

Principes généraux du programme

Groupe 3 Bleu : Sokoban

Goron Nathan, De La Rosa Louis-David, Basset Emilien, Demé Quentin

1 Introduction

Pour commencer, nous avons choisi de programmer notre sokoban en orienté objet car cela présente quelques avantages majeurs comme la possibilité de stocker plein d'information assez explicitement dans des listes. On a également choisi l'orienté objet car c'est ce qui nous semblait être le plus intuitif.

2 Conception du jeu

2.1 Les classes

Il y a en tout 5 classes :

- Sprite
- Personnage (qui hérite de Sprite)
- Caisse (qui hérite également de Sprite)
- Niveau
- LevelCollection

Les classes Sprite, Personnage et Caisse vont servir à représenter tout les objets du jeu voici la liste de tout les objets Existant :

- le personnage - il représentera comme son nom l'indique le personnage contrôlé par le joueur.
- les caisses - elles seront poussées par le personnage
- les murs - ils sont là pour bloquer tout ce qui se déplace, c'est à dire le personnage et les caisses
- les cibles - ce sont les emplacements sur lesquels il faut placer les caisses
- le vide et les espaces - ils représentent les endroits où il n'y a rien

Les murs, les espaces, le vide, et les cibles sont des instances de la classe Sprite. les caisses des instances de la classe Caisse et le personnage sera une unique instance de la classe Personnage.

La classe Niveau représentera le niveau qui sera actuellement chargé. Il contient entre autre la méthode qui affiche puis rafraîchit tout le jeu. Il contient également quelques fonctions utiles comme la recherche de la position du personnage ou encore une partie du test de victoire.

La classe LevelCollection sert à charger une collection de niveau présente dans un fichier .slc. On demande un niveau via la méthode load().

2.2 représentation du plan

Pour représenter le sokoban, nous avons choisi de le construire de la sorte : le programme se compose de 2 grilles nommées gameP et gameO. Ces dernières contiennent des objets cités plus haut.

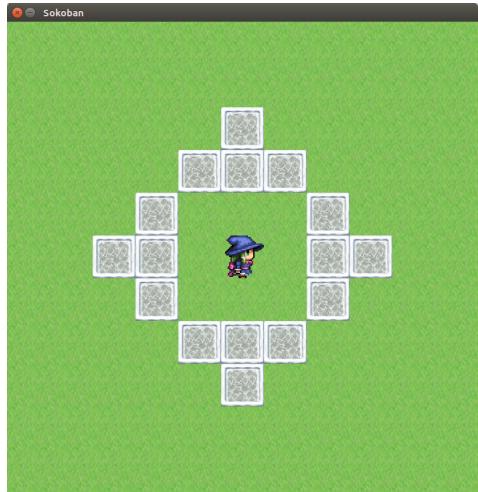
la grille gameP (game Plan) contient les cibles et les espaces. la grille gameO (game Obstacle) contient les murs, les caisses et le personnage.



aperçue graphique du plan.



aperçue graphique de la grille gameO.



aperçue graphique de la grille gameP (ne pas considérer le joueur).

3 Le joueur

3.1 gestion des déplacements

Le joueur est une instance de la Classe Personnage. il a donc une méthode spécialisée nommée `deplace()`. ci dessous l'explication de son fonctionnement :



Le personnage teste sa destination et ne se déplace pas si la case est prise.



Le personnage test sa destination est rencontré une caisse (croix rouge). il demande donc à la caisse de tester si sa destination est libre (croix bleue).



Exemple de déplacement impossible.

3.2 représentation graphique

Le personnage a une représentation graphique un peu particulière. Etant donné qu'il doit se déplacer dans 4 directions, il faut que celui ci ait une image différente pour chaque déplacements. On a aussi fait le choix d'animer le personnage lorsqu'il se déplace, il font donc une frame par pas. Dans notre cas le personnage a 3 frames par pas dont une qui se repète. on a donc 4 fois 3 images pour notre personnage.

Sur internet on a réussi à trouver ce type d'image appelé "tileset" et en bonus un 8 styles différents pour le personnage :



4 Les règles du jeu

4.1 Les déplacements

Avec le système d'objet que l'on a choisi, les déplacements respectent les règles du jeu qui sont :

- On ne peut que pousser les caisses
- On ne peut pas pousser plus d'une caisse à la fois
- Les caisses et le personnage ne peuvent pas traverser les murs
- La partie est gagné lorsque toutes les caisses se trouvent sur les cibles
- Le joueur a la possibilité de revenir en arrière si un déplacement n'aurait pas du être.
- l'objectif est de réussir à mettre toutes les caisses sur les cibles en faisant le moins de déplacements possible.

4.2 Condition de victoire

Le fonctionnement de la condition de victoire est simple : une fonction va voyager dans la grille gameP. Si il y a une cible, alors vérifier dans la grille GameO si une caisse est dessus, si c'est le cas, c'est gagné.



La condition n'est pas vérifiée.