

# SOUTENANCE

# SOKOBAN

Présenté par :

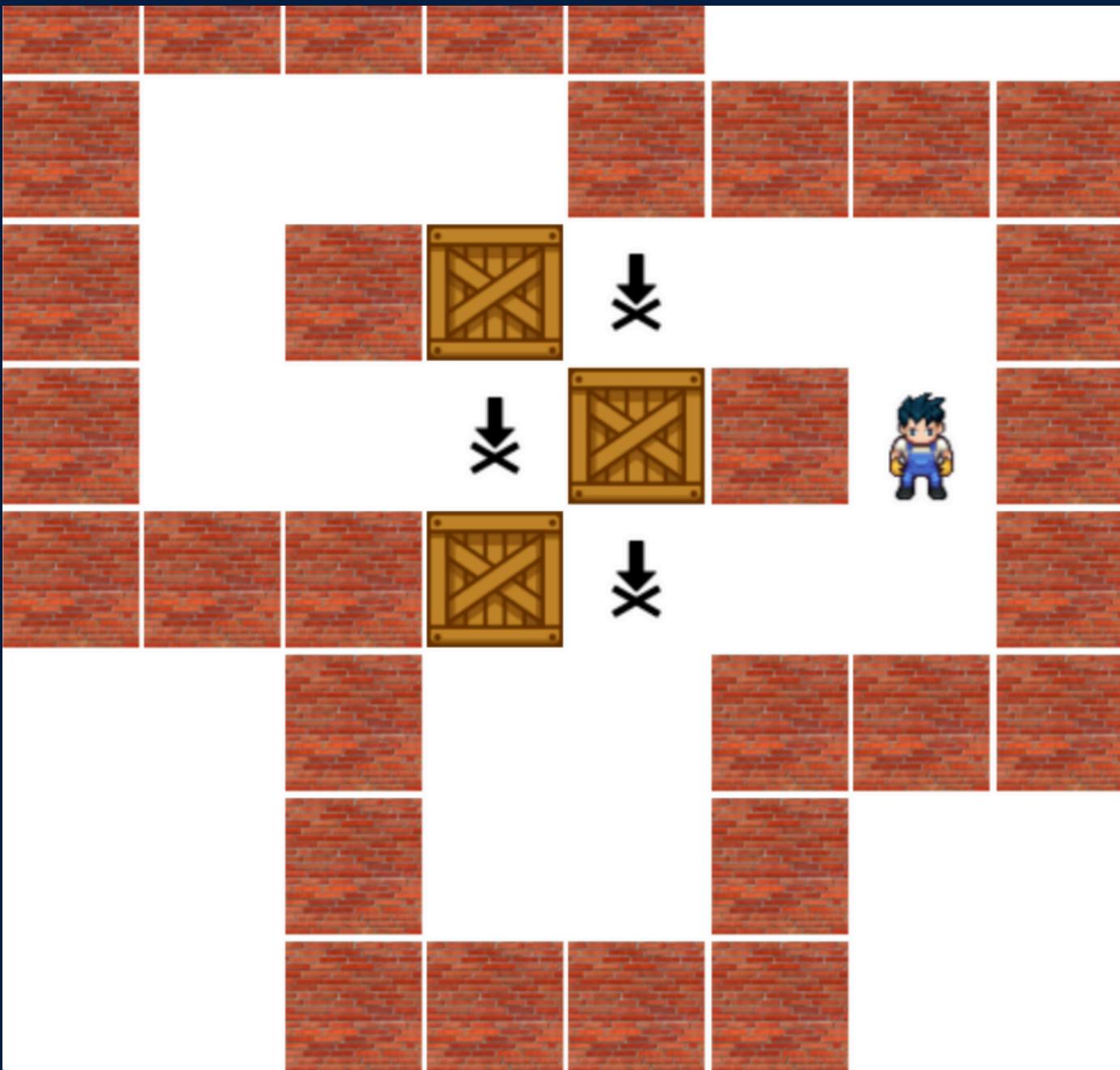
- Gautam DEMEULEMEESTER
- Abdelkader HEDDI
- Khalil BOUCHAMA

LIEN GITHUB :  
 <https://github.com/K-Boo/Myg>

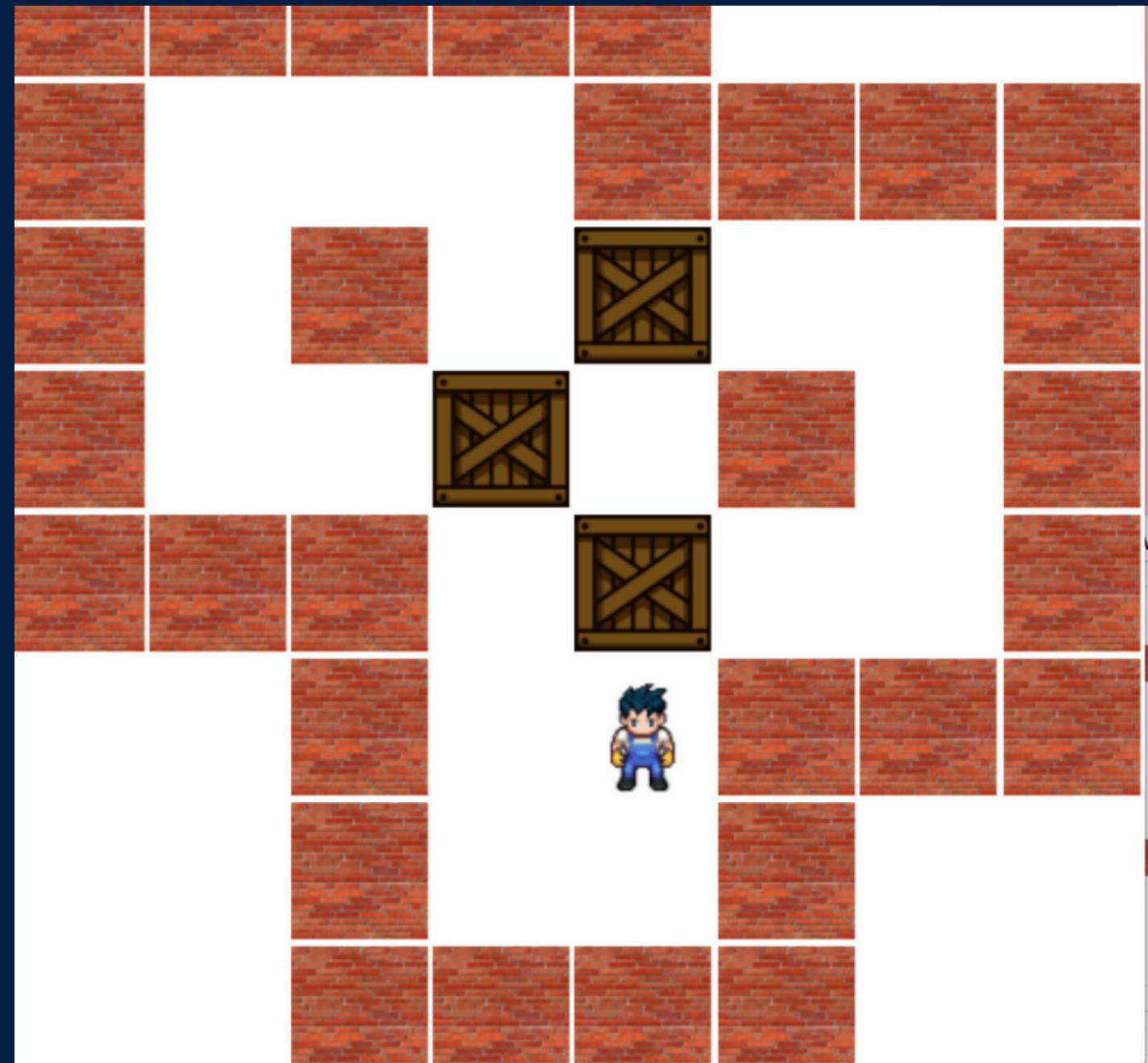


# PRÉSENTATION DU JEU

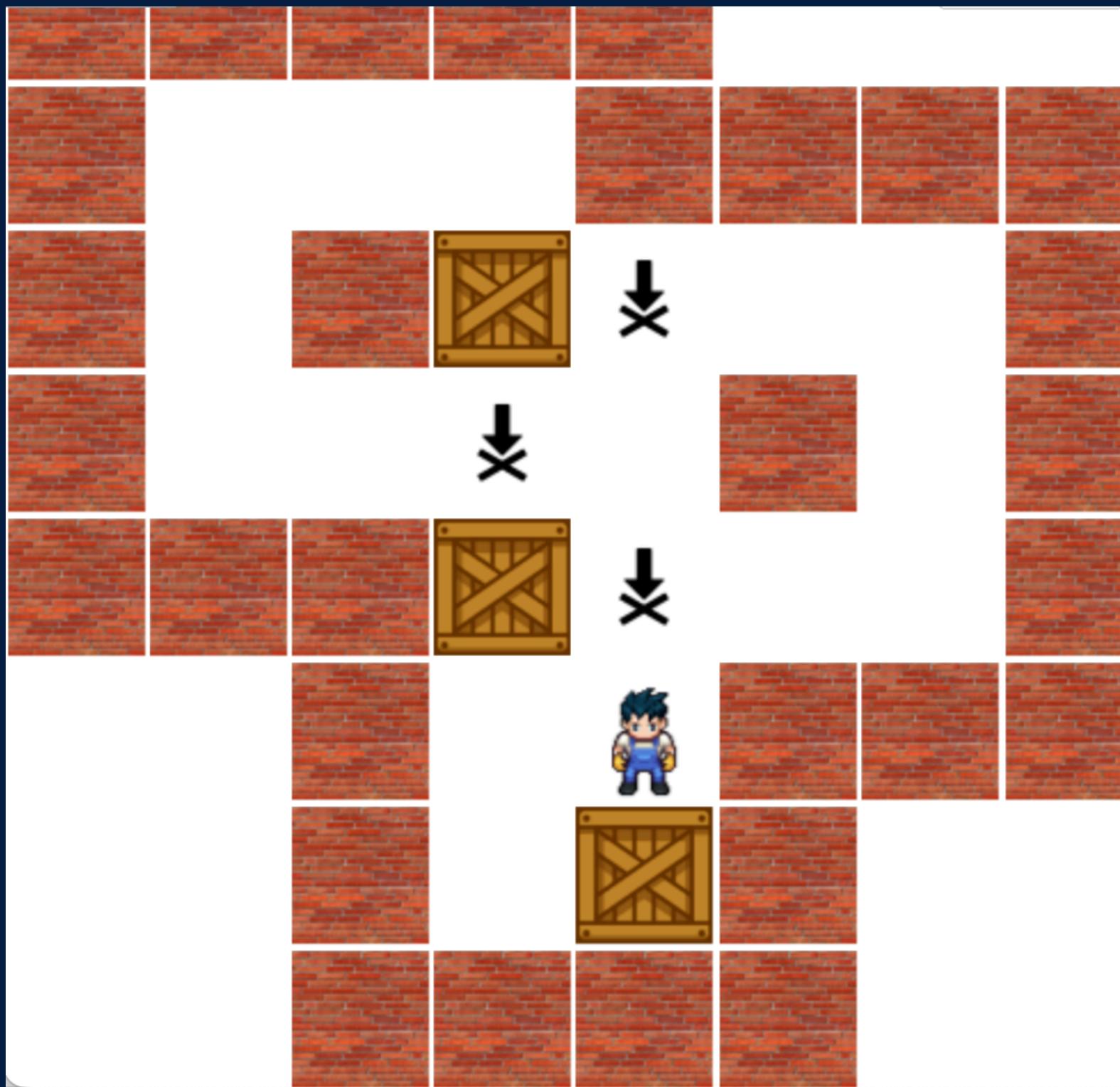
01 DÉBUT DE LA PARTIE



02 PARTIE GAGNÉE



# EXEMPLE DE PARTIE PERDUE ✗



# PRÉSENTATION DU DESIGN ORIGINAL

01

Architecture Modulaire



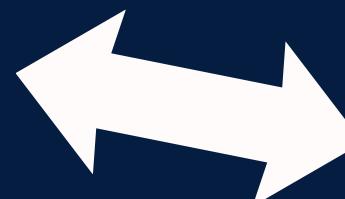
02

Hiérarchie de Classes (POO)



03

Double DISPATCH



# ARCHITECTURE MODULAIRE



## 2 PACKAGES PRINCIPAUX :

01

Modèle (Myg-Sokoban-Model) :

- Contient toute la logique du jeu (Board, Player, Box)
- Totalement indépendant de l'affichage.

#MODEL

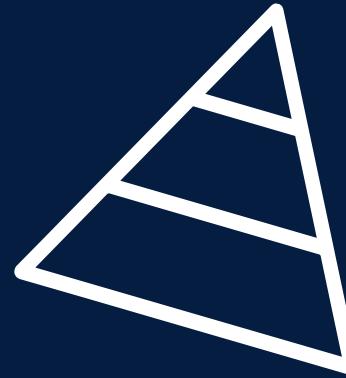
02

Vue (Myg-Sokoban-UI) :

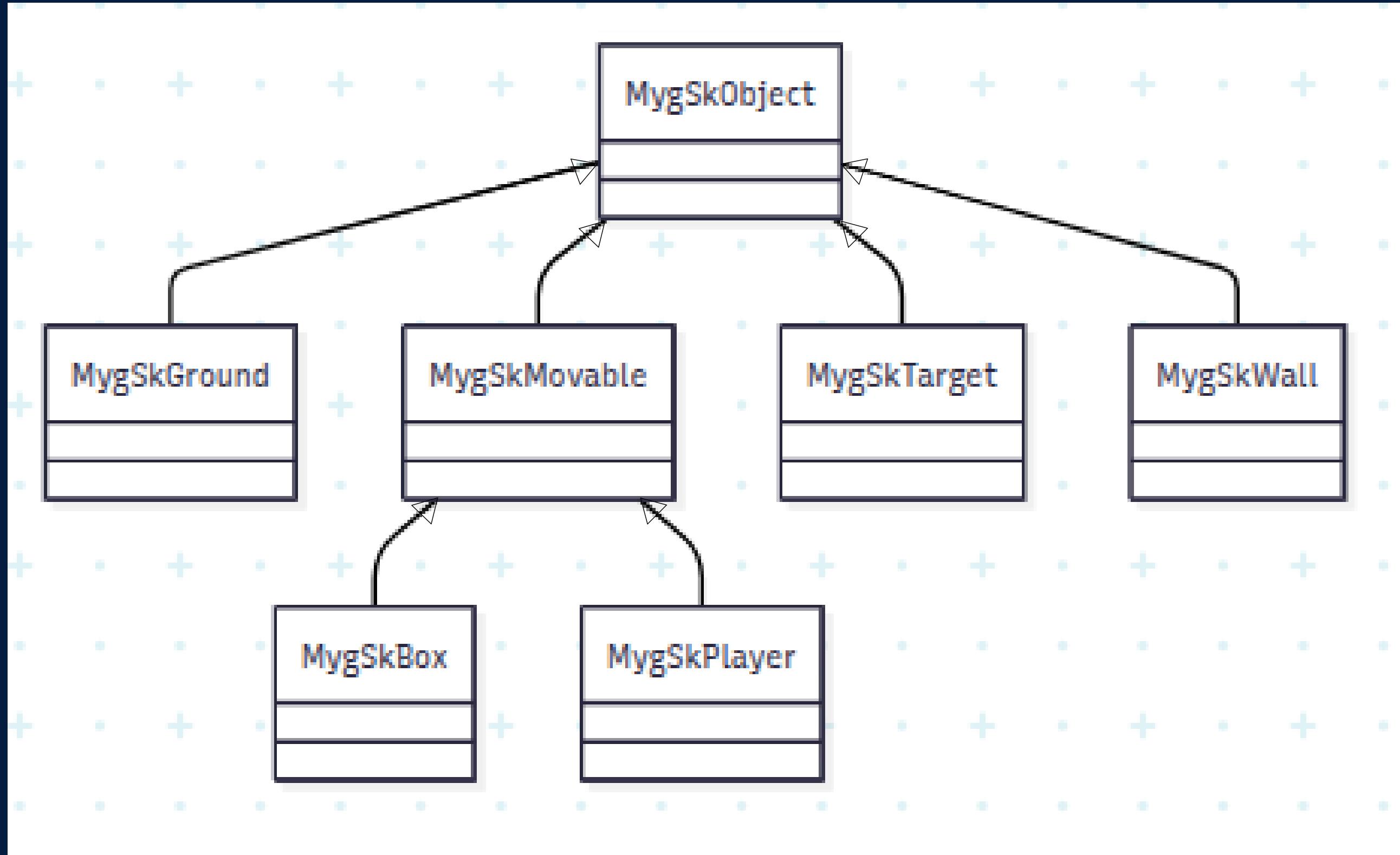
- Utilise la librairie graphique Bloc.
- Gère l'affichage et capture les événements clavier.

#UI

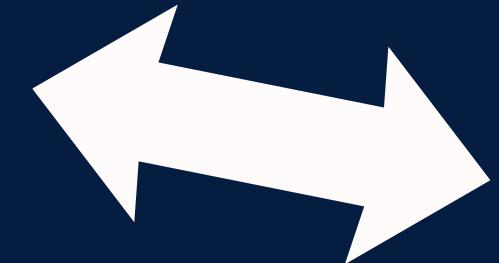
# HIÉRARCHIE DE CLASSES (POO)



- 01 **MygSkObject** : Classe racine (position, lien avec le plateau).
- 02 **MygSkMovable** : Pour les objets qui bougent (Joueur, Caisse).
- 03 **MygSkGround / MygSkWall** : Pour les éléments statiques.



DOUBLE  
DISPATCH



# Déplacement d'objet

## 1- Objet demande à bouger (**move: direction**)

Cela appelle la case voisine (**moveIn:going:**) “Puis-je entrer ?”

## 2- La case réponds en fonction du type (**bringIn:Going:**)

mur refuse / sol accepte / boite vérifie si elle peut être poussée à son tour

# NOUVELLES FONCTIONNALITÉS

01

Fonction **UNDO**



02

Compteur de mouvements

Moves: 11

03

Couleur des boîtes/cibles



# FONCTIONNALITÉ UNDO



## BESOIN

Permettre au joueur de corriger une erreur de déplacement ou de stratégie.



## SOLUTION

Ajouter une action "Annuler" (Undo) pour revenir à l'état précédent.

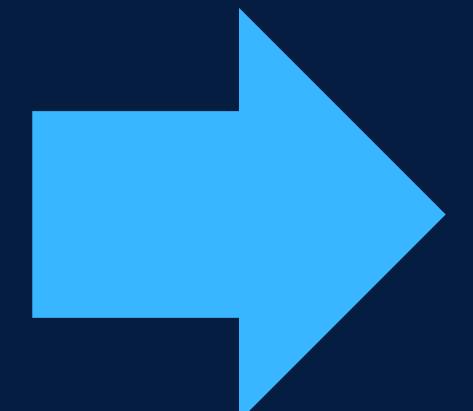
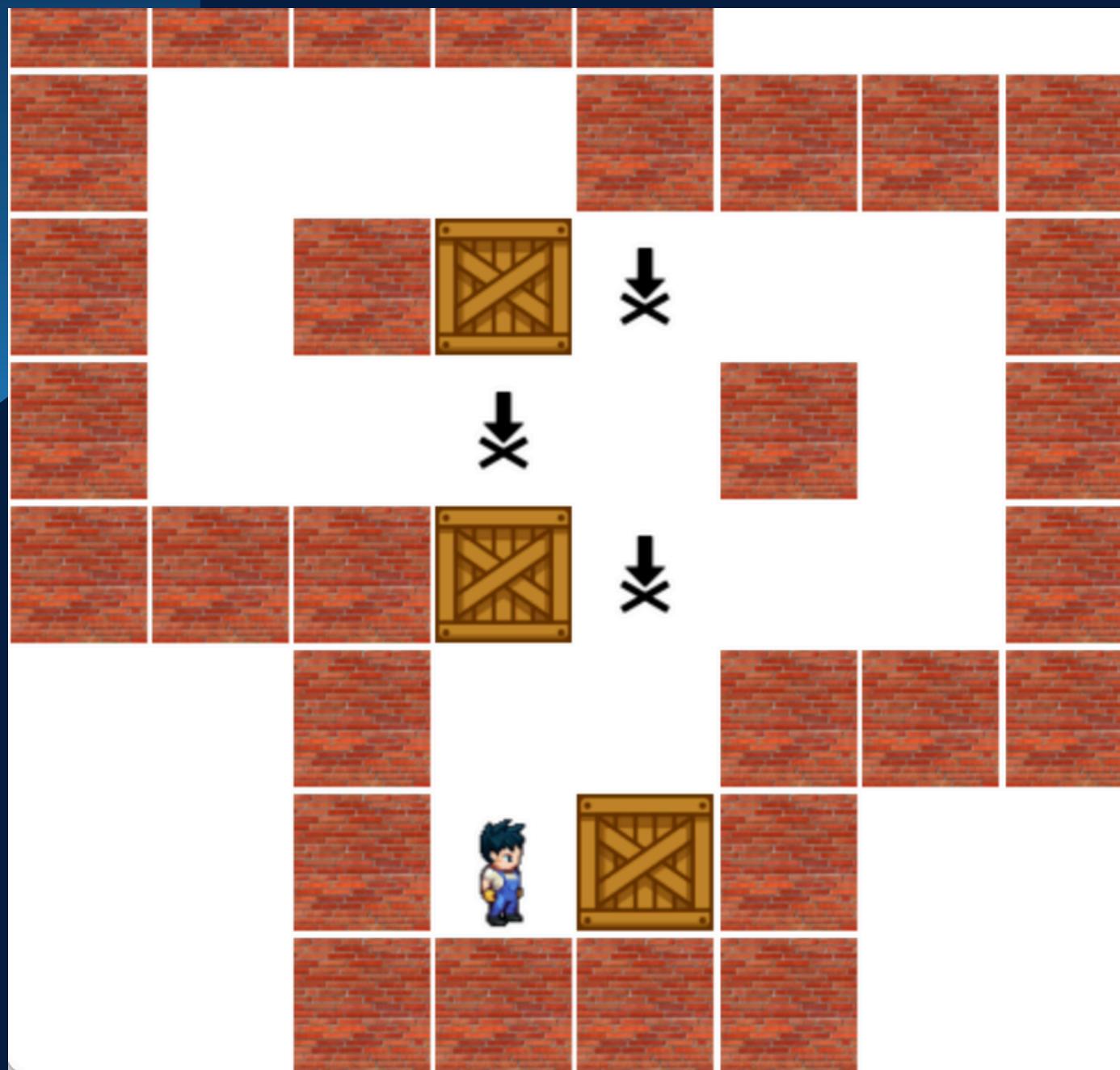


## BUT FINAL

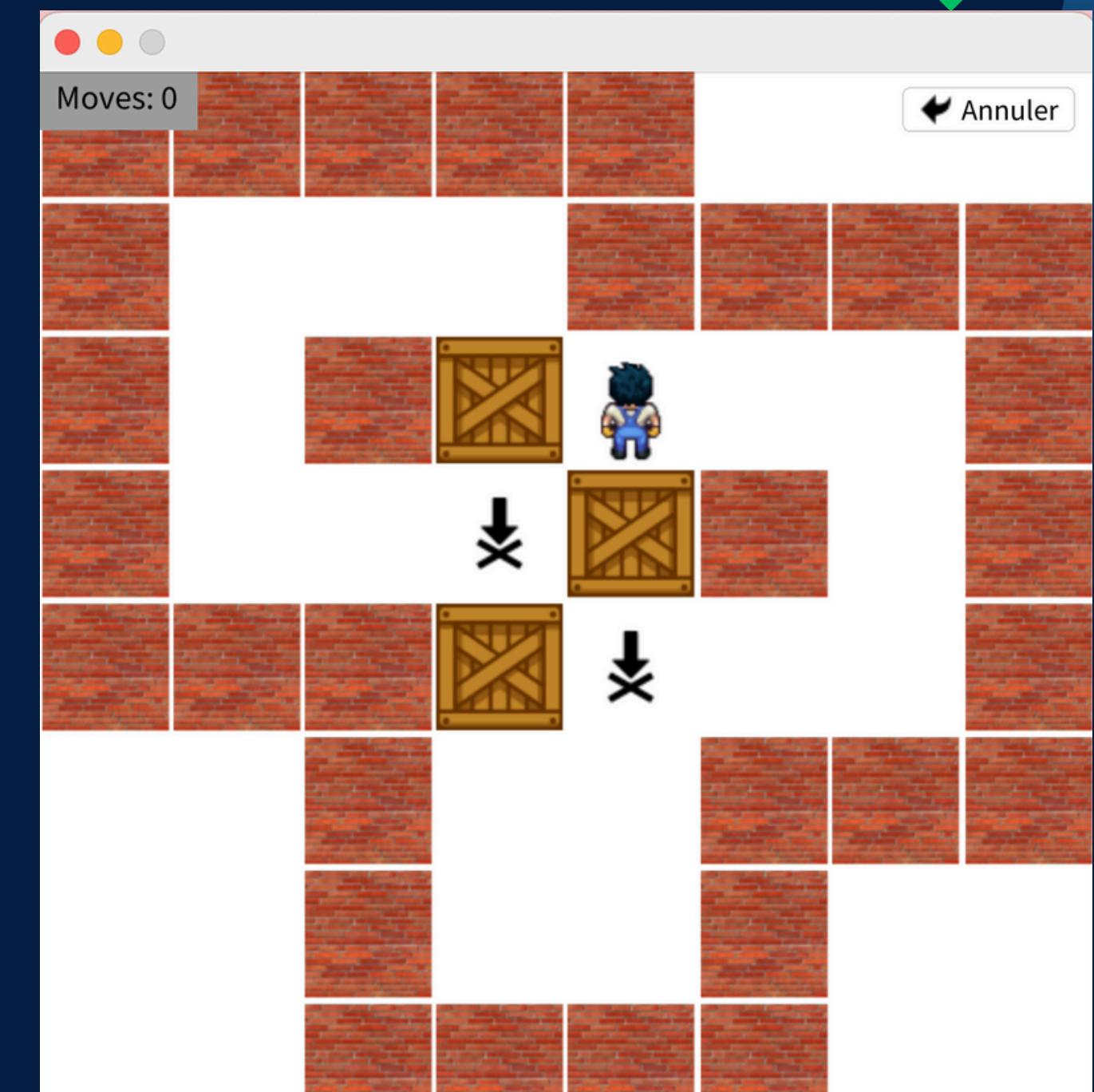
Améliorer l'expérience utilisateur en évitant de devoir recommencer tout le niveau à la moindre erreur.

# FONCTIONNALITÉ UNDO

Avant l'implémentation ✗



Après l'implémentation ✓



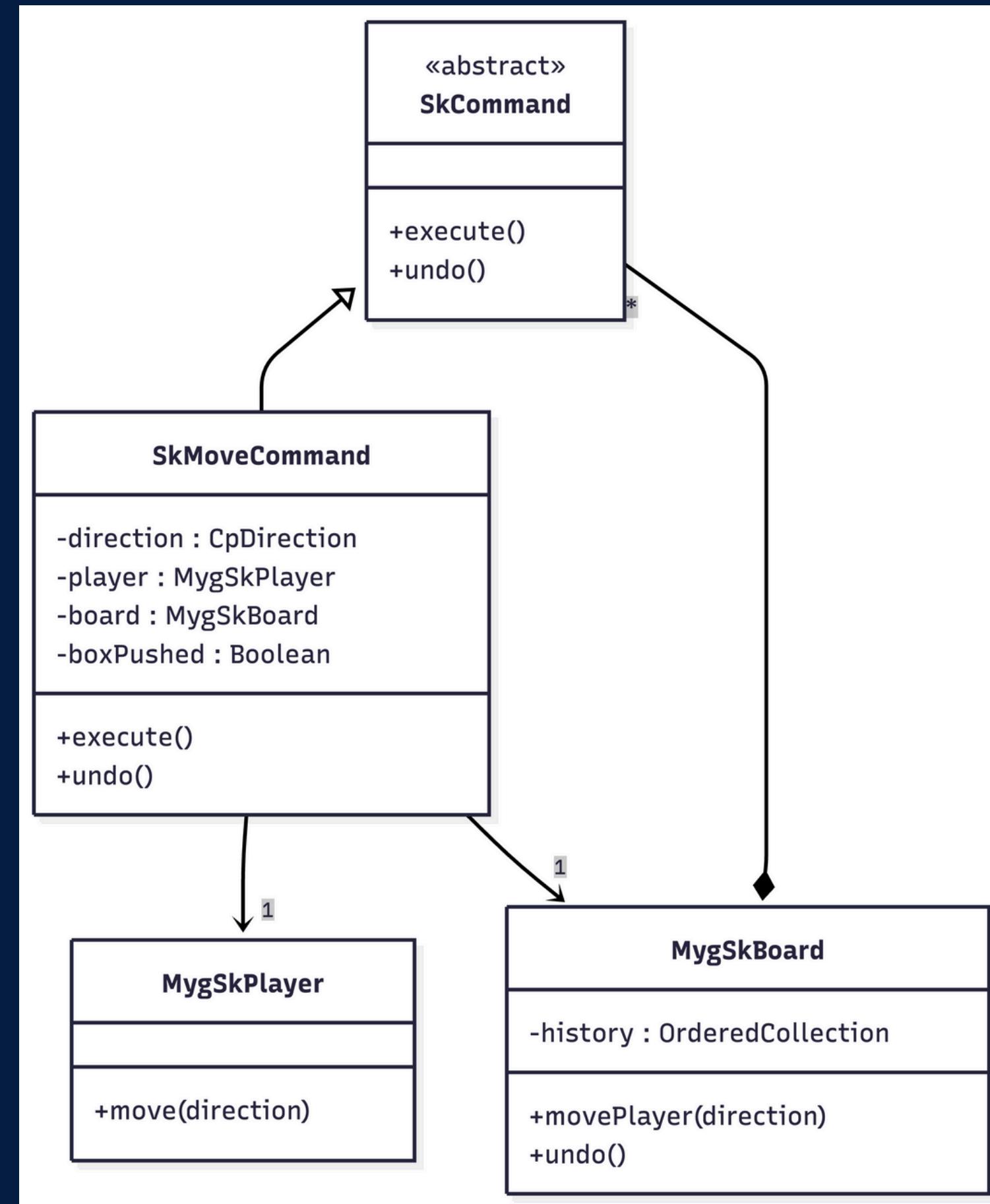
# DESIGN UNDO

→ Approche TDD

- testMultipleUndos
- testResetClearsHistory
- testUndoBoxPush
- testUndoEmptyHistory
- testUndoSimpleMove

→ Utilisation du Pattern COMMAND

# DESIGN UNDO



# FONCTIONNALITÉ COMPTEUR



## BESOIN

Permettre au joueur de voir son nombre de coups en temps réel.



## SOLUTION

Ajouter un label qui se met à jour automatiquement

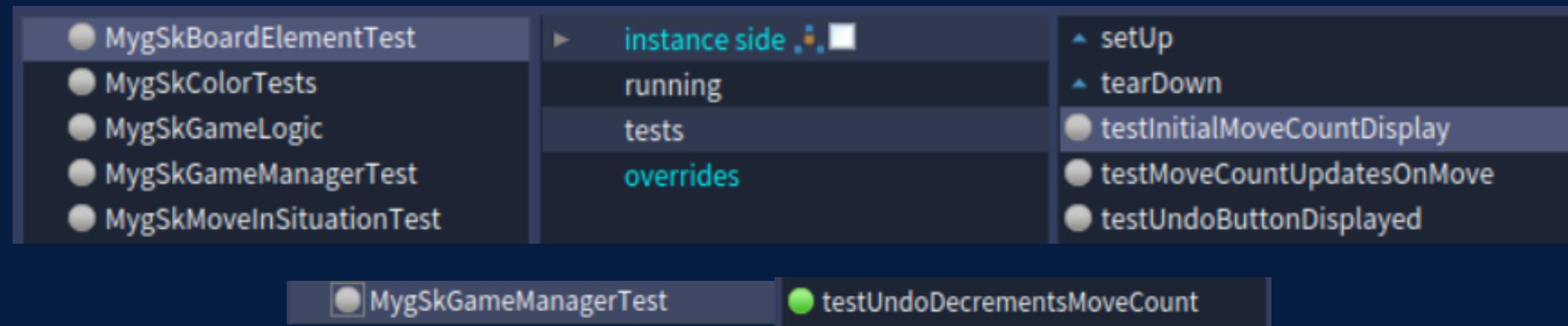


## BUT FINAL

Apporter l'information du nombre de déplacement utilisé pour finir un niveau.

# DESIGN UNDO

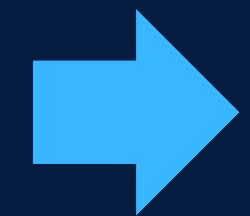
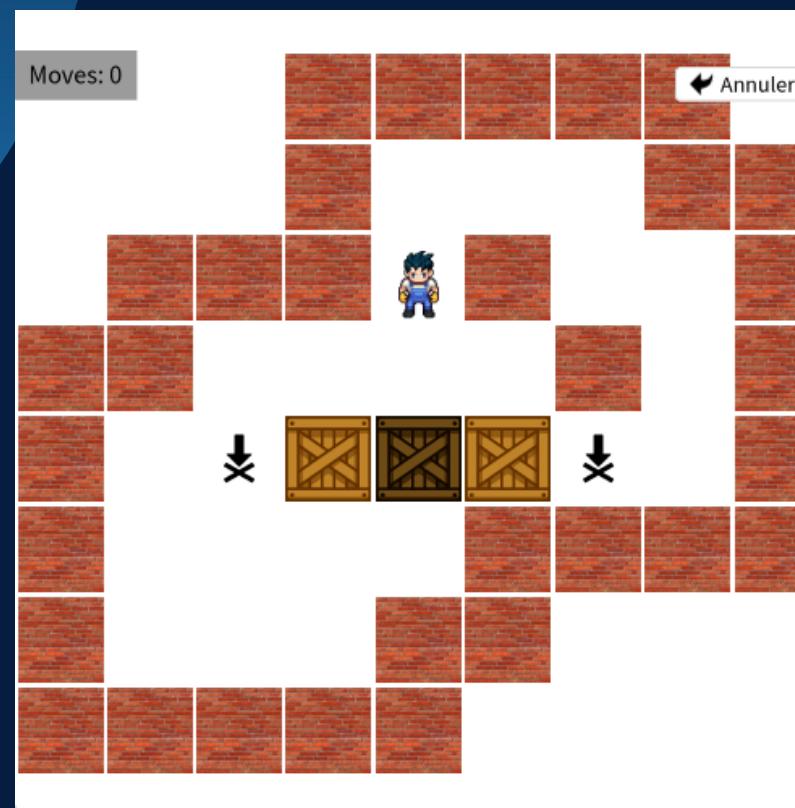
## → Approche TDD



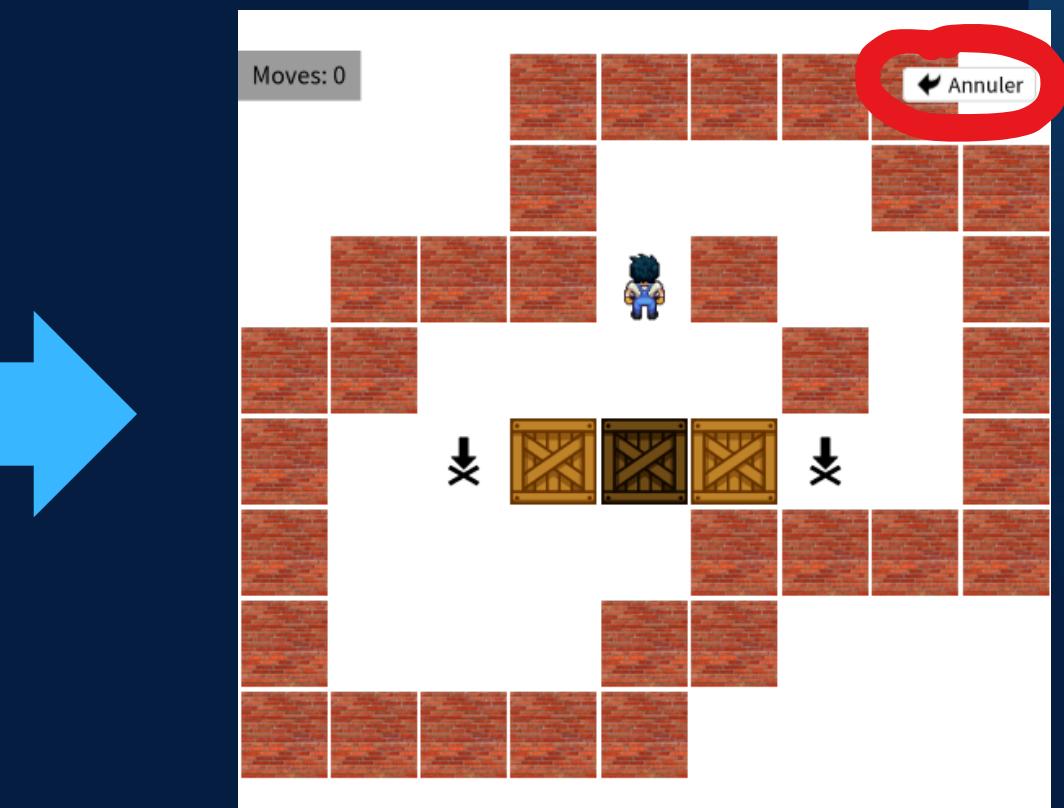
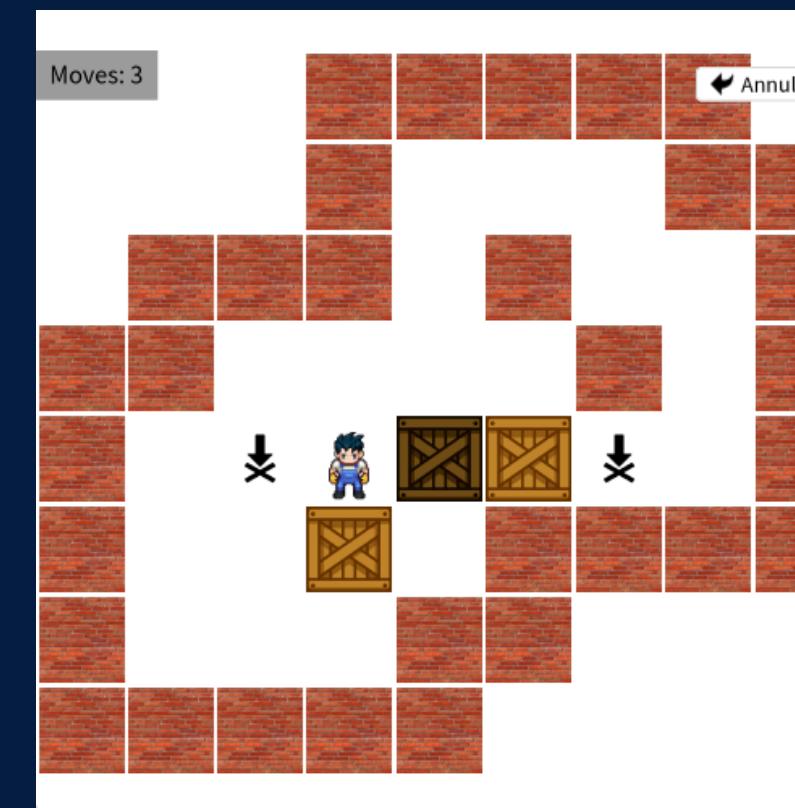
- 1. Crédit d'une classe de test : **MygSkBoardElementTest**
- 2. Initialisation du compteur : **MygSkBoardElement.initializeMouvCountLabel ()**
- 3. Mise à jour du compteur pour chaque mouvement réussi : **updateMoveCountLabel**
- 4. Décrémentation à chaque undo : modification direct dans **MygSkBoard.undo**.

# FONCTIONNALITÉ COMPTEUR

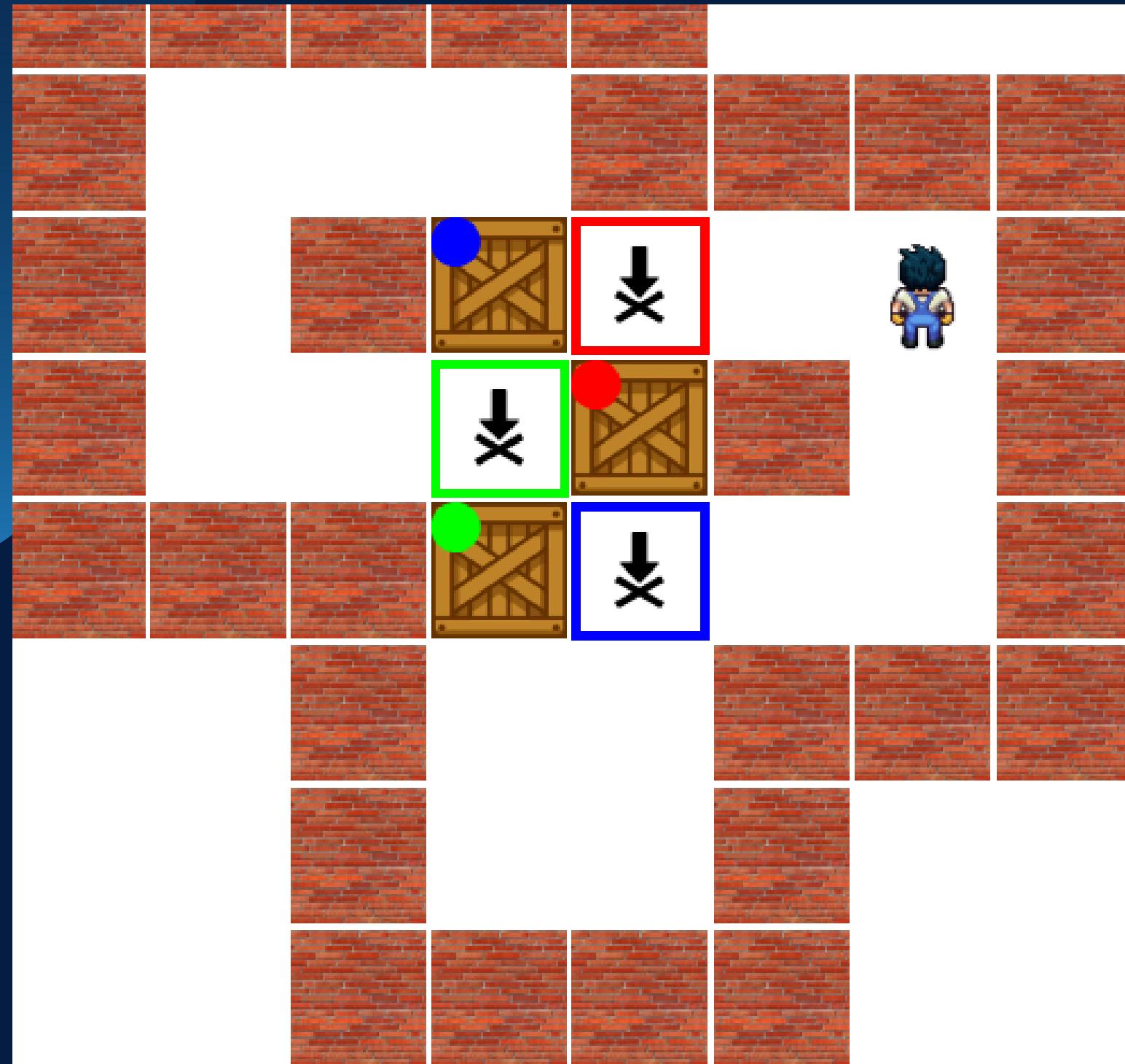
Avant l'implémentation



Après l'implémentation



# Ajout de la notion de couleur



Avant : Cible valide si une boite est présente dessus

Après : Cible valide si une boite de la même couleur est présente dessus

# 1 - Tests (TDD)

Acceptation des boites par les cibles

Nouvelle logique de fin de partie

Assignment aléatoire des couleur (rouge,bleu,vert)

# 1 - Tests (TDD)

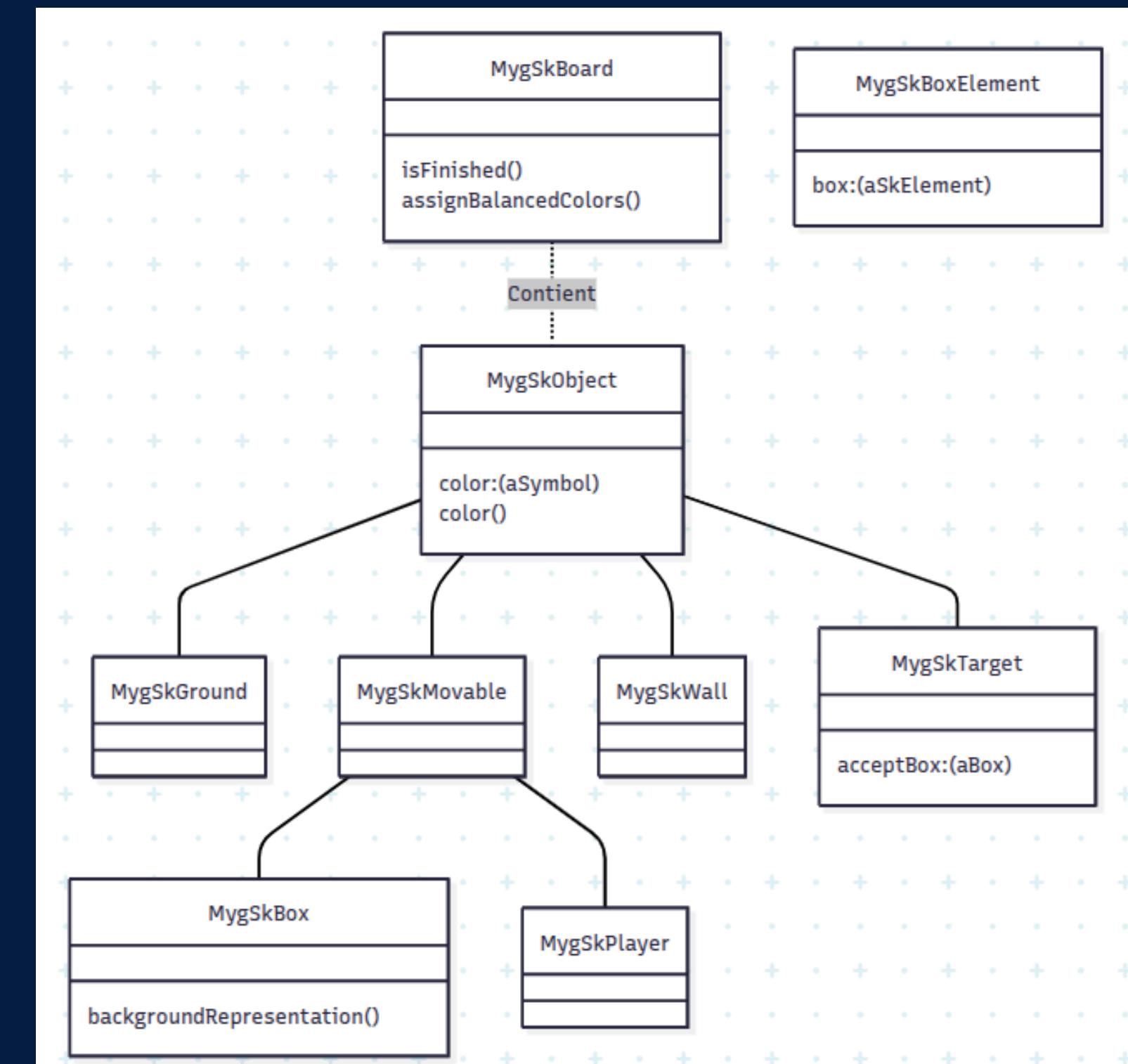
Acceptation des boites par les cibles

Nouvelle logique de fin de partie

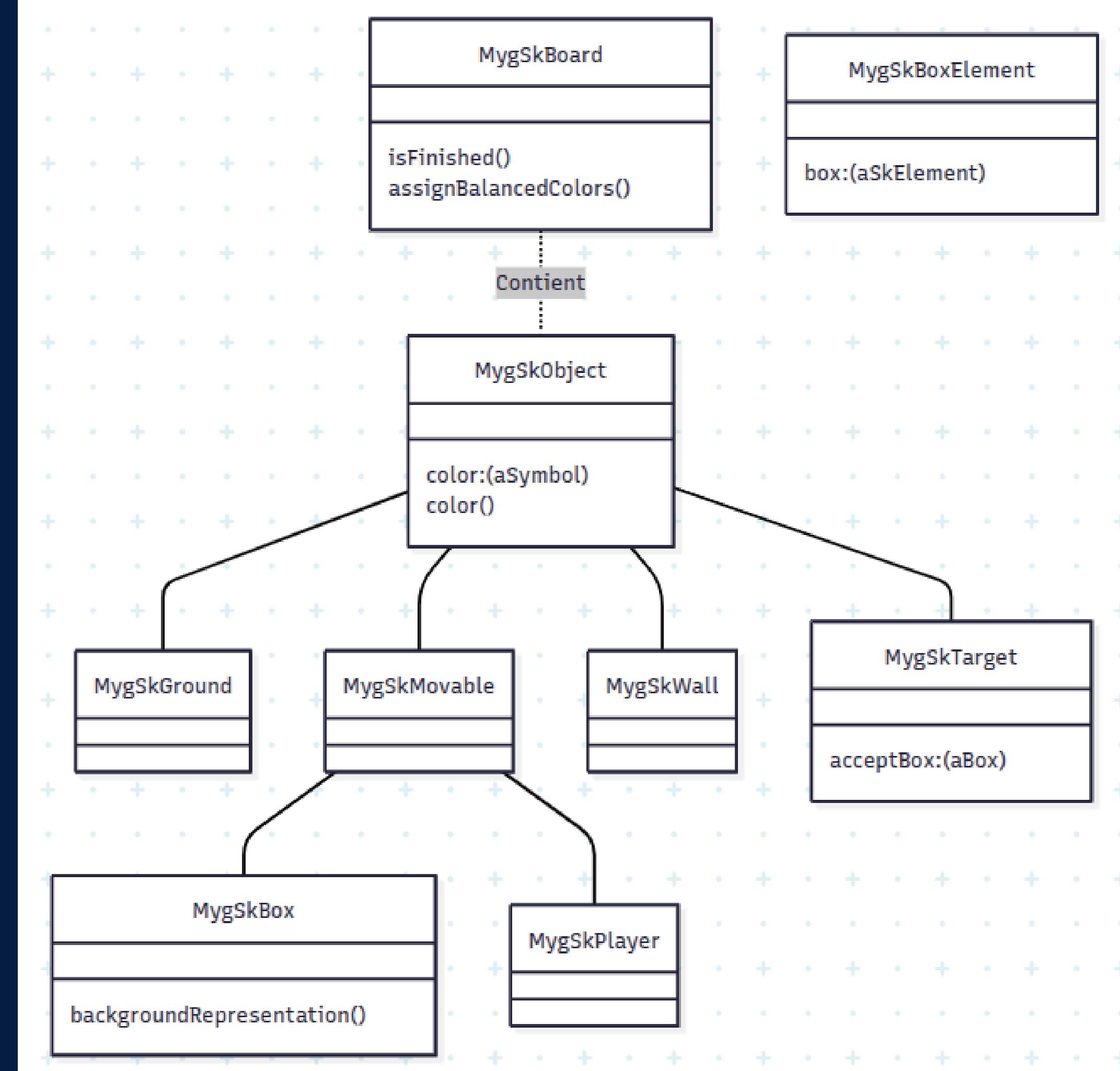
Assignment aléatoire des couleur (rouge,bleu,vert)

## 2 - Ajout de la notion de couleur

Utilisation de l'héritage



# Utilisation de l'héritage



# 1 - Tests (TDD)

Acceptation des boites par les cibles

Nouvelle logique de fin de partie

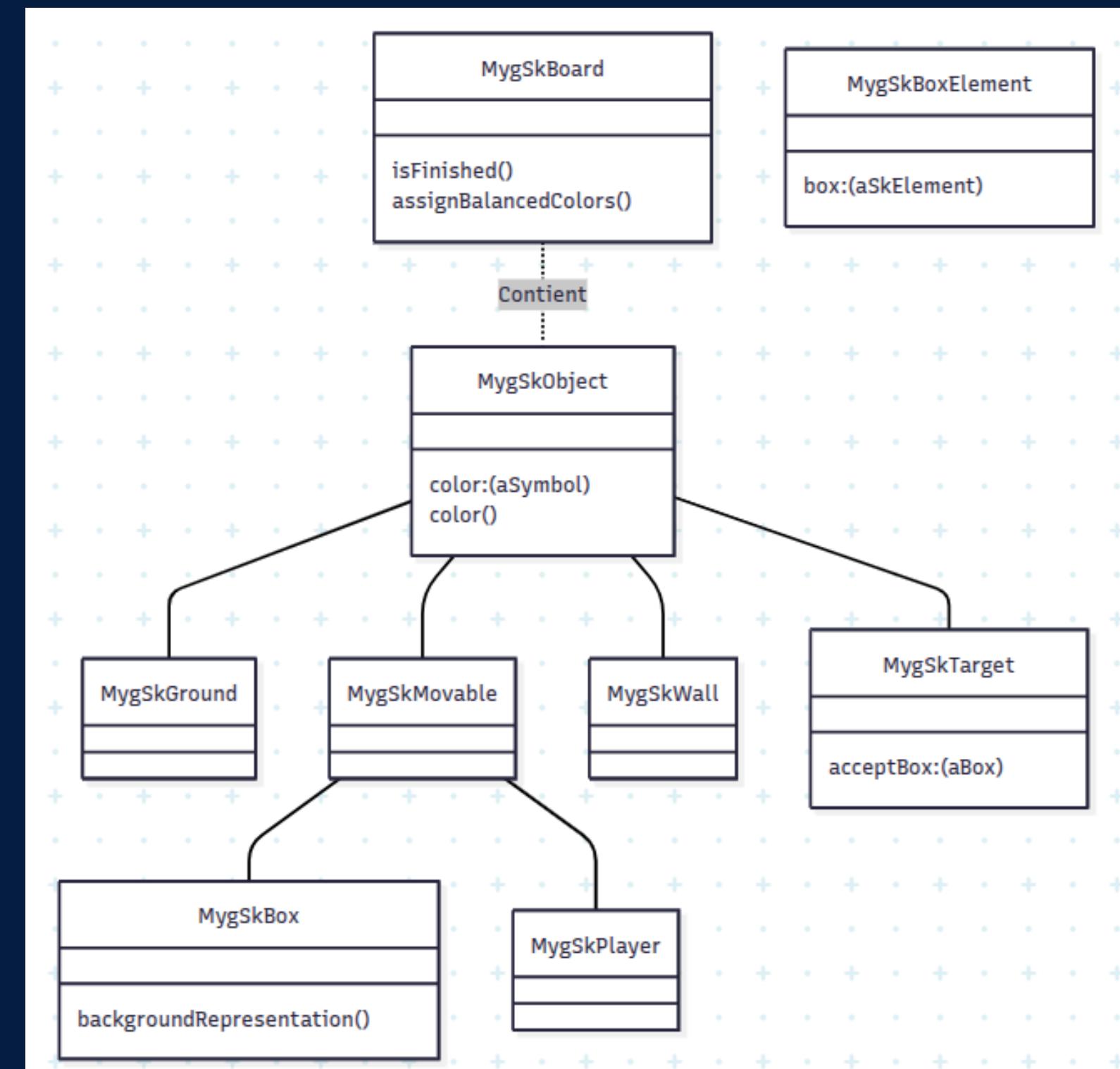
Assignment aléatoire des couleur (rouge,bleu,vert)

## 2 - Ajout de la notion de couleur

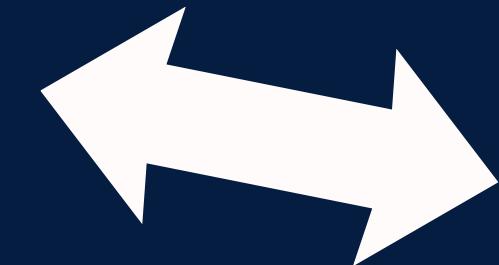
Utilisation de l'héritage

## 3 - Ajout de la logique de fin de partie

Utilisation du double dispatch



## DOUBLE DISPATCH



# Validation d'une boite sur cible

1- Plateau demande à la cible si elle accepte la boite  
(acceptBox:aBox)

2- La cible demande à la boite sa couleur  
color

# 1 - Tests (TDD)

Acceptation des boites par les cibles

Nouvelle logique de fin de partie

Assignment aléatoire des couleur (rouge,bleu,vert)

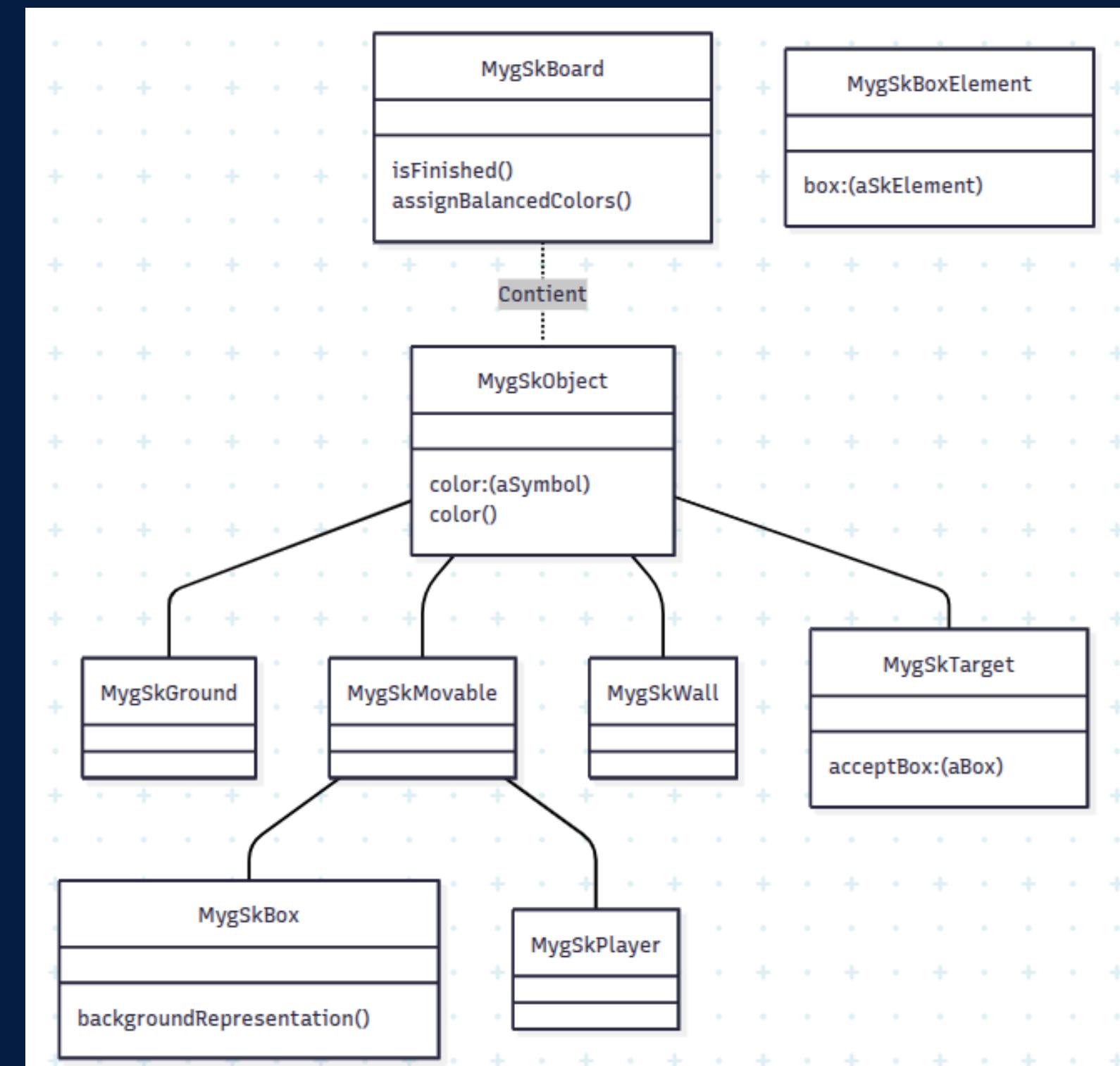
## 2 - Ajout de la notion de couleur

Utilisation de l'héritage

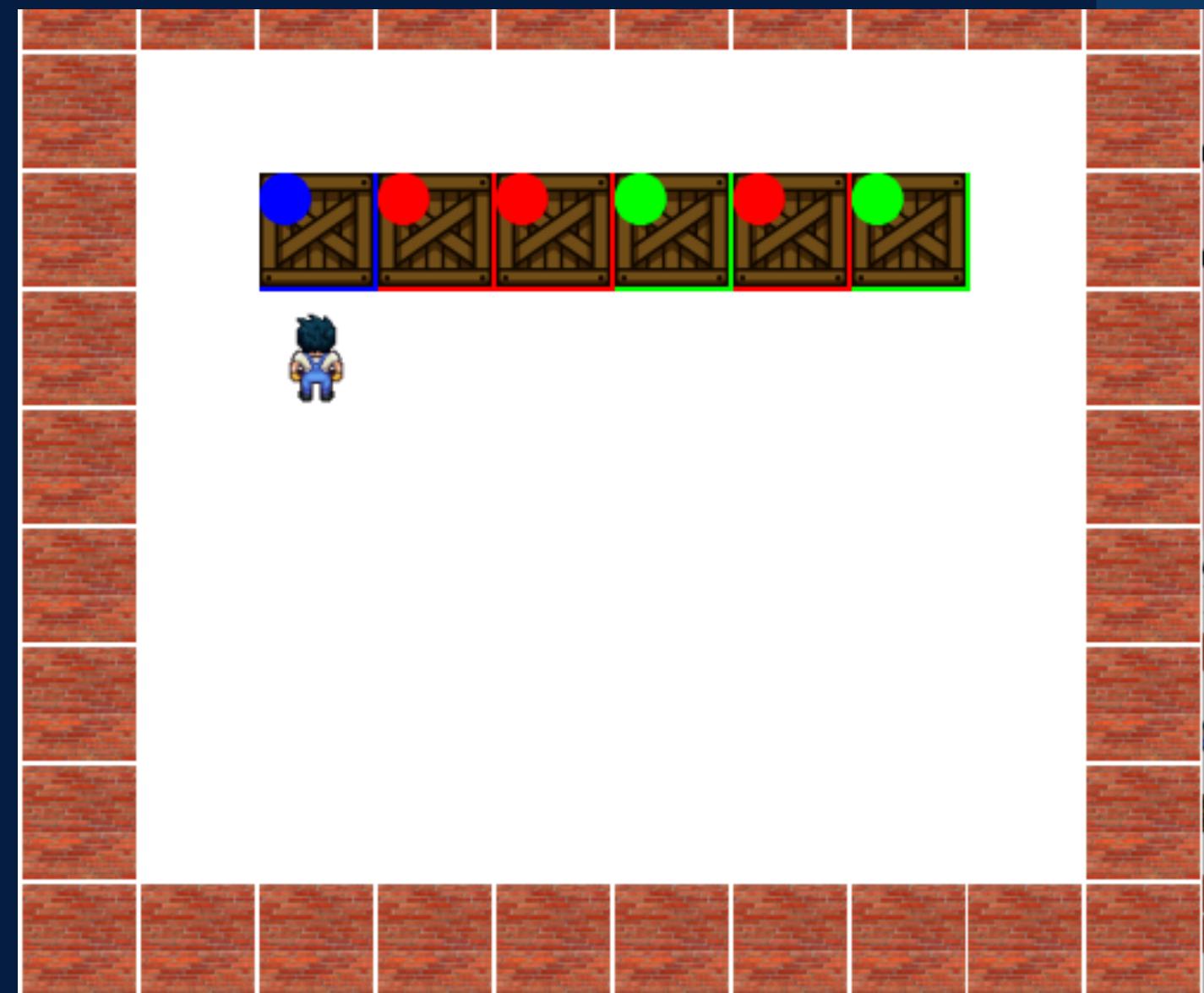
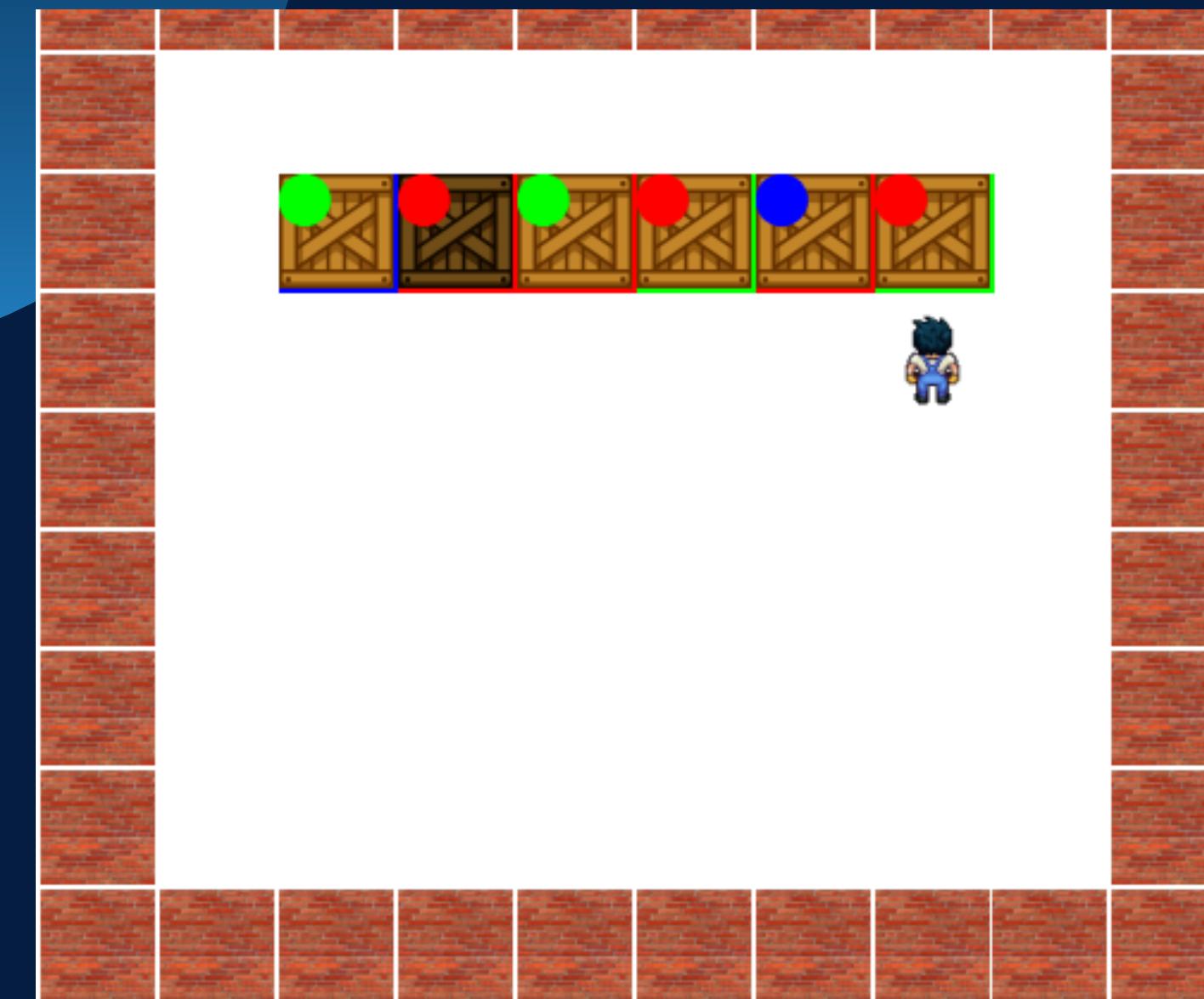
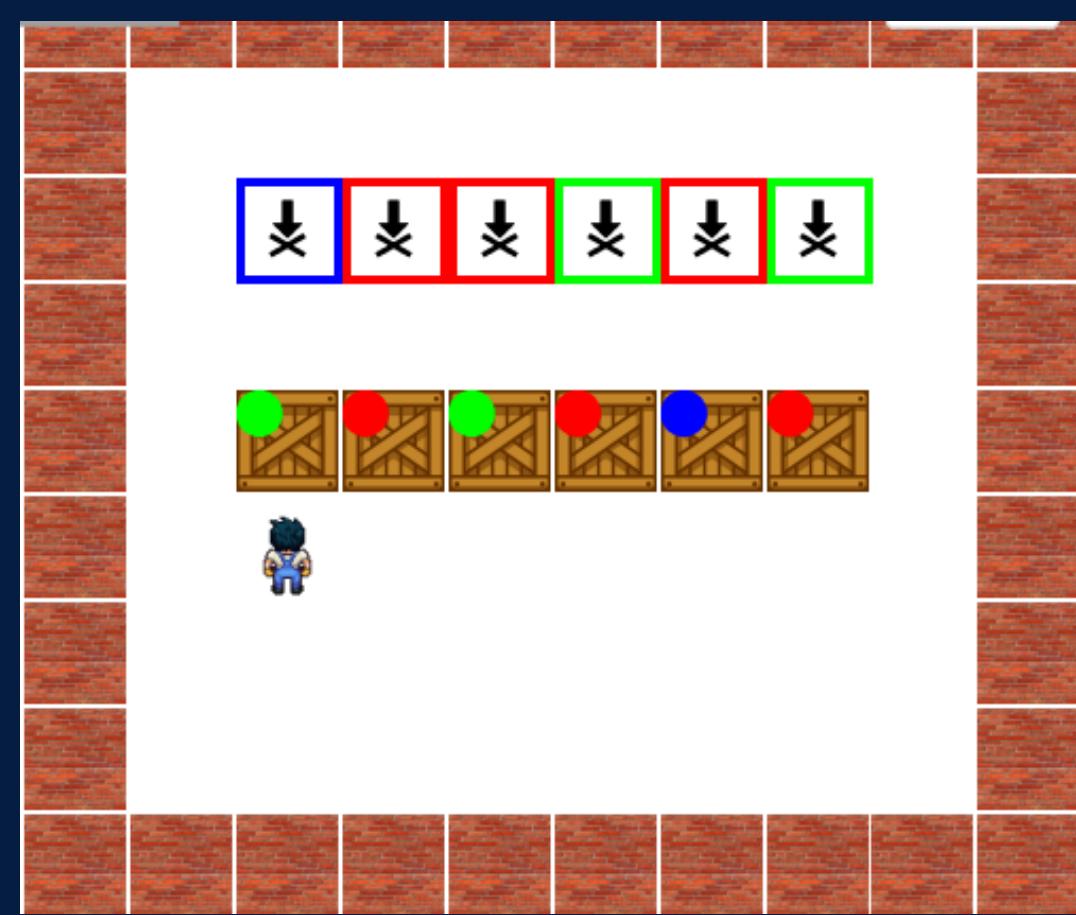
## 3 - Ajout de la logique de fin de partie

Utilisation du double dispatch

## 4 - Affichage visuel



# Affichage visuel



MERCI POUR  
VOTRE ECOUTE

