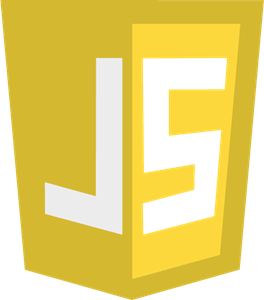
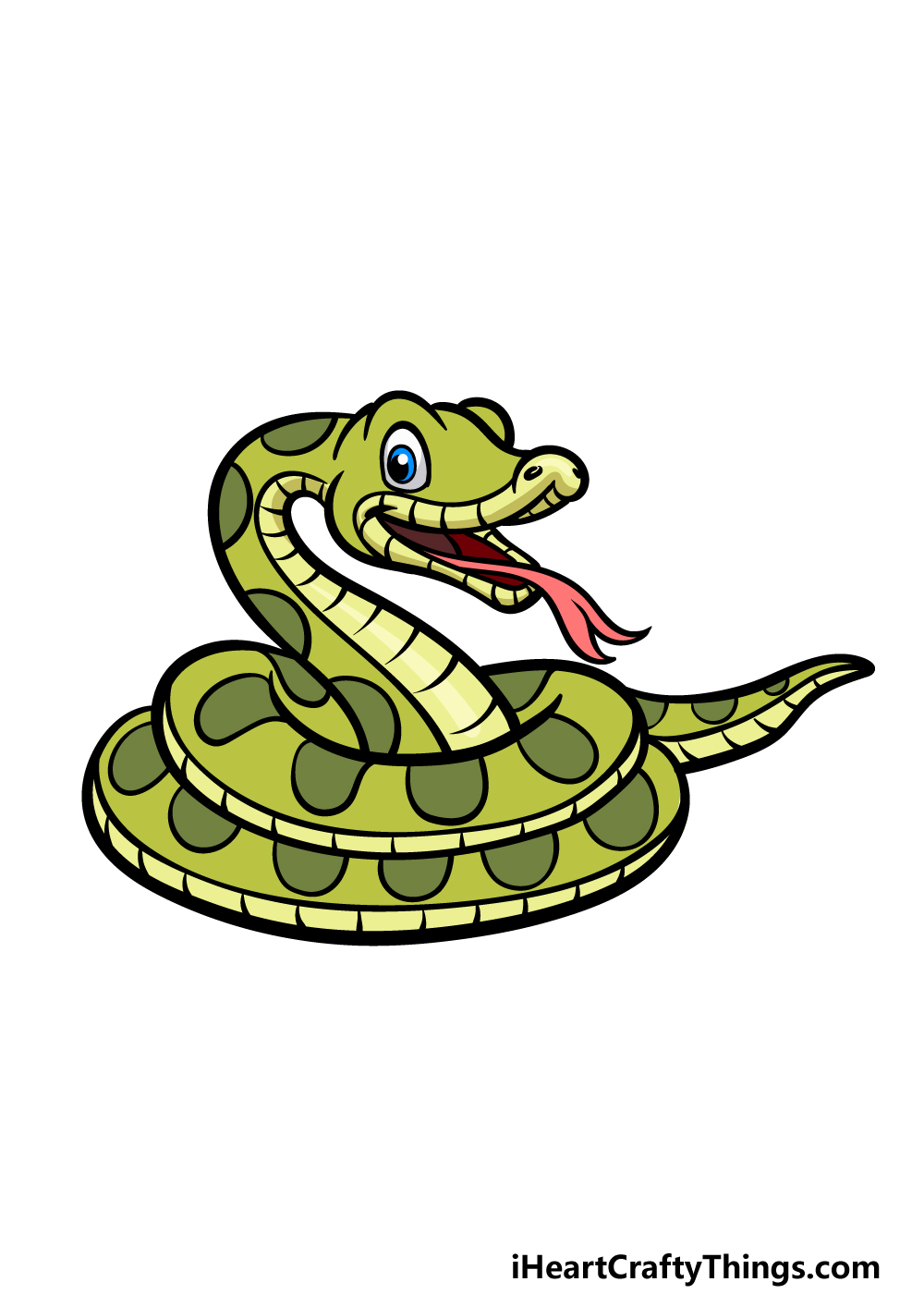
Support de cours

Snake - P\_Bulles

Cyril Napoleone





# Introduction

Avant de commencer le code concrètement j’ai dû faire plusieurs étapes préalables que voici  (L’entièreté de ce support de cours se déroule dans l’ordre chronologique de mon travail):

* Installation VM linux
* Mise en place Node.js
* Arrangements pour le bon fonctionnement du programme (live server)

## VM linux

Le professeur nous a fourni une machine virtuelle déjà prête (quelques modifications nécessaires pour que l’import fonctionne) puis j’ai installé le toute sur mon SSD pour pouvoir l’utiliser n’importe où.

La VM a déjà Visual studio code d’installé (PowerShell aussi) et est entièrement configurée.

## Installation Node.js

J’ai installé Node.js sur linux via le terminal, la classe a eu un souci lié au fait que la marche à suivre datait d’autres versions. Nous avons utilisé cette référence pour l’installer avec succès : <https://blog.crowncloud.net/post/installing-node-js-on-ubuntu-23-10/>

## Arrangements additionnel

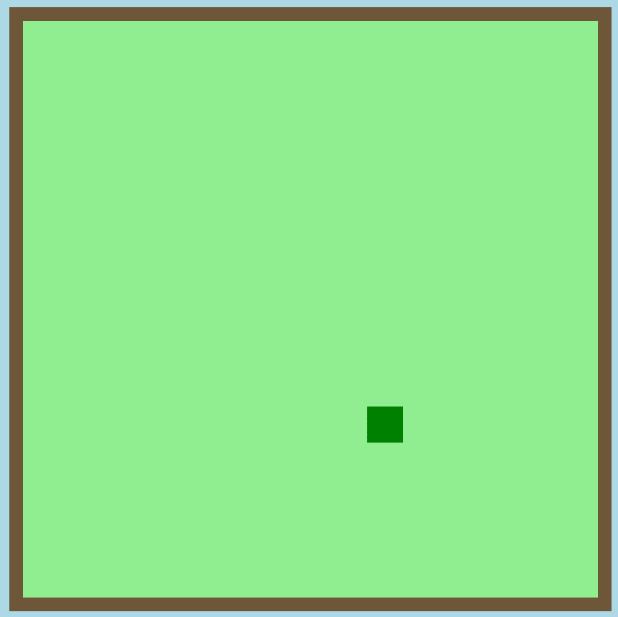
Il y’a eu un autre souci qui a rendu le commencement du concret plus long que prévu, le fichier html ne prenait pas en compte le fichier js, en utilisant l’extension « live server » qui permet d’héberger localement et temporairement son projet. Mais nous avons découvert que plusieurs résolutions du problème étaient possibles (dossiers partagés, vite, etc..).

# Familiarisation avec JavaScript

## Familiarisation seul

Étant donné que je n’ai jamais utilisé JavaScript avant ce projet et que nous avons pas de théorie en parallèle à ce cours, j’ai décidé de me familiariser seul avec le langage avant de me renseigner en ligne. J’ai donc touché au réglages du code que nous avons par défaut, j’ai compris comment il marchait et à quoi il servait puis comment afficher des choses, les variables, un peu de style et quelques fonctionnalités.

Je suis très vite arrivé à un résultat car j’ai réussi à faire un carré qui se déplace seul et qui s’il atteint la limite du terrain, apparait de l’autre côté de celui-ci (et cela dans tous les sens). Sans avoir encore utilisé internet.



## Documentation internet

Le point à partir du quel j’ai senti nécessaire de commencer à me documenter sur des forums, sites, etc… est quand j’ai voulu pouvoir contrôler mon carré (qui se déplaçait déjà) avec mes flèches plutôt que de devoir changer le code manuellement pour voir des changements de directions.

<https://stackoverflow.com/questions/5597060/detecting-arrow-key-presses-in-javascript>

J’ai donc implémenter ce que j’ai appris (lien ci-dessus) et maintenant je peux me déplacer comme je veux sur mon terrain avec les limites fonctionnelles (quand je touche une flèche, elle change la variable direction qui elle est utilisé dans le déplacement. Par ex : si direction = 1 alors le carré va à droite jusqu’à changement de la variable direction)

Maintenant que j’ai déjà pris mes marques avec ce langage je peux commencer à faire les parties + complexe du code (système serpent, points, pommes, etc…) et puis l’utilisation des classes.

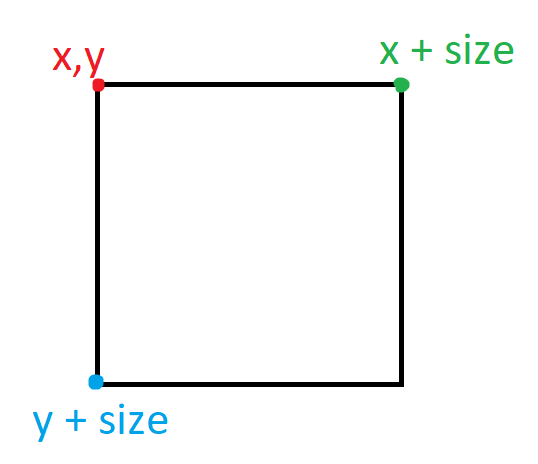
# Code

Avant d’ajouter des fonctionnalités j’ai bien pris en considération le CDC. J’ai donc, avant tout, fait en sorte de comprendre (via <https://www.w3schools.com/js/js_classes.asp> ) et utiliser les classes pour pouvoir mettre mon serpent dans une et pour que je me familiarise déjà avec ce concept avant qu’il soit trop tard (car en C# POO j’ai voulu tout convertir en orienté objets après avoir fini le code sur une page, ce qui a été une erreur).

J’ai réussi sans trop de soucis puis me suis penché sur le fonctionnement de l’aléatoire en JavaScript car il est un peu différent du C#.

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

Grâce à cette page, j’ai pu comprendre son fonctionnement et l’utiliser pour l’apparition de ma pomme, qui d’ailleurs m’a posé un souci après ça. Le problème était que je voulais lui attribuer une nouvelle position quand mon carré la touche (manger la pomme) sauf que mes détections de collisions ne marchaient pas. J’ai donc fais un schéma rapide sur Paint pour mieux visualiser ce qui devait être pris en compte.



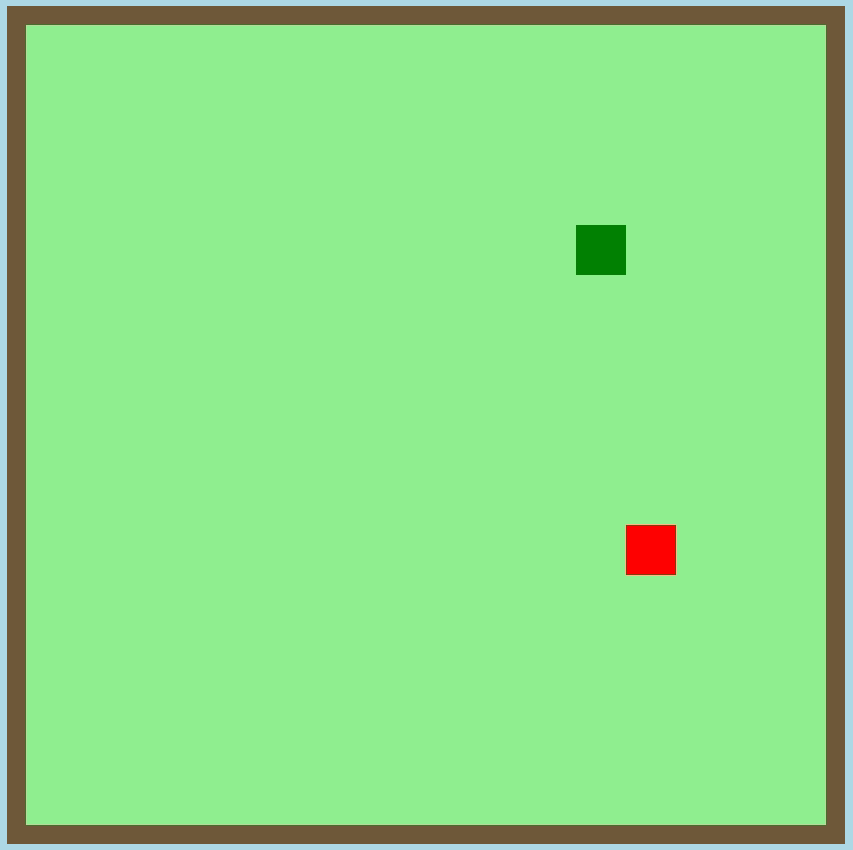
Car effectivement, ma position x et y représente seulement le coin en haut à gauche du carré, donc tout le reste et une combinaison de variables (position et taille du carré). Grâce à ça j’ai pu créer un if comportant 4 conditions :

if (snake.x < apple.x + apple.size && snake.x + snake.size > apple.x && snake.y < apple.y + apple.size && snake.y + snake.size > apple.y)

Maintenant, dès que mon carré touche la pomme, cela est détecté, elle réapparait autre part aléatoirement et le score est incrémenté.

Après une discussion avec monsieur Cheneaux sur le système actuel de mon jeu, j’ai décidé de changer cette aspect « mouvements libres » de mon Snake, car oui je pouvais bouger sans apparences de « quadrillage », c’est-à-dire le serpent et la pomme ne s’emboitaient pas parfaitement par exemple. Et au lieu de régler cet aspect plus tard, j’ai préféré faire ça tout de suite.

J’ai simplement trouvé un équivalent du Thread.Sleep(150) en JavaScript via [cette page](https://www.sitepoint.com/delay-sleep-pause-wait/#:~:text=intrusive%20sleep%20method%3A-,function%20sleep(ms)%20%7B%20return%20new%20Promise(resolve%20%3D,%2C%20then%20log%20%E2%80%9CWorld!%E2%80%9D), que j’ai ajouté à mon code juste avant chaque mouvement (que j’ai incrémenté pour faire des « bons » plus gros ). Puis, pour que la pomme apparaisse toujours à un endroit qui concorde aux déplacements de 50 par 50 du serpent, j’ai repris un système (avec quelques adaptations) que j’avais fait en collaboration avec Yohan pour son code à lui mais que je n’avais pas implémenté au miens. Il consiste à utiliser un tableau de valeurs (dans mon cas, les multiples de 50, de 0 à 750 (limites terrain)) et piocher aléatoirement une case du tableau comme position x ou y. J’ai adapté ce principe pour en faire des fonctions qui retournent donc une position aléatoire quand je veux, ce qui a rendu cette partie plutôt simple.



Le carré rouge (la pomme) réapparait à une nouvelle position aléatoire dès qu’elle est touchée par le carré vert (le serpent). Le score est aussi incrémenté de 1 à chaque pomme mangée.

Après ça je me suis attaqué au système d’agrandissement du serpent, A REMPLIR

! changement stratégie, limite libre -> limite dur, système de « cardillage »,  !

(meme si ça aurait été plus simple de faire avec des cases, j’ai préféré faire un snake plus libre parce que plus moderne et originiale et (préfération personnelle))