HLIN405
Projets de Programmation de L2
(Dirigé par Stéphane Bessy)
-
"Blob Wars"
-

Allouch Yanis - Roux Jérémie - Villaroya Kévin

2018 - 2019

Table des matières

Première partie

Introduction

Dans le cadre de l'UE HLIN405 (Projet de Programmation de L2) nous avons réalisé un Blob Wars [1] (Figure 1). Notre groupe est composé de trois membres : Allouch Yanis, Roux Jérémie et Villaroya Kévin.

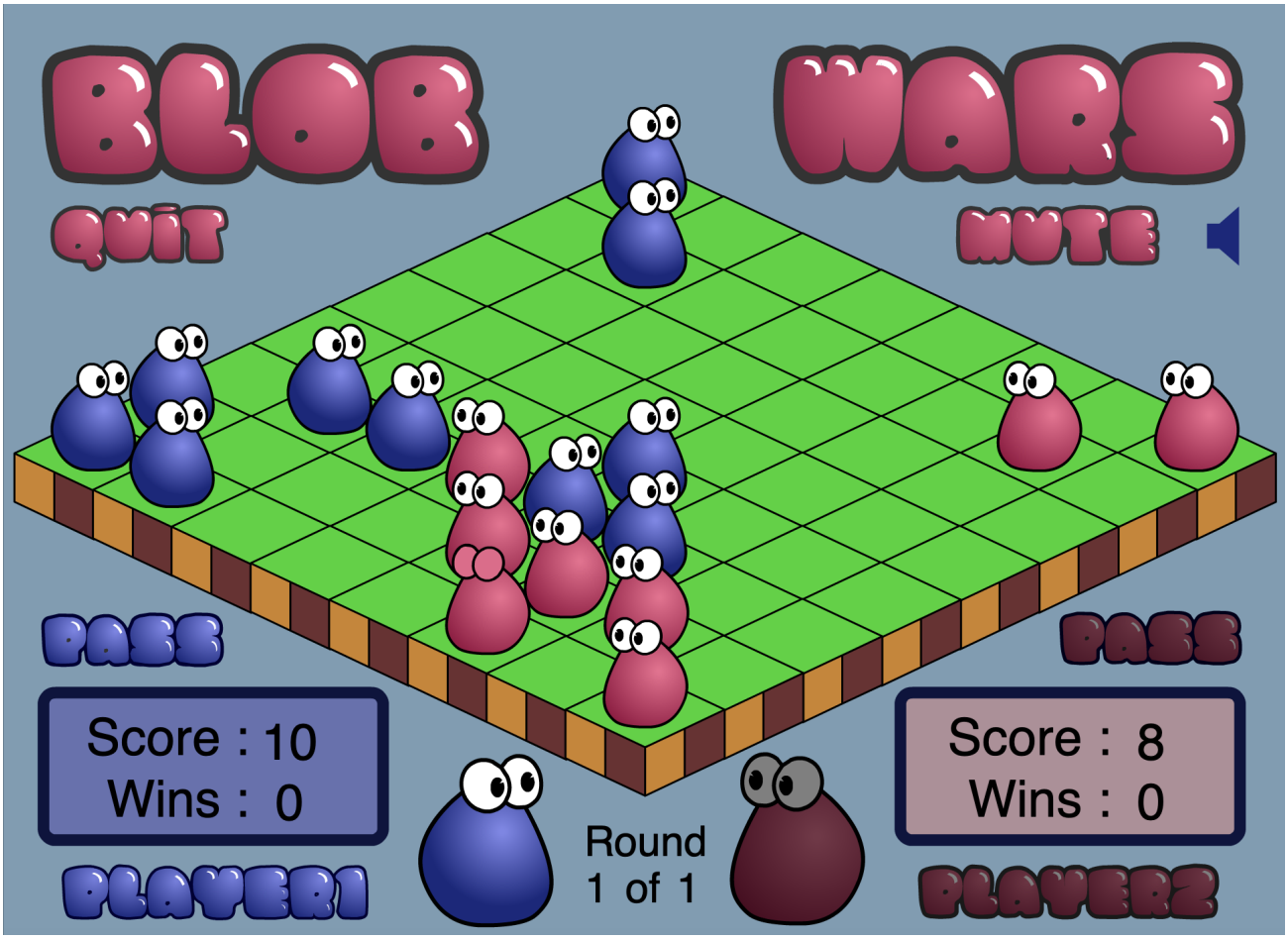

FIGURE 1 – Exemple d'un Blob Wars

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) 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

2 Diagramme de Gantt

Nous avons conçu un diagramme de Gantt (Figure 3) afin de nous organiser dans le développement et mieux gérer le temps disponible et les échéances.

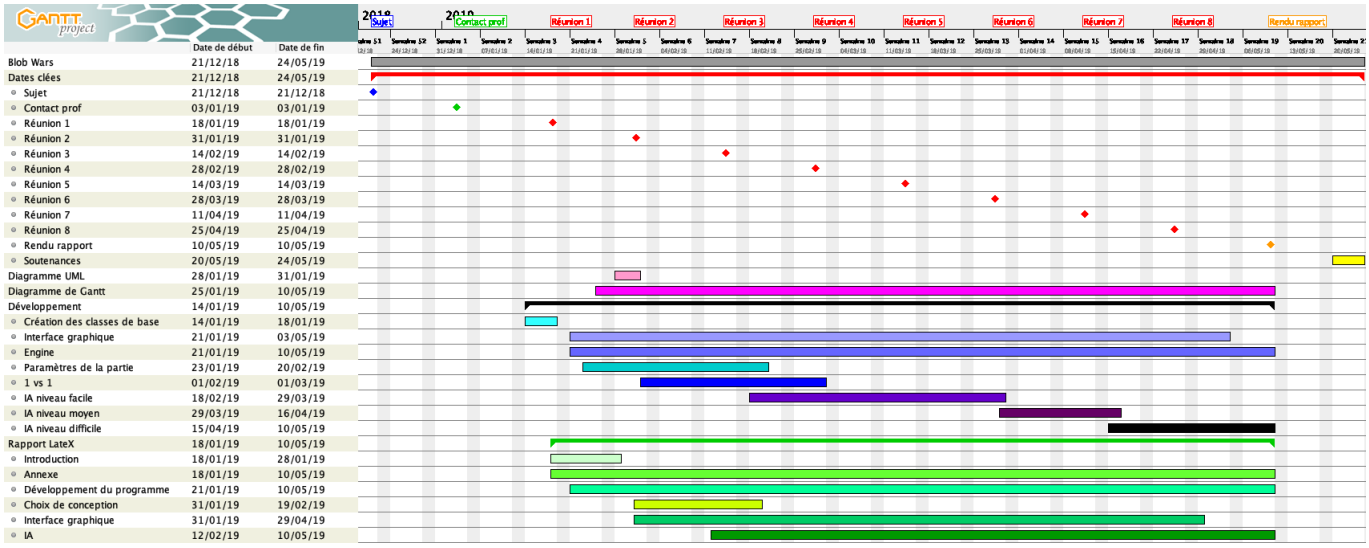

FIGURE 3 – Diagramme de Gantt initial

Le Blob Wars a été codé en Java 8 conjointement avec la librairie Slick [5] sur l'environnement de développement "intégré" Eclipse (Figure 4). Notre projet est suivi par M. Stéphane Bessy [6] que nous remercions pour son soutien, ses conseils et sa présence lors de nos réunions bimensuelles [7]. M. Bessy est maître de conférences en informatique à l' Université de Montpellier et enseignant au département info de la FdS. Il est membre de l'équipe **AL**gorithmes de **G**raphes et **Co**mbinatoire (AlGCo) du LIRMM.

FIGURE 4 – Logo de l'environnement de développement "intégré" (IDE) Eclipse

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 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.

ANNO - 2018

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 passionnés par certains domaines de l'informatique :

— programmation et développement

— algorithmie

— 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, 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	1
#02	Blob Wars	5	4	4	13	2
#19	Kakuro	5	2	5	12	3
#23	Résolution de Picross	6	2	4	12	4
#10	Algorithme de Aho-Corasick	3	6	2	11	5
#07	Résolution d'un casse-tête Nurikabe	4	1	4	9	
#18	Quarto	3	1	5	9	
#06	ASCII art	4	4	1	9	6
#16	Dancing with Donald	4	3	2	9	7
#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 5 – 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 tableur qu'il a conçu pour l'occasion (Figure 5). 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.

4 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 grise foncée (adjacente à votre pion) pour le dupliquer sur cette case
  - Sélectionner une case grise claire (adjacente à 2 cases de votre pion) pour le déplacer sur cette case
- Pour chaque pion ennemi adjacent à la case sur laquelle vous vous êtes déplacé ou dupliqué, il 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 deux joueurs ou plus 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

Organisation du développement

1 Présentation du projet

Le Blob Wars est un jeu-vidéo de type réflexion qui implique une grande part de programmation "applicative" 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 :

— impératif

— orienté objet

— parallèle

— fonctionnelle

— événementielle

— web

Et les différents langages qui interprètes ces différents paradigmes [8] sont favorisés dans le choix du développement d'une application.

Prenons un exemple naif ayant pour cahier des charges de concevoir une application qui soit :

— adaptative (traduit par *responsive*)

— accessible partout

— visuellement belle et esthétique

Le choix se tournera vers une association des langages du **Web**.

Reprenons le blob wars, c'est un jeu vidéo. Par définition, un jeu vidéo peut se voir comme un enchaînement d'événement, certains événement peuvent influer certains objet du jeu qui vont à leur tour relancer un événement et ainsi de suite jusqu'à ce qu'un état final soit atteint. De plus 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 (Fig **??** p. **??**) qui regroupe les pour et contre en fonction des langages qui nous sont connus [10] et de nos critères.

Ce qui ressort c'est C++, Java, PHP/JS, Python, au moment du choix du langage une application web n'était pas considéré du a la taille du jeu. Ensuite la simplicité d'écriture et de prototypage ainsi que les centaines de librairies de python aurait pu être utiliser a notre avantage cependant on décider de le refuser et au final parce qu'on a commencé a programmer en Java au second semestre on a décidé que ça pouvait être un bon choix que de joindre l'utile a l'agréable de plus le Java permet de construire du code structuré et facilement maintenable. Au final pour écrire du code Java nous nous sommes orientés vers la solution la plus efficace et la plus utilisé c'est a dire Eclipse qui fournit tous les outils pour écrire du code et le debugger.

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

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

TABLE 1 – Comparatif des langages possibles

2 Objectif

Le but premier de ce sujet de TER reste la création d'un jeu vidéo pour y implémenter des algorithmes de recherche notamment :

1. MinMax
2. Alpha-Bêta

Faire de l'heuristique sur ces algorithmes et puis tout ce qui nous semble intéressant et constructif de plus sur ce projet.

3 Structure des 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, ainsi que deux boutons un pour quitter et un pour revenir sur la fenêtre précédente

⇒ Boutons pour quitter le jeu et y revenir

— **Game :**

⇒ Affiche les scores à gauche, individuels et par équipe ainsi que le plateau du jeu en lui-même.

— **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, donc c'est le jeu en lui-même.

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

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

4 Les fichiers de configuration

```
2019/04/02 00:23:58

4

1 tres_facile Joe red firefox 2
2 joueur William blue ie 5
3 moyen Jack green windows 6
4 joueur Averell yellow chrome 4
2

8

11000400
11004400
100XX000
30XXXX02
00XXXX20
000XX020
30000000
30000000
```

FIGURE 6 – Exemple d'un fichier de configuration

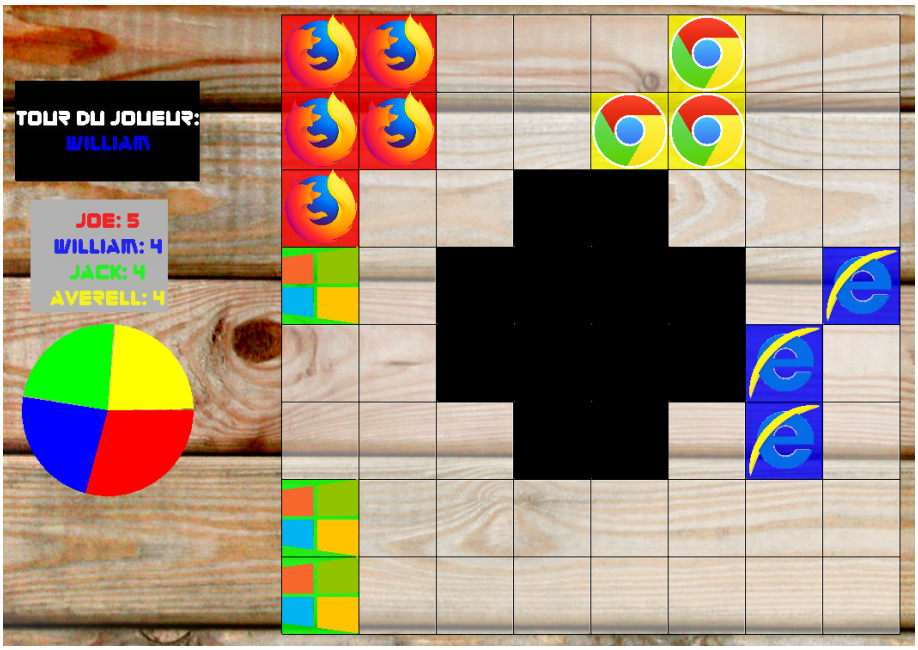

FIGURE 7 – Capture d'écran de la partie correspondant au fichier de configuration de la Figure 6

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 :

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).

5 IA minmax

5.1 Fonctionnement de la classe

IA est une classe un peu spéciale car elle devient une instance qu'une seule fois [11] ! Effectivement une IA, va agir de la même façon face à une situation de jeu donnée. Elle contient les méthodes nécessaires au calcul du prochain coup.

5.2 Méthodes de IA

eval : Permet a 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 puisse 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 positions de calculer leurs coup possible.alors que avant nous parcourions le jeu pour connaitre l'emplacement des pions de tous les joueurs.

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 de mettre un random lorsque deux choix possible pour 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 l'alpha/bêta une fois implémenté

Voici l'implémentation Java du Minimax :

```
1  private Etat minImax(Etat terrain, int joueur, GestionTour gestionTour, int profondeur){
2    boolean myTurn = gestionTour.getTour().getID() == joueur;
3    Etat[] choix = simulerTousLesCoups(terrain, gestionTour.getTour().getID());
4    Etat choisis = null;
5    if (myTurn) {
6        profondeur--;
7    }
8    gestionTour.nextTour();
9    if ((profondeur > 0 || myTurn) && choix.length != 0) {
10     for (int i = 0; i < choix.length; i++) {
11       minImax(choix[i], joueur, new GestionTour(gestionTour), profondeur);
12       if (profondeur == 0 && myTurn) {
13         choix[i].valeur = Etat.calculerValeurEtatJeu(choix[i], joueur);
14       }
15     }
16     if (myTurn || Joueur.isAllied(gestionTour.getTour().getID(), joueur)) {
17       choisis = max(choix);
18       terrain.valeur = choisis.valeur;
19     } else {
20       choisis = min(choix);
21       terrain.valeur = choisis.valeur;
22     }
23   }
24   return choisis;
25 }
```

Etant donnée qu'on ne s'intéresse qu'a la partie algorithmique nous pouvons réécrire ce code comme suit :

11. Ce que l'on appelle un singleton [12]

```
minMax(terrain, joueur, gestionTour, profondeur): Jeu
debut
  boolean myTurn <- True si c'est au tour du joueur;
  Etat[] choix <- On simule tous les coups du jeu en cours;
  Etat choisis <- NULL;
  Si (myTurn) alors
      profondeur--;
  Fin si
  nextTour(); //On passe au joueur suivant
  Si ((profondeur > 0 ou myTurn) et choix.length != 0) alors
    Pour i de 0    choix.length() faire
      minImax(choix[i], joueur, new Instance d'un tour, profondeur);
      Si (profondeur == 0 et myTurn) alors
        choix[i].valeur <- calculerValeurEtatJeu(choix[i], joueur);
      Fin si
    Fin pour
    Si (myTurn ou celui d'un allie) alors
      choisis <- max(choix);
      terrain.valeur <- choisis.valeur;
    Sinon
      choisis <- min(choix);
      terrain.valeur <- choisis.valeur;
    Fin si
  Fin si
  retourner choisis;
}
Fin algorithme
```

6 Élagage alpha/beta

6.1 Compréhension de cette élagage

L'objectif de cette élagage est comme son nom l'indique de couper des branches de notre arbre, pour avoir moins de contenu a explorer, qui dit moins de contenue a explorer dit moins de calcul a effectuer. Cette élagage est extrêmement utilisé car elle ne permet d'avoir aucune perte d'information , donc de meilleur coup par sa présence tout en diminuant les calculs a effectuer.

6.2 Son fonctionnement

7 Heuristique

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

Temps maximale autorisé de chaque coup : quelques dizaines de secondes voir 1 ou 2 minutes...
Temps moyen pour un joueur humain : quelques dizaines de secondes.
Temps moyen de l'algorithme sur un plateau 6x6 jusqu'à 10x10 :

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

Il sera intéressant de compléter cette section et ce tableau pour plus tard avec un nouveau de jeu de donnée sur des plateaux de taille plus variées, en incluant l'alpha-bêta, et en n'étant plus précis sur les mesures des données actuelles et future.

Troisième partie

Annexe

1 Cahier des charges

1.1 Premier RDV

— UML
— Gantt
— Répartition des tâches
— Rapport

1.2 Deuxième RDV

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

1.3 Troisième RDV

— Implémentation Minimax
— Implementation Alpha-Beta
— Load
— Rapport

2 Rapports d'entrevue

2.1 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 attendu [13] : 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'accords 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 implémentions 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 a réaliser avec un planning (diagramme de Gantt [14] avec EdrawMax)

⇒ Se répartir les tâches (diagramme en camembert ou directement intégré dans le diagramme de Gantt)

13. Moodle de l'UE HLIN405 : https ://moodle.umontpellier.fr/course/view.php ?id=1295
14. Page Wikipédia du diagramme de Gantt : https ://fr.wikipedia.org/wiki/Diagramme_de_Gantt

2.2 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 contrer [15], 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 sélection [16] et sa personnalisation pour les 4 joueurs.

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

  Map Creator La possibilité de créer sa propre map [17].

  Volume L'intégration d'un fond sonore (réalisé par Jérémie) avec volume+ [18] 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 save une partie en cours, via une **"fonction"** d'encodage.

Eval() Récupérer la fonction d'évaluation de l'IA du site 1980-games [19] en faisant du reverse engineering [20] et d'un autre côté aller a tâtons pour la nôtre et la **"tuner"** au maximum.

IA
- Implémenter le Minimax
  - On maximise la prise de position sur les bords + orienté défensive puis offensive
- Implémenter Alpha-Bêta

15. Même la plus simple étant donnée qu'elle fait constamment des choix logiques
16. C'est à dire entre un Pseudo/Image/Equipe
17. Par ailleurs il est bon de noter qu'une map devra garder une symétrie.
18. Réglable par une barre coulissante qui n'est rien d'autre qu'un objet de la Classe *Bouton*
19. http ://www.1980-games.com/
20. 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.

2.3 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 clair [21]
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'encadrer 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 due 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 pat [22] 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 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 posaient pour optimiser les calculs sur les coups à jouer représentés avec un arbre [23] : "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" [24] de partie entre deux IA.
Finalement le dernier point abordé est la fonction de sauvegarde pour immédiatement fixer un format de sauvegarde.

21. la couleur permet de différencier le type de déplacement effectué
22. Référence au pat des échecs, qui apparaît quand on veut faire une égalité avec son adversaire
23. pour le moment d'une hauteur maximale théorique 3
24. expressément demandée par M. Bessy

2.4 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 buggé a été légèrement modifié et prends en charge les fins de partie a 1v1 [25]
2. Système de sauvegarde
3. Fonction/Méthode d'évaluation du terrain (this.calculerScoreTerrain())
4. Structure sous-jacente peu de chose 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. Jeremie : Implémente le load()
13. Kevin : Fini 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 allouz à l'IA, d'un autre côté nous n'avons pas de structure sous-jacente à présenter à M. Bessy plus détaillé 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 inclus une sauvegarde de partie 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.

25. 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()

2.5 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 terminer
3. Fonction de chargement présente des bogues
4. Détail de la fonction d'évaluation
5. Remarque sur le rapport
6. Remarque sur l'algo Minmax

Lors de cette réunion après 3 semaines (avec 1 semaine de vacance) pour avancer le projet, nous avons fais une petite démo d'une IA vs IA avec différente profondeur.
S'ensuit une explication de la fonction d'évaluation tel que décrite ci-après. La fonction d'évaluation attribue un point a chacun des pions sur le plateau et plus un coup possible permet de crée de pion, plus il est **intéressant**.
Ce qui a mis en évidence ce que vous avions déjà remarqué lors de nos phases de test, d'abord le temps de calcul est fortement impacté par la taille du plateau de manière très significatif et qu'il faudra qu'on optimise la façon dont les coups sont calculer c'est a dire ne sauvegarder que les coups "jouables" et non pas un **ArrayList** < des plateaux possibles > représenté tel quel **ArrayList**< **ArrayList** < de position/case > >.
Par ailleurs après une première étude du code source [26] on tire la conclusion que leur IA n'est basée que sur du backtracking [27] avec une fonction d'évaluation qui est dilué dedans ce qui ne la rends pas explicite et compréhensible. Donc nous laissons cette éventualité de côté pour le moment.
Plus encore lors de cette réunion Jeremie a pu terminer la fonction save qui ne présente plus de bug qui aurait pu être rapporté lors de la précédente réunion cependant certains bug persiste dans la fonction de chargement.
Kevin 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 a des test unitaires.
L'avant dernier point étant une remarque pertinente sur le rapport mais tout a fais normal étant donnée que le développement du plan n'a pas encore vraiment débuté et mis a par les quelques erreurs d'orthographes alors présente dans ce dernier.
D'abord 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 Mr Bessy à fais une précision quand a l'optimisation de notre algorithme en supprimant la ligne 103 dans la classe IA qui implique de calculer chaque noeud découvert [28] alors que l'algorithme veut qu'on calcule a partir des feuilles.

La prochaine réunion est fixée au 27 mars 2019.

26. Voir https ://pastebin.com/8gFy3qPH
27. étant donnée la structure du code
28. et dont la valeur sera au final écrasé par celle venant des feuilles

3 Rapports de version

3.1 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

3.2 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

3.3 17 mars 2019 (V2)

- ✔ IA vs IA d'un niveau important fait planté le render()
- ✔ Le load() plante quand une IA est concerné ?
- ✔ 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 adjacent 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