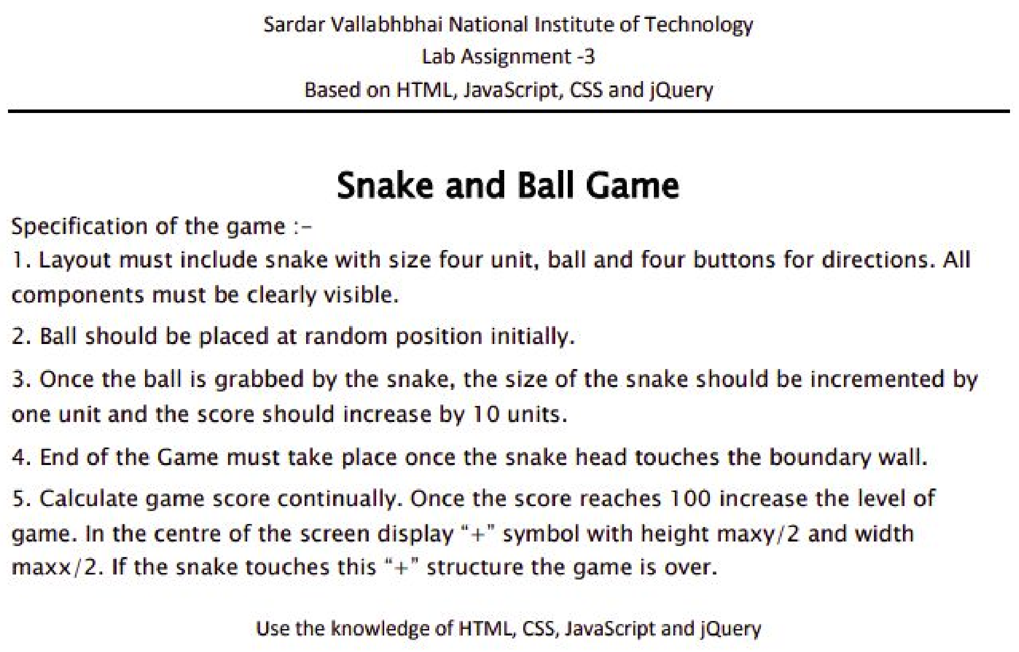
ITA LAB 3

U18CO021: SAHIL BONDRE



Sardar Vallabhbhai National Institute of Technology
Lab Assignment -3
Based on HTML, JavaScript, CSS and jQuery

Snake and Ball Game

Specification of the game :-

1. Layout must include snake with size four unit, ball and four buttons for directions. All components must be clearly visible.
2. Ball should be placed at random position initially.
3. Once the ball is grabbed by the snake, the size of the snake should be incremented by one unit and the score should increase by 10 units.
4. End of the Game must take place once the snake head touches the boundary wall.
5. Calculate game score continually. Once the score reaches 100 increase the level of game. In the centre of the screen display "+" symbol with height maxy/2 and width maxx/2. If the snake touches this "+" structure the game is over.

Use the knowledge of HTML, CSS, JavaScript and jQuery

index.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Snake Game</title>
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/water.css@2/out/dark.css"
    />
    <link rel="stylesheet" href="style.css" />
  </head>

  <body>
    <div id="score">0</div>
    <div id="end"></div>
    <canvas id="snakeboard" width="400" height="400"></canvas>
    <div id="gamepad">
      <button id="left">Left</button>
      <button id="right">Right</button>
      <button id="up">Up</button>
      <button id="down">Down</button>
    </div>
  </body>
  <script src="lib/jquery-min.js"></script>
  <script src="util.js"></script>
  <script src="script.js"></script>
</html>
```

style.css

```
#snakeboard {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
#score {
  text-align: center;
  font-size: 140px;
}
#end {
  text-align: center;
  font-size: 20px;
  font-weight: bolder;
  color: red;
}
#gamepad {
  position: absolute;
  top: 80%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

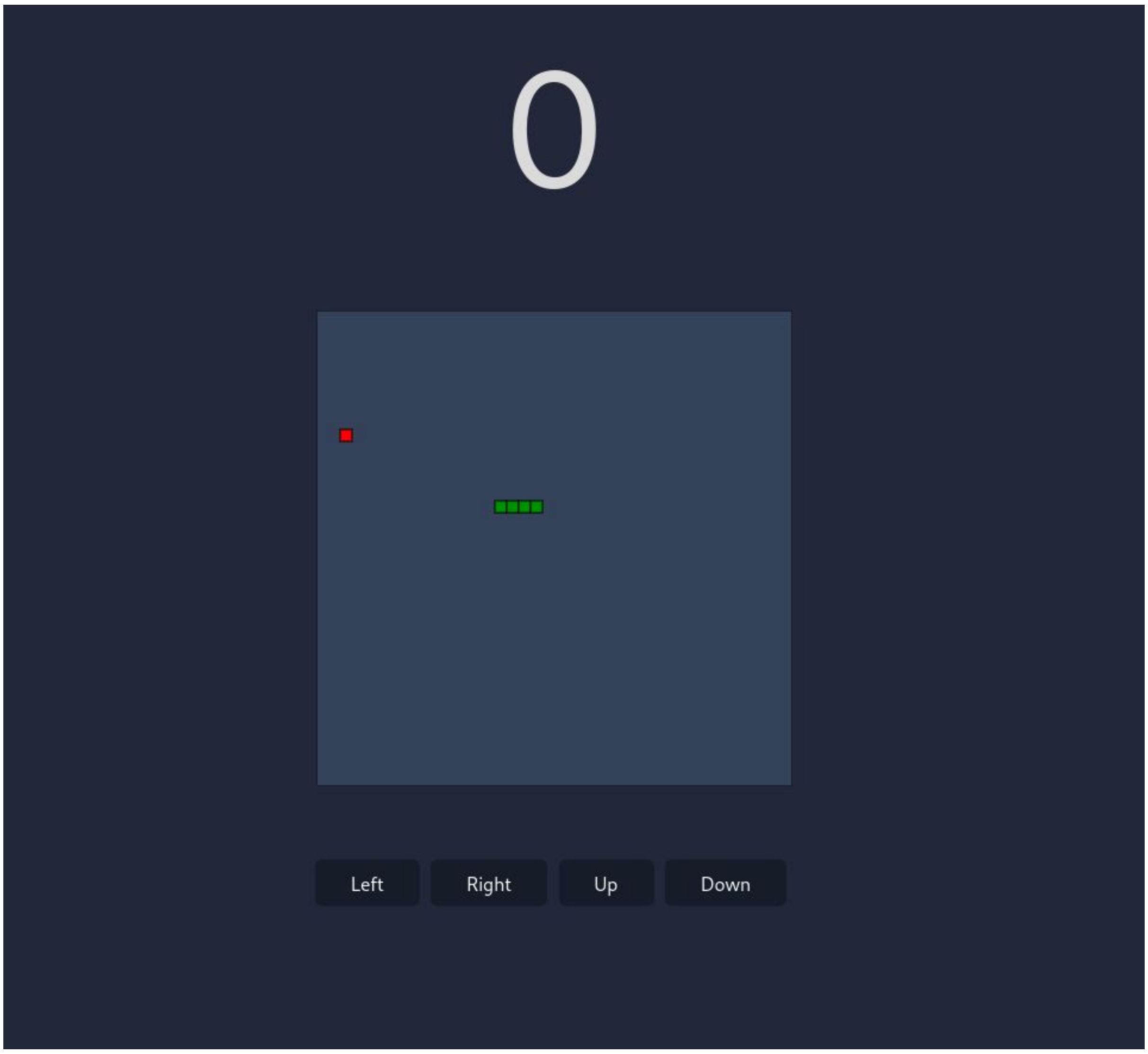
util.js

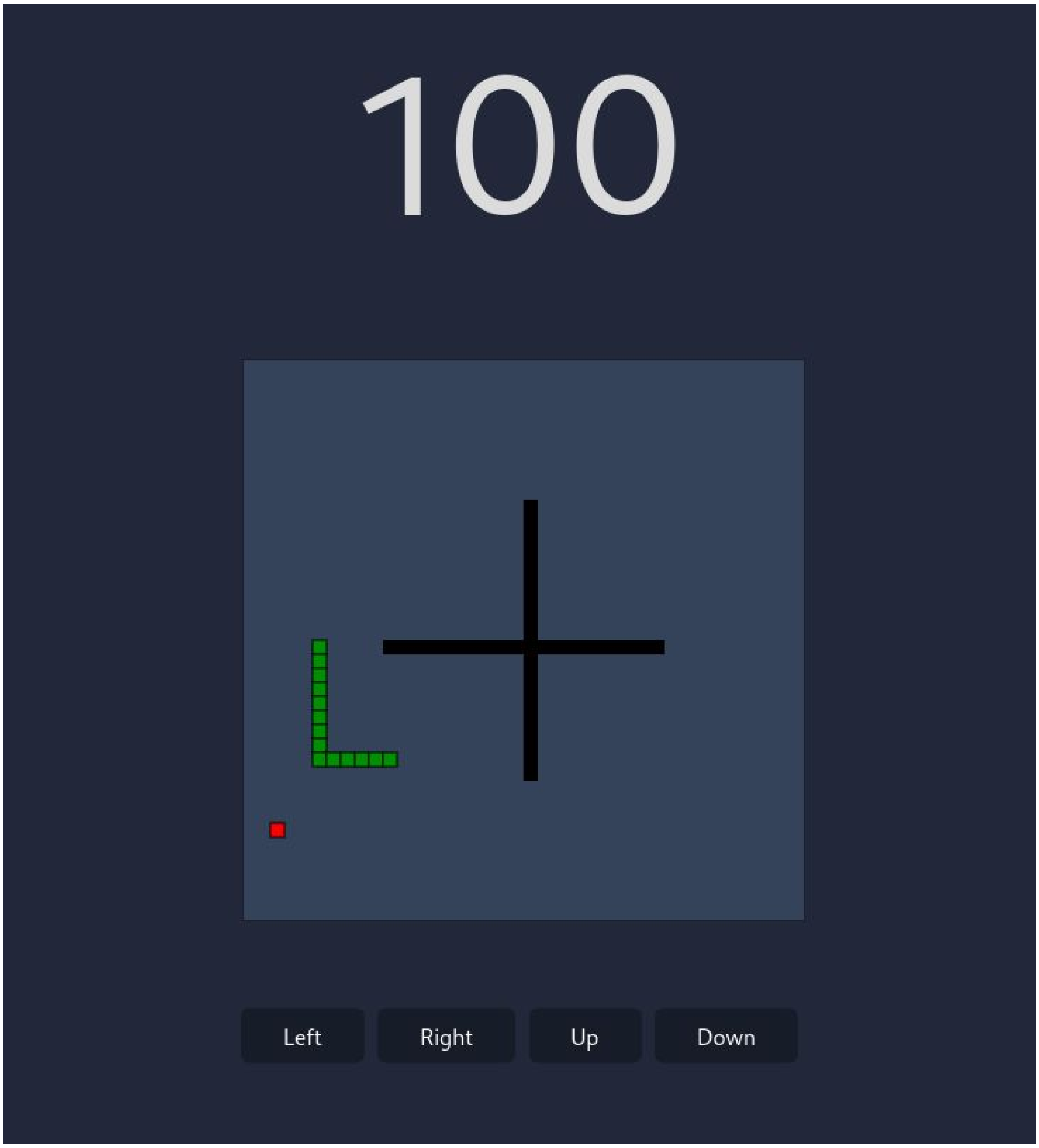
```
// main function called repeatedly to keep the game running
const main = () => {
  if (hasGameEnded()) {
    $("#end").html("Game Over");
    return;
  }
  changingDirection = false;
  setTimeout(function onTick() {
    clearBoard();
    drawFood();
    moveSnake();
    drawSnake();
    drawCross();
    // Repeat
    main();
  }, 100);
};

// draw a border around the canvas
const clearBoard = () => {
  //  Select the colour to fill the drawing
  snakeboard_ctx.fillStyle = "#324759";
  //  Select the colour for the border of the canvas
  snakeboard_ctx.strokestyle = "transparent";
  // Draw a "filled" rectangle to cover the entire canvas
  snakeboard_ctx.fillRect(0, 0, snakeboard.width, snakeboard.height);
  // Draw a "border" around the entire canvas
  snakeboard_ctx.strokeRect(0, 0, snakeboard.width, snakeboard.height);
};

// Draw the snake on the canvas
const drawSnake = () => {
  // Draw each part
  snake.forEach(drawSnakePart);
};

const drawFood = () => {
  snakeboard_ctx.fillStyle = "red";
  snakeboard_ctx.strokestyle = "white";
  snakeboard_ctx.fillRect(food_x, food_y, 10, 10);
  snakeboard_ctx.strokeRect(food_x, food_y, 10, 10);
};

const drawCross = () => {
  if (isCross) {
    const width = snakeboard.width;
    const height = snakeboard.height;
    snakeboard_ctx.fillStyle = "black";
    snakeboard_ctx.strokestyle = "white";
    snakeboard_ctx.fillRect(width / 4, height / 2, width / 2, 10);
    snakeboard_ctx.fillRect(width / 2, height / 4, 10, height / 2);
  }
};

// Draw one snake part
const drawSnakePart = (snakePart) => {
  // Set the colour of the snake part
  snakeboard_ctx.fillStyle = "green";
  // Set the border colour of the snake part
  snakeboard_ctx.strokestyle = "blue";
  // Draw a "filled" rectangle to represent the snake part at the
coordinates
  // the part is located
  snakeboard_ctx.fillRect(snakePart.x, snakePart.y, 10, 10);
  // Draw a border around the snake part
  snakeboard_ctx.strokeRect(snakePart.x, snakePart.y, 10, 10);
};

const hasGameEnded = () => {
  for (let i = 4; i < snake.length; i++) {
    if (snake[i].x === snake[0].x && snake[i].y === snake[0].y) return
true;
  }
  const hitLeftWall = snake[0].x < 0;
  const hitRightWall = snake[0].x > snakeboard.width - 10;
  const hitTopWall = snake[0].y < 0;
  const hitBottomWall = snake[0].y > snakeboard.height - 10;
  const hitCrossVertical =
    snake[0].x == snakeboard.width / 2 &&
    snake[0].y >= 100 &&
    snake[0].y <= 300;
  const hitCrossHorizontal =
    snake[0].y == snakeboard.width / 2 &&
    snake[0].x >= 100 &&
    snake[0].x <= 300;
  if (isCross && (hitCrossHorizontal || hitCrossVertical)) return true;
  return hitLeftWall || hitRightWall || hitTopWall || hitBottomWall;
};

const randomFood = (min, max) => {
  return Math.round((Math.random() * (max - min) + min) / 10) * 10;
};

const genFood = () => {
  // Generate a random number the food x-coordinate
  food_x = randomFood(0, snakeboard.width - 10);
  // Generate a random number for the food y-coordinate
```

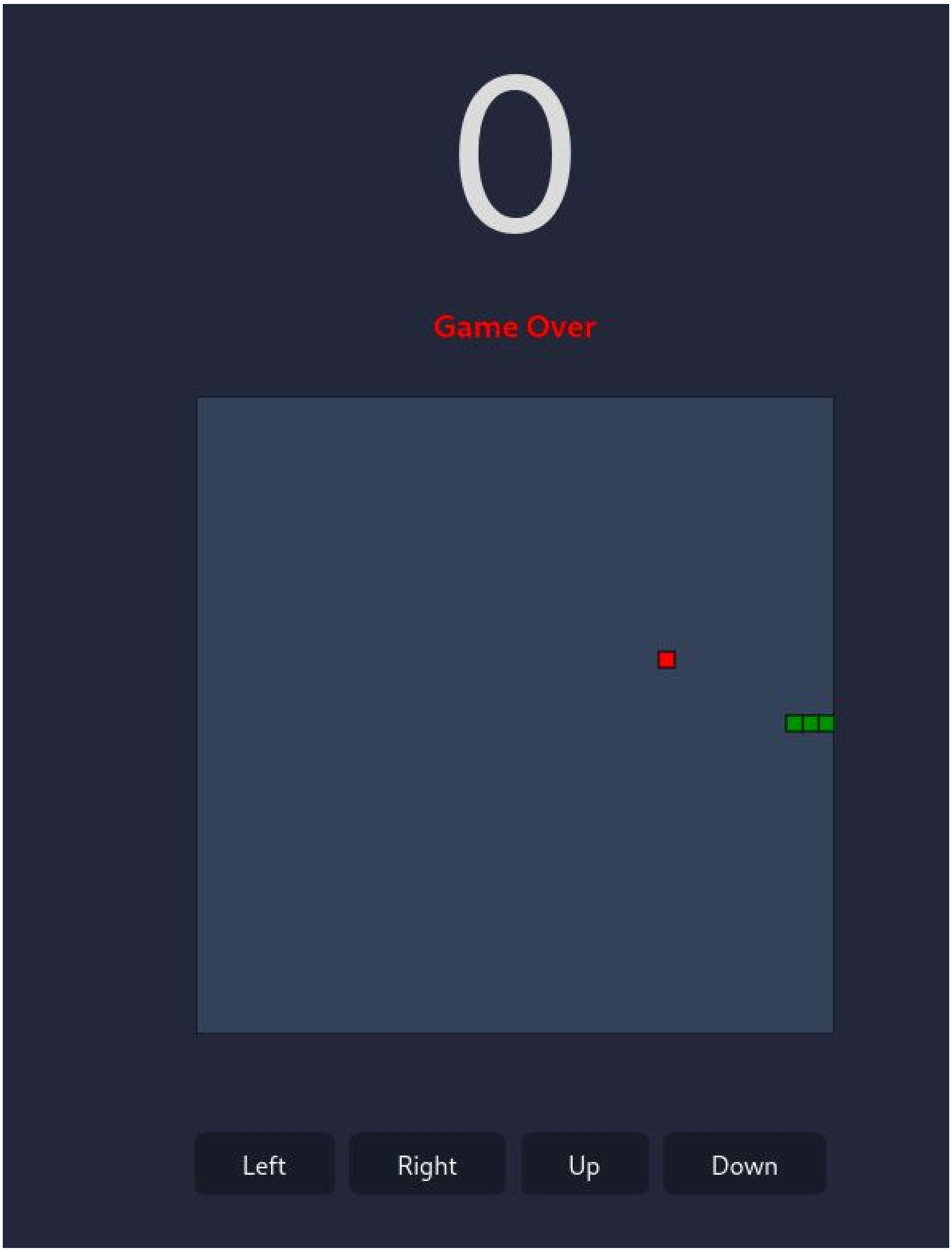
```javascript
  food_y = randomFood(0, snakeboard.height - 10);
  // if the new food location is where the snake currently is, generate
a new food location
  snake.forEach(function has_snake_eaten_food(part) {
    const has_eaten = part.x == food_x && part.y == food_y;
    if (has_eaten) genFood();
  });
};

const changeDirection = (event) => {
  const LEFT_KEY = 37;
  const RIGHT_KEY = 39;
  const UP_KEY = 38;
  const DOWN_KEY = 40;

  // Prevent the snake from reversing

  if (changingDirection) return;
  changingDirection = true;
  const keyPressed = event.keyCode;
  const goingUp = dy === -10;
  const goingDown = dy === 10;
  const goingRight = dx === 10;
  const goingLeft = dx === -10;
  if (keyPressed === LEFT_KEY && !goingRight) {
    dx = -10;
    dy = 0;
  }
  if (keyPressed === UP_KEY && !goingDown) {
    dx = 0;
    dy = -10;
  }
  if (keyPressed === RIGHT_KEY && !goingLeft) {
    dx = 10;
    dy = 0;
  }
  if (keyPressed === DOWN_KEY && !goingUp) {
    dx = 0;
    dy = 10;
  }
};

const changeDirectionButton = (direction) => {
  if (changingDirection) return;
  changingDirection = true;
  const goingUp = dy === -10;
  const goingDown = dy === 10;
  const goingRight = dx === 10;
  const goingLeft = dx === -10;
  if (direction === "LEFT" && !goingRight) {
    dx = -10;
    dy = 0;
  }
  if (direction === "UP" && !goingDown) {
    dx = 0;
    dy = -10;
  }
  if (direction === "RIGHT" && !goingLeft) {
    dx = 10;
    dy = 0;
  }
  if (direction === "DOWN" && !goingUp) {
    dx = 0;
    dy = 10;
  }
};

const moveSnake = () => {
  // Create the new Snake's head
  const head = { x: snake[0].x + dx, y: snake[0].y + dy };
  // Add the new head to the beginning of snake body
  snake.unshift(head);
  const has_eaten_food = snake[0].x === food_x && snake[0].y === food_y;
  if (has_eaten_food) {
    // Increase score
    score += 10;
    if (score === 100) {
      isCross = true;
    }
    // Display score on screen
    $("#score").html(score);
    // Generate new food location
    genFood();
  } else {
    // Remove the last part of snake body
    snake.pop();
  }
};
```

script.js

```
let snake = [
  { x: 200, y: 200 },
  { x: 190, y: 200 },
  { x: 180, y: 200 },
  { x: 170, y: 200 },
];

let score = 0;
// True if changing direction
let changingDirection = false;
// food coords
let food_x;
let food_y;
// Horizontal velocity
let dx = 10;
// Vertical velocity
let dy = 0;

let isCross = false;

// Get the canvas element
const snakeboard = $("#snakeboard")[0];
// Return a two dimensional drawing context
const snakeboard_ctx = snakeboard.getContext("2d");

$("#left").click(() => changeDirectionButton("LEFT"));
$("#right").click(() => changeDirectionButton("RIGHT"));
$("#up").click(() => changeDirectionButton("UP"));
$("#down").click(() => changeDirectionButton("DOWN"));
$(document).keydown(changeDirection);
// Start game
main();

genFood();
```


0
Left
Right
Up
Down


100
Left
Right
Up
Down


0
Game Over
Left
Right
Up
Down