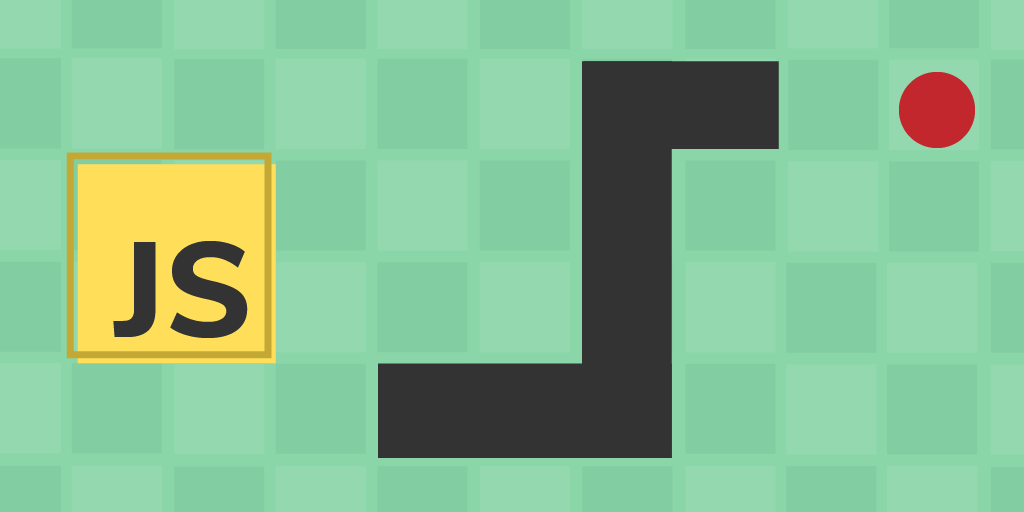
Projet Bulles



**Rapport de Projet**

Zeqiri Amir – CID2A

ETML, Sébeillon – N512A

40 Périodes

Maître : M. Chenaux

Table des matières

[1 Description du projet dans son ensemble 3](#_Toc155710237)

[1.1 Titre 3](#_Toc155710238)

[1.2 Sujet 3](#_Toc155710239)

[2 Support de cours 3](#_Toc155710240)

[2.1 Importation de la machine virtuelle 3](#_Toc155710241)

[2.2 Installation des prérequis 3](#_Toc155710242)

[2.3 Codage du Snake en JS 4](#_Toc155710243)

[2.3.1 Utiliser const et let mais jamais var 6](#_Toc155710244)

[2.3.2 Créer des classes (Snake / Apple) 6](#_Toc155710245)

[2.3.3 Utiliser les fonctions fléchées 8](#_Toc155710246)

[2.3.4 Les modules : import / export 10](#_Toc155710247)

[2.3.5 Utiliser l’opérateur rest pour décomposer un tableau 10](#_Toc155710248)

[2.3.6 Appréhender certains concepts de programmation 11](#_Toc155710249)

[3 Sources & Aides 11](#_Toc155710250)

[4 Conclusion 11](#_Toc155710251)

# Description du projet dans son ensemble

## Titre

Snake en JS

## Sujet

Faire un réplica du jeu célèbre jeu Snake

# Support de cours

## Importation de la machine virtuelle

La machine virtuelle sur laquelle nous allons faire notre projet nous a été donné par le maître M. Chenaux. Nous avons dû l’importer sur Oracle VM VirtualBox afin de pouvoir l’ouvrir comme il se doit. La machine était toute configurée et Visual Studio Code était installé au préalable. Personnellement, j’ai installé l’extension Live Preview sur VS Code. C’est une extension Microsoft qui me permet d’avoir un rendu de mon code et de voir les changements de mon code en temps réel sur mon navigateur (localhost). La VM fonctionne sur Linux Ubuntu, c’est un changement pour moi car la grande majorité de mon temps passé sur un ordinateur est sur Windows. Malgré ça, c’est un OS où j’ai déjà pu travailler dessus et qui n’est pas compliqué à utiliser.

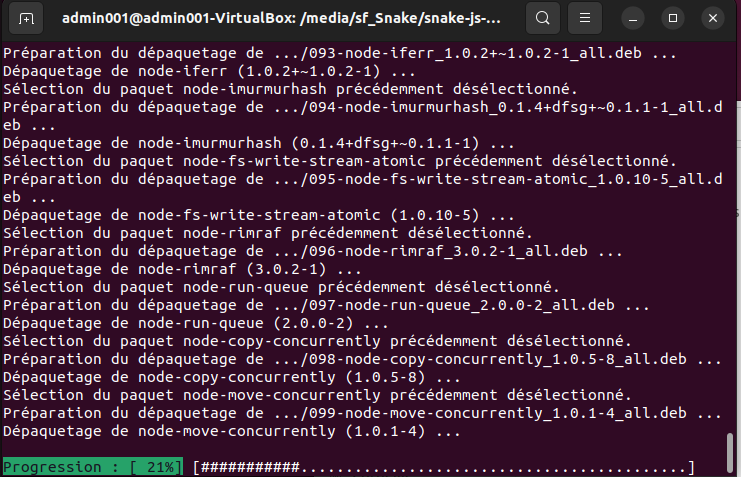
## Installation des prérequis

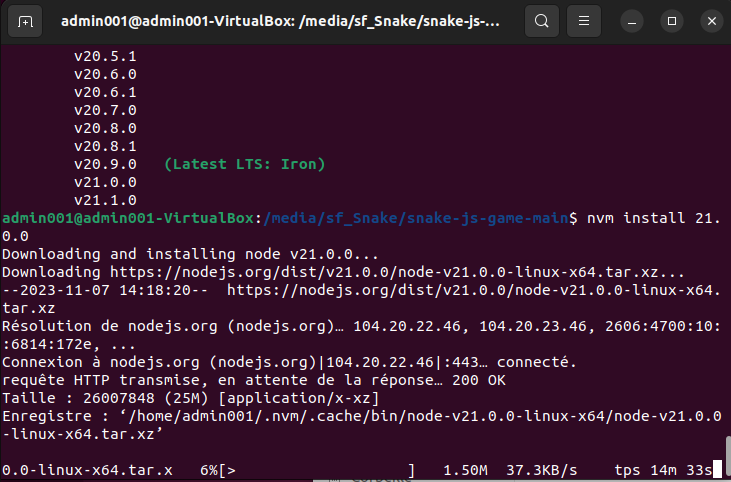
Avant de commencer quoi que ce soit, j’ai dû installer les dépendances JS (Node JS) pour le bon fonctionnement du projet.

Voici comment j’ai procédé :

1. Ouvrir le terminal dans le dossier où se trouve le snake
2. Ecrire "sudo apt install npm"
3. Ecrire "sudo apt update"
4. Ecrire "sudo wget –q0-"
5. Ecrire "source ~/.profile"
6. Ecrire "nvm –v"
7. Ecrire "nvm ls-remote"
8. Ecrire "node –v"

Et voici quelques photos prises durant la procédure :





## Codage du Snake en JS

Il est maintenant grand temps de commencer à coder mon Snake en JavaScript. Nous disposons d’une base pour commencer le projet, en effet, nous avons le fichier index en HTML, le fichier source en CSS, le fichier main en JavaScript et les différents packages que je ne devrai normalement pas toucher. La base qui nous a été donné ne fonctionnais pas parfaitement, le CSS n’était pas lié à l’HTML. J’ai simplement ajouté cette ligne dans le "head" de ma page index :

<link rel="stylesheet" href="css/style.css">.

Une fois ceci ajouté, j’avais ce rendu :



On peut voir que le canvas est là et que le CSS est bien relié. Je peux maintenant, de moi-même, coder le snake.

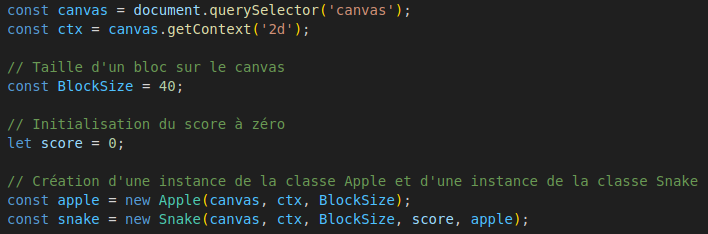
Je commence donc par mettre les bords du canvas en vert, comme la couleur du serpent et je fais en sorte à ce qu’il n’y ait pas de curseur sur le canvas afin que, quand je joue, cela ne me dérange pas.

Voici le rendu :



Ensuite, je vais me consacrer au sérieux, c’est-à-dire afficher un bloc vert qui sera la tête du serpent, et à chaque fois que le serpent mangera une pomme un bloc s’ajoutera à lui qui suivra sa trace.

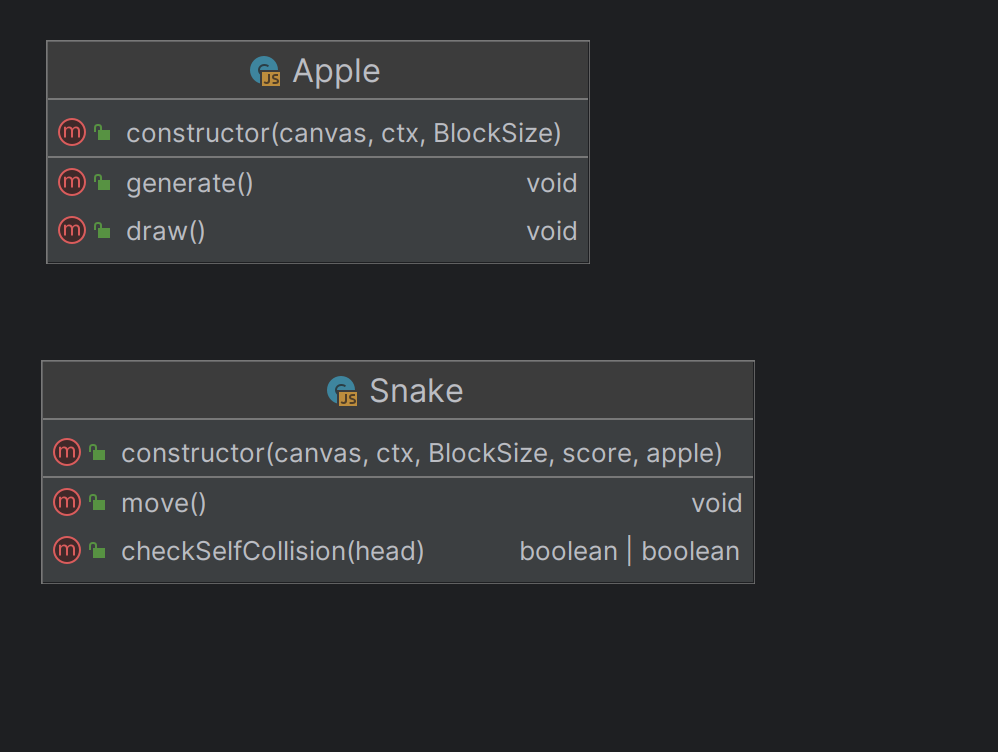
### Utiliser const et let mais jamais var



Ce printscreen montre les variables que j’utilise dans le main. On peut voir qu’aucune variable var n’est déclarée. 5 sont déclarés en const et une seule en let. Je n’utilise pas de variables var car, non seulement, c’est demandé dans le cahier des charges, mais aussi, car celles-ci peuvent commettre des erreurs invisibles.

### Créer des classes (Snake / Apple)

Voici le diagramme de classes :



Les classes "Apple" et "Snake" dans ce code sont utilisées pour créer des objets représentant la pomme (représentée par la classe apple) et le serpent (représenté par la classe snake) dans le jeu Snake codé en JavaScript.

**Apple**

- La classe "Apple" a un constructeur qui prend en paramètre le canvas, le contexte 2D (ctx), et la taille des blocs (BlockSize). Elle est responsable de la génération aléatoire de la position de la pomme sur le canvas.

- La méthode "generate" génère une nouvelle position aléatoire pour la pomme en fonction de la taille des blocs du canvas.

- La méthode "draw" dessine la pomme sur le canvas en utilisant un rectangle rouge à la position définie par les coordonnées "(this.x, this.y)".

**Snake**

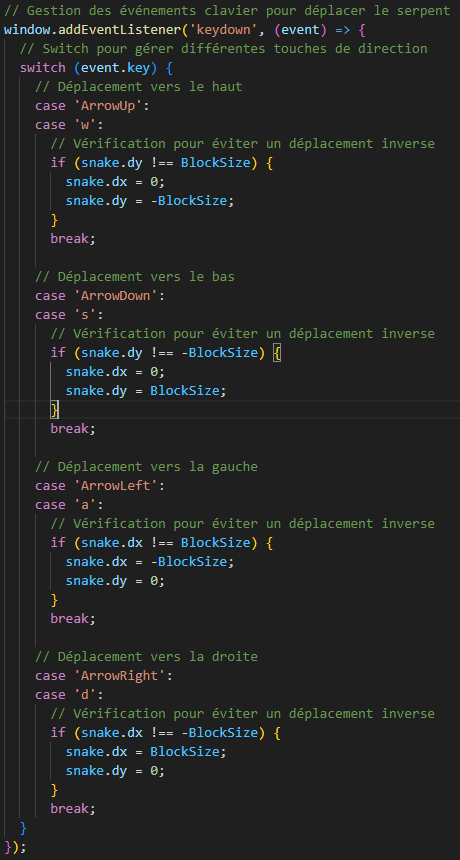
- La classe "Snake" représente le serpent dans le jeu. Son constructeur prend en paramètre le canvas, le contexte 2D (ctx), la taille des blocs (BlockSize), le score initial, et une instance de la classe "Apple".

- La méthode "move" gère le déplacement du serpent. Elle vérifie les collisions avec les bords du canvas, la collision avec la pomme, et la collision avec le corps du serpent. Elle met à jour la position du serpent, le score, et redessine le canvas à chaque mouvement.

- La méthode "checkSelfCollision(head)" vérifie si la tête du serpent entre en collision avec son propre corps.

- La classe "Snake" utilise le contexte 2D pour dessiner le serpent et la pomme sur le canvas. Le corps du serpent est représenté par des rectangles verts, la pomme par un rectangle rouge, et le score est affiché en haut à gauche du canvas.

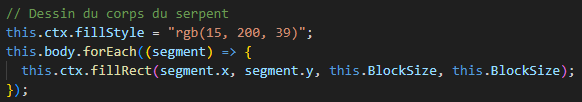
### Utiliser les fonctions fléchées

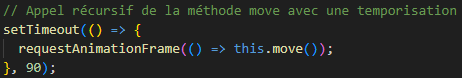


Sur cette capture de mon code, on voit la première fonction fléchée.

La première fonction fléchée fonctionne comme un gestionnaire d'événements pour écouter les touches du clavier. Elle est liée à l'événement 'keydown' sur l'objet window. Cette fonction permet de détecter quelle touche du clavier est enfoncée et ajuste ensuite la direction du serpent en conséquence. Elle gère les déplacements vers le haut, le bas, la gauche et la droite en fonction des touches fléchées et des touches 'W', 'S', 'A' et 'D'.

La deuxième fonction fléchée sert de callback dans la méthode requestAnimationFrame. Elle est appelée à chaque frame d'animation et elle déclenche la méthode move de l'instance de la classe Snake. Ça établit une boucle principale qui actualise la position du serpent à chaque frame et ça assure un mouvement fluide.

La troisième fonction fléchée, utilisée comme callback dans setTimeout, enveloppe un appel à requestAnimationFrame. Cette dernière utilise une autre fonction fléchée comme callback pour appeler récursivement la méthode move. Tout cela crée une sorte de boucle non synchronisée qui permet au serpent de se déplacer en continu, avec un petit délai entre chaque étape, ce qui donne une animation fluide.

Cette fonction fléchée est utilisée comme callback pour la méthode forEach appliquée à l'array this.body. Elle est utilisée pour dessiner chaque segment du corps du serpent sur le canvas en utilisant les coordonnées fournies par chaque élément de l'array.

J’ai utilisé des fonctions fléchées car, non seulement, c’est demandé dans le cahier des charges, mais aussi, car celles-ci réduisent la quantité de code nécessaire pour définir une fonction

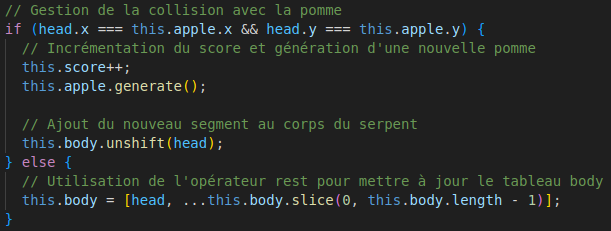
### Les modules : import / export

Dans la première image, j’exporte la classe "Snake"

Dans la deuxième image, j’exporte la classe "Apple"

Et dans la dernière image, j’importe les différentes classes dans le main

### Utiliser l’opérateur rest pour décomposer un tableau



Voici le bout de mon code où j’utilise un opérateur rest.

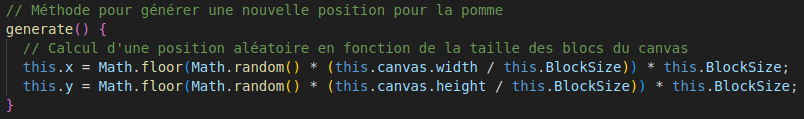
Dans mon cas, l'opérateur rest est utilisé pour mettre à jour le tableau body du serpent en conservant la tête actuelle et en éliminant le dernier élément du reste du corps. Voici comment cela fonctionne :

Dans la ligne de code où est l’opérateur rest, head représente la nouvelle position de la tête du serpent. L'opérateur rest (...) est utilisé pour prendre tous les éléments restants du tableau this.body après avoir retiré le dernier élément. Ensuite, ces éléments sont regroupés avec la tête du serpent dans un nouveau tableau, qui remplace ensuite le tableau existant this.body.

### Appréhender certains concepts de programmation

**Génération aléatoire**

Voici la méthode qui fais l’apparition aléatoire des pommes.



Cette méthode génère une nouvelle position aléatoire pour une pomme dans le jeu en utilisant des coordonnées entières qui sont des multiples de la taille des blocs du canvas. Cela garantit que la pomme apparaît à un emplacement où le serpent peut la capturer.

# Sources & Aides

<https://www.w3schools.com/js/default.asp>

Le site w3schools m’a été fortement utile, c’est le site principal que j’ai utilisé pour apprendre le JavaScript et c’est celui que j’ai utilisé pour la grande partie de mon snake.

<https://chat.openai.com/>

ChatGPT m’a été utile dans la compréhension de code. Je m’explique, quand je ne comprenais pas une partie de code que je trouvais sur internet, je demandais à ChatGPT et il me donnait la réponse dont j’avais besoin. Ce qui m’a également permis de mieux expliquer ce que je faisais dans le support de cours. En plus de tous cela, il m’a été utile dans les commentaires, quand je ne trouvais pas les mots pour commenter un bout de code, je lui demandais et il m’aidait.

<https://codepen.io/gsoldateli/pen/yZLbMW>

Ce site proposant son propre snake m’a également été utile lors de la réalisation du projet. Je l’ai regardé au tout début du snake, j’essayais de comprendre le fonctionnement du jeu. Il ne m’a plus servi depuis.

# Conclusion

J’ai appris beaucoup de chose grâce à ce projet. En effet, je n’avais jamais toucher au JavaScript auparavant. Cela m’a permis de faire connaissance avec ce langage et d’avoir la satisfaction de terminer, avec succès, un snake avec un langage pareil. De plus, aucun rapport n’étant demandé, cela m’a permis de me consacrer plus au développement du jeu qu’autre chose.