Soutenance 2

PAC-MAN IA

par A2CR STUDIO

Composé par :

Alexandre Bourcier
Clément Bruley
Clément Iliou
Rémi Monteil

Table des matières

1 Introduction

Ce rapport de soutenance va présenter toutes les avancées de notre projet du S4. Ce projet a pour but de créer une intelligence artificielle et de l'a montré visuellement à l'utilisateur. Le but ici n'est évidemment pas de recoder le célèbre jeu Pac-Man, mais bien de se concentrer sur comment développer une IA qui fera un gros score.

Ce rapport va donc décrire tout d'abord l'équipe et l'origine du projet avec une présentation des différents membres. Il y aura ensuite une description du projet. Pour continuer nous vous proposerons une description détaillé de tout ce que nous avons pu faire depuis la dernière soutenance. Nous y ajouterons des images pour permettre de mieux visualiser notre projet. Enfin nous vous présenterons les objectifs que nous avons pour la soutenance finale.

2 Présentation du projet

2.1 Présentation de l'équipe

Alexandre Bourcier : Entré à l'EPITA en 2019, je me suis toujours efforcé de faire beaucoup de programmation pour avoir des acquis et une expérience solide dans cette discipline. Ainsi, avec le projet de S2, j'avais pris en charge une partie très importante de la programmation d'un jeu vidéo(multijoueur et gameplay). Lors du projet de S2, j'ai pris en charge l'interface graphique, le pré-traitement et la segmentation de l'OCR. Ainsi, toutes ces expériences m'ont permis d'avoir des acquis assez solides et d'aborder ce projet de S4 sereinement. L'idée de refaire une intelligence artificielle me plaît beaucoup, car lors de notre projet de S3, le réseau de neurones était loin d'être impeccable et je souhaite pouvoir prendre ma revanche sur ce projet.

Pendant les projets, j'apprécie toujours commencer tout de suite les projets pour pouvoir vraiment étaler tout le travail sur le temps imparti et ne pas courir à la fin et rentre quelque chose de moyennement fonctionnel. Un point très important aussi est de discuter de la conception du projet, notamment le prototype des fonction et comment les faire interagir. Tout cela nous permettra, j'espère de faire un beau projet fonctionnel.

Clément Bruley : La plupart des élèves d'EPITA ont commencé à toucher à l'informatique depuis de nombreuses années. Pour ma part, j'ai commencé réellement lorsque je suis entré à EPITA. Lors des deux derniers projets, nous avons approfondi nos connaissances en c# lors du S2 et en C99 lors du S3. Ce projet va me permettre d'avoir de plus grandes connaissances en C qui est un langage très utile dans le domaine de l'informatique. Possiblement, nous allons pouvoir appliqué ce que nous avons appris lors du séminaire UNIX. J'ai toujours été intéressé par les IA. Grâce à ce projet, je vais pouvoir développer mes compétences dans ce domaine.

Avec le reste du groupe nous avons décider de m'attribuer le rôle de chef de projet. Il me semble que c'est important d'avoir quelqu'un qui va "diriger" le projet. Néanmoins, ce rôle ne va pas être très grand, car il ne s'agit pas d'un grand projet tel que pourrait le faire des entreprises.

Clément Iliou : Je suis étudiant en deuxième année de classe préparatoire à EPITA. J'ai effectué une terminale S spécialité SVT. Avant EPITA, je n'avais pas fait de programmation dans un cadre scolaire. Cependant, je me suis toujours intéressé à l'informatique en général. J'aime beaucoup les travaux de groupe, car l'on peut ainsi profiter des différents atouts de tous. La partie graphique du jeu m'encourage à approfondir le domaine du traitement d'image que je trouve passionnant. Ce projet est un moyen pour moi de croiser les connaissances acquises en développement de jeu vidéo durant le projet de S2 et celles sur le machine learning et les réseaux de neurones vu pendant le projet de S3. J'ai hâte après EPITA de pouvoir développer ce genre de programme dans un cadre professionnel. Ce projet devrait me permettre d'approfondir mes connaissances en C99 et en GTK.

Rémi Monteil : Je suis rentré à EPITA en 2019. Depuis j'ai appris plusieurs langages tels que le Caml, c# et le c. Mais surtout j'ai appris à les exploiter au maximum avec les projets. J'avais développé le gameplay du jeu de S2 et je me suis occupé du Neural-Network de l'OCR. Je suis très attiré par ce qui touche à l'IA autant dans ces utilisations exotiques que de son apprentissage si divers. La difficulté et les erreurs que j'ai commises lors de l'OCR ne sont que de l'expérience supplémentaire pour appréhender cette nouvelle IA qui me semble plus difficile par sa méthode d'apprentissage non supervisée. J'espère partagées au mieux mes connaissances des projets précédents pour accompagner mes collègues dans ce projet.

2.2 Nature du projet

2.2.1 Description du projet

Le principe du projet est assez simple. Il s'agit de reproduire le jeu Pac-Man et de faire une intelligence artificielle qui essaie de faire le score le plus élevé possible. Le but de ce projet n'est pas de coder un jeu vidéo, mais de se concentrer vraiment sur de l'intelligence artificielle avec du traitement de donnée dans un réseau de neurones. L'implémentation visuelle du jeu nous permettra également de faire un peu de traitement d'image.

2.3 Objet d'étude

Ce projet peut apporter beaucoup en terme d'expérience de programmation et de conception d'IA. Déjà dans un premier temps, il faudra programmer le jeu. Cela va demander surtout d'utiliser des connaissances en GTK+ et en traitement d'image. Il faudra programmer une mini-intelligence des fantômes, les faire bouger, rendre les pacgums interactifs, activer les super pacgums, augmenter le score et évidemment afficher graphiquement tout ça dans notre fenêtre avec GTK+.
Il faudra ensuite se concentrer sur l'intelligence artificielle et là, il y a une grosse partie de réflexion sur comment nous allons faire une intelligence artificielle. Nous pensons utiliser un réseau de neurones, mais nous allons probablement devoir faire de nombreux tests de modèle d'IA, changer les paramètres d'entrée,...

3 Réalisé lors de la dernière soutenance

3.1 Graphismes

3.1.1 GTK et Glade

Afin d'afficher le jeu, nous utilisons la bibliothèque GTK. Cette bibliothèque permet de créer des boutons, ainsi qu'une interface visuelle afin que le joueur puisse jouer. Pour structurer l'interface, nous utilisons Glade. Dans Glade, nous avons créé deux boutons : Start et Pause, qui permettent de démarrer ou de mettre le jeu en pause. Il y a aussi trois GtkLabel qui permettent d'afficher le score, le nombre de vies restantes au joueur et le niveau.

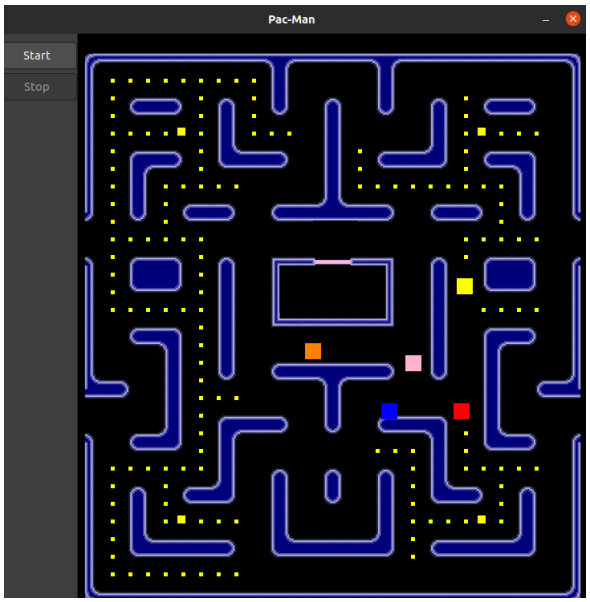


FIGURE 3.1 – Premier graphisme

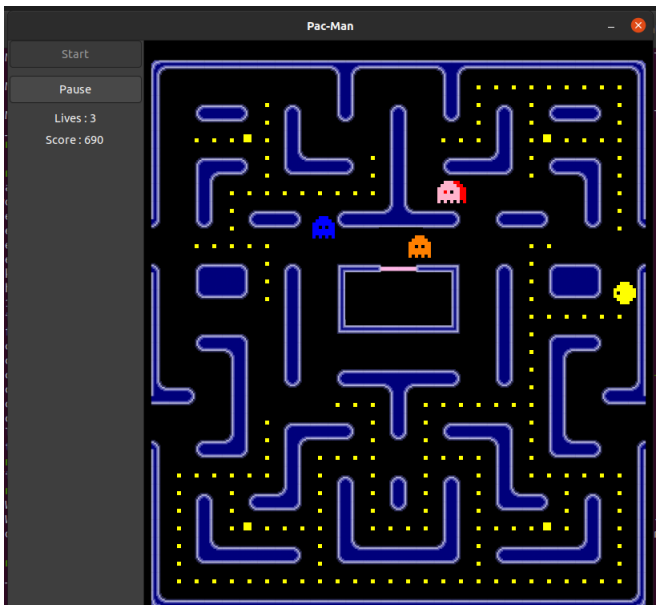


FIGURE 3.2 – Graphismes finaux

3.1.2 L'affichage de la carte et des Sprites

L'affichage de la carte et des Sprites sont séparés. Nous superposons grâce à un GtkOverlay une image et une Drawing Area. Nous avons choisi ce mode d'affichage, car cela nous permet d'avoir un jeu plus agréable visuellement et plus optimisé puisqu'à chaque lancement, la carte n'a pas besoin d'être reconstruite. Pour savoir si une case est un mur, un couloir vide, un couloir avec une pacgum ou une super pacgum nous avons réalisé une matrice de 28 par 31 comme dans le jeu original. Au début, de chaque partie, nous affichons toute les pacgums à partir de la matrice. Pour afficher Pac-man et les fantômes nous avons créé d'autres matrices afin de les afficher en Pixel Art. Pour Pac-man, une image sur trois celui-ci ouvre la bouche comme dans le jeu original. Pour que la direction du pixel de Pac-man change, nous changeons juste le sens du parcours de la matrice.

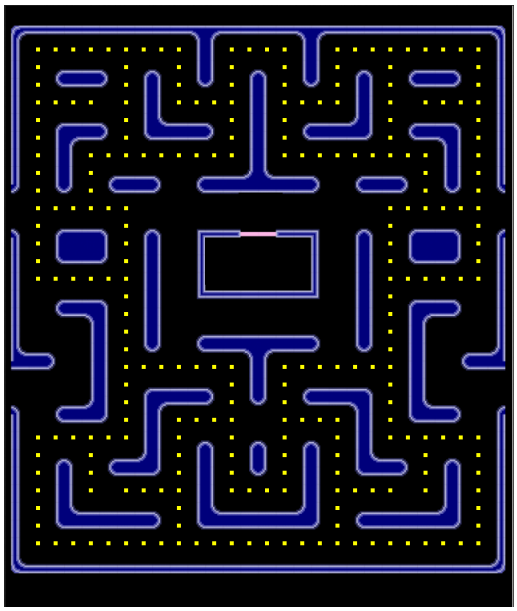


FIGURE 3.3 – Map affichée

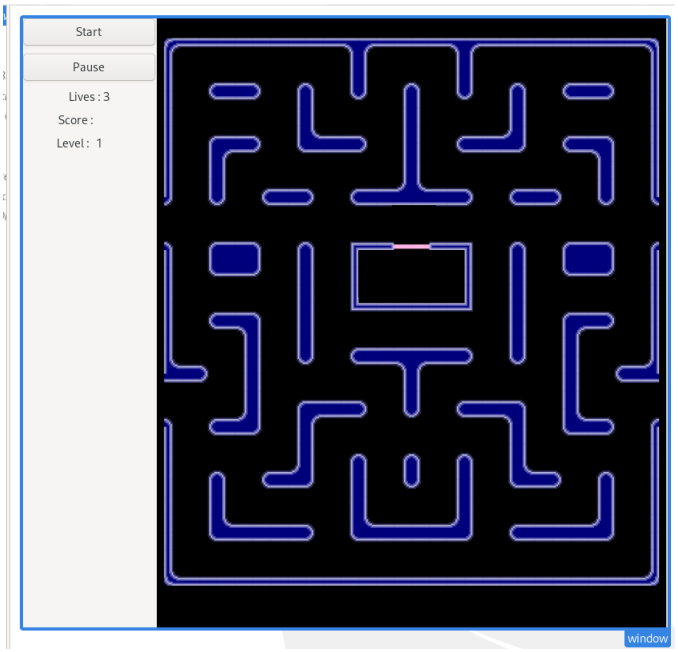


FIGURE 3.4 – Fichier glade utilisé pour faire l'interface et l'affichage de la carte

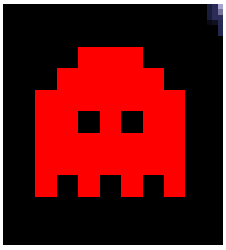


FIGURE 3.5 – Sprite de fantôme

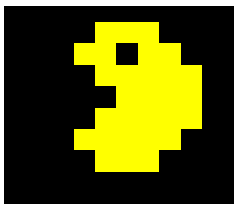


FIGURE 3.6 – Sprite de Pac-man avec la bouche ouverte

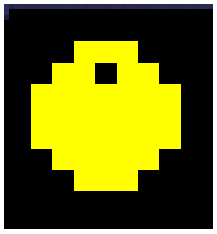
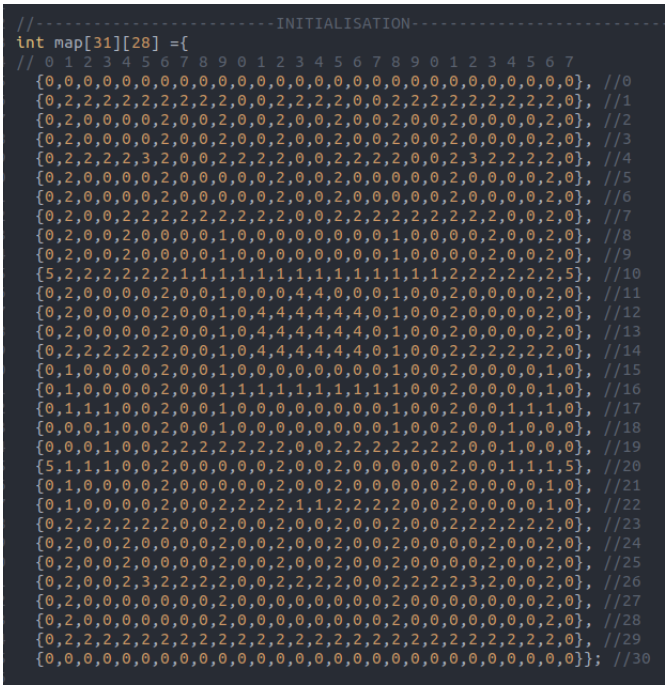


FIGURE 3.7 – Sprite de Pac-man avec la bouche fermée

3.2 Le gameplay du joueur

3.2.1 Structure

La structure du jeu comporte de nombreux paramètres. Cette structure comporte d'abord 5 joueurs : Pac-man et les 4 fantômes. Ces joueurs possèdent 3 caractéristiques : leur coordonnées (en x et en y) et leur direction. La structure du jeu possède aussi le statut pour savoir si le jeu est en pause ou en cours, le mode chasse pour savoir si Pac-man est invincible ou pas, le score, le nombre de vies et le niveau.



```
//------------------------INITIALISATION-------------------------
int map[31][28] ={
// 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, //0
  {0,2,2,2,2,2,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,2,2,2,2,2,0}, //1
  {0,2,0,0,0,0,2,0,0,2,0,0,2,0,0,2,0,0,2,0,0,2,0,0,0,0,2,0}, //2
  {0,2,0,0,0,0,2,0,0,2,0,0,2,0,0,2,0,0,2,0,0,2,0,0,0,0,2,0}, //3
  {0,2,2,2,2,3,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,3,2,2,2,2,0}, //4
  {0,2,0,0,0,0,2,0,0,0,0,0,2,0,0,2,0,0,0,0,0,2,0,0,0,0,2,0}, //5
  {0,2,0,0,0,0,2,0,0,0,0,0,2,0,0,2,0,0,0,0,0,2,0,0,0,0,2,0}, //6
  {0,2,0,0,2,2,2,2,2,2,2,2,2,0,0,2,2,2,2,2,2,2,2,0,0,2,0}, //7
  {0,2,0,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,2,0,0,2,0}, //8
  {0,2,0,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,2,0,0,2,0}, //9
  {5,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,5}, //10
  {0,2,0,0,0,0,2,0,0,1,0,0,0,4,4,0,0,0,1,0,0,2,0,0,0,0,2,0}, //11
  {0,2,0,0,0,0,2,0,0,1,0,4,4,4,4,4,4,0,1,0,0,2,0,0,0,0,2,0}, //12
  {0,2,0,0,0,0,2,0,0,1,0,4,4,4,4,4,4,0,1,0,0,2,0,0,0,0,2,0}, //13
  {0,2,2,2,2,2,2,2,0,0,1,0,4,4,4,4,4,4,0,1,0,0,2,2,2,2,2,0}, //14
  {0,1,0,0,0,0,2,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,2,0,0,0,0,1}, //15
  {0,1,0,0,0,0,2,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,2,0,0,0,0,1}, //16
  {0,1,1,1,0,0,2,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,2,0,0,1,1,1}, //17
  {0,0,0,1,0,0,2,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,2,0,0,1,0,0}, //18
  {0,0,0,1,0,0,2,2,2,2,2,2,2,0,0,2,2,2,2,2,2,2,0,0,1,0,0,0}, //19
  {5,1,1,1,0,0,2,0,0,0,0,0,2,0,0,2,0,0,0,0,0,2,0,0,1,1,1,5}, //20
  {0,1,0,0,0,0,2,0,0,0,0,0,2,0,0,2,0,0,0,0,0,2,0,0,0,0,1,0}, //21
  {0,1,0,0,0,0,2,0,0,2,2,2,2,1,1,2,2,2,2,0,0,2,0,0,0,0,1,0}, //22
  {0,2,2,2,2,2,2,2,0,0,2,0,0,2,0,0,2,0,0,2,0,0,2,2,2,2,2,0}, //23
  {0,2,0,0,2,0,0,0,0,0,2,0,0,2,0,0,2,0,0,2,0,0,0,0,2,0,0,2,0}, //24
  {0,2,0,0,2,0,0,0,0,2,0,0,2,0,0,2,0,0,2,0,0,0,0,2,0,0,2,0}, //25
  {0,2,0,0,2,3,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,3,2,0,0,2,0}, //26
  {0,2,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,2,0}, //27
  {0,2,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,2,0}, //28
  {0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0}, //29
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}}; //30
```

FIGURE 3.8 – Matrice utilisé pour la map

3.2.2 Score, mort et niveaux

Le joueur possède trois vies en début de partie, il en perd une à chaque fois qu'il se fait manger pas un fantôme. Nous vérifions que Pac-man et un des fantômes ne se trouvent pas sur la même case. Lorsque Pac-man se fait manger, celui-ci ainsi que les fantômes retournent à leur emplacement de départ. Nous utilisons alors la fonction Sleep(3) pour permettre au joueur de réfléchir à une nouvelle stratégie. Il possède 2 secondes pour réfléchir.

Lorsque le joueur mange une pacgum nous modifions la valeur de la case dans la matrice. Pour vérifier si le joueur a mangé toutes les pacgums nous utilisons un compteur du nombre de pacgum mangé. Nous réinitialisons la matrice à chaque changement de niveau. Nous ne mettons pas de limite de niveau comme dans le jeu original.

Lorsque Pac-man mange une super-pacgum (il y en a 4 par niveau) celui-ci devient bleue et il peut manger les fantômes. Les mouvements des fantômes changent et ils passent en mode fuite. L'effet de la pacgum dure environ 10 secondes, à la fin Pac-man redevient jaune.

3.2.3 Déplacements

Comme dit plus haut, nous avons créé une matrice de 28 par 31, et nous avons donc un premier système de coordonnées, c'est-à-dire que pour chaque entité nous avons des coordonnées dans cette matrice. Ensuite, nous avons un deuxième système de coordonnées, celles qui correspond au pixels de zone de dessin de GTK.

Ainsi, pour ne pas se perdre dans ce double système de coordonnées, nous avons décider de séparer en 2 fichiers :

- un premier fichier GTK.c qui traite de tout ce qui est en lien avec l'interface graphique et donc le système de coordonnées relatives à la zone de dessin.
- un second fichier pac-man.c qui traite de la gestion du jeu en lui-même avec le système de coordonnées de la matrice.

Pour la direction de Pac-man, nous créons une fonction d'appel dans GTK.c, qui est déclenché lorsqu'on appuie sur une touche. Ensuite on appelle une seconde fonction dans pac-man.c qui vérifie grâce aux coordonnées de la matrice si Pac-man ne va rentrer dans un mur en changeant de direction. On change ensuite la direction de Pac-man.

Concernant les déplacements de Pac-man, on définit la vitesse de celui-ci comme étant le nombre de pixel de déplacement par frame (le jeu est en 24 frames par seconde). Ainsi, à chaque frame, on vérifie que Pac-man ne va pas rentrer dans un mur avec les coordonnées en de la matrice et la direction

puis on modifie les coordonnées en pixel pour que Pac-man soit ensuite re-dessiner dans la zone de dessin GTK.
Pour le déplacement des fantômes, la procédure est similaire à celle de Pac-man sauf que la direction est redéfini à touts les frames et que celle ci tient compte des obstacles. Ainsi, il ne reste plus qu'à mettre à jours le coordonnées.

3.3 L'IA des fantômes

3.3.1 Le pathfinding

Les fantômes utilisent un algorithme de pathfinding. Au moment de la création de la fonction, nous étions et nous sommes toujours dans l'étude des graphes en cours d'algorithmique. L'algorithme privilégié pour trouver le plus court chemin était le parcours largeur. Nous avons donc adapté ce parcours de graphe à celui de l'exploration de labyrinthe dans la matrice.

Comme vu précédemment nous utilisons une matrice qui défini chaque cases du plateau de jeu. Pour faire le plus simple possible nous avons attribué des numéros à chaque "état" que peut prendre la case. Par exemple les cases "0" sont des murs et "1", "2" et "3" sont des couloirs avec ou sans pacgum. Les cases "4" correspondent à la maison des fantômes et les cases 5 sont les télé-porteurs que nous ignorons pour le moment. Pour réussir l'implémentation du parcours largeur, il a fallut avoir les queues qu'on utilise en cours d'algorithmique. Nous avions trouvé la librairie "sys/queue.h". Néanmoins, suite à des difficultés de manipulation et des doutes ressentit, nous avons donc décidé de créer notre propre struct queue qui à été optimisé et corrigé au fil du projet.

Nous avons effectué des recherches sur d'autres algorithmes de pathfinding et avons trouvé le a* qui est décrit comme le plus performant. Mais au vue de la première soutenance, nous avons conclu que ce serait une perte de temps et dangereux de le faire à 1 semaine de la soutenance d'autant plus que notre fonction de pathfinding actuel fonctionne. Mais le changement vers cet algorithme a* sera probablement envisagé et étudié pour la prochaine soutenance.

3.3.2 Le mode chasseur

En mode chasseur les fantômes : Blinky (le rouge), Pinky (le rose), Inky (le bleu) et Clyde (le orange) vont pourchasser Pac-man. Mais ils ont chacun leurs propre technique :

- Blinky : Il va simplement poursuivre Pac-man.
- Pinky : Il est un peu plus malin. Il va tenter de piéger Pac-man. Pour cela il se dirige vers le mur où Pac-man irait s'il continuait dans sa direction actuel.
- Inky : Il est un peu plus compliqué. Il va considérer que Pac-man essayera de fuir en priorité Blinky (étant donné qui semble le plus agressif). Ainsi, il prend comme destination une coordonnés qui est le symétrique ou au alentours du symétrique de la position de Blinky par rapport à Pac-man. Dans certains cas, ce symétrique se trouve hors de la map, il va alors effectuer la même chose que Blinky.
- Clyde : Il est un peu plus fantaisiste. Il va toujours fuir Pac-man. Il va vers les coordonnées étant la plus proche de lui et dans hors d'un rayon de 5 cases autour de Pac-man.

Note : les noms japonais expriment très bien leur mode chasseur respectif :

- Blinky = chasseur
- Pinky = embusqué
- Inky = inconstants
- Clyde = stupide.

3.3.3 Le mode patrouille

Les fantômes ne poursuivent pas constamment Pac-man. Au bout d'une certaine distance de Pac-man, il décide d'arrêter leur chasse et de passer en mode patrouille. Ainsi le fantôme en question va se promener dans le coin le plus proche ou il se trouve. Pour le moment ce mode n'est pas activé.

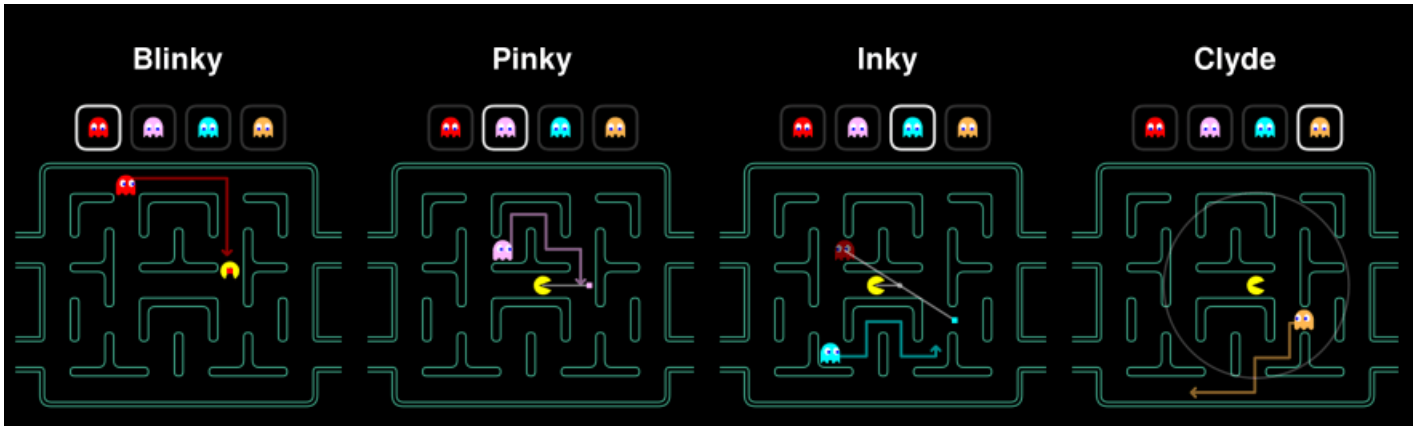


FIGURE 3.9 – Mode chasseur de chaque fantômes

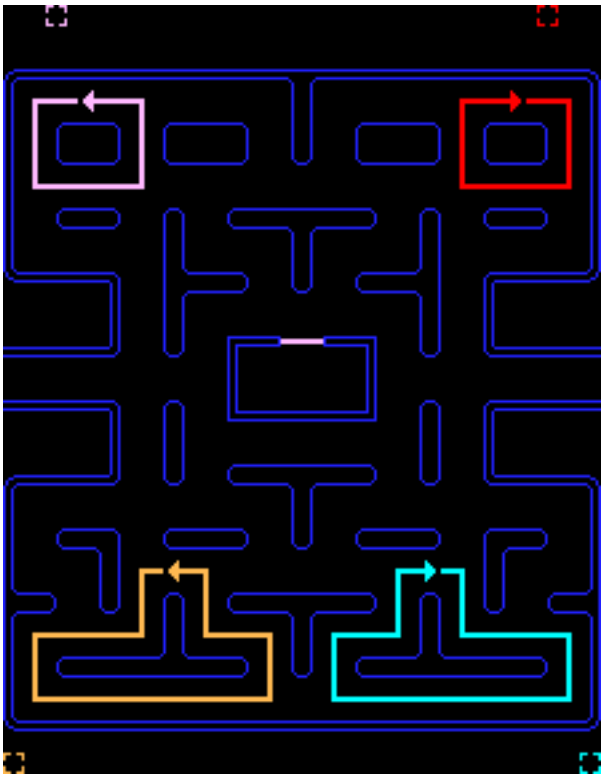


FIGURE 3.10 – Mode patrouille de chaque fantôme

3.3.4 Le mode fuite

Ce mode peut être déclenché uniquement par Pac-man lorsqu'il mange une super pacgum. A partir de ce moment Pas-man à 10 secondes pour manger les 4 fantômes et ainsi obtenir plus de point. A ce moment les mouvements des fantômes sont aléatoires comme dans le jeu original.

4 Découpage et avancement du projet

4.1 Découpage du projet

Tâches à effectuer	Alexandre Bourcier	Clément Bruley	Clément Iliou	Rémi Monteil
Création graphique	⊗		✓	
Gameplay du joueur	✓		⊗	
IA des fantômes		⊗		✓
IA de Pac-Man	⊗	⊗	⊗	✓
Site web	✓	⊗		
Git	✓	⊗		
LaTeX	⊗	✓		

✓ = représentant ⊗ = *assistant*

Les tâches énumérées précédemment ne sont pas de difficulté ni de durée équivalente. En effet, la gestion du git est très rapide à faire tandis que la création de l'IA du joueur est assez difficile.

4.2 Avancement du projet

Tâches à effectuer	**Prévu**	**Effectué**
Création du visuel	99,9%	99%
Animation du joueur	100%	100%
Déplacement des fantômes	100%	96%
IA du joueur	50%	50%
Site web	95%	95%

— Clément Iliou a avancé sur toute la partie gameplay ainsi que la partie visuelle, car c'était déjà sa partie pour la première soutenance et qu'ainsi il maîtrise bien son code et sait comment il faut implémenter les comportements. Il a donc avancé le graphisme des fantômes (de beaux yeux, un mode fuite, ...), il a aussi créé le mode fuite des fantômes et amélioré les déplacements.

— Rémi a créé la structure d'un neurone avec son biais, sa liste de poids, sa liste d'entrée et sa fonction d'activation.

— Alexandre a implémenté tout le système pour faire un réseau de neurone : initialisation, chargements des données, exécution, sauvegarde du réseau et destruction. Il a également implémenté le système de générations, c'est-à-dire la création d'une génération, l'exécution de tout les réseaux de celle-ci ainsi que la sauvegarde des meilleurs réseaux et la création des réseau suivant. Il a aussi implémenté le système de Q-learning, avec son tableau et sa Q fonction. Enfin, il a fini par implémenté l'agent naïf/réflexe pour la soutenance.

— Clément Bruley s'est surtout concentré sur toute la documentation pour accompagner Alexandre et Rémi dans la création du réseau. Il a aussi créé ce latex et s'est occupé de toute la mise en page.

5 Finition du visuel

La finition du visuel est une partie importante car il rend le jeu plus agréable pour l'utilisateur. Le pixel art des fantômes ont été améliorés pour les rendre plus intéressants et réaliste. Lorsque Pac-man mange une super pacgum, les fantômes et Pac-man deviennent bleue et pendant les dernières secondes de l'effet Pac-man clignote pour indiquer au joueur que l'effet va prendre fin. De plus, lorsqu'un fantôme est mangé par Pac-man celui-ci ne peut pas se faire manger à nouveau, il faut attendre que Pac-man mange une autre super pacgum. Les fantômes ne pouvant pas être remangé garde leur couleur d'origine.

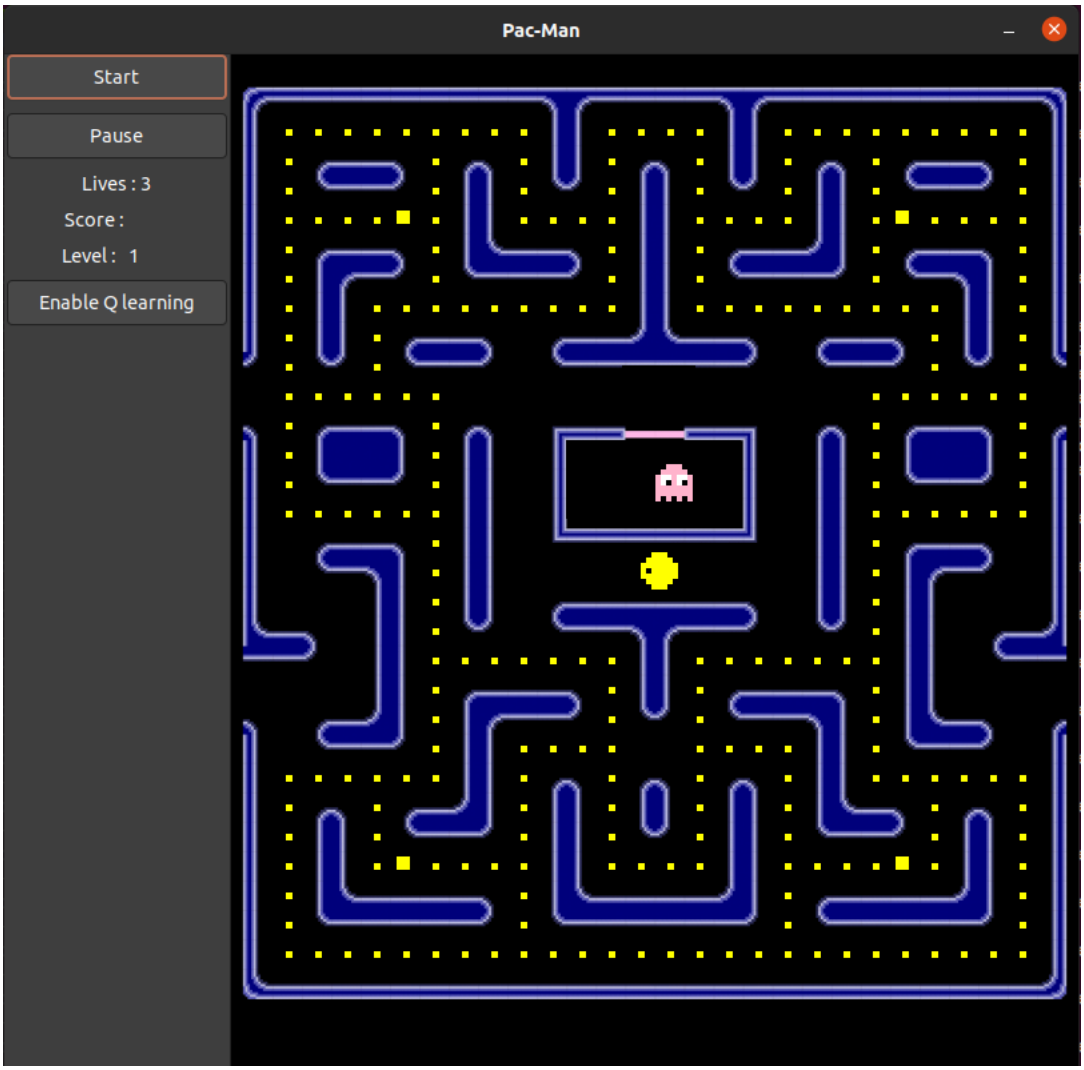


FIGURE 5.1 – Nouvelle interface

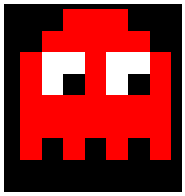


FIGURE 5.2 – Sprite de fantôme

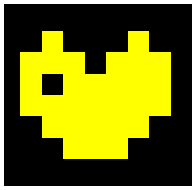


FIGURE 5.3 – Sprite de Pac-man avec la bouche ouverte

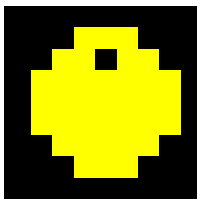


FIGURE 5.4 – Sprite de Pac-man avec la bouche fermée

6 Optimisation de l'IA des fantômes

6.1 Le pathfinding ou A*

Nous avions comme objectif d'implémenter un nouveau pathfinding en utilisant l'algorithme A*. Néanmoins près quelques recherches nous nous sommes aperçu que le temps passé pour concevoir l'algorithme était une perte de temps. En effet étant donnée que nous sommes dans un labyrinthe le parcours largeur classique est presque aussi performant que A*. Ce dernier est conçu pour être très efficace dans de grand espace comme la recherche d'un objet dans une matrice vaste.

6.2 Le mode patrouille

Nous avons ajouté le mode patrouille. Les fantômes alternent entre le mode chasseur et le mode patrouille. Au début du niveau, ils passent 7 secondes en mode patrouille puis 20 secondes en mode chasse et ainsi de suite. Pour chaque fantômes deux éléments ont été rajoutés à leur structure, une liste et un compteur. La liste correspond à une liste contenant les points que le fantôme doit parcourir durant le mode patrouille. À chaque fois qu'un point est atteint le compteur est incrémenté de 1 et donc sélectionne le point suivant dans la liste. Pinky et Blinky possèdent 4 points dans leur liste tandis que les fantômes Clyde et Inky en possèdent 5.

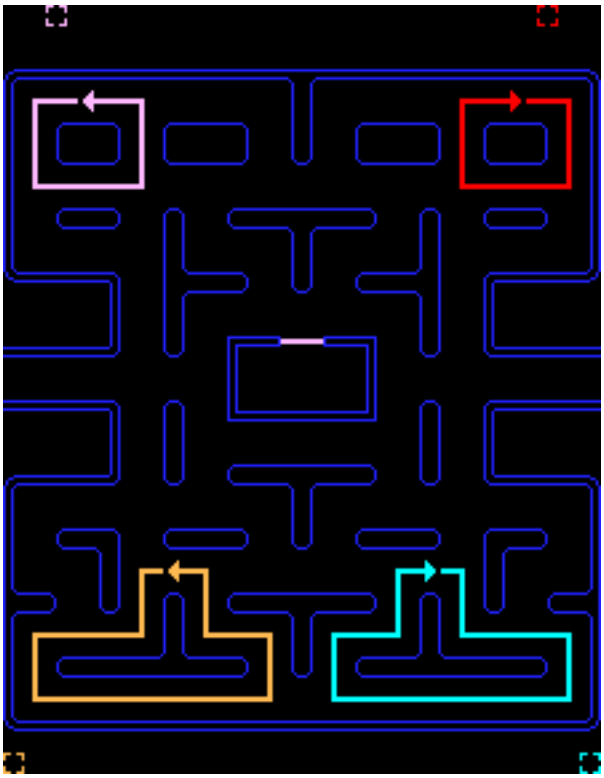


FIGURE 6.1 – Mode patrouille de chaque fantôme

Pour déterminer la direction dans laquelle les fantômes se déplacent on utilise la fonction qui détermine le parcours de Blinky en y remplaçant les coordonnées de Pac-man par le point de la liste.

6.3 Le mode fuite

Le mode fuite est un mode de déplacement aléatoire. Ce mode n'est pas encore totalement achevé en effet nous utilisons la fonction de valeur aléatoire d'une des bibliothèque de base du C mais celle-ci change de valeur moins souvent que l'on lance la boucle du jeu ce qui pose certaine problème. Certaines fois, les fantômes peuvent rester coincé dans un mur pendant quelques secondes. Nous cherchons encore une solution à ce problème.

6.4 Bonus : le mode aléatoire

Le mode aléatoire pour chaque fantôme est déjà implémenté, néanmoins nous ne comptons pas l'activer pour le moment. En effet, notre IA n'est pas du tout assez puissante pour supporter de l'aléatoire chez les 4 fantômes.

7 L'IA de Pac-man

7.1 Le réseau de Neurones

7.1.1 La théorie

Pour l'IA de Pac-man, nous avons donc opté pour un réseau de neurones en ayant pour base l'expérience acquise lors du projet du S3 : l'OCR. Ainsi nous avons récupéré les codes nous permettant la création des neurones, la propagation du réseau ... Ce code sera modifié pour l'adapter au machine learning de Pac-man (modification expliquée dans la partie suivante).

Nous sommes donc encore une fois dans du deep learning. Mais le machine learning de Pac-man est diffèrent de l'OCR. Celui du projet précédent était un apprentissage supervisé. On pouvait lui indiquer à chaque exécution qu'elle était la réponse attendue et les réponses non-attendues et donc pouvoir le corriger avec de la back-propagation. Mais nous ne sommes pas dans un apprentissage non-supervisé, cela fut une erreur de ma part de dire cela en début de projet en me justifiant par mon manque de connaissance dans le sujet. On pourrait croire que cela est le bon machine learning étant donné qu'on est dans l'inconnu pour savoir qu'elle est la bonne action et les mauvaises actions. Inconnue typique à l'apprentissage non-supervisé.

Mais dans le cas des jeux tel que Pac-man cela est nuancé. On ne sait pas qu'elle est la MEILLEUR bonne actions. Dans le projet actuel nous sommes donc dans un apprentissage semi-supervisé ou plus communément appelé apprentissage par renforcement : machine learning adepte des théories du jeux et des jeux. L'apprentissage par renforcement va pouvoir répondre à la question de la meilleure bonne action. Dans ce machine learning, l'IA va étudier son environnement : les murs, les fantômes, les pacgums à son état actuel dans sa partie et ainsi opter pour l'action qui va lui apporter la meilleure récompense. Il existe plusieurs algorithmes permettant le calcule de cette récompense qui se distingue par l'utilisation du réseau. Pour Pac-man, nous avons décidé d'exploiter le Q-learning (explication dans la partie suivantes) qui nous semble être le plus adéquat pour notre Pac-man.

7.1.2 La pratique

EEn terme d'implémentation, nous avons souhaité pouvoir facilement modifier l'architecture de notre réseau de neurones, c'est-à-dire pouvoir modifier en quelques secondes le nombre de couches et de neurones par couche. Ainsi, en fonction de toutes ces données, on va constituer trois listes :

- Une liste d'entrées qui est égale au nombre total de neurones de toutes les couches.
- Une liste de poids qui est égale à la somme du nombre de neurones de la couche précédente multiplié par ceux de la couche actuelle pour chaque couche sauf la première.
- Une liste de biais qui est égale au nombre total de neurones sauf la première couche.

Ces trois listes suffisent alors à pouvoir mettre en place tout notre réseau. On a créé un structure "neurone" pour chaque neurone du réseau. Cette structure est alors constituée de :

- une variable "size" qui contient la taille de la liste d'entrées des neurones.
- la liste "input" qui contient la liste des entrées du neurone (toutes les sorties des couches précédentes)
- la liste "weights" qui contient tous les poids associés aux entrées.
- la liste "bias" qui contient le biais de notre neurone.

Ainsi, pour initialiser correctement le réseau, on déplace les pointeurs de nos 3 listes aux bons endroits en fonction des neurones. Pour pouvoir exécuter le réseau, on appelle une fonction d'activation pour chaque neurone qui fait la somme des poids multipliés par les entrées et son biais correspondant.

On stocke ensuite le résultat dans la liste d'entrées pour la couche suivante. Après avoir exécuté les fonctions pour tous les neurones, on regarde dans la couche finale, le neurone qui a le résultat le plus élevé et l'on retourne la direction qui lui a été attribué.

Concernant les données d'entrée, nous avons 17 inputs :

— La présence des murs dans les 4 directions -> inputs à 1 s'il n'y a pas de murs dans la direction correspondant.

— Le nombre de fantômes dans les 4 directions -> inputs égal au nombre de fantômes divisé par 4 dans la direction correspondante (s'il y a un mur dans la direction, c'est 0).

— Le nombre de pacgum dans les 4 directions -> inputs égal a nombre de pacgum divisé par cases parcouru dans la direction correspondant (s'il y a un mur dans la direction, c'est 0).

— Le nombre de super-pacgum dans les 4 directions -> inputs égal a nombre de super-pacgum divisé par 4 dans la direction correspondante (s'il y a un mur dans la direction, c'est 0).

Enfin, pour finir sur le système de réseau de neurones, nous avons décidé de faire un système de génération de réseau. On a donc fait une liste de réseaux de neurones et quand le jeu est lancé, on exécute chaque réseau puis on conserve les meilleurs réseaux que l'on sauvegarde dans nos fichiers et on en fait des dérivé en modifiant les poids et biais.

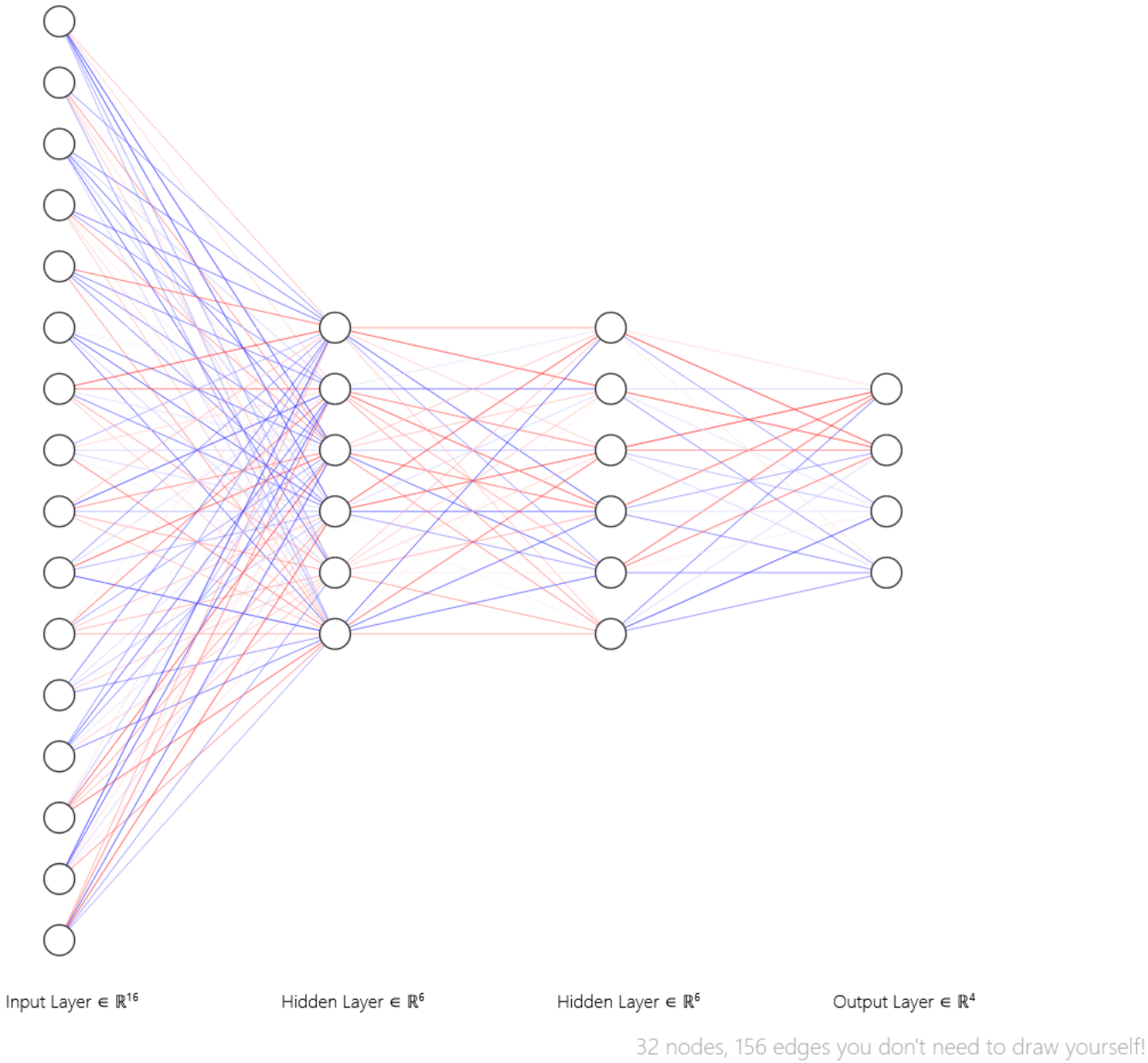


FIGURE 7.1 – Représentation graphique du réseau de neurones

7.2 Le Q-learning

Le Q-learning est une des techniques utilisé pour permettre à une IA de reinforcement learning d'apprendre rapidement. Pour cela, le q-learning va avoir besoin de connaître la récompense que nous allons donner à chaque état que va prendre le jeu. Ainsi, il va pouvoir en déduire une certaine récompense en fonction de chaque état où il se trouve.

Il existe 2 phases pour le Q-learning. La première consiste à de l'exploration. Il va chercher de nouveau chemin la plupart du temps pour savoir s'il y a des chemins qui vont lui donner plus de récompenses au final ou pas. La seconde phase est l'exploitation. Cela correspond à l'utilisation des données qu'il aura accumulé au fur et à mesure de son exploration. En réalité, ces 2 phases s'effectuent en même temps lors de l'exécution du réseau de neurones. En effet, sur les premières générations, le Q-learning va privilégier l'exploration pour trouver de nouvelles récompenses plus élevées. Au contraire, il ne va faire que peu d'exploitation, car les récompenses peuvent facilement être augmentées normalement sauf si nous avons une chance incroyable dès le début. Puis au fur et à mesure que les générations passent il y aura moins d'exploration et passera majoritairement en exploitation des résultats. Il passera alors de 90% d'exploration à 10% tandis que l'exploitation passera de 10% à 90%.

Nous pouvons donc dire que la méthode du Q-learning permet à l'IA de se créer un "dataset" qui lui permet ensuite de prendre la meilleure option en fonction de son état actuel. Le nom exact pour designer le "dataset" créé par le Q-learning est Q-table (cf Figure 7.2). Comme on peut le voir sur l'image ci-dessous plus les valeurs sont élevé plus il est bon d'être dans cet état.

Le problème que nous avons pour le moment est que nous possédons 17 inputs qui peuvent avoir chacun plusieurs états. Ainsi, si nous voulons déclarer absolument tous les états possibles que pourrait prendre le jeu, il nous faudra effectuer une imbrication de 17 'for' pour l'initialisation. Cela est énorme et n'est pas trop envisageable.

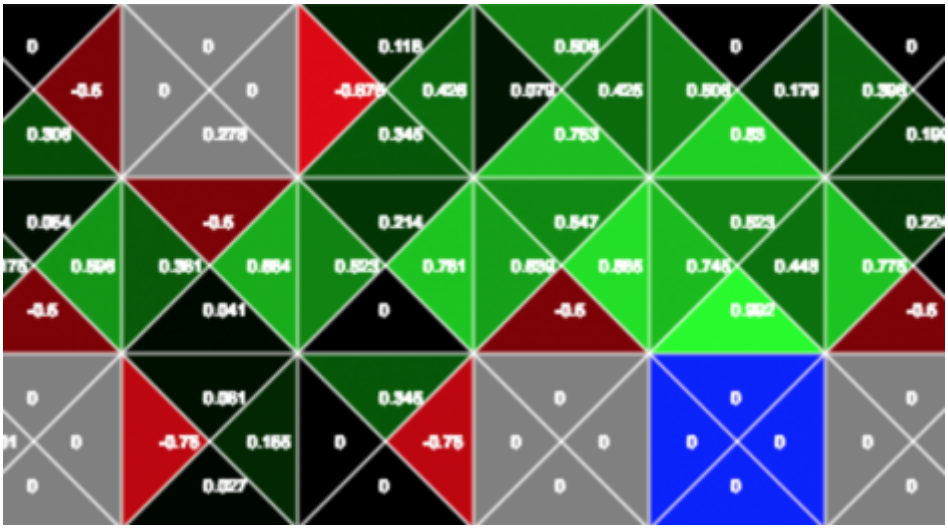


FIGURE 7.2 – Exemple de q-table

8 Les tâches annexes

8.1 Le site web

En ce qui est du site web, nous avons fait quelques améliorations notamment dans la partie projet où nous avons ajouté une timeline qui montre l'avancée du projet depuis le début. De plus, nous avons ajouté du contenu dans la partie 'à propos' et dans la page des téléchargements.

FIGURE 8.1 – Page d'accueil

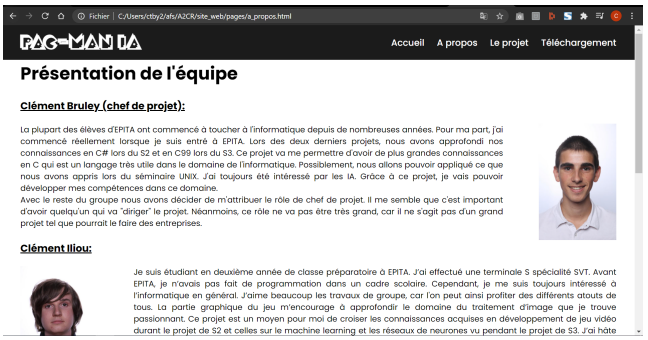


FIGURE 8.2 – À propos de nous

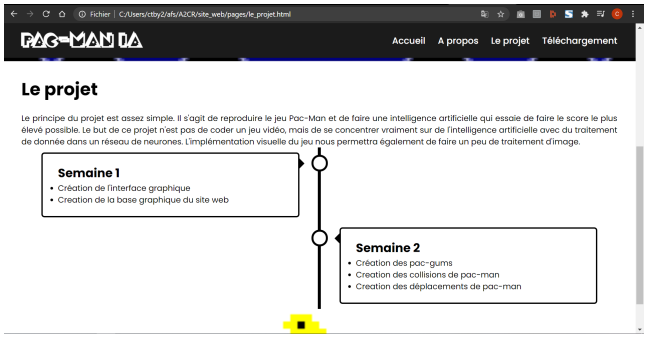


FIGURE 8.3 – Le projet

FIGURE 8.4 – Téléchargement possible

8.2 Gestion du dépôt git

La gestion du dépôt git a été un peu plus mouvementé, car les changements ont été fait principalement sur la branche principale. En effet, lorsque nous avons commencé à avancer à nouveau le projet, nous n'avons pas mis à jour nos branches respectives. Ainsi, nous avons fait la majeur partie des changements sur la même branche.

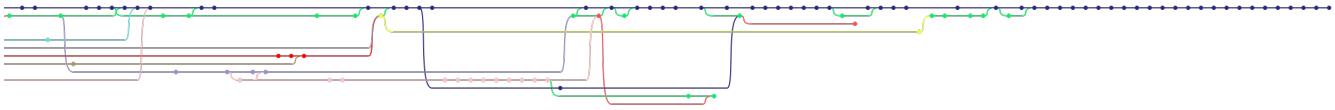


FIGURE 8.5 – Tout les commits que nous avons fait

8.3 LaTeX

Pour pouvoir effectuer le LaTeXcorrectement nous avons préparer la structure du rapport juste après la première soutenance. Ainsi, nous avons pu rédiger au fur et à mesure que nous avancions dans le projet ce qui nous permet de ne pas avoir de rush à faire juste avant la soutenance pour finaliser notre rapport.

9 Plan de soutenance (théorique)

9.1 Tableau d'avancement

	Soutenance 3
Création du visuel	100%
Animation du joueur	100%
Déplacement des fantômes	100%
IA du joueur	100%
Site web	100%

9.2 Détail de l'avancement et des objectifs

9.2.1 Deuxième soutenance :

- Interface graphique du logiciel et finitions terminés.
- Concepts d'IA mis en places : Q-learning, réseaux de neurones modulables ainsi qu'un agent réflexe pour démonstration à la soutenance.
- Ajout du contenu pour le site web.
- Discussion autour de comment faire l'intelligence artificielle.

Soutenance final :

- Avoir une IA fonctionnelle qui fait des gros scores, une IA qui sait prévoir des actions à l'avance et qui optimise ses scores.
- Site web et graphiques finis.

10 Conclusion

Pour conclure, entre la dernière soutenance et cette soutenance, nous avons eu très peu de temps pour avancer. Il faut aussi ajouter qu'il existe tellement de concepts d'intelligence artificielle qu'il y a eu beaucoup de discussions sur le concepts et ce que nous avons choisi. Nous avons espoir que le Q-learning et notre réseau de neurones puissent être la bonne solution. Le problème que nous avons actuellement réside surtout dans quelles données d'entrée fournir au réseau de neurones et comment pouvoir modifier les biais et les poids pour avoir quelque chose de fonctionnel.

Concernant l'équipe autour de ce projet, nous restons tous bien soudé et l'ambiance est vraiment bonne, mais nous constatons une baisse de motivation chez certains membres de l'équipe devant la difficulté que représente l'intelligence artificielle. En effet, derrière la magie des IA se cache des concepts mathématiques complexe et il existe une multitude de concepts d'IA possibles.