TomNoob's blender

Rapport de soutenance 2

Table des matières

1 Introduction

TomNoob's Blender, un nom difficile à prononcer. EMSY, simplement initiaux des prénoms des membres de l'équipe. Le projet TomNoob's Blender par EMSY avance à grands pas. 4 amis se lancent dans cette aventure amusante pour créer un ... Candy Crush ! Pour cette deuxième soutenance, les étapes à faire ont été réalisées, nous n'avons pas pris de retard. Commençons la présentation de l'avancement du projet TomNoob's Blender.

2 Tâches individuelles

2.1 Dragan

Après avoir fait la structure du réseau de neurones, je me suis attaqué à son aspect évolutif. L'évolution du réseau de neurones est quelque chose de complexe, non seulement il faut faire "muter" le réseau pour le pousser à évoluer aléatoirement, en espérant qu'un des changements sur le réseau sera positif. De plus il faut aussi le faire "fusionner", pouvoir que deux réseau de neurones fusionnent l'un avec l'autre. C'est assez évidemment une tâche complexe.
Tout d'abord j'ai dû coder en plus de la fonction mutate, une fonction cross. Cette fonction a pour but de faire se croiser deux matrices de doubles, afin de fabriquer deux matrices filles issues de croisement des deux matrices mères. Suite à ça, il ne restait plus qu'à faire l'exploitation de ces fonctions.
La fonction de mutation de réseau de neurones a été faite. Nous avions déjà une fonction mutation et il a suffit de l'appliquer impérativement au reste du réseau de neurones sur les couches FC, c'est-à-dire les couches qui contiennent les matrices poids et biais.
La fonction Network Cross exploite la fonction Cross, cela permet donc de fusionner deux réseaux de neurones parents pour avoir deux réseaux de neurones enfants.

2.2 Céline

En tant que cheffe de groupe, j'ai continué à organiser les réunions pour que l'on prenne, chacun, connaissance de l'avancée du projet.

Redistribuer le travail si nécessaire et surtout donner des dates précises pour tel travail pour ne pas prendre de retard.
Étant donné l'avance prise pour la première soutenance, je n'avais plus de tâches à faire dans la partie mécaniques à part régler les bugs. Cependant il restait une grosse partie qui est le réseau de neurones, nous avons donc décidé de répartir les fonctions à réaliser pour ne pas laisser Dragan et Katia avec tout ce travail. J'ai donc finalement aidé l'équipe s'occupant du réseau de neurones.

2.3 Katia

Ayant déjà travaillé sur le réseau de neurones et les mécaniques du jeu, je me suis concentrée sur les fonctions qui donneront au réseau de neurones ses données d'entrées. Ceci a impliqué des petites modifications dans certaines des fonctions faites précédemment et donc parfois quelques problèmes et bugs. Mais ça, c'est parce que la perfection n'existe pas.

2.4 Kenjy

Pour continuer l'interface graphique, je me suis occupé de réaliser un drag and drop pour que le jeu soit jouable intégralement avec l'interface graphique. Le drag and drop m'a demandé quelques modifications des fonctions déjà existantes pour pouvoir fonctionner, mais avec l'aide des fonctions de Katia et Céline j'ai réussi à le rendre fonctionnel.

3 Récit de la réalisation

Ce projet a apporté beaucoup de nouvelles expériences pour chaque membre. Tout d'abord cela a permis d'améliorer le travail d'équipe. En effet ce genre de projet nous oblige à communiquer entre les membres. Il faut communiquer pour une bonne avancée et surtout il ne faut pas avoir peur de montrer où l'on est moins bon pour pouvoir s'améliorer soi-même. Être ingénieur c'est travailler en équipe, c'est donc une qualité importante à acquérir. Savoir communiquer et s'exprimer pour se faire comprendre et surtout donner son opinion et savoir faire des compromis. Tous les membres d'un groupe n'ont pas forcément les mêmes avis ou opinions, il faut donc savoir gérer ça et trouver la meilleure solution qui va être la plus bénéfique au projet même si cette dernière ne va pas plaire à tout le monde. Nous avons aussi développer notre capacité à nous organiser entre les cours, les devoirs, le projet et le temps de loisir. Nous avons organisé des réunions toutes les semaines et si nécessaire plus pour faire un point sur ce qui devait être fait et sur ce qu'il y avait à faire, discuter des problèmes rencontrés et tenter de trouver une solution ensemble.

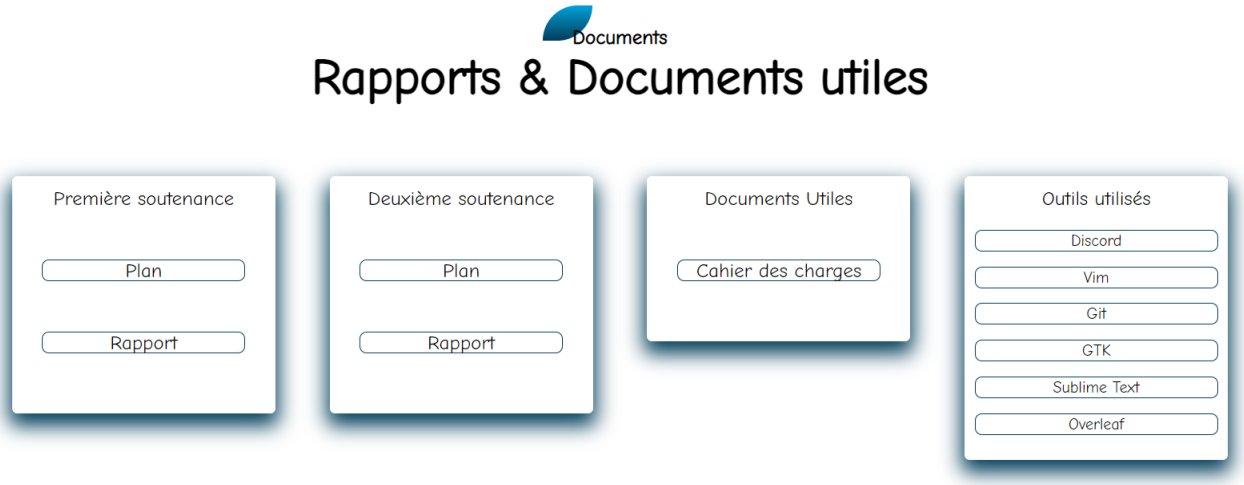
Ensuite ce projet nous a permis d'acquérir de nouvelles compétences et aussi un nouveau point de vue sur le codage en C. Malgré le fait que tout le monde n'a pas eu l'occasion de participer à toutes les parties, nous avons tous eu la chance d'apprendre de nouvelles choses en C en faisant des recherches pour ce projet. On en a aussi appris en grâce à l'explication des membres pour comprendre les problèmes rencontrés et tenter de trouver une solution tous ensemble même

si nous n'y participons pas explicitement.En faisant l'interface graphique, cela a permis de mieux comprendre GTK. Nous n'avons pas eu l'opportunité de voir en détail ce que fait GTK pendant les TP mais avec ce projet nous avons dû faire beaucoup de recherche pour avoir l'interface que nous désirions. Pour le réseau de neurones c'était une tout autre histoire.
Le plus compliqué était la fusion avant les soutenances de toutes les branches créées depuis celle d'avant. Lors de cette fusion, de nombreux bugs apparaissaient et il fallait donc les résoudre. Nous ne pouvions pas fusionner avant car nous travaillions toujours dessus mais pas trop tard pour avoir le temps d'arranger les problèmes.

4 Avancée globale

4.1 Site Web

Le site n'a pas changé, si ce n'est que le plan ainsi que le rapport de soutenance ont été ajoutés. Dans la partie documents, nous avons ajouté un bloc pour les documents de la deuxième soutenance mais mis à pas cela nous pensions pas que le site nécessite de changement pour le moment.



4.2 Réseau de neurones

4.2.1 Informations sur les mouvements

Pour que le réseau de neurones sache quel mouvement est bien ou non, il faut lui donner des informations pour qu'il puisse évaluer ce qui va lui faire gagner le plus de points. Pour cela, nous avons réalisé une fonction qui va récupérer les informations de tous les mouvements possibles comprenant les coordonnées des fruits à échanger, le score engendré par le mouvement et si le mouvement engendre un SF, un FL ou un FC. Toutes ces informations seront stockées dans une liste contenant des listes avec les informations de chaque mouvement possible. Cette liste contient des double, c'est-à-dire des flottants pour permettre au réseau de neurones de Dragan de pouvoir utiliser ces informations.

Cette liste (double *info_list) est composée de listes qui contiennent :

- info_list[nb*8 + 0] = numéro de ligne du premier fruit / 10
- info_list[nb*8 + 1] = numéro de colonne du premier fruit / 10
- info_list[nb*8 + 2]= numéro de ligne du deuxième fruit / 10
- info_list[nb*8 + 3] = numéro de colonne du deuxième fruit / 10
- info_list[nb*8 + 4] = score engendré par le mouvement / 10000 (il faut un flottant pour que le réseau de neurones puisse l'utiliser correctement)
- info_list[nb*8 + 5] = 1 si un SF apparaît suite au mouvement sinon 0
- info_list[nb*8 + 6] = 1 si un FC apparaît suite au mouvement sinon 0
- info_list[nb*8 + 7] = 1 si un FL apparaît suite au mouvement sinon 0

Tout d'abord il faut récupérer tous les mouvements possibles. Pour cela on a une fonction qui va tester pour tout fruit aux coordonnées i et j si ce fruit peut engendrer un mouvement. Si c'est le cas alors la fonction va remplir la liste avec les informations obtenues. Les tests sont très explicites, il va tester tous les cas de mouvements possibles, c'est-à-dire 28 cas. (La fonction est très très longue). Si un mouvement possible est détecté alors la fonction va changer les valeurs de la liste de liste correspondante à ce mouvement.

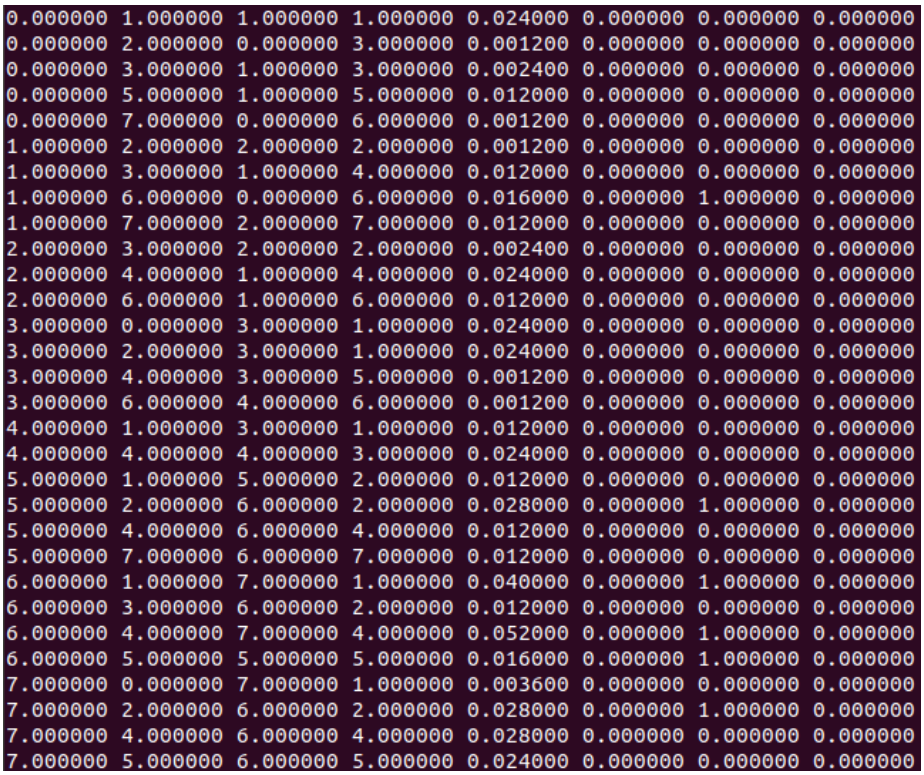
Pour calculer le score engendré par un mouvement, on a la fonction qui "simule" un swap de fruit. Les fruits tombent, cependant,

contrairement à lorsqu'on joue, de nouveaux fruits aléatoire ne tombent pas. Cela permet d'avoir un score minimum obtenu pour le mouvement. Minimum car en fonction des fruits qui tomberont lorsqu'on qu'on fait réellement le mouvement des combinaisons peuvent se créer et donc ajouter de nouveaux points. Cette partie n'est pas contrôlable car ces fruits sont générés aléatoirement et nous voulons laisser la chance faire son travail.

Pour éviter de réecrire les fonctions fruitsfall, cmd, parmi d'autres (en réalité presque toutes les fonctions), nous avons ajouté un condition à ces fonctions, qui correspond à si oui ou non nous voulions de nouvelles de nouveaux fruits aléatoire, ainsi les fonctions se font passer la condition.

Il aurait été impossible de ne générer au moins une erreur, non ? Vous voyez où je veux en venir, il y a des situations que nous n'avions pas prévu. Ces problèmes sont décris dans la section "Les problèmes rencontrés".

Voici à quoi ressemble le fichier nommé *information* lorsque cette fonction est exécutée :



4.2.2 L'évolution

Elle se passe en trois étapes, la 1ère est une étape d'évaluation. Chaque réseau est évalué individuellement et un score lui est attribué. Pour le moment nous faisons des tests avec XOR, l'évaluation est donc faite en fonction de la différence qu'a le résultat attendu avec le résultat espérée. Plus la différence est petite, plus le réseau est réussi. Mais quand nous en arriverons à la phase finale, il suffira de changer la méthode d'évaluation. Le réseau de neurones jouera et le score obtenu sera alors la note. Ensuite les réseaux sont triés du plus efficace au moins efficace.

Finalement on applique les diverses modifications au réseau, le but

est de faire des modifications aléatoires. Pour cela on a deux opérations, la mutation et le croisement qui sont détaillés un peu plus bas dans ce rapport. Dans les détails de la phase de modifications, le nombre de réseaux va être découpé en 3 parties, 10% d'entre eux seront considérés comme les meilleures réseaux, 10% seront les moins efficaces donc les derniers et les 80% restants. Les meilleurs réseaux sont conservé tels quels dans la génération suivante, pour pouvoir piocher des réseaux fonctionnels. Les derniers réseaux, eux, sont supprimés et remplacés par des réseaux tout nouveaux pour avoir de nouveaux réseaux utilisables pour les croisements, pour ajouter de la diversité aux réseaux. Ensuite, le reste des réseaux est déterminé en piochant au hasard dans les réseaux de la précédente génération. Les meilleurs réseaux ont plus de chance d'être choisis que les autres pour la mutation et le croisement, afin d'avoir un résultat plus proche de quelque chose de fonctionnel. Après que toute la génération soit remplie, le cycle évolutif recommence.

4.2.3 La mutation

Pour la première soutenance, nous avions présenté une fonction nommée "mutate" qui fait muter aléatoirement une matrice. Nous avons maintenant utilisé cette fonction pour faire muter un réseau de neurones. Nous partons du principe que la mutation se fait uniquement sur les matrices poids et biais pour les matrices type F, les matrices A, elles, sont simplement régénérées. Pour cette fonction, elle fut assez simple à réaliser mais son application reste très limité.

4.2.4 Les croisements

Contrairement à la mutation, il est en général plus probable de pouvoir utiliser des croisements pour avoir des réseaux performants. Le but du croisement est de mélanger deux réseaux de neurones ensemble en espérant obtenir un réseau plus avancé et mieux construit sur la base de deux autres réseaux. Il est probable que le croisement soit un véritable avancement pour l'apprentissage évolutif mais, même si la fonction "cross" marche correctement, nous rencontrons quelques problèmes à faire marcher proprement l'arithmétique des pointeurs qui va avec. En réalité, toute les fonctions qui en découlent fonctionnent aussi, mais elles se heurtent à quelques problèmes sur l'espace mémoire et la libérations des précédents réseaux de neurones. Il faudra donc faire un peu de ménage avant de pouvoir utiliser de façon optimale toutes les possibilités qu'offrent le croisement.

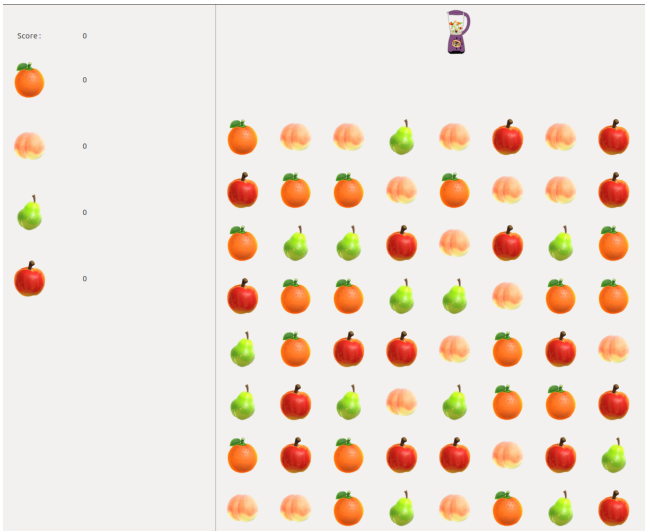
4.3 Interface & terrain

Pour réaliser le drag and drop, j'ai dû utiliser un aspect de GTK que j'avais très peu utilisé jusqu'alors durant ce projet : les signaux. Le terrain étant composé de GTKImages, cliquer sur celles-ci ne produisait aucun signal. J'ai donc rajouté un GTKEventbox par dessus le GTKGrid contenant les images, ce qui a eu pour effet de détecter les clics dans l'espace de la grille.

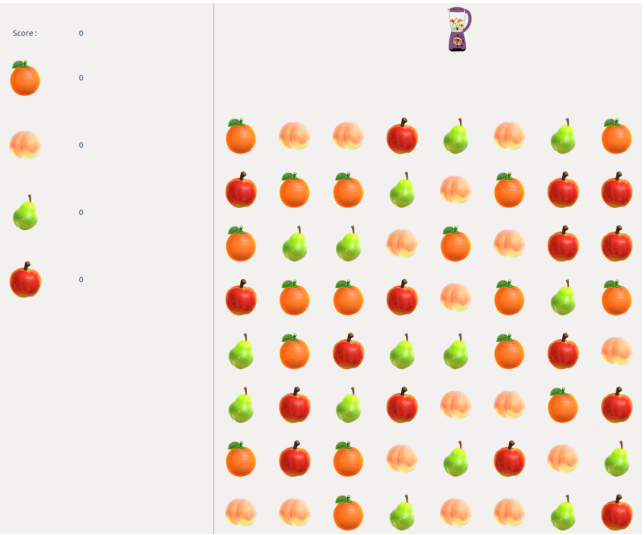
J'ai ensuite utilisé les coordonnées de la souris ainsi que les signaux on-click et on-release des GTKButton pour rendre le drag

and drop fonctionnel. Pour ça, j'ai créé une variable globale de type int*, qui allait contenir les coordonnées des cases à échanger, puis j'ai créé des fonctions qui se font appeler chaque fois que les signaux on-click et on-release apparaissent. Ces fonctions stockent alors les coordonnées de la souris au moment du click et au moment où on relâche le clic dans la liste.

Il ne me restait plus qu'à utiliser les fonctions de Céline et Katia pour rendre le jeu fonctionnel. En créant cette liste de coordonnées, j'avais facilement accès aux fruits à échanger. Et avec mes précédentes fonctions, j'avais facilement accès à la matrice de int représentant le plateau. Le drag and drop est alors fonctionnel.



Après avoir déplacer un fruit le terrain devient ça :



4.4 Jeu (Candy Crush)

Le plus gros a été fait pour la première soutenance, il ne restait qu'à corriger les bugs. Par exemple lorsqu'un fruit spécial était échangé vers le haut avec un fruit normal, le fruit normal ne disparaissait pas. Plusieurs bugs identiques étaient présents et nous avons dû trouver leur origine et les corriger.

De plus certaines fonctions qui avaient été réalisées pour la première soutenance n'étaient pas explicites rendant leur compréhension à la lecture confuses (cf. le rapport de soutenance 1). C'était notamment le cas de la fonction cmd qui comprenait de nombreux if dans un switch et aussi un switch dans un switch. Nous avons décidé de pallier cette confusion en supprimant les switchs et en les remplaçant par des simples if pour chaque cas possible. Pour rappel, la fonction cmd teste le type des deux fruits sélectionnés et fait appel aux fonctions correspondantes à leur type. Après ce changement, la fonction

est beaucoup plus longue mais plus explicite pour, vous, le jury à lire.

5 Les problèmes recontrés

5.1 Le réseau de neurone

Nous avons rencontré au moins une infinité de problèmes durant la conception de l'évolution pour le réseau de neurones. Le 1er d'entre eux, et un problème que nous rencontrons encore est la gestion de la mémoire. Vous imaginez bien que créer des générations de 500 réseaux de neurones environ 1000 fois pour les entraîner est relativement coûteux en espace mémoire. Il faut donc faire très attention à la gestion de la mémoire, à l'ensemble des réservations qui sont faites pour le réseau de neurones et à où ces références de réservations sont stockées.

Dans la même veine de problèmes, peu de problèmes sur le réseau en soi ont été rencontrés. Mais il y avait un moment où l'algorithme de notation se basait sur la matrice de sortie de la 1ère couche au lieu de travailler avec les résultats de la dernière couche du réseau de neurones. Cela rendait assez évidemment l'entraînement du réseau de neurones très complexe car il ne lisait pas du tout les bonnes valeurs de sortie.

Pour faire de la résolution de bug dans le réseau de neurones, j'ai effectué beaucoup de changements internes, dont un "léger" changement sur la compilation pour avoir des codes d'erreur plus précis (comme par exemple l'ajout du flag "fsanitize=address"), l'erreur

que j'ai eu n'était pas en C, mais dans le Makefile en lui-même. En réalité il y avait bien une variable avec tous les flags de compilation mais la commande de compilation était exécutée sans ces "flags" rendant l'identification des lignes impossible par exemple (pas le flag -g). Suite à la résolution de ce petit problème, nous avons pu régler de nombreux petits problèmes au sein du programme, comme par exemple les boucles "for" faisait des comparaisons entre des "int" (la valeur qui grandit entre chaque itération) et des "size_t" (la valeur qui correspond à la "taille" d'un réseau, son nombre de couches par exemple).

5.2 Informations sur les mouvements

Les modifications faites sur les fonctions précédemment implémentées nous ont fait comprendre que nous n'avions pas pensé à tous les cas possibles. Par exemple, un alignement de plus de trois "cases" vides qui a provoqué une boucle infinie, que nous avons heureusement rapidement corrigé.

Un problème où nous avons eu plus de mal à corriger est dû au type. Le score est un entier non signé que nous avons stocké dans un tableau d'entiers. Originellement, nous avions cast lors de l'affectation au tableau, mais le programme n'était pas content et nous donnait des 0. La solution est de cast lors du return de la fonction simulation.

6 À réaliser

Pour la prochaine soutenance, nous avons déjà commencé à réfléchir à comment afficher chaque étape du jeu. Le but étant de comprendre d'où vient les points et le nombre de fruits détruits. Ce qui pourrait être sympatique lorsqu'on réussit à faire de très gros combos. De plus il faudra actualiser le score car il ne l'est pas encore sur l'interface. Il faudra aussi rajouter le meilleur score et le fruit spécial, fruit ligne et fruit colonne sur la partie gauche de l'interface.

Si vous vous en souvenez, nous avions dit lors de la première soutenance que nous allions lier le réseau de neurones avec le jeu. Nous l'avons fait, mais il est vrai que nous n'avons pas encore fait jouer le réseau, c'est dans notre to-do list. Toujours sur le sujet du réseau de neurones, il nous faudra améliorer ce dernier pour prévenir l'apparition de bugs et autres soucis d'adressage mémoire tout en améliorant la vitesse de mutation pour augmenter les chances de mutations et modifications à impact positive sur le réseau.

Une autre fonctionnalité à rajouter est la possibilité de jouer. En effet lors de la rédaction du cahier des charges, nous n'avions pas pensé à rendre le jeu jouable par quelqu'un et pas seulement par l'intelligence artificielle. Nous allons donc tenter de rajouter une option pour passer du jeu par l'IA au jeu lui-même.

7 Planning

Quelques modifications ont été faites quant à la répartition des tâches.

C = Commencé B = Bien avancé T = Terminé

P = Participation (correction/ajustement)

7.1 Soutenance 1

Tâches	Dragan	Kenjy	Katia	Céline
Site Web			T	P
Structure du réseau de neurones	T		T	
Entraînement du réseau de neurones	B		B	
Interface		T		~~T~~ P
Le terrain		C		C
Le jeu		~~C~~	B	~~C~~ B

7.2 Soutenance 2

Tâches	Dragan	Kenjy	Katia	Céline
Site Web			T	P
Structure du réseau de neurones	T		T	
Entraînement du réseau de neurones	B		B	
Lien réseau / jeu			B	B
Interface		T		P
Le terrain		B		B
Le jeu			T	T

Concernant le terrain, il restera à le rendre plus harmonieux et attrayant.

7.3 Soutenance 3

Tâches	Dragan	Kenjy	Katia	Céline
Site Web			T	P
Structure du réseau de neurones	T		T	
Entraînement du réseau de neurones	T		T	
Lien réseau / jeu			T	T
Interface		T		P
Le terrain		T		T
Le jeu			T	T

8 Conclusion

La priorité pour la seconde soutenance était le drag and drop ainsi que l'entraînement ou plus précisément la création de la "connexion", du lien entre le réseau de neurones et le jeu.

Il nous reste la plus grosse partie : tout relier. Pour cela nous allons devoir travailler en équipe pour fusionner le travail de tout le monde. Comme toute fusion, les bugs vont apparaître et nous allons devoir les supprimer. Des bugs de tout type qui ne seront pas forcément compréhensibles mais heureusement internet est là pour nous aider.

Quant à cette soutenance encore, bien qu'elle soit rapidement arrivée, nous avons réussi a être dans les temps. L'entraînement du réseau de neurones est bien avancé mais demande encore quelques améliorations et l'ajout d'une fonctionnalité de fusion. Le lien entre le jeu et le réseau a bien avancé mais demande encore du travail, nous somme même en avance là dessus. L'interface est terminée et le jeu aussi ! Il nous reste peu de choses et elles sont toutes bien avancées.

À travers ce projet, nous avons l'occasion d'appliquer des connaissances et des compétences nouvellement acquises. Même si l'aspect code est important, la gestion de projet est particulièrement réussi cette année, avec des réunions régulières, une entraide importante entre les membres du projet et une gestion de git particulièrement

efficace qui permet à tous de travailler sans crainte de déranger les autres membres alors qu'ils travaillent aussi.

Je pense ne pas avoir faux lorsque je dis que les projets depuis le S2 nous ont permis de mieux gérer notre temps, communication et gestion de projet. Le coronavirus a changé certaines choses. Par exemple lors des semaines en distanciel nous n'avions pas de temps de trajet donc plus de temps pour dormir, manger, faire du thé ou travailler en fonction des personnes.