– Algorithmique & Programmation – D.E.S.S. Imoi –

Projet : jeu de stratégie à 2 joueurs

Le but de ce projet est de vous faire découvrir comment un ordinateur peut être programmé pour jouer "intelligemment" à un jeu de stratégie. On utilisera autant que possible les possibilités offertes par l'héritage en C^{++}, de façon à définir des comportements *génériques* (c'est-à-dire indépendants du jeu de stratégie considéré). Ainsi, ce projet sera découpé en trois étapes :

1. découvrir les algorithmes utilisés pour déterminer le meilleur coup pour chaque situation de jeu (*cf.* article ci-joint) ;
2. définir les classes (et leurs méthodes) nécessaires à la mise en place de ces algorithmes ;
3. programmer ces classes et vérifier leur fonctionnement sur un exemple de jeu de stratégie, l'awélé (*cf.* annexe A en fin d'énoncé).

1 Partie algorithmique

La première partie de ce projet regroupe les deux premières étapes citées ci-dessus. Il s'agit tout d'abord de se familiariser avec les deux algorithmes de recherche du meilleur coup dans une situation de jeu donnée, les différentes évolutions possibles du jeu pour un certain nombre de coups étant évaluées. Les deux algorithmes présentés dans l'article ci-joint s'appellent `min-max` et `alpha-béta`, le second étant une amélioration du premier.

Lorsque ces deux algorithmes seront compris, votre travail consistera à mettre au point un ensemble de classes permettant de définir n'importe quel jeu de stratégie à deux joueurs, de proposer un joueur ordinateur capable d'affronter un adversaire (utilisateur ou programme définit par vous-même ou un autre groupe) et mettant en œuvre ces classes dans le cas de l'awélé (dont les règles sont indiquées dans l'annexe A en fin d'énoncé). Pour ces classes, il faudra préciser l'héritage (le cas échéant) et sa nature (publique, protégé ou privé) ainsi que les différentes méthodes (ou fonctions).

Pour chaque méthode, on devra donner son en-tête soit en C^{++} (par exemple, `char toto(const int) const`), soit clairement (la fonction `toto` prend comme paramètre un argument entier non modifié, et retourne un caractère sans modifier l'objet sur laquelle elle est appelée). Cet en-tête devra être suivi d'un algorithme précisant le fonctionnement de la méthode. Il n'est pas nécessaire de trop détailler cet algorithme, mais il faut y avoir suffisamment réfléchi pour ne pas découvrir ensuite (dans la partie programmation) qu'il ne fonctionne pas.

Cette partie donnera lieu à la rédaction d'un rapport, dactylographié ou **rédigé très proprement** (attention, la présentation entrera en compte pour la note). Ce rapport devra être rendu avant les vacances de Noël. En cas de question quant à la constitution de ce rapport, ne pas hésiter à me contacter.

2 Partie programmation

La seconde partie de ce projet consistera à programmer et à tester les classes mises au point dans la partie algorithmique. Lorsque les rapports synthétisant la première partie auront été tous rendus, un corrigé proposant un ensemble de classes et les entêtes de leurs méthodes vous sera distribué. À l'aide de ce corrigé et du travail que vous aurez effectué, vous devrez développer les classes ébauchées précédemment, et vérifier leur bon fonctionnement en utilisant le jeu awélé (*cf.* annexe A).

Cette partie donnera lieu à l'élaboration d'un dossier regroupant les fichiers entête des classes, leur fichiers source, ainsi que les fichiers de test et leur utilisation : les premiers définissant les fonctions de test, il faut aussi décrire comment utiliser ces fonctions, c'est-à-dire donner de nombreux exemples de tests et leur résultats. Un énoncé précisant votre travail sera distribué avec le corrigé de la première partie.

Attention : la notation de cette partie sera fortement influencée par les deux critères suivants :

- la lisibilité des fichiers en-tête et source, et en particulier la présence et la précision des commentaires,
- la qualité et la variété des tests effectués pour vérifier le bon fonctionnement des classes définies.

A Règles du jeu de l'awélé

L'awélé se compose d'un plateau creusé de deux rangées, appelée *camp* et composée de six trous, et de deux trous, situés hors de ces rangées et appelés *greniers* (*cf.* fig. 1). L'un des camp sera désigné comme le camp nord, l'autre comme le camp sud, et chaque camp se verra attribué un grenier. Chaque trou permet de contenir des graines, le but du jeu étant d'accumuler le plus grand nombre de graines possible dans le grenier de son camp.

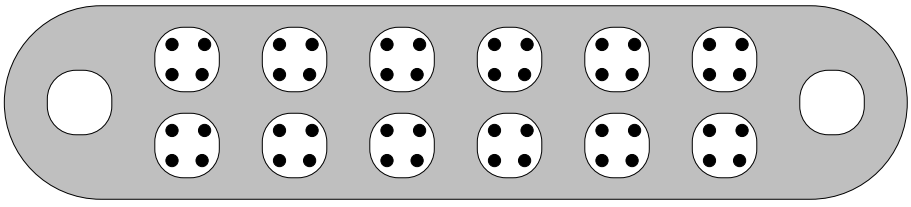

FIG. 1 – *Position de départ des graines*

Au départ, chaque trou des deux camps contient exactement quatre graines (*cf.* fig. 1). Ensuite, chaque joueur choisit **à son tour** un trou de son camp, le vide et **sème** les graines recueillies : il dépose une (et une seule) graine dans chacun des trous (des deux camps) suivants le trou vidé, dans le sens trigonométrique. Lorsque le semis fait un tour complet (ou plus), on saute saute le trou initialement vidé (qui sera donc vide à la fin du semis).

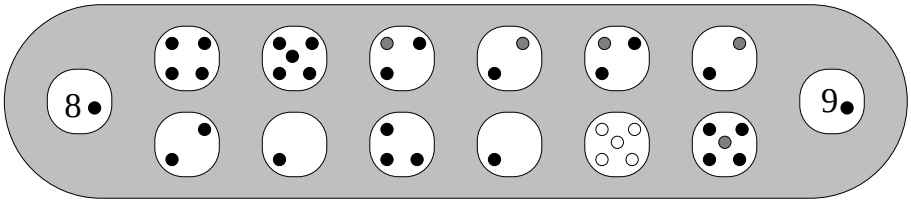

FIG. 2 – *Exemple de semis*

Ainsi, si le camp sud (représenté en bas de la figure 2) choisit de semer les graines de son cinquième trou (ces graines étant dessinées en blanc), il vide ce trou et rajoute une graine dans son sixième trou (cette graine est dessinée en grisé) et une dans chacun des quatre premiers trous du camp nord (ce sont les trous en haut à droite du camp nord).

Ce semis peut permettre de **capturer** des graines et de les entreposer dans son grenier. Lorsque la dernière graine semée a été déposée dans un trou du camp adverse ne contenant à l'origine qu'une ou deux graines (le trou en contient alors deux ou trois en incluant celle semée), le joueur peut alors ramasser les graines contenues dans ce trou et les déposer dans son grenier. En outre, la capture se propage en arrière : si le trou précédent remplit également les conditions de capture (il est situé dans le camp adverse et contient deux ou trois graines en incluant celle semée), il peut aussi être vidé au profit du grenier du joueur, et ainsi de suite jusqu'au premier trou ne remplissant plus ces conditions.

Dans l'exemple de la figure 2, les graines des quatre premiers trous du camp nord peuvent donc être capturées par le camp sud, dont le grenier gagne alors dix graines (*cf.* fig. 3).

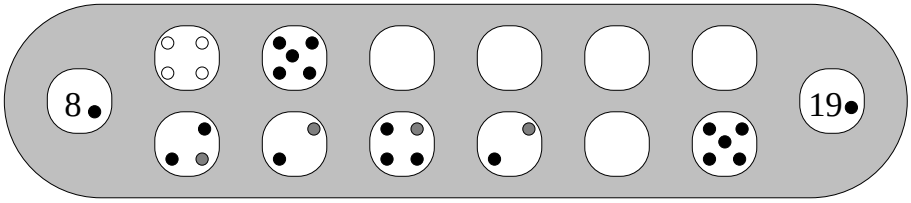

FIG. 3 – *Autre exemple de semis*

Si le camp nord choisit de vider son dernier trou (*cf.* fig. 3), il ne gagnera pour sa part que deux graines, sa capture se limitant à un seul trou.

Cependant, tous les coups possibles ne sont pas **valides** : à chaque tour, le joueur est obligé de nourrir son adversaire, c'est-à-dire qu'il doit laisser au moins une graine en fin de tour dans le camp de son adversaire.

La partie se termine lorsqu'un joueur ne peut plus nourrir son adversaire, ou lorsqu'aucune graine ne peut plus être capturée. Attention : cela peut signifier que la partie cycle sans permettre de capture (ce cas de figure est assez dur à repérer). Chaque joueur compte alors les graines contenues dans son grenier, le gagnant étant celui qui en possède le plus.

Vous pouvez trouver des compléments sur les règles de ce jeu à l'adresse `http://www.loria.fr/~scheuer/fr/Enseignement/AP/awele.html`.