L2S3 – Langage C – Projet de Programmation

Pierre Coucheney, Maël Guiraud, Franck Quessette, Yann Strozecki

07 novembre 2016

Table des matières

Le but est de programmer le jeu sokoban pour pouvoir non seulement jouer mais également créer des niveaux.

1 Le jeu

Sokoban (http://fr.wikipedia.org/wiki/Sokoban) est un jeu où un personnage dans un entrepôt fermé, doit ranger des caisses. La difficulté du jeu est que le personnage ne peut que pousser les caisses et ne peut pas les tirer. Le plateau de jeu est découpé en cases carrées et les déplacements se font d'un case vers une case adjacente.

1.1 Les niveaux

Les niveaux sont stockés par lot dans un fichier au format `.xsb` (l'extension du fichier est parfois `.txt`). Chaque niveau est stocké de la façon suivante :

```
    #####
    #   #
    #$  #
  ###  $##
  #  $ $ #
### # ## #   ######
#   # ## #####  ..#
# $  $          ..#
##### ### #@##  ..#
    #     #########
    #######
```

où `#` est un mur, `$` est une caisse, `.` est une position de rangement d'une caisse, `@` est le personnage. Il existe également deux autres symboles pour coder une caisse déjà sur une position de rangement `*` et le personnage sur une zone de rangement `+`.

Dans ces fichiers `.xsb` avant et après chaque niveau il y a une ligne vide ou une ligne de commentaire commençant par un `;`.

Des exemples de tels fichiers sont disponibles sur internet, par exemple http://www.sourcecode.se/sokoban/levels. De plus le fichier sasquatch1.xsb est fourni sur e-campus.

2 Fonctionnalités à réaliser

2.1 Lire un fichier .xsb

Le nom de fichier et éventuellement le niveau sera lu sur la ligne commande (voir section 2.4).

2.2 Interface de jeu d'un niveau

Une fois un fichier de niveaux chargé :

1. l'interface de jeu commence soit au premier niveau soit à celui spécifié sur la ligne de commande (voir section 2.4).
2. l'interface doit permettre de faire jouer l'utilisateur avec la souris et les flèches du clavier.

3. l'interface doit afficher, le nom du fichier, le numéro du niveau et le nombre de coups joués.
4. l'interface doit afficher, six boutons clicables : `Undo`, `Redo`, `Init`, `Précédent`, `Suivant` et `Quit`.
   — `Undo` et `Redo` permettent de reculer et d'avancer dans l'historique des déplacements effectués par l'utilisateur. Dès que l'on joue un nouveau coup, la position courante devient la dernière de l'historique.
   — `Init` revient au début du niveau sans effacer l'historique.
   — `Précédent` et `Suivant` permettent de changer de niveau. Le changement de niveau entraîne la perte de l'historique du niveau courant.
   — `Quit` sort du programme.
   De plus l'appui sur les touches `U`, `R`, `I`, `P`, `S` et `Q` doivent faire les mêmes actions que les clics.

2.3 Interface de création de niveau

L'accès à l'interface de création d'un niveau se fait grace à la ligne de commande (voir section 2.4).

Afin de créer des niveaux résolubles, l'interface de création va permettre d'effectuer les déplacements ≪ à l'envers ≫.

Dans un premier temps l'interface doit permettre de créer un niveau en positionnant à la souris : personnage, murs et caisses sur les positions de rangement. Il faudra vérifier si l'entrepôt est bien fermé et qu'il y a un et un seul personnage.

Ensuite l'interface doit permettre de jouer à l'envers soit en utilisant le clavier et/ou la souris ou en faisant jouer à l'ordinateur `N` coups au hasard à l'envers. `N` est une constante du programme.

Une fois la position de départ obtenu, un bouton `Save` permet de sauver le niveau dans le fichier spécifié sur la ligne de commande.

2.4 Ligne de commande

Le programme `sokoban` doit accepter les options suivantes sur la ligne de commande :

`sokoban ficin.xsb` lance l'interface pour jouer à partir du premier niveau contenu dans le fichier `ficin.xsb`.

`sokoban -n num ficin.xsb` lance l'interface pour jouer à partir du niveau `num` contenu dans le fichier `ficin.xsb`.

`sokoban -c ficout.xsb` lance l'interface dans le mode de création de niveau, le niveau sera sauvé dans le fichier `ficout.xsb`.

3 Ce qui doit être rendu

Si votre nom est James BOND et votre numéro d'étudiant est 215007007, tous le fichiers à rendre doivent être dans un répertoire nommé `BOND_James_215007007` qui sera compressé en un fichier qui se nommera `BOND_James_215007007.zip` ou `BOND_James_215007007.tgz`. Ce fichier `.zip` ou `.tgz` devra être déposé sur e-campus (voir la section 5).

Dans ce répertoire, il devra y avoir :

— Un fichier `Makefile` ayant une cible `clean` pour effacer tous les fichiers issus de la compilation, une cible `compil` permettant de tout compiler et une cible `test` qui devra effacer tous les fichiers, tout compiler et lancer la commande :

`./sokoban -n 8 sasquatch1.xsb`
L'éxecutable devra donc s'appeler `sokoban`.
— Le fichier `sasquatch1.xsb`.
— Tous les fichiers `.c` et `.h` nécessaires à votre code.
— Un fichier de rapport du projet, maximum trois pages, contenant :
  — La description des structures de données utilisées.
  — La description des algorithmes un peu compliqués.
  — Les améliorations possibles.
  — Les critiques sur votre code.
  — Les éventuelles références à des livres ou site web.
  — Toute autre information que vous jugerez pertinente.
  Ce rapport sera *de préférence* rédigé en LaTeXet vous devrez en fournir à la fois le source et une version pdf.

Dans chaque fichier, hormis `sasquatch1.xsb` il doit y avoir sous forme de commentaire ou autre vos nom, prénom et numéro d'étudiant.

4 Notation du projet

Dans la notation du projet il sera très fortement tenu compte :
— du découpage pertinent en fichier ;
— de la longueur des fonctions (moins de 15 lignes sauf exception) ;
— du nomage pertinent des structures de données, des variables et des fonctions ;
— de la présentation du code. En particulier vous pouvez utiliser la commande indent :
`indent -linux fichier.c`
pour aligner votre code proprement. L'idéal étant d'intégrer cette commande au `Makefile` ;
— de la gestion parcimonieuse de la mémoire. En particulier toute la mémoire devra être libérée avant de quitter le programme ;
— de la qualité du Makefile ;
— de la qualité des commentaires (peu nombreux, mais pertinents) ;
— du respect des règles énoncées dans la section 3 sur ce qui doit être rendu et comment il doit l'être.

Dans la notation du projet il sera très très faiblement tenu compte :
— de la qualité graphique de l'affichage.

Vous pouvez ajouter des fonctionnalités supplémentaires, mais celles-ci n'auront pas d'impact sur la notation.

5 Modalités pratiques

— Le fichier `.zip` ou `.tgz` doit être obligatoirement déposé sur e-campus. Aucun envoi par email ou remise sur clé USB ne sera accepté.
— La date limite de dépôt est le **lundi 09 janvier 2017 à 08h00**. Toute seconde de retard sera sanctionnée de $\frac{1}{2^5 3^3 5}$ point en moins, le projet étant noté sur 20.
— Une présentation et une démonstration de votre projet aura lieu dans la semaine du 16 au 20 janvier 2017.
— Le projet est à faire individuellement.
— Le projet est à faire en langage C à l'exclusion de tout autre langage.

— Un logiciel de comparaison de code sera appliqué à l'ensemble des projets rendu. Ce logiciel permet de détecter très facilement les projets similaires.

Ce projet est long et il est impossible de le faire sérieusement en s'y mettant une semaine avant la date de remise. Il faut donc le commencer le plus tôt possible.

De plus les statistiques des années précédentes montrent que ceux qui avaient beaucoup avancé dans le projet avait 4 à 5 points de plus à l'examen que ceux qui ne l'avaient pas commencé ou peu commencé. L'explication est simple : faire le projet sérieusement est une manières très efficace de comprendre et de connaître le cours.