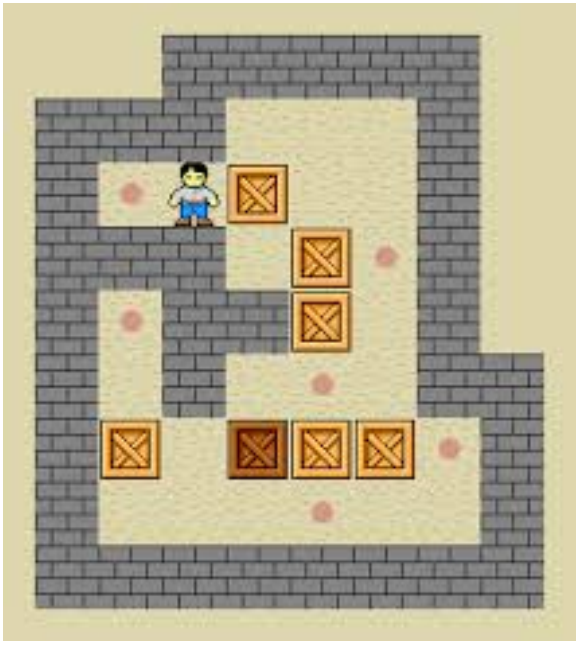
Projet : Sokoban



Le but du projet est de réaliser un jeu de Sokoban : le joueur est un gardien d'entrepôt (divisé en cases carrées) qui doit ranger des caisses (`Box`) sur des cases cibles (`Target`). Il peut se déplacer dans les quatre directions, et pousser (mais pas tirer) une seule caisse à la fois. Le niveau est réussi une fois toutes les caisses rangées.

Le jeu se joue sur une grille. Initialement, chaque case peut être soit vide, soit contenir le gardien ou l'un des objets suivants :

- un `Wall`,
- un `Box`,
- un `Target`,
- une `Key`,
- une `Door`.

Il y a une case spéciale qui sert de point de départ pour le gardien. Le but est, en le déplaçant sur la grille avec les flèches, d'amener le gardien à pousser une caisse sur chaque cible. On ne peut pas traverser les murs. Si l'on passe sur une clé, on la ramasse et si l'on va sur une porte avec une clé, la porte disparaît et l'on perd la clé. Si l'on pousse une caisse sur une clé, la clé est cachée mais elle réapparaît si l'on pousse à nouveau la caisse. Le jeu se termine quand tous les caisses sont sur des cibles.

Vous pouvez tester une version minimaliste du jeu sous linux en récupérant le fichier `sokoban` et les 5 fichiers `map1`, `map2`, ... Pour lancer le programme, tapez la ligne suivante dans le terminal (vous pouvez remplacer `map1` par un autre fichier `map`) :

```
./sokoban map1
```

Ce programme est un simple prototype. Vous pourrez notamment constater qu'il est possible de pousser une caisse en dehors du plateau et que cela provoque une erreur. C'est le genre de chose qui ne doit jamais arriver dans votre projet lorsqu'il sera fini.

1 Le programme à réaliser

Vous allez réaliser votre jeu en 3 phases. Chaque phase devra être terminée et fonctionnelle avant de pouvoir passer à la suivante.

1.1 Phase 1 : la base

Dans un premier temps, vous aller créer la base du jeu. Dans cette phase, un niveau de jeu sera codé "en dur", c'est à dire directement dans le programme. C'est à vous de choisir la structure de données que vous utiliserez pour cela. On doit pouvoir :

1. lancer le jeu avec un niveau qui contient uniquement des murs, des caisses et des cibles (en plus des cases vides, bien sûr) ;

2. déplacer le gardien sur des cases vides ;
3. pousser des caisses sur des cases vides ou des cibles ;
4. vérifier si la partie est gagnée ;
5. ajouter la possibilité de faire “reset”, pour améliorer la jouabilité.
6. ajouter un mode *debug* : on rajoute une touche au choix qui permet d'activer un mode où le gardien se déplace de façon aléatoire (en poussant éventuellement des caisses). Si on appuie sur cette touche à nouveau, on peut recommencer à jouer normalement. On doit pouvoir passer en mode *debug* aussi souvent qu'on le souhaite.

Conseil 1 La liste ci-dessus est une bonne indication de l'ordre dans lequel procéder pour écrire votre programme (sauf la dernière, qui a intérêt à être implémentée dès que possible pour servir de test).

Conseil 2 Vous avez fortement intérêt à séparer votre programme en 3 parties principales :
— celle qui contrôle le jeu (la boucle principale du jeu, la gestion des touches, ...) ;
— celle qui gère l'affichage (et notamment la correspondance entre votre structure de données et les cases affichées) ;
— celle qui maintient toutes les données du jeu (emplacement des éléments sur la grille, point de départ, position du gardien, ...)

1.2 Phase 2 : les maps

Votre programme doit pouvoir **lire un plateau depuis un fichier** dans le format suivant :

```
Titre : "Facile"
W..T..W.....
W.....W...K.
W.....W..WWW
W.....W....W
W....BD....W
W.....W.....
W.....W.....
WWW...W.....
W....WW....S
```

La première ligne donne le titre du niveau entre guillemets. Chaque ligne suivante correspond à une ligne du plateau (toutes les lignes doivent avoir le même nombre de caractères). Les noms des objets sont donnés par des lettres majuscules et les cases vides par de points. Les objets possibles sont : `S` pour `Start`, `W` pour `Wall`, `B` pour `Box`, `T` pour `Target`, `K` pour `Key` et `D` pour `Door` (au passage, `Start` n'est pas un objet mais l'endroit où le gardien commence la partie). Vous devrez rajouter vos propres niveaux pour pouvoir faire plus de tests. Vous pouvez aisément trouver des niveaux sur le Net, mais dans ce cas, vous devez indiquer les sources de ces niveaux dans le jeu.

Vous rajouterez également la **gestion des clés et des portes** avec un affichage latéral des informations du jeu (comme le nombre de clés ou le nombre de déplacement effectués, ...).

Enfin, vous rajouterez de quoi **sauvegarder** la partie en cours et la **recharger**.

1.3 Phase 3 : extensions

Extension 1 Cette extension consiste à rajouter un menu de sélection des niveaux et/ou de la difficulté, pour ne pas avoir à préciser d'argument en ligne de commande.

Extension 2 Rajouter la possibilité de tirer les caisses. Pour cela, vous devez vous placer à côté d'une caisse pour la sélectionner (à l'aide d'une touche particulière). Une fois la caisse sélectionnée, vous pouvez la tirer. N'oubliez pas de faire en sorte de pouvoir la dé-sélectionner.

Autres Extensions

1. Rajouter la `Glace` et le `Pic`. La `Glace` se comporte comme une caisse à la différence que l'on peut pousser un bloc amalgamé de `Glace`. Vous créez un bloc amalgamé de glace en plaçant 2 blocs côte à côte. Vous pouvez amalgamer des blocs de glace simples et des blocs de glace déjà amalgamés. On peut ramasser le `Pic` comme une clé et si l'on se déplace vers un bloc de `Glace` avec le `Pic` alors on détruit tous les blocs de `Glace` qui sont connectés à ce bloc.
2. Construire un éditeur de niveau graphique qui permettra de construire à la souris des fichiers de cartes, en choisissant le differents éléments dans un menu.
3. Construire un programme qui peut dire si, pour un niveau ne contenant que `Wall`, `Door`, `Key` et une seule cible `Target`, il est possible d'atteindre `Target` depuis le point `Start`.
4. ★ Construire un programme qui peut dire si, pour un niveau ne contenant que des murs, une boite `Box` et une seule cible `Target`, il est possible de pousser la boîte sur la cible.
5. Rajouter de quoi lire un fichier de solution (les déplacements exacts du gardien à partir du point de départ) afin de montrer son déroulement au joueur.

Notation et Extensions

Vous devez impérativement réaliser les deux premières phases du sokoban. Vous serez alors notés sur 16. Vous pouvez ensuite réaliser des extensions. Attention, vous devez **d'abord** avoir une version du jeu sans extension qui fonctionne parfaitement (pensez à faire des tests régulièrement et des sauvegardes des versions qui fonctionnent).
Si vous réalisez l'extension 1 ou 2 en plus, vous serez noté sur 18.
Pour être noté sur 20, vous devez réaliser au moins une des autres extensions proposées. Si vous souhaitez proposer de nouvelles extensions, parlez-en à votre chargé de TP avant.

2 Consignes de rendu

Vous devez rendre une version du projet à la fin de chacune des 3 phases

- La date limite pour le premier rendu est le **vendredi 10 novembre 2018 à 23h55**. Passé ce délai, la note attribuée à votre projet sera 0. Les séances de TP de la semaine suivante seront consacrées à la vérification de votre rendu et à amorcer la seconde partie. Remarque : vous ne devez pas commencer la phase 2 avant d'avoir parfaitement rempli l'objectif de la phase 1.
- La date limite pour le second rendu est le **15 décembre 2018 à 23h55**.
- La date limite pour le troisième rendu est le **13 janvier 2019 à 23h55**.
  Des mini-soutenances seront organisées quelques jours après le rendu, pendant lesquelles vous ferez une démonstration sur une machine des salles de TP et serez interrogés sur le projet (les choix techniques/algorithmiques que vous avez faits, les difficultés rencontrées, l'organisation de votre travail, ...). Les contributions de chaque participant seront évaluées, et il est donc possible que tous les participants d'un même groupe n'aient pas la même note.

Contraintes

- Ce projet est à faire en binôme (s'il y a besoin de faire une exception, contactez les enseignants). Vous devrez sélectionner votre binôme sur e-learning dans la semaine suivant la publication du sujet.
- **Vous DEVEZ utiliser** `upemtk`. L'utilisation de modules autres que les modules standards de Python est interdite ; en cas de doute, n'hésitez pas à poser la question.
- **Votre programme devra impérativement s'exécuter sans problème sur les machines de l'IUT.** Prévoyez donc bien de le tester sur ces machines en plus de la vôtre et d'adapter ce qui doit l'être dans ce cas (par exemple, les vitesses de déplacement). Il sera donc utile de permettre à l'utilisateur de préciser certaines options auxquelles vous attribuez des valeurs par défaut plutôt que de devoir modifier le code à chaque exécution.

Le format de chaque rendu est une archive à déposer sur la plate-forme e-learning et contenant :

1. **les sources de votre projet**, commentées de manière pertinente (quand le code s'y prête, pensez à écrire les doctests). De plus :
   - les fonctions doivent avoir une longueur raisonnable ; n'hésitez pas à morceler votre programme en fonctions pour y parvenir ;
   - plus généralement, le code doit être facile à comprendre : utilisez des noms de variables parlants, limitez le nombre de tests, et veillez à simplifier vos conditions ;
   - quand cela se justifie, regroupez les fonctions par thème dans un même module ;
   - chaque fonction et chaque module doit posséder sa *docstring* associée, dans laquelle vous expliquerez le fonctionnement de votre fonction et ce que sont les paramètres ainsi que les hypothèses faites à leur sujet.
2. un **fichier texte** `README.txt` expliquant :
   - les extensions qui ont été implémentées,
   - l'organisation du programme,
   - les choix techniques,
   - les éventuels problèmes rencontrés.

   Vous pouvez y ajouter tous les commentaires ou remarques que vous jugez nécessaires à la compréhension de votre code.

Si le code ne s'exécute pas, la note de votre projet sera 0. Vous perdrez des points si les consignes ne sont pas respectées, et il va sans dire que si deux projets trop similaires sont rendus par 2 binômes différents, la note de ces projets sera 0.