DE MURAT Daniel TIEO Sonia	M1BIB 07/12/2018

RAPPORT: PROJET FANORONA

I/ Introduction

Le Fanorona est un jeu d'échecs malgache. La variante *Fanoron-tsivy*, présente un pleateau avec 5 lignes horizontales , 9 lignes verticales et des diagonales dans chaque carré. Les 22 pions noirs et les 22 pions blancs sont placés sur tous les points d'intersection des carrés excepté celle du milieu.
Le but du jeu est d'éliminer tous les pions de son adversaire du plateau.
Pour se faire, tant que le pion joué peut capturer des poins adverses, il peut enchaîner plusieurs captures au sein d'un même tour. Néanmoins,il doit changer à chaque fois de direction , ne pas revenir sur une place qu'il a déjà occupée.
Le but de notre projet était d'implémenter ce jeu. Dans ce rapport, nous présentons les différents choix que nous avons fait et les difficultés que nous avons rencontrées.

II/ Mise en place du programme

1) Affichage

Concernant l'affichage du plateau, nous avons décidé de le faire en ASCII. Le plateau est d'abord généré comme une matrice de taille HAUTEUR x LARGEUR. Plusieurs fonctions permettent ensuite de remplir la matrice et d'afficher le plateau de jeu:
- une première fonction *initialiser_plateau_vide* remplit la matrice de 0.
- une fonction *placer_pions* place les pions des deux joueurs dans la matrice selon les règles du jeu.
- une dernière fonction *afficher_plateau* affiche la matrice créée et ajoute les diagonales entre les lignes.

2) Mouvements

Pour matérialiser le déplacement des pions sur le plateau, nous avons choisi de créer une structure déplacement, elle-même constituée de structures position pos_i et pos_f, contenant les numéros de lignes et de colonnes du pion sélectionné avant et après son déplacement dans le plateau.

Le joueur est ainsi amené à entrer le numéro de ligne et de colonne du pion qu'il souhaite déplacer puis sélectionne parmi la liste de directions, celle vers laquelle il souhaite déplacer son pion.

La position du pion position finale (après déplacement) est calculée grâce à la fonction *position_suivante*, celle-ci est ensuite testée par plusieurs fonctions pour savoir si le déplacement satisfait bien les règles du jeu:
- La fonction *test_deplacement* teste si, dans le cas où le joueur veut effectuer un déplacement en diagonale, ce déplacement est possible.
- La fonction *est_occupe* teste si, la case du jeu correspondant à la position finale du pion, est bien vide.

- La fonction *mon_pion* teste si, les coordonnées de la case sélectionnée par le joueur correspond bien à l'emplacement d'un de ses pions.

3) Captures

Les captures sont gérées dans notre programme de la manière suivante:

Tout d'abord, le joueur sélectionne un déplacement (comme décrit précédemment), si celui-ci est valide. Le programme teste ensuite si le déplacement engendre une capture par approche (grâce à la fonction *capture_approche*) ou une capture par retrait (grâce à la fonction *capture_retrait*).
Si aucune capture n'est possible, on vérifie grâce à la fonction *capture_possible_plateau* , si un pion du joueur courant peut effectuer une capture. Cette fonction parcourt toutes les positions des pions du joueur courant, et pour chaque direction possible, teste si une capture est possible. Si oui, le joueur doit nécessairement rentrer un nouveau déplacement qui permet une capture.
Si non, le joueur peut se déplacer sans capturer.
La détection d'une capture par approche s'effectue de la manière suivante avec la fonction *capture_approche*. Après une simulation de déplacement du pion joué selon l'utilisateur, les coordonnées de la position d'arrivée sont stockées. A partir de celles ci, un autre déplacement est simulé dans la même direction. Si la position d'arrivée correspond à l'emplacement d'un pion adverse, la fonction détecte une capture par approche possible.

Pour la détection des captures par retrait. Un déplacement opposé à celui entré par le joueur est calculé par la fonction *capture_retrait* à partir de la position initiale du pion et de la direction choisie. Si la position d'arrivée du déplacement est occupée par un pion adverse, une capture par retrait est détectée.

A cela, s'ajoutent les fonctions *pions_manges_ret* et *pions_manges_app* qui permettent de capturer les pions lorsqu'une capture est possible. La fonction *pions_manges_app* re-simule un mouvement dans la direction qui a été entrée jusqu'à ce que la position d'arrivée ne soit plus occupée par un pion adverse et remplace ces positions par 0. De même, la fonction *pions_manges_ret* re-simule un mouvement dans la direction opposée à celui entré jusqu'à ce que la position d'arrivée ne soit plus occupée par un pion adverse et remplace ces positions par 0.

4) Fonctionnement global du programme

Plus globalement, les joueurs doivent entrer leur coup chacun à leur tour. Ils jouent jusqu'à ce qu'un coup valide soit entré. Dans ce cas, le plateau de jeu est modifié en conséquence, le tour se termine et c'est à l'autre joueur de jouer.
Le jeu se termine lorsque le nombre de pions d'un des deux joueurs est égal à 0.

III/ Problèmes non résolus

Parmi les problèmes que nous avons rencontré, nous recensons tout d'abord le contrôle de la saisie au clavier: celle-ci n'est pas sécurisée. Dans l'idéal, l'utilisateur doit entrer dans le terminal les

entiers correspondant aux coordonnées du pion qu'il souhaite déplacer et la direction qu'il souhaitée du mouvement souhaitée. Cependant, si un double ou une chaîne de caractère est donnée en arguments par l'utilisateur au lieu d'entiers, une erreur de segmentation apparaît.

De plus, le programme ne respecte pas entièrement les consignes du jeu car nous n'avons pas réussi à implémenter la capture successive. Nous avions commencé à coder quelques fonctions servant à la capture successive mais cela n'a pas abouti. Nous avions crée une fonction qui prenait en argument la position initiale du déplacement et qui testait les déplacements possibles dans toutes les directions. Pour gérer le fait qu'une position, lors des déplacements successifs , ne soient occupée qu'une fois, nous avions voulu stocker dans deux tableaux dynamiques différents, les lignes et les colonnes prises au cours des déplacements successifs. Ainsi à chaque fois que le joueur rentre une nouvelle position pour un éventuel déplacement successif, cette position est comparée simultanément avec les éléments des deux tableaux lignes et colonnes.

Par ailleurs, nous avons aussi manqué de temps pour créer une option de jeu qui permet de jouer contre l'ordinateur et développer une intelligence artificielle (IA) . Avec davantage de temps, nous aurions opté tout d'abord pour une IA qui choisit les déplacements au hasard, puis pour une IA qui privilégie les déplacements permettant le plus de captures.

Des améliorations auraient pu être apportée en plus, notamment concernant l'affichage du plateau de jeu. Il aurait été plus pratique d'afficher le numéro des lignes et des colonnes du plateau de jeu pour que le joueur puisse plus facilement sélectionner les numéros de lignes et de colonnes du pion qu'il souhaite déplacer.

Enfin, au lieu de faire une boucle qui oblige le joueur à rentrer un pion jusqu'à ce qu'une capture soit possible, le mieux aurait été de faire une ou plusieurs fonctions qui stockent en mémoire les positions des pions du joueur courant qui peuvent capturer, puis de les proposer au joueur.