TomNoob's blender

Rapport de soutenance 1

Table des matières

Page

1 Introduction

Pour correctement aborder notre projet, il est important de savoir que cela est le 2e projet que nous réalisons en tant que EMSY et que même pour le projet de S2, le groupe était déjà composé de Céline, Kenjy et Dragan. Nous travaillons bien ensemble et ce projet ne fait pas exception, voire même, nous avons le sentiment que tous ces mois de travaux ensemble nous ont préparé à ce projet qui se déroule sans accroc jusqu'à maintenant. Nous sommes très confiants de pouvoir mener à bien ce projet jusqu'à son terme. Ces projets nous offrent l'opportunité de travailler en groupe et d'avoir un aperçu de ce que peut être notre futur métier.
Finissons cette parenthèse et passons aux choses sérieuses.

2 Avancées individuelles

EMSY est une équipe composée de 4 étudiants de l'EPITA qui est une école d'ingénieur en informatique. Nous avons réuni nos compétences pour vous proposer un nouveau projet pour ce semestre 4. Concernant l'équipe, chaque membre a ses compétences qui permet d'avoir un groupe homogène. De plus nous nous connaissons depuis la SUP et avons déjà travaillé ensemble. Nous savions donc déjà comment nous organiser et comment les autres travaillent. Le groupe s'est fait naturellement. Nous connaissons les faiblesses de chacun et grâce à cela nous pouvons les surmonter pour pouvoir apprendre les uns des autres.

2.1 Dragan

De mon côté j'ai travaillé sur deux points, notamment sur le réseau de neurones et sur la gestion du repository git.

Tout d'abord sur le réseau de neurones, j'ai pu travailler sur la structure générale du réseau, une structure qui convient à tout type de réseau et à toutes sortes de spécifications. Ce travail m'a étrangement changé de celui que j'ai pu faire sur l'OCR et je suis bien plus à l'aise avec la structure générale du réseau et l'organisation d'un projet en plusieurs parties.

J'ai ensuite aussi énormément travaillé cette année encore sur l'outil de gestion de version git. Mon expertise s'étant révélée bien plus utile cette année sachant que la gestion du git fait partie de ce qui nous est demandé.

2.2 Céline

Étant cheffe de projet, j'ai dû organiser les réunions et faire en sorte que chacun avance correctement de leur côté. De plus on peut dire que je suis la designeuse du groupe, c'est-à-dire je m'occupe de l'esthétique.
Depuis la validation de notre cahier des charges, j'étais en charge des mécaniques du jeu avec Kenjy. Cependant l'interface lui a pris une grande partie du temps dédié au projet alors Katia m'a rejoint sur les mécaniques. Les mécaniques du jeu comprenant par exemple le déplacement des deux fruits sélectionnés ou encore la tombée des fruits après la destruction d'une ligne ou d'une colonne ou tout ce qui suit. Toutes les fonctions réalisées seront décrites ci-après.

2.3 Katia

Je me suis focalisée dans un premier temps sur le site internet. Contrairement à mon précédent site, celui-ci contient en plus du html et du css du javascript. Ce dernier de rendre le site plus fluide et agréable pour l'utilisateur.

De plus, j'ai étudié avec Céline les différents bonbons et mouvement de Candy Crush. Nous avons étudié les manieères de recréer ces mouvements pour optimiser les fonctions et les tests au maximum.

Enfin, j'ai bien sûr participé au réseau de neurones, nous avons discuté des différentes possibilités d'implémenter la structure, et pour aller plus loin, l'entraînement (avec la potentialité d'utiliser

des threads).

2.4 Kenjy

Je me suis occupé de la partie interface graphique, où j'ai principalement utilisé GTK. J'ai donc codé en xml et en c pour utiliser correctement les outils de la librairie. J'ai également dû comprendre le code de Céline pour pouvoir intégrer les mécaniques à la partie graphique du projet.

3 Avancée globale

3.1 Site Web

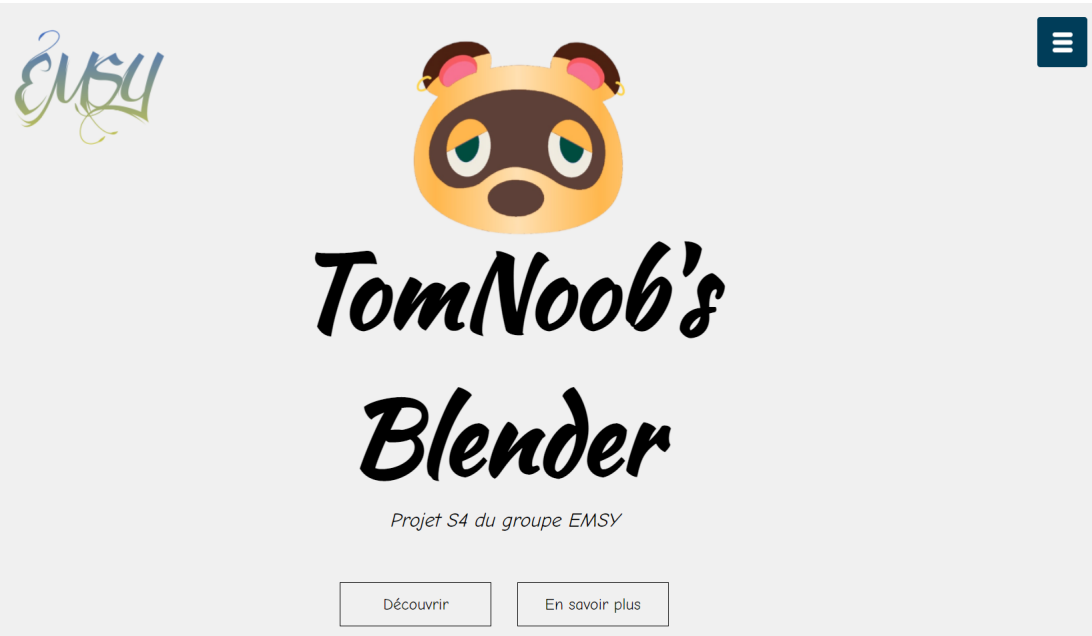
Le site est terminé, en effet le squelette du site est prêt et les documents tels que les rapports et plans de soutenance et manuel d'utilisateur y seront ajoutés plus tard, le cahier des charges en revanche est déjà accessible. Le site sera régulièrement mis à jour pour tenir les utilisateurs au courant des nouveautés et des changements.

Le site est hébergé par GITHUB, le lien est le suivant :

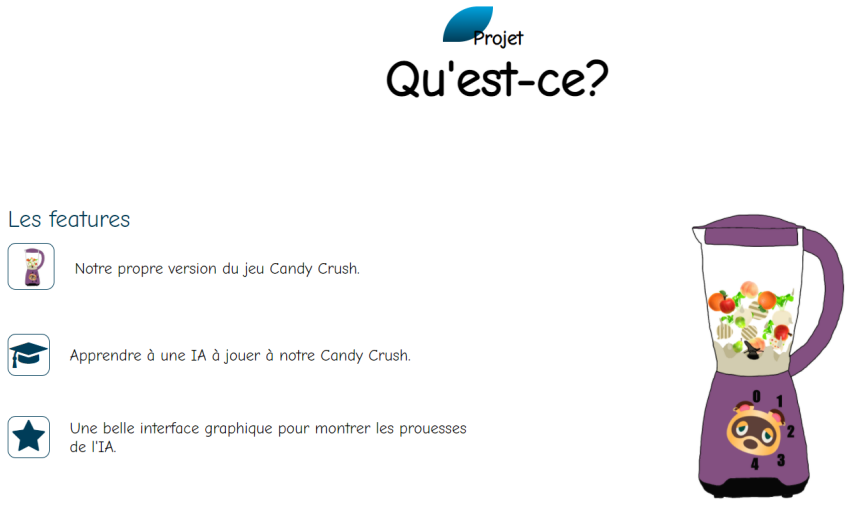
https://epiemsy.github.io.

Le site est composé de quatre parties :

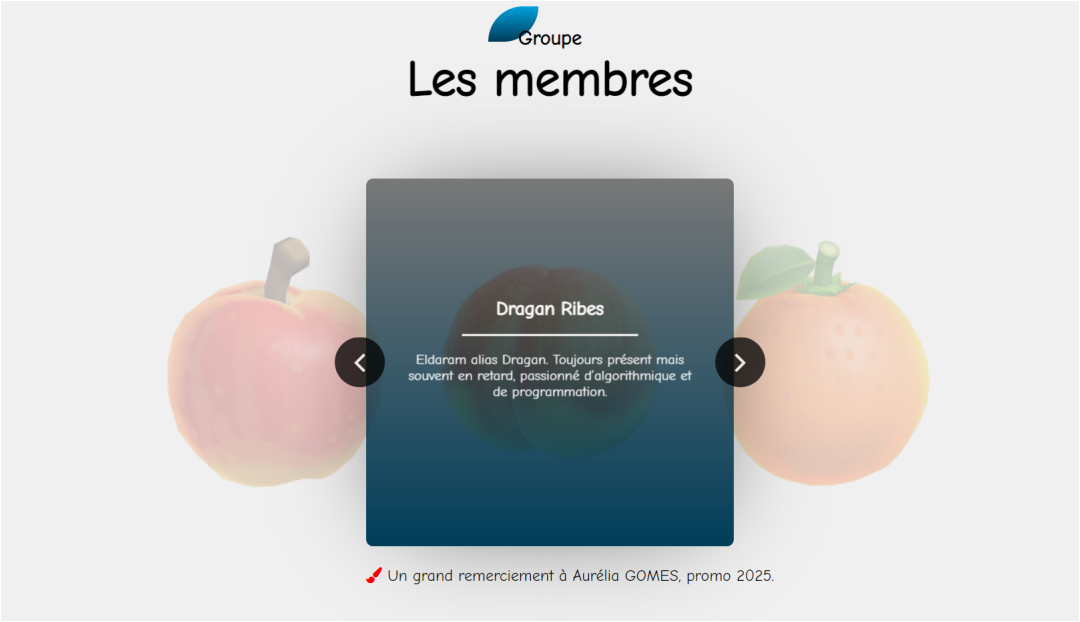
— La bannière



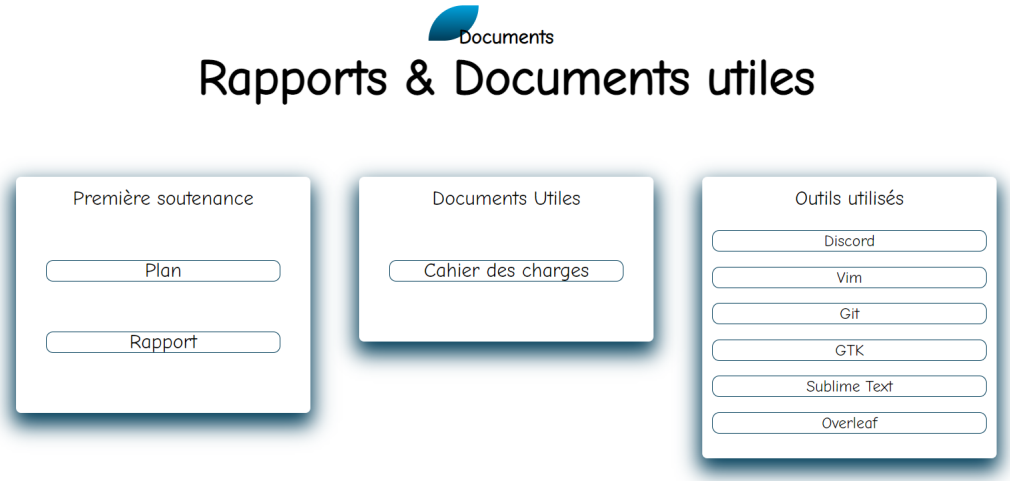
— Le projet avec quelques un de ses features



— Le groupe avec une simple présentation de chaque membre sous forme de carrousel



— Les documents, on peut y retrouver notamment le cahier des charges, plans et rapports de soutenance ainsi que les outils utilisés



Il y a aussi un menu sur la droite qui permet de faire défiler la

page jusquà la partie désirée. Du Javascript a été utilisé pour cette fonction, ainsi que pour le carrousel et le menu.

3.2 Réseau de neurones

3.2.1 Structure

Comme promis la structure du réseau de neurones est terminée ! Le but de cette structure est de faire quelque chose de très très modelable car le nombre de couches et de neurones par couche doit pouvoir varier. En réalité, il y a quelques similarités avec l'OCR, mais l'OCR était moins complexe, simplement parce que le réseau de neurones était déjà verrouillé sur un type. Ici nous devons gérer bien plus de variables et il faut bien plus détailler les calculs pour les modifier. Il a aussi fallu créer un nouvel objet de type Network** qui sert à faire une liste de réseau de neurones, accompagnée d'une fonction qui génère des réseaux de neurones. La structure est ainsi

terminée et nous permettra de développer sereinement l'entraînement de ce dernier.

Majoritairement les erreurs rencontrées sont assez communes, beaucoup d'erreurs vis-à-vis des pointeurs. L'utilisation des pointeurs simplifie énormément le travail de création du réseau mais il demande une certaine rigueur et énormément de précisions. L'allocation de mémoire dynamique pose aussi ses propres problèmes, en effet les fonctions comme "Malloc" ou "Free" rendent la gestion de la mémoire nécessaire pour un projet de cette envergure. À l'avenir, il faudra être très précautionneux vis-à-vis de l'entraînement car il y aura de nombreux réseaux de neurones et à chaque génération 90% d'entre eux ne seront plus utilisés, il faudra donc penser à les libérer.

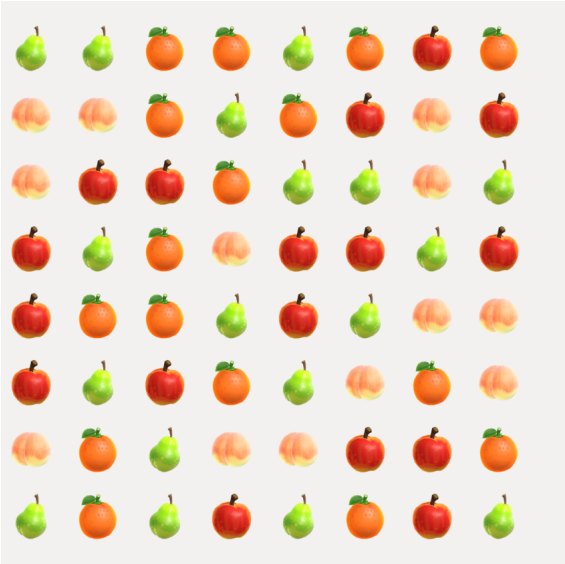
3.2.2 Entraînement

L'entraînement a été commencé, majoritairement en adaptant la procédure "forward propagation", la procédure de prédiction.

3.3 Interface

L'interface à été codée en utilisant la librairie GTK. Le code est donc divisé en deux parties : Un code en C et un code en XML. Je me suis énormément aidé de la documentation GTK disponible sur le site développer.gnome.org pour appréhender la librairie. Pour créer l'interface, j'ai commencé par la construire avec un peu de code XML que j'ai récupéré dans le tp pong réalisé au S3 puis je l'ai modifié pour séparer l'écran en 2 parties aux dimen-

sions définies. Une de ces parties accueille le score et le nombre de fruits détruits et l'autre est une grille représentant le terrain qui accueille les fruits. La grille faite 8 fruits par 8. J'ai ensuite créé les 64 fruits à la main sur XML pour les gérer plus facilement individuellement. Une fois le code XML réalisé, l'interface était prête, seulement tous les fruits étaient les mêmes. C'est là qu'intervient le code en C. J'ai récupéré la fonction de Céline qui génère une matrice de int aléatoire, chaque int représentant un fruit différent, puis j'ai créé une matrice de GtkImage auxquelles j'ai modifié le fichier d'image. De la l'interface générait une matrice de fruits aléatoire bien visuelle. J'ai continué en reprenant la fonction de Céline vérifiant si une une matrice possède des enchaînements de plus de trois fruits et je l'ai adapté à ma matrice de GtkImage.



3.4 Terrain

En attendant que l'interface soit réalisée, la grille ressemble à cela :

```
    0   1   2   3   4   5   6   7
  -----------------------------------
0 | 3 | 2 | 2 | 3 | 2 | 3 | 1 | 1 |
  -----------------------------------
1 | 2 | 1 | 2 | 3 | 0 | 1 | 0 | 0 |
  -----------------------------------
2 | 1 | 3 | 1 | 1 | 2 | 3 | 3 | 0 |
  -----------------------------------
3 | 3 | 0 | 2 | 0 | 1 | 2 | 1 | 1 |
  -----------------------------------
4 | 1 | 0 | 3 | 2 | 3 | 0 | 3 | 2 |
  -----------------------------------
5 | 0 | 2 | 2 | 3 | 1 | 3 | 1 | 0 |
  -----------------------------------
6 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 |
  -----------------------------------
7 | 1 | 3 | 2 | 0 | 0 | 2 | 1 | 1 |
  -----------------------------------
score = 0
APPLE  : 0  |  ORANGE : 0  |  PEER   : 0  |  PEACH  : 0  |  FL     : 0  |  FC     : 0  |  FS     : 0  |
```

Les fruits sont représentés par leur numéro (cf. 3.5.1). Le score et le nombre de fruits détruits sont également affichés.

Ensuite pour sélectionner les deux fruits, le joueur entre le numéro

de ligne et le numéro de colonne du premier fruit qu'il veut déplacer puis le numéro de ligne et le numéro de colonne du deuxième fruit comme cidessous :

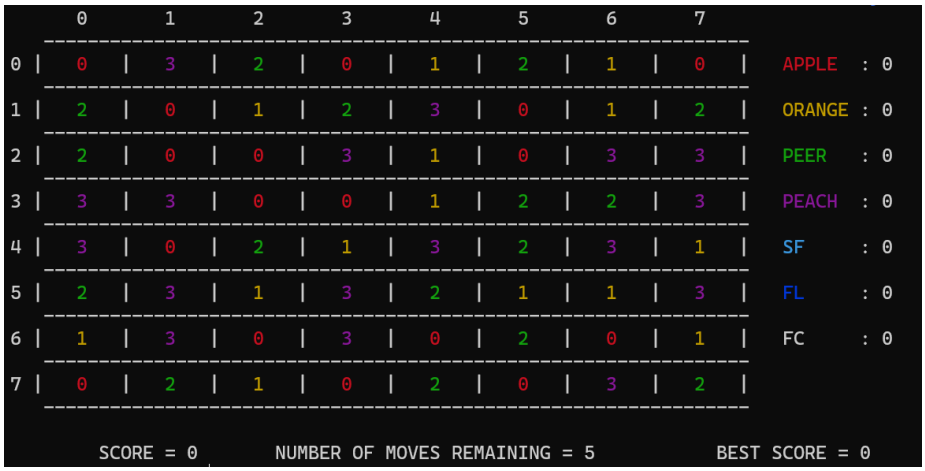
```
First fruit line: 1
First fruit column: 1
Second fruit line: 2
Second  fruit column: 1
```

Nous avons apporté quelques modifications pour rendre cela plus esthétique.

```
❯ ./jeu
    0       1       2       3       4       5       6       7
  ---------------------------------------------------------------
0 |   0   |   0   |   1   |   2   |   3   |   0   |   1   |   2   |   APPLE  : 0
  ---------------------------------------------------------------
1 |   1   |   1   |   0   |   1   |   3   |   2   |   2   |   1   |   ORANGE : 0
  ---------------------------------------------------------------
2 |   2   |   1   |   3   |   2   |   1   |   3   |   2   |   0   |   PEER   : 0
  ---------------------------------------------------------------
3 |   0   |   0   |   2   |   2   |   3   |   0   |   0   |   1   |   PEACH  : 0
  ---------------------------------------------------------------
4 |   1   |   2   |   3   |   0   |   3   |   1   |   3   |   3   |   SF     : 0
  ---------------------------------------------------------------
5 |   1   |   3   |   2   |   1   |   0   |   0   |   2   |   2   |   FL     : 0
  ---------------------------------------------------------------
6 |   3   |   1   |   1   |   0   |   1   |   3   |   0   |   2   |   FC     : 0
  ---------------------------------------------------------------
7 |   3   |   3   |   0   |   2   |   0   |   1   |   2   |   1   |
  ---------------------------------------------------------------

                        SCORE = 0
```

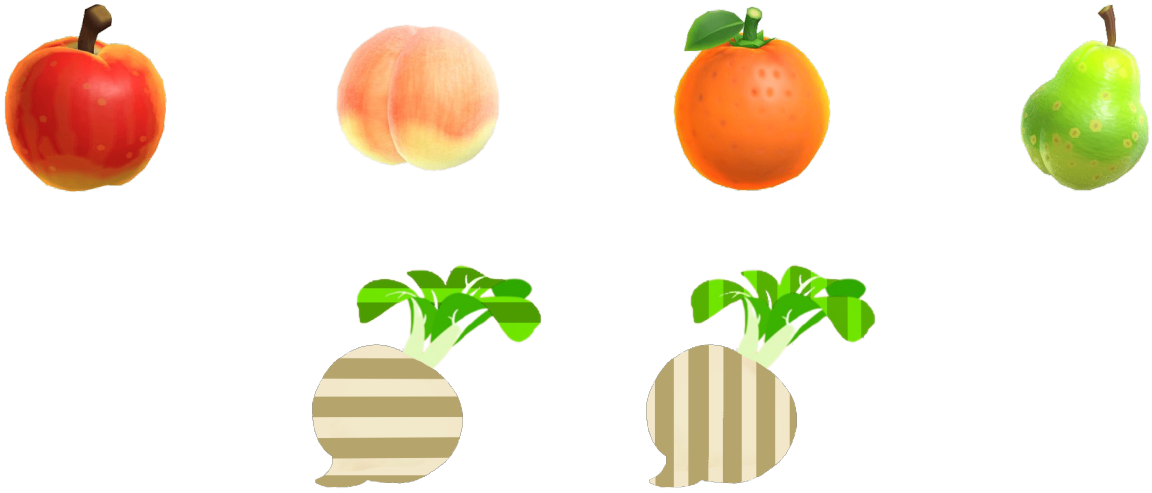
Ensuite nous avons rajouté le nombre de mouvements restants et aussi le meilleur score qui sera actualisé à chaque fois que ce dernier sera dépassé.



3.5 Jeu

3.5.1 Les fruits

Tout d'abord présentons les fruits normaux sans effet spécial. Pour ce qui est des fruits nous avons choisi la pomme (0), la pêche (1), l'orange (2) et la poire (3).



Le fruit ligne (FL : 5) et le fruit colonne (FC : 6) détruisent res-

pectivement une ligne et une colonne lorsqu'elles sont sélectionnées. Nous avons ajouté quelques particularités. Lorsqu'un FL et un FL ou un FL et un FC ou un FC et un FC sont sélectionnés, la ligne d'un FC ou FL et la colonne de l'autre fruit sont détruites (ce choix dépend de l'ordre des fruits sélectionnés).

Et enfin le fruit spécial (SF : 4). Ce fruit, lorsqu'il est sélectionné avec un fruit normal, détruit tous les fruits du même type que le fruit normal choisi. Le fruit spécial avec un FC ou un FL détruit tous les fruits d'un type normal au hasard de la grille. Deux fruits spéciaux sélectionnés détruisent TOUTE la grille.

Pour l'histoire, ces fruits sont des navets qui sont présents dans Animal Crossing : New Horizons. Porcelette les vend tous les dimanches sur notre île. Nous avons demandé à Aurélia GOMES en SUP (l'artiste renommée d'EPITA) de modifier le navet originel pour avoir les fruits colonne, ligne et spécial.

3.5.2 Mécaniques

Les mécaniques comprennent, comme dit précédemment, l'échange des deux fruits sélectionnés, la tombée des fruits après ce mouvement ou encore les fruits spéciaux et leur effet.

Nous allons maintenant lister les fonctions importantes réalisées :

- void generategrid(int matrix[][]) remplit la matrice matrix avec des fruits au hasard choisis entre la pomme, la poire, la pêche et l'orange. Pour cette fonction nous utilisons la fonction rand(), provenant de la librairie stdlib.h, avec % 4 pour avoir un chiffre entre 0 et 3.
- void fruitsfall(int matrix[][], size_t column) fait tomber les fruits de la colonne column après la destruction de fruits. Les fruits du dessus tombent et lorsque tous les fruits sont tombés, de nouveaux fruits sont générés pour compléter le nombre de fruits manquants.
- void destroyline(int matrix[][], size_t line) détruit la ligne line entièrement et appelle la fonction fruitsfall appelée pour chaque case pour régénérer la ligne.
- void destroycolumn(int matrix[][], size_t column) détruit la colonne column entièrement et appelle la fonction fruitsfall pour régénérer la colonne.
- void transSF(int matrix[][], int type) détruit tous les fruits type de la grille et fait appel à la fonction fruitsfall pour faire retomber les fruits lorsqu'un fruit type est détruit.
- int checkaround(int matrix[][], size_t l, size_t c, int condition)

vérifie autour du fruit aux coordonnées (l, c). Si une ligne ou une colonne d'au moins 3 fruits existe alors ces fruits vont être détruits et la fonction fruitsfall va être appelée pour faire tomber les autres fruits. Si la condition est START alors le score ne changera pas malgré les modifications de la grille sinon le score est incrémenté selon le mouvement réalisé et le compteur de fruits détruits change également. Cette fonction retourne TRUE s'il y a eu un changement dans la matrice sinon FALSE.

— void cmd(int matrix[][], size_t l1, size_t c1, size_t l2, size_t c2) va tester les deux fruits sélectionnés. S'ils sont identiques et des fruits normaux alors rien ne se passe. Sinon les deux fruits sélectionnés sont échangés par une fonction swap puis les fruits sont testés et les appels aux fonctions sont faits par rapport aux fruits sélectionnés.

— int checkgrid(int matrix[][], int condition vérifie toute la grille en appelant la fonction checkaround. checkgrid est appelé après un mouvement car la retombée des fruits peut créer des combinaisons. Cette fonction retourne TRUE si un quelconque changement est réalisé sinon elle retourne FALSE.

3.5.3 Combinaisons

SF = fruit spécial FL = fruit ligne FC = fruit colonne

FN = fruit normal (pomme, pêche, poire, orange)

Nous avons fait une liste des combinaisons possibles dans Tom-Noob's Blender :

— 3 FN = 120 points. Les 3 FN sont détruits.

— 4 FN = 160 points. Les 4 FN sont détruits et FC (ou FL) apparaît si les 4 FN sont sur la même ligne (ou respectivement même colonne).
— 5 FN = 200 points. Les 5 FN sont détruits et SF apparaît.
— FN + SF = 40 * (nb FN) + 240 points. Tous les fruits FN de la grille sont détruits.
— FN + FC/FL = 180 + (nb FN) * 40 + (nb FL) * 40 + (nb FC) * 40 points. La colonne/la ligne des coordonnées d'arrivée du FC/FL.
— SF + FC/FL = 240 + 180 + (nb FN) * 40 points. Tous les fruits d'un type choisi au hasard sont détruits de la grille.
— FC/FL + FL/FC = 180 * 2 + (nb FN) * 40 + (nb SF) * 240 + (nb FC/FL) * 180 points. La ligne et la colonne sont détruites formant donc une croix.
— SF + SF = 240 * 2 + (nb FN) * 40 + (nb FL/FC) * 180 + (nb SF) * 240 points. Toute la grille est détruite et est régénérée entièrement.

4 À faire

Il faudra finaliser toutes les mécaniques du jeu car certains bugs apparaissent. Nous avons aussi pensé à rajouter des fonctionnalités. (pas sûr du tout) L'interface est presque terminée et fonctionnelle, il faudra la modifier pour la rendre plus harmonieuse. Ne reste plus pour l'interface qu'à parvenir à faire bouger les fruits, d'abord par clic entre les deux fruits à échanger, puis si possible par drag and drop. Après avoir tout perfectionner, nous allons lier le jeu avec l'in-

telligence artificielle pour lui permettre de jouer.

5 Chefs d'œuvre

Un grand remerciement à Aurélia GOMES qui est actuellement dans la promo 2025 en SUP. Nous la connaissons depuis plus d'un an, malheureusement nos chemins se sont séparés dû à son redoublement. Cependant nous restons toujours en contact car c'est une très bonne amie pour tous les membres du groupe. Il n'est pas faux de dire que Aurélia fait partie de l'équipe EMSY. En effet, la majorité des dessins qui sont présents dans notre projet viennent d'elle. Nous avons suscité ses talents et elle a accepté volontiers.
Voici tous les dessins qu'elle a réalisé :

Bon courage à elle pour cette année!!!

6 Planning

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

C = Commencé B = Bien avancé T = Terminé

P = Participation (correction/ajustement)

6.1 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

Pour la soutenance 1, nous avons pris de l'avance sur le jeu en lui-même. Les mécaniques ont bien avancé mais doivent encore être corrigées car des bugs sont encore présents.

6.2 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	
Interface		T		P
Le terrain		B		B
Le jeu			C	C

6.3 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	
Interface		T		P
Le terrain		T		T
Le jeu			T	T

7 Conclusion

Pour résumer, ce qui devait être fait a été fait avec un peu d'avance. Cette avance nous permettra de rajouter des fonctionnalités et d'améliorer le jeu. Celui-ci est fonctionnel mais nous voulons le rendre plus esthétique. De plus il reste une grosse partie à faire qui est de relier le réseau de neurones avec le jeu. La liaison entre le terrain fait avec GTK et les mécaniques a demandé beaucoup de temps car il fallait créer 64 signaux pour relier chaque case à sa valeur dans la matrice.
Ce projet étant né d'un accord entre les membres nous intéresse et c'est pour cela qu'il avance à un bon rythme et sans problème (pour l'instant).