Snake JS



Luca Premat – FID2

Table des matières

[1. Introduction 3](#_Toc212822544)

[2. Fonctionnalités 3](#_Toc212822545)

[3. Déroulement de la programmation : 4](#_Toc212822546)

[3.1. Collisions : 4](#_Toc212822547)

[3.2. Contrôles : 6](#_Toc212822548)

[3.3. Nourriture : 7](#_Toc212822549)

[3.4. Score et Timer : 8](#_Toc212822550)

[3.5. Snake : 10](#_Toc212822551)

[3.6. Main : 11](#_Toc212822552)

[4. Usage de l’ia 13](#_Toc212822553)

[5. Conclusion 15](#_Toc212822554)

# Introduction

Lors de ce projet nous allons développer le jeu Snake grâce au langage de programmation Javascript, le projet sera effectué sur un total de 24 périodes.

# Fonctionnalités

* 1. **Déplacement du serpent :**

Le serpent peut être contrôlé à l'aide des touches directionnelles (haut, bas, gauche, droite).

* 1. **Gestion des collisions :**

Si le serpent touche un mur ou son propre corps, la partie est terminée.

* 1. **Gestion du Score** :

Un système de score affiche le nombre de fois où le serpent a mangé de la nourriture (représentée en rouge sur l’image ci-dessus).

* 1. **Chargement Asynchrone de la Configuration**

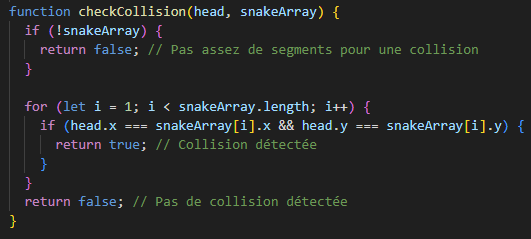
Externaliser les paramètres essentiels du jeu, tels que la taille des cases et la vitesse du jeu, dans un fichier de configuration **config.json**, qui sera chargé de manière asynchrone au démarrage du jeu. Cette approche permet de centraliser et de gérer facilement les configurations, tout en préparant le code à des évolutions futures.

* 1. **Gestion du Temps**

Ajouter un chronomètre pour mesurer le temps écoulé pendant chaque partie, affiché en haut de l'écran.

# Déroulement de la programmation :

## Collisions :



Ici on vérifie si le serpent entre en collision avec sa propre queue. On commence par s’assurer que le serpent a une queue, si ***snakeArray*** = 0, on retourne false car il n’y a pas de collision.

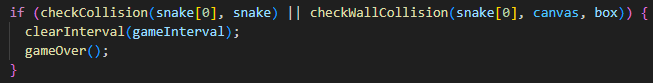
Ensuite, si le serpent a une queue, on détecte une collision en regardant si les coordonnées x et y de la tête sont les mêmes que les coordonnées x et y de l’un des segments de la queue. Si c’est le cas on retourne ***true*** pour indique qu’il y a une collision, sinon false.

De plus il faut vérifier si la tête entre en collision avec un des quatre murs, pour se faire nous avons ce code :

A screen shot of a computer screen

AI-generated content may be incorrect.

Ici on regarde si les coordonnées ***X*** et ***Y*** de la tête du serpent, sortent des limites du canevas

Dans le fichier main, nous avons ce code qui fait gameOver si une collision est détectée :  


Finalement on détecte les collisions avec la nourriture :

A screen shot of a computer code

AI-generated content may be incorrect.

On vérifie si les coordonnées ***X***et ***Y*** de la tête sont égales à celles de la nourriture.

Si une collision avec la nourriture est détectée, on agrandit le serpent :

A black screen with white text

AI-generated content may be incorrect.

## Contrôles :

Pour les déplacements on attend dans le main qu’une touche soit pressée :

A black screen with white text

AI-generated content may be incorrect.

Puis on vérifie quelle touche a été pressée et on retourne la direction qui correspond :

A screen shot of a computer code

AI-generated content may be incorrect.

Finalement on utilise cette direction pour redessiner le serpent au bon endroit :

A computer screen shot of code

AI-generated content may be incorrect.

## Nourriture :

Pour la nourriture on commence par définir où celle-ci peut apparaître, puis on génère une coordonnées aléatoire comprise dans les limites du canevas :

A screen shot of a computer code

AI-generated content may be incorrect.

Puis on dessine la nourriture :

A black background with white text

AI-generated content may be incorrect.

## Score et Timer :

On crée l’affichage du score à l’écran :

A screen shot of a computer code

AI-generated content may be incorrect.

A chaque fois que le serpent mange de la nourriture on incrément ***score*** de 1 :

A black screen with white text

AI-generated content may be incorrect.

(Note : on aurait aussi pu faire ***score++*** mais comme ça on peut choisir combien vaut la nourriture)

Et on affiche le score final au gameOver :



On crée l’affichage du timer à l’écran :

A computer screen shot of a code

AI-generated content may be incorrect.

Au début du jeu on initialise *startTime* pour avoir la date de début de la partie :

A screen shot of a computer code

AI-generated content may be incorrect.

Pour calculer le temps on utilise cette formule :



On fait *Date.now()* à chaque rafraichissement puis on soustrait *startTime* pour avoir le temps qui s’est écoulé depuis le début de la partie, et à ça on vient soustraire *pausedTime*, qui est le temps passé en pause, calculé ainsi :

A computer screen with text

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

À la fin de la partie on affiche le temps total :



On divise par 1000 pour passer de millisecondes à secondes.

## Snake :

Ici on gère l’affichage du serpent, on y trouve ce bout décrit précédemment :

A screen shot of a computer code

AI-generated content may be incorrect.

Ainsi que le code qui dessine le serpent :

A computer code on a black background

AI-generated content may be incorrect.

## Main :

Dans le fichier main on trouve le cœur du jeu ;

Les variables :

A screen shot of a computer

AI-generated content may be incorrect.

Méthode ***mainMenu()***qui affiche le menu principal :

A screen shot of a computer code

AI-generated content may be incorrect.

Méthode ***startGame()***:

A computer screen shot of text

AI-generated content may be incorrect.

***gameOver()***:

A computer code on a black background

AI-generated content may be incorrect.

***pauseGame()***:

A screen shot of a computer code

AI-generated content may be incorrect.

***resumeGame()***:

A screen shot of a computer code

AI-generated content may be incorrect.

Gère les évènements de mise en pause et de reprise du jeu :

A screen shot of a computer program

AI-generated content may be incorrect.

# Usage de l’ia

J’ai utilisé l’ia pour monter en compétence sur divers points, notamment pour l’utilisation et le fonctionnement de Date.Now().

J’ai utilisé copilot pour vérifier que mon projet correspondait bien aux attentes du cahier des charges, en lui donnant celui-ci.

Voici le prompt :

Est-ce que mon projet respecte tous les critères demandés, sans tenir compte du tableau des scores et des test unitaires : [cahier des charges]

Et le résultat :

A screenshot of a computer screen

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer program

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

# Conclusion

Le projet Snake m’a permis de mettre en pratique les bases du développement JavaScript. J’ai également appris à utiliser **JSDoc** pour documenter le code de manière professionnelle. Ce projet m’a aidé à mieux comprendre la logique de jeu, la gestion du temps et des collisions, tout en renforçant mes compétences en programmation et en organisation du code.

Ce projet m’a également donné envie de continuer à explorer JavaScript, notamment pour créer d’autres petits jeux ou interfaces interactives.