```
<!DOCTYPE html>
<html>
<head>
    <title>CEC Blue Dragon</title>
    <style>
        /* Main container styling */
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            background-color: #1a1a1a;
            color: white;
            font-family: 'Arial', sans-serif;
            margin: 0;
            height: 100vh;
        }

        /* Main menu styling */
        #mainMenu {
            text-align: center;
            margin-top: 50px;
        }

        /* Options menu styling */
        #optionsMenu {
            display: none;
            position: absolute;
            background: rgba(0, 0, 0, 0.9);
            padding: 20px;
            border-radius: 10px;
        }

        /* Game container hidden initially */
        #gameContainer {
            display: none;
            position: relative;
        }

        /* Game canvas styling */
        canvas {
            border: 3px solid #2962FF;
            border-radius: 10px;
            box-shadow: 0 0 20px rgba(41, 98, 255, 0.3);
        }
```

```
/* Common button styling */
.menuButton {
    background: linear-gradient(45deg, #2962FF, #00B0FF);
    border: none;
    color: white;
    padding: 15px 32px;
    margin: 10px;
    cursor: pointer;
    border-radius: 25px;
    width: 200px;
    transition: transform 0.3s;
}

.menuButton:hover {
    transform: scale(1.05);
}

/* Difficulty buttons styling */
.difficultyButton {
    width: 150px;
    padding: 10px;
    margin: 5px;
}

/* Score display styling */
#score {
    font-size: 24px;
    margin: 10px;
    color: #00B0FF;
}

/* Game over overlay styling */
#gameOver {
    display: none;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: rgba(0, 0, 0, 0.9);
    padding: 30px;
    border-radius: 15px;
    border: 2px solid #2962FF;
    text-align: center;
```

```
    }
  </style>
</head>
<body>
  <!-- Game title -->
  <h1 style="color: #00B0FF; text-shadow: 0 0 10px rgba(0, 176, 255, 0.5);">CEC Blue
Dragon</h1>

  <!-- Main menu -->
  <div id="mainMenu">
    <button class="menuButton" id="playButton">Play</button>
    <button class="menuButton" id="optionsButton">Options</button>
    <button class="menuButton" id="infoButton">Info</button>
    <button class="menuButton" id="exitButton">Exit</button>
  </div>

  <!-- Options menu -->
  <div id="optionsMenu">
    <button class="menuButton difficultyButton" data-difficulty="easy">Easy</button>
    <button class="menuButton difficultyButton" data-difficulty="medium">Medium</button>
    <button class="menuButton difficultyButton" data-difficulty="hard">Hard</button>
    <button class="menuButton difficultyButton" data-difficulty="extreme">Extreme</button>
  </div>

  <!-- Game container -->
  <div id="gameContainer">
    <div id="score">Score: 0</div>
    <canvas id="gameCanvas" width="600" height="400"></canvas>
    <div id="gameOver">
      <h2 style="color: #FF4081">Game Over!</h2>
      <p>Press Enter to play again</p>
      <button class="menuButton" id="quitButton">Quit</button>
    </div>
  </div>

  <!-- Sound effects -->
  <audio id="eatSound"
src="data:audio/wav;base64,UklGRl9vT19XQVZFZm10IBAAAAABAAEAQB8AAEAfAAABAAgA
ZGF0YU"></audio>

  <script>
    // Game Constants
    const GRID_SIZE = 10;
    const INITIAL_SPEEDS = {
```

```
    easy: 150,
    medium: 130,
    hard: 100,
    extreme: 75
};
let currentDifficulty = 'easy';

// Game Elements
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
const eatSound = document.getElementById('eatSound');

// Game State
let dragon = [];
let food = {};
let direction = 'right';
let score = 0;
let gameSpeed = INITIAL_SPEEDS[currentDifficulty];
let gameLoop;
let mouthOpen = 0; // For mouth animation

// Initialize game state
function initGame() {
    dragon = [
        {x: 300, y: 200},
        {x: 290, y: 200},
        {x: 280, y: 200}
    ];
    direction = 'right';
    score = 0;
    mouthOpen = 0;
    document.getElementById('score').textContent = `Score: ${score}`;
    spawnFood();
}

// Generate new food position
function spawnFood() {
    food = {
        x: Math.floor(Math.random() * (canvas.width / GRID_SIZE)) * GRID_SIZE,
        y: Math.floor(Math.random() * (canvas.height / GRID_SIZE)) * GRID_SIZE
    };
    // Ensure food doesn't spawn on dragon
    dragon.forEach(segment => {
        if (segment.x === food.x && segment.y === food.y) spawnFood();
```

```
    });
}

// Update game speed based on difficulty and score
function updateGameSpeed() {
    switch(currentDifficulty) {
        case 'medium':
            gameSpeed = Math.max(50, INITIAL_SPEEDS.medium - dragon.length);
            break;
        case 'hard':
            gameSpeed = Math.max(30, INITIAL_SPEEDS.hard - (score * 0.5));
            break;
        case 'extreme':
            gameSpeed = Math.max(10, INITIAL_SPEEDS.extreme - (score * 1));
            break;
    }
}

// Move dragon and check collisions
function moveDragon() {
    updateGameSpeed();
    const head = {x: dragon[0].x, y: dragon[0].y};

    // Update head position based on direction
    switch(direction) {
        case 'right': head.x += GRID_SIZE; break;
        case 'left': head.x -= GRID_SIZE; break;
        case 'up': head.y -= GRID_SIZE; break;
        case 'down': head.y += GRID_SIZE; break;
    }

    // Wall wrapping
    head.x = (head.x + canvas.width) % canvas.width;
    head.y = (head.y + canvas.height) % canvas.height;

    // Check self-collision
    if (dragon.some(segment => segment.x === head.x && segment.y === head.y)) {
        gameOver();
        return;
    }

    dragon.unshift(head);

    // Check food collision
```

```
    if (head.x === food.x && head.y === food.y) {
        score += 10;
        document.getElementById('score').textContent = `Score: ${score}`;
        eatSound.play();
        mouthOpen = 1; // Trigger mouth animation
        spawnFood();
    } else {
        dragon.pop();
        mouthOpen = Math.sin(Date.now() / 100) * 0.5 + 0.5; // Animate mouth
    }
}

// Draw dragon head with animated mouth
function drawDragonHead(x, y) {
    ctx.save();
    ctx.translate(x + GRID_SIZE/2, y + GRID_SIZE/2);

    // Rotate head based on direction
    switch(direction) {
        case 'left': ctx.rotate(Math.PI); break;
        case 'up': ctx.rotate(-Math.PI/2); break;
        case 'down': ctx.rotate(Math.PI/2); break;
    }

    // Head shape
    ctx.beginPath();
    ctx.moveTo(0, 0);
    ctx.bezierCurveTo(15, -15, 25, -15, 30, 0);
    ctx.bezierCurveTo(25, 15, 15, 15, 0, 0);
    ctx.fillStyle = '#2196F3';
    ctx.fill();

    // Animated mouth
    ctx.beginPath();
    ctx.moveTo(-5 * mouthOpen, 5 * mouthOpen);
    ctx.lineTo(10, 0);
    ctx.lineTo(-5 * mouthOpen, -5 * mouthOpen);
    ctx.strokeStyle = '#FF4081';
    ctx.lineWidth = 2;
    ctx.stroke();

    // Horns and eyes (remaining same as previous)
    // ... (horn and eye drawing code from previous example)
```

```
    ctx.restore();
}

// Main draw function
function draw() {
    ctx.fillStyle = '#1a1a1a';
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    // Draw dragon
    dragon.forEach((segment, index) => {
        if(index === 0) drawDragonHead(segment.x, segment.y);
        else {
            // Body drawing code from previous example
        }
    });

    // Draw food
    ctx.fillStyle = '#FF4081';
    ctx.beginPath();
    ctx.arc(food.x + GRID_SIZE/2, food.y + GRID_SIZE/2, GRID_SIZE/2, 0, Math.PI * 2);
    ctx.fill();
}

// Event Listeners and remaining code
document.addEventListener('keydown', (e) => {
    // Movement controls same as before
});

// Button Handlers
document.getElementById('optionsButton').addEventListener('click', () => {
    document.getElementById('mainMenu').style.display = 'none';
    document.getElementById('optionsMenu').style.display = 'block';
});

document.querySelectorAll('.difficultyButton').forEach(button => {
    button.addEventListener('click', (e) => {
        currentDifficulty = e.target.dataset.difficulty;
        gameSpeed = INITIAL_SPEEDS[currentDifficulty];
        document.getElementById('optionsMenu').style.display = 'none';
        document.getElementById('mainMenu').style.display = 'block';
    });
});

// Other button handlers and game loop management
```

```
        // ... (remaining code from previous example)
    </script>
</body>
</html>
```