Rapport final de projet

Reversi

Réalisé par :
Thezenas Kylian, Perry Ethan, Duplessis Kergomard Ewan,
Laux Jules-Arthur.

Sous la direction de :
Hisler Gaelle

Pour l'obtention de DUT Informatique

Année universitaire 2020-2021
https://github.com/JulesArthurLAUX/Reversi

Remerciements

Tout au long de ce projet, plusieurs personnes ont apporté leur aide et leur soutien. Nos remerciements vont donc à notre tutrice, Mme Hisler, qui nous a guidé et aiguillé du début à la fin, nous permettant de résoudre nombre de problèmes et de passer nombre d'obstacles, et sans qui le projet n'aurait jamais pu arriver à son terme. Des remerciements vont également à Mme Messaoui qui a suivi de près et nous a conseillés lors de la rédaction du rapport de projet. Il faut également remercier l'ensemble du corps enseignant de l'IUT Montpellier-Sète qui a su répondre à nos questionnements et ainsi permettre l'avancée du projet Reversi sans encombre.

Sommaire

Table des figures

Glossaire

Algorithme : Suite finie d’opérations permettant de résoudre un ou plusieurs problèmes.

Fonction : Ensemble d’instructions faisant partie d'un programme ou d’un sous-programme ayant la capacité d'effectuer une tâche indépendamment du programme.

Réseau : Mise en relation de plusieurs équipements permettant le partage et l'échange de données.

Intelligence Artificielle (IA) : Programme informatique complexe capable de simuler des traits de l’intelligence humaine.

minmax : Algorithme qui s'applique à la théorie des jeux, pour les jeux à deux joueurs à somme nulle consistant à minimiser la perte maximum.

Jeu à somme nulle : Jeu où la somme des gains et des pertes de tous les joueurs est égale à 0. Autrement dit, le gain d’un joueur est égal à la perte de son adversaire.

Introduction

Les jeux de plateau en réseau voient s'affronter généralement deux joueurs par écrans interposés. Mais parfois l'un d'entre eux n'est autre qu'un programme, une intelligence artificielle (IA) qui doit être capable de s'opposer à des débutants ou d'offrir un challenge à des joueurs aguerris. Il faut donc développer pour ce genre de jeu des IA capables de présenter plusieurs niveaux de difficulté.

Notre problématique consiste premièrement, à créer une telle intelligence artificielle, capable d'adapter son niveau à la demande de l'utilisateur. L'enjeu est donc d'avoir une intelligence artificielle adaptable. Secondement, si nous sommes capables de répondre à la première problématique, nous aborderons le fait de mettre en ligne le projet afin de proposer une expérience de joueur contre joueur. L'enjeu secondaire serait alors la mise en réseau du projet.

Au cours de ce projet, l'intelligence artificielle sera développée pour un Reversi : un jeu de plateau classique dont les règles sont simples, afin de pouvoir se concentrer sur la conception de l'IA capable de proposer à l'utilisateur des difficultés à plusieurs niveaux. Le Reversi étant un jeu de plateau à deux joueurs, la mise en réseau permettra à n'importe quel utilisateur d'en affronter un autre autour du reversier (nom du plateau).

Après avoir expliqué les règles du Reversi, nous présenterons une analyse du sujet, et une analyse des différents besoins. Par la suite, nous nous intéresserons à la partie technique, aux différentes fonctions qui seront essentielles au bon fonctionnement du programme, et à l'algorithme minmax sur lequel reposera l'intelligence artificielle. Enfin, nous observons les résultats à la fin du projet, et son déroulement avec une rétrospective et une analyse du travail fourni, avant de conclure.

1. Analyse

Analyse du sujet et de son contexte

Le Reversi est un jeu de plateau à deux joueurs, très connu qui date des années 1880. Les adversaires s'affrontent sur un plateau unicolore, de 8 par 8 (64 cases) représentant l'espace de jeu, muni de pions bicolores (noirs et blancs).

Au début d'une partie de Reversi 4 pions sont disposés au centre du plateau, 2 noirs et 2 blancs. Tour à tour, les joueurs placent alors un pion sur le reversier. Cependant, pour qu'un coup soit valide et puisse être joué, le joueur doit poser le pion de sa couleur sur une case vide, adjacente à un pion adverse. Il faut également qu'en posant son pion, il encadre un ou plusieurs pions adverses entre le pion qu'il pose et un pion à sa couleur, déjà placé sur le plateau. Cette prise en sandwich peut se faire aussi bien horizontalement, verticalement, qu'en diagonale. Le joueur retourne le ou les pions qu'il vient d'encadrer, qui deviennent ainsi de sa couleur.

Afin de remporter la partie, il faut à la fin de cette dernière, posséder plus de pions que son adversaire. Une partie se termine lorsque aucun des deux joueurs ne peut plus jouer de coups ou qu'il ne reste plus de case vide.

Le Reversi a été inventé et réinventé, on peut même qualifier ce jeu de jeu "terminé", ce qui signifie que dans des conditions où les deux joueurs jouent parfaitement bien on peut savoir qui gagnera dès les premiers coups. Le but de ce projet est de développer une intelligence artificielle capable de prévoir un certain nombre de coups à l'avance.

Le logiciel a pour but d'être simple, et prévoit donc une utilisation simple et un visuel ergonomique, qui facilite la prise en main pour l'utilisateur. En effet, même s' il ne connaît pas les règles, l'utilisateur ne doit pas être perdu, celles-ci devront donc être renseignées. L'implémentation du Reversi sera sous la forme d'un site web, permettant à l'utilisateur de jouer contre une IA ou, si la mise en réseau est réussie, de jouer contre une seconde personne en ligne.

Le Reversi est un vieux jeu, en effet, plus de cent vingt ans d'existence, il a donc connu énormément d'adaptations en jeu de société mais il possède également de multiples adaptations sur numérique sur internet, que ce soit en ligne ou en local via une application. Nous avons pu trouver différentes adaptations très similaires en beaucoup de points. Mais si certaines se distinguent des autres, c'est principalement en deux points: le visuel, et l'implémentation de leur IA.

Analyse des besoins fonctionnels

Afin de proposer à l'utilisateur un Reversi simple d'utilisation, de nombreuses fonctionnalités sont nécessaires.

Premièrement, pour suivre le déroulement et l'avancée de la partie, chaque coup de l'intelligence artificielle doit être clairement visible par le joueur. Il faudra donc faire en sorte que l'IA ne joue pas instantanément après le tour joueur, mais après un très court laps de temps, pour que ce dernier puisse voir où le pion est posé et quels pions sont retournés.

Deuxièmement, pour faciliter le jeu aux débutants, il nous semble intéressant de leur proposer de mettre en évidence les coups possibles par des marques sur les cases vides *(Figure 1)* où il est possible de jouer. Ainsi, le joueur saura où il peut jouer sans avoir à chercher ni omettre des possibilités.

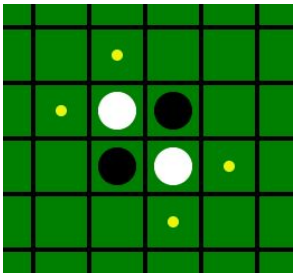


Figure 1: Mise en évidence des coups jouables

Ensuite, il semble judicieux d'ajouter un bouton permettant de passer son tour, dans le cas où le joueur ne peut pas jouer. Ce bouton déclencherait donc le tour de l'IA si cette dernière peut jouer. Si ce n'est pas le cas, la partie devrait s'arrêter.

L'objectif principal étant d'implémenter un programme à plusieurs niveaux de difficulté, il faudra bien évidemment prévoir un moyen pour l'utilisateur de sélectionner la difficulté souhaitée.

Enfin, il faut que l'utilisateur puisse se renseigner sur les règles sans avoir à chercher sur internet. Il faudra donc incorporer une zone expliquant le but du Reversi et son fonctionnement.

Analyse des besoins non-fonctionnels

La plus grande majorité du programme est invisible pour l'utilisateur, les différentes méthodes de recherche, vérification et calcul lui sont inconnues, mais sont belles et bien présentes.

Pour savoir si un coup est valide à jouer, il faudra au programme un algorithme de recherche qui vérifiera séquentiellement les huit directions autour de chaque case vide, afin de savoir si oui ou non un pion peut y être légitimement posé. De même, un algorithme devra trouver et changer de couleur les pions gagner à chaque coup. Ces deux algorithmes sont très similaires, mais leurs méthodes de recherche étant légèrement différentes, il semble plus judicieux de les dissocier. Une autre des fonctions basiques pour obtenir un Reversi fonctionnel est un moyen de vérifier si la partie est terminée, ou si elle doit continuer.

Il faudra aussi intégrer une intelligence artificielle avec plusieurs niveaux de difficulté pour les parties contre l'ordinateur. Cinq niveaux de difficulté semblent suffisants, cependant il faudra considérer l'éventualité de devoir ajouter quelques niveaux supplémentaires et donc coder cette intelligence artificielle en conséquence. Le reversi étant un jeu à deux joueurs à somme nulle (lorsqu'un joueur gagne des points, son adversaire en perd un nombre équivalent), un algorithme du type minmax semble approprié pour la pseudo intelligence artificielle.

Un objectif supplémentaire étant le jeu en joueur contre joueur, nous devons prévoir la mise en réseau, et la possibilité de remplacer l'IA par un autre utilisateur.

2. Rapport technique

Conception

Le projet est de créer un jeu de Reversi. Il a d'abord été conçu via un diagramme de classes. Le premier langage retenu fut le Java, car nos compétences avec celui-ci auraient grandement diminué la durée de production du code pour nous permettre de créer un visuel plus développé. Cependant, suite à des discussions avec notre tutrice, et un des objectifs étant, à terme, la mise en réseau du projet, il a été décidé de passer en JavaScript. En effet, il s'agit d'un langage réseau, il est donc plus adapté à ce but.

On peut voir sur la Figure 2, une représentation d'une partie à l'aide d'un diagramme de séquence. On remarque que l'utilisateur est l'acteur principal de jeu. Tout d'abord il va choisir le niveau de difficulté de la partie et ainsi pourvoir soit jouer un coup, soit passer son tour. Dans les deux cas, va avoir lieu une vérification de fin de partie. Si la partie est effectivement terminée alors le jeu va pouvoir afficher le score et proposer à l'utilisateur de relancer une partie. Dans le cas contraire, c'est au tour de l'intelligence artificielle de jouer. Pour ce faire, elle va explorer les coups possibles puis utiliser un algorithme MinMax afin de jouer le meilleur possible. Suite au tour de l'IA, le programme effectue une nouvelle vérification de fin de partie. Finalement le déroulement d'une partie de Reversi peut se représenter par une boucle où le joueur et l'IA enchaînent les coups tant que la partie n'est pas terminée.

On peut voir à l'aide de la figure 3, les différents états d'une partie de Reversi. Les états par lesquels passe la partie sont semblables à tous les jeux de plateau tels que les échecs ou encore le jeu de dames. Tout d'abord le choix de la difficulté fait par l'utilisateur entraîne un changement d'état afin de créer la partie. Suite à cela va s'enchaîner des coups successifs de la part de l'utilisateur et de l'intelligence artificielle jusqu'à que l'un des deux remporte la victoire : état final de la partie.

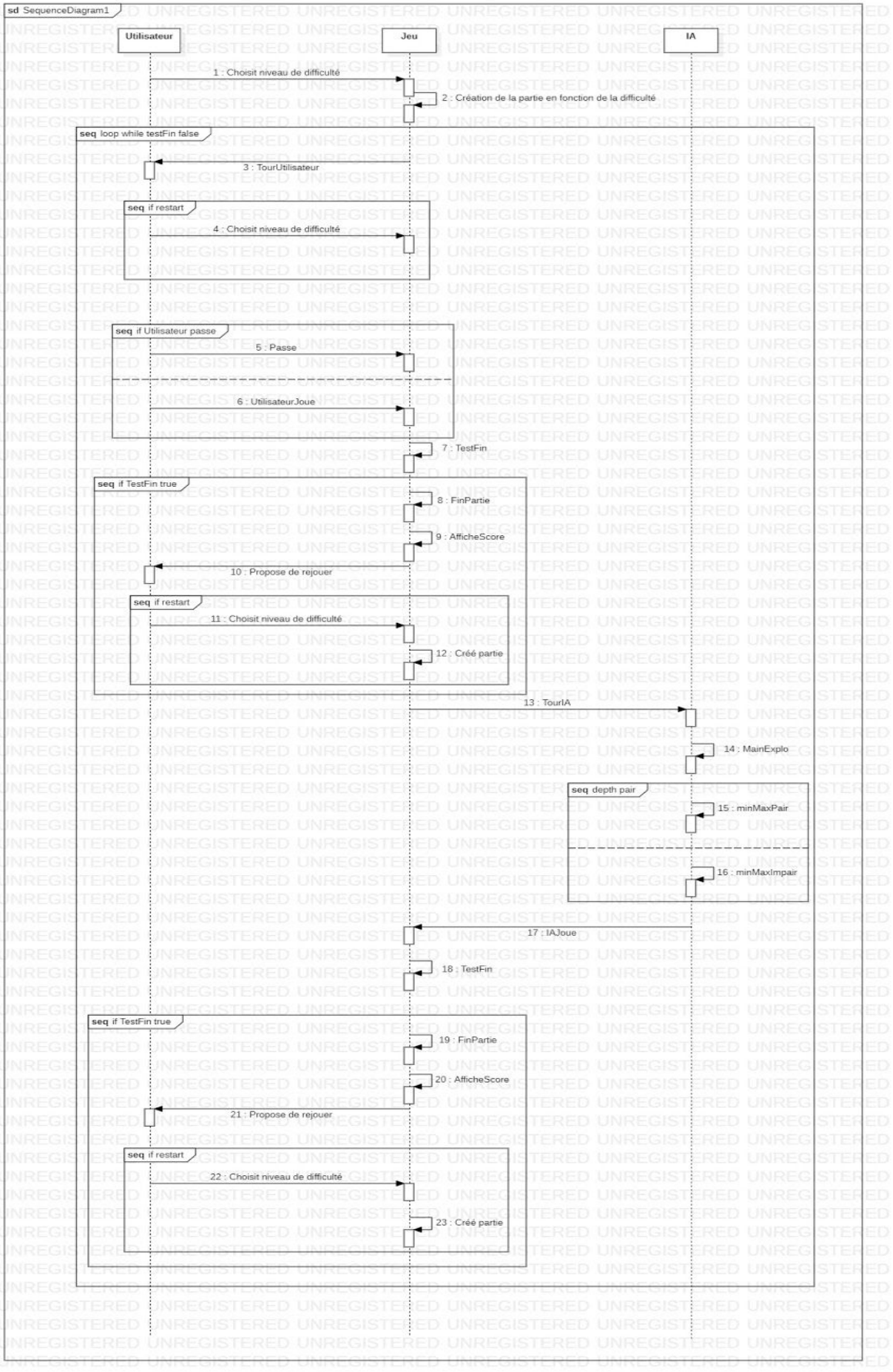


Figure 2: Diagramme de séquence

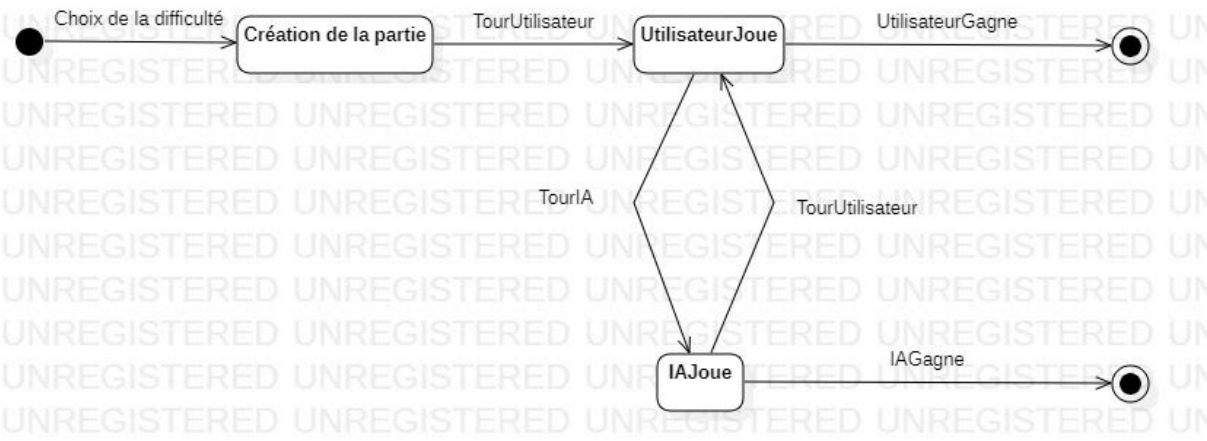


Figure 3: Diagramme d'état-transition

Algorithme

Le programme reposera sur de nombreuses fonctions. Mais certaines seront d'une importance capitale pour le bon fonctionnement du jeu. On peut notamment penser aux fonctions d'explorations, aux fonctions permettant à l'intelligence artificielle de jouer intelligemment, et aux fonctions d'évaluations de l'état de la partie.

Comme il l'a été expliqué précédemment dans les règles du jeu, un coup n'est légitime que si le pion est placé sur une case vide adjacente à un pion adverse, et qu'il encadre, avec un autre pion de la même couleur déjà placé sur le plateau, un ou plusieurs pions adverses. Il est donc nécessaire de coder un algorithme qui explorera le plateau, pour déterminer quels sont les coups jouables et quels pions devront être retournés, le plateau étant représenté par un tableau de 0 (case vide), de 1 (pion de l'ia) et de 2 (pion de l'utilisateur) afin de pouvoir attribuer un pion à un joueur ou à une case vide.

Dans un reversi, la réflexion stratégique doit se faire horizontalement, verticalement mais aussi en diagonale. C'est pourquoi l'exploration des cases autour d'un pion devra se faire dans 8 directions (*Figure 4*).

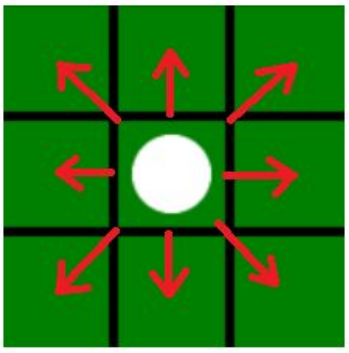


Figure 4: Illustration de l'exploration 1

L'algorithme d'exploration enregistrera une couleur "active" (celle du joueur dont le tour est en cours) et parcourra le plateau case par case. A chaque case vide, l'idée sera d'observer les huit cases adjacentes afin de déterminer si un pion y est déjà présent ou pas, et si oui, de quelle couleur. Plusieurs cas de figures se présentent alors :

- La case est vide ou comporte un pion de la même couleur : dans ce cas, rien de particulier ne se passe, et l'algorithme continue son exécution en passant à la case adjacente suivante.
- La case comporte un pion de la couleur opposée : dans ce cas, l'algorithme va analyser une à une les cases suivantes dans la même direction, tant qu'elle comporte des pions de cette couleur. S'il sort du plateau ou tombe sur une case vide, il reprend son exécution. En revanche, s'il finit par tomber sur une case abritant un pion de la couleur "active", la case d'origine est considérée comme étant une case sur laquelle un coup peut être joué.

Illustrons avec un exemple :

Ici la couleur active est la couleur noire, l'exploration doit donc permettre de déterminer où peut être placer un pion noir.

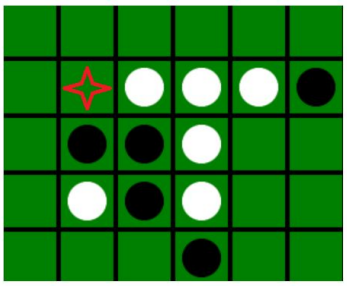


Figure 5: Illustration de l'exploration 2

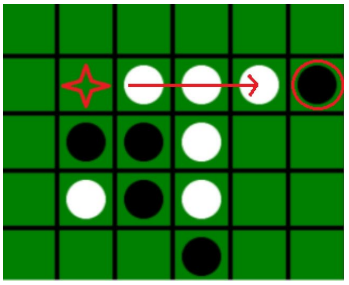


Figure 6: Illustration de l'exploration 3

Dans cette situation (*Figure 5*) si l'on explore selon l'algorithme le plateau autour de la case où figure une croix rouge, dans cinq des huit directions la case adjacente est vide. L'algorithme continue son exécution. Dans deux des huit directions la case adjacente comporte un pion de la couleur active (noir). L'algorithme continue son exécution. En revanche, la case adjacente à droite, comporte un pion de la couleur opposée (blanc), l'algorithme va donc observer la case suivante. cette dernière abritant elle aussi un pion blanc, il va continuer, jusqu'à finir par tomber sur un pion noir ou sortir du tableau.

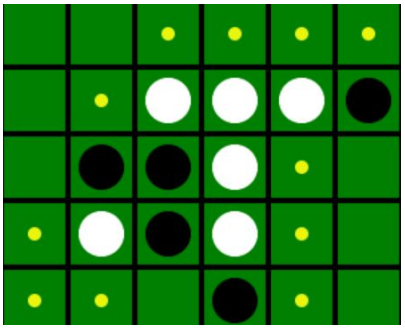


Figure 7: Mise en évidence des coups jouables bis

La case avec une croix rouge est donc une case où un coup peut être joué. Si l'on applique cette méthode d'exploration à tout le tableau, et que l'on met en évidence les cases où un coup peut-être joué, on obtient la figure ci-dessus (*Figure 7*).

Pour ce qui est du bon fonctionnement de l'intelligence artificielle, et de ses différents niveaux de difficulté, l'essentiel reposera sur un algorithme du type minmax. Tout d'abord, expliquons en quoi consiste le minmax, ensuite observons son application au projet Reversi.

L'algorithme MinMax est un algorithme applicable aux jeux à deux joueurs et à somme nulle. Il consiste à minimiser la perte maximum. C'est-à-dire, opter pour un coup, qui minimise la perte de points pour le joueur. Une pseudo intelligence artificielle suivant cet algorithme passe en revue toutes les possibilités pour un nombre défini de coups, et leur attribue une valeur en fonction du bénéfice en matière de points pour elle et pour son opposant. Une fois l'intégralité des coups jouables analysée, le meilleur est celui qui lui garantit une perte minimum tout en supposant que son adversaire cherche à les maximiser. Autrement dit : l'IA considérera que le meilleur coup est celui qui lui fera perdre le moins point en sachant que son adversaire essayera de lui en faire perdre le plus possible

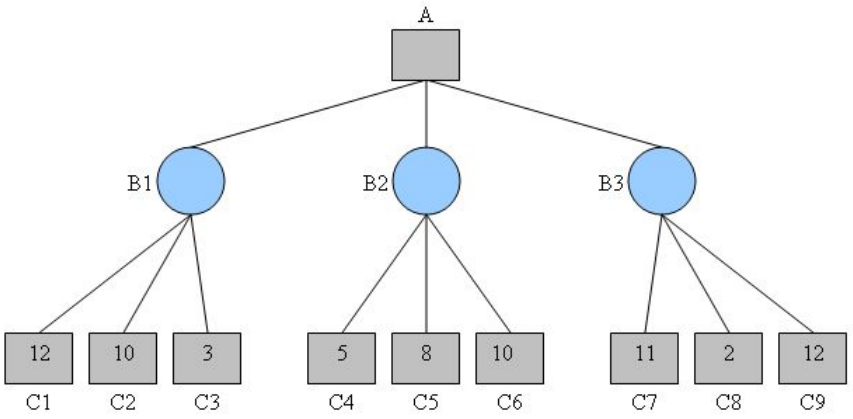


Figure 8: Arbre de décision de l'algorithme minMax

Dans l'exemple représenté ci-dessus, les nœuds gris sont les nœuds du joueur, tandis que les nœuds bleus sont ceux de son opposant. Pour déterminer la valeur du nœud A et le coup à jouer, on choisit la valeur maximum B. Et pour déterminer les valeurs de B1, B2 et B3, il faut prendre la valeur minimum C.

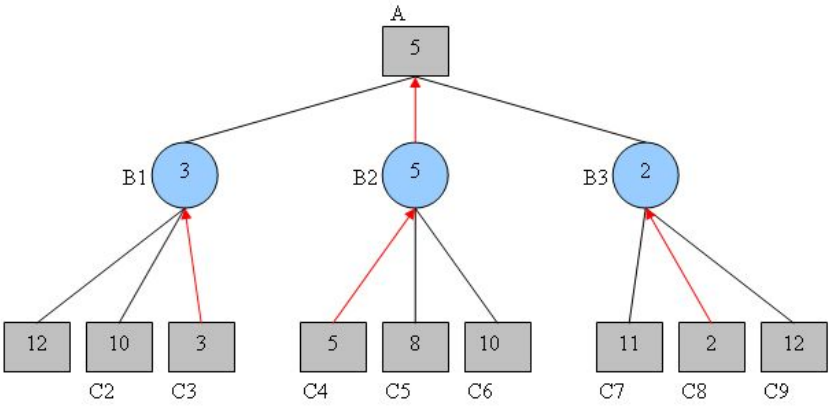


Figure 9: Arbre de décision de l'algorithme minMax bis

Le nœud A prend par conséquent la valeur 5. Le joueur devrait donc jouer le coup amenant au nœud B2. En observant l'arbre ci-dessus, on peut voir que l'algorithme considère que l'opposant va jouer de manière optimale et amener au nœud rapportant le moins de point : C4. Sans cette considération, le choix du joueur se porterait vers les nœuds C1 ou C9 qui proposent un plus grand gain. Dans ce cas-ci, le joueur devrait jouer le coup amenant au nœud B1, mais prend alors le risque que l'opposant joue un coup amenant amenant au nœud C3, et donc un gain pour le joueur de seulement 3.

Cet algorithme, appliqué au reversi, permettrait d'offrir à l'utilisateur plusieurs niveaux de difficulté. Le niveau de l'IA à laquelle il serait opposé correspondrait à la profondeur de l'arbre, et donc au nombre de coups prévisibles à l'avance par cette dernière.

Observons maintenant à quoi ressemble l'algorithme minmax en matière de code, avec ce pseudo-code que nous suivront pour programmer l'intelligence artificielle du projet.

```
int fonction minimax (int depth){
  if (game over or depth = 0){
    return winning score or eval();
   }
  int bestScore;
  move bestMove;

  if (nœud == MAX) {                    //type MAX = Programme
    bestScore = -INFINI;
    for (each possible move m) {
      make move m;
      int score = minimax (depth - 1)
      unmake move m;
      if (score > bestScore) {
        bestScore = score;
        bestMove = m ;
      }
    }
  } else {                              //type MIN = adversaire
    bestScore = +INFINI;
    for (each possible move m) {
      make move m;
      int score = minimax (depth - 1)
      unmake move m;
      if (score < bestScore) {
        bestScore = score;
        bestMove = m ;
      }
    }
  }
  return bestscore ;
}
```

Figure 10: pseudo-code de l'algorithme minMax

Comme on peut le voir à l'observation de ce code, l'algorithme minmax est un algorithme récursif découpé en trois parties : le cas de base, le Max, et le Min.
Le cas de base repose sur deux conditions indépendantes. Il met fin à l'algorithme en retournant le meilleur score si la profondeur est arrivée à 0, ou si les conditions de fin de partie sont réunies.

```
if (game over or depth = 0){
    return winning score or eval();
}
```

Ensuite, les parties Max et Min sont très similaires, mais diffèrent sur la considération du meilleur score. Dans le Max, les coups analysés sont ceux du programme, on considère donc que le meilleur score est le plus élevé (maximum). Tandis que dans le Min, ce sont les coups adverses qui sont évalués, on estime donc que le meilleur score est le plus faible (minimum). Cela se traduit en pseudo-code par :

Dans le Max

```
if (score > bestScore) {
        bestScore = score;
        bestMove = m ;
}
```

Dans le Min

```
if (score < bestScore) {
        bestScore = score;
        bestMove = m ;
}
```

Le Max et le Min sont les parties qui analysent toutes les possibilités de jeu à partir de l'état actuel de la partie. Elles suivent le principe suivant: à partir d'une position, on génère tous les coups possibles. Puis à partir de ces nouvelles positions (niveau 1) on génère toutes les réponses possibles et la profondeur diminue d' un (niveau 2). On recommence l'opération jusqu'à tomber sur le cas de base (profondeur = 0) qui définit alors le score et la valeur du nœud.

Observons maintenant une autre partie importante du programme : les fonctions d'évaluation. Parmi elles, deux seront primordiales : la fonction de test de fin, et la fonction qui compte les pions. Ces fonctions permettront d'arrêter la partie quand elle se termine, et de définir un gagnant. La fonction de compte des pions pourra également être utilisée pour évaluer le score dans le minmax. Ces fonctions ne présenteront à priori que très peu de difficulté en matière de réalisation, mais ne devront en aucun cas être omises, sans quoi le reversi et l'intelligence artificielle ne pourront pas être fonctionnels.

Commençons par la fonction qui compte les pions. Cette fonction permettra d'indiquer précisément le nombre de pions noirs et de pions blancs sur le plateau, et le nombre de cases vides.
Le plateau étant représenté par un tableau rempli de 1 (pion intelligence artificielle), de 2 (pion joueur) et de 0 (case vide), compter le nombre de pions noirs revient a compter le nombre de 2 dans le tableau. De même, pour compter les pions blancs il suffit de compter le nombre de 1. Et pour avoir le nombre de cases vides, on peut donc soit compter le nombre de 0 dans le tableau, soit soustraire le nombre de pions noirs et de pion blanc à 64 (nombre de case total 8 x 8).

Continuons ensuite avec la fonction qui vérifie si la partie doit se terminer ou pas. Pour rappel, une partie de reversi se termine si plus aucun des deux joueurs ne peut jouer, donc dans deux cas :

- Il n'existe plus de coup jouable pour aucun des deux joueurs
- Il n'existe plus de case vide

Plutôt que d'écrire une fonction qui vérifie ces deux conditions, il est plus simple de remarquer que si les fonctions d'explorations fonctionnent correctement, il ne doit plus y avoir de coup jouable s'il n'y a plus de case vide. Vérifier cette unique condition sera donc suffisant.

3. Résultat

Au terme de ce projet, l'objectif principal est atteint, et le jeu de plateau reversi est fonctionnel et utilisable sur navigateur. Il propose à l'utilisateur d'affronter une intelligence artificielle reposant sur un algorithme de minmax, capable d'offrir plusieurs niveaux de difficulté (5 niveaux).

Cependant, l'objectif secondaire qui était de mettre en réseau ce jeu pour permettre à deux utilisateurs de s'affronter, n'a pas pu être atteint dans les temps. Il a été décidé en début de projet, de travailler en javaScript pour pouvoir, si le temps nous le permettait, octroyer à l'utilisateur la possibilité d'affronter un autre utilisateur.

Bien que le but principal de ce projet ait été atteint, il reste des pistes d'amélioration. Trois points peuvent être améliorés pour proposer un jeu Reversi plus qualitatif que la version actuelle de ce projet.

- Continuer à travailler et finaliser la mise en réseau : cet objectif n'a pas pu être atteint, mais représente un enjeu intéressant, en particulier pour l'utilisateur, car affronter une intelligence artificielle est amusant, mais affronter un autre joueur, c'est mieux.
- Améliorer l'aspect esthétique du jeu : l'interface qui accompagne le programme est des plus basiques, et elle serait plus agréable à manipuler si elle présentait plus de détails et d'éléments. En l'état actuel, le plateau n'est qu'une grille verte sur lequel des cercles noirs et blancs sont disposés en guise de pions. Il serait bien plus attrayant de jouer sur un plateau qui en a l'apparence, avec des pions animés lorsqu'ils sont posés ou retournés.
- Parfaire le minmax : pour déterminer le meilleur coup jouable (ou le "moins pire"), l'algorithme minmax génère et évalue chaque coup. Une coupe Alpha-Beta éviterait une si longue exécution. L'Alpha-Beta utilise la recherche déjà effectuée lors du début de la recherche minmax pour borner les variations possibles et ainsi ébrancher des sous-arbres de la recherche.

Installation

Une fois le dossier téléchargé, il suffit d'ouvrir le fichier "index.html" dans un navigateur supportant HTML5 pour lancer le jeu.

Tests et Validation

Ne disposant pas de batteries de tests automatisés, ils furent tous effectués à la main. Ces tests furent mis en place en créant des situations particulières sur le tableau initial, par des affichages console et des traces.

Manuel d'utilisation

Un fois "index.html" ouvert dans un navigateur, l'utilisateur peut choisir un niveau de difficulté en cliquant sur le bouton correspond puis démarrer la partie en cliquant sur "Start". La partie peut être lancée en cliquant uniquement sur "Start", le niveau par défaut de l'adversaire est 1.

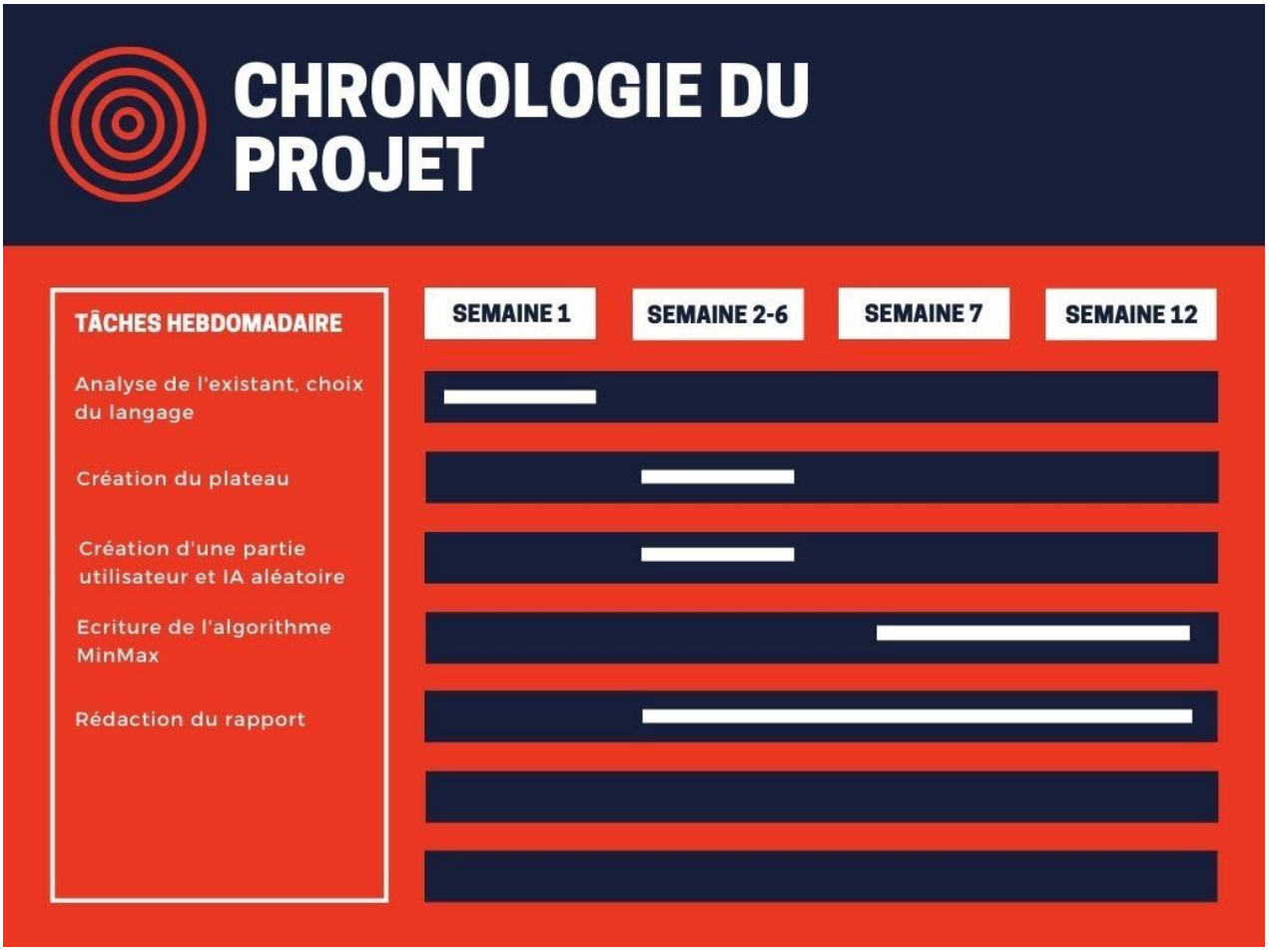
La partie lancée, le tableau s'affiche avec les quatre pions initiaux, le joueur commence, il n'est possible de cliquer que sur les cases avec des points jaunes, ils représentent les coups valides, après un léger délai permettant de voir les conséquences du coup joué, l'intelligence artificielle joue son tour. La partie se termine lorsque le plateau est plein, que l'un des joueurs n'a plus aucun pion, ou que plus aucun coup n'est possible. Dans le cas où un des joueurs ne peut pas jouer de coup valide mais que la partie n'est pas finie, son adversaire joue deux fois.

4. Rapport d’activité

Planification

Avant le commencement d’un projet il faut d’abord avoir un plan d’action, une vision de comment va se dérouler le projet. C’est pour cela que nous avons découpé notre projet en différentes parties/étapes afin de mieux visualiser ce dernier. La première et pas des moindres fut l’analyse de l’existant qui nous a permis de mieux comprendre comment fonctionnait le Reversi et comment l’utilisateur interagissait avec le jeu. Nous nous sommes basés sur les différentes versions de Reversi trouvables sur le net ou encore sur des applications mobiles et principalement le projet de Kylian Thezenas datant du lycée, qui consistait à concevoir un Reversi en Python. Son travail fut une base solide pour le commencement du projet et nous a permis de démarrer dans de bonnes conditions tout en nous focalisant sur le sujet principal de ce projet, l’IA. Pour finir avant le début du projet nous nous sommes concertés afin de décider quel langage nous allions utiliser pour coder le Reversi.

Sur la figure suivante (*Figure 11*), on peut voir la présentation de la planification du projet à l’aide d’un diagramme de Gantt. On y retrouve les différentes étapes citées précédemment telles que l’analyse de l’existant, la création du plateau et d’une partie avec une intelligence artificielle qui joue aléatoirement, l'écriture de l’algorithme MinMax et la rédaction du rapport de projet.



.

Figure 11: Diagramme de Gantt du projet

Organisation

En ce qui concerne l'organisation du travail pour le projet Reversi, il s'est organisé sous la forme de sprints. trois outils se sont donc révélés indispensables:

- un github, pour partager le code, et l'avancement du projet
- un serveur discord, pour pouvoir se réunir, travailler en groupe et échanger
- un un google document pour pouvoir travailler tous ensemble sur le rapport de projet.

De plus, afin de ne pas s'engager sur une mauvaise voie et respecter au mieux les attentes de la tutrice, tout le groupe se réunissait lors d'audioconférence (skype) avec Mme Hisler chaque semaine. Ces réunions ont eu pour but de faire état de l'avancement du projet, fixer les nouveaux objectifs, répondre aux éventuels questionnements et s'assurer que le groupe suivait la bonne direction.

Au début du projet, tous les membres du groupe se réunissaient à des heures définies pour travailler ensemble. Cependant, au fur et à mesure de l'avancement, et chacun étant pris par ses obligations, ces regroupements n'ont plus eu lieu aussi régulièrement qu'au départ.

Il a donc été décidé de définir après chaque réunion avec la tutrice, les objectifs de la semaine. Et chacun tenterait de travailler dans son temps libre à la réalisation de ces objectifs.

Le premier sprint a été consacré à l'analyse de l'existant et des besoins. Le second était le début du travail de programmation avec pour objectif la conception du jeu. Le troisième et dernier sprint fut consacré à l'étape principale du projet : l'intelligence artificielle et l'algorithme minmax.

La rédaction du rapport a commencé à partir du deuxième sprint, pour ne pas prendre de retard et avoir à y consacrer un sprint entier.

Rétrospective sur le déroulement du projet

Le projet Reversi a duré environ 12 semaines. Cependant, toutes les semaines n'ont pas pu être consacrées entièrement au projet à cause des obligations de chacun des membres du groupe. Par conséquent, on peut estimer que le projet est resté sans avancée pendant une durée totale d'environ 2 semaines. De plus, les conditions sanitaires actuelles ont contraint les membres du groupe à être séparés, et de travailler en télétravail sur toute la durée du projet, compliquant leurs échanges, et leur collaboration. Les conséquences de ces conditions de travail particulières ont été une disparité au sein du groupe dans l'assiduité à la tâche, (certains membres ont fourni un travail conséquent tandis que d'autres ont eu plus de difficulté à apporter leur participation au projet), et que certains des objectifs fixés au départ n'ont pas pu être atteints.

Nonobstant, malgré les difficultés rencontrées, le but principal de ce projet, à savoir : concevoir une pseudo intelligence artificielle reposant sur un algorithme minmax intégré à un Reversi, a été rempli. Certes, tout ne s'est pas déroulé comme il l'avait été imaginé au départ de ce projet, mais le groupe a tout de même su s'entendre et s'accorder autour de ce travail commun, afin de produire un résultat.

Conclusion

En conclusion, bien que certains objectifs envisagés initialement n'ont pas pu être remplis par manque de temps, notre groupe est satisfait des résultats de ce projet puisqu'il nous a permis de développer nos compétences à la fois en matière de programmation mais aussi de réflexion.

En effet, il a été choisi pour ce projet de coder en javaScript, un langage qui était jusqu'alors encore inconnu aux membres du groupe, et qui s'est avéré très intéressant à prendre en main. Nous avons également pu découvrir des algorithmes applicables à la théorie des jeux comme le minmax ou encore la coupe Alpha-beta. Ces algorithmes célèbres et particulièrement utiles dans des cas de jeu à somme nulle sont compliqués à comprendre, concevoir et appliquer, mais très enrichissant à découvrir.

Ensuite en termes de réflexion, ce projet s'est montré très instructif. Prévision, anticipation et adaptation ont été les mots-clés de ce travail de groupe, ou aucune partie ne devait être laissée au hasard.

Enfin, nous n'avons pas la prétention d'avoir su répondre à la problématique dans son intégralité, mais nous sommes convaincus que le travail fourni n'est pas négligeable, et n'est qu'une étape dans le développement de nos compétences et de nos connaissances. Nous estimons également que ce projet mérite une étude plus approfondie, et de poursuivre sa réalisation.

Bibliographie

B. Causse, R. Delorme, S. Nicolet “Algorithmes (ou comment pensent les ordinateur)”
Fédération Française d’Othello https://www.ffothello.org/informatique/algorithmes/

contributeurs "Réseau informatique”
Wikipedia https://fr.wikipedia.org/wiki/R%C3%A9seau_informatique

contributeurs “Algorithme minimax”
Wikipedia https://fr.wikipedia.org/wiki/Algorithme_minimax

contributeurs “Algorithme minimax”
Wikipedia https://fr.wikipedia.org/wiki/Algorithme

rédacteurs “Dictionnaire Français”
L’internaute : https://www.linternaute.fr/dictionnaire/fr/definition/reseau-informatique/

Wandrille Krafft “UN MORPION EN JAVASCRIPT”
VonKraft : https://vonkrafft.fr/tutoriels/un-morpion-en-javascript/ 13/12/2013

contributeurs ”Utilisation de base des canvas”
developer.mozilla.org
https://developer.mozilla.org/fr/docs/Web/API/Canvas_API/Tutoriel_canvas/Utilisation_de_base 13/11/2020

contributeurs “Javascript”
developer.mozilla.org : https://developer.mozilla.org/fr/docs/Web/JavaScript 10/06/2020