IPIPIP : Chocoban

Sophie.Demassey@emn.fr

Chocoban est un jeu de plateau dans lequel un robot est chargé de ranger des objets dispersés dans une pièce, en effectuant le moins de déplacements possibles.

1 Le jeu

À l'intérieur d'une pièce rectangulaire, représentée par une grille de 10×12 cases, se trouvent : un robot, cinq lourdes pierres et cinq paniers fixes. Ces 11 objets occupent chacun une case différente.

Le robot doit ranger les 5 pierres dans les 5 paniers. Un panier ne peut contenir plus d'une pierre à la fois et il ne peut pas être déplacé. Le robot peut déplacer une pierre, en la poussant par devant lui, horizontalement ou verticalement, et d'une case à la fois. On appelle cela un *déplacement*. Le robot ne peut déplacer qu'une seule pierre à la fois et une pierre ne peut être déplacée si elle est bloquée par un mur ou par une autre pierre. Les mouvements du robot se font également horizontalement et verticalement, et il ne peut passer par une case occupée (par un mur, une pierre ou un panier).

On compte alors le nombre de déplacements de pierres (et non le nombre de mouvements du robot) jusqu'à ce que toutes les pierres soient rangées.

2 Le programme

Au début de son exécution, le programme crée une grille, positionne les objets aléatoirement sur la grille, puis les affiche. Il donne alors la main à l'utilisateur pour commander son robot sur une case adjacente : Sud, Ouest, Nord ou Est. Le programme vérifie alors l'ordre entré par l'utilisateur :

- si la case est libre ou occupée par un panier vide, le robot est placé sur cette nouvelle case ;
- sinon, si la case est occupée par une pierre et si la case par devant est libre ou occupée par un panier vide, alors la pierre est déplacée sur cette nouvelle case ; le robot avance d'une case également ;
- sinon, rien.

Le programme réaffiche l'état de la grille et le score (le nombre de déplacements effectués) puis redonne la main à l'utilisateur. Le programme s'arrête lorsque toutes les pierres sont rangées dans les paniers. Parfois cependant, une pierre ne peut plus être déplacée, quand elle se trouve dans un angle par exemple. Le jeu atteint alors un état de blocage et ne peut donc plus terminer. On équipera donc le programme d'un test pour détecter les cas de blocage et forcer l'arrêt du programme et l'échec de l'utilisateur.

3 L'implémentation

Le code du programme devra satisfaire les exigences suivantes :

- la taille de la grille et le nombre des objets doivent être modifiables : comme toutes valeurs numériques, elles ne peuvent apparaître plus d'une fois dans le code ;
- les fonctions doivent être les plus élémentaires possibles ;
- le code doit comprendre a minima 3 classes pour représenter : un objet principal Chocoban (contient, initialise et met à jour les autres objets), un objet Grille (génère, enregiste, affiche la grille et renseigne sur l'état d'une case donnée), un objet Robot (reçoit les ordres, fait valider et réagit).

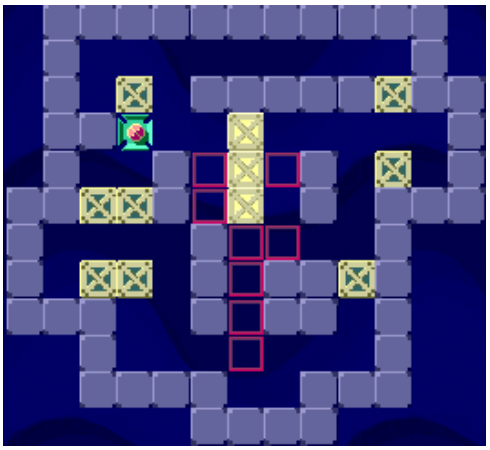
L'affichage de la grille se fera en console. Les schémas ci-après présentent deux exemples de grille. Le robot est noté `X`, une pierre `O`, un panier vide `_`, un panier plein `Q`, un mur `#` et une paroi mobile `*`. Le joueur commande le robot en entrant l'un des caractères `S`, `O`, `N`, ou `E`.

Exemple d'un état de blocage (les deux pierres placées en `Ra` et `Sa` ne peuvent être déplacées) :

```
  A B C D E F G H I J K L M N O P Q R S T
  # # # # # # # # # # # # # # # # # # # #
a #                               O O   #
b #         Q                     #     #
c #                               #     #
d #  _                  X O # # # # #     #
e #       # # # #       O   #             #
f #       #                 #   _         #
g #       #           _     #             #
h #       #                               #
i #       #                               #
j #       #                 _             #
  # # # # # # # # # # # # # # # # # # # # #
Joueur? (sud S, ouest O, nord N, est E, quitter Q)
```

Un autre exemple, difficile à résoudre :

```
    A B C D E F G H I J K
    # # # # # # # # # # #
a   #                   #
b   #   O   # # # # # O # #
c   # # X     *           #
d   #     # _ * _ #   O   #
e # # O O # _ *   #   # # #
f #         # _ _     #
g #   O O   # _ # # O #
h # # #     # _ # #   #
i     #       _       #
j     # # # #     # # #
            # # # #
```



4 Les bonus

Ajoutez des fonctionnalités à votre programme, à volonté ! Quelques idées (de facile à très difficile) :

- rajouter des obstacles et murs dans la pièce, rajouter des parois mobiles (que le robot doit pousser) ou des obstacles dynamiques qui se lèvent aléatoirement durant la partie... voir `http://www.games4brains.de/sokoban-leveldesign.htm` pour des idées.
- mémoriser les scores, charger des grilles pré-générées ou des parties en cours
- organiser les parties en niveaux à franchir de difficulté croissante
- remplacer les pierres par des ballons : dans ce mode, un ballon roule jusqu'à ce qu'il rencontre une case occupée
- donner des roulettes au robot : dans ce mode, l'utilisateur peut spécifier une case non adjacente, le programme doit alors vérifier que cette case est accessible depuis la position initiale
- donner un cerveau au robot : dans ce mode non-interactif, l'IA du programme tente de ranger les objets en minimisant le nombre de déplacements...

5 La difficulté

Modélisation objet : ***
Programmation : ***
Algorithmique : * (avec IA : *****)
Modélisation du problème : *