Projet d'Informatique

Séance "Analyse"
Exemple d'analyse

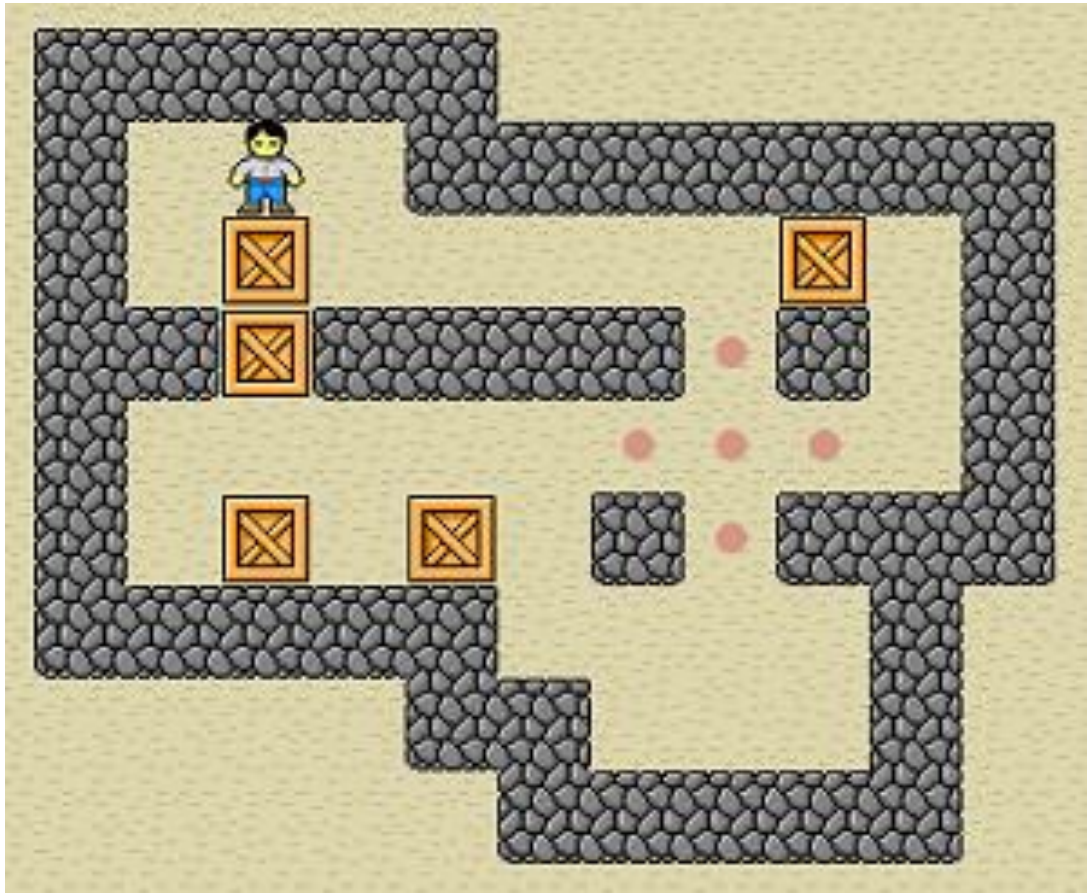
UMONS

Sébastien Bonte

Département d'Informatique

Faculté des Sciences

Enormément de questions à se poser



Répartition en MVP

Vue :

- Maquette graphique
- Quelles ressources / images utiliser ?
- …

Présentation :

- Comment évolue une partie ?
- Quelles sont les règles d'une partie ?
- Quand se termine une partie ?
- …

Modèle :

- Comment représenter en mémoire la carte sur laquelle on joue ?
- Comment va-t-on stocker ces entités ? Quelles classes va-t-il falloir créer pour les stocker ?
- …

Quelques exemples de cas à analyser

- Boucle de jeu
- La carte en mémoire
- (Baba Is You) Quelles classes pour les différentes entités ?

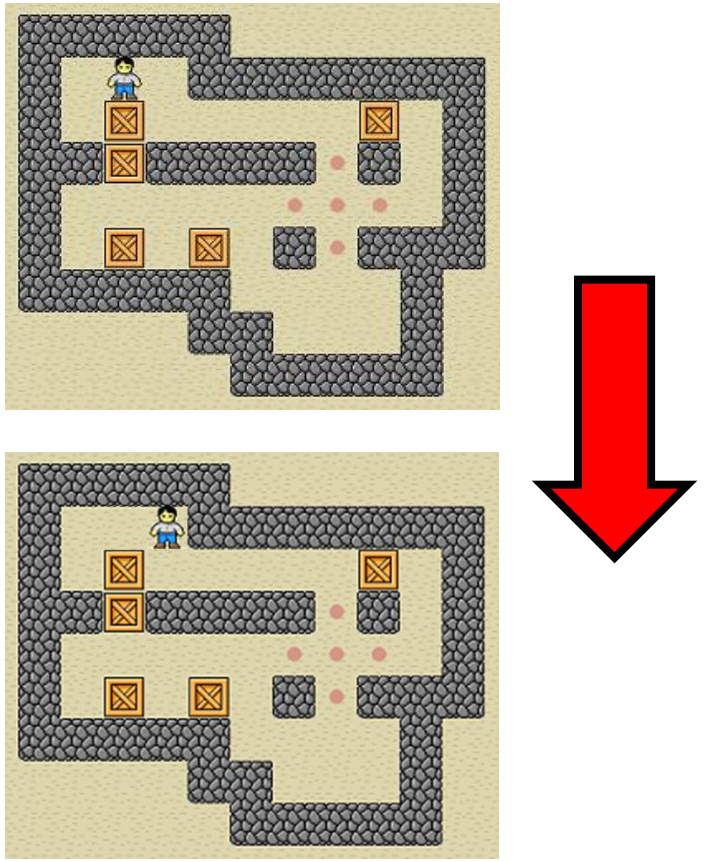
Boucle de jeu : compréhension globale

Quand on commence la partie, on est dans un certain **état** de jeu.

Nous nous trouvons ici dans un cas où le jeu ne progresse que si l'utilisateur effectue une action, c'est-à-dire un déplacement du personnage du joueur.

Quand un déplacement est effectué, il se passe un certain nombre de choses et on peut arriver dans un nouvel état. Il est également possible que la partie s'arrête pour diverses raisons.

Bien sûr, il faut toujours être prêt à gérer d'autres inputs : quitter la partie, recommencer le niveau, ...



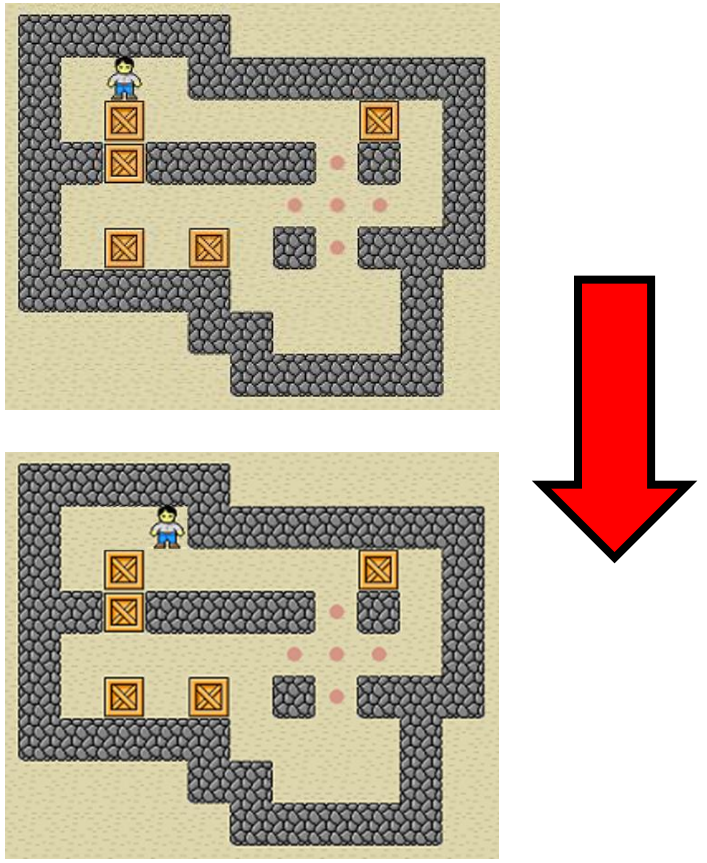
Progression d'un premier état vers un second, le joueur a bougé.

Boucle de jeu : Progression d'un état

Quand un déplacement est effectué, il faut alors gérer un grand nombre de cas possibles :

- Le personnage peut simplement s'être déplacé ou avoir marché contre un mur.
- Le personnage peut avoir marché contre un objet qui peut être poussé. Dans ce cas là il faut gérer le déplacement d'un autre objet, ce qui peut avoir des conséquences en soit.
- La dernière caisse pourrait être poussé sur une case de victoire, dans quel cas il faut gérer la victoire du niveau.
- …

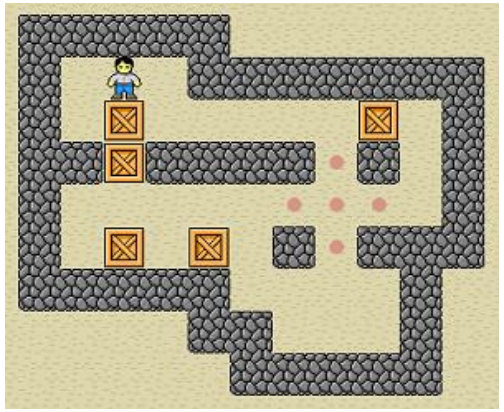
On peut donc s'imaginer *simplement* une boucle dans laquelle on attend à chaque fois un input de l'utilisateur.



Progression d'un premier état vers un second, **you** a bougé.

La carte en mémoire

La carte n’est pas statique, le joueur peut se déplacer et pousser différentes entités. Il faut donc garder la carte en mémoire !



Première solution qui pourrait venir en tête : Garder des listes des différents types d’objets et stocker la position d’un objet sur la carte sous la forme des attributs de cet objet.

Problèmes :

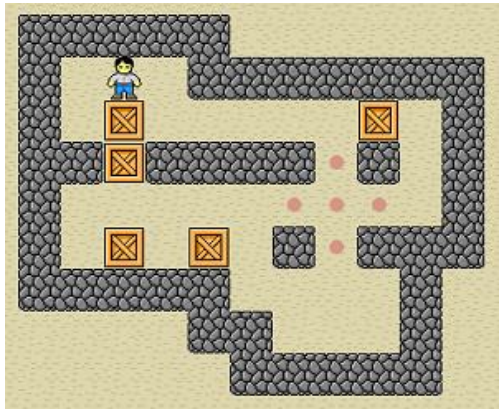
- Comment retrouver quels objets sont sur une case ? Il va alors falloir vérifier la position de chaque objet pour être certain, ce qui prend un temps assez conséquent !
- Quand on déplace une des caisses, il va falloir vérifier également les cases alentours pour vérifier que le chemin n’est pas bloqué!
- …

La carte en mémoire

Autre solution possible : une seule matrice où chaque paire d'indice (i,j) référence une case de la carte.

Problèmes :

- Plusieurs objets peuvent être sur la même case. (=> matrice de listes ? Plusieurs matrices ? …)
- Il faut garder en tête qu'on ne saura pas directement quel objet se trouve dans une case, il faut donc savoir traiter efficacement cette information.
- …



Quelles classes pour les différentes entités ?

On pourrait être tenté de simplement créer une classe pour chaque type d'objet selon ce qu'on voit à l'écran : une classe pour les **rock**s, une classe pour **baba**, …

Est-ce vraiment une bonne idée de faire les choses ainsi ? Les règles pour un type d'entité peuvent changer selon la position des entités de type **texte** sur la carte.

Comment adapter facilement les objets pour que leur fonctionnement puisse changer ?

Par exemple, un mur peut ne pas être traversable et soudainement le devenir.