SAÉ S1.01 : implémentation d'un besoin client

Le jeu du Sokoban : consignes pour la version 1

Présentation générale

Vous êtes chargés de programmer le jeu de Sokoban en langage C, en mode non graphique. Dans sa première version votre programme devra, dans cet ordre :

1. demander à l'utilisateur de saisir le nom d'un fichier `.sok`,
2. "charger" une partie en exploitant le fichier dont le nom a été saisi par l'utilisateur,
3. afficher le plateau de jeu, précédé d'un entête d'informations,
4. gérer les déplacements de l'utilisateur jusqu'à la victoire ou l'abandon de la partie. Après chaque déplacement le plateau sera modifié puis réaffiché avec son entête. L'utilisateur aura aussi la possibilité de recommencer la partie depuis le début.
5. à la fin de la partie le programme indiquera si la partie est gagnée ou si elle a été abandonnée. S'il y a eu abandon, le programme proposera à l'utilisateur de sauvegarder la partie dans l'état où elle se trouve au moment de l'abandon (ce qui permettra de la reprendre plus tard). Si l'utilisateur accepte, le programme devra enregistrer la partie dans un fichier `.sok` dont le nom sera saisi au clavier.

Détail des étapes

Étape 1

Vous trouverez sur Moodle plusieurs fichiers `.sok`, qui vous permettront de tester votre programme. Chaque fichier fourni représente une partie (ou niveau) qui est :

- **bien formée** c'est-à-dire que le plateau est entouré de murs, ne contient qu'un seul sokoban, contient autant de cibles que de caisses, etc.
- et **réalisable**, c'est-à-dire qu'il existe une solution pour cette partie.

Les fichiers `.sok` sont composés des caractères suivants :

`#`	pour représenter un mur,
`$`	pour représenter une caisse,
`.`	pour représenter une cible,
`@`	pour représenter Sokoban,
`*`	pour représenter une caisse sur une cible ,
`+`	pour représenter Sokoban sur une cible,
(*ESPACE*)	pour les autres cases, les cases "vides".

Tout plateau est composé d'au maximum 12 lignes et 12 colonnes.

Étape 2

La procédure C qui permet de lire un fichier `.sok` est fournie. Elle consiste à lire les 144 caractères du fichier `.sok` correspondant aux 12x12 cases du plateau de jeu.

Étape 3

Pour l'affichage du **plateau**, vous utiliserez uniquement les caractères suivants :

`#`	pour les murs,
`$`	pour les caisses,
`.`	pour les cibles,
`@`	pour Sokoban,
(ESPACE)	pour les autres cases.

Exemple d'affichage d'un plateau à l'écran :

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

Pour l'affichage, vous ne distinguerez pas les "caisses" des "caisses sur cible". Dans les deux cas, vous afficherez un **`$`**. De même, vous afficherez un **`@`** pour représenter Sokoban dans les deux cas, qu'il soit présent ou non sur une cible.

L'affichage de **l'entête** devra précéder celui du plateau de jeu et présenter de manière succinte :

- le nom de la partie (*ie* nom du fichier)
- la liste des actions possibles et des touches correspondantes : **q**, **z**, **s**, **d** pour les déplacements de Sokoban respectivement vers la gauche, vers le haut, vers le bas et vers la droite ; **x** pour abandonner et **r** pour recommencer la partie.
- le nombre de déplacements effectués ; votre programme devra compter tous les déplacements de Sokoban, qu'il s'agisse de déplacements seuls ou de déplacements en poussant une caisse.

Votre programme effacera l'écran avant tout nouvel affichage.

Étape 4

Cette étape consiste à lire au clavier puis traiter un caractère (voir plus loin les détails techniques), et recommencer tant que la partie n'est ni gagnée ni abandonnée :

- s'il s'agit de **q**, de **z**, de **s** ou de **d** il faudra, si c'est possible, déplacer sokoban ou bien déplacer sokoban et une caisse (poussée) ;
- s'il s'agit d'un **x**, cela mettra fin à la partie (abandon) ;
- enfin s'il s'agit d'un **r**, il faudra recommencer la partie depuis le début en rechargeant le fichier `.sok` (l'utilisateur devra d'abord confirmer qu'il souhaite bien recommencer).

Tout autre caractère que ceux évoqués ci-dessus seront ignorés.

Étape 5

La procédure C qui permet d'enregistrer un fichier `.sok` est fournie. Elle consiste à écrire dans le fichier indiqué les 12x12 cases du plateau.

Détails techniques

Représentation du tableau de jeu
Le plateau de jeu sera représenté en mémoire par **un tableau 2D** de taille 12 x 12 (c'est ce tableau qui sera initialisé lors de la lecture d'un fichier `.sok`). Vous créerez un type `t_plateau` pour ce tableau.

Effacer l'écran
Pour effacer l'écran, écrivez l'instruction : `system("clear") ;`

Lecture d'une touche au clavier
Pour une meilleure fluidité du jeu, la lecture d'une touche au clavier se fera sans écho (*ie.* sans que le caractère frappé apparaisse à l'écran) et sans validation par la touche Entrée. Vous programmerez donc la saisie d'un caractère comme ceci :

```
touche = '\0';
if (kbhit()){
     touche = getchar();
}
```

Le code de la fonction `kbhit()` vous est fourni. Ajoutez-le dans votre programme en incluant aussi ces trois bibliothèques :

- `<termios.h>`
- `<unistd.h>`
- `<fcntl.h>`

En aucun cas vous ne devez modifier le code de cette fonction `kbhit()`.

Manipulation de fichiers
Les procédures de lecture et d'écriture de fichier sont fournies sur Moodle ; elles sont à ajouter au code source de votre programme :

- `void chargerPartie(t_plateau plateau, char fichier[])` : lit les données du fichier dont le nom est passé en paramètre pour initialiser le tableau passé en paramètre,
- `void enregistrerPartie(t_plateau plateau, char fichier[])` : enregistre les éléments du tableau passé en paramètre dans le fichier dont le nom est passé en paramètre.

En aucun cas vous ne devez modifier ces deux procédures.

Consignes

Votre programme devra obligatoirement comporter les procédures ou fonctions suivantes.

- `afficherEntete(…)` : cette procédure doit effacer l'écran et afficher les informations utiles au joueur.

- `afficherPlateau(…)` : cette procédure doit afficher le plateau de jeu.

- `deplacer(…)` : cette procédure effectue si c'est possible le déplacement de sokoban ou de sokoban et d'une caisse ; cette procédure modifiera donc le contenu du tableau 2D représentant le plateau de jeu.

- `gagne(…)` : cette **fonction booléenne** doit retourner `true` si la partie est gagnée et retourner `false` sinon.

Important

Vous produirez un code source **bien structuré**. Pour cela, vous pourrez créer d'autres fonctions et procédures que vous jugerez nécessaires.

Enfin, une attention particulière sera portée au respect des **conventions de codage**, et notamment :

- pas de variables courtes (une lettre, sauf i, j, x, y),
- que du camelCase pour les noms de variable,
- que du snake_case en majuscule pour les constantes,
- que du snake_case pour les fonctions,
- espace partout sauf après une parenthèse ouvrante et avant une parenthèse fermante,
- maximum de 80 caractère par ligne,
- pas plus de 5 indentations,
- pas plus de 60 lignes par procédure ou fonction.