# 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](#_Toc187396588)

[1. Introduction 3](#_Toc187396589)

[2. Description du projet 3](#_Toc187396590)

[Paramètres initiaux 3](#_Toc187396591)

[**Objectif du jeu** 3](#_Toc187396592)

[**Fonctionnalités principales** 3](#_Toc187396593)

[Technologies utilisées 4](#_Toc187396594)

[3. Structure du projet 4](#_Toc187396595)

[**4. Problèmes rencontrés et solutions** 4](#_Toc187396596)

[Problème 1 : Apprentissage de JavaScript et Utilisation des Ressources en Ligne 4](#_Toc187396597)

[Problème 2 : Sauvegarde des scores 5](#_Toc187396598)

[Problème 3 : Gestion des collisions 6](#_Toc187396599)

[Processus de test et débogage 6](#_Toc187396600)

[5. Recours à l’IA 7](#_Toc187396601)

[6. Évaluation personnelle 7](#_Toc187396602)

[7. Instructions pour exécuter le projet 8](#_Toc187396603)

[Annexes : 9](#_Toc187396604)

## 1. Introduction

Le projet consiste en la réalisation d’un jeu Snake en JavaScript, une réplique moderne du célèbre jeu classique.Ce projet vise à créer un jeu classique du serpent avec des fonctionnalités modernes, telles que la gestion des scores, un menu interactif et des contrôles de jeu fluides. Le développement a été réalisé en utilisant des technologies web modernes (HTML, CSS, JavaScript) pour assurer la compatibilité avec les navigateurs web.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 (et même aléatoire). Cette taille reste fixe pendant toute la partie.
* **Longueur initiale du serpent** : Le serpent commence avec une longueur de 1 segment(tête).

**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.
* **Menu interactif** : Avant de commencer, l'utilisateur peut sélectionner la difficulté et démarrer une nouvelle partie. Un bouton de **replay** permet de rejouer après la fin de la partie.

### Technologies utilisées

* **Node.js et npm** : Gestion des dépendances. ([Télécharger Node.js](https://nodejs.org/))
* **Modules ES** : Organisation du code en fichiers modulaires.
* **JSDoc** : Documentation du code.
* **API jsonbin.io** : Sauvegarde des scores.
* **HTML** : Structure de la page, y compris le canvas pour afficher le jeu.
* **CSS** : Mise en page et design du je2u, y compris l'animation des boutons.
* **JavaScript** : Logique du jeu, gestion des collisions, mouvement du serpent et gestion 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).
* **Index.**html : le squelette de pret(joue).

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

**4.** Problèmes rencontrés et solutions

### Problème 1 : Apprentissage de JavaScript et Utilisation des Ressources en Ligne

* **Description du problème** : Le projet m'a demandé de bien comprendre JavaScript, surtout pour des concepts un peu plus avancés comme la gestion des événements, l'organisation du code en modules, et l'utilisation des API. Comme je n'avais pas encore une maîtrise complète de JavaScript, j'ai dû faire face à plusieurs défis pour résoudre des problèmes spécifiques et comprendre certains concepts essentiels.
* **Solution** : Pour surmonter ces obstacles et améliorer mes compétences en JavaScript, j'ai cherché activement des solutions et des tutoriels sur des sites comme YouTube, StackOverflow, et la documentation officielle de JavaScript. Ces ressources m'ont permis d'apprendre de nouvelles techniques, de trouver des solutions à des problèmes complexes et de mieux comprendre les erreurs courantes. Par exemple, des vidéos YouTube et des discussions sur StackOverflow m'ont bien aidé à comprendre la logique derrière des fonctionnalités comme la gestion des collisions, l'utilisation de localStorage et la gestion des fonctions asynchrones.

J'ai aussi consulté des articles et des tutoriels pour me familiariser avec les bonnes pratiques de développement, comme organiser le code en modules et gérer l'état du jeu de manière efficace.

**Exemples de ressources utilisées :**

1. **YouTube** : Tutoriels vidéo sur la gestion des événements en JavaScript, les promesses asynchrones et l'utilisation de localStorage.
2. **StackOverflow** : Discussions sur des problèmes spécifiques tels que la gestion des scores, la création de la grille du jeu et l'optimisation des performances.

* **Résultats obtenus :** Cette approche d'apprentissage autodidacte m'a permis non seulement de surmonter les défis techniques du projet, mais aussi de renforcer mes compétences en JavaScript. Grâce à ces ressources, j'ai pu maîtriser des concepts avancés de programmation, améliorer la structure de mon code, et résoudre efficacement les problèmes liés à l'implémentation des fonctionnalités du jeu.

### Problème 2 : Sauvegarde des scores

* Description du problème : Le projet avait besoin d'un mécanisme pour sauvegarder le meilleur score de chaque session de jeu. La gestion des scores a été un défi important, car il fallait non seulement enregistrer le score à chaque fois que le joueur terminait une partie, mais aussi s'assurer que le meilleur score était conservé au-delà des sessions de jeu. Il était crucial de garder une trace des meilleurs scores tout en permettant au joueur de consulter cette information lors de chaque nouvelle session.
* Solution mise en place : Pour résoudre ce problème, la méthode choisie a été d'utiliser localStorage pour stocker de manière persistante le meilleur score. En utilisant localStorage, les scores sont conservés même après que l'utilisateur quitte le jeu ou ferme le navigateur. Le système compare le score actuel du joueur à celui stocké et met à jour le meilleur score lorsque cela est nécessaire.

### Problème 3 : Gestion des collisions

* **Description du problème** : Le dernier gros problème était lié à la gestion des collisions avec le corps du serpent. Au début, tout semblait bien fonctionner, mais j'ai vite remarqué qu'il y avait des bugs : parfois le serpent se mordait tout seul sans que la partie ne s'arrête. Le problème venait de ma méthode de détection des collisions entre la tête du serpent et son propre corps. Ce n'était pas assez précis et ça causait des erreurs.
* **Solution** : Pour régler ça, j'ai créé une méthode plus fiable qui compare la position de la tête avec celle de chaque segment du serpent. Si la tête se retrouve sur un segment déjà occupé, la partie est terminée. Cela a bien résolu le problème et le jeu fonctionne maintenant comme prévu

### Processus de test et débogage

* **Description** : Des tests manuels ont été effectués pour valider le bon fonctionnement de chaque module et de leurs interactions.
  + 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.
  + Test de sauvgardes autimatiques et imediats.
* **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.
  + **Bug 4 :** Problème avec le tableau des meilleurs scores qui ne se mettait pas à jour après chaque partie. Résolu en assurant une sauvegarde et une récupération des scores correctement synchronisées avec l’API de stockage.
  + **Bug 5 :** Mouvement du serpent qui ne répondait pas correctement aux entrées dans certains cas (par exemple, lorsque les touches étaient pressées rapidement). Résolu en améliorant la gestion des événements clavier, en désactivant les mouvements opposés immédiats pour éviter des comportements inattendus.

## 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.
* Découvrir l’utilisation d’outils modernes comme node.js.
* 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. Exécuter le serveur avec la commande npm run dev ou utiliser l'extension **Live Server** de Visual Studio Code pour démarrer le projet(personnellement j’ai utilisé l’extension « Live Srver »).
4. Accéder au jeu à l'adresse [http://localhost:3000](http://localhost:3000/) OU avec extension « Live Server ».

Personelement j ;ai utiliser le faeture de vsc liveserver

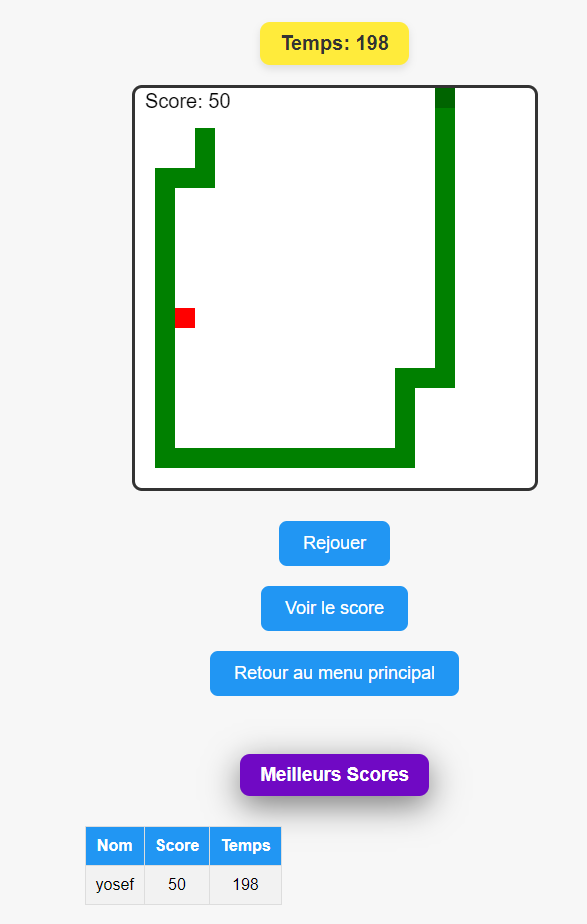
## Annexes :

* **Captures d’écran du jeu** :
  + Capture de l’écran d’accueil.

A screenshot of a chat

Description automatically generated

* + Capture en cours de jeu avec score affiché.



* **Liens vers le dépôt GitHub** : https://github.com/Josefnademo/Bulle-JS
* **Documentation générée avec JSDoc(**Le code est documenté avec **JSDoc)** : Disponible dans le répertoire docs.