P\_Bulles-Snake

Ethan Schafstall

CID2B, ETML

07.11.2023-10.01.2024

Table des matières

[1 Introduction 1](#_Toc155640233)

[2 Analyse 1](#_Toc155640234)

[2.1 Déplacement du serpent 1](#_Toc155640235)

[2.2 Interaction pomme/serpent 2](#_Toc155640236)

[2.3 Interaction serpent/bordure 4](#_Toc155640237)

[2.4 Interaction serpent tête/corps 6](#_Toc155640238)

[3 Implémentation 6](#_Toc155640239)

[4 Conclusion 6](#_Toc155640240)

[4.1 Tests 6](#_Toc155640241)

[5 Annexe 7](#_Toc155640242)

# Introduction

Le projet P\_Bulles a comme bût la familiarisation, et l’apprentissage de javascript. Pour attendre ces objectives il est demandé de créer le célèbre jeux Snake en application web en utilisant l’**HTML**, le **CSS**, et le **javascript** avec l’environnement **node.js**

Durant le projet c’est aussi fortement recommandé de créer un "cheetsheet" personnel avec des infos/astuces qui nous servirais pour de futurs modules durant le long de notre formation.

# Analyse

Pour la réalisation de notre jeu Snake, nous devons effectuer une analyse du fonctionnement du jeu afin de le recréer. Le jeu Snake peut être divisé en quatre éléments principaux :

1. **La grille de jeu** : La grille de jeu sert d'environnement sur lequel le serpent et la pomme vont être positionnés. Elle délimite la zone dans laquelle les deux autres éléments ont le droit d'exister.
2. **La pomme** : La pomme représente l'objectif du jeu, le but que le joueur doit atteindre. Dès que le serpent mange la pomme, une nouvelle pomme apparaît de manière aléatoire sur la grille du jeu, prête à être consommée à nouveau.
3. **Le serpent** : Le serpent est l'entité qui représente l'utilisateur et constitue le véhicule avec lequel l'utilisateur peut interagir dans le monde du jeu. Le serpent se déplace de manière automatique, mais l'utilisateur a l'influence sur la direction de son déplacement.

## Déplacement du serpent

La première chose à analyser est le déplacement du serpent. À chaque frame (mise à jour du visuel), le serpent se déplace d'un carré selon la direction de sa tête. Pendant ce déplacement, deux événements se produisent :

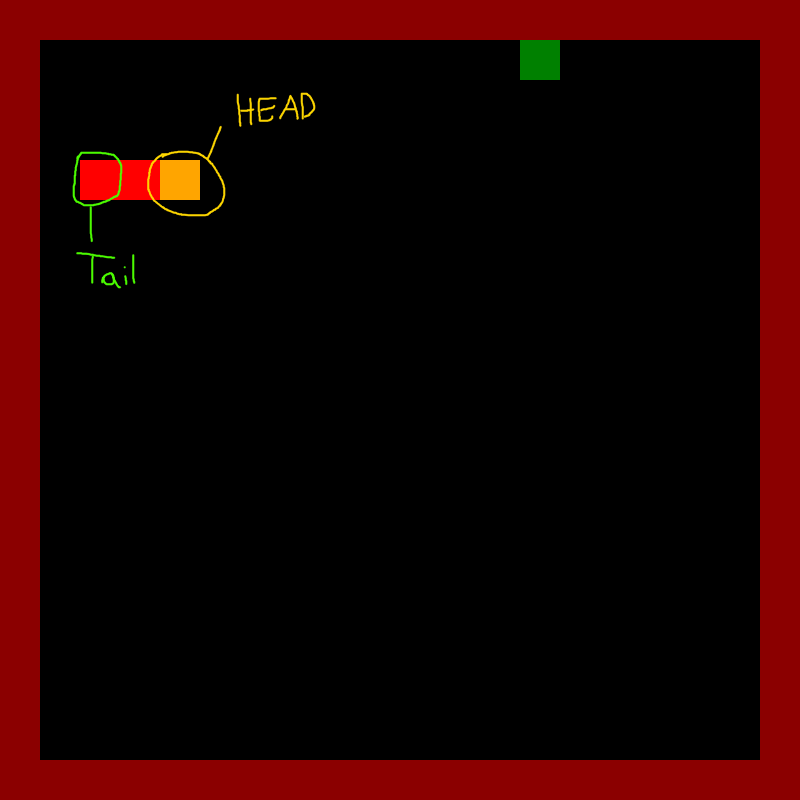
1. Un nouveau carré remplace la tête.
2. Le carré qui représente sa queue disparaît.

Avec ces informations, on peut comprendre que seuls le premier et le dernier carré du serpent changent à chaque frame, tandis que toute la partie du milieu reste statique. Le serpent ne se déplace pas réellement, mais c'est plutôt une illusion de déplacement. À chaque frame, seuls deux carrés changent de couleur, mais chaque carré du serpent a sa propre position sur la grille de jeu qui ne change pas, excepté lors du changement de couleur.

Ainsi, deux informations sont essentielles pour la réalisation du jeu :

1. Les coordonnées de chaque carré qui représente le serpent.
2. La direction de la tête.

La suppression de la queue est automatiquement effectuée à chaque frame et n'est pas influencée directement par la direction. Il est simplement utile de connaître sa position.



Annexe 1 : Déplacement du serpent – GIF

## Interaction pomme/serpent

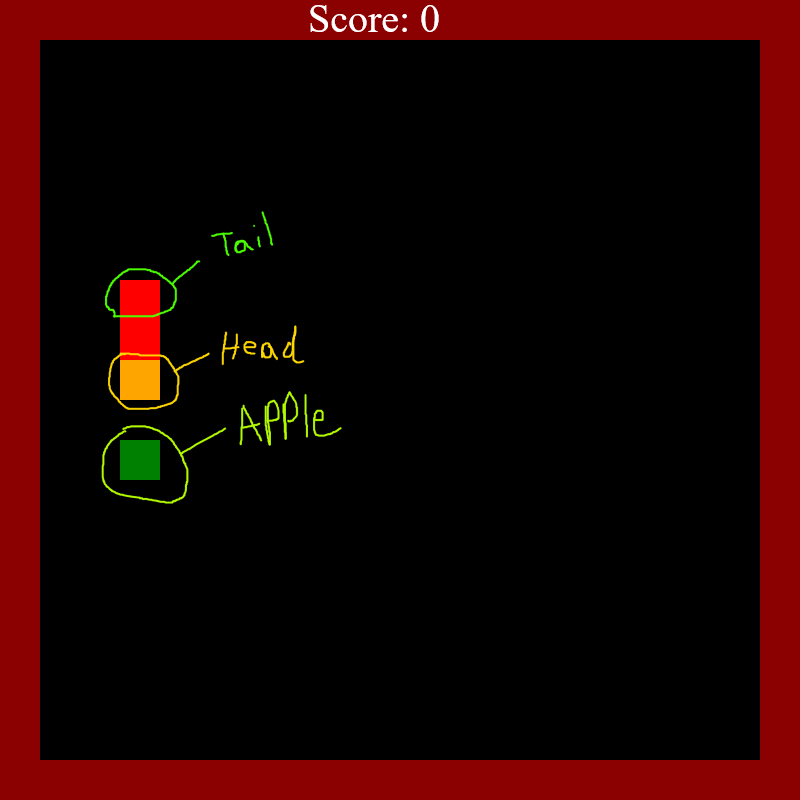
Après que la tête du serpent entre en collision, ce qui signifie que la pomme a été mangée, trois événements se déroulent dans le frame suivant.

Dans le frame où la collision entre la pomme et le serpent a lieu, la pomme change de position sur la grille de jeu et le score augmente de 1. Il n'y a pas de transition, cela se produit immédiatement dès que la tête du serpent couvre la pomme.

Dans le deuxième frame, après qu'il y a eu cette collision, on observe que la tête a avancé d'un carré dans sa direction, mais que sa queue n'a pas disparu. Ainsi, le serpent a grandi d'un carré.

Donc pendant chaque frame de jeu il faut vérifier les positions de la tête du serpent, et celle de la pomme. Quand il y a une collision :

* Le score devrait augmenter de 1
* Un nouveau segment de serpent devrait être ajouter
* La pomme doit changement de position, sans être en conflit avec les positions du serpent



Annexe 2 : Interaction pomme et serpent – GIF

## Collision serpent/bordure

Quand il y a une collision entre le serpent et la bordure de la grille de jeu, il jeu se termine et le joueur a perdu. Le serpent étant un Object dont les positions sont en constant changement à chaque frame de jeu, la bordure quand a elle reste statique.

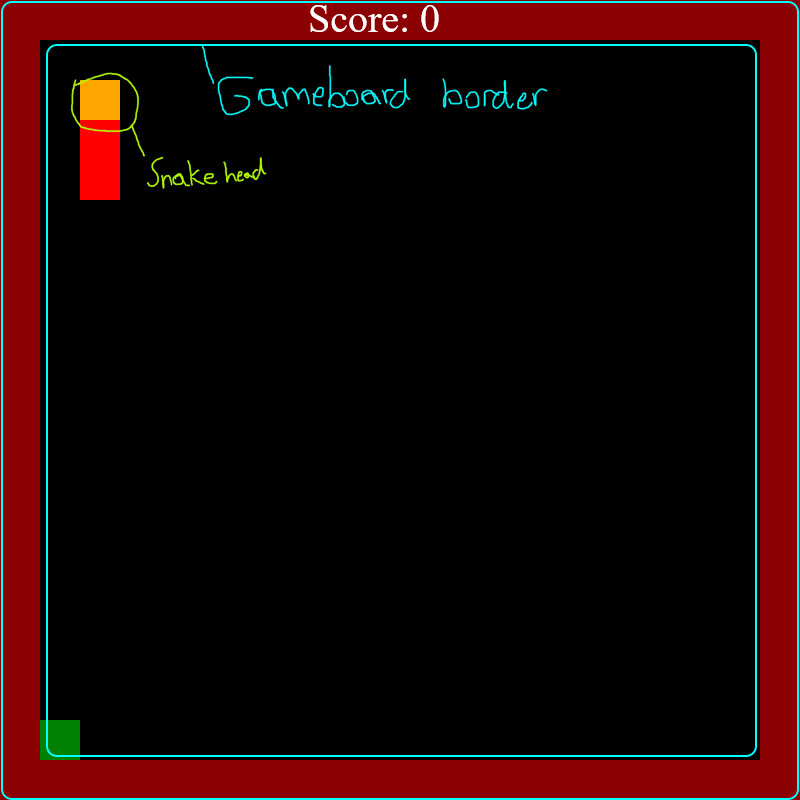
Il n’y a pas forcement de complexité dans les calculs qui doivent être faite. Le zone de jeu, ou le serpent et pomme on le droit d’exister est défini par un largeur et une hauteur. Toute parti du serpent qui ne se trouve pas cette zone est considérer une bordure.

E.G. la zone est 50x50, et commence à (0 ; 0), les quatre coins de la zone seraient :

|  |  |  |
| --- | --- | --- |
| **Nom Coin** | **Position X** | **Position Y** |
| **Haut Droit** | **0** | **0** |
| **Haut Gauche** | **0** | **50** |
| **Bas Droit** | **50** | **0** |
| **Bas Gauche** | **50** | **50** |

Avec c’est cordonné on sait que si le serpent se déplace à droit et a comme position Y 51, il est sorti de la zone et a donc touché une bordure.

C’est aussi avec ces donnes que les calculs pour la position d’une pomme, la position de départ du serpent va être généré



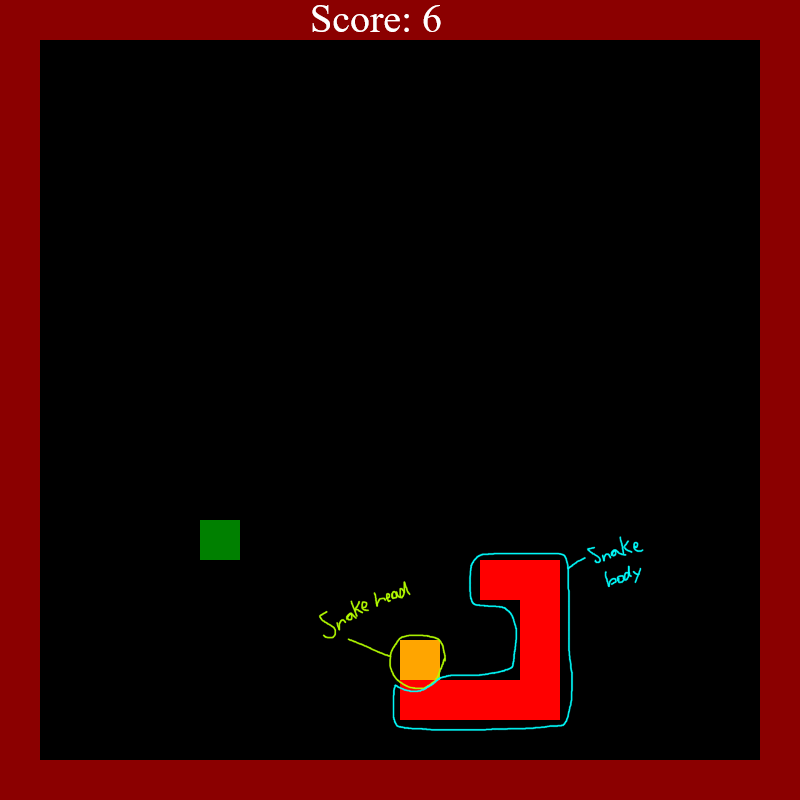
Annexe 3 : Interaction serpent et bordure de jeu - GIF

## Interaction serpent tête/corps

Quand le serpent rentre en contact avec lui-même (tête et corps), ce qui est seulement possible à partir de 5 segments en contant sa tête, le jeu fini.

Parce que c’est obligé de connaitre toutes positions de chaque segment du s’éprend, à chaque frame, c’est seulement nécessaire de faire une simple comparaison des positions.

Si la tête du serpent partage les mêmes positions x et y avec n’importe qu’elle autre segment, c’est qu’il est rentré en lui-même.



Annexe 4 : Interaction serpent tête et corps - GIF

# Implémentation

## Classes

Quatre classes ont été utilisé dans ce jeu.

* **Main.js :** Class contenant toutes fonctionnalité du jeu, la boucle de jeu, ainsi que toutes code en rapport de la visuelle dans le navigateur avec l’utilisation d’un canvas.
* **Apple.js :** Class objet qui repensent la pomme, elle a que comme propriété qu’une position x et y, des intergers.
* **Segment.js :** Class objet qui est utiliser comme bloc de construction pour créer le serpent, elle a que comme propriétés qu’une position x et y, des intégrer.
* **Snake.js :** Class objet qui représente le serpent. Comme propriétés il y a "segments", un tableau pour toutes objets segments qui le compose. Et "direction" qui est la direction dans le quelle il se déplace.

C’est classes sont leurs propres fichiers et vont être utiliser avec les mot clés **export** et **import**, qui serre a exporté et importé une classe a une autre qui se trouve dans un autre fichier.



Figure 1 Class à être exporté



Figure 2 Importer la class depuis l’emplacement ficher

## Constants

Toutes variables qui sont utilisé plusieurs et dois rester statique sont déclarer comme **const**

Avec des noms pertinents pour que la relecture, et compréhension du code soit le plus facile.

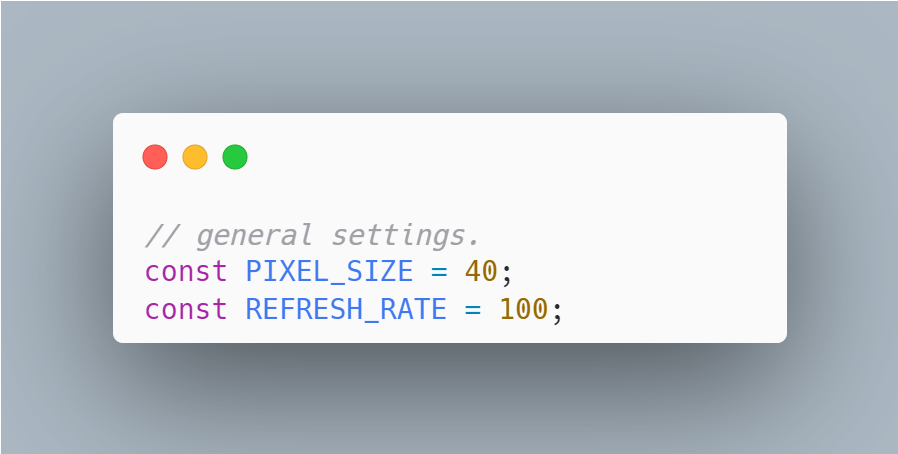


Figure 3 Paramètres général du jeu en const. Taile des pixeles, fréquence de rafraîchissement

## Variables var et let

La différence entre déclarer une variable avec var et let est leur porté et levage.

Un let est seulement accessible dans le porté (scope), les accolade. Et le let est accessible dès qu’elle est déclaré, partout dans le code.

Les variables déclarées avec var sont hissées (hoisted) en haut de leur portée de fonction ou globale pendant la phase de compilation

Donc en cas pratique ça veut dire qu’avec les vars il y a plus de mémoire utiliser pendant que l’application tourne, parce que chaque var qui se trouve dans le code existe et est accessible (même sans avoir de valeur) au lancement du code.

## Fonctions fléchées

Les fonctions fléchées se comporte comme toutes autre fonctions mais est déclaré comme un **const** et avec une syntaxe différente.

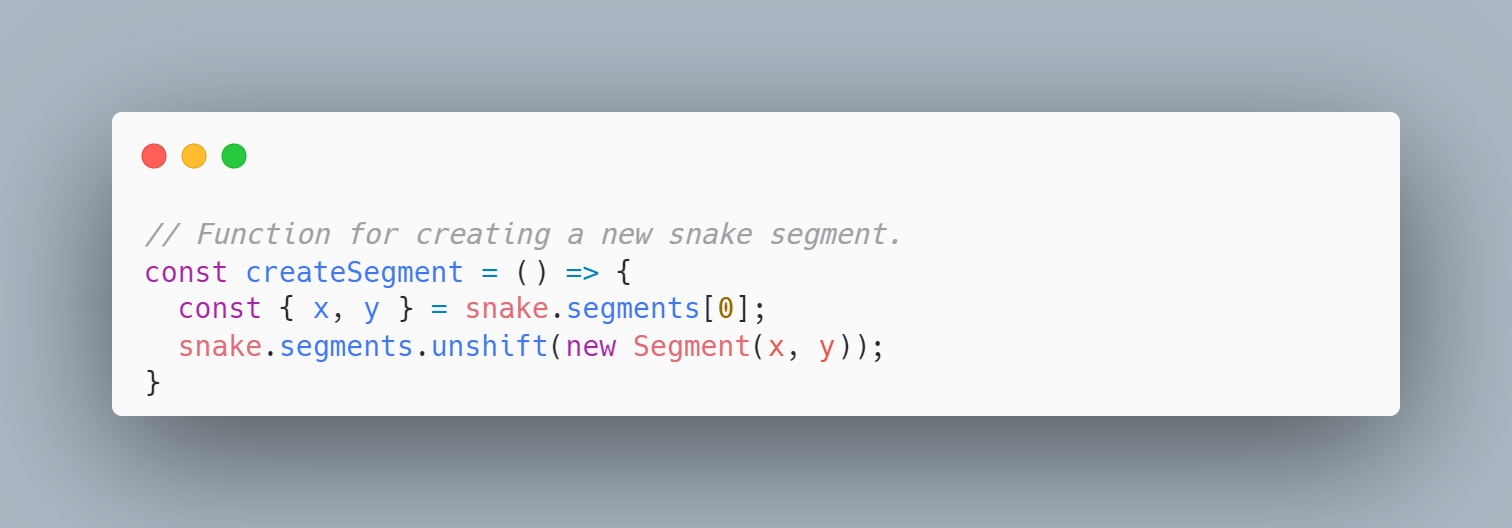


Figure 4 Fonction fléché sans paramètres



Figure 5 Fonction fléché avec paramètres

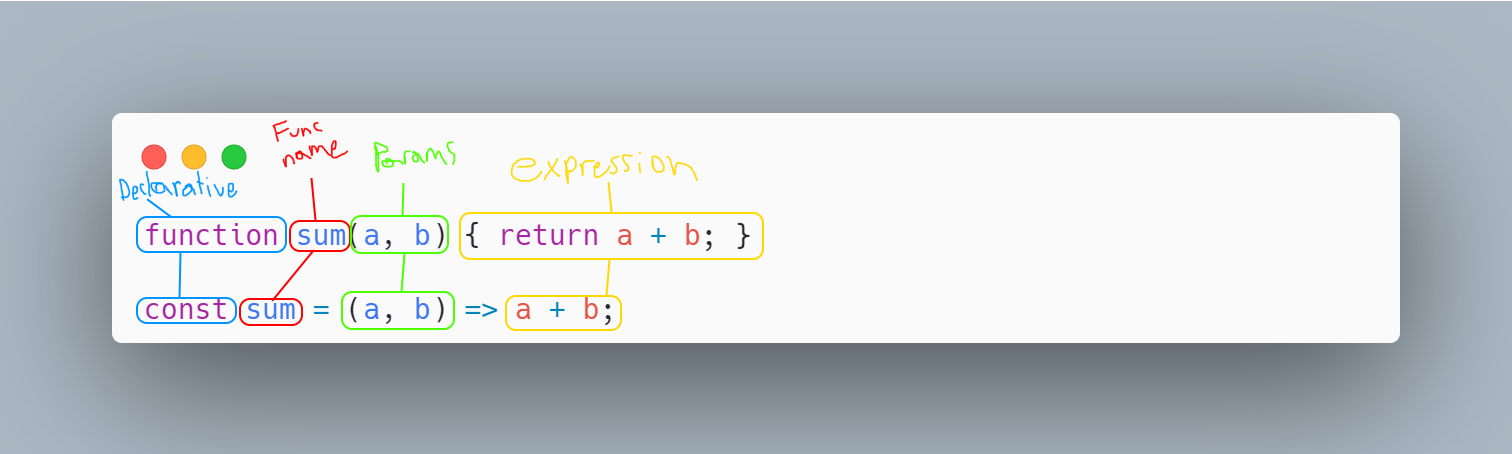


Figure 6 Comparaison entre fonction normale et fléché

# Conclusion

## Tests

|  |  |  |
| --- | --- | --- |
| **Nom** | **Attendu** | **Résultat** |
| **Test de Déplacement avec WASD** | **Le serpent change de position conformément à la touche appuyée. (W:Haut ; A:Gauche ; S:Bas ; D:Droit).** | **OK** |
| **Test de Collision Serpent avec la Bordure** | **La collision avec la bordure déclenche la fin du jeu.** | **OK** |
| **Test de Croissance du Serpent après avoir Mangé une Pomme** | **La taille du serpent est augmentée après la consommation de la pomme.** | **OK** |
| **Test Spawn Aléatoire du Serpent** | **Le serpent spawn dans une position aléatoire à chaque démarrage de jeu.** | **OK** |
| **Test de Génération Aléatoire de la Position de la Pomme** | **Une nouvelle pomme est générée à une position qui n'est pas occupée par le serpent.** | **NOK** |
| **Test de Collision avec Soi-même** | **La collision avec la queue du serpent déclenche la fin du jeu** | **OK** |
| **Test d'Augmentation du Score après avoir Mangé une Pomme** | **Le score est augmenté après la consommation de la pomme.** | **OK** |

## Conclusion

### Amélioration

Durant le projet j’ai beaucoup appris en **JavaScript** qui a donné à des constants changements et optimisations du code. Et toujours maintenant je remarque qu’il y a des choses que j’aurais fait autrement.

Par exemple une réduction du nombre de classes.

Les classes **Apple** et **Segment** sont identique sauf le nom, et où ils sont utilisés dans le main. Il est possible d’en faire qu’un class pour les deux. Un class "bloc" avec que comme propriétés "x" et "y".

La class **Snake** peut aussi être enlever. Le tableau "segments" et sa "direction" peut simplement être remplacer par dans variables qui se trouve dans la main.

Une fonctionnalité sympa que j’aurais pu ajouter pour justifier c’est classes, est de généré une ton couleur de vert aléatoire pour chaque pomme ou de rouge pour chaque segment du serpent.

# Annexe

1. Déplacement du serpent – GIF
2. Interaction pomme et serpent – GIF
3. Interaction serpent et bordure de jeu - GIF
4. Interaction serpent tête et corps - GIF