Projet d'IA pour les jeux : Développement d'une IA pouvant jouer à Battle For Wesnoth

Nolhan Dumoulin
Alexandre Mahjoub

2022-2023

Introduction

Ce rapport est fait dans le cadre du projet du module d'IA pour les jeux du département Informatique de l'INSA de Rennes. Il a pour but de présenter le déroulement du projet, nos choix durant son déroulement, les problèmes auxquels nous avons été confrontés, etc...

1 Présentation du sujet

1.1 Sujet

Le but de ce projet est pour nous d'utiliser nos connaissance acquises durant le modules sur les différents systèmes d'intelligence artificielle pour pouvoir l'appliquer à un jeu que nous connaissions. Notre but a donc été tout d'abord de trouver un jeu sur lequel nous pouvions appliquer une des solutions d'IA. Mais nous ne voulions pas non plus que le jeu sur lequel nous allions développer notre IA soit trop simple à résoudre. La première étape pour nous a donc été de chercher un jeu adapté à ces contraintes.

1.2 Le jeu

Afin de réaliser nos objectifs, nous avons donc décidé de nous projeter sur un jeu de stratégie : Battle for Wesnoth. Ce jeu, paru le 18 juin 2003, a été développé en opensource par des développeurs indépendants. C'est un jeu de stratégie au tour par tour opposant deux joueurs.

Le jeu consiste en une alternance de tours où chaque joueur pourra déplacer ses unités, attaquer les unités adverses ou alors en recruter de nouvelles sous certaines conditions développées plus bas. Les cartes sont constituées de cases hexagonales possédant des caractéristiques de terrains qui vont favoriser certaines unités. Ces cases peuvent aussi contenir des châteaux et des villages. Vous pourrez voir un exemple de carte à la figure 1.

FIGURE 1 – Exemple d'une carte de Battle For Wesnoth

Chaque joueur possède une liste d'unités parmi lesquelles se trouve pour chacun une unité héros. Cette unité est la seule unité à pouvoir recruter d'autres unités (plus faibles que le héros). Pour ce faire, celle-ci doit se situer sur l'une des cases château du plateau. Recruter des unités demande aussi un certain nombre de pièces d'or. Ces pièces d'or peuvent s'obtenir en capturant des villages. Chaque village permet aux joueurs, s'ils en capturent, de produire de l'argent à chaque tour de jeu. Pour avoir un ordre d'idée, à l'initialisation de notre implémentation, chaque joueur possède 100 d'or et en produit deux par tour. Cette production peut ensuite augmentée de 1 pour chaque village capturé par les joueurs.

Les unités obtenues ainsi que le héros peuvent se déplacer d'un nombre de cases limités (dans notre cas, ce sera de 2 cases). Si une unité ennemie est présente sur la case ciblée alors un combat s'engage et cette-dernière subira des dégats. Si elle n'a plus de PV, elle disparaîtra. Ensuite, si l'unité sélectionnée se déplace sur une case village, celui-ci sera capturé à la fin du tour. A chaque tour, les joueurs n'ont droit qu'à un seul mouvement (attaque, déplacement ou formation de troupes). Au final le gagnant est celui qui parvient à éliminer l'unité héros adverse.

2 Nos choix

2.1 Réalisation du jeu

Pour pouvoir éditer l'IA du jeu, nous avons décidé de ne pas nous baser sur le code opensource du jeu mais de réaliser notre propre jeu simplifié qui nous permettra de développer notre propre IA plus aisément.

2.2 Langage de programmation

Afin de réaliser cela, nous avions besoin de choisir le langage avec lequel nous allions développer notre jeu et son IA. Nous avions tout d'abord pensé à utiliser java que nous maîtrisions bien. Il nous a cependant été montré que C++ pouvait être plus adapté à nos problématiques. Il est plus efficace, moins gourmand en ressource et plus adapté aux jeux de stratégie comme le nôtre. De plus, comme nous aimerions travailler dans le jeu vidéo, nous nous sommes dit qu'il serait utile d'avoir cette compétence car il est très utilisé dans le milieu.

Ce choix nous a causé par la suite des difficultés car nous devions alors non seulement découvrir le fonctionnement de l'IA mais aussi celui d'un nouveau langage de programmation. Toutes les premières séances ont donc servi à nous familiariser avec le langage avant de pouvoir bien programmer le jeu.

2.3 Interface graphique

Par la suite, nous pu choisir une interface graphique. Après quelques recherches, nous avons opté pour SFML (Simple and Fast Multimedia Library). Ce choix d'utiliser cette bibliothèque nous permet de ne pas nous préoccuper de certains aspects du développement qui ne nous interresse pas ici. Cela concerne par exemple l'interface graphique et les inputs clavier. Cette librairie a été utile car elle est simple à comprendre et à utiliser tout en ayant une interface légère.

Ce choix nous a aussi fait subir des difficultés d'adaptation à une interface nouvelle pour nous.

3 Développement du jeu

Après avoir appris à bien utiliser C++ et sfml dans le cadre de la programmation d'un jeu, nous sommes parvenu à des versions jouables de notre jeu avec les fonctionnalités et l'interface présentée dans les figures ci-dessous.

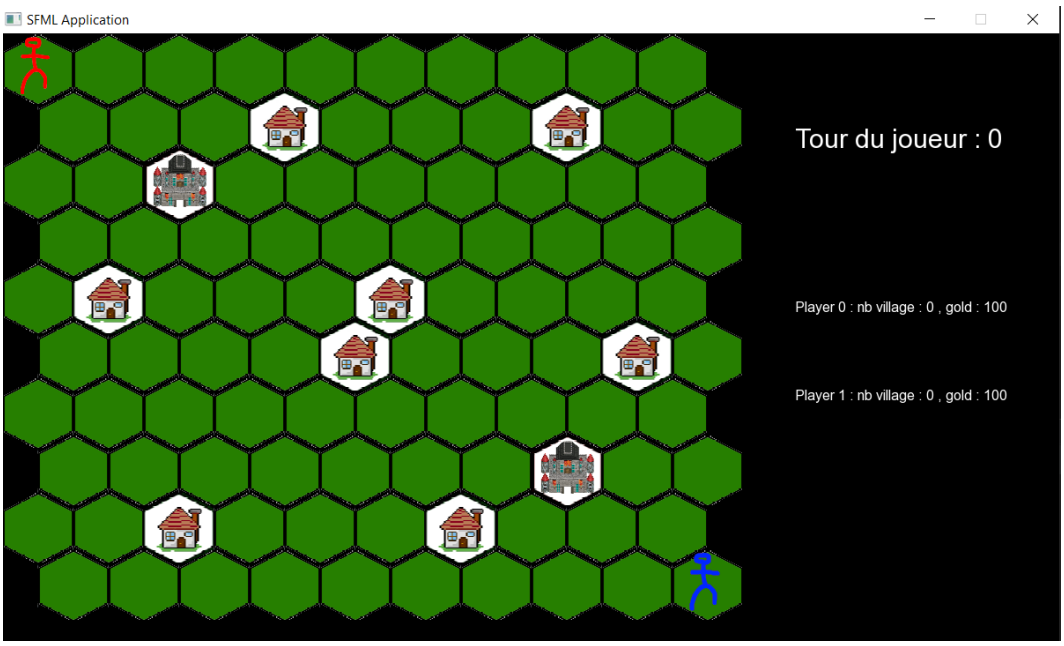

FIGURE 2 – Interface à l'initialisation

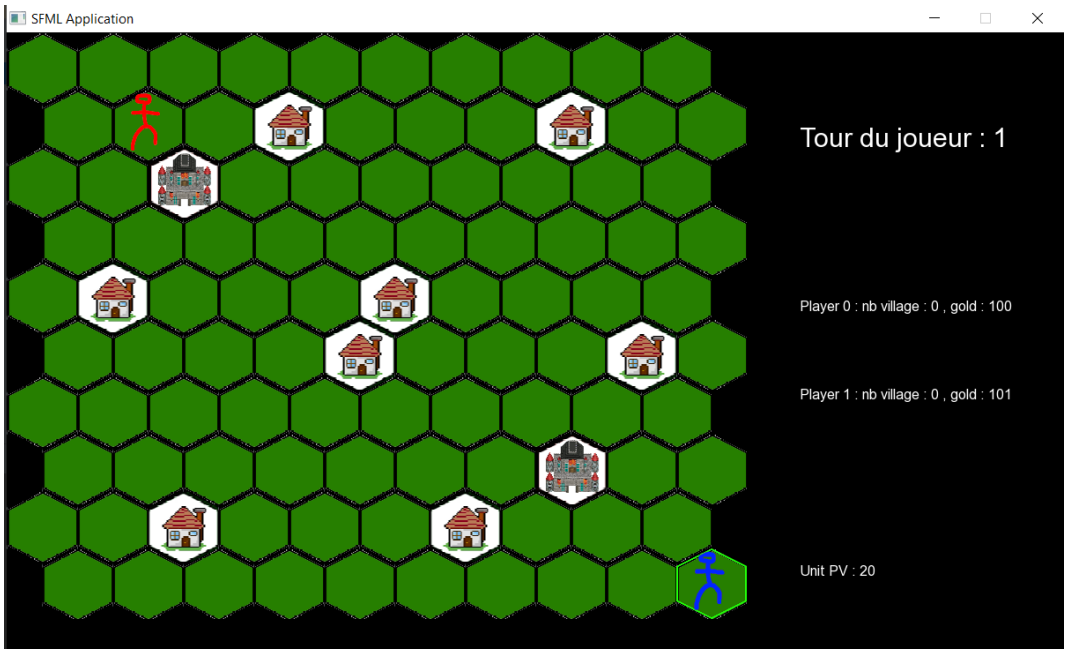

FIGURE 3 – Interface après le déplacement du joueur 0

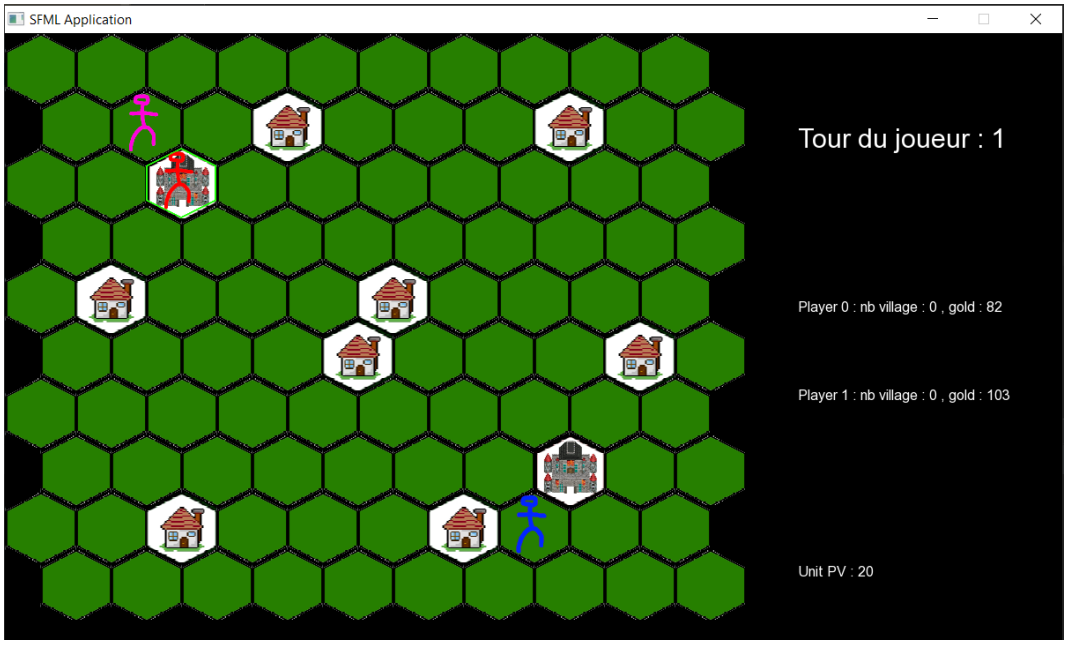

FIGURE 4 – Interface après le recrutement d'une unité par le héros du joueur 0

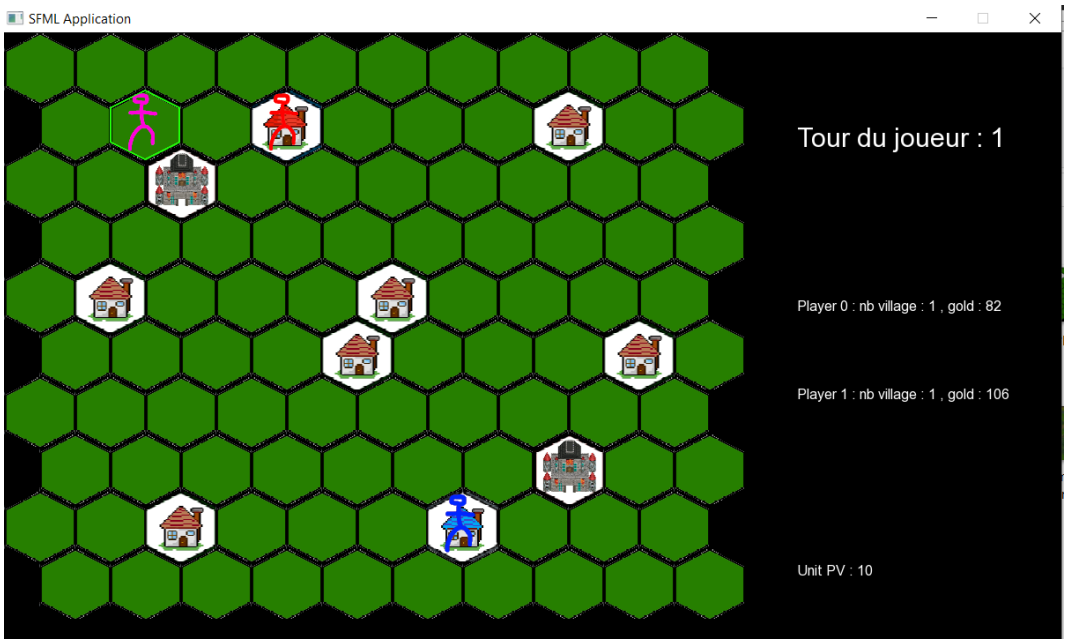

FIGURE 5 – Interface après la capture d'un village par une unité du joueur 1

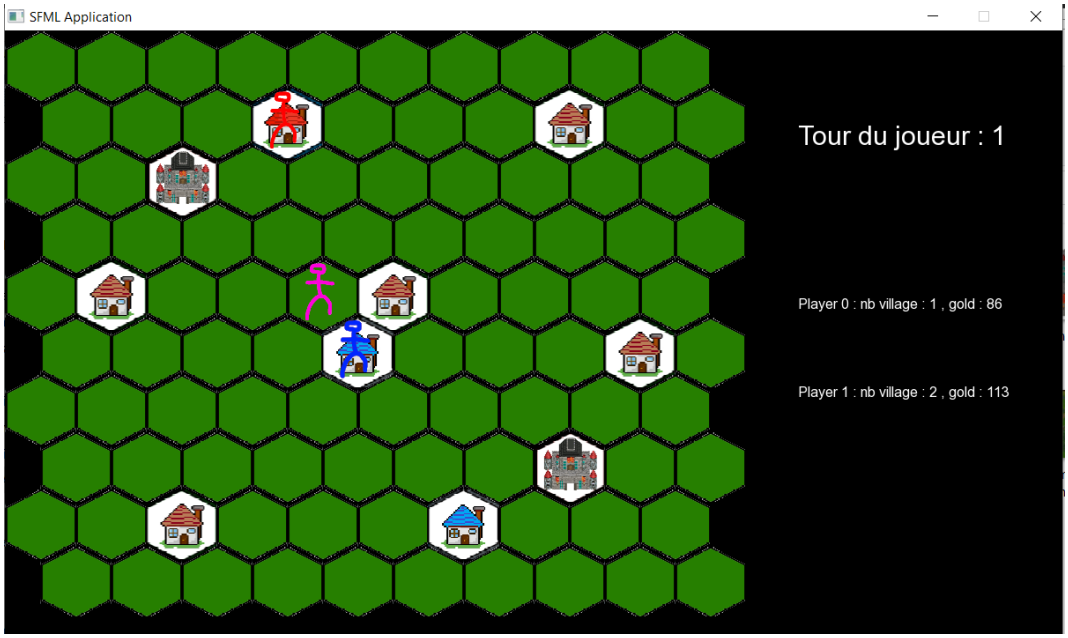

FIGURE 6 – Interface avant l'attaque d'une unité du joueur 0 par le héros du joueur 1

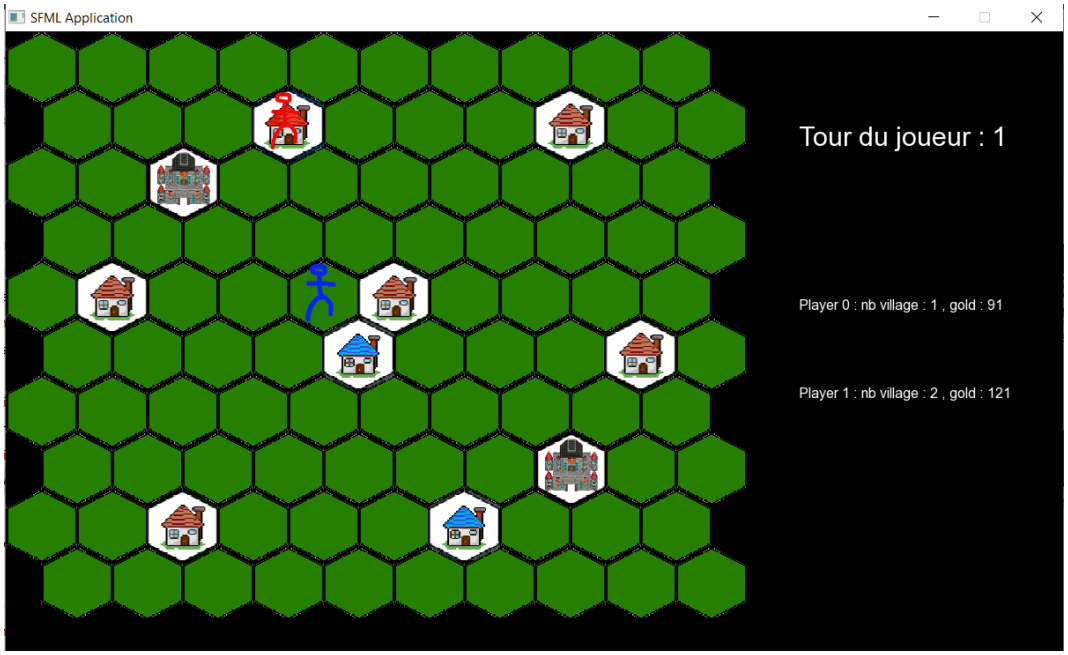

FIGURE 7 – Interface après l'attaque

4 Développement de l'Intelligence Artificielle

4.1 Choix de l'IA à utiliser

Notre sujet est un jeu au tour par tour où il n'y a pas d'informations cachées et un certains nombre d'actions sont légales. Nous avons donc décidé que la meilleure solution était une IA basée sur une arbre de recherche Monte Carlo. Nous avons cependant décidé de réduire l'arbre à un certain nombre d'itérations. En effet, chaque unité pouvant effectuer chacune 20 actions au maximum (19 cases de déplacement dans un rayon de 2 cases autour de l'unité + l'action de création d'unité) la complexité augmente rapidement. Pour évaluer notre position à un certain noeud de l'arbre nous avons utilisé une heuristique prenant en compte la victoire d'un joueur (si un plateau final est atteint) la différence du nombre d'unités entre les deux adversaires, le nombre de villages possédés et le nombre de pièces d'or.

4.2 Difficultés rencontrées

Lors de notre première version du jeu, nous avions orienté notre jeu pour qu'il soit jouable par un humain. Cependant, lorsque nous avons commencé à vouloir implémenter l'IA, de nombreux problèmes de compatibilités se sont présentés. Il nous est alors paru évident qu'il ne serait pas possible de faire du Monte Carlo Tree Search avec notre architecture. Cette version était donc jouable mais dans le cadre de notre projet, n'était pas satisfaisante.

Nous sommes donc repartis à 0 pour coder un programme qui soit parfaitement imbriquable dans un Monte Carlo Search Tree. Pour cela, nous avons beaucoup simplifié les classes du jeu et nous avons pu réaliser notre MCTS. Comme nous débutions en c++ et ne connaisions pas la bibliothèque SFML nous n'avons cependant pas eu assez de temps pour pouvoir perfectionner et optimiser le code et l'heuristique et beaucoup de bugs subsistent encore. De plus, nous n'avons donc pas pu la calibrer, le modèle n'est donc pas encore très efficace.

Conclusion

En conclusion, ce projet nous a permis de découvrir comment implémenter une intelligence artificielle à un jeu. Par la même occasion, nous avons aussi pu aussi découvrir le développement jeu vidéo en C++ à travers la bibliothèque SFML pour simplifier certaines tâches

Nous avons au final une IA qui parvient à jouer à notre jeu malgré des crashs assez réguliers.

Aide utilisateur

Afin de lancer le jeu il faut lancer le main après avoir corrigé les paths des fichiers d'images et configuré la bibliothèque SFML (voir github).

Après vous pourrez jouer en cliquant sur les unités puis sur la case visées, la formation d'unité se fait en appuyant sur la touche N quand votre héro est sur un chateau.