Comment développer l'IA du mineur ?

Remarques préalables :

- Delirium 2 s'exécute sous Windows XP ou Vista. Vous ne pourrez donc pas a priori développer l'IA du mineur sur une version Linux ou sur Mac, à moins de disposer d'un émulateur Windows XP sur ces systèmes d'exploitation.
- Pour développer l'IA du mineur, vous devez impérativement avoir installé SWI-Prolog version **5.6.49** (disponible dans le répertoire *IA41\Software*).
- Une fois SWI-Prolog 5.6.49 installé, allez dans le *Panneau de configuration*, cliquez sur *Système*, allez dans l'onglet *Avancé* et cliquez sur le bouton *Variables d'environnement*. Ajoutez dans la variable *Path* le chemin où les binaires de SWI-Prolog ont été installés. Par exemple si vous avez installé SWI-Prolog 5.6.49 dans le répertoire C:\Program *Files\pl* alors les binaires seront dans le répertoire C:\Program *Files\pl\bin*.

Prédicats implémentant l'IA du mineur :

L'IA du mineur devra être développée via le fichier 'ai.pl'. Deux prédicats devront être définis :

- le prédicat setViewPerimeter(+Dx, +Dy), où *Dx* et *Dy* indique le nombre de cases que le mineur perçoit autour de lui.
  Si setViewPerimeter a été défini avec une valeur *Dx* ou *Dy* à 0, alors le mineur connaît la totalité des cases du sous-terrain.
- et l'un des prédicats move/8 ou move/12. Le prédicat move/8 permet de définir le comportement du mineur sans que celui-ci puisse modifier son périmètre de vue dynamiquement. Le prédicat move/12 permet au mineur de modifier son périmètre de vue avant d'effectuer un déplacement.
  - move/8 est défini tel que move(+L,+X,+Y,+Pos,+Size,+CanGotoExit,-Dx,-Dy), où *L*, *X*, *Y*, *Pos*, *Size* et *CanGotoExit* sont des paramètres instanciés par le jeu, *Dx* et *Dy* sont les résultats du prédicat. *L* est la liste plate des cases perçues par le mineur autour de lui en utilisant le périmètre de vue défini dans le prédicat setViewPerimeter/2.
    X et *Y* sont les coordonnées absolues du mineur dans le sous-terrain, *Pos* indique la position du mineur à l'intérieur de la liste. *Size* représente la largeur d'une ligne dans la liste *L*. Ainsi, *Pos-1*, *Pos+1*, *Pos-Size* et *Pos+Size* représentent les cases respectivement à gauche, à droite, en haut et en bas du mineur dans la liste *L*.
    CanGotoExit indique si le mineur a ramassé tous les diamants nécessaires pour accéder au sous-terrain suivant. Si oui, *CanGotoExit*=1, sinon *CanGotoExit*=0.
    Dx et *Dy* indique le déplacement que le mineur doit effectuer à partir de sa position actuelle, selon la légende suivante :
    - Si *Dx* = -1 et *Dy* = 0 : déplacement à gauche
    - Si *Dx* = 1 et *Dy* = 0 : déplacement à droite
    - Si *Dx* = 0 et *Dy* = -1 : déplacement en haut
    - Si *Dx* = 0 et *Dy* = 1 : déplacement en bas
    - Toute autre combinaison de *Dx* et *Dy* ne générera aucun déplacement de la part du mineur.
  - move/12 est défini tel que move(+L,+X,+Y,+Pos,+Size,+CanGotoExit,-Dx,-Dy, +VPx, +VPy, -NewVPx, -NewVPy). Les 8 premiers paramètres sont les mêmes que dans move/8. *VPx* et *VPy* représentent le périmètre de vue actuel. *NewVPx* et *NewVPy* représente le nouveau périmètre de vue. Dans le cas où *NewVPx* >= 0 et *NewVPy* >= 0, le mineur ne se déplacera pas. Sinon, si *NewVPx* < 0 ou *NewVPy* < 0, le mineur se déplacera dans la direction indiquée par (*Dx*, *Dy*). Un exemple d'utilisation de move/12 peut être consulté dans 'ai00.pl'.

Création de sous-terrains et codification des cases

Il est recommandé de définir vos propres sous-terrains. En effet il vous sera ainsi plus facile de vérifier si l'algorithme que vous avez développé pour permettre au mineur de se déplacer est robuste dans n'importe quelle situation.
Pour définir un sous-terrain, vous devrez définir un fichier XML. Le fichier Media/Undergrounds/smallmap.xml est un modèle de sous-terrain. Voici dans le tableau suivant la signification de quelques balises utilisées pour définir un sous-terrain :

Nom de la balise	**Présent dans le tag**	**Description**
Name	Map	Nom du sous-terrain
Width	Map	Longueur du sous-terrain
Height	Map	Largeur du sous-terrain
Gather	Goal	Nombre de diamants à récolter

La description des cases constituant le sous-terrain est réalisée entre les balises <Data> et </Data>.
Une fois que votre fichier XML de sous-terrain est défini, vous pouvez l'ajouter à la liste des sous-terrain du jeu en modifiant le fichier Media/Undergrounds/MapsList.xml.

Le tableau ci-dessous indique la codification des cases en Prolog et dans un des fichiers XML décrivant un sous-terrain (voir dans Media/Undergrounds/) :

Code Prolog	**Identifiant XML**	**Type de case**
0	<espace>	Case vide
1	.	Case non visitée (herbe)
2	Diam	Diamant
3	Stone	Pierre
4	StW1	Mur en pierre (peut être destruit après une seule explosion)
5	StW2	Mur en pierre (peut être destruit après deux explosions)
6	StW3	Mur en pierre (peut être destruit après trois explosions)
7	StW4	Mur en pierre (peut être destruit après quatre explosions)
8	SlW1	Mur indestructible
9	SlW2	Mur indestructible
10	Avatar1	Mineur
11	M1	Monstre rouge
12	M2	Monstre bleu
16	Amoeba	Amoeba (se transforme en diamants si entourée d'obstacles)
17	Exit	Sortie

En utilisant les codes Prolog, voici un exemple de ce que peut percevoir le mineur si l'on a défini son périmètre de vue par setViewPerimeter(2, 1) :

9	0	1	1	2
9	3	**10**	2	3
9	4	4	4	4

Les paramètres *L*, *X, Y et Pos* qui seront alors passés au prédicat move/8 seront instanciés comme suit :

- L = [9, 0, 1, 1, 2, 9, 3, 10, 2, 3, 9, 4, 4, 4, 4]
- X = 2
- Y = 1
- Pos = 8

Par défaut, le fichier 'ai.pl' définit un périmètre de vue de 3x3 (setViewPerimeter(3,3)) et un déplacement aléatoire du mineur. Vous pouvez redéfinir ce périmètre en fonction des hypothèses émises lors de l'élaboration de votre méthode de décision pour le mineur. Sachez seulement qu'une fois fixé, le périmètre de vue ne peut être modifié en cours de jeu si vous utilisez move/8.

Débuggage

Pour débugger votre programme Prolog en affichant des messages, utilisez le prédicat *ecrire/1* défini ci-dessous :

ecrire(T) :- open('trace.txt', append, L), write(L, T), nl(L), close(L).

Le terme passé en paramètre de ce prédicat sera ajouté dans le fichier 'trace.txt'.

Dernière étape avant de tester l'IA de votre mineur dans le jeu

Une fois que vous avez (re)défini l'IA du mineur, et donc modifié le fichier 'ai.pl', exécutez le batch 'build.bat'.
Celui-ci a pour effet :

- de créer une DLL 'AvatarPrologAIp.dll' (à partir de la DLL 'AvatarPrologAI.dll') contenant un moteur Prolog et une version compilée du programme 'ai.pl'.
- d'exécuter le jeu 'Delirium2.exe' qui chargera sous forme de plug-in la DLL 'AvatarPrologAIp.dll' qui contient désormais votre IA du mineur.

Une fois le jeu exécuté, appuyez sur 'P' pour passer du mode normal au « mode Prolog » et inversement. Si tout se passe bien, le mineur se déplace sans votre participation lorsque le jeu est en « mode Prolog ».

Pour me contacter...

N'hésitez pas à m'envoyer un mail (fabrice.lauri@utbm.fr) ou à venir directement à mon bureau (D210) si vous êtes bloqués sur un problème technique d'installation du jeu ou si vous avez la moindre questions concernant ce projet.
Préférez autant que possible la deuxième solution (visite à mon bureau), car les explications pourront éventuellement être étayées de démonstrations sur l'ordinateur...