Internet Technology and Applications Practicals

Assignment 3

Krunal Rank
U18CO081

Snake and Ball Game

Specification of the game :-

1. Layout must include a snake with size four unit, ball and four buttons for directions. All components must be clearly visible.
2. Ball should be placed at a 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.

index.html:

```
<html>

<head>
   <title>The Snake Game</title>
   <link rel="icon" href="./assets/images/banner.png">
</head>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
   integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
<style>
   body {
       background-image: url("./assets/images/background.svg");
       background-size: cover;
   }

   .container {
       text-align: center;
   }

   .gameBoard {
       margin-top: 10;
       margin-bottom: 10;
       width: 90%;
       border: 1px solid black;
       background-image: url("./assets/images/canvas-background.svg");
   }
   .row .col-12{
```

```
    margin-top: 10;
    margin-bottom: 10;
}


.startButton h3 {
    background-color: #04bbfb;
    text-align: center;
    margin: auto;
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.16), 0 3px 6px rgba(0, 0, 0, 0.23);
    font-size: 24px;
    padding: 5;
    transition: all 0.3s ease-in-out;
}

.startButton h3:hover {
    background-color: #0489e7;
    cursor: hand;
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.25), 0 6px 12px rgba(0, 0, 0, 0.22);
}

.stopButton h3 {
    background-color: #e25ca3;
    text-align: center;
    margin: auto;
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.16), 0 3px 6px rgba(0, 0, 0, 0.23);
    color: white;
    font-size: 20px;
    padding: 5;
    transition: all 0.3s ease-in-out;
}

.stopButton h3:hover {
    background-color: #8e0441;
    cursor: hand;
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.25), 0 6px 12px rgba(0, 0, 0, 0.22);
}

.banner {
    background-image: url("./assets/images/banner.png");
    background-size: contain;
    background-position: center;
    background-repeat: no-repeat;
}

.scoreboard h3 {
```

```
        text-align: center;
        margin: auto;
        border: 1px solid black;
    }
</style>

<body>
    <div class="container">
        <div class="container dashboard" id="gameCanvasContainer">
            <div class="row">
                <div class="col-12 col-md-4 banner"></div>
                <div class="col-12 col-md-4 scoreboard">
                    <h3 id="score">Score : 0</h3>
                </div>
                <div class="col-12 col-md-4 startButton" id="startButtonDiv">
                    <h3 id="startButton" style="font-size: 20px;letter-spacing:
5;">START</h3>
                </div>
            </div>
        </div>
        <canvas id="gameBoard" class="gameBoard"></canvas>
    </div>
</body>

<script src="https://code.jquery.com/jquery-3.5.1.min.js"
    integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"

integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous">
    </script>
<script src="./script.js"></script>

</html>
```

script.js:

```
const rows = 50
const cols = 50

var tileWidth, tileHeight;
//Create canvas with the device resolution.

var isGameRunning = false;
var isGameOver = false;

var defaultSnakePos = [
   { x: 26, y: 24 },
   { x: 25, y: 24 },
   { x: 24, y: 24 },
   { x: 23, y: 24 },
   { x: 22, y: 24 }
];
var snake = [...defaultSnakePos];
var cross = [
   { x: 24, y: 24 },
   { x: 24, y: 25 },
   { x: 24, y: 23 },
   { x: 25, y: 24 },
   { x: 23, y: 24 },
];


var crossThreshold = 100;
var snakeColor = '#e25ca3';
var strokeColor = '#8e0441';
var foodColor = '#04bbfb';
var foodStroke = '#0489e7';
var crossColor = 'red';
var crossStroke = 'black';
var score = 0;
var food;

var direction = 1; // 0 - left, 1 - right, 2 - top, 3 - bottom

var audio = new Audio()

var PIXEL_RATIO = (function () {
   var ctx = document.createElement("canvas").getContext("2d"),
       dpr = window.devicePixelRatio || 1,
       bsr = ctx.webkitBackingStorePixelRatio ||
```

```
            ctx.mozBackingStorePixelRatio ||
            ctx.msBackingStorePixelRatio ||
            ctx.oBackingStorePixelRatio ||
            ctx.backingStorePixelRatio || 1;

    return dpr / bsr;
})();
createHiDPICanvas = function (ratio) {
    if (!ratio) { ratio = PIXEL_RATIO; }
    var can = document.getElementById("gameBoard");
    var w = Math.round($("#gameCanvasContainer").width() * 0.70);
    var h = Math.round((3 * w) / 4);
    can.width = w * ratio;
    can.height = h * ratio;
    can.style.width = w + "px";
    can.style.height = h + "px";
    tileWidth = Math.round(can.width / cols);
    tileHeight = Math.round(can.height / rows);
    return can;
}

var gameBoard = createHiDPICanvas();
const startButton = document.getElementById("startButton");

const gameBoardContext = gameBoard.getContext("2d");

const changeButtonAppearence = function () {
    if (isGameRunning) {
        $("#startButtonDiv").removeClass("startButton");
        $("#startButtonDiv").addClass("stopButton");
        $("#startButton").html("STOP");
    } else {
        $("#startButtonDiv").removeClass("stopButton");
        $("#startButtonDiv").addClass("startButton");
        $("#startButton").html("START");
    }
}

const checkSnakeCollision = function (newHead) {
    for (var i = 0; i < snake.length; i++) {
        if (newHead.x == snake[i].x && newHead.y == snake[i].y)
            return true;
    }
    return false;
}
const checkCrossCollision = function (newHead) {
```

```
    if (score < crossThreshold) return false;
    for (var i = 0; i < cross.length; i++) {
        if (newHead.x == cross[i].x && newHead.y == cross[i].y)
            return true;
    }
    return false;
}

const moveSnake = function () {
    var newHead;
    if (direction === 0) {
        newHead = { x: (snake[0].x - 1 + cols) % cols, y: snake[0].y };
    } else if (direction === 1) {
        newHead = { x: (snake[0].x + 1) % cols, y: snake[0].y };
    } else if (direction === 2) {
        newHead = { x: snake[0].x, y: (snake[0].y - 1 + rows) % rows };
    } else if (direction === 3) {
        newHead = { x: snake[0].x, y: (snake[0].y + 1 + rows) % rows };
    }

    if (newHead.x == food.x && newHead.y == food.y) {
        snake.unshift(newHead);
        generateFood();
        score += 10;
        renderScore();

    } else {
        if (checkSnakeCollision(newHead) || checkCrossCollision(newHead)) {
            isGameOver = true;
            isGameRunning = false;
            changeButtonAppearence();
        }
        snake.unshift(newHead);
        snake.pop();
    }
}

var toggler = 0;
const gameRunner = async function () {
    if (isGameRunning) {
        setTimeout(() => {
            renderRunner();
            gameRunner();
        }, 60)
    }
    else if (isGameOver) {
```

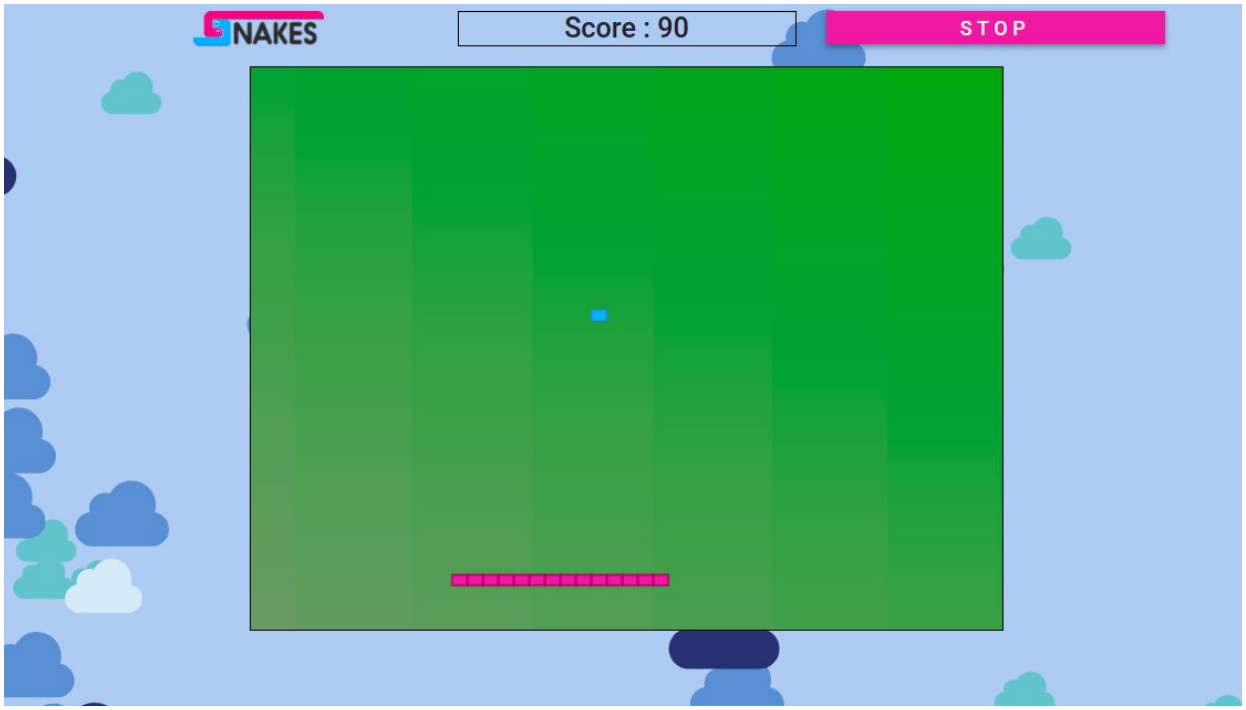
```
        setTimeout(() => {
            renderGameOver();
            gameRunner();
        }, 60);
    }

}

const renderGameOver = function () {
    clearScreen();
    if (toggler <= 4) {
        renderSnake();
        renderGameOverText();
    }
    if (score >= crossThreshold) renderCross();
    toggler = (toggler + 1) % 10;
}

const renderRunner = function () {
    moveSnake();
    clearScreen();
    renderSnake();
    renderFood();
    if (score >= crossThreshold) renderCross();

}

const generateFood = function () {
    food = { x: Math.round(Math.random() * 49), y: Math.round(Math.random() * 49) }
}

const renderFood = function () {
    gameBoardContext.fillStyle = foodColor;
    gameBoardContext.strokeStyle = foodStroke;
    gameBoardContext.fillRect(food.x * tileWidth, food.y * tileHeight, tileWidth,
tileHeight);
    gameBoardContext.strokeRect(food.x * tileWidth, food.y * tileHeight, tileWidth,
tileHeight);

}

const renderCross = function () {
    gameBoardContext.fillStyle = crossColor;
    gameBoardContext.strokeStyle = crossStroke;
    cross.forEach(crossElem => {
```

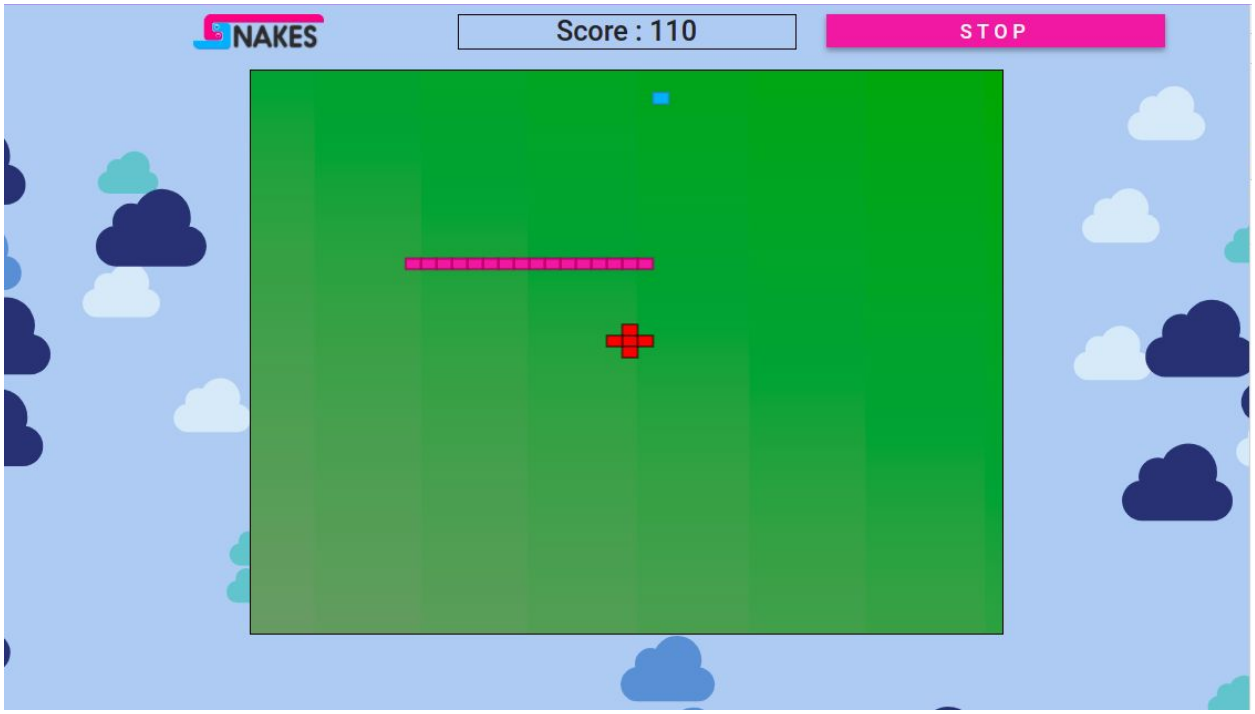
```
        gameBoardContext.fillRect(crossElem.x * tileWidth, crossElem.y * tileHeight,
tileWidth, tileHeight);
        gameBoardContext.strokeRect(crossElem.x * tileWidth, crossElem.y * tileHeight,
tileWidth, tileHeight);

    })

}

const renderGameOverText = function () {
    gameBoardContext.fillStyle = crossColor;
    gameBoardContext.strokeStyle = crossStroke;
    gameBoardContext.font = "30px Arial";
    gameBoardContext.fillText("Game Over", 3 * tileWidth, 3 * tileHeight);


}
const renderSnakeUnit = function (snakeUnit, idx) {
    gameBoardContext.fillStyle = snakeColor;
    gameBoardContext.strokeStyle = strokeColor;
    gameBoardContext.fillRect(snakeUnit.x * tileWidth, snakeUnit.y * tileHeight,
tileWidth, tileHeight);
    gameBoardContext.strokeRect(snakeUnit.x * tileWidth, snakeUnit.y * tileHeight,
tileWidth, tileHeight);
}

const clearScreen = function () {
    gameBoardContext.clearRect(0, 0, gameBoard.width, gameBoard.height);
}

const renderSnake = function () {
    snake.forEach(renderSnakeUnit);
}

const renderScore = function () {
    $("#score").html("Score : " + score);
}

const renderScreen = function () {
    gameBoard = createHiDPICanvas();
    renderSnake();
    generateFood();
    renderFood();
}

const createGame = function () {
```

```
    renderScore();
    renderScreen();
}

const keyDownEvents = function (event) {
    const LEFT_KEY = [37, 65];
    const RIGHT_KEY = [39, 68];
    const UP_KEY = [38, 87];
    const DOWN_KEY = [40, 83];

    const SPACE = [32];


    const keyPressed = event.keyCode;
    if (LEFT_KEY.includes(keyPressed) && direction != 1) {
        direction = 0;
    }

    if (UP_KEY.includes(keyPressed) && direction != 3) {
        direction = 2
    }

    if (RIGHT_KEY.includes(keyPressed) && direction != 0) {
        direction = 1;
    }

    if (DOWN_KEY.includes(keyPressed) && direction != 2) {
        direction = 3;
    }

    if (SPACE.includes(keyPressed)) {
        $('#startButton').click();
    }
}

$(document).on('keydown', keyDownEvents)

$(document).ready(() => {
    createGame();
})
$(window).resize(() => {
    createGame();
})

const resetGame = function () {
    snake = [...defaultSnakePos];
```

```
    direction = 1;
    isGameOver = false;
    score = 0;
    renderScore();
    renderScreen();
}

$("#startButton").on('click', () => {
    isGameRunning = !isGameRunning;
    if (isGameRunning) {
        changeButtonAppearence();
        resetGame();
        gameRunner();
    } else {
        changeButtonAppearence();
    }

})
```

Screenshots:




NAKES
Score : 110
STOP


NAKES
Score : 130
START
Game Over