UNIWERSYTET RZESZOWSKI
WYDZIAŁ NAUK
ŚCISŁYCH I TECHNICZNYCH
INSTYTUT INFORMATYKI

Kacper Zoła
134993

Informatyka

Snake - JavaScript

Praca projektowa

Praca wykonana pod kierunkiem
dr Katarzyna Garwol

Rzeszów 2025

Spis treści

Rozdział 1

Opis założeń projektu

1.1 Cel projektu

Celem projektu jest stworzenie klasycznej gry o nazwie "Snake"z użyciem technologi JavaScript.

1.2 Wymagania funkcjonalne

1. **JavaScript** - Cała mechanika strony musi zostać napisana w JavaScript.
2. **HTML** - HTML pozwala na użycie JavaScript w przeglądarce.
3. **CSS** - Wygląd strony został napisany przy użyciu CSS.

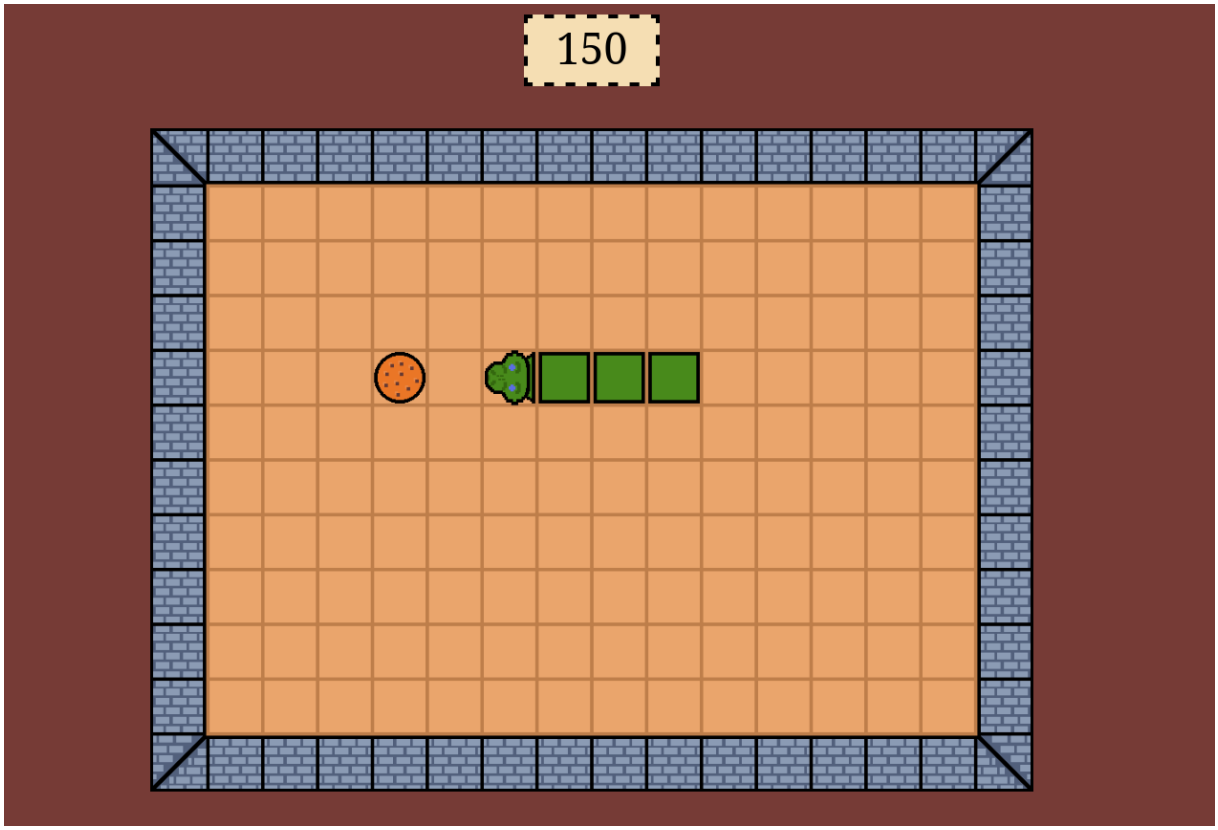
1.3 Wymagania sprzętowe

1. **Przeglądarka internetowa** - dowolna przeglądarka będąca wstanie obsługiwać nowoczense wersje JavaScript, HTML i CSS.

Rozdział 2

Struktura strony

2.1 Wygląd strony



2.2 HTML

2.2.1 head

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SNAKE</title>
</head>
<style>
    kod CSS...
</style>

<body>
    <div class="game-container">
        <p id="points">0</p>
        <canvas id="mainCanvas" width="1024" height="768"></canvas>
    <script>
       kod javascript...
    </script>
</body>

</html>
```

W części HTML są zawarte podstawowe rzeczy do poprawnego wyświetlania strony. Najważniejszymi elementami tutaj są <p> o id "points"który jest używany do wyświetlania punktów, oraz <canvas> o id "mainCanvas"który służy do renderowania grafiki gry.

2.3 CSS

2.3.1 Metody domyślne

```css
body {
    margin: 0;
    height: 100vh;
    background-color: #763b36;
    display: flex;
    justify-content: center;
    align-items: center;
}

.game-container {
    display: flex;
    flex-direction: column;
    align-items: center;
}

#mainCanvas {
    border: 4px solid black;
}

#points {
    width: 150px;
    height: 75px;
    text-align: center;
    font-size: 50px;
    color: black;
    background-color: wheat;
    margin-bottom: 50px;
    border: 5px dashed black;
}
```

W sekcji CSS są minimalne zmiany kolorystyczne strony, oraz ustawianie elementów na środku strony.

2.4 Java Script

2.4.1 Consts

```
const scoreboard = document.getElementById("points");
const canvas = document.getElementById("mainCanvas");
const ctx = canvas.getContext("2d");

const WIDTH = canvas.width;
const HEIGHT = canvas.height;
const TILE_SIZE = 64;

let running = true;
let points = 0;
let timeDifficulty = 250;

const cols = WIDTH / TILE_SIZE;
const rows = HEIGHT / TILE_SIZE;

const cookieSprite = new Sprite("assets/cookie.png");

const ground = new Sprite("assets/ground.png");
const wallTop = new Sprite("assets/walls/top.png");
const wallBottom = new Sprite("assets/walls/bottom.png");
const wallRight = new Sprite("assets/walls/right.png");
const wallLeft = new Sprite("assets/walls/left.png");
const wallBottomLeftCorner = new Sprite("assets/walls/bottom-left.png");
const wallBottomRightCorner = new Sprite("assets/walls/bottom-right.png");
const wallTopLeftCorner = new Sprite("assets/walls/top-left.png");
const wallTopRightCorner = new Sprite("assets/walls/top-right.png");

 const snakeBody = new Sprite("assets/snake/snake_body.png");
 const snakeFront = new Sprite("assets/snake/snake_front.png");
 const snakeBack = new Sprite("assets/snake/snake_back.png");
 const snakeLeft = new Sprite("assets/snake/snake_left.png");
 const snakeRight = new Sprite("assets/snake/snake_right.png");

let snakePosition = new Vector2(2, 3);
let snakeLastPosition = new Vector2(0, 0);
let direction = new Vector2(1, 0);
```

Na początku kodu JavaScript zawarte są inicjalizowane wszelke stałe oraz zmienne globalne. W stałych są to w większości textury do rysowania elementów gry ale także szerokość i wysokość canvasa, rozmiar renderowanych textur oraz elementy w HTML. Do zmiennych należą takie parametry jak pozycja węża, jego ostania pozycja, czy gra powinna operować, liczba punktów oraz prędkość działania gry.

2.4.2 Klasa LinkedList, Node, Vector2, Cookie, Sprite

```
class Node {
    constructor(x, y) {
        this.x = x;
        this.y = y;
        this.lastPositionX = x;
        this.lastPositionY = y;
        this.next = null;
    }
}
```

```
class LinkedList {
    constructor() {
        this.head = null;
    }
    insert(x, y) {
        let newnode = new Node(x, y);
        if (!this.head) {
            this.head = newnode;
            return;
        }
        let current = this.head;
        while (current.next) {
            current = current.next;
        }
        current.next = newnode;

    }
    updateList() {
        let current = this.head;
        let previous = null;

        while (current) {
            current.lastPositionX = current.x;
            current.lastPositionY = current.y;

            if (previous !== null) {
                current.x = previous.lastPositionX;
                current.y = previous.lastPositionY;
            }
            else {
                current.x = snakeLastPosition.x;
                current.y = snakeLastPosition.y;
            }
            previous = current;
            current = current.next;
        }
    }
    drawBody() {
        let current = this.head;
```

```
40          while (current) {
41              ctx.drawImage(snakeBody.sprite, current.x * TILE_SIZE, current.y *
                ↪  TILE_SIZE);
42              current = current.next;
43          }
44      }
45      checkCollision() {
46          let current = this.head;
47          while (current) {
48              if (snakePosition.x === current.x && snakePosition.y === current.y)
                ↪  return true;
49              current = current.next;
50          }
51          return false;
52      }
53  }
```

Klasa Node i LinkedList oraz jej funkcje została zaimplementowana do przechowywania informacja o ciele węża, zawiera funkcje pozwalające na dodanie elementu ciała na końcu, zmiane danych w elementach ciała, renderowanie ciała oraz sprawdzenie pozycji.

```
1   class Vector2 {
2       constructor(x, y) {
3           this.x = x;
4           this.y = y;
5       }
6   }
```

```
1   class Cookie {
2       constructor() {
3           this.position = new Vector2(0, 0);
4           this.position.x = Math.floor((Math.random() * (WIDTH / TILE_SIZE - 2))
            ↪  + 1);
5           this.position.y = Math.floor((Math.random() * (HEIGHT / TILE_SIZE - 2))
            ↪  + 1);
6           this.points = 50;
7       }
8       eat() {
9           bodyList.insert(snakeLastPosition.x, snakeLastPosition.y);
10          points += this.points;
11      }
12      draw() {
13          ctx.drawImage(cookieSprite.sprite, this.position.x * TILE_SIZE,
            ↪  this.position.y * TILE_SIZE);
14      }
15  }
```

Klasa Cookie pozwala na losowe stworzenie ciastka na mapie które posiada funkcje zwiększająca liczbę punktów i ciała węża, oraz renderowanie.

```
class Sprite {
    constructor(path) {
        this.sprite = new Image(TILE_SIZE, TILE_SIZE);
        this.sprite.src = path;
    }
}
```

Klasa Sprite pozwala na wygodniejsze przechowywanie stałych textur.

2.4.3 Game loop

```
function Main() {
    if (!running) {
        if (confirm("GAME OVER\nTry again?")) window.location.reload();
        return;
    }

    Player();
    Draw();
    Collision();
    setTimeout(Main, timeDifficulty);
}
```

Funkcja Main zawiera w sobie wszystkie pozostałe funkcje definiujące funkcjonalność gry i uruchamia się sama po określonym czasie, lub przerywa grę.

2.4.4 Renderowanie

```
function Draw() {
    ctx.clearRect(0, 0, WIDTH, HEIGHT);

    // ground and obstacles
    for (let x = 0; x < WIDTH; x += TILE_SIZE) {
        for (let y = 0; y < HEIGHT; y += TILE_SIZE) {
            ctx.drawImage(ground.sprite, x, y);
        }
    }

    // Walls
    for (let y = 0; y < HEIGHT; y += TILE_SIZE) {
        ctx.drawImage(wallLeft.sprite, 0, y);
        ctx.drawImage(wallRight.sprite, WIDTH - TILE_SIZE, y);
    }
    for (let x = 0; x < WIDTH; x += TILE_SIZE) {
        ctx.drawImage(wallTop.sprite, x, 0);
        ctx.drawImage(wallBottom.sprite, x, HEIGHT - TILE_SIZE);
    }
    // Corners
    ctx.drawImage(wallBottomLeftCorner.sprite, 0, HEIGHT - TILE_SIZE);
    ctx.drawImage(wallBottomRightCorner.sprite, WIDTH - TILE_SIZE, HEIGHT -
    ↪  TILE_SIZE);

    ctx.drawImage(wallTopRightCorner.sprite, WIDTH - TILE_SIZE, 0);
    ctx.drawImage(wallTopLeftCorner.sprite, 0, 0);

    if (cookie !== null) cookie.draw();
    bodyList.drawBody(); // DRAW BODY

    // FRONT
    if (direction.y === -1 && direction.x === 0)
        ctx.drawImage(snakeFront.sprite, snakePosition.x * TILE_SIZE,
        ↪  snakePosition.y * TILE_SIZE);

    // DOWN
    if (direction.y === 1 && direction.x === 0)
        ctx.drawImage(snakeBack.sprite, snakePosition.x * TILE_SIZE,
        ↪  snakePosition.y * TILE_SIZE);

    // LEFT
    if (direction.x === -1 && direction.y === 0)
        ctx.drawImage(snakeLeft.sprite, snakePosition.x * TILE_SIZE,
        ↪  snakePosition.y * TILE_SIZE);

    // RIGHT
    if (direction.x === 1 && direction.y === 0)
        ctx.drawImage(snakeRight.sprite, snakePosition.x * TILE_SIZE,
        ↪  snakePosition.y * TILE_SIZE);
```

```
45 }
```

Funkcja Draw odpowiada za renderowanie textur w canvasie bazując na odpowiednich informacjach, na przykład wktórą strone ma być renderowana głowa węża.

2.4.5 Event Listener

```
1  let direction = new Vector2(1, 0);
2
3  document.addEventListener("keydown", (event) => {
4      if (direction.y !== 1 && (event.key === "ArrowUp" || event.key === "w")) {
5          direction.x = 0;
6          direction.y = -1;
7      }
8      if (direction.y !== -1 && (event.key === "ArrowDown" || event.key === "s"))
       ↪  {
9          direction.x = 0;
10         direction.y = 1;
11     }
12     if (direction.x !== 1 && (event.key === "ArrowLeft" || event.key === "a"))
       ↪  {
13         direction.x = -1;
14         direction.y = 0;
15     }
16     if (direction.x !== -1 && (event.key === "ArrowRight" || event.key ===
       ↪  "d")) {
17         direction.x = 1;
18         direction.y = 0;
19     }
20 });
```

Sekcja ta odpowiada za przechwytywanie kliknięć gracza i zmienianie wartości vectora odpowiadającego za kierunek w którym przemieszcza się wąż. Kierunek można zmieniać po przez przyciski strzałek lub WASD.

2.4.6 Player

```
let bodyList = new LinkedList();
let cookie = new Cookie();

function Player() {
    if (cookie !== null) {
        if (snakePosition.x === cookie.position.x && snakePosition.y ===
        ↪  cookie.position.y) {
            cookie.eat();
            if (timeDifficulty > 150) timeDifficulty -= 10;
            console.log(timeDifficulty);
            cookie = null;
        }
    }
    else {
        cookie = new Cookie();
    }
    scoreboard.textContent = points;

    snakeLastPosition.x = snakePosition.x;
    snakeLastPosition.y = snakePosition.y;
    snakePosition.x += direction.x;
    snakePosition.y += direction.y;
    bodyList.updateList();
}
```

Sekcja ta odpowiada za złączenie funkcjonalności związanych z wężem (graczem). Aktualizuje pozycje węża bazując na kierunku oraz tworzy instancje klasy Cookie która tworzy ciastko na mapie i sprawdza czy z nim nie koliduje, jeżeli pozycja węża będzie taka sama jak pozycja ciastka, to zostanie ono usunięte, liczby punktów się zwiększy wraz z ciałem węża, a czas zostanie zmniejszony co spowoduje przyśpieszenie działa gry.

2.4.7 Collision

```
function Collision() {
    if (snakePosition.x > (WIDTH / TILE_SIZE) - 2 ||
        snakePosition.x < 1 ||
        snakePosition.y > (HEIGHT / TILE_SIZE) - 2 ||
        snakePosition.y < 1) {
        running = false;
    }
    if (bodyList.checkCollision()) {
        running = false;
    }
}
```

Funkcja ta sprawdza czy wąż nie uderzył w ściane co powoduje przerwanie gry.

Rozdział 3

Podsumowanie

3.1 Możliwości rozwojowe

1. - Dodanie większej liczby poziomów z różnymi kształtami mapy,
2. - Dodanie przeszkód bądź przeciwników których trzeba unikać,
3. - Dodanie możliwości zmiany koloru węża

Stworzono grę za pomocą języka JavaScript przy użyciu technologi HTML i CSS, działającą w przeglądarce co pozwala na uruchomienie gry na większości komputerów stacjonarnych. Projekt skupia się głównie na JavaScript zgodnie z założeniem projektu.