# Rapport Final : Projet Snake en JavaScript



Yosef Nademo – CID2B

Rue de Sébeillon 12, 1004 Lausanne

40 périodes

Table des matières

[Rapport Final : Projet Snake en JavaScript 1](#_Toc185595077)

[1. Introduction 3](#_Toc185595078)

[2. Description du projet 3](#_Toc185595079)

[Paramètres initiaux 3](#_Toc185595080)

[**Objectif du jeu** 3](#_Toc185595081)

[**Fonctionnalités principales** 3](#_Toc185595082)

[Technologies utilisées 4](#_Toc185595083)

[3. Structure du projet 4](#_Toc185595084)

[**4. Problèmes rencontrés et solutions** 4](#_Toc185595085)

[Problème 1 : Organisation modulaire 4](#_Toc185595086)

[Problème 2 : Sauvegarde des scores 5](#_Toc185595087)

[Problème 3 : Gestion des collisions 5](#_Toc185595088)

[Processus de test et débogage 5](#_Toc185595089)

[5. Recours à l’IA 6](#_Toc185595090)

[6. Évaluation personnelle 6](#_Toc185595091)

[7. Instructions pour exécuter le projet 7](#_Toc185595092)

[Annexes : 7](#_Toc185595093)

## 1. Introduction

Le projet consiste en la réalisation d’un jeu Snake en JavaScript, une réplique moderne du célèbre jeu classique. Ce rapport documente le développement, les fonctionnalités implémentées, les défis rencontrés, et les technologies utilisées, tout en mettant en avant l’apprentissage et les compétences acquises.

## 2. Description du projet

### Paramètres initiaux

Les paramètres suivants définissent le comportement initial du jeu et son évolution au cours de la partie :

* **Vitesse initiale du serpent** : La vitesse commence à 5 cases par seconde. Cette vitesse augmente progressivement de 10 % chaque fois que le serpent mange une pomme.
* **Taille de la grille** : La grille mesure initialement 20x20 cases, offrant un équilibre entre difficulté et jouabilité. Cette taille reste fixe pendant toute la partie.
* **Longueur initiale du serpent** : Le serpent commence avec une longueur de 3 segments.

**Objectif du jeu**

Le joueur contrôle un serpent qui se déplace sur une grille. L'objectif est de manger de la nourriture pour grandir tout en évitant de heurter les murs ou le propre corps du serpent. Le score augmente à chaque fois que le serpent mange de la nourriture.

**Fonctionnalités principales**

* **Déplacement du serpent :** Le serpent est contrôlé à l'aide des touches directionnelles (haut, bas, gauche, droite). Les déplacements opposés immédiats sont désactivés.
* **Gestion des collisions :** La partie se termine si le serpent heurte un mur ou son propre corps.
* **Score :** Un score est affiché en haut de l’écran et augmente à chaque fois que le serpent mange une pomme.
* **Chargement asynchrone des configurations :** Les paramètres tels que la taille de la grille et la vitesse sont externalisés dans un fichier config.json.
* **Top 5 des scores :** Les meilleurs scores sont sauvegardés via une API distante, avec gestion de l’égalité par le temps de jeu.
* **Pause/Reprise :** L’utilisateur peut mettre en pause ou reprendre la partie avec la touche espace. Pendant la pause, le chronomètre est suspendu.

### Technologies utilisées

* **Node.js et npm** : Gestion des dépendances. ([Télécharger Node.js](https://nodejs.org/))
* **Vite** : Serveur de développement rapide avec rechargement automatique. ([Documentation Vite](https://vitejs.dev/))
* **Modules ES** : Organisation du code en fichiers modulaires.
* **JSDoc** : Documentation du code.
* **API jsonbin.io** : Sauvegarde des scores.

## 3. Structure du projet

Le projet est organisé selon les bonnes pratiques de développement modulaire :

* **main.js :** Point d'entrée principal du jeu.
* **snake.js :** Gestion du serpent (initialisation, déplacement, dessin).
* **food.js :** Génération et dessin de la nourriture.
* **controls.js :** Gestion des entrées utilisateur.
* **collision.js :** Vérification des collisions.
* **score.js :** Gestion et affichage du score.
* **config.json :** Paramètres du jeu (taille de grille, vitesse).

Le répertoire contient également un fichier README.md décrivant le projet et les instructions pour l’exécuter.

**4.** Problèmes rencontrés et solutions

### Problème 1 : Organisation modulaire

* **Description** : Difficultés initiales à diviser les responsabilités en modules.
* **Solution** : Recherche sur les bonnes pratiques de conception modulaire en JavaScript.

### Problème 2 : Sauvegarde des scores

* **Description** : Complexité de la gestion des scores via une API distante.
* **Solution** : Documentation de l’API jsonbin.io et tests approfondis des requêtes.

### Problème 3 : Gestion des collisions

* **Description** : Vérification inefficace des collisions avec le corps du serpent.
* **Solution** : Optimisation des boucles et des comparaisons dans collision.js.

### Processus de test et débogage

* **Description** : Des tests manuels ont été effectués pour valider le bon fonctionnement de chaque module et de leurs interactions.
  + Vérification du chargement et de la mise à jour des scores via jsonbin.io : Tests effectués avec différentes valeurs pour s'assurer que les données sont correctement sauvegardées et récupérées.
  + Simulation de scénarios de jeu extrêmes, tels que la vitesse maximale ou des grilles remplies, pour identifier des comportements inattendus.
  + Tests des contrôles utilisateur pour s’assurer qu’ils répondent sans latence.
* **Bugs identifiés et corrigés** :
  + **Bug 1** : Décalage lors de l’augmentation de la vitesse du serpent. Résolu en ajustant les paramètres du setInterval.
  + **Bug 2** : Duplication de la nourriture dans certains cas. Résolu en vérifiant l’absence de collision avant la génération d’une nouvelle pomme.
  + **Bug 3** : Non-mise à jour du score dans certains navigateurs. Résolu en ajoutant des polyfills pour la compatibilité JavaScript.

## 5. Recours à l’IA

L’IA a été utilisée dans les contextes suivants :

* **Apprentissage :** Compréhension des fonctionnalités avancées de JavaScript et de l'utilisation de Vite.
* **Inspiration :** Recherche d'idées pour organiser le code et résoudre des problèmes spécifiques (ex. : gestion des collisions).

Aucun code n'a été copié directement de l'IA. L’objectif principal était de renforcer mes compétences en JavaScript tout en respectant les exigences du projet.

## 6. Évaluation personnelle

Ce projet m'a permis de :

* Approfondir mes connaissances en JavaScript, notamment sur les modules ES et la gestion asynchrone.
* Découvrir l’utilisation d’outils modernes comme Vite.
* Renforcer ma capacité à structurer un projet et à collaborer via un système de versionnement (Git).

Je suis satisfait du résultat final et des fonctionnalités implémentées, tout en identifiant des pistes d’amélioration pour mes futurs projets.

**Idées pour futurs améliorations du jeu**:

* **Ajout de niveaux** : Introduire des niveaux avec des obstacles plus complexes, des ennemis supplémentaires ou des variations de la grille pour offrir une expérience de jeu plus diversifiée et stimulante.
* **Amélioration de la graphisme** : Refonte visuelle avec de nouveaux sprites pour le serpent, la nourriture et les murs. Intégration de l’animation pour le mouvement du serpent et des transitions plus fluides pour une meilleure expérience utilisateur.
* **Nouvelles mécaniques de jeu** : Ajout de nouvelles fonctionnalités telles que des power-ups, des pièges, ou des défis temporels pour augmenter la difficulté et l'intérêt du jeu. Par exemple, permettre au joueur d’utiliser des items spéciaux pour rétrécir ou élargir temporairement le serpent, ou introduire un mode difficile avec des obstacles mobiles.
* **Compatibilité multiplateforme** : Améliorer la compatibilité pour différents appareils et navigateurs, en particulier pour les appareils mobiles. Intégration d’un mode tactile pour une meilleure jouabilité sur smartphones et tablettes.
* **Mode multijoueur** : Ajouter un mode multijoueur local ou en ligne où plusieurs joueurs peuvent s’affronter sur la même grille. Cela apporterait une dimension compétitive au jeu.
* **Fonctionnalités sociales** : Intégrer des fonctionnalités sociales, comme la possibilité de partager les scores sur les réseaux sociaux ou de comparer les performances avec des amis via un classement global.

## 7. Instructions pour exécuter le projet

1. **Pré-requis :**
   * Installer Node.js.
   * Cloner le dépôt GitHub.
2. **Installation :** cd snake npm install
3. **Lancement du serveur de développement :** npm run dev
4. Accéder au jeu à l'adresse [http://localhost:3000](http://localhost:3000/) OU avec extension « Live Server ».

## Annexes :

* **Captures d’écran du jeu** :
  + Capture de l’écran d’accueil.
  + Capture en cours de jeu avec score affiché.
* **Liens vers le dépôt GitHub** : GitHub – Snake Project.
* **Documentation générée avec JSDoc(**Le code est documenté avec **JSDoc)** : Disponible dans le répertoire docs.