Rapport Console BabalsYou 54309 - 54383

Nous allons commencer par les règles de déplacement de notre implémentation :
Par défaut pour déplacer le joueur entrer une commande telle que *“move x y dir”* où x représente la position à l’abscisse , y la position à l’ordonnée et dir le sens du mouvement.

Dans le cas où le joueur courant n’est plus l'élément “baba” suit à un déplacement d’une règle les objets correspondants au nouvel élément de la règle “X IS You” peuvent êtres déplacés avec la commande ***“move dir”*** où dir représente le sens du mouvement

Notre rapport donnera des justifications sur le choix de certaines classes et fonctions dont la logique ne paraît pas assez évidente.

- **moveCommand** : Représente une commande basique où 4 entrées sont requises. Exemple : “move 6 5 left”.

- **moveCommandNoPosition** : Utile lorsque l'élément “baba” n’est plus le joueur courant suite à un mouvement des règles du plateau. La position n’est plus requise. Par exemple : **“move left”** permet de déplacer tous les objets représentant le joueur courant sur le plateau de jeu.

Ces classes sont implémentées suite à l'utilisation du pattern command.

- **Position** : Une classe pour représenter les positions sur le plateau de jeu. Elle contient des méthodes pour gérer les coordonnées x et y, ainsi que des opérateurs pour comparer deux positions.

- **Direction** : Une classe pour représenter les directions de déplacement dans le jeu. Elle est utilisée pour déplacer les composants du jeu.Elle a un constructeur privé car on empêche d'instanciation de nouvelles directions.

- **findComponentPosition** : Une fonction qui prend en paramètre le nom d'un composant et retourne la position de la case contenant ce composant. Cette fonction est utile pour localiser un élément spécifique sur le plateau.

- **updateCurrentPlayer** : Cette fonction est utile pour gérer les situations où il y a plusieurs instances d'un composant "you". Si le nom du composant est "baba", les position playerX et playerY sont affectées. Sinon, les positions de toutes les cases contenant un composant dont le nom correspond au newPlayerName sont enregistrées dans un vecteur de positions, qui est une variable membre de la classe Game. La variable player est mise à jour avec un composant correspondant au newPlayerName, et les variables playerX et playerY sont reset à -1 avant de quitter la fonction.

- **move** : Dans la fonction move, plusieurs appels de fonction ont été effectués pour simplifier le code et mieux gérer les différents cas de jeu.
- La fonction de déplacement a été factorisée et divisée en plusieurs fonctions spécifiques à une tâche précise pour améliorer la lisibilité et la maintenabilité du code. Les fonctions sont appelées sont:

- **isInside** : Vérifie si une position est à l'intérieur des limites du plateau de jeu.
- **isElementInRules** : Vérifie si un élément est présent dans les règles du jeu.

- **rules.clear() et rules = extractRules()** : Ces appels de fonction permettent de réinitialiser et de récupérer les nouvelles règles du jeu après chaque déplacement.

- **isLastBoxApplicable()** : Cet appel de fonction vérifie si l'effet d'une règle s'applique à la dernière case d'une séquence d'objet de même effet dans une direction donnée. Cela permet de déterminer si un déplacement est possible ou non en fonction des effets de la règle applicable au dernier element.

- **getPushableComponentsInDirection()** : Cette fonction est utilisée pour récupérer les composants poussables dans une direction donnée à partir d'une position de départ. Cela permet de gérer le déplacement des composants poussables sur le plateau de jeu.

- **clearPushableComponentsFromBoard()** : Cet appel de fonction supprime les composants poussables du plateau de jeu avant leur déplacement. Cela garantit que les composants ne sont pas dupliqués lors de la mise à jour de leur position.

- **updatePositionsInMapWithDirection()** : Cette fonction met à jour les positions des composants poussables en fonction de la direction de poussée. Cela garantit que les composants sont correctement placés sur le plateau de jeu.

- **addComponentsToBoard()** : Cet appel de fonction ajoute les composants poussables mis à jour au plateau de jeu après leur déplacement. Cela permet de mettre à jour l'affichage du jeu en fonction des nouvelles positions des composants.

- **applyEffectsAndMove()** : Cette fonction applique les effets des règles du jeu et déplace le joueur et les autres composants en conséquence. Cela permet de gérer les effets tels que "stop", "sink", "push", "win" et "kill" et d'appliquer les déplacements appropriés aux composants concernés.

- **handleNonRuleComponentName** : Cette fonction gère les cas où le nom du composant rencontré ne correspond pas à un nom de composant de règle (par exemple, "baba", "rock", etc.). Elle vérifie si des effets de règle tels que "stop", "sink" et "win" s'appliquent à ces composants et agit en conséquence.

- **moveNonRuleComponent** : Cette fonction s'occupe du déplacement des composants non liés aux règles (tels que "baba", "rock", etc.) en fonction des règles applicables. Elle détermine si un composant peut être poussé, s'il doit être déplacé dans une direction donnée, ou s'il doit être supprimé du plateau en raison d'effets tels que "sink".

- **applyEffectsAndMove** : Cette fonction applique les effets des règles du jeu et déplace le joueur et les autres composants en conséquence. Cela permet de gérer les effets tels que "stop", "sink", "push", "win" et "kill" et d'appliquer les déplacements appropriés aux composants concernés.

- **handleRuleComponentName** : Cette fonction gère les cas où le nom du composant rencontré correspond à un nom de composant de règle (par exemple, "BABA IS YOU"). Elle détermine si le joueur peut interagir avec ces composants et comment les déplacer en fonction des règles applicables.

- **sinkPlayer** : Cette fonction s'occupe de couler (supprimer) le joueur en cas de rencontre avec un composant ayant l'effet "sink". Elle supprime le joueur du plateau de jeu et réinitialise sa position (playerX et playerY) à -1.

- **movePlayerToNewPosition** : Cette fonction déplace le joueur vers une nouvelle position en fonction de la direction donnée et des règles applicables. Elle met à jour la position du joueur sur le plateau de jeu et modifie les coordonnées playerX et playerY pour refléter la nouvelle position.

Les classes **win, push, you, kill, sink, best** représentent les effets des objets présents sur le plateau

Les classes **rock, wall, baba, grass, water…** représentent les éléments des objets présents sur le plateau

La classe **Is** représente le connecteur.

Fonctionnalités non implémentées :

Les commandes ***save*** et ***load permettent*** respectivement de sauver et charger une partie.