

HLIN405
Projets de Programmation de L2
Encadré par Stéphane Bessy

Blob Wars

Allouch Yanis - Roux Jérémie - Villaroya Kévin

2018 - 2019



Table des matières

I	Introduction	3
1	Choix des outils de développement	3
2	Diagramme de Gantt	4
3	Règles du jeu	5
II	Développement	6
1	Présentation du projet	6
2	Objectif	7
3	Structure des principales classes du programme	7
4	Structure globale	8
5	Icône et "blobs"	8
6	Fichier de configuration	9
III	Intelligence artificielle	10
1	IA minmax	10
2	Heuristique	11
3	Élagage alpha-bêta	12
IV	Aperçu du jeu	14
V	Conclusion	16
1	Bilan	16
2	Extension/Perspectives d'avenir	16
VI	Annexe	18
1	Choix du sujet	18
2	Cahier des charges	20
3	Rapports d'entrevue	21
4	Calculs de temps d'exécution de l'algorithme MinMax	30
5	Rapports de version	31

Première partie

Introduction

Dans le cadre de l'UE HLIN405 (Projet de Programmation de L2) nous avons réalisé un Blob Wars¹ (FIGURE 1). Notre groupe est composé de trois membres : Allouch Yanis, Roux Jérémie et Villaroya Kévin. Pour plus d'informations par rapport au choix du sujet, veuillez vous rendre dans l'Annexe.



FIGURE 1 – Exemple d'un Blob Wars

1 Choix des outils de développement

Pour permettre un bon travail d'équipe, nous utilisons un espace de travail collaboratif GitLab^{2 3 4} (FIGURE 2 en page 4) afin de nous organiser plus facilement en nous répartissant le travail. Cela permet de sécuriser l'avancée de notre travail en faisant des sauvegardes régulières depuis différents ordinateurs et dans différents lieux.

Nous gardons également une trace des tâches qu'il nous reste à accomplir et que nous avons déjà accompli grâce à un diagramme de Gantt. Notre espace GitLab est hébergé par l'Université de Montpellier et est accessible par l'ensemble des membres du projet qui peuvent y faire des modifications.

1. Exemple d'un Blob Wars : <https://www.twoplayergames.org/Blob-Wars/2.html>
2. Site internet de GitLab : <https://about.gitlab.com/>
3. Page Wikipédia de GitLab : https://fr.wikipedia.org/wiki/GitLab_CE
4. Lien vers le GitLab du projet : https://gitlab.info-ufr.univ-montp2.fr/e20170000656/Blob_Wars_HLIN405



FIGURE 2 – Logo du système de gestion GitLab

Le Blob Wars a été codé en Java 8 conjointement avec la librairie Slick⁵ sur l'environnement de développement intégré Eclipse (FIGURE 3). Notre projet est suivi par M. Stéphane Bessy⁶ que nous remercions pour son soutien, ses conseils et sa présence lors de nos réunions bimensuelles⁷. M. Bessy est maître de conférences en informatique à l' Université de Montpellier et enseignant au département info de la Faculté des Sciences. Il est membre de l'équipe ALgorithmes de Graphes et Combinatoire (AlGCo) du LIRMM.



FIGURE 3 – Logo de l'environnement de développement intégré (IDE) Eclipse

2 Diagramme de Gantt

Nous avons conçu un diagramme de Gantt (FIGURE 4) afin de nous organiser dans le développement, mieux gérer le temps disponible et les échéances.

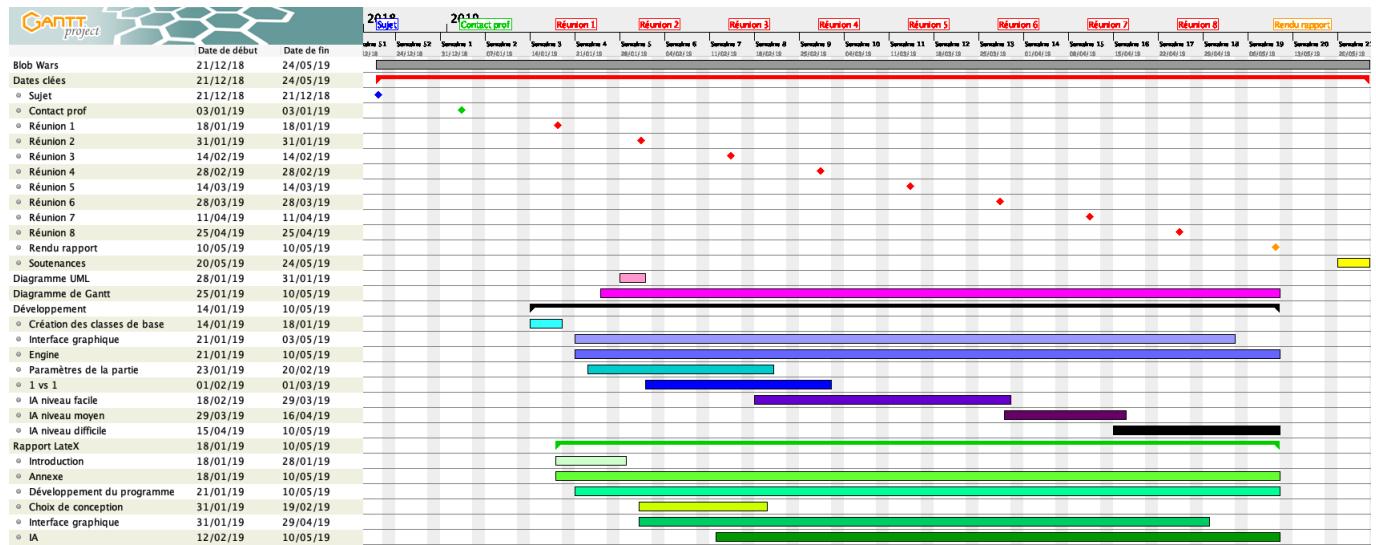


FIGURE 4 – Diagramme de Gantt initial

-
5. Slick2D est une version arrangée pour développer des jeux en 2D facilement
 6. Toutes les informations le concernant se trouvent sur <https://www.lirmm.fr/~bessy/>
 7. Compte-rendu de chaque réunion en annexe

3 Règles du jeu

Objectif : Conquérir le plateau de jeu en attaquant les pions ennemis en les convertissant en vos pions.

Comment jouer : Vous pouvez à chaque tour faire apparaître un pion ou déplacer un de vos pions déjà présent.

- Sélectionner un pion de votre équipe avec votre souris, le pion choisi sera mis en surbrillance et les déplacements possibles aussi (en nuances de gris)
 - Sélectionner une case gris foncé (adjacente à votre pion) pour le dupliquer sur cette case
 - Sélectionner une case gris clair (adjacente à une case adjacente à un de vos pion) pour le déplacer sur cette case
- Chaque pion ennemi adjacent à la case sur laquelle vous vous êtes déplacé ou dupliqué sera converti en votre pion

Comment gagner : Le gagnant est celui qui a le plus de pion à la fin de la partie. Convertir tous les pions du plateau est également une condition de victoire.

Fin de la partie : Elle intervient une fois que le plateau est complet ou quand $n - 1$ joueurs ne peuvent plus jouer (soit n le nombre de joueurs au total).

Egalité : Il y a égalité lorsqu'à la fin de la partie plusieurs joueurs ont le (même) meilleur score.

Passage automatique du tour : Lorsque vous ne pouvez effectuer aucun mouvement.

Plus de pion sur le plateau : Vous ne pourrez plus jouer, la partie s'arrête là pour vous.

Equipes : Le jeu se joue aussi par équipes de 1 à 3 joueurs (2 équipes minimum et 4 équipes maximum puisque 2 joueurs minimum et 4 joueurs maximum dans une partie) :

- Un joueur qui n'est pas dans votre équipe est considéré comme un ennemi.
- Un joueur de votre équipe est considéré comme un allié dont les pions ne peuvent ni être contrôlés ni être convertis.

Deuxième partie

Développement

1 Présentation du projet

Le Blob Wars est un jeu-vidéo de type réflexion qui implique une grande part de programmation "appllicative" qui est à ce jour la plus utilisée pour développer (sans mentionner l'utilisation des frameworks). Les langages qui prennent en compte ce paradigme de programmation sont le Python, le C, le C++, le C#, le VB, le JS, etc. Tout comme il existe plusieurs langages, il existe aussi différentes façons de concevoir une "application". Grâce à notre cursus universitaire nous avons abordé différents pans de la programmation :

- impérative
- orientée objet
- parallèle
- fonctionnelle
- événementielle
- web

Les différents langages associés à ces différents paradigmes⁸ sont favorisés dans le choix du développement d'une application.

Le Blob Wars est un jeu de réflexion. Un jeu est un enchaînement d'événements qui peuvent influer certains objets du jeu qui vont à leur tour relancer un événement et ainsi de suite jusqu'à ce qu'un état final soit atteint. Chaque événement peut être modélisé comme une suite d'étapes qui sont associées à un comportement, à une entrée et qui *ne change pas*⁹. Pour le dire autrement, on décrit un algorithme qui permet de traiter un état donné. Si on résume les différents point de notre projet, on y retrouve :

- Des objets
- Des événements
- De l'impératif/procédural

Voici un tableau (TABLE 1) qui regroupe les pour et contre en fonction des langages qui nous sont connus¹⁰ et de nos critères.

Langages	Objets	Events	Utiliser
C	-	NA	+
C++	+	+	+
C#	+	+	-
Java	+	+	+
JavaScript	+	+	+
PHP	+	+	+
Python	+	+	+
Ruby	+	+	-
Perl	+	+	-

TABLE 1 – Comparatif des langages possibles

Au moment du choix du langage, une application web n'était pas considéré à cause de la taille du jeu. Ce qui ressort c'est donc C++, Java et Python. Cependant, nous avons décidé de commencer à programmer en Java car nous l'étudions au même moment. De plus, Java permet de construire du code structuré et facilement modifiable.

8. Certains paradigmes sont imbriqués les uns aux autres, on distingue des "niveaux"

9. Sauf si c'est son objectif...

10. Que ce soit de nom ou par une utilisation pratique

2 Objectif

Le but principal de ce sujet de TER est l'intégration d'une Intelligence Artificielle (IA). On va implémenter à cette IA des algorithmes comme notamment :

1. le MinMax (parcourt d'un arbre et choix de la branche la plus optimale)
2. l'Alpha-Bêta (élagage de l'arbre parcouru par le MinMax et diminution du temps de calcul)

3 Structure des principales classes du programme

3.1 Dossier *Slick*

Il s'agit ici de la gestion de l'interface graphique. Toutes ces classes sont des éléments affichés.

— **Menu :**

⇒ Contient des boutons qui permettent l'accès au lancement du jeu, l'aide, les crédits, les options ou encore quitter.

— **Options :**

⇒ Contiendra la possibilité de changer le volume de la musique et des bruitages
⇒ Boutons pour quitter le jeu et y revenir

— **Game :**

⇒ Affiche les scores à gauche, individuels et par équipe ainsi que le plateau du jeu

— **Crédit :**

⇒ Affiche un rectangle contenant toutes les informations concernant la création du projet

— **Aide :**

⇒ Fenêtre très similaire à celle de Crédit, mais qui affiche des informations concernant l'utilisation de ce programme

— **Bouton :**

⇒ Facilite l'ajout de boutons dans une fenêtre

— **Slide :**

⇒ Facilite l'ajout de "slide" dans une fenêtre (la barre de volume par exemple)

3.2 Dossier *Engine*

Classe qui permet les autres calculs graphiques liés au programme

Case : Une case contient sa position, sait si un joueur l'occupe et si elle est accessible.

Damier : Classe contenant un tableau de deux dimensions de cases, c'est donc le jeu

Joueur : Une classe qui contient les informations du joueur, son nom, son score, son équipe,...

IA : Une classe qui renvoie le prochain coup à jouer pour un joueur et un état de jeu donné.

4 Structure globale

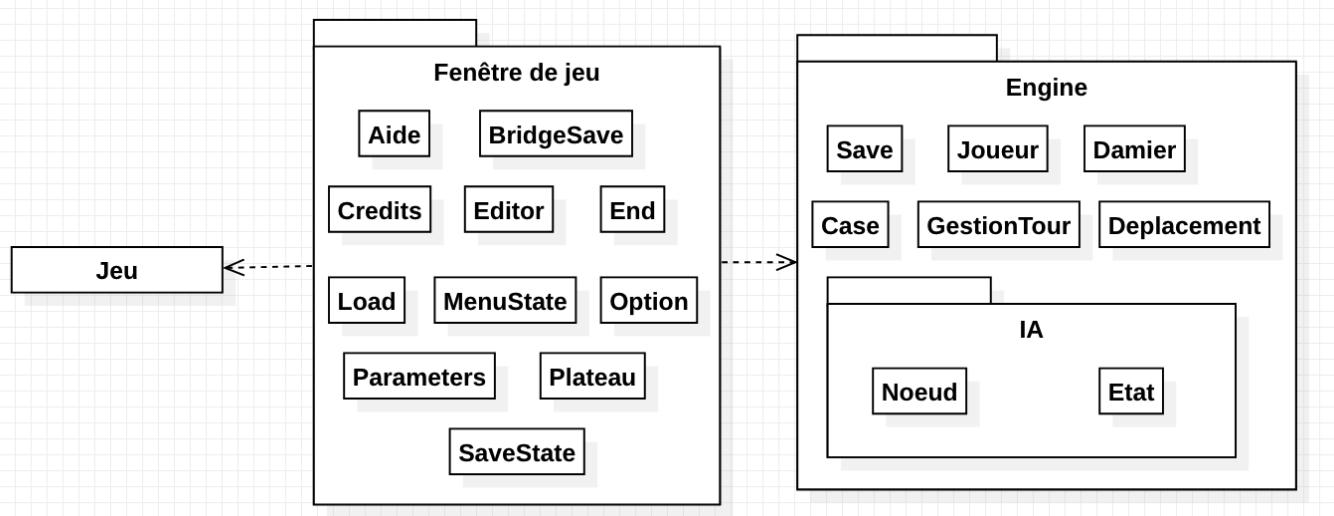


FIGURE 5 – Diagramme des classes simplifié global

La structure de notre jeu Blob Wars est composée d'un fenêtre de jeu (interface graphique) qui dialogue avec le moteur du jeu ainsi que le jeu (voir FIGURE 5).

5 Icône et "blobs"

Nous avons créé une icône pour l'application (voir FIGURE 6). Dans notre version du Blob Wars, les "blobs" sont des logo moteurs de recherche ou de systèmes d'exploitations.



FIGURE 6 – Icône de l'application Blob Wars

6 Fichier de configuration

L'utilisateur a la possibilité de générer et charger des fichiers de configuration afin de sauvegarder ou charger des parties. Chaque fichier de configuration représente l'état d'une partie. Les fichiers sont générés de la manière suivante (voir FIGURES 7 et 8) :

La date et l'heure : Sur une ligne (ici l.1)

Le nombre de joueurs : Sur une ligne (ici l.3)

Les propriétés des joueurs : Sur autant de lignes qu'il y a des joueurs. Pour chaque joueur il y a son index, son niveau (s'il s'agit d'une IA) ou la mention joueur (si c'est un joueur), son nom, son équipe, son avatar et l'index de son avatar (ici l.5 à 8)

L'index du joueur qui doit jouer : Sur une ligne (ici l.9)

La taille du plateau de jeu : Sur une ligne (ici l.11)

Le plateau : Sur une ligne. Chaque numéro correspond à l'index du joueur. Le chiffre zéro représente une case vide et la lettre X une case inaccessible (avec des murs). (ici l.13 à 20).

```
1 2019/04/02 00:23:58
2
3 4
4
5 1 tres_facile Joe red firefox 2
6 2 joueur William blue ie 5
7 3 moyen Jack green windows 6
8 4 joueur Averell yellow chrome 4
9 2
10
11 8
12
13 11000400
14 11004400
15 100XX000
16 30XXXX02
17 00XXXX20
18 000XX020
19 30000000
20 30000000
```

FIGURE 7 – Exemple d'un fichier de configuration

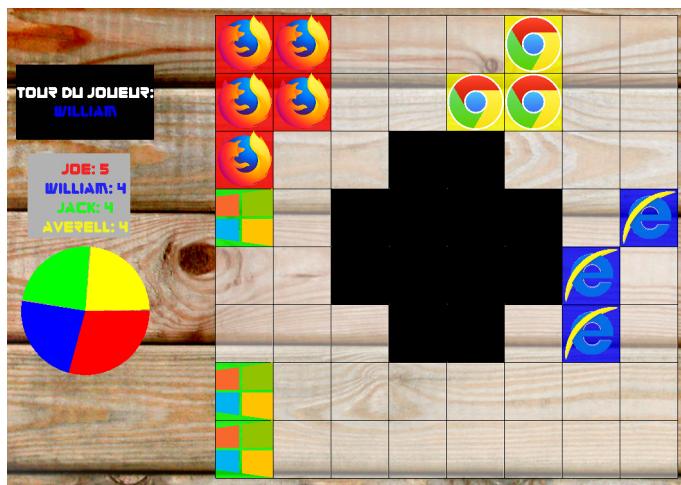


FIGURE 8 – Capture d'écran de la partie correspondant au fichier de configuration de la FIGURE 6

Troisième partie

Intelligence artificielle

1 IA minmax

1.1 Fonctionnement de la classe

La classe IA est un peu spéciale car elle ne devient une instance qu'une seule fois^{11 12} ! A profondeur égale, une IA va agir toujours de la même façon face à une situation de jeu donnée. Elle contient les méthodes nécessaires au calcul du prochain coup.

1.2 Méthodes de la classe IA

eval : Permet à partir d'un état de jeu et d'un joueur de lui attribuer un chiffre qui correspond à s'il est loin ou proche de la victoire.

simulerCoupDePosition : Permet à partir d'une case de jeu de fournir tous les coups que le pion situé au-dessus peut effectuer, donc les cases accessibles autour de cette même case.

simulerTousLesCoups : Chaque état de jeu contient les positions des joueurs, il suffit pour chacune de leur position de calculer leurs coups possibles.

min et max : Deux fonctions qui à partir d'une liste d'état, nous renvoie soit le meilleur état pour le joueur dans le cas de max, soit le pire dans le cas de min. (On peut penser à mettre une fonction de randomisation lorsque deux choix sont possibles afin de faire varier les coups)

minimax : Renvoie le coup que le joueur doit effectuer pour faire le meilleur coup calculé en créant l'arbre des possibilités par récursion et le remontant en appliquant le max ou min si c'est le tour d'un ennemi ou d'un allié.

Note : Le MinMax n'est pas assez optimal en temps de calcul et nous utiliserons donc l'élagage alpha-bêta du MinMax

1.3 Implémentation de l'algorithme MinMax

Voici le pseudo-code produit à partir de notre implémentation du minimax (**Algorithme 1**).

Le cas de base est si on se trouve sur une feuille ici identifiée par **profondeur >= 0 et JeuPasFini**. Dans les autres cas, on identifie si c'est notre tour. Si c'est le cas, on va calculer le coup qui nous fait gagner le plus de points¹³. Sinon on cherche à calculer le meilleur coup pour l'adversaire c'est à dire en d'autre mot le minimum.

Cette opération est répétée pour toutes les cases autour de notre pion et on simule autant de coup pour chacune de ces cases là que de profondeur choisies pour notre algorithme. Cela nous donne une complexité en $\Theta(n^p)$ avec n le nombre de coups possibles pour une position et p la profondeur de calcul.

Pour expliquer cela, un schéma est le moyen le plus adapté. Au lieu d'en refaire un, nous utilisons celui disponible sur la page Wikipédia de l'alpha-bêta qui est très bien fait et que vous pourrez retrouver sur la FIGURE 9 en page 12 dont il suffit simplement d'ignorer les élagages. Vous trouverez en Annexe plusieurs courbes de temps de parcours du MinMax dans lesquelles on fait varier le nombre de joueurs, la profondeur des IA et la taille de la zone de jeu.

11. Ce que l'on appelle un singleton

12. Cela garantie qu'une unique instance d'une classe donné sera créée et offre un point d'accès universel à cette instance.

13. en fonction d'une fonction d'évaluation ici appelée **calculerValeurEtatJeu**

Algorithme 1 : *minMax (joueur : entier, gestionTour : GestionTour, noeudCourant : Noeud, noeudPere : Noeud) : Etat*

Variables : *monTour? : booléen, joueurActuel : entier*
monTour? ← joueurActuel(gestionTour) = joueur;
tourEquipe? ← estAllieA(joueur);
simulerTouslesCoups();
profondeur ← profondeur - 1;
tourSuivant();

si *profondeur >= 0 et JeuPasFini alors*

pour *i de 0 à nbNoeud faire*

si *minMax(joueur, gestionTour, noeudCourant, noeudCourant) = null alors*

FinDuCalcul;

fin

fin

si *monTour? ou tourEquipe? alors*

noeudChoisi ← maxDeTousLesNoeuds;

noeudCourant ← noeudChoisi;

si *noeudPere ≠ null alors*

noeudPere.alpha ← max(noeudChoisi, noeudCourant.alpha);

fin

si *noeudCourant.beta ≤ noeudCourant.alpha alors* // Elagage alpha/beta

renvoyer null;

fin

sinon

noeudChoisi ← minDeTousLesNoeuds;

noeudCourant ← noeudChoisi;

si *noeudPere ≠ null alors*

noeudPere.beta ← min(noeudChoisi, noeudCourant.beta);

fin

si *noeudCourant.beta ≤ noeudCourant.alpha alors* // Elagage alpha/beta

renvoyer null;

fin

fin

sinon

noeudCourant ← calculerValeurEtatJeu(joueur);

fin

renvoyer *noeudChoisi;*

2 Heuristique

Dans cette section nous analysons empiriquement et sans outils de précision les données mesurées lors de nos phases de test pendant le développement.

- Temps moyen du tour pour un joueur humain : quelques dizaines de secondes.
- Temps moyen de l'algorithme sur un plateau 6x6 jusqu'à 10x10 : de quelques secondes à quelques minutes sans élagage du MinMax, de quelques secondes à quelques dizaines de secondes avec élagage

Des tests plus détaillés et plus précis sont disponibles en Annexe page 30. Ils montrent le besoin de rapidement optimiser cet algorithme.

Algorithme	profondeur	quelques sec.	dizaines de sec.	quelques min.	infinie
MinMax	1	x	x		
MinMax	2	x	x		
MinMax	3		x	x	
MinMax	4		x	x	
MinMax	5			x	x
MinMax	6				x

TABLE 2 – Observation du temps de calcul de chaque coup pour chaque algorithme implémenté

3 Élagage alpha-bêta

Comment optimiser l'algorithme MinMax ? C'est la question que certains informaticiens se sont posées et dont ils ont trouvé la solution.

3.1 Compréhension de cet élagage

L'objectif de cette optimisation est de couper (ou élaguer) les branches de notre arbre pour avoir moins de contenu à explorer. Qui dit moins de contenu à explorer dit moins de calculs à effectuer. Cette opération est communément appelé un élagage. Cet élagage est très utilisé car il permet de n'avoir aucune perte d'information (et ainsi perdre aucun meilleur coup) tout en diminuant le nombre de calculs à effectuer.

3.2 Illustration de son fonctionnement sur un exemple

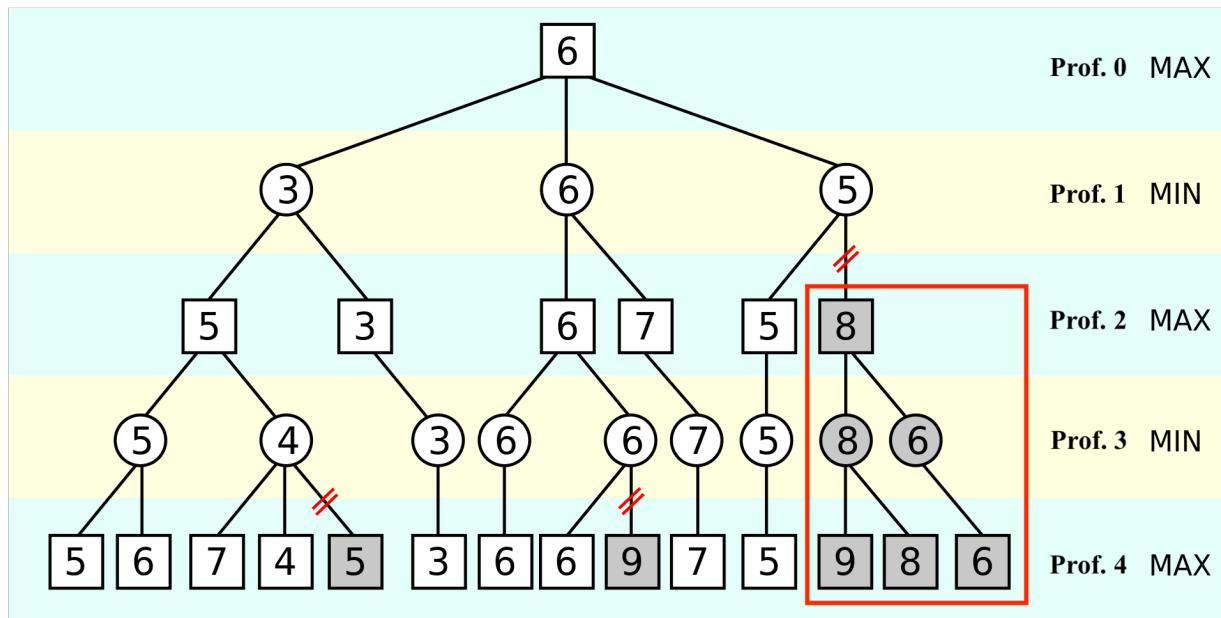


FIGURE 9 – Exemple d'un arbre étiqueté avec les valeurs d'un MinMax avec élagage alpha-bêta

Sur la FIGURE 9, on constate que plusieurs coupures ont été réalisées. On étudie l'élagage du sous-arbre encadré en rouge. Le noeud MIN (profondeur 1) vient de mettre à jour sa valeur courante à 5. Celle-ci, qui ne peut que baisser, est déjà inférieure à $\alpha = 6$, la valeur actuelle du noeud MAX précédent (profondeur 0). Ce noeud MAX cherchant la valeur la plus grande possible, il ne la choisira donc pas de toute façon. On peut donc stopper le parcours du MinMax en élaguant avec l'implémentation alpha-bêta du MinMax le sous-arbre encadré en rouge.

3.3 Calculs de temps d'exécution à partir de l'implémentation de l'alpha-bêta

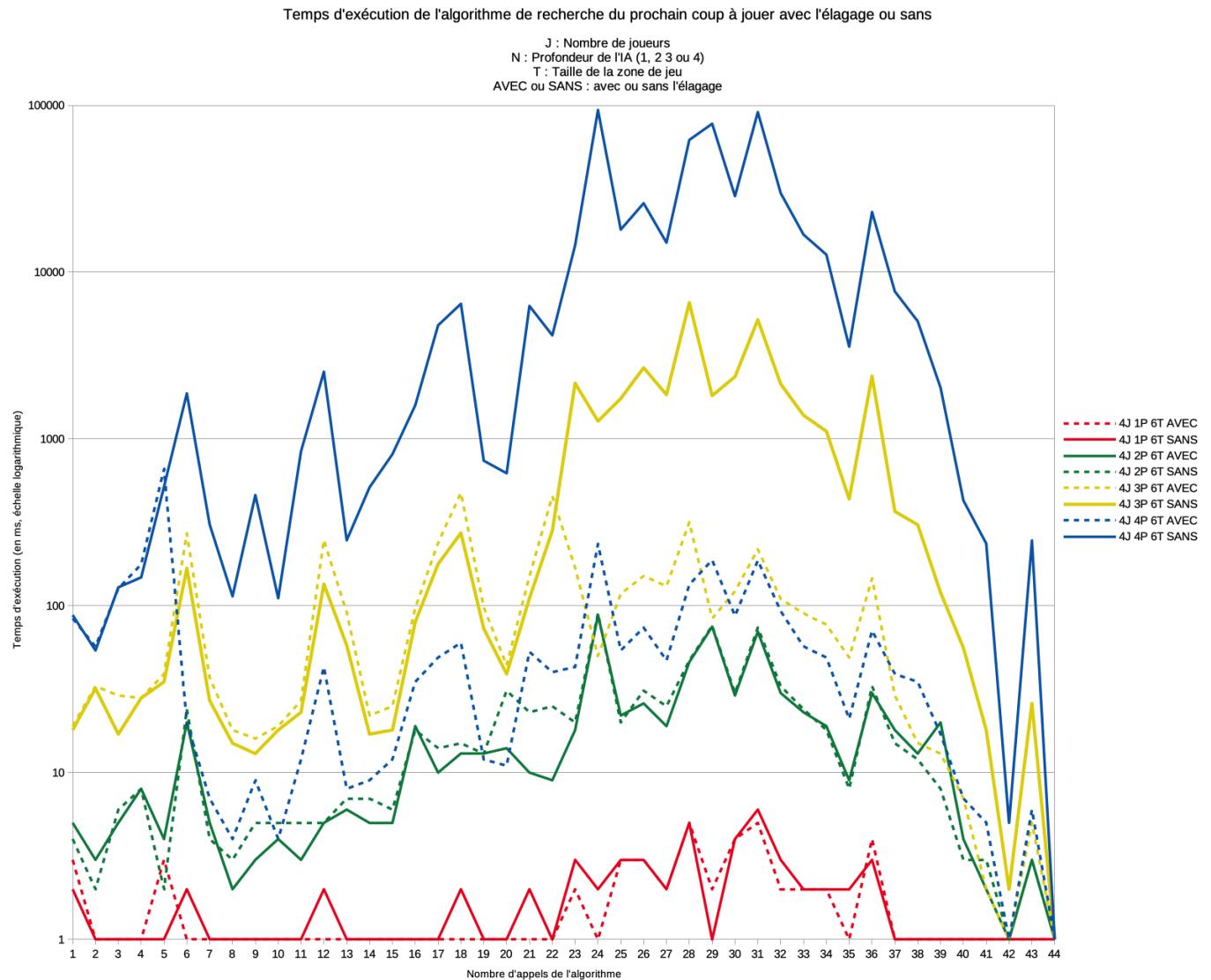


FIGURE 10 – Exécution de l'algorithme MinMax avec élagage alpha-bêta ou non

On exécute le programme pour 4 joueurs et une taille de terrain de 6 cases de côté. On fait varier la profondeur de l'IA de 1 à 4 et on active (courbe en pointillé) ou pas (courbe avec trait continu) l'élagage. Les résultats sont rassemblés sur la FIGURE 10. On constate que :

- Pour les profondeurs 1 et 2 (courbes rouges et vertes), l'élagage ne fait quasiment pas varier le temps d'exécution de l'algorithme par rapport aux courbes sans l'élagage de ces profondeurs.
- Pour la profondeur 3 (courbes jaunes), l'élagage fait gagner un peu de temps d'exécution notamment au moment de l'explosion de calcul (autour de l'appel numéro 22 de la courbe jaune continue) dû au passage à 2 joueurs.
- Pour la profondeur 4 (courbes bleues), l'élagage est encore plus bénéfique en temps de calcul que précédemment. Le temps d'exécution à cette profondeur est voisin des courbes de profondeur 3.
- Suite à d'autres tests, on a constaté que le gain de temps de temps par l'élagage est d'autant plus important que la profondeur de l'IA est importante.

Quatrième partie

Aperçu du jeu

Voici maintenant une présentation des principales fenêtres de l'application ainsi que de leur utilité.

Sur la FIGURE 11, voici une capture d'écran du menu de création d'une partie. On peut définir le nom et le nombre de joueurs ou IA (et leur niveau). Il y a également possibilité de changer la taille de la carte et jouer sur des cartes ayant des murs ou pas. Il existe également un éditeur de carte (Figure 14) accessible depuis ce menu.

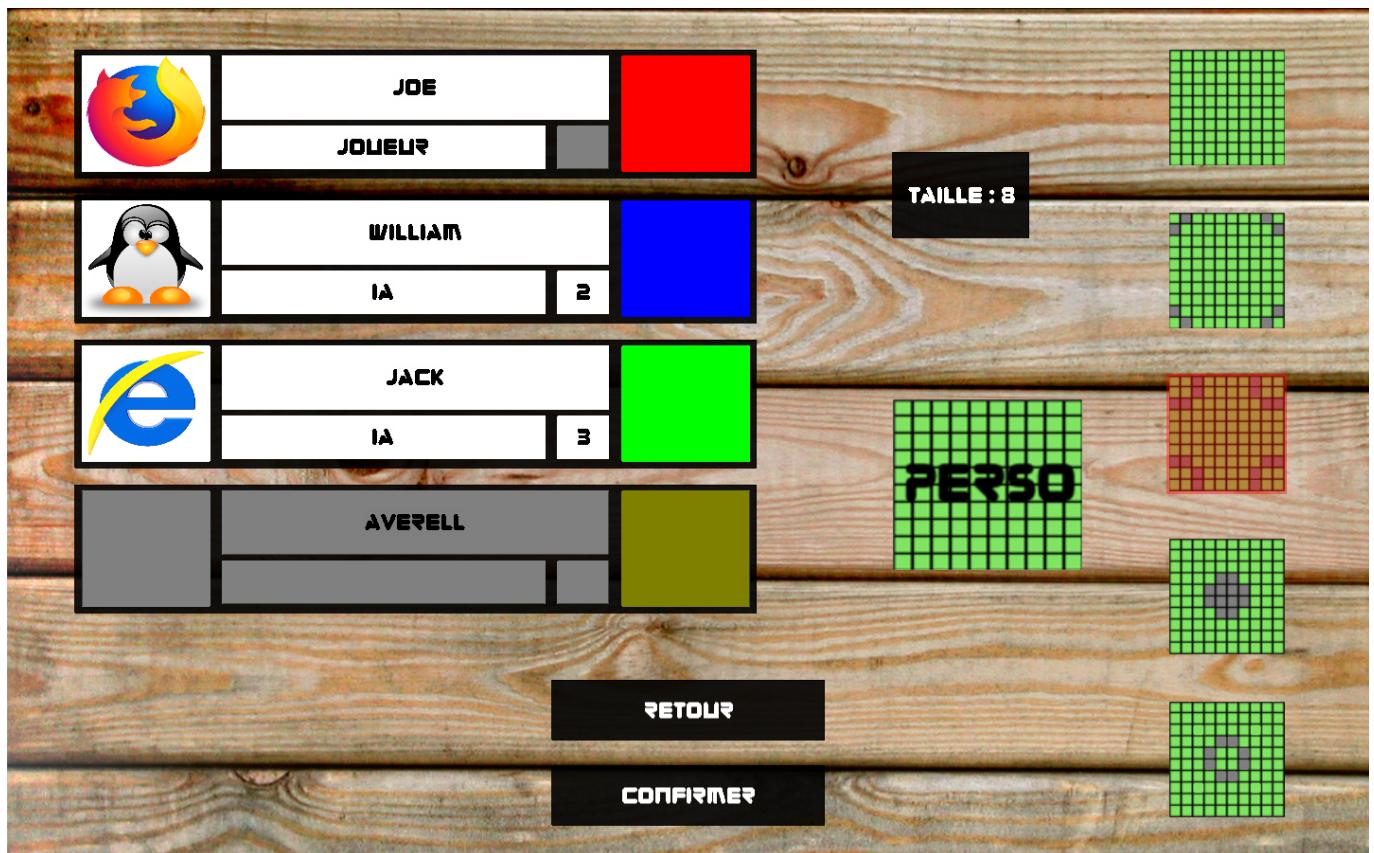


FIGURE 11 – Création d'une partie

Les FIGURES 12 à 17 sont situées sur la page suivante.

La FIGURE 12 est une capture d'écran du menu principal depuis lequel on a accès au bouton JOUER qui permet de créer ou charger une partie (FIGURE 14). Le bouton AIDE dirige vers une fenêtre qui explique comment utiliser l'application. Le bouton OPTION permet d'accéder au menu présenté en FIGURE 17 pour les options sonores. Le bouton CREDITS permet d'accéder aux crédits et le bouton QUITTER permet de quitter l'application.

Pendant une partie (FIGURE 16) on peut accéder au menu pause (FIGURE 17) via la touche Echap du clavier afin de changer le volume des différents effets sonores (réalisés par nos soins). Lorsque la partie est terminée, on peut accéder au menu principal du jeu (FIGURE 17).



FIGURE 12 – Menu principal

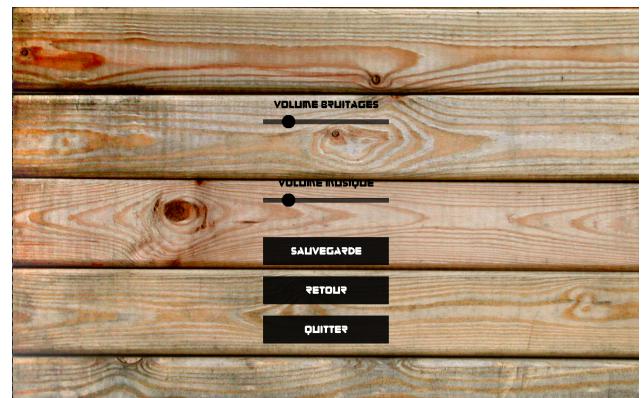


FIGURE 13 – Options sonores



FIGURE 14 – Chargement et sauvegarde de parties

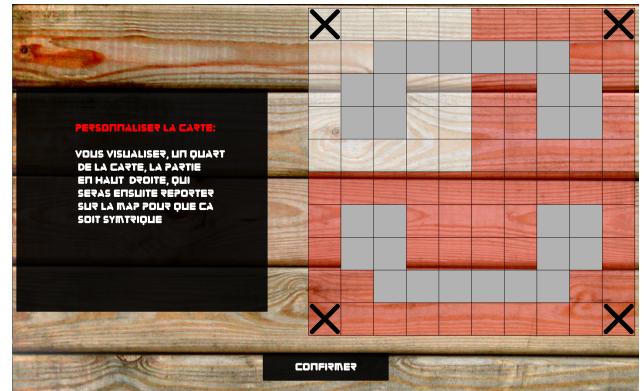


FIGURE 15 – Éditeur de carte

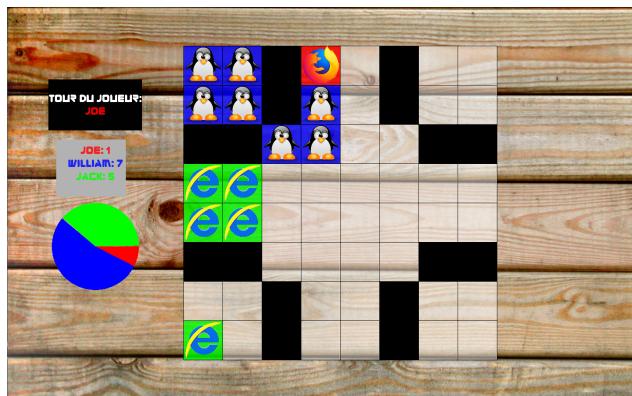


FIGURE 16 – Pendant une partie



FIGURE 17 – Fin d'une partie

Cinquième partie

Conclusion

Le principal rôle du projet de programmation est d'aussi bien nous faire réfléchir à l'organisation de notre groupe en utilisant un diagramme de Gantt que de savoir utiliser des outils de travail comme Git, StarUML ou Eclipse, pour développer un jeu-vidéo dans lequel il faut implémenter une IA.

Cette phase de conception (de pré-traitement) prends du temps pendant lequel nous réfléchissons ensemble sans toucher à un ordinateur. C'est une étape importante dans notre projet, cela en devient de plus en plus clair aujourd'hui.

Lors de cette phase de pré-traitement, nous avons décidé de choisir Jérémie comme porte-parole pour les interactions avec notre encadrant. Cette phase consistait à faire un état de l'art de ce qui existait déjà, c'est à dire quelles technologies ou langages sont utilisés pour faire un jeu-vidéo.

A la suite de cela, nous avions un cahier des charges, un langage, un IDE et nous nous sommes créé du jour 1 jusqu'au 10 mai un calendrier avec les différentes étapes de développement, d'analyses. Nous nous sommes confrontés à une situation qui est survenue plus tard lors de la phase de développement qui est que Kévin était le seul à coder la majeure partie du programme¹⁴.

Cela dit, les étapes de réflexion et de travail étaient communes et nous discutions constamment des points à améliorer, à changer, etc.

1 Bilan

Pour résumer, nous avons développé un jeu-vidéo de type Othello appelé Blob Wars. Ce jeu-vidéo suit les règles précédemment énoncées en page 5 et nous y avons ajouté la possibilité de jouer contre une IA qui suit l'algorithme du MinMax (voir page 11) que nous avons rapidement dû optimiser en ce que l'on appelle l'élagage alpha-bêta (voir page 12). En effet, on arrivait à des temps de calculs infinis pour nos machines pour une profondeur supérieure à 4 ce qui était d'autant plus mauvais si on jouait contre une autre IA.

D'autre part, nous avions expliqué lors d'une réunion avec M. Bessy comment nous avions utilisé le logiciel Sothink SWF Decompiler pour récupérer le jeu du site donnée dans le sujet et qui nous a montré que leur IA du jeu Blob Wars en ligne était basée sur du backtracking (voir la réunion du 18 mars page 25).

2 Extension/Perspectives d'avenir

En date du 7 mai, voici une liste non-exhaustive de ce qui nous souhaiterions implémenter en plus que ce qui avait été demandé (avec les difficultés et possibilités que cela ouvrirait si ces implantations voyaient le jour)

— Un développement orienté web

1. Une difficulté sur laquelle nous nous sommes mis d'accord réside dans le fait que le code que nous avons ne peut pas simplement être transposé dans un des langages du web que nous avons précédemment vu, la conception du programme devant être revue.
2. Cependant cela serait fait avec l'objectif de pouvoir jouer en multijoueurs, de pouvoir faire du monitoring par exemple en ouvrant plusieurs instances du jeu.

— La possibilité de changer, de customiser l'apparence du jeu.

14. Voir les logs du dépôt

- Ajouter des animations pour les pions.
- L'ajout de threads dans le programme actuel.
 - Pourquoi les threads ? Pour pouvoir sauvegarder une partie pendant que les programmes calculent les différents noeuds par exemple.
 - Cependant nous n'avons pas bien identifié les difficultés éventuelles que cela pourrait induire sur notre programme
- Un Bot qui pourrait jouer avec notre IA et contre une IA d'un autre Blob Wars.
- D'autres fonctions d'évaluations
- D'autres algorithmes

Ces implémentations ne sont pas anodines. Par exemple la possibilité de sauvegarder et reprendre une partie est intéressant sur deux points de vues. La premier étant direct, le deuxième pour observer le comportement de notre algorithme à partir d'un moment précis dans le jeu sans avoir à calculer les états précédents.

Pour en finir avec l'intelligence artificielle, nous avons utilisé le minimax qui est très simple dans sa façon de procéder et qui de notre point de vue peut être à peine être qualifié d'**intelligent** pour ce qui est de sa façon de jouer.

C'est pourquoi on pourrait se tourner vers une IA de type réseaux de neurones qui serait la façon ultime pour une IA de battre un joueur humain.

Sixième partie

Annexe

1 Choix du sujet

Avant de présenter les différentes phases de développement, voici comment notre choix de sujet s'est porté sur Blob Wars et la manière dont nous avons pris contact avec l'enseignant.

C'est l'histoire de 3 étudiants qui se connaissaient depuis quelques mois, aux passés et objectifs de carrières différents mais avec le même point commun d'être chacun passionné par certains domaines de l'informatique :

- programmation et développement
 - algorithmique
 - infrastructures et réseaux
 - bases de données

Autant dire que le choix d'un sujet était très compliqué tant les passions étaient diverses. Notre groupe s'est constitué dès le mois de novembre 2018 : nous avions commencé à nous connaître un peu plus et savoir ce qui nous intéressait vraiment dans les différents pans de l'informatique. À ce moment là, nous nous sommes rendu-compte que nous avions des avis différents sur le choix du sujet :

- Kévin adorait le Java et avait déjà de l'expérience dans le développement de jeux vidéos
 - Jérémie et Yanis avaient une préférence pour les projets impliquant plus de théorie algorithmique

NUMERO	SUJET	Jérémie	Yanis	Kévin	SOMME	#
#17	Sudoku en réalité augmentée	6	5	6	17	
#02	Blob Wars	5	4	4	13	
#19	Kakuro	5	2	5	12	
#23	Résolution de Picross	6	2	4	12	
#10	Algorithme de Aho-Corasick	3	6	2	11	
#07	Résolution d'un casse-tête Nurikabe	4	1	4	9	
#18	Quarto	3	1	5	9	
#06	ASCII art	4	4	1	9	
#16	Dancing with Donald	4	3	2	9	
#11	Implémentation du Jeu Othello-reversi	2	3	3	8	
#24	Utilisation du moteur d'échecs Stockfish sur une page web	4	2	2	8	
#32	Affichage d'un arbre binaire en mode console	3	3	2	8	
#15	Tri Linéaire	1	2	3	6	
#30	Machine Learning (1)	2	1	3	6	
#31	Machine Learning (2)	2	1	3	6	
#03	Visualisation de fractales	0	2	3	5	
#04	Terre et Lune	0	2	3	5	
#08	Résolution d'un casse-tête Slither Link	2	2	1	5	
#20	Enrichissement de maquettes pédagogiques	1	3	1	5	
#26	Implémentation d'un index de type arbre B+	1	2	2	5	
#27	Implémentation d'un moteur de requêtes JSON	1	2	2	5	
#01	De quelles humeurs sont mes tweets aujourd'hui ?	1	2	1	4	
#09	Ranger les villes	1	3	0	4	
#25	Implémentation d'un moteur de requêtes SQL simples	0	4	0	4	
#28	Détection de conflits entre requêtes et mises à jour	0	3	1	4	
#33	Complexité	1	3	0	4	
#12	Explorateur de dossiers	1	2	0	3	
#14	Sur le jeux FANORANA	1	1	1	3	
#29	Implémentation d'un moteur de requêtes en étoile sur des graphes de données	2	1	0	3	
#05	Gabarit et surfaces de révolution	0	1	1	2	
#13	Monopoly revisité	0	1	1	2	
#21	Tag clouds	1	1	0	2	
#22	Aggrégation de données	1	0	1	2	
	0	NON, plutôt mourir				
	1	NON, ça craint				
	2	NON, mais à discuter				
	3	Pourquoi pas				
	4	OUI, ça me tente				
	5	OUI, carrément				
	6	OUI, totalement				

FIGURE 18 – Tableur regroupant nos choix de sujets parmi ceux proposés

Heureusement, c'est là que Jérémie a eu la bonne idée de se servir d'un tableau qu'il a conçu pour l'occasion (Figure 18). Cela a permis de rationaliser le choix des sujets et de leur hiérarchie. Plus important encore, vous observerez que le sujet Blob Wars était notre second choix après le sujet de résolution de sudoku en réalité augmentée mais qu'il était en moyenne apprécié par tout le monde. Jérémie nous a aussi montré qu'il avait la capacité à diriger et superviser un groupe (par exemple en s'occupant de faire les démarches de création de groupe, etc.). Il a donc été naturel qu'il soit le représentant du groupe.

Arrivés à la fin du mois de décembre et une fois le sujet attribué, nous avons pris contact une première fois par mail avec M. Bessy et avons convenus d'un premier rendez-vous le 18 janvier 2019. À ce jour, nous avons gardé un rythme d'une réunion toutes les deux semaines ce qui nous permet d'avancer convenablement et de ne pas rentrer en conflit avec notre cursus universitaire.

2 Cahier des charges

2.1 Premier RDV

- UML
- Gantt
- Répartition des tâches
- Rapport

2.2 Deuxième RDV

- Code minimal défini dans l'UML
- Prototype de jeu fonctionnel
- Save
- Rapport

2.3 Troisième RDV

- Implémentation Minimax
- Implémentation Alpha-Bêta
- Load
- Rapport

3 Rapports d'entrevue

3.1 18 Janvier 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (11 h 08 - 11 h 57)

- Quid du langage le plus adapté ? Deux choix principaux avec C++ et Java avec respectivement les librairies SFML et Slick.
- Rappel du fonctionnement du jour de la soutenance orale avec un jury, et une soutenance publique.
- Stéphane Bessy (notre professeur référent pour ce projet) est spécialisé dans l'algorithmie et les maths.
- Rendu attendu¹⁵ : une archive + un mémoire 10 pages suivant un plan ou d'après Stéphane Bessy : interface graphique qu'est ce qui a été fait ? Diagramme des classes ?...)
- Première explication des types d'IA possibles :

Niveau 1 : Simuler toutes les possibilités et maximiser un critère ou des critères (avec une note/point ...) et jouer le meilleur coup de manière locale.

Niveau 2 : Faire le niveau 1 mais re-simuler pour l'adversaire avec cette configuration (algo du min/-max) et re-simuler avec cette nouvelle configuration le meilleur coup à jouer. On peut étendre cela autant que possible. Point négatif : temps de calcul très grand (Ordre de grandeur/Corrélation entre temps de calcul et la profondeur de décision ?)

- Une version jouable à 2 joueurs est cependant attendue rapidement peu importe que le côté graphique soit pris en charge.
- Cependant, un cahier des charges minimal (pas trop ambitieux) est nécessaire et demandé sous une à deux semaines (nous nous sommes mis d'accord pour le prochain rendez-vous encore non défini au moment de l'entrevue). Ce cahier des charges doit contenir le minimum comme : la version jouable du jeu à 2 au moins, les implémentations qu'on peut d'ores et déjà faire, etc.
- Récapitulatif des attentes de la prochaine rencontre (nous pourrons commencer à coder seulement après avoir effectué ces tâches) :
 - ⇒ Commencer le rapport (c'est à dire ce qu'on est en train de faire)
 - ⇒ Le cahier des charges
 - ⇒ Réfléchir à une structure du code
 - ⇒ Définir les tâches à réaliser avec un planning (diagramme de Gantt¹⁶ avec EdrawMax)
 - ⇒ Se répartir les tâches (diagramme en camembert ou directement intégré dans le diagramme de Gantt)

15. Moodle de l'UE HLIN405 : <https://moodle.umontpellier.fr/course/view.php?id=1295>

16. Page Wikipédia du diagramme de Gantt : https://fr.wikipedia.org/wiki/Diagramme_de_Gantt

3.2 31 Janvier 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (14 H 00 - 15 H 04)

Les points abordés :

Une aide comme l'IA est difficile à contrer¹⁷, proposer une aide qui affiche le 'meilleur' coup potentiel que le bot aurait pu jouer.

Présentation :

- du diagramme de Gantt.
- du diagramme de classe en UML.
- du code et d'un prototype qui permet à ce jour :

Joueur Une sélection¹⁸ et sa personnalisation pour les 4 joueurs.

Map Selector Le choix parmi X plateaux de jeu déjà existants.

Map Creator La possibilité de créer sa propre map¹⁹.

Volume L'intégration d'un fond sonore (réalisé par Jérémie) avec volume+²⁰ et volume-.

State traduit littéralement par un 'état' de jeu qui sont les différentes fenêtres/boutons/états du jeu c'est à dire Menu, Jouer, Credits, Aide, Option, ...

- une mise en ligne du jeu éventuelle via Unity qui permet avec des plugins de gérer du Java ...

Sauvegarde() Prévoir un moyen de sauvegarder une partie en cours, via une "**fonction**" d'encodage.

Eval() Récupérer la fonction d'évaluation de l'IA du site 1980-games²¹ en faisant du reverse engineering²² et d'un autre côté aller à tâtons pour la notre et l'optimiser au maximum.

IA — Implémenter le Minimax

- On maximise la prise de position sur les bords plus orientée défensive puis offensive
- Implémenter Alpha-Bêta

17. Même la plus simple étant donnée qu'elle fait constamment des choix logiques

18. C'est à dire entre un Pseudo/Image/Equipe

19. Par ailleurs il est bon de noter qu'une map devra garder une symétrie.

20. Réglable par une barre coulissante qui n'est rien d'autre qu'un objet de la Classe *Bouton*

21. <http://www.1980-games.com/>

22. La rétro-ingénierie, ou ingénierie inverse ou inversée, est l'activité qui consiste à étudier un objet pour en déterminer un. Le terme équivalent en anglais est reverse engineering.

3.3 14 Février 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (14 H 00 - 14 H 31)

Les points abordés :

1. Revoir les règles du jeu (correction + ajout pour le PAT par exemple)
2. Fonction "d'échec et mat" : le "PAT" est buggé mais si le terrain connexe normalement il a une case qui touche le jour à chaque fois ?....
3. Explication de la structure sous-jacente au jeu : plateau de jeu, case, joueur ...
4. Une boucle de -1 à 1 pour avoir les cases jouables en gris foncé
5. La même boucle appliquée "depuis" la première pour avoir les autres cases jouables en un gris clair²³
6. Animation à revoir
7. Faire la fonction Eval() pour intégrer une IA
8. Inclure un diagramme pour représenter le score et se passer du bug de l'encadré gris pour les joueurs
9. Pouvoir activer une aide
10. La fonction Save() "d'encodage"
11. Ça ne vaut pas le coup de garder l'arbre calculé pour chaque nouveau tour car possiblement le temps de calcul pour parcourir l'arbre sera supérieur à le re-calculer ...
12. Pouvoir "simuler" une partie

Ce début de réunion a débuté sur une remarque de M. Bessy sur les règles écrites pour notre jeu qui ne sont pas très claires pour une personne qui ne connaît pas le jeu.

En effet des étourderies de français, franglais présentes ont dûes être corrigées. Ensuite une imprécision sur la façon de jouer a été corrigée (sur les cases "adjacentes" à deux de là où il se trouve, le pion sera déplacé.). Au final nous avons ré-écrit les règles en changeant le mot "blob" par "pion" en incluant la notion de couleur dans les déplacements qui permet au joueur de manière intuitive de comprendre qu'il existe une différence dans le choix des cases que lesquelles se déplacer ou se dupliquer.

On a aussi mentionné le fait qu'on a inclus une fonction qui permet de faire un PAT²⁴ cependant elle a besoin de quelques ajustements.

Le troisième point ayant **une importance pour M. Bessy** qui souhaite voir apparaître une explication détaillée de structures sous-jacentes aux fonctionnement du jeu, des choix effectués, etc. Par exemple comment les positions sont calculées.

Pour le cinquième point c'est un détail esthétique qui est loin d'être prioritaire cependant pour un produit final, il serait en effet bon de revoir les effets de transition et de garder un jeu assez fluide. Dans le même genre, on a la représentation du score qu'on peut dire classique jusqu'à maintenant qui pourrait éventuellement changer pour une représentation en camembert ou autre diagramme plus visuel et moins "buggé".

D'un autre coté *la fonction d'évaluation* pour juger quel déplacement est plus intéressant qu'un autre n'est toujours pas codée/fonctionnelle, ce qui nous empêche d'implémenter le *Minimax*. Par ailleurs ça influe aussi sur l'ajout d'une fonction d'aide pour le joueur humain étant donné l'absence d'une IA.

Sinon sans avoir pour le moment codé le nécessaire, M. Bessy nous a apporté une réponse à une question qu'on se posait pour optimiser les calculs sur les coups à jouer représentés avec un arbre²⁵ : "le temps de calcul de parcours de l'arbre sera peut-être plus long/important que le calcul de l'arbre lui-même". Donc nous n'allons pas nous compliquer la tâche mais concevoir une IA implémentée de manière brute.

A la suite de ça, nous pourrons alors d'après notre conception du jeu permettre une "simulation"²⁶ de partie entre deux IA.

Finalement le dernier point abordé est la fonction de sauvegarde pour immédiatement fixer un format de sauvegarde.

23. la couleur permet de différencier le type de déplacement effectué

24. Référence au pat des échecs, qui apparaît quand on veut faire une égalité avec son adversaire

25. pour le moment d'une hauteur maximale théorique 3

26. expressément demandée par M. Bessy

3.4 28 Février 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (14 H 12 - 15 H 00)

Les points abordés :

1. La fin de partie "PAT" précédemment buggée a été légèrement modifiée et prend en charge les fins de parties en 1v1²⁷
2. Système de sauvegarde
3. Fonction/Méthode d'évaluation du terrain (this.calculerScoreTerrain())
4. Structure sous-jacente, peu de choses à dire
5. On arrive doucement à un produit opérationnel
6. Une collection de Collection x Collection pour les coups possibles
7. Load() Charger une partie
8. Débugger le PAT pour N équipes
9. Finir l'implémentation de l'IA : Minimax() > Min(), Max()
10. Du scripting pour faire des stats ...
11. Yanis : Debug le PAT()
12. Jérémie : Implémente le load()
13. Kévin : Finit d'implémenter le Minimax()

Réunion assez sommaire avant ce début de vacances, il n'y a pas eu d'avancées majeures dans le jeu.

La plupart du temps étant alloué à l'IA, d'un autre côté nous n'avons pas de structure sous-jacente à présenter à M. Bessy plus détaillée que ce qu'on lui avait déjà fourni.

Nous avons défini ensemble l'objectif de la prochaine réunion qui est d'avoir un jeu 100% opérationnel dans tous les cas dans un premier temps ce qui inclut une sauvegarde de parties et la reprise de celle-ci.

Il faut que le jeu se termine correctement avec plusieurs équipes en jeu et finalement avoir une IA qui fonctionne.

La prochaine réunion est fixée au 18 mars 2019.

²⁷. peu importe le nombre de joueurs c'est le nombre d'équipe/couleur qui importe car c'est une boucle qui itère sur ces derniers qui permet de faire appel à endOfGame()

3.5 18 Mars 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (13 H 53 - 14 H 42)

Les points abordés :

1. IA
 - Algorithme Minmax implémenté
 - Optimiser le calcul de coup (heuristique)
 - La taille du plateau de jeu impacte le temps de calcul de chaque coup
 - IA du site TwoPlayersGames.org
2. Fonction de sauvegarde terminée
3. Fonction de chargement présente des bogues
4. Détail de la fonction d'évaluation
5. Remarque sur le rapport
6. Remarque sur l'algorithme Minmax

Lors de cette réunion après 3 semaines (avec 1 semaine de vacance) pour avancer le projet, nous avons fait une petite démonstration IA vs IA avec différentes profondeurs.

S'en suit une explication de la fonction d'évaluation telle que décrite ci-après. La fonction d'évaluation attribue un point à chacun des pions sur le plateau et plus un coup possible permet de créer de pions, plus il est **intéressant**.

Ce qui a mis en évidence ce que nous avions déjà remarqué lors de nos phases de test, d'abord le temps de calcul est fortement impacté par la taille du plateau et qu'il faudra qu'on optimise la façon dont les coups sont calculés (c'est à dire ne sauvegarder que les coups "jouables" et non pas un **ArrayList** < des plateaux possibles > représentés tels quels **ArrayList**< **ArrayList** < de position/case > >).

Par ailleurs, après une première étude du code source²⁸, on tire la conclusion que leur IA n'est basée que sur du backtracking²⁹ avec une fonction d'évaluation qui est diluée dedans (ce qui ne la rend pas explicite et compréhensible). Donc nous laissons cette éventualité de côté pour le moment.

Plus encore, lors de cette réunion Jérémie a pu terminé la fonction save qui ne présente plus de bug qui aurait pu être rapportée lors de la précédente réunion cependant certains bugs persistent dans la fonction de chargement.

Kévin a fait une remarque très pertinente sur l'écriture du code d'une méthode qui ne devrait pas prendre plus d'une dizaine de ligne qui rendu possible par une imbrication de méthode qui procède à des test unitaires.

L'avant dernier point étant une remarque pertinente sur le rapport mais tout à fait normale étant donné que le développement du plan n'a pas encore vraiment débuté et mise à part les quelques fautes d'orthographes alors présentes dans ce dernier.

Il faudra au moins une partie où nous justifions le choix du langage, de la librairie et une autre qui explique en terme algorithmique le Minmax avec une analyse de complexité si nécessaire.

Pour terminer cette réunion, M. Bessy à fait une précision quand à l'optimisation de notre algorithme en supprimant la ligne 103 dans la classe IA qui implique de calculer chaque noeud découvert³⁰ alors que l'algorithme veut qu'on calcule à partir des feuilles.

La prochaine réunion est fixée au 27 mars 2019, finalement reportée au 02 avril 2019.

28. Voir <https://pastebin.com/8gFy3qPH>

29. étant donnée la structure du code

30. et dont la valeur sera au final écrasée par celle venant des feuilles

3.6 02 Avril 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (15 H 30 - 16 H 17)

Les points abordés :

1. Remarque sur le rapport
2. IA
 - Minimax
 - Alpha-Bêta
 - Amélioration, granularité
 - Fonction d'évaluation
3. Fin du jeu
4. Fonction de chargement
5. Interface WEB
6. Complexité
7. Une conclusion + explication d'une soutenance

Il faut noter que nous avions choisi de reporter cette réunion d'une semaine par rapport à la date prévue, c'est à dire au 2 avril pour avoir plus de matière à travailler avec M. Bessy.

Pour le premier point, une remarque de M. Bessy qui a trouvé notre rapport plutôt correct. Il y a cependant plusieurs points à améliorer comme une meilleure explication de l'implémentation de l'IA et d'une analyse de complexité du Minimax et encore une conclusion à rajouter. Kévin pense à supprimer (revoir) la partie "ANNO 2018" qui pour le moment explique brièvement comment le groupe s'est formé (avec une touche d'humour ?) qui selon lui ne fait pas de sens. Jérémie est intervenu à ce sujet et pense au même titre que Yanis qu'il faudrait revoir aussi ce paragraphe et nous sommes d'accord pour revoir la section Introduction.

Ensuite, le second point le plus important de la séance qui est la raison principale du report de la réunion est l'implémentation du Minimax qui a été améliorée, débuggée par Kévin en tenant compte des précisions sur le code par M. Bessy lors de la précédente réunion qui fait le bon calcul cette fois-ci et l'ajout d'un random dans le choix du meilleur coup à jouer^{31 32} et une première tentative d'implémentation de l'Alpha-Bêta qui à ce jour n'est pas complète. Une remarque de Kévin est qu'il ne comprend pas pourquoi cet algorithme arrive à faire remonter l'alpha et le bêta d'un autre coup et qui pense que ça vient de là.

Pour débugger la fin du jeu, on repart sur une base propre, le code associé à la fin du jeu³³ a été supprimé pour repartir sur une bonne base au lieu d'essayer de comprendre ce qui avait été écrit.

La fonction de chargement d'une partie³⁴. Nous avons pour idée d'implémenter une interface qui permet de charger 5 sauvegardes au maximum³⁵. L'interface de re-jouabilité serait accessible depuis la slide Jouer avec un bouton "Charger" pour ne pas changer des **design patterns de conception** usuels.

S'en suit une explication courte de l'évaluation de la soutenance qui se compose de 15 min de prestation orale avec un premier plan possible :

- Commencer par une démonstration

31. Lors de plusieurs choix possibles

32. On distingue un vrai changement "comportemental" dans la façon de jouer de l'IA qui est moins bête

33. endOfGame et endOfGameAllStuck et la méthode isStuck

34. Une unique partie à ce jour

35. De manière arbitraire

- Suivi d'une mise en perspective pour expliquer comment on a fait ? (pour que tout prenne son sens)
- Expliquer deux ou trois problèmes avec le support projeté³⁶

Pour la prochaine fois on aura au moins fait l'élagage du Minimax et réparé de bug de la fin du jeu.

La prochaine réunion est fixée au 11 avril 2019.

36. Des diapositives

3.7 11 Avril 2019

Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (11 H 51 - 12 H 41)

Les points abordés :

1. Observation temps de calcul coup/ia/nbJoueur/prof/tailleTab
2. Analyse Alpha-Beta
3. Seconde observation du temps de calcul sans élagage
4. Faire un bot
5. L'interface de re-jouabilité
6. Peaufinage des graphismes
7. Pat
8. Monitoring
9. Ré-explication du déroulé de la soutenance

Lors de cette réunion, Jérémie avait proposé de se concentrer sur une analyse avec comme objectif l'étude du temps de calcul de chaque coup sur une partie complète par IA pour voir à quel niveau on pouvait se limiter par exemple. Avec comme paramètre d'étude la profondeur d'une IA, le nombre d'IA en partie, et la taille d'un plateau de jeu.

Il sera fait pour la prochaine fois éventuellement une analyse sans l'élagage pour avoir plus de matière à comparer. Les dents de scie sont présentes sur le graphique actuel car les mesures ont été faites telles qu'elles : on fait un cumul du temps nécessaire aux mêmes IA pour un coup donné.

On précise un peu plus la fonction d'évaluation, si on arrive à finir complètement l'adversaire³⁷ on compte 100 points³⁸ de plus.

Au cours de la réunion, M. Bessy a ramené une idée très intéressante par rapport au jeu déjà existant : faire jouer notre IA contre celle du jeu en ligne³⁹ pour voir laquelle est la plus **forte**.

Pendant cet échange, Yanis a proposé de faire un bot codé en AutoIT par exemple, mais n'importe quel langage de scripting⁴⁰ pourrait tout aussi fonctionner. AutoIT a simplement une bonne réputation pour ce qui est de faire des macros, des mini-bots mouse-clicker.

Comme idée de développement future, on peut changer les animations (en l'état comment les animations sont codées il suffit de changer les sprites)⁴¹ et quelques lignes de code pour que ce soit effectif.

Mais aussi proposer à l'utilisateur de changer le BG⁴², l'UI⁴³ pour que ça devienne au goût de tout le monde. Le PAT a été complètement re-codé, malheureusement même si les fins de jeu sont opérationnelles, parfois des PAT arrivent alors qu'il n'y a pas de raison. Par exemple J1 peut encore jouer mais J2 non et malgré que ce soit au tour de J1, il y a PAT.

Pour rester dans les idées à venir à programmer, il y a un fichier de log qui aurait pour objectif de montrer plus facilement les parties avec une granularité très fine⁴⁴ pour pouvoir faire par exemple des statistiques aisément.

Il faudra cependant voir comment gérer les I/O pour que cela reste performant.

On a par ailleurs toujours ce **doute** que l'alpha-bêta ne calcule pas ce qu'il calcule et M. Bessy aimerait bien qu'on enlève ce doute pour la prochaine réunion et la fin du projet.

La prochaine réunion est fixée au 26 avril 2019.

37. On rappelle qu'un adversaire ce sont tous les autres pions qui ne sont pas de notre équipe

38. On rappelle que de base un "blob" de plus donne 1 point en plus

39. La version flash qu'on avait reverse

40. Exemple LUA, Python

41. ressources des animations, les images de base

42. Background

43. User-Interface

44. Timestamps à chaque coup + la case joué de où à où

3.8 11 Avril 2019

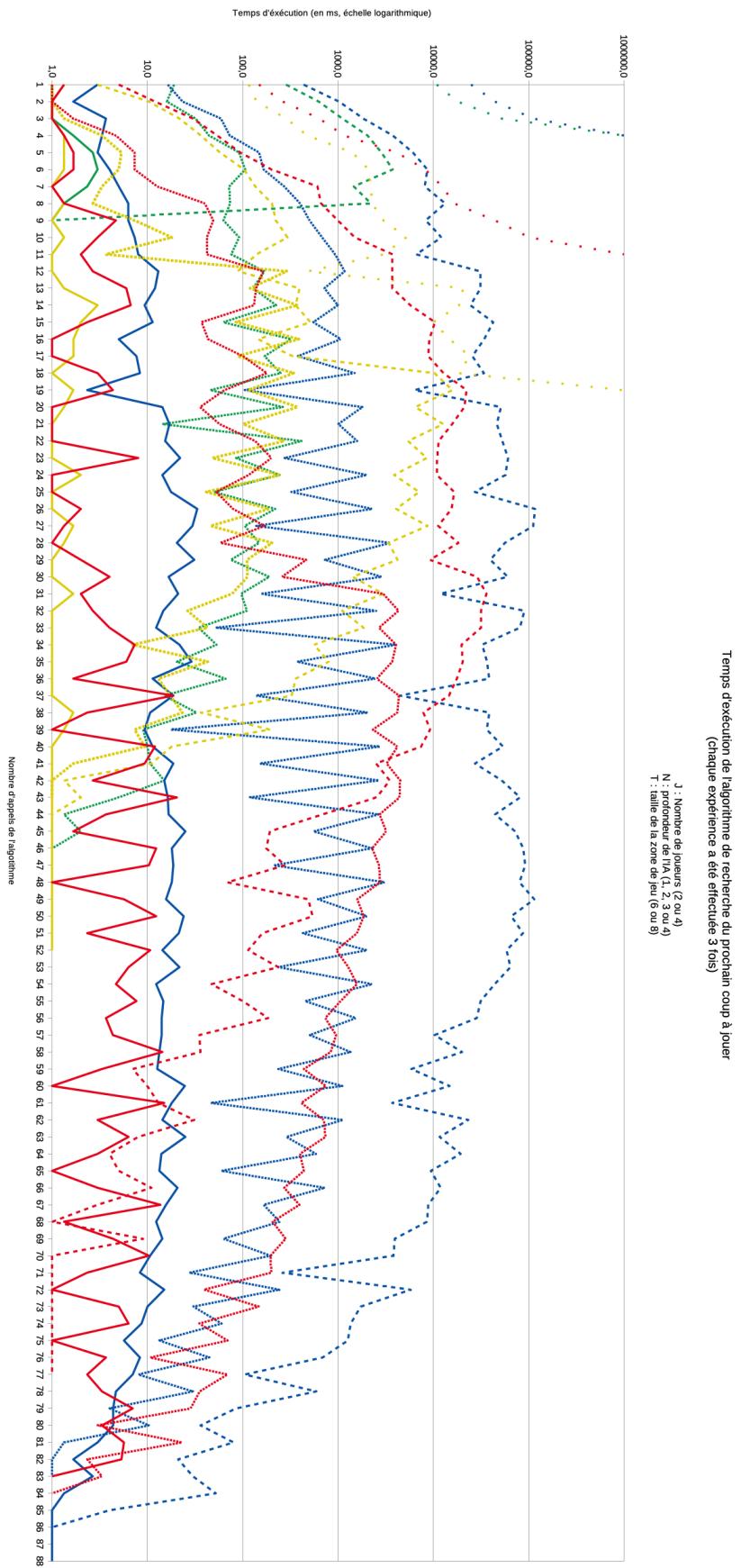
Allouch Yanis, Roux Jérémie, Villaroya Kévin et Bessy Stéphane (11 H 51 - 12 H 41)

Les points abordés :

1. Observation temps de calcul coup/ia/nbJoueur/prof/tailleTab
2. Analyse Alpha-Beta
3. Seconde observation du temps de calcul sans élagage
4. Faire un bot
5. L'interface de re-jouabilité
6. Peaufinage des graphismes
7. Pat
8. Monitoring
9. Ré-explication du déroulé de la soutenance

La prochaine réunion est fixée au 7 mai 2019.

4 Calculs de temps d'exécution de l'algorithme MinMax



23 IN 6T
 23 2N 6T
 23 3N 6T
 23 4N 6T
 23 1N 8T
 23 2N 8T
 23 3N 8T
 23 4N 8T
 43 IN 6T
 43 2N 6T
 43 3N 6T
 43 4N 6T
 43 1N 8T
 43 2N 8T
 43 3N 8T
 43 4N 8T

5 Rapports de version

5.1 Légende

- ✓ Exemple de description d'un bogue non résolu
- ✓ Exemple de description d'un bogue résolu
- ✗ Exemple de description d'une suggestion non implémentée
- ✗ Exemple de description d'une suggestion implémentée

5.2 2 Février 2019 (V1)

- ✗ Fin de partie quand 1 - nombre total de joueur ne peut plus jouer (mais dont il reste au moins une case)
- ✗ Fin de partie quand toutes les cases sont occupées
- ✗ Ajouter le score en étiquette du diagramme en camembert
- ✗ Indiquer les joueurs avec le nom en couleur et l'icône de leur personnage si possible en les séparant par équipe (le tout sous forme d'un tableau/liste)
- ✗ Switch On-Off les animations
- ✗ Un bouton dans 'HOME' qui configure et lance x parties automatiquement et permettent de sauvegarder le résultat
- ✗ Un Timer pour avoir des données plus précises.
- ✗ Threads

5.3 17 mars 2019 (V2)

- ✓ Cadres des équipes et de l'annonce du tour qui empiètent sur le terrain
- ✓ Terrain non-régénéré lors du lancement d'une nouvelle partie sans relancer l'application
- ✓ Accents qui disparaissent (lors du passage par Git Windows?)
- ✓ Blanc lors du "bouclage" de la musique (blanc à la fin du fichier audio)
- ✓ Considère qu'une case adjacente à deux ne peut être atteinte que lorsqu'au moins une case directement adjacente peut être atteinte
- ✓ Lorsqu'on sélectionne un terrain personnalisé et des listes de personnages et si on clique sur Confirmer, alors, affichage d'un fenêtre avec "PAT".
- ✗ Diagramme en camembert qui représente la proportion d'occupation de chaque équipe

5.4 11 avril 2019 (V3)

- ✓ PAT
- ✓ Élagage
- ✓ IA vs IA d'un niveau important fait planter le render()
- ✓ Le load() plante quand une IA est concerné ?
- ✓ Les fins de parties sont prises en charge, plus de crash quand il y a une IA en jeu
- ✗ Une mise sur le WEB
- ✗ Revoir les animations
- ✗ Proposer une personnalisation de l'interface
- ✗ Un bot pour faire jouer notre version contre une autre