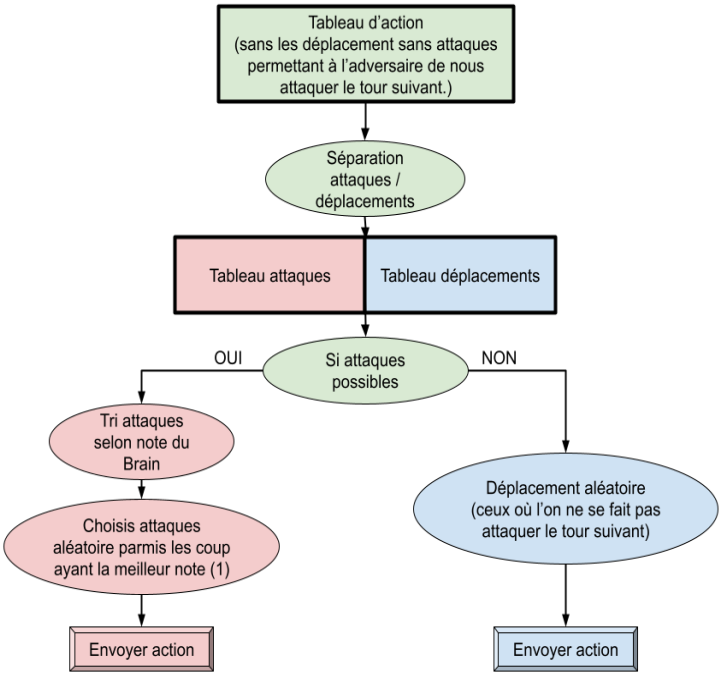
Rapport Intelligence artificielle Lowatem

1. IA basée sur les statistiques.

Pour notre première intelligence artificielle, nous avons décidé de la baser sur un tableau de données généré sur Libre Office. Ce tableau contient de nombreuses informations, comme les points de vie de l'attaquant et du défenseur après une attaque et les dégâts subis par chacun lors de cette attaque, en fonction des points de vie des deux unités. Il contient aussi des booléens permettant de savoir si l'attaquant ou le défenseur est mort de l'attaque, si l'attaquant ne subit aucun dégât ou encore si les dégâts faits lors de l'attaque sont supérieurs au dégâts reçu. Suite à ce tableau, nous avons dressé une liste de coups, du plus efficace au moins efficace :

1. L'attaquant ne subit aucun dégât et le défenseur meurt.
2. L'attaquant ne subit aucun dégât.
3. Le défenseur meurt lors de l'attaque et l'attaquant survit.
4. Les dégâts infligés sont supérieurs au dégâts reçu, sans mort de l'attaquant.
5. Les dégâts infligés sont supérieurs au dégâts reçus, mais l'attaquant meurt.
6. Les autres coups possibles donnant le moins d'avantages.

À chaque situation entre un soldat attaquant et un défenseur correspond donc une "note", allant de 1 à 6. Ces valeurs sont stockées dans un tableau d'entiers, nommé "Brain". Pour utiliser ce Brain, nous avons stocké les notes des coups avec en indice de ligne, 10 fois les points de vie de l'attaquant + les points de vie du défenseur, ce qui correspond à une seule situation.

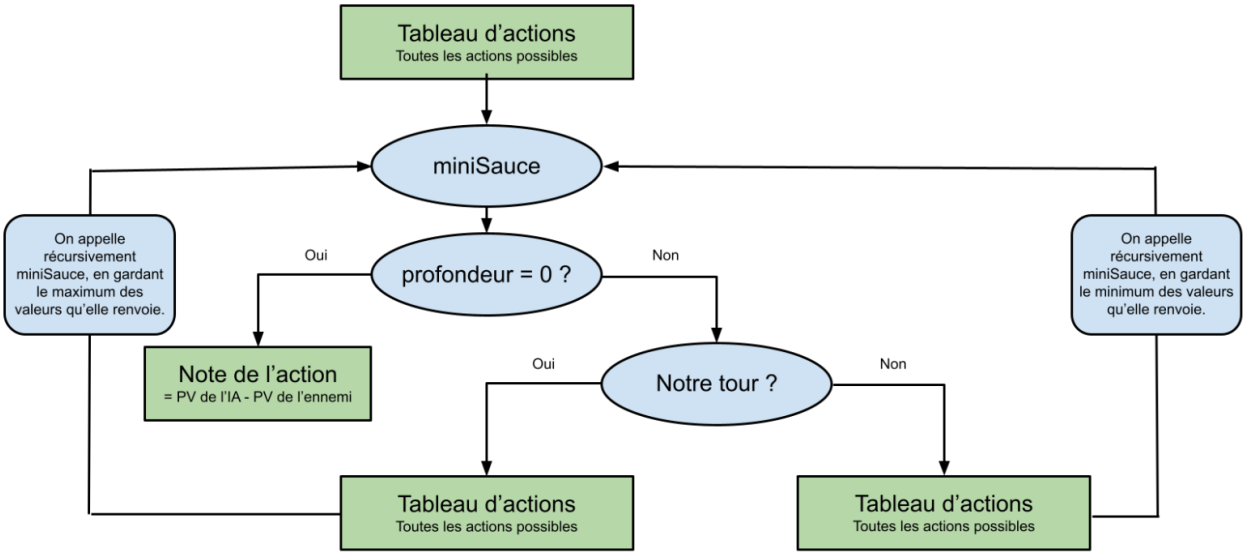


Notre stratégie est donc d'attaquer en priorité. Dans le cas où aucune attaque n'est disponible , nous choisissons de nous déplacer aléatoirement sur une case où nous ne pouvons pas nous faire attaquer le tour suivant. Pour cela, lors de la construction du tableau d'action possible, dans la classe JoueurLowatem, nous n'ajoutons pas les actions où l'unité qui se déplace se fait attaquer au tour suivant. Nous appliquons donc un filtre sur nos tableaux d'action possibles pour ne pas a avoir à analyser des action jugées inutiles

- Avantages : L'algorithme est assez simple et assez rapide. Il est donc assez simple à modifier ou à débugger. La stratégie est facile à comprendre. De plus, il n'y a pas de gros calculs, dans l'algorithme. ce sont juste des données stockées dans le brain. Nous ne faisons donc que des comparaisons.

- Inconvénients : L'efficacité de l'IA dépend grandement de notre compréhension du jeu, et de notre propre définition d'un "meilleur coup". De plus, notre stratégie n'est pas la meilleure pour le calcul de point, basé sur la maximisation de l'écart des points de vie plutôt que sur les dégâts infligés. Cette IA cherchera plus à faire le maximum de dégâts, en minimisant ses pertes, mais sans aller voir les futurs coups.

2. IA basée sur l'algorithme MiniMax.



Pour notre seconde intelligence artificielle, nous avons décidé de rechercher récursivement le meilleur coup possible, en "regardant dans le futur", c'est-à-dire en simulant les situations possibles en fonction du plateau et du dernier coup joué, et ce à répétition. La fonction miniSauce prend en paramètre le plateau "actuel", le coup qui va être joué, la profondeur à laquelle on veut analyser la partie, si c'est au tour de l'IA, et la couleur du joueur jouant l'action.

L'algorithme a trois parties :

- Si la profondeur est à 0, on évalue simplement l'état de la partie : plus on a de points par rapport à l'adversaire, mieux c'est.
- Sinon, si c'est à notre tour de jouer, on évalue toutes nos actions possibles, et on appelle miniSauce afin de trouver laquelle est la meilleure d'entre elles.
- Sinon enfin, c'est à l'adversaire de jouer : on évalue ses actions possibles, et trouve celle qui nous mettrait le plus en difficulté au niveau de la différence de points de vie, à l'aide de miniSauce. On suppose que l'adversaire choisira cette action quand cela sera à lui de jouer.

Cet algorithme vise donc à réduire le plus possible les dégâts que l'on peut recevoir de la part de l'ennemi, tout en maximisant les dégâts que l'on lui fera.

- Avantages : Cette IA est supposée être relativement performante par rapport à son apparente simplicité.

- Inconvénients : miniSauce étant une fonction récursive, nous sommes plus aptes aux erreurs de compréhension de l'algorithme. De plus, l'efficacité de l'IA dépend fortement de facteurs externes : si l'adversaire ne joue pas de la manière que l'on suppose, chaque action de notre IA est alors peu optimale. L'algorithme que nous utilisons est aussi particulièrement lent; nous devons donc limiter la profondeur utilisée afin de ne pas dépasser le maximum de 2 secondes pour envoyer les actions de notre IA.

3. Choix de l'IA déposée pour la fin du projet :

Pour le rendu projet, nous avons choisi de déposer notre IA basée sur les statistiques. Nous considérons en effet que cette IA est plus intéressante ,par son fonctionnement, que l'IA miniSauce, moins originale. Nous avons en effet constaté qu'une partie conséquente des étudiants ont choisi de créer une IA basée sur l'algorithme miniMAx, et nous préférons donc nous démarquer en proposant une IA basée sur des calculs et un algorithme que nous avons réalisés.

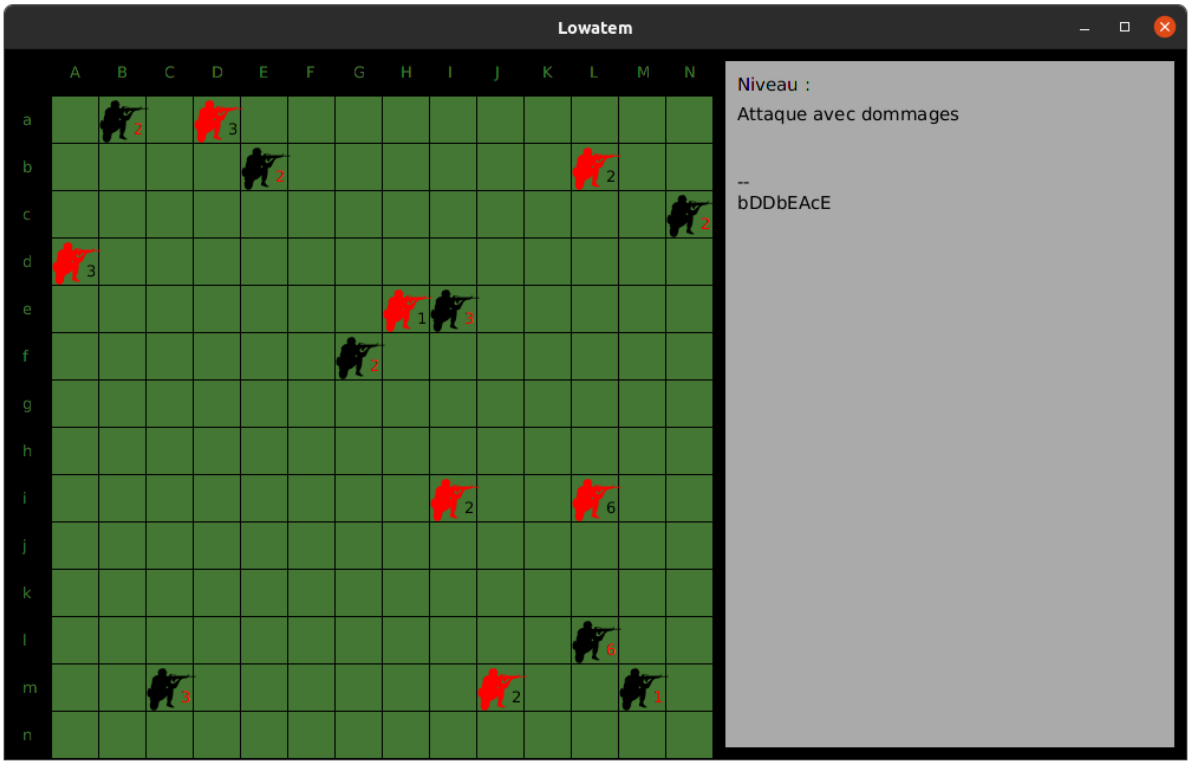
Nous avons réalisé de nombreux tests sur l'IA Brain, et elle a participé à plusieurs simulation, ce qui nous a permis de l'améliorer au fil du temps, en essayant de corriger nos stratégies. Cependant pour l'IA miniSauce, les tests ont été difficiles, et peu concluants, rendant l'amélioration de l'IA difficile. Comme elle ne marchait que très peu au départ, nous n'avons pas jugé utile de la faire participer aux simulations, jusqu'à ce qu'elle puisse tenir une partie entière. Après de nombreux changements dans la manière dont l'IA fonctionne, et une réécriture complète du code de l'IA, nous n'avons pas eu la possibilité de conduire assez de tests et de participer à une simulation pour évaluer son efficacité.

Nous n'avons donc pas pu faire assez de tests pour affirmer que l'IA miniSauce était meilleure que l'IA Brain. Cette IA avait une fâcheuse tendance à utiliser des actions qu'elle n'était pas censée utiliser, pour des raisons que nous ne comprenions pas, notamment des déplacements de cases vides. Nous avions aussi du mal à adapter l'algorithme miniMax au jeu. Par ailleurs, le debuggage et l'amélioration de cette IA sont très difficile, car la récursivité nous ne sommes que peu familiers avec les algorithmes récursifs, une notion qui n'a été abordée que dans des contextes simples en cours.

Pour conclure, nous avons commencé par implémenter l'IA Brain avant l'IA miniSauce. De ce fait, nous avons préféré améliorer l'IA Brain au maximum avant de faire l'IA miniSauce. L'IA Brain a donc eu plus d'expérience et d'amélioration, et nous trouvions qu'il était plus intéressant de travailler dessus.

Portfolio Thibaud :

Ce projet d'implémentation des règles du jeu Lowatem, ainsi que la création d'une intelligence artificielle pour le jeu était pour moi une grande nouveauté. En effet mes projets précédents étaient surtout accès sur de l'utilitaire, et non pas sur le ludique. C'était aussi mon premier projet conséquent en Java, ce qui m'a permis de consolider mes bases dans ce langage, principalement sur le plan de l'utilisation de classes et de méthodes.



Portfolio Tom:

Lors de ce projet, j’ai appris à développer un jeu, étapes par étapes, en ajoutant des règles petit à petit. J’ai donc appris à analyser et adapter mon code pour y ajouter des fonctionnalités. Ensuite, lors du développement de l’IA (ma première), j’ai appris à analyser la situation, penser à une stratégie gagnante et à répartir le travail dans mon binôme avant de développer sur machine. Cette phase de réflexion me permettra plus tard de mieux mettre en place mes futurs projets.