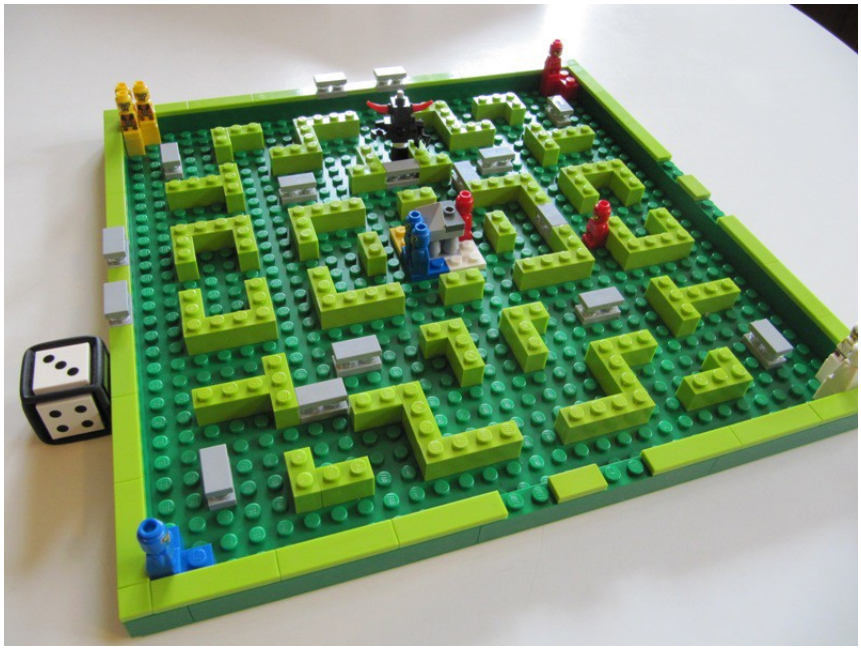
Algorithmes et Structures de Données 2

Projet Minotaurus



On désire pouvoir jouer de deux à quatre joueurs sur un ordinateur au jeu Minotaurus de Lego, dont les règles sont fournies en pièce attachée. En voici un résumé.

Le plateau de jeu correspond à une grille de 32x32. Sur ce plateau, on place :

- l'enceinte extérieure qui ferme le labyrinthe,
- les haies du labyrinthe,
- des murs déplaçables,
- des héros,
- le minotaure.

Le but du jeu est de placer un ou deux héros au centre du labyrinthe, dans le temple secret. A son tour, le joueur :

- lance un dès
- sur un 1, il déplace un mur à l'endroit de son choix
- sur 2, 3, 4, 5, il déplace un de ses héros du nombre de case correspondante.
- sur 6, il déplace de 8 cases le minotaure, pour tenter de capturer et renvoyer à la zone de départ un héros adverse.

Dans votre programme, il faudra :

- lire le plateau de jeu depuis un fichier texte dont l'utilisateur saisira le nom. Cela permettra de faire varier facilement le labyrinthe
- sauvegarder la partie dans un fichier texte, à priori au même format que le fichier précédent.
- faire jouer chaque joueur à son tour, en gérant correctement le déplacement des héros et du minotaure.
- détecter quand la partie se termine.

A chaque tour de jeu, l'ordinateur devra :

1. afficher le plateau de jeu,
2. demander autant de fois que nécessaire au joueur dont c'est le tour de déplacer un héros ou le minotaure.
3. vérifier à chaque fois si le déplacement est valide,
4. mettre à jour le plateau de jeu,
5. vérifier si la partie est finie et si c'est le cas, indiquer qui a gagné.

Pour stocker le plateau de jeu, il faudra au minimum utiliser un tableau à deux dimensions. Définir des types entités peut être utile pour stocker l'état du jeu, la positon des murs, des héros, du minotaure, etc...

Les modalités de réalisation du projet sont les suivantes :

- Les courriers électroniques seront envoyés à l'adresse **projet.asd.l1.blasi@gmail.com**, avec **obligatoirement** dans le sujet [PROJET L1 G**x** nom1 nom2] avec x valant 1, 2 ou 3 en fonction de votre groupe d'ED et nom1 et nom2 étant les noms des deux binômes dans l'ordre alphabétique.
- Le projet est à réaliser en binôme d'un même groupe d'ED. En cas de nombre impair de personnes, un seul monôme est autorisé et la personne seule aura un bonus de notation. La constitution de chaque groupe sera communiquée par un premier courrier vide au plus tard vendredi 14 mars 2014.
- Le projet comprendra une seule partie Python, à rendre le 2 mai 2014. Néanmoins, il est conseiller de passer par une phase algorithmique avant de commencer à coder en Python. Rendre cette phase donnera des points bonus, qui seront bien utiles si au final votre programme ne fonctionne pas.
- Vous enverrez chaque vendredi (au plus tard à 23h59) par courrier électronique l'état d'avancement de votre projet dans une archive ZIP (algorithmes dans un fichier Open Writer, programmes dans un fichier Python 3.2), avec de plus en quelques lignes une explication du travail effectué cette semaine et de celui que vous prévoyez de faire la semaine suivante.
- **Chaque courrier manquant entraînera une pénalité croissante sur la note finale du projet (-1 pour un courrier manquant, - 3 pour le deux, -6 pour trois, -10 pour quatre, -15 pour cinq, -21 pour six). Le premier courrier aura lieu le vendredi 14 mars 2014 avec une archive vide pour indiquer la constitution du binôme.**
- **Le correcteur portera une attention particulière aux programmes comportant de fortes similitudes. Ceux-ci verront leur note suspendue dans l'attente d'une soutenance orale devant un jury, permettant de valider ou d'invalider la réalité du travail rendu.**

Barème de notation :

- Begin End bien placés : 0,5 point
- Assert pour vérifier le type des paramètres des fonctions : 1 point
- Déclaration propre des variables : 0,5 point
- Noms de variables significatifs:1 point
- Commentaires : 2 points
- Variables globales : -1 point
- Duplication de code : -1 point

Annexes

Exemple de format de fichier pour le labyrinthe :

```
EEEEEEEEEEEEEWWEEWWEEEEEEEEEEEEE
E11                           22E
E1                             2E
E   WW  H   HHH  HHH   H  WW   E
E       H   H      H   H       E
E       H   H      H   H       E
E  HH  HHH  HHH  HHH  HHH  HH  E
E  H                          H  E
E  H                          H  E
E     H  W  HHH  HHH  W  H     E
E     H  W  H      H  W  H     E
E  HHHH     H  HH  H     HHHH  E
E  H        H      H        H  E
W  H     H    aabb    H     H  W
W  H   HHH    a  b    HHH   H  W
E           H d Mb H           E
E           H ddcc H           E
W  H   HHH             HHH   H  W
W  H     H     HH      H     H  W
E  H        H      H        H  E
E  HHHH     H      H     HHHH  E
E     H  W  H      H  W  H     E
E     H  W  HHH  HHH  W  H     E
E  H                          H  E
E  H                          H  E
E  HH  HHH  HHH  HHH  HHH  HH  E
E       H   H      H   H       E
E       H   H      H   H       E
E   WW  H   HHH  HHH   H  WW   E
E4                            3E
E44                          33E
EEEEEEEEEEEEEWWEEWWEEEEEEEEEEEEE
```

avec :

- **E** pour les bords
- **W** pour les murs déplaçables
- **H** pour les haies
- **1, 2, 3, 4** pour les héros des joueurs
- **M** pour le minotaure
- **a, b, c, d** pour les temples secrets des joueurs 1, 2, 3, 4.

Conseils de développement :

- Prévoyez une structure de données pour le plateau du jeu, avec au minimum un tableau à deux dimensions de caractère pour le plateau, mais aussi la position du minotaure et la position des murs et la position des temps. Une entité regroupant tout ceci sera pratique.
- Une entité joueur contenant la position de chaque héros sera utile.
- Actions et fonctions conseillées : lire_partie, sauvegarde_partie, affiche_plateau, tour_joueur, deplace_heros, deplace_minotaure, deplace_mur.