Explication du code JavaScript Snake



Lucas Simões Pólvora

MID2 – 2022-2023

Chef de projet : Aurélie Curchod

Table des matières

[1. Introduction 3](#_Toc153281778)

[2. Analyse 3](#_Toc153281779)

[3. Réalisation 3](#_Toc153281780)

[4. Code 4](#_Toc153281781)

[4.1 Move() 4](#_Toc153281782)

[4.2 events() 4](#_Toc153281783)

[4.3 readInput() 4](#_Toc153281784)

[4.4 appleSpawn() 4](#_Toc153281785)

[4.5 drawSnake() 4](#_Toc153281786)

# Introduction

Ce projet a été réalisé dans le cadre de l’ETML, avec comme chef de projet Mme. Aurélie Curchod. Ce projet a été réalisé afin d’approfondir nos connaissances en JavaScript.

Qu’est-ce que le jeu Snake ? Ce jeu est un des années 80 environ qui a fait un succès énorme à l’époque. Son concept est pourtant simple, manger pour grandir en évitant les bordures et son propre corps. Facile à dire mais difficile à terminer.

# Analyse

Avant de commencer à coder, j’ai réfléchi à comment on pourrait faire pour être le plus optimal. J’ai trouvé la solution de retirer la queue du serpent (dernier carré) et le mettre devant comme si c’était la tête. Cela permettrait de déplacer qu’un seul carré et économiser plus de ressources.

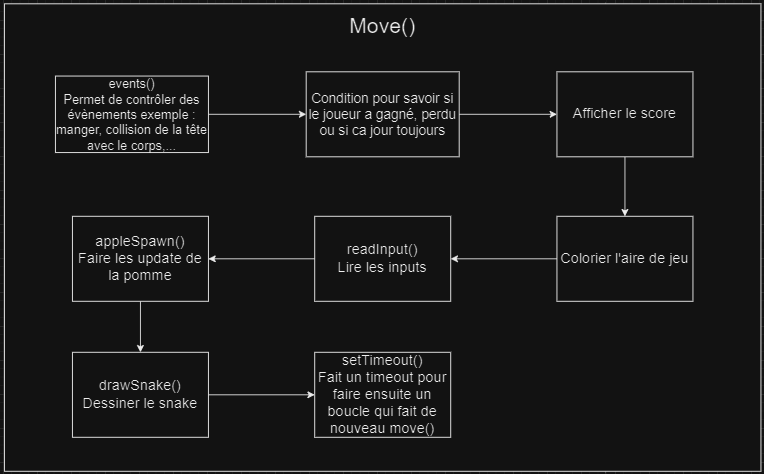
Pour la pomme j’ai réfléchi à une manière afin qu’elle n’apparaisse pas dans le serpent, ce serait de la même manière que pour vérifier que le serpent ne s’est pas collisionné avec son corps. On ferait un array.some qui permettrait de parcourir tous les paramètres du tableau.

J’avais déjà beaucoup joué au Snake donc j’ai instantanément fait en sorte que le serpent ne puisse pas aller en arrière et que le serpent avancerait même si l’on lâchait la touche.

# Réalisation

Pour débuter le programme, j’ai toujours eu l’habitude de faire beaucoup de variables et constantes pour pouvoir modifier les attributs très facilement. Ensuite il y a une fonction move() qui est la base du programme qui va se répéter à chaque frame.

Cette fonction va être composée de conditions et d’autres fonctions. Voici un schéma qui explique l’ordre de réalisation dans cette fonction :



# Code

## Move()

function move() {

  // Makes all the events of the game

  events();

  // If the snake is dead it will show GameOver

  if (!isSnakeAlive) {

    if (scoreValue <= SQUARE\_HEIGHT \* SQUARE\_WIDTH - 3) {

      gameOver.textContent = "GameOver";

    }

    else {

      gameOver.textContent = "You Win !";

    }

    return;

  }

  // Shows the current score

  score.textContent = "Score : " + scoreValue

  // Draws the game board

  ctx.fillStyle = TABLE\_COLOR;

  ctx.fillRect(0, 0, PLAY\_TABLE\_WIDTH, PLAY\_TABLE\_HEIGHT);

  // Reads the input from the user

  readInput();

  // Makes the apple spawn

  appleSpawn();

  // Draws and updates the snake movement

  drawSnake();

  // Refreshes each 100ms

  setTimeout(() => {

    requestAnimationFrame(move);

  }, 100);

}

## events()

// Treats the different events that occur in the game (Snake dies, eats apple)

function events() {

  // Events that allows to know if the snake is dead by the borders

  // Right Border

  if (coordX >= PLAY\_TABLE\_WIDTH) {

    isSnakeAlive = false

  }

  // Left Border

  else if (coordX < 0) {

    isSnakeAlive = false

  }

  // Top Border

  else if (coordY < 0) {

    isSnakeAlive = false

  }

  // Down Border

  else if (coordY >= PLAY\_TABLE\_HEIGHT) {

    isSnakeAlive = false

  }

  // If the snakes touches himself he dies

  if (partOfTheSnake.some((segment, index) => index !== 0 && segment.x === partOfTheSnake[0].x && segment.y === partOfTheSnake[0].y)) {

    compteur++;

    if (compteur !== 1) {

      isSnakeAlive = false;

    }

  }

  // Event that allows the snake to eat the apple

  if (appleX == coordX && appleY == coordY) {

    partOfTheSnake.unshift({

      x: coordX,

      y: coordY,

      w: SNAKE\_WIDTH,

      h: SNAKE\_HEIGHT

    });

    isAppleAlive = false;

    scoreValue++;

  }

}

## readInput()

// Reads the input of the user

function readInput() {

  // Is used when the key is down

  document.addEventListener('keydown', function (event) {

    // For the left

    if (event.keyCode == 37 && !goRight && didAMovement) {

      goLeft = true;

      goDown = false;

      goUp = false;

      goRight = false;

      didAMovement = true;

    }

    // For the top

    else if (event.keyCode == 38 && !goDown && didAMovement) {

      goUp = true;

      goDown = false;

      goLeft = false;

      goRight = false;

      didAMovement = true;

    }

    // For the right

    else if (event.keyCode == 39 && !goLeft) {

      goRight = true;

      goLeft = false;

      goDown = false;

      goUp = false;

      didAMovement = true;

    }

    // For below

    else if (event.keyCode == 40 && !goUp && didAMovement) {

      goDown = true;

      goUp = false;

      goLeft = false;

      goRight = false;

      didAMovement = true;

    }

  }, false)

}

## appleSpawn()

// Makes the apple spawn

function appleSpawn() {

  if (!isAppleAlive) {

    // Makes random numbers for the square that the apple will spawn

    appleX = Math.floor(Math.random() \* SQUARE\_WIDTH);

    appleY = Math.floor(Math.random() \* SQUARE\_HEIGHT);

    // Makes the width and the ehight of the apple

    appleX = appleX \* APPLE\_WIDTH;

    appleY = appleY \* APPLE\_HEIGHT;

    // If the apple spawns in a square that's already occupied by the snake, we spawn it again

    if (partOfTheSnake.some((segment) => appleX == segment.x && appleY == segment.y)) {

      console.log("a");

      appleSpawn();

    }

    // Resets value

    isAppleAlive = true;

  }

  // Shows the apple

  ctx.fillStyle = APPLE\_COLOR

  ctx.fillRect(appleX, appleY, APPLE\_WIDTH, APPLE\_HEIGHT)

}

## drawSnake()

// Draws and moves the snake

function drawSnake() {

  // Draws the snake's head

  ctx.fillStyle = SNAKE\_COLOR;

  partOfTheSnake.some(v => {

    ctx.fillRect(v.x, v.y, v.w, v.h)

  })

  // Makes the movement according to the input

  if (goUp && coordY > - 10) {

    coordY -= SNAKE\_HEIGHT;

  }

  else if (goDown && coordY + SNAKE\_HEIGHT < PLAY\_TABLE\_HEIGHT + 10) {

    coordY += SNAKE\_HEIGHT;

  }

  else if (goRight && coordX + SNAKE\_WIDTH < PLAY\_TABLE\_WIDTH + 10) {

    coordX += SNAKE\_WIDTH

  }

  else if (goLeft && coordX > -10) {

    coordX -= SNAKE\_WIDTH;

  }

  // If the snake makes a movement, it unshows the last square (tail) and puts it in front (Becomes the head)

  if (didAMovement) {

    partOfTheSnake.pop();

    partOfTheSnake.unshift({

      x: coordX,

      y: coordY,

      w: SNAKE\_WIDTH,

      h: SNAKE\_HEIGHT

    });

  }

}