Gold Digger

Projet Java ING3

Objectifs

Gold Digger est un jeu multi-joueurs dans lequel le programme représente un prospecteur d'or dans une concession inexplorée. Il n'est pas seul dans la zone, et devra amasser plus d'or que ses petits camarades en un temps donné.

Le but du projet est d'écrire un programme en Java capable de s'interfacer avec un serveur sur lequel tous les participants seront en compétition sur une surface carrée inexplorée. Il faudra donc créer une intelligence artificielle capable d'évaluer la situation, et d'appliquer la meilleure stratégie possible pour devenir riche.

Règles du jeu

La partie se déroule en un temps limité.
Tous les joueurs commencent avec un score de 0 et un camp de base. A chaque tour, les *prospecteurs* explorent la carte pour trouver des piles d'or, et les ramener au camp de base. Ils ne peuvent en transporter que 10 kilos à chaque fois. On peut également piller les bases des adversaires.
Une fois arrivé à la base, lorsque le *prospecteur* dépose son butin, la moitié est gardée dans les réserves de la base, et l'autre moitié est dépensée. Le score augmente du nombre de kilos ramenés.

A la fin du temps réglementaire, on ajoute au score la taille de la pile de la base de chacun des joueurs, et celui qui a le meilleur score remporte la partie.

Actions possibles

Notation : dans ce qui suit, les commandes, ainsi que les résultats sont notés de la façon suivante
commande <argument 1> <argument 2>... [<argument optionnel 1> <argument optionnel 2> ...] -> résultat

Actions informatives
Pendant le tour, les joueurs peuvent effectuer autant d'actions informatives que nécessaire (dans des limites raisonnables).

- position -> "(x;y)"
  renvoit la position courante du *prospecteur*
- loadout -> "l"
  renvoit la quantité d'or dans le sac
- list -> "joueur1
  joueur2
  ...
  joueurN"
  renvoit la liste des joueurs présents dans la partie
- status <joueurX> -> "S"
  renvoit le score S du joueur appelé "joueurX"
- map -> "(a;b;c;d;e;f;g;h;i)"
  renvoit les informations sur ce qui entoure le *prospecteur*, avec la notation suivante

a	b	c
d	e	f
g	h	i

n indique une pile d'or de taille n
* indique la présence d'un autre *prospecteur*
(x:n) indique la base du joueur x, contenant une pile de taille n
l (pipe ou barre verticale) indique la bordure
. indique un espace hors du plateau
dans le cas où c'est une case vide, on a 0

Actions du tour
Le tour se termine lorsque tout le monde a choisi son action pour le tour. Si jamais deux actions sont conflictuelles (par exemple deux *prospecteurs* essayant d'aller sur la même case), ils se percutent, reviennent à leur position initiale, et font tomber leur or (correspond à un “drop”). "OK" signifie que l'action a été effectuée, "KO" signifie que c'est impossible.
Chaque joueur choisit une et une seule action parmi les possibilités suivantes:

- move D -> "OK" / "KO"
  D ∈ {"N","S","E","W"}. Bouge d'une case dans la direction indiquée.
- collect -> "OK"
  charge jusqu'à 10 kilos d'or depuis la case où on se touve. La pile est vidée d'autant
- drop -> "OK"
  décharge le *prospecteur* sur la case courante. Si on est sur la base, les règles de calcul expliquées ci-dessus s'appliquent
- scout X Y -> S
  renvoit le contenu de la case (X;Y) en suivant les normes de la commande "map"
- skip -> “OK”
  passer un tour

Complications
Le contrôleur des impôts est particulièrement vindicatif envers les chercheurs d’or. La moitié de l’or ramené dans la base est prélevée aussitôt déposée.

Mais ce n’est pas tout. Tous les 5 tours, le contrôleur prélève un point au score de chaque joueur. Si un joueur est trop endetté (si son score est inférieur à -50), il est éliminé de la partie. Les joueurs peu scrupuleux peuvent donc essayer de les empêcher de payer leurs dettes...

 Cahier des charges

Par groupes de 4, vous devrez concevoir un programme capable de jouer au jeu décrit ci-avant, et l'écrire en Java.

Les points clefs de ce développement sont:
- Le modèle : votre modèle devra être clair et adapté aux problèmes posés. De lui découle la façon dont votre IA se comporte.
- L'Intelligence Artificielle : sans être nécessairement très compliquée, votre IA doit être capable de prendre une décision correspondant à votre stratégie (donc **explicable**) en fonction des données qu'elle possède. Elle doit également contrôler sa mémoire : quelles sont les informations indispensables à la prise de décision? Quelles sont celles qu'on doit mettre à jour?
- L'interface graphique : le développement de l'interface graphique viendra en dernier. On exigera une version de votre programme qui marche sans aucune visualisation, et une version qui fonctionne avec une représentation visuelle de votre choix.

Une librairie vous sera fournie pour communiquer avec le serveur et tester vos programmes. Le serveur de tests en ligne pour affiner vos IA ne sera disponible qu'après la première phase de réflexion.

 Calendrier

- *Semaines du 4 février et du 11 février :* Présentation du projet et mise en place des équipes
- *Semaine du 10 mars :* Pré-version, étude préliminaire et planning de développement à rendre
- *Semaine du 31 mars :* Soutenance de projet (Note)
- *Date à fixer :* Compétition entre les projets