2025

Jessica GrisalesB

FID2

15/09/2025

SNAKE GAME

Projet Personnel

****

Table des matières

[Introduction 2](#_Toc210638211)

[Planification 3](#_Toc210638212)

[Analyse de la structure modulaire 3](#_Toc210638213)

[Environnement de développement 3](#_Toc210638214)

[Conception du projet 4](#_Toc210638215)

[Accessibilité 4](#_Toc210638216)

[Conclusion 4](#_Toc210638217)

[Usagede l’IA 4](#_Toc210638218)

## Introduction

Lors du module P\_Bulle un défi nous a été confié, la réalisation d’un classique et incontournable jeu du serpent. Ce projet devra être réalisé en utilisant le JavaScript natif pour le navigateur et en suivant les instructions données dans le cahier des charges.

Ce réplica devra être dessinée sur un canvas HTML, le css pour le style et le code JavaScript utilisera la syntaxe des modules ES (ECMAscript Modules, ESM). L’utilisation du serveur de développement sera gérée par Vite qui possède des atouts qui mettra en valeur la rapidité et une configuration minimale.

La structure du projet a été donné préalablement, cette mise en forme est imposé. Il n’est pas autorisé à la supprimer. Celle-ci compte avec architecture modulaire qui facilite la compréhension de chaque élément et donne un ordre car chaque fonctionnalité est distinctement déparée.

## Planification

La structure de travail pour mener à bien la réalisation du jeux snake, se fera à l’aide d’un journal de travail qu’on doit veiller à qu’il soit à jour. De plus, l’outil GitHub sera également un outil indispensable pour suivre l’avancé du projet.

## Analyse de la structure modulaire

Premièrement avant de commencer, il est important de connaître et comprendre le rôle et la fonction de chaque module. Le projet possède le module snake qui sert à dessiner, initialiser, mettre en mouvement et dessiner le serpent. Puis, le module score est assez simple, il permet de compter et afficher le score. En ce qui concerne le module food, celui-ci est intéressant car il est charger de dessiner et de générer la nourriture. Le module controls aura comme particularité donner la direction de mouvement au serpent grâce aux touches directionnelles du clavier. Et pour finir, le module collision, consiste à vérifier si le serpent a bien touché l’une des limites du carré.

## Environnement de développement

Le jeux Snake sera lancé directement dans le navigateur web, afin de permettre son affichage de manière correcte, la configuration du serveur est capitale. Il faut bien qu’il soit sur le port : 3000.

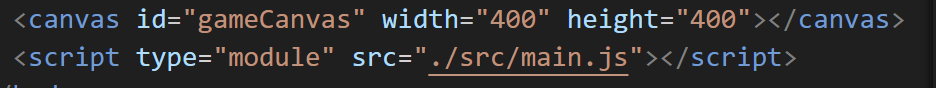
Lors de l’installation et configuration de cet environnement, Une erreur empêchait de voir le jeux sur le navigateur. Le problème était lié au fait que l’exécution du fichier HTML directement avec open with live server ne donner pas directement accès au port 3000. Donc, une fois vérification que le server tournait bien à la racine du fichier, le lancement de ce dernier avec npm run dev montrait le lien http du jeux pour voir son affichage sur le navigateur.

La raison pour laquelle il n’est pas possible d’éxécuter le jeux sans le lien http est dû au modules EMAScript (ESM) (<script type="module">). Les navigateurs actuels bloquent le chargement des modules locaux pour des raisons de sécurité (CORS et origin null). C’est pourquoi il est conseillé d’utiliser le server de développement Vite.

Procédure de lancement (Vite) : Installation des dépenses avec npm install, lancement du server avec npm run dev, et pour finir http://localhost:3000/.

## Conception du projet

Premièrement la prise de connaissance du fichier HTML car l’information qui est représentée donne le point de départ.



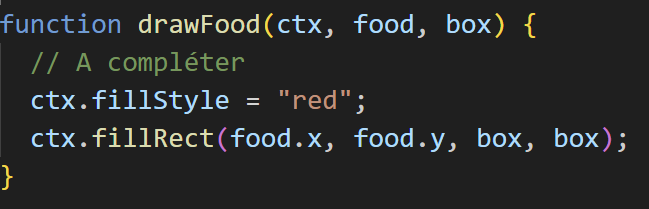
1. **Création du code du module snake et food :**

Cette partie est fondamental pour réussir à afficher le premier carré de nourriture et la tête du serpent. En premier lieu, il faut créer la fonction initSnake qui sera le point de départ du serpent. Grâce au tableau d’objet {Array<{x: number, y: number}>} qui représente un carré du corps. Ensuite, la fonction drawSnake sert à enfin dessiner le serpent dans le canvas. Mais ce n’était pas si évident, car trouver le moyen d’assigner la couleur vert au carré (tête) dû à la syntaxe spécifique a été difficile de mettre en place.

Une image contenant texte, capture d’écran, Police

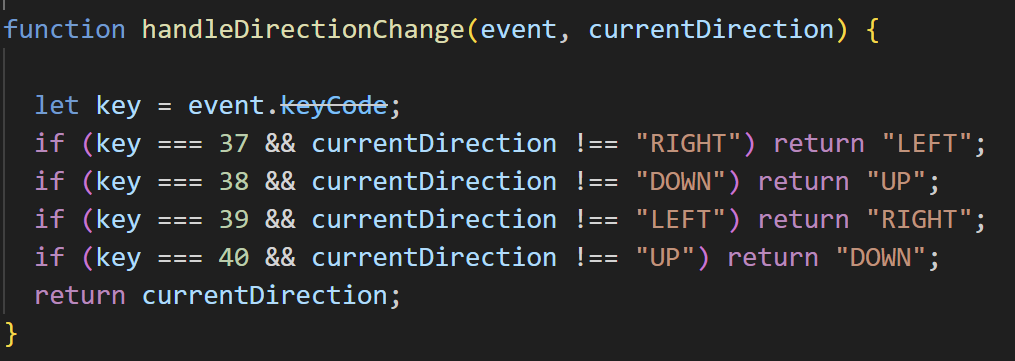
Le contenu généré par l’IA peut être incorrect.

En ce qui concerne la partie nourriture (food), Il fallait afficher un carré rouge cette fois-ci. Mais avec la spécificité que le carré de nourriture devait s’apparait aléatoirement dans le canvas. C’est pourquoi, l’option d’utiliser random est plutôt pertinente. Cependant, dans ce cas, réaliser la syntaxe adéquate pour que ce soit valide a également été difficile. Maintenant, il ne manque plus qu’attribuer la couleur rouge et donner la position dans le canvas ainsi que la taille du carré grâce à cette fonction.



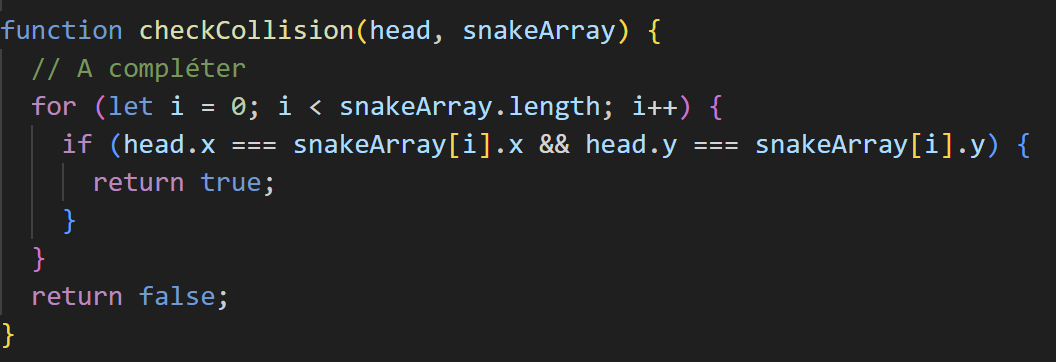
Une fois que les modules food et snake sont prêts pour donner vie au jeux, il faut compter avec la fonction draw dans le module main qui sera expliqué plus tard.

1. **Création des directionnelles (controls)**

Cette partie est fondamental pour changer le serpent de direction et que ce soit jouable. Il est donc important de créer la fonction handleDirectionChange(event, currentDirection). Cette partie est plus simple et courte à construire, toutefois la problématique de savoir et trouver qu’il fallait utiliser keyCode grâce à Event n’a pas été si évident. Une fois la variable let key = event.keyCode; mis en place, le reste du code était devenu plus simple à élaborer. 

1. **Définition des collisions**

Etablir les conditions de collisions qui permettent de savoir ce qui va se passer lorsque le serpent touchera les bords du canvas ou son propre corps. Actuellement sans une définition adéquate le serpent aura tendance à continuer son parcourt même s’il touche son propre corps. C’est alors qu’il faut créer une fonction conditionnelle comme checkCollision(head, snakeArray) pour que la partie s’arrête lorsque le serpent se touche lui-même.



La difficulté lors de la création se cette condition a été de comprendre qu’en JavaScript pour donner une égalité il faut utiliser un triple égale.

Maintenant, il faudrait aussi définir les conditions de collisions lorsque le serpent touche les bords du canvas. Grâce à la fonction checkCollision(head, snakeArray). Cette fonction est importante car pour l’instant le serpent ne détecte pas les bords et continue son chemin en dehors du canvas. L’élaboration de cette fonction a été compliquée car il fallait utiliser le .width et .height pour qu’il prenne en compte la longueur et largeur du canvas.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

1. Mise en place du score sss

## Accessibilité

## Conclusion

## Usagede l’IA