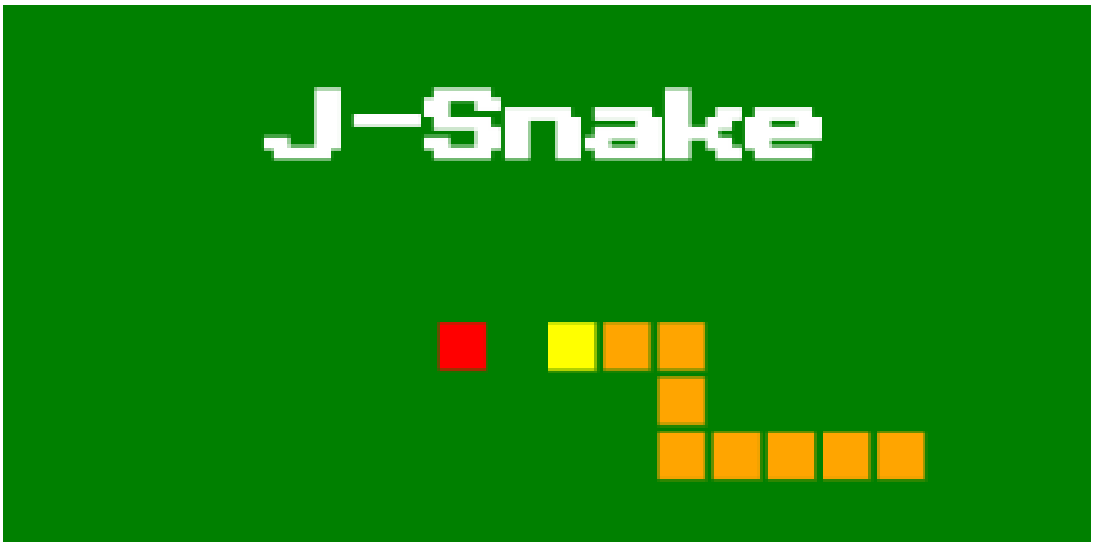
ECOLE TECHNIQUE
ECOLE DES MÉTIERS LAUSANNE

« J-SNAKE »

P_Bulles :

Faire un réplica du célèbre jeu Snake en JavaScript



07 NOVEMBRE 2023
ADRIAN TOLEDO
Enseignant : Aurélie Curchod

Table des matières

1. P_Bulle : Snake

1.1. Introduction

Le célèbre jeu vidéo Snake, vit dans notre société depuis des années comme un jeu vidéo culte pour son impact sur la société et la simplicité de son gameplay. Dans ce qui suit, ce projet proposera une version simple du jeu adaptée à une page web avec JavaScript.

1.2. Objectif

Pendant 8 semaines avec un total de 5 périodes de 45 minutes, un projet JavaScript représentant le célèbre jeu vidéo Snake doit être réalisé. Ce jeu doit être développé avec les éléments de base suivants : mouvement, collisions, score, interaction avec le clavier et une syntaxe moderne et optimale pour le code du jeu.

2. Principes du JavaScript

JavaScript est un langage de programmation interprété qui permet d'interagir et de travailler sur le code HTML des pages web. Il tire son origine du dialecte ECMAScripti, défini comme un code orienté objet, basé sur des prototypes, faiblement typé et dynamique.

Voici une explication de la syntaxe et de la nomenclature pour coder avec JavaScript

2.1. Variables

Variable	**Description**	**Exemple**
Let	Enregistrez des variables telles que du string, un nombre ou tout autre type	let name = 'adri'
Const	Variables dont la valeur ne sera pas modifiée	const COLOR = '#F00'
Var	Déclare variables (il est recommandé d'utiliser Let pour éviter les problèmes)	var name = 'adri'

2.2. Types de données

Type	**Exemple**
Nombres	`let n=123`
Décimales	`let n=12.3`
String	`let name = 'adri'`
Boolean	`let alive = true`
Nul	`let status = null`
Undefined	`let age = undefined`
Object	`let obj = new Object (value)`
Function	`let bucle = (x) => {}`

2.3. Array

Description	Exemple
Création d'un tableau	`let array = [1,2]` `let array = new Array(item1, item2)`
Ajouter un élément au tableau	`array[1] = 3`
Longueur de la chaîne	`array.length`
Connaître la dernière valeur	`array.at(-1)`
Pop – supprime la dernière valeur	`array.pop()`
Push – ajoute une valeur à la fin de la chaîne	`array.push(7)`
Shift – supprime la première valeur de la chaîne	`array.shift()`
Unishift – ajouter une valeur au début de la chaîne	`array.unshift(5)`
Matrice	`matrice = [[1, 2, 3], [4, 5, 6],[7, 8,]]` `alerte(matrice[1][1]) = 5`
ForEach	`arr.forEach(function(item, index, array){}`
Rest : est utilisé pour pouvoir ajouter des informations non définies dans un array	`function sum(...theArgs){}`

2.4. Mathématiques - Opérateurs classiques - Conversions

Type	Exemple
Somme	`let add = 2+1`
Soustraction	`let soustraction = 2-1`
Multiplication	`let multi = 2*3`
Division	`let division = 2/3`
Reste d'une division	`let reste = 2%1`
Exponentielle	`let expo = 2**2`
Random	`Math().random`
Round	`Math().round`
Absolut	`Math().abs`
Cos/Sin/Tan	`Math().cos/sin/tan`
ToString	`let value = true;` `value = String(value)`
ToNumber	`let value = '123';` `value = Number(value)`
ToBoolean	`let value = 1;` `value=Boolean(value)`

2.5. Comparateurs, conditionnels et opérateurs logiques

Une structure peut être créée pour définir une condition-action dans le code avec « if else » ou « Switch».

Type	Exemple
If	`If(condition){}` `else {}`

Switch	`Switch(value)`
	`{ case 'value' : ___`
	`[break]`
	`default : ___`
	`[break]}`

Supérieur à	`a>b // a>=b`
Inférieur à	`b<a // b <=a`
Egal à	`a==b // a!=b`
Opérateur OR	`a||b`
Opérateur AND	`A&&b`
Opérateur NOT	`!a`
Nullish Coalescing	`a ?? b`

2.6. Boucles

Type	Exemple
While	`While (condition) {}`
Do	`Do {} While (condition)`
For	`For (let i=0 ; i >3 ; i++ ){}`

2.7. Interactions

Type	Description	Exemple
Alert	Un message s'affiche jusqu'à ce que l'utilisateur accepte	`alert('hello world')`
Prompt	Accepte l'introduction de deux paramètres où le premier sera utilisé pour afficher un message et le second sera utilisé pour enregistrer ce que l'utilisateur veut taper.	`let test = prompt('Quel âge avez-vous ?','[answer]')`
Confirm	S'affiche un message et s'attend à ce que l'utilisateur choisisse entre confirmer ou annuler.	`confirm("Press a key!");`

2.8. Objets

Type	Exemple
Declare	`let user = new Object()` `let user = {}`
Propriétés	`let user = {name : 'Fede', age :30}`
Définir la propriété	`user['celibataire'] = true`
Récupère la propriété	`user['age']`
Supprimer la propriété	`delate user ['age'];`
Définir le constructor	`constructor (x){this.x = x;}`

2.9. Inputs key

Type	Exemple
Keydown	`document.addEventListener("keydown", function(event) { if (event.key === "Escape") { }});`
Keyup	
Keypress	

2.10. Import/export

En JavaScript, vous pouvez utiliser la fonction "import/export" pour pouvoir extraire des fonctions d'un code à un autre sans qu'elles se trouvent dans le même fichier. Exemple :

export let GameboyMode = () =>{} // import { GameboyMode } from "./emulator.js"

2.11. QuerySelector & GetElementById

Type	Description	Exemple
querySelector()	est une fonction permettant d'appeler un élément à partir de sa classe en HTML et d'accéder à ses valeurs en HTML ou CSS	`const intro = document.querySelector('.intro');`
getElementById()	est une fonction permettant d'appeler un élément à partir de sa ID en HTML et d'accéder à ses valeurs en html ou css.	`const main = document.getElementById('main');`

2.12. Canvas

Type	Description	Exemple
Canvas - HTML	Le canevas est une zone rectangulaire utilisée pour les graphiques d'une page web.	`<canvas id="myCanvas" width="200" height="200">  </canvas>`
Canvas - JS	Pour utiliser le canvas, il faut l'initialiser dans le JS et donner un varlo au style graphique.	`var c = document.getElementById("myCanvas"); var ctx = c.getContext("2d");`
.font .fillText	méthodes d'affichage de texte à l'écran	`ctx.fillText("Hello World!", 10, 50);ctx.font = "30px Verdana"`
.fillStyle .fillRect	méthodes pour afficher un rectangle à l'écran	`ctx.fillStyle = "red"; ctx.fillRect(20, 20, 150, 100);`

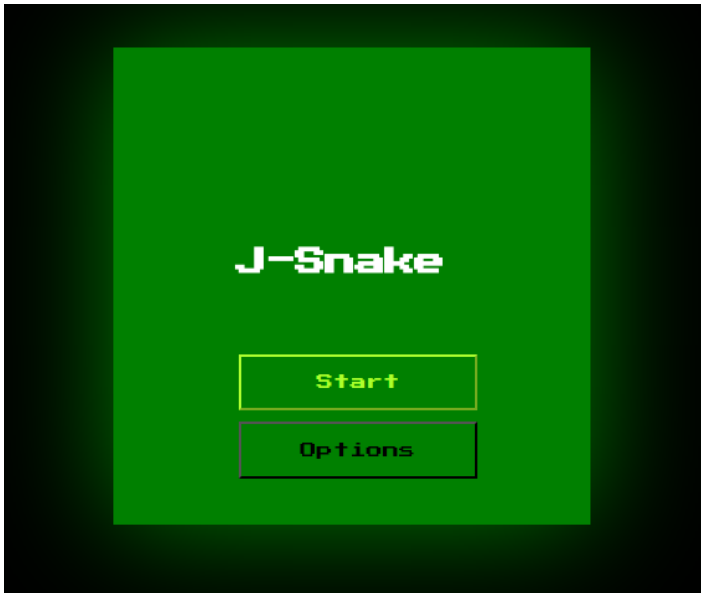
3. J-Snake Game

J-Snake est une réplique du célèbre jeu Snake sorti dans les années 1960. Dans ce jeu, le joueur contrôle un serpent dont l'objectif est de manger autant de pommes que possible sans se manger lui-même ou sortir des limites.

Ce jeu est connu pour être sorti sur une multitude d'appareils et de consoles, notamment le fameux Nokia 3310. Pour cette raison, ce site web offrira au joueur l'expérience de pouvoir choisir un émulateur comme le Nokia ou un GameBoy avec des boutons fonctionnels pour pouvoir jouer à J-Snake.

3.1. Menus de J-Snake

J-Snake est présenté sur un canvas de 400x400px avec une couleur verte et un fond noir. Son premier écran affiche le titre du jeu et deux options, « Start » ou « Options ».



Sous Options, le joueur pourra choisir l'arrière-plan et la manette de jeu (Nokia, Gameboy ou aucune).

La première option initialise l'écran de jeu et n'est activée que lorsque le joueur décide de la direction du serpent. Au cours du jeu, le joueur doit manger le fruit qui lui rapportera 1 point et le score sera affiché.

Enfin, lorsque le jeu est perdu, le joueur peut redémarrer le jeu ou revenir au menu principal.

J-Snake se caractérise par une grande accessibilité des commandes de jeu. Dans les menus, vous pouvez utiliser les claviers pour sélectionner les options désirées, ou vous pouvez utiliser la souris ou les boutons de la manette de jeu du joueur.

Pour vous déplacer dans le jeu, vous pouvez utiliser les touches fléchées, les touches WASD ou cliquer sur les boutons de la manette de jeu.

3.2. Analyse du jeu en code

J-Snake est un site web conçu avec Java Script, HTML et CSS. L'utilisation de Java Script a été concentrée sur le développement d'un moteur de jeu et de boutons

fonctionnels. HTML a été utilisé pour la création de différents éléments qui structurent le site web et CSS a été utilisé pour gérer les couleurs, les positions, les interfaces et l'art visuel du site web.

HTML	CSS	JS
Index.html	css/style.css	js/emulator.js js/fruitClass.js js/snakeClass.js js/index.js

3.2.1. Emulator.js

Le fichier emulator.js stocke et contrôle toutes les données relatives aux contrôleurs de jeu, à leurs boutons et à leurs images.

emulator.js	Description
gameboyMode() nokiaMode() gameboyModeInOption() nokiaModeInOptions() noneModeInOptions()	Ces méthodes ont pour fonction d'afficher les images avec les boutons correspondants sur l'écran. Selon le mode choisi, l'un ou l'autre mode sera activé.
allHiddenAssets() allDisplayAssets()	Méthodes pour cacher tous les éléments visuels indésirables ou les faire réapparaître

3.2.2. FruitClass.js

Le fichier fruitClass.js contient les spécifications pour la création d'un élément comestible qui interagit avec le jeu.

fruitClass.js	Description
Class FoodItem - Constructor (x, y)	Création de la classe fruit de avec ses éléments de position X et Y.
rndFoodPosition()	Méthode pour changer la position des fruits de manière aléatoire avec Math.floor(Math.random()*tileCount)
drawFood()	Méthode pour dessiner l'élément

```
JS fruitClass.js > FoodItem
export class FoodItem {
    constructor (x, y){
        this.x = x;
        this.y = y;
    }
    rndFoodPosition(tileCount){
        this.x = Math.floor(Math.random()*tileCount);
        this.y= Math.floor(Math.random()*tileCount);
    }
    drawFood=(ctx, tileCount, titleSize)=>{
        ctx.fillStyle="red";
        ctx.fillRect(this.x*tileCount, this.y*tileCount, titleSize, titleSize);
    }
}
```

3.2.3. SnakeClass.js

Le fichier snakeClass.js contient les fonctionnalités nécessaires à la création du serpent contrôlé par le joueur. Ce serpent est divisé en deux parties, la tête et la queue, dans un tableau. En outre, il fournit d'autres fonctions pour le positionnement, le dessin et ses valeurs de base.

snakeClass.js	Description
Class FoodItem - Constructor (x, y)	Création de la classe Snake de avec ses éléments de position X et Y
snakePosition ()	Méthode pour changer la position des fruits pour chaque update
drawSnake(ctx, tileCount, titleSize)	Méthode pour dessiner le snake. L'utilisation d'une boucle forEach() permet de mettre à jour la position de tous les corps dans la liste. Une fois la nouvelle position de la tête détectée, le jeu utilise unshift() pour ajouter de nouvelles informations pour la prochaine position prévue et la fonction pop() pour supprimer la dernière case dessinée.
SnakePauseMode()	Enregistre les informations de position du serpent lorsqu'il est en mode Pause
SnakeOrigin()	Rétablit les valeurs par défaut du serpent

3.2.4. Index.js

Le fichier index.js contient à la fois les fonctions destinées à faire fonctionner le moteur du jeu, les collisions et les interactions du HUD du jeu. En outre, il rejoint et interagit avec les autres fichiers à utiliser dans le jeu.

indexClass.js	Description
GameEngine()	Dans cette méthode, il est chargé de développer le jeu. Son fonctionnement consiste à gérer l'état du jeu, qu'il soit en menu, en options, en pause ou en jeu, à mettre à jour les valeurs et à afficher chaque élément.
Menu()	Cette méthode définit le statut du jeu sur Menu uniquement et n'affiche que les éléments du menu : le titre et les boutons de lecture ou d'options.

Options()	Cette méthode permet d'ajuster l'état du jeu en mode Options où les éléments de sélection de l'arrière-plan, les manettes, les images et les boutons seront affichés.
IsGameOver()	Est une méthode qui renvoie l'état du jeu en fonction des conditions de défaite. Si le joueur est en collision avec lui-même ou avec les limites de la carte, le jeu s'arrête et s'affiche Game Over, le score et deux boutons pour rejouer ou retourner au menu de démarrage.
PauseMode()	Dans ce mode, le jeu est mis en pause en enregistrant la dernière position du serpent et en affichant le message Pause.
Movement()	Dans cette méthode, le mouvement du joueur est déterminé en fonction de la direction donnée.
CheckCollision()	Pour contrôler si le joueur touche le fruit ou si le fruit apparaît à l'intérieur du serpent et doit être repositionné.
ChangeButtonMenu()	Méthode qui change la couleur des boutons sur l'écran pour savoir sur quelle option nous nous trouvons.
GameEnterBtn()	Méthode qui accepte l'option lorsqu'elle est sélectionnée. Selon l'état du jeu, vous pouvez accéder à l'option ou la confirmer.
ClearScreen()	Méthode pour nettoyer l'écran de tous ses éléments.
DrawScore()	Affiche le score du joueur à l'écran
BtnActions()	Liste d'actions pour les boutons du jeu ou les boutons de la manette utilisée.

3.2.5. JavaScript caractéristiques

JavaScript possède des fonctionnalités uniques qui permettent d'adapter le code pour les web site et de développer un code plus optimal avec une variété d'outils uniques. Beaucoup d'entre eux ont été expliqués dans la section 2 de ce document, et ci-dessous, une explication détaillée de la façon dont ces fonctionnalités ont été appliquées dans ce jeu :

- Syntaxe moderne avec les variables const et let : En raison de l'intégration des variables Const et Let, il n'a pas été nécessaire de créer des variables de style Var.
- Les classes Snake et Fruit ont été saisies avec leurs fichiers de type module correspondants afin qu'elles puissent être exportées et importées dans le code principal.

```
import { SnakeTailPart } from "./snakeClass.js";
import {FoodItem} from "./fruitClass.js";

//Snake & Fruit class
let snakeTailPart = new SnakeTailPart()
let fruitItem = new FoodItem()
```

- Seules les boucles ForEach ont été utilisées pour le développement du jeu et les contrôles d'état. De même, les fonctions ont également été optimisées pour utiliser le système de fonctions avec des flèches afin d'optimiser le code.

```
snakeTailPart.snakeTail.forEach((SnakeTailPart, i )=>{
    if(snakeTailPart.snakeTail[i].x==fruitItem.x && snakeTailPart.snakeTail[i].y ==fruitItem.y){
        fruitItem.RndFoodPosition(tileCount);
    }
```

- Enfin, l'opérateur Rest a été utilisé comme élément pour pouvoir afficher les données dans la console en format de tableau.

```
let CheckCollision = (...positionInfo) =>{
        positionInfo.forEach((info, e)=>{
            console.log(info)
        })
```

4. Conclusions

J-Snake est mon deuxième projet de site web en JavaScript, où j'ai pu apprendre de nouvelles fonctionnalités de ce langage et compléter mes connaissances antérieures.

La construction du jeu Snake a été développée facilement grâce à mes connaissances antérieures de ce jeu en C#. En raison de l'utilisation de JavaScript, l'adaptation de ce code a présenté des difficultés telles que les interactions avec les boutons, un ordre correct dans la structure du code, l'utilisation de différentes fonctions et de nouveaux outils pour les tableaux. Mais une fois que j'ai appris à utiliser ces nouveaux éléments, j'ai pu développer le jeu complètement et librement selon mes idées.

C'est pour cette raison que le développement d'un jeu simple avec une liberté totale de réalisation m'a rendu curieux de ce langage et m'a donné envie de développer plus d'éléments dans le jeu et d'apporter de l'originalité.

4.1. Améliorations

Voici une liste d'idées et d'améliorations diverses que, par manque de temps ou de connaissances, je n'ai pas pu appliquer à mon projet :

- Créer une classe pour chaque élément : Score, boutons de contrôle et HUD.
- Utiliser RequestAnimationFrame() comme alternative à SetTimeout()
- Avoir un site web responsive qui soit correct et adaptable à l'écran.
- Bugs : en mode pause, un élément peut être temporairement masqué à l'écran.

5. Sources

- W3School : https://www.w3schools.com/
- JavascriptInfo : www.javascript.info
- DeepL : www.deepl.com/translator
- Chat GPT n'a pas été utilisé pour la réalisation de ce projet