Snake Game Tech-Specs

Manos Vourgourakis

1. Architecture:

The application is structured as a single-page web application (SPA) with three primary screens:

- **Start Screen:** Displays configuration options and a button to start the game.
- **Game Screen:** The actual game play area with a canvas that updates based on the game logic.
- **Game Over Screen:** Displays the final score and a button to restart the game.

---

2. Technologies:

- **HTML:** Used for the structure of the page, including the start screen, game screen, and game over screen.
- **CSS:** Used for styling the UI elements such as buttons, text, and the canvas.
- **JavaScript (ES6):** Manages the game logic, including handling user input (keyboard events), game state (food, snake, obstacles), and drawing to the canvas.

---

3. UI Components:

- **Start Screen:**
    - Input for the number of food items (<input> of type number).
    - Dropdown for selecting the game speed (<select> with options for slow, regular, and fast).
    - Dropdown for selecting the snake color (<select> with color options).
    - Dropdown for selecting the food color (<select> with color options).
    - Dropdown for selecting the game mode (classic or with obstacles).
    - Button to start the game.
- **Game Screen:**
    - A score display showing the current score.
    - A canvas where the game is drawn, with the snake, food, and obstacles.
- **Game Over Screen:**
    - Displays a message indicating the game is over.
    - Shows the final score.
    - A button to restart the game.

---

4. Game Logic:

Main Concepts:

- **Snake:**
  - The snake is represented as an array of objects, where each object represents a segment of the snake (with x and y coordinates).
  - The snake moves by adding a new head at the front of the array and removing the last segment unless food is eaten.
- **Food:**
  - Food items are represented as an array of objects with x and y coordinates.
  - When the snake collides with food, the score is incremented, and a new food item is placed on the canvas.
- **Obstacles:**
  - If the game mode is set to "obstacles", a number of obstacles are randomly placed on the canvas.
  - The snake must avoid colliding with obstacles, or the game ends.
- **Game Speed:**
  - The game speed is adjustable via the gameSpeed variable, which defines the interval (in milliseconds) for the game loop.

Game Phases:

- **Initialization:**
  - On clicking the "Start Game" button, the game initializes based on the user-selected settings. The game state (snake, food, obstacles, score) is reset.
- **Game Loop:**
  - The drawGame function is executed repeatedly based on the interval set by the selected game speed. It handles:
    - Drawing the snake and food on the canvas.
    - Checking for food consumption and updating the score.
    - Handling snake movement and direction based on keyboard input.
    - Detecting collisions with walls, snake itself, or obstacles.
- **Game Over:**
  - The game ends when the snake collides with the canvas boundary, itself, or an obstacle. The game interval is cleared, and the final score is displayed.
- **Restarting:**
  - The page reloads when the player presses the "Play Again" button on the game over screen.

---

5. Game Features:

Customizable Settings:

- **Food Count:** Users can set the number of food items (1–10).
- **Game Speed:** Users can choose the speed of the game (slow, regular, fast).
- **Snake Color:** Users can pick a color for the snake (Red, Orange, Yellow, Green, Blue, Purple, Pink).
- **Food Color:** Users can choose the color for food (Red, Orange, Yellow, Green, Blue, Purple, Pink).
- **Game Mode:** Users can select between two game modes:
    - **Classic:** No obstacles.
    - **Obstacles:** Adds obstacles to the game, which the snake must avoid.

Canvas & Background:

- The game is rendered on a <canvas> element (400x400px).
- The background is a light blue checkerboard pattern with alternating shades of blue (#ADD8E6 and #B0E0E6).

The background is drawn by filling squares in a grid pattern. Each square is either one of the two light blue shades, depending on its position on the grid.

---

6. JavaScript Functions:

Initialization:

- startGame(): Initializes game settings, creates food items and obstacles (if applicable), and starts the game loop.
- generateFoodItems(count): Generates a specified number of food items at random positions on the canvas.
- generateObstacles(): Generates 14 random obstacles in the game space if the game mode is set to "obstacles".

Game Loop:

- drawGame(): This function clears the canvas and redraws all game elements:
    - Background (checkerboard pattern).
    - Obstacles (if in "obstacles" mode).
    - Food items.
    - Snake (in its current position).
    - Checks for collisions with food, walls, self, or obstacles.

User Input:

- changeDirection(event): Listens for keyboard arrow key events and updates the direction of the snake.

End Game:

- endGame(): Stops the game, displays the game over screen, and shows the final score.

Restart Game:

- restartGame(): Reloads the page to restart the game.

---

7. Error Handling and Edge Cases:

- **Invalid Input:** The game uses basic input validation by restricting food count to a range between 1 and 10, and ensures valid key events for changing direction.
- **Collisions:** The game checks for collisions with the walls, snake itself, and obstacles, ending the game when a collision occurs.

---

8. Performance Considerations:

- The game loop is controlled by the setInterval() function, which executes the drawGame() function at intervals based on the game speed.
- The canvas is cleared and redrawn every frame, and only the game elements (snake, food, obstacles) are redrawn each time to maintain performance.

---

9. User Interface/UX Considerations:

- Clear instructions on the start screen for selecting game settings.
- Score is displayed throughout the game.
- The game over screen shows the final score and provides an option to restart.
- The game's difficulty can be adjusted based on speed, food count, and the presence of obstacles.