58008 & 58063

17/02/2023

MODELISATION UML (REMISE 1)

1. DIAGRAMME DE CLASSE

A. Package Model

i. **LevelLoader** : permet de créer un Level à partir d'un fichier .text et sera un attribut de la façade Game. Contient un Level en attribut qui sera le niveau 'officiel' de notre jeu.

ii. **Game** : façade de notre jeu, elle est la seule classe qui peut communiquer directement avec le controller. Des méthodes permettent de lancer et détecter la fin du jeu, sauvegarder la partie, bouger les éléments, recommencer un niveau, et revenir en arrière (undo).

iii. **Level** : instancié par le LevelLoader, il contient toutes les infos relatives au jeu, à savoir : la taille + hauteur, la liste des règles actives, une liste d'éléments ainsi que leur position respective. Il y a également des méthodes qui permettent de gérer les éléments ainsi que des méthodes relayant l'état du jeu (isWon, move, findAll qui retourne tous les gameobjects du type passé en paramètre et findAllRules qui scanne l'ensemble des règles actives) et une méthode d'affichage console (to_string).

iv. **ElementType** : enum contenant les 3 types de règles (sujet/verbe/complément) ou une pièce de jeu.

v. **Rule** : représente une règle complète (sujet+verbe+complément). Est donc composée de 3 éléments.

vi. **Element** : type d'élément de jeu. Il s'agit donc d'un élément de règle ou d'une pièce du puzzle.

vii. **GameObject** : object pouvant être placé sur le plateau de jeu. Possède un élément et une position

viii. **Position** : instancie une position sur le tableau pour un élément et sa méthode next qui modifie la position en fonction d'une direction.

ix. **Direction** : représente les points cardinaux nécessaires pour le déplacement d'une position.

B. Package Util

i. Contient les implémentations de classes utiles tels que les design patterns (Observable, Command).

C. Notre implémentation : nous avons fait le choix de représenter l'état actuel du jeu sous forme d'un vecteur de GameObject afin d'alléger la mémoire et être plus performant. Nous avons également un vecteur qui désigne l'ensemble des règles actives pour le jeu. La sauvegarde de partie se fera en réécrivant le fichier txt représenter le jeu.

2. DIAGRAMME DE SEQUENCE

Notre diagramme de séquence commence par un appel de la méthode PlayShot (Game) qui appelle elle-même la méthode move(dir) de la classe Level. Celle-ci trouvera tous les éléments concernés par la règle ‘isYou’ via findAll(). Pour chacun de ces éléments, elle vérifiera si la position de l’élément est valide et contenue dans le plateau de jeu (contains()) et pourra ensuite regarder si l’élément peut-être bougé (via isMovable() – n’est pas bloqué par une règle ‘Stop’ par exemple). Si ces 2 conditions sont respectées, alors l’élément est déplacé (via next()) et sa nouvelle position est analysée pour détecter si le joueur ne doit pas mourir, n’a pas gagné ou n'a pas poussé un autre élément. A la fin du déplacement, on scanne l’ensemble des règles qui ont changées (via findAllRules()).