General Assembly - Software Engineering Immersive

SNAKE GAME 🐍

Overview

This was my first project in the General Assembly Software Engineering Immersive course. It followed a fortnight of learning JavaScript. The project was completed on my own in one week.

You can play the game via GitHub pages here.

Brief

The project brief stated that we had to:

- Render a game in the browser
- Have logic for playing the game and for winning
- Include separate HTML / CSS / JavaScript files
- Stick with KISS (Keep It Simple Stupid) and DRY (Don't Repeat Yourself) principles
- Use JavaScript for DOM manipulation
- Use semantic markup for HTML and CSS (adhere to best practices)

Technologies Used

- HTML
- CSS
- JavaScript (ES6)
- Git and GitHub

Approach

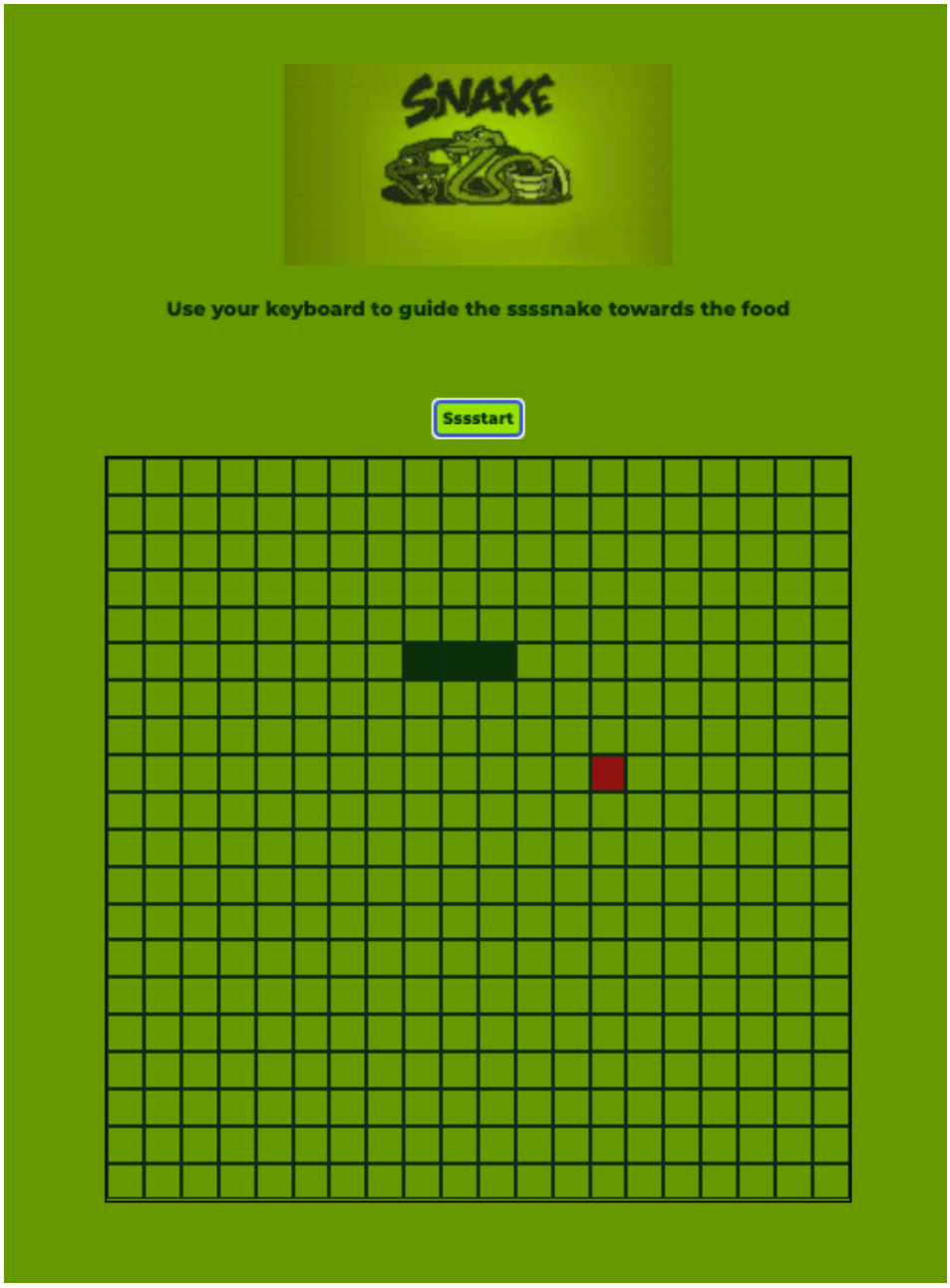
I considered the games that were offered as options and went for Snake as I used to play it when I had a Nokia mobile phone in the early 2000s. It's a fun game and it would allow me to meet the criteria set. It also allowed me to be experimental with the frontend, and I decided to try to replicate the actual Nokia mobile phone screens, with their green tinge.

I set about writing down the rules of the game - and was surprised at how many there actually are, such as the way the snake moves, the way the user controls it, how the food appears and how the snake grows when it 'eats' the food, and how the game can end - such as through crashing into a wall or the snake crashing into itself.

I then looked at how I could code each of these variables, and how to code the overall game grid. It wasn't particularly easy and it took a lot longer than I had thought it would to create a working game. I was glad I spent the time planning the functionality and considering what the code could look like before I started to write the actual code.

Screenshot

This is the game page:



Bugs

The game works well but the snake travels through the walls and so I need to spend some time fixing that. The snake also bounces once to the left when the game begins so I need to understand what is causing that error and fix it.

Lessons Learned

Many! The hardest part of this was engineering the movement of the snake. Making the entire snake move up, down, left or right was easy but making the snake move in the direction of the

snake's head was much more challenging. The head was the element that needed to respond to the user's input, not the entire snake body. I used this code to manage the movement:

```
const snakeMove = setInterval(() => {
    snakeBody.forEach((part) => {
      cells[part].classList.remove('snakeColor')
    })
    const snakeHead = nextSnakeHead()
    snakeBody.unshift(snakeHead)
    if (snakeHead === food) {
      cells[food].classList.remove('food')
      points += 100
      randomFood()
    } else {
      snakeBody.pop()
    }
    snakeBody.forEach((part) => {
      cells[part].classList.add('snakeColor')
    })
  }, 270)
})
```

I also realised that the arrow keys, which control the snake, actually control the viewport as well and so I had to input some code to stop the keys from doing that.

Potential Future Features

It would be good to include different levels and perhaps expand the gameplay options - such as allowing the snake to travel through walls and emerge at the opposite side of the screen, as Nokia did in some of its later versions of Snake.

On the UX front I could move away from snakes to give the game more depth. And a scoreboard would add to the user experience, especially if it remembered the highest score that have been achieved on the game.