**Snake Game Documentation**

V1.0

Danfe Creatives

1. Introduction

This snake game will be a clone of popular retro videogame concept where a user controls a linear object (Snake) in an attempt to gather Fruit and score points. Every time the Snake gathers the fruit, it will grow in size and get the specified number of points. The Snake itself and a designated bounding box for the game are the primary obstacles. The game will end when the snake collides with itself or the boundary of the bounding box.

This game will be browser based and will be developed using vanilla JavaScript, HTML5 and CSS3.

1. Game Objects
   1. Game Board
   2. Game Board Design

* A rectangular game board should appear on screen which will serve as the bounding box and holds all other game objects such as snake and fruits.
* Create a div with id “game-board” in the html file and add CSS attributes like borders and position to it.
* The dimension of the game board should be in multiple of the unit snake body dimension.
  1. Snake

1. Snake Design

* A linear object representing a snake should appear on the screen.
* Create three div’s within the “game-board” div and give them a CSS class of “snake-body”. Aggregating all the snake-body div’s will create a snake.
* Each snake-body div should be square shaped with dimension 24px in width and 24px in height.
  1. Snake Movement
* Use arrow keys to control the movement of the snake object. Snake should move in the specified direction by detecting the arrow key that is being pressed.
* Create a function called move(direction) which takes a direction as a parameter and moves the snake in that specified direction. Change the top and left position of each snake body of the snake to move the snake.
* The snake should keep on moving in the specified direction unless another direction key is pressed or it collides with the game-board boundary.
* Snake should not be allowed to move in the direction that is directly opposite to its current direction. For example, if the snake is moving in upwards direction, pressing the down arrow key should not move the snake downwards.
  1. Snake growing in size
* Each time the snake collides with the fruit, the size of snake should increase by a unit snake body dimension.
* Append a new child div with class snake-body inside the game-board div. Since the snake is an aggregation of all the div’s with snake-body class, it will appear that the snake has grown in size.
  1. Fruit

1. Fruit Design
   * + A square shaped object i.e. fruit should appear on the screen.
     + A fruit is a square shaped object having a dimension of width 24px and height 24px.
     + The fruit should be positioned at a random position within the bounds specified by the game-board.
     + Fruit should be of different color than the snake.
2. Fruit regeneration

* Each time the snake collides with the fruit, the fruit should disappear from its current position and should be generated at a random position within the bounds of the game-border.
* While regenerating the food at a random position, it should not appear in the positions which is already occupied by the snake.

1. Fruit Type

The game has three different types of food viz. apple, carrot and beer. 5, 10 and 20 points will be awarded whenever the snake gathers an apple, a carrot or beer respectively. The speed with which the snake is moving is also increased whenever the snake drinks beer.

1. Game Functionalities
   1. Collision Detection
2. Game board wall collision detection.

* The game should be over when the head of the snake collides with the boarder of the game-board.
* Should call the gameOver function.

1. Snake collision with fruit

* Should update score
* Increase the size of the snake by adding a tail.
* Generate a fruit at the random position
* Algorithm to check for the collision between snake and the fruit:

Let the top left coordinates for snake and fruit be (Sx, Sy) and (Fx, Fy) respectively. A collision is detected if the following condition is satisfied:

(Sx < Fx + fruitWidth AND Sx + snakeWidth > Fx AND

Sy < Fy + fruitHeight AND Sy + snakeHeight)

1. Snake collision with itself

* Whenever the snake collides with itself the game should be over and the gameOver() function should be called.
* Use the collision detection algorithm described in snake collision with fruit section to check for collision of snake head’s with all of its snake-body sections.
  1. Update Score
* Should update the score based on the type of the fruit captured by the snake.
* This function gets called every time the snake collides with the fruit.
* Should update the score based on the type of the fruit and display it in the score display section.

3.3 Game Over

* + The game is over when the snake collides with itself or collides with the boundary of the game-board.
  + Should cancel all running intervals, cancel event listeners, hide all display objects snake and fruit from the screen.
  + Should display the game over prompt.
  + Should display the current score and high score
  + Should display button to allow user to play again.