch.clientY;

    // Fire projectile on touch start
    if (e.type === 'touchstart' && !gameOver && gameStarted) {
        fireProjectile();
    }
}

// Handle mouse input
function handleMouse(e) {
    touchX = e.clientX;
    touchY = e.clientY;

    // Fire projectile on mouse down
    if (e.type === 'mousedown' && !gameOver && gameStarted) {
        fireProjectile();
    }
}

// Fire a projectile
function fireProjectile() {
    const centerX = canvas.width / 2;
    const centerY = canvas.height / 2;

    projectiles.push({
        x: centerX,
        y: centerY,
        angle: gunAngle,
        speed: PROJECTILE_SPEED,
        radius: PROJECTILE_RADIUS,
        distance: 0,
        maxDistance: Math.min(canvas.width, canvas.height) * 0.7
    });
}

// Update all projectiles
function updateProjectiles() {
    for (let i = projectiles.length - 1; i >= 0; i--) {
        const proj = projectiles[i];

        // Move projectile
        proj.x += Math.cos(proj.angle) * proj.speed;
        proj.y += Math.sin(proj.angle) * proj.speed;
        proj.distance += proj.speed;

        // Check if projectile reached max distance
        if (proj.distance >= proj.maxDistance) {
            projectiles.splice(i, 1);
            continue;
        }

        // Check for collisions with viruses
        for (let j = viruses.length - 1; j >= 0; j--) {
            const virus = viruses[j];
            const dx = proj.x - virus.x;
            const dy = proj.y - virus.y;
            const distance = Math.sqrt(dx * dx + dy * dy);

            if (distance < proj.radius + virus.size / 2) {
                // Hit a virus
                viruses.splice(j, 1);
                projectiles.splice(i, 1);
                increaseScore();
                checkLevelUp();
                break;
            }
        }
    }
}
```

```
        // Draw all projectiles
        function drawProjectiles() {
            projectiles.forEach(proj => {
                ctx.save();
                ctx.translate(proj.x, proj.y);

                // Projectile glow
                ctx.shadowColor = themeToggle ? '#0066ff' : '#00ffaa';
                ctx.shadowBlur = 10;

                // Projectile trail
                const gradient = ctx.createRadialGradient(0, 0, 0, 0, 0, proj.radius);
                gradient.addColorStop(0, themeToggle ? 'rgba(0, 102, 255, 1)' : 'rgba(0, 255, 170,
1)');
                gradient.addColorStop(1, themeToggle ? 'rgba(0, 102, 255, 0)' : 'rgba(0, 255, 170,
0)');

                ctx.fillStyle = gradient;
                ctx.beginPath();
                ctx.arc(0, 0, proj.radius, 0, Math.PI * 2);
                ctx.fill();

                ctx.restore();
            });
        }

        // Increase player score
        function increaseScore() {
            score += VIRUS_SCORE_VALUE;
            scoreBox.textContent = `Score: ${score}`;
        }

        // Decrease player health
        function decreaseHealth() {
            health = Math.max(0, health - HEALTH_LOSS_PER_VIRUS);
            healthBar.style.width = `${health}%`;

            // Change health bar color based on health
            if (health < 30) {
                healthBar.style.backgroundColor = '#ff0033';
            } else if (health < 60) {
                healthBar.style.backgroundColor = '#ff9900';
            } else {
                healthBar.style.backgroundColor = themeToggle ? '#0066ff' : '#00ffaa';
            }

            // Check for game over
```

```
    if (health <= 0) {
        endGame();
    }
}

// Check if player should level up
function checkLevelUp() {
    if (score >= level * LEVEL_UP_SCORE) {
        levelUp();
    }
}

// Level up the player
function levelUp() {
    level++;
    levelBox.textContent = `Level: ${level}`;
    virusSpeed += VIRUS_SPEED_INCREMENT;

    // Show level up notification
    levelUpNotification.style.display = 'block';
    setTimeout(() => {
        levelUpNotification.style.display = 'none';
    }, 1500);
}

// End the game
function endGame() {
    gameOver = true;
    finalScore.textContent = score;
    finalLevel.textContent = level;
    gameOverScreen.style.display = 'flex';
}

// Restart the game
function restartGame() {
    // Reset game state
    score = 0;
    level = 1;
    health = INITIAL_HEALTH;
    viruses = [];
    projectiles = [];
    virusSpeed = INITIAL_VIRUS_SPEED;
    gameOver = false;

    // Reset UI
    scoreBox.textContent = `Score: ${score}`;
    levelBox.textContent = `Level: ${level}`;
    healthBar.style.width = `${health}%`;
```

```
            healthBar.style.backgroundColor = themeToggle ? '#0066ff' : '#00ffaa';
            gameOverScreen.style.display = 'none';

            // Restart game loop
            requestAnimationFrame(gameLoop);
        }

        // Toggle between light and dark theme
        function toggleTheme() {
            themeToggle = !themeToggle;
            document.body.classList.toggle('light-mode');

            // Update health bar color if needed
            if (health >= 60) {
                healthBar.style.backgroundColor = themeToggle ? '#0066ff' : '#00ffaa';
            }
        }

        // Start the game when the page loads
        window.onload = init;
    </script>
</body>
</html>
```