<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Classic Snake Game</title>

<script src="https://cdn.tailwindcss.com"></script>

<style>

#game-board {

border: 2px solid #4a5568;

box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);

}

.snake-segment {

border-radius: 3px;

transition: all 0.1s ease;

}

.food {

animation: pulse 1s infinite;

border-radius: 50%;

}

@keyframes pulse {

0% { transform: scale(0.95); }

50% { transform: scale(1.05); }

100% { transform: scale(0.95); }

}

.game-over {

animation: shake 0.5s;

}

@keyframes shake {

0% { transform: translateX(0); }

25% { transform: translateX(-5px); }

50% { transform: translateX(5px); }

75% { transform: translateX(-5px); }

100% { transform: translateX(0); }

}

</style>

</head>

<body class="bg-gray-100 min-h-screen flex flex-col items-center justify-center p-4">

<div class="text-center mb-8">

<h1 class="text-4xl font-bold text-gray-800 mb-2">Classic Snake Game</h1>

<p class="text-gray-600 mb-4">Use arrow keys to control the snake. Eat the food to grow!</p>

<div class="flex justify-center gap-8 mb-6">

<div class="bg-white p-4 rounded-lg shadow">

<p class="text-gray-500">Score</p>

<p id="score" class="text-2xl font-bold text-green-600">0</p>

</div>

<div class="bg-white p-4 rounded-lg shadow">

<p class="text-gray-500">High Score</p>

<p id="high-score" class="text-2xl font-bold text-purple-600">0</p>

</div>

</div>

<button id="start-btn" class="bg-green-500 hover:bg-green-600 text-white font-bold py-2 px-6 rounded-full shadow-lg transition duration-300">

Start Game

</button>

</div>

<div id="game-container" class="relative">

<canvas id="game-board" width="400" height="400" class="bg-white rounded-lg"></canvas>

<div id="game-over" class="absolute inset-0 flex flex-col items-center justify-center bg-black bg-opacity-70 rounded-lg hidden">

<h2 class="text-4xl font-bold text-red-500 mb-4">Game Over!</h2>

<p class="text-xl text-white mb-6">Your score: <span id="final-score" class="font-bold">0</span></p>

<button id="restart-btn" class="bg-green-500 hover:bg-green-600 text-white font-bold py-2 px-6 rounded-full shadow-lg transition duration-300">

Play Again

</button>

</div>

</div>

<div class="mt-8 text-center text-gray-500">

<p class="mb-2">Controls:</p>

<div class="flex justify-center gap-2">

<div class="bg-gray-200 p-2 rounded">↑</div>

<div class="bg-gray-200 p-2 rounded">↓</div>

<div class="bg-gray-200 p-2 rounded">←</div>

<div class="bg-gray-200 p-2 rounded">→</div>

</div>

</div>

<script>

document.addEventListener('DOMContentLoaded', () => {

const canvas = document.getElementById('game-board');

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

const scoreDisplay = document.getElementById('score');

const highScoreDisplay = document.getElementById('high-score');

const startBtn = document.getElementById('start-btn');

const restartBtn = document.getElementById('restart-btn');

const gameOverDisplay = document.getElementById('game-over');

const finalScoreDisplay = document.getElementById('final-score');

const gridSize = 20;

const tileCount = canvas.width / gridSize;

let highScore = localStorage.getItem('snakeHighScore') || 0;

highScoreDisplay.textContent = highScore;

let snake = [];

let food = {};

let direction = 'right';

let nextDirection = 'right';

let score = 0;

let gameSpeed = 150;

let gameRunning = false;

let gameLoop;

function initGame() {

snake = [

{x: 5, y: 10},

{x: 4, y: 10},

{x: 3, y: 10}

];

direction = 'right';

nextDirection = 'right';

score = 0;

scoreDisplay.textContent = score;

gameSpeed = 150;

generateFood();

gameRunning = true;

if (gameLoop) clearInterval(gameLoop);

gameLoop = setInterval(gameStep, gameSpeed);

gameOverDisplay.classList.add('hidden');

}

function generateFood() {

food = {

x: Math.floor(Math.random() \* tileCount),

y: Math.floor(Math.random() \* tileCount)

};

// Make sure food doesn't appear on snake

for (let segment of snake) {

if (segment.x === food.x && segment.y === food.y) {

return generateFood();

}

}

}

function gameStep() {

if (!gameRunning) return;

direction = nextDirection;

// Calculate new head position

const head = {x: snake[0].x, y: snake[0].y};

switch(direction) {

case 'up':

head.y--;

break;

case 'down':

head.y++;

break;

case 'left':

head.x--;

break;

case 'right':

head.x++;

break;

}

// Check for collisions

if (

head.x < 0 || head.x >= tileCount ||

head.y < 0 || head.y >= tileCount ||

snake.some(segment => segment.x === head.x && segment.y === head.y)

) {

gameOver();

return;

}

// Add new head

snake.unshift(head);

// Check if snake ate food

if (head.x === food.x && head.y === food.y) {

score++;

scoreDisplay.textContent = score;

// Increase speed slightly every 5 points

if (score % 5 === 0) {

gameSpeed = Math.max(50, gameSpeed - 5);

clearInterval(gameLoop);

gameLoop = setInterval(gameStep, gameSpeed);

}

generateFood();

} else {

// Remove tail if no food was eaten

snake.pop();

}

drawGame();

}

function drawGame() {

// Clear canvas

ctx.fillStyle = 'white';

ctx.fillRect(0, 0, canvas.width, canvas.height);

// Draw grid (optional)

ctx.strokeStyle = '#f0f0f0';

ctx.lineWidth = 0.5;

for (let i = 0; i < tileCount; i++) {

ctx.beginPath();

ctx.moveTo(i \* gridSize, 0);

ctx.lineTo(i \* gridSize, canvas.height);

ctx.stroke();

ctx.beginPath();

ctx.moveTo(0, i \* gridSize);

ctx.lineTo(canvas.width, i \* gridSize);

ctx.stroke();

}

// Draw snake

snake.forEach((segment, index) => {

// Head is a different color

if (index === 0) {

ctx.fillStyle = '#4CAF50'; // Green head

} else {

// Gradient body from dark to light green

const intensity = 150 + Math.floor(105 \* (index / snake.length));

ctx.fillStyle = `rgb(50, ${intensity}, 50)`;

}

ctx.fillRect(

segment.x \* gridSize + 1,

segment.y \* gridSize + 1,

gridSize - 2,

gridSize - 2

);

// Add eyes to head

if (index === 0) {

ctx.fillStyle = 'white';

// Eye positions based on direction

let leftEyeX, leftEyeY, rightEyeX, rightEyeY;

switch(direction) {

case 'up':

leftEyeX = segment.x \* gridSize + 5;

leftEyeY = segment.y \* gridSize + 5;

rightEyeX = segment.x \* gridSize + gridSize - 8;

rightEyeY = segment.y \* gridSize + 5;

break;

case 'down':

leftEyeX = segment.x \* gridSize + 5;

leftEyeY = segment.y \* gridSize + gridSize - 8;

rightEyeX = segment.x \* gridSize + gridSize - 8;

rightEyeY = segment.y \* gridSize + gridSize - 8;

break;

case 'left':

leftEyeX = segment.x \* gridSize + 5;

leftEyeY = segment.y \* gridSize + 5;

rightEyeX = segment.x \* gridSize + 5;

rightEyeY = segment.y \* gridSize + gridSize - 8;

break;

case 'right':

leftEyeX = segment.x \* gridSize + gridSize - 8;

leftEyeY = segment.y \* gridSize + 5;

rightEyeX = segment.x \* gridSize + gridSize - 8;

rightEyeY = segment.y \* gridSize + gridSize - 8;

break;

}

ctx.beginPath();

ctx.arc(leftEyeX, leftEyeY, 2, 0, Math.PI \* 2);

ctx.fill();

ctx.beginPath();

ctx.arc(rightEyeX, rightEyeY, 2, 0, Math.PI \* 2);

ctx.fill();

}

});

// Draw food

ctx.fillStyle = '#FF5252';

ctx.beginPath();

ctx.arc(

food.x \* gridSize + gridSize / 2,

food.y \* gridSize + gridSize / 2,

gridSize / 2 - 2,

0,

Math.PI \* 2

);

ctx.fill();

}

function gameOver() {

gameRunning = false;

clearInterval(gameLoop);

// Update high score

if (score > highScore) {

highScore = score;

localStorage.setItem('snakeHighScore', highScore);

highScoreDisplay.textContent = highScore;

}

finalScoreDisplay.textContent = score;

gameOverDisplay.classList.remove('hidden');

gameOverDisplay.classList.add('game-over');

// Remove shake animation after it's done

setTimeout(() => {

gameOverDisplay.classList.remove('game-over');

}, 500);

}

// Event listeners

startBtn.addEventListener('click', () => {

initGame();

drawGame();

startBtn.textContent = 'Restart Game';

});

restartBtn.addEventListener('click', () => {

initGame();

drawGame();

});

document.addEventListener('keydown', (e) => {

// Prevent opposite direction movement

switch(e.key) {

case 'ArrowUp':

if (direction !== 'down') nextDirection = 'up';

break;

case 'ArrowDown':

if (direction !== 'up') nextDirection = 'down';

break;

case 'ArrowLeft':

if (direction !== 'right') nextDirection = 'left';

break;

case 'ArrowRight':

if (direction !== 'left') nextDirection = 'right';

break;

case ' ':

if (!gameRunning) initGame();

break;

}

});

// Initial draw

drawGame();

});

</script>

</body>

</html>