Youcef ANNAB
William-Arno CLEMENT

Projet Tutoré

Le jeu du serpent

Langage C

PRELUDE

Dans le cadre de nos études en IUT Informatique, nous avons eu l'opportunité de réaliser un jeu du serpent, dit *Snake*, à l'aide du langage C et de la librairie graphique mise a disposition par notre établissement.
Basée sur la librairies X11, cette dernière permet de manipuler une interface graphique à partir d'appels à partir de fonctions préconçues.
Notre objectif pour la réalisation de notre travail est double:

- Continuer notre apprentissage pour une bonne maitrise du langage C
- Apprendre à travailler en équipe sur un projet informatique

Dans ce document, nous allons vous expliquer notre démarche de développement et vous faire découvrir l'évolution de notre travail.

Nous vous souhaitons une agréable lecture.

SOMMAIRE

LE PROJET ET SES CONTRAINTES

Le jeu du serpent est un jeux vidéo dans lequel je serpent doit manger des pastilles pour grandir, tout en essayant de ne jamais se mordre la queue. Le cœur du gameplay consiste donc à manger un maximum de pastilles en un minimum de temps.
Mais un élément perturbateur vient se confronter au serpent: il grandit à chaque pastille avalée.
Dans un premier temps, nous avons dû procéder à une lecture du cahier des charge dont voici ci-dessous les contraintes imposées pour la réalisation de notre projet.

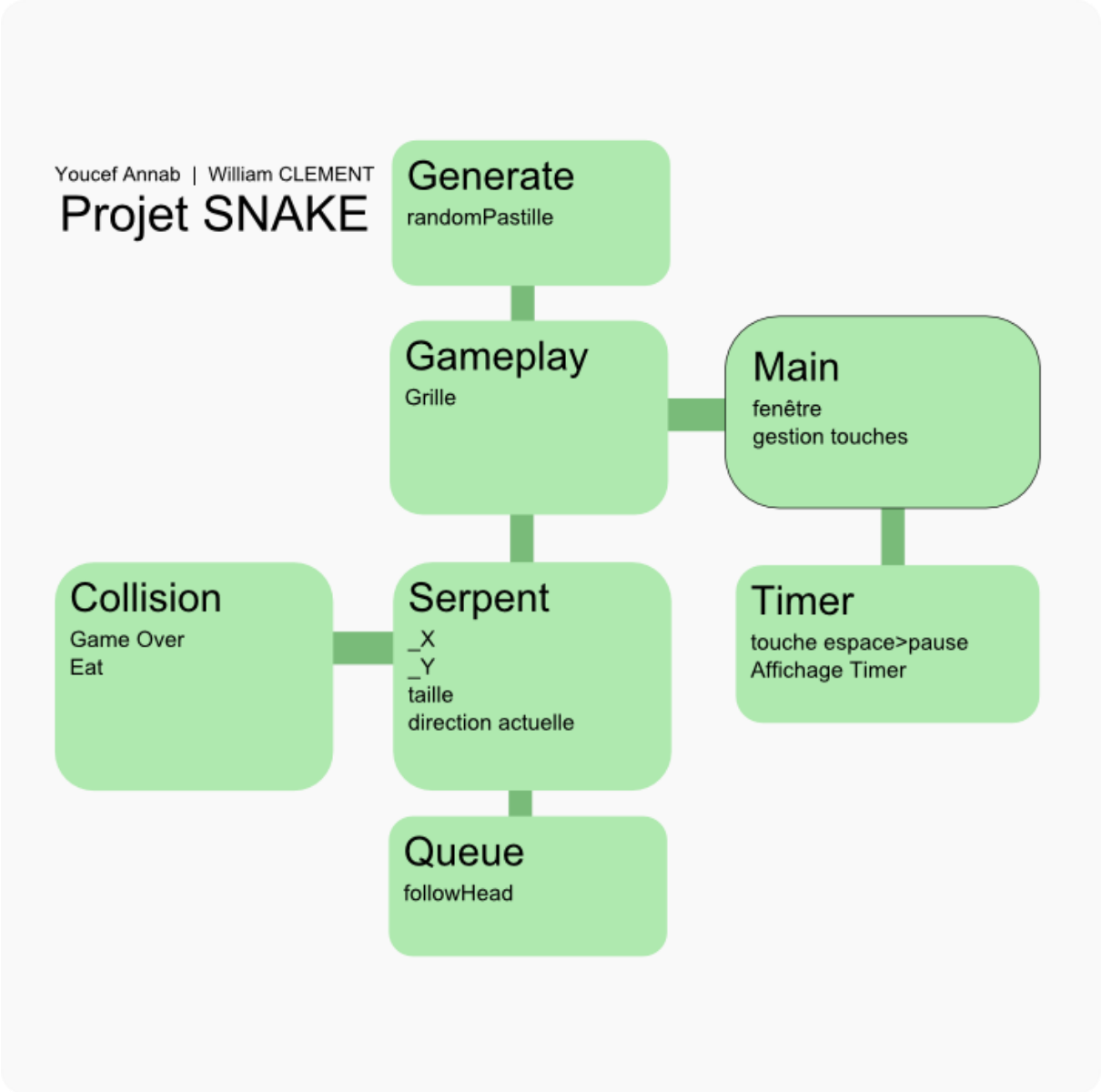
- Le terrain de jeu doit prendre la forme d'une grille de 40 lignes sur 60 colonnes.
- Le serpent - incarné par le joueur- doit parcourir le terrain en essayant de « manger » le plus de pastilles qui sont posées aléatoirement au début de la partie, elles sont aux nombres de 5. Lorsqu'une pastille est mangée, une nouvelle apparaît aléatoirement également.
- Le serpent qui possède 10 segments de base, prend 2 segments en plus à chaque fois qu'il mange une pastille.
- Le joueur peut changer de direction grâce aux touches « haut », « bas », « gauche » et « droite » du clavier.
- Le joueur doit pouvoir voir le temps écoulé depuis le début de la partie, ainsi que son score.

En plus de ces contrainte s'ajoute une notion supplémentaire que nous
avons choisi: l'accélération de la vitesse du serpent au fur et à mesure de son aventure.

PREMIERE EBAUCHE ET MODELISATION DU SYSTEME

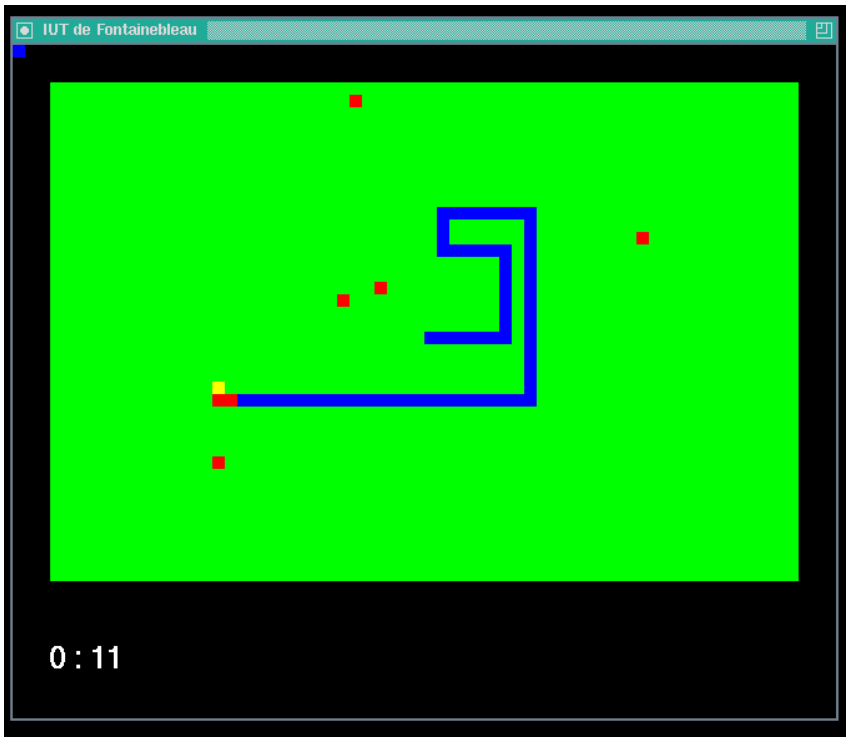
Suite à l'évaluation du cahier des charge, nous avons réalisé une première modélisation de la structure de notre jeu.

Comme vous pouvez le constater à la lecture de cette première ébauche, nous sommes partis sur une successions de notions interconnectées les unes avec les autres. L'objectif était à ce moment de déterminer les différentes parties à mettre en place dans le code du Snake.
La première étape a donc été de coder le fichier Main -le centre opérationnel de notre programme- duquel seront gérés l'affichage de la fenêtre et la gestion des touches.

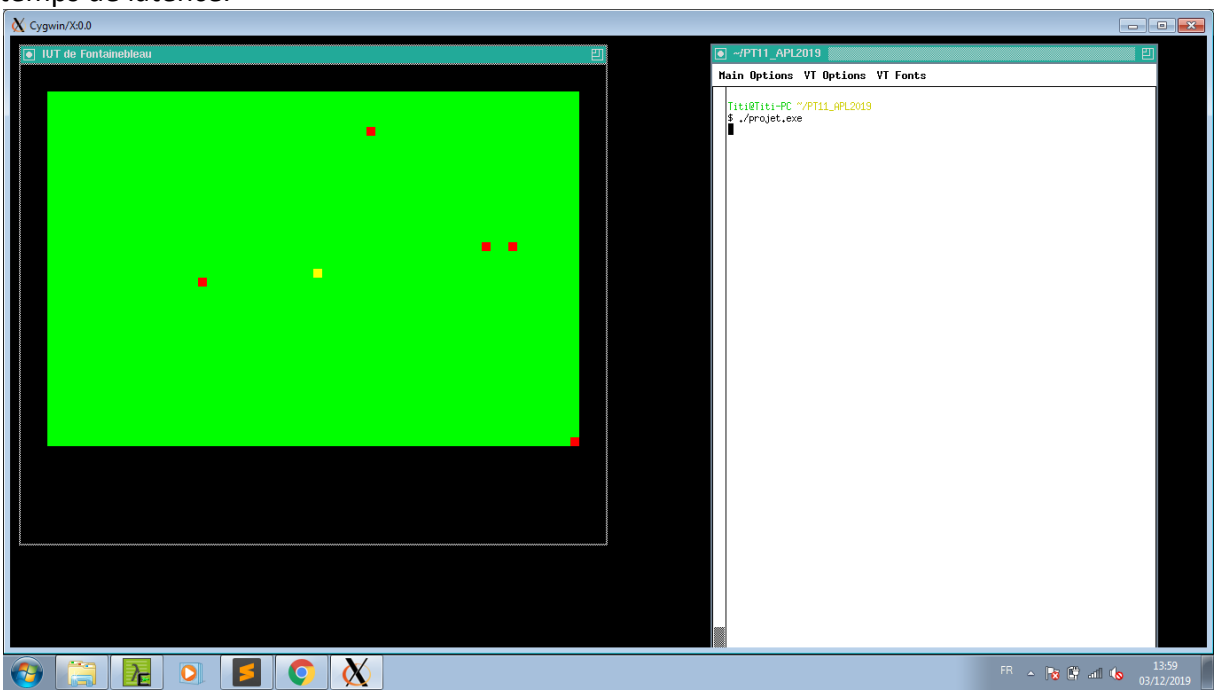


LA TÊTE DU SERPENT

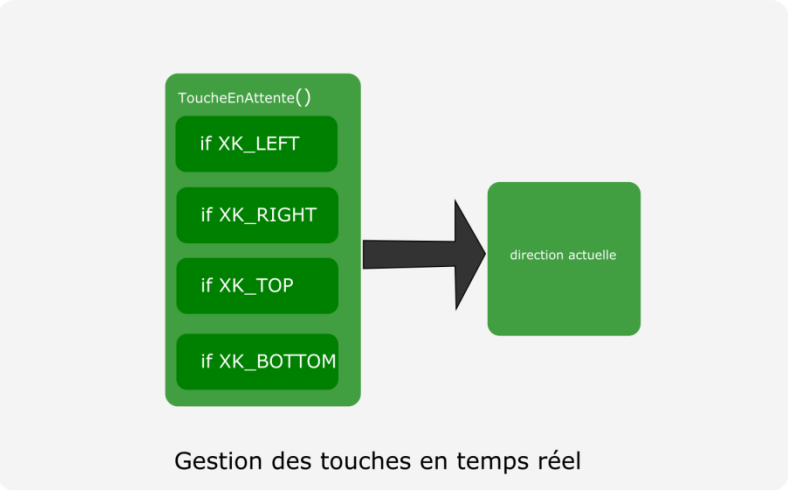
Dans un premier temps, notre objectif est d'afficher la tête du serpent, représentée par un carré jaune. Celui-ci se déplace de case en case, à chaque fois que le joueur appuie sur une touche directionnelle.



A l'aide de la fonction native de la librairie graphique `ToucheEnAttente()` nous sommes capable de réceptionner une information à l'instant ou la touche est pressée : ceci réduit le temps de latence.



Néanmoins, pour faire en sorte que le serpent se s'arrête plus, et qu'il avance constamment, nous décidons d'ancrer sous forme de variable la direction actuelle du serpent. Autrement dit, la dernière touche pressée dit à la tête du serment si elle doit se déplacer à gauche, à droite, en haut ou en bas.



Gestion des touches en temps réel

LA QUEUE DU SERPENT

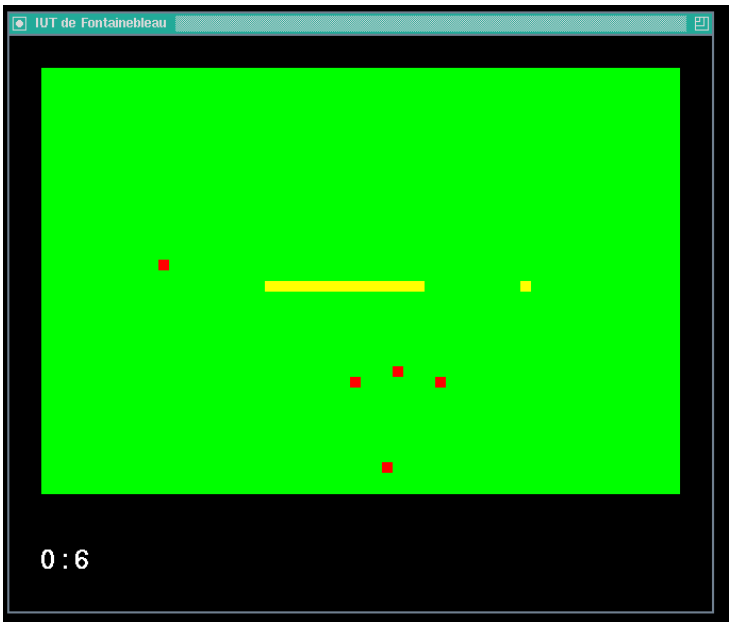
Suite à la réussite de cette étape, nous voulions continuer sur notre lancé en ajoutant la queue au serpent, cependant, après plusieurs tentatives infructueuses, un problème de taille se présentait devant nous : devions nous utiliser un tableau de structure ou un tableau multidimensionnel ? Nous avons décidé de reporter cette étape de quelques jours pour nous laisser le temps de la réflexion.

Pour continuer, nous avons réalisé le *timer* grâce à la fonction `Microsecondes()`, elle nous permet de calculer le temps que le joueur a mis dès lors qu'il a lancé la partie, jusqu'au moment où la partie s'arrête.

A la base, nous étions partis sur l'idée d'utiliser la fonction `Microsecondes()` en convertissant en temps réel le résultat de la dite fonction sous forme de chaîne de caractères, puis de récupérer une à une les valeurs qui nous intéressaient. Néanmoins, cette idée n'a pas aboutie car :

- cette solution était lourde et peu optimisée
- la fonction ne semblait se réinitialiser qu'à chaque compilation et non à chaque nouvelle exécution de notre programme
- enfin, récupérer chaque valeur de la chaine de caractère -*string* - était une étape longue et difficile à mettre en œuvre à partir du moment où les valeurs changeaient à chaque exécution.

C'est pourquoi nous avons décidé d'utiliser une nouvelle variable: à savoir la variable secondes, qui va s'incrémenter en fonction de microsecondes() et de la variable suivant, elle-même incrémentée par la constante définie `CYCLE`.



Néanmoins, un problème subsistait: la vitesse du serpent était contrainte : le serpent avançait toutes les secondes. Afin d'accélérer sa vitesse, nous nous sommes appliqués à séparer le timer de la vitesse du serpent. Ainsi, la vitesse du serpent n'était plus dépendante de la variable suivant mais était influencée par la variable celerite.
Le *timer* est enfin prêt, nous pouvons désormais reprendre où nous nous étions arrêtés : réaliser la queue du serpent.

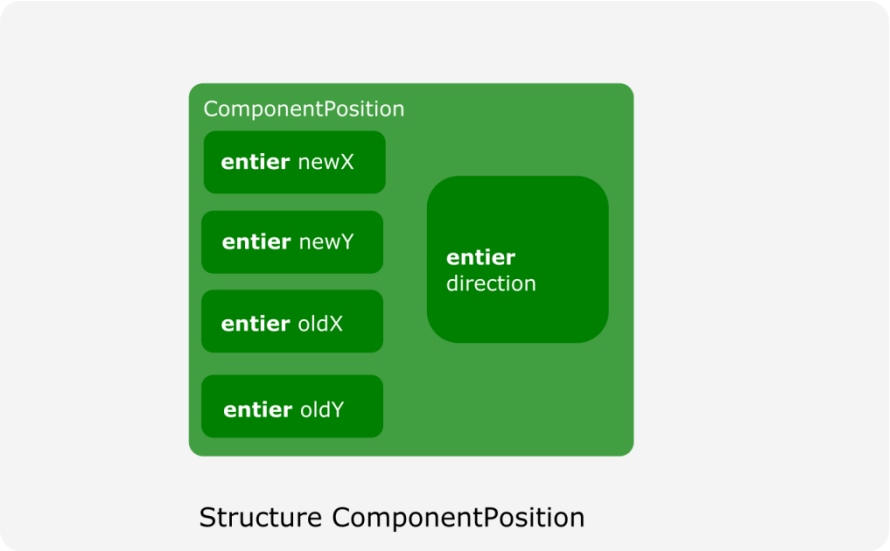
queue[positionQueue, XouY]

Pour ce faire, nous avons dans un premier temps essayé avec un tableau multidimensionnel, de type queue[taille][2], dans la deuxième dimension nous y avions mis stockés les positions Y et X, bien que les résultats aient été plutôt concluant, nous nous sommes rendu compte que la queue était « un bloc » et non pas une multitude de segment indépendant.

```
int queue[TAILLE_MAX_SERPENT][2];
queue[0][1] = Snake_X;
queue[0][2] = Snake_Y;
```

Prenant conscience de cette faille, nous avons décidé de revoir notre façon de concevoir la queue puis nous avons conclu qu'il fallait abandonner le tableau multidimensionnel pour une structure, cette solution sera plus concluante puisqu'elle permet de faire de chaque

composant de la queue une partie indépendante. On peut ainsi retravailler une à une les valeurs correspondantes des positions X actuelle, Y actuelle; mais aussi et surtout user des anciennes valeurs de X et Y.

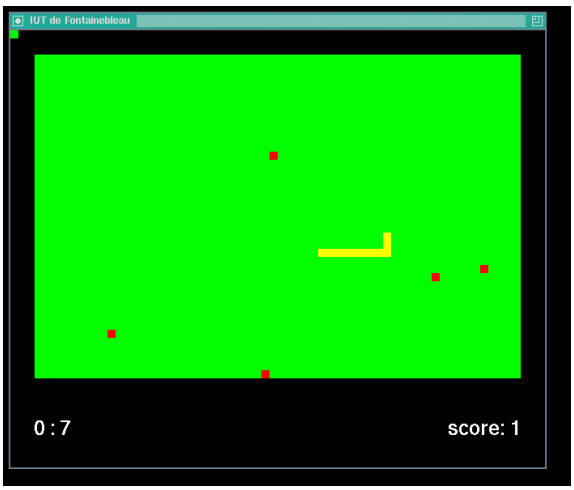


Structure ComponentPosition

Nous avons pu grâce à cette structure jouer avec les positions successives que le *Snake* aura parcouru, ce qui rendra plus facile l'ajout de nouveaux segments et surtout la prise de virage.

COLLISIONS ET SCORE

Suite à cette étape qui aura été la plus laborieuse du projet, nous avons pu mettre en place le système de collision. Le principe de collision est simple : lorsque la tête du serpent touche une partie de sa queue, ou lorsque la tête sort du terrain, c'est-à-dire la zone délimitée en vert, le joueur a perdu et obtient son score et son temps de réalisation. Puis viens l'ajout du score, cette étape a été mise en place grâce à notre fonction `currentScore`. L'ajout c'est fait de la manière suivante : lorsque le snake « mange » une pastille, l'utilisateur voit son score évolué de 5 points et la queue son nombre de segments augmenté de deux.



L'image suivante nous montre déjà l'évolution du serpent, le serpent est opérationnel, le *timer* est ajouté ainsi que la gestion du score, cependant il reste encore quelques défauts , notamment dans la gestion de la pause et du Game Over.

PAUSE: OU COMMENT METTRE LE JEU EN ATTENTE

Pour réaliser la gestion de la pause, nous sommes partis du principe très rependu que lorsque le joueur qui est alors en pleine partie appuie sur la touche « espace », le jeu se met en pause. Nous avons donc réalisé un code portant sur le même principe en déclarant une variable nommé pause, qui lorsque celle-ci sera égale à 1, le jeu continuera à jouer, sinon le jeu se met en pause.

Une fonction nommé `paused()` servira quant à elle à prévenir le joueur que la partie est en pause. Si le joueur souhaite reprendre la partie, il n'aura qu'à appuyer de nouveau sur la touche « espace ».

GAME OVER

Pour finir ce projet, nous avons mis en place une fonction qui annonce la fin de la partie. Dans un premier temps nous avons mis en place en place une variable nommé `still_playing`, si celle-ci est égale à 0, cela veut dire que le joueur à partie et si elle est égale à 1 cela veut dire que la partie continue.

L'affichage et le décor du Game Over est effectué grâce à la fonction `gameOver()` .

Cette dernière fonction achève notre projet, ceci sera le dernier élément que nous avons mis en place pour réaliser à bien l'objectif qui nous a été donné. J'espère que vous prendrez autant de plaisir à jouer à notre *Snake* que nous avons pris à le réaliser .

CONCLUSIONS PERSONNELLES

William CLEMENT

Réaliser ce jeu en C a été un vrai défi, aussi bien technique qu'humain. N'ayant pas pu réaliser le projet en C de l'année passé, ce Snake m'a permis de remettre le pied à l'étrier.

La plupart des projets que j'entame sont souvent abandonnées peu après leur phase de prototypage. J'aime concevoir un travail mais le terminer est une expérience difficile. Néanmoins, ce projet et le fait d'être en équipe, d'avoir quelqu'un qui dépend de mon travail, m'a poussé à donner le meilleur de moi même. Avant ce projet, je pensais que le travail d'équipe allait être dur voir impossible.

Youcef m'a prouvé le contraire et je pense que ce projet m'a apporté beaucoup au niveau du travail d'équipe.

Ce projet marque pour moi la fin d'une page, et le début d'une nouvelle. Et je retiens de cette expérience une idée: l'envie d'apprendre à apprendre.

Youcef ANNAB

Ce projet a été bénéfique pour moi car j'ai pu améliorer et découvrir des méthodes de programmation dans le langage C que je ne connaissais pas encore, prenons par exemple la queue du Snake et ses fonctionnalités (la queue qui s'incrémente, qui peut prendre les virages…). J'ai aussi appris à travailler en groupe, moi qui lors de ma scolarité et de mes expériences professionnelles passées ai toujours travaillé en autonomie, ce travail en groupe a été une première et m'a permis à mieux dialoguer avec mon collègue, ce qui est essentiel pour le monde du travail.
Les contraintes rencontrées se sont principalement portées sur le temps défini pour mettre à bien ce projet, mais aussi de bien pouvoir se coordonner avec mon collègue pour pouvoir se voir le plus possible et pouvoir réaliser le Snake côte à côte, ce qui nous a permis d'échanger d'innombrables idées. Ces rencontres m'ont beaucoup appris, car je l'admets volontiers, les conseils de mon collègue William m'ont beaucoup aidé à comprendre des notions fondamentales que j'ai pu mettre en œuvre pour ce projet.