**Projet Sokoban**
**— Étape 3 —**

# 1 Objectifs

Dans la troisième étape du projet, vous devez programmer le déplacement des caisses ainsi que limiter les déplacements aux cases valides du scénario. Ensuite, nous allons tester si le joueur a gagné.

# 2 Déplacement des caisses

Le déplacement des caisses se fait indirectement avec l'avatar du joueur. C'est uniquement quand l'avatar avance sur une caisse que celle ci est poussée par l'avatar du joueur. À part cela, son mouvement est programmé de manière similaire à celui de l'avatar.

**Mise à jour des caisses :** Dans le module caisse (`caisse.py`), vous devez écrire la fonction `update`. Cette fonction reçoit une liste contenant des dictionnaires d'attributs de caisse comme argument. La fonction doit changer la position de chaque caisse de la liste selon sa vitesse de déplacement. Par exemple, si la vitesse ligne d'une caisse est 1 alors cette caisse doit se déplacer d'une ligne vers le bas.

**Déplacement des caisses dans le scénario :** Pour que les caisses se déplacent dans le jeu, vous devez le mettre à jour dans le scénario. Pour cela, dans la boucle principale de la fonction `execute` du module scénario, vous devez ajouter un appel à la fonction `update` que vous avez créée pour les caisses. Cet appel doit être fait juste après la mise à jour de l'avatar mais avant de dessiner le scénario.

**Déplacement de caisses par l'avatar :** Pour qu'une caisse soit poussée par l'avatar, nous devons faire quelques changements dans le module avatar.

Premièrement, nous allons changer l'initialisation de l'avatar. L'avatar doit avoir une référence à son scénario. Cela permettra plus tard de vérifier s'il est en train de pousser une caisse ou pas. La fonction `init` de l'avatar doit recevoir un argument de plus : le dictionnaire des attributs d'un scénario. Il faut rajouter l'attribut `scen` à l'avatar et l'initialiser avec cette référence. N'oubliez pas non plus d'adapter l'appel à cette fonction `init` dans `scenario.py`.

Deuxièmement, la fonction `update` de l'avatar doit être changée. Maintenant, avant de changer la position de l'avatar, il faut vérifier si, une fois déplacé, l'avatar se trouvera sur une caisse du scénario. Pour tester cela, utilisez l'attribut `scen` de l'avatar, qui contient la liste des caisses du scénario. Si c'est le cas, il faut changer la vitesse de déplacement de la caisse en question de manière à ce qu'elle se déplace dans le même sens que l'avatar.

**Test du déplacement :** Testez votre programme maintenant. L'avatar doit être capable de pousser les caisses. Pourtant, les caisses peuvent être poussées en dehors du scénario et les unes sur les autres. Dans la suite, nous allons corriger cela.

# 3 Limiter les déplacements

**Vérifier la présence d'une caisse ou d'un mur :** Dans un premier temps, spécifiez et écrivez les deux fonctions suivantes dans le module `avatar`

- la fonction `est_sur_mur` qui prend trois arguments, un numéro de ligne `lig`, un numéro de colonne `col`, et le dictionnaire des attributs d'un scénario `scen`. Cette fonction retourne vrai s'il y a un mur en position `(lig,col)` dans `scen`. *Indice : quel est l'attribut de* `scen` *qui permet de savoir où sont les murs ?*
- la fonction `est_sur_caisse` qui prend trois arguments, un numéro de ligne `lig`, un numéro de colonne `col`, et le dictionnaire des attributs d'un scénario `scen`. Cette fonction retourne vrai s'il y a une caisse en position `(lig,col)` dans `scen`. *Indice : quel est l'attribut de* `scen` *qui permet de savoir où sont les caisses ?*

**Limiter le déplacement du joueur :** Dans la fonction `update` de l'avatar, vous devez rajouter quelques vérifications.

D'abord, il faut vérifier si, une fois le déplacement fait, l'avatar se trouve sur un mur. Dans ce cas, le déplacement ne sera pas fait. *Indice : calculez la position d'arrivée de l'avatar, puis testez si cette position est sur un mur. Si ce n'est pas le cas, vous pouvez réellement changer la postion de l'avatar.*

**Test :** Testez votre programme maintenant. L'avatar ne peut plus sortir du scénario de jeu. Pourtant, il est encore possible de pousser les caisses les unes sur les autres et sur les murs.

**Limiter le déplacement des caisses :** Vous avez déjà fait (dans la section précédente) la vérification qui permet de savoir si, une fois le déplacement de l'avatar fait, il se trouvera sur une caisse. Maintenant, il faut aller plus loin. Si la caisse qui est poussée peut se trouver elle-même sur un mur ou sur une autre caisse, alors ni la caisse ni l'avatar ne seront déplacés. *Indice : reprenez le même principe que précédemment pour l'avatar.*

**Test :** Testez votre programme maintenant. Les caisses ne peuvent plus aller les unes sur les autres ni sur les murs.

# 4 Est-ce que j'ai gagné ?

Ajoutez, dans la fonction `execute` du scénario un test à la fin de la boucle principale : Si toutes les caisses sont dans une cible, alors le joueur a gagné ! Dans ce cas, arrêtez la phase du jeu, félicitez le joueur et passez à la prochaine phase. *Indice : passez par une fonction* `caisses_sur_cibles` *qui teste si chaque caisse est sur une cible.*

# 5 Bonus

Vous pouvez ajouter quelques fonctionnalités bonus à votre jeu :

- Une touche pour réinitialiser la phase du jeu.
- Un compteur de mouvements (pour essayer de réussir avec le moins de mouvements possibles).
- Lire les grilles de niveau dans un fichier (plutôt que de les initialiser dans le code).
- Un chronomètre (pour savoir qui termine la phase le plus rapidement).
- Sauvegarder les meilleurs scores dans un fichier et les afficher en début et fin de chaque phase.
- Etc.