Groupe: Thomas Van Tuijcom et Thomas Vervondel (58444 - 58594)

QHB - Mr. HOUBEN

Modélisation project Baba Is You

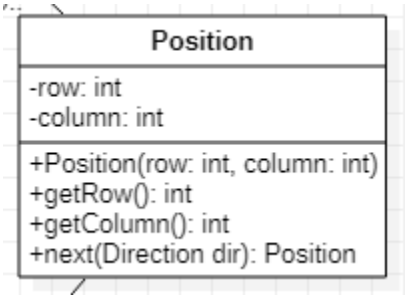
17 février 2023

Baba Is You

Dans le cadre de notre projet de développement 4, nous devions réaliser la modélisation du jeu Baba Is You. Ce document sert comme appui textuel à nos décisions prises lors de la modélisation.

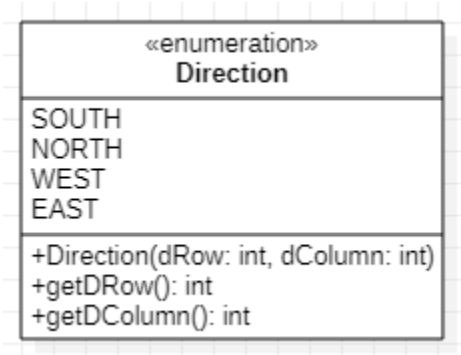
DESCRIPTION DIAGRAMME DE CLASSES

- POSITION :



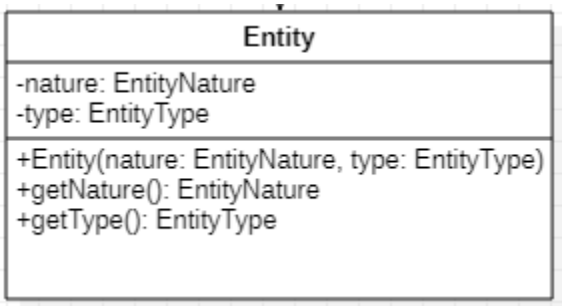
Sert à pouvoir représenter une position donnée du tableau de jeu. Elle possède uniquement des getters à l'exception d'une méthode qui sert à renvoyer la position suivante par rapport à une direction donnée.

- DIRECTION :



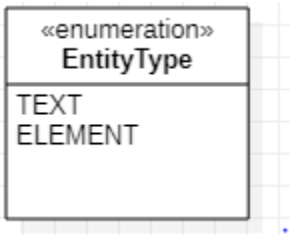
Sert à représenter un déplacement par rapport à une position. L'énumération aura son constructeur privé qui sert à donner la différence sur l'axe X et Y. Exemple: NORTH(-1,0), car on passe sur la ligne au-dessus.

- ENTITY



Sert à représenter tous les éléments du jeu. Elle a une nature qui représente quel élément l'entité est, par exemple: Baba, Wall, Rock, Is, You… Le type nous renseigne si l'objet est un text ou un élément. (On fait la différence entre TEXT_WALL et WALL).

- ENTITYTYPE

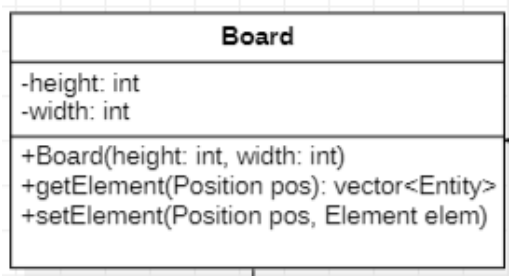


Sert à définir le type d'un objet Entity.

- ENTITYNATURE

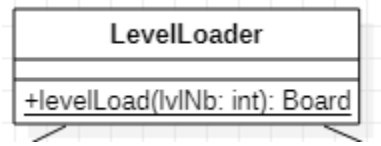
Sert à définir la nature d'un objet Entity.

- BOARD



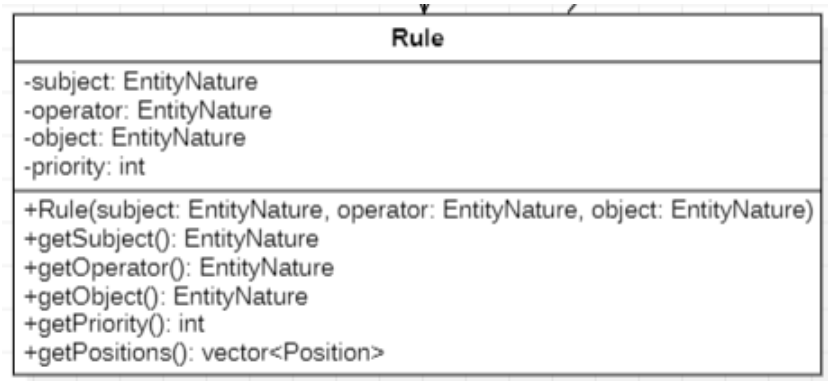
Sert à représenter le tableau de jeu. La classe Board possède aussi un attribut “entities” qui a pour type array<array<vector<Entity>>>. C’est une structure tridimensionnelle pour pouvoir placer plusieurs objets de type Entity sur la même case. La taille et la hauteur n’ont pas besoin d’être dynamiques cependant le nombre d’entités sur une case peut varier.

- LEVELOADER



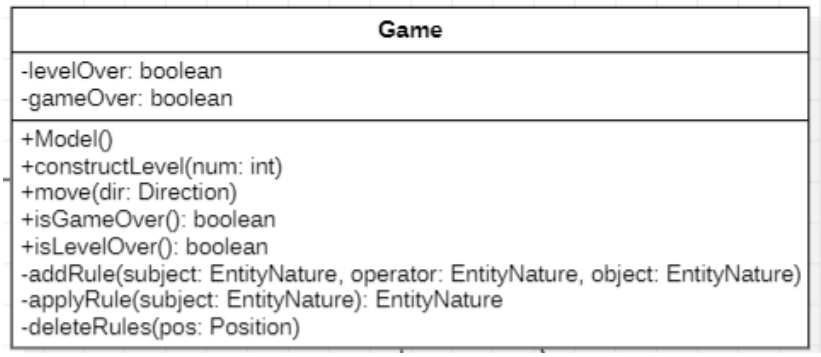
Sert à créer un board qu’on remplit grâce à la lecture d’un fichier. C’est une classe utilitaire.

- RULE



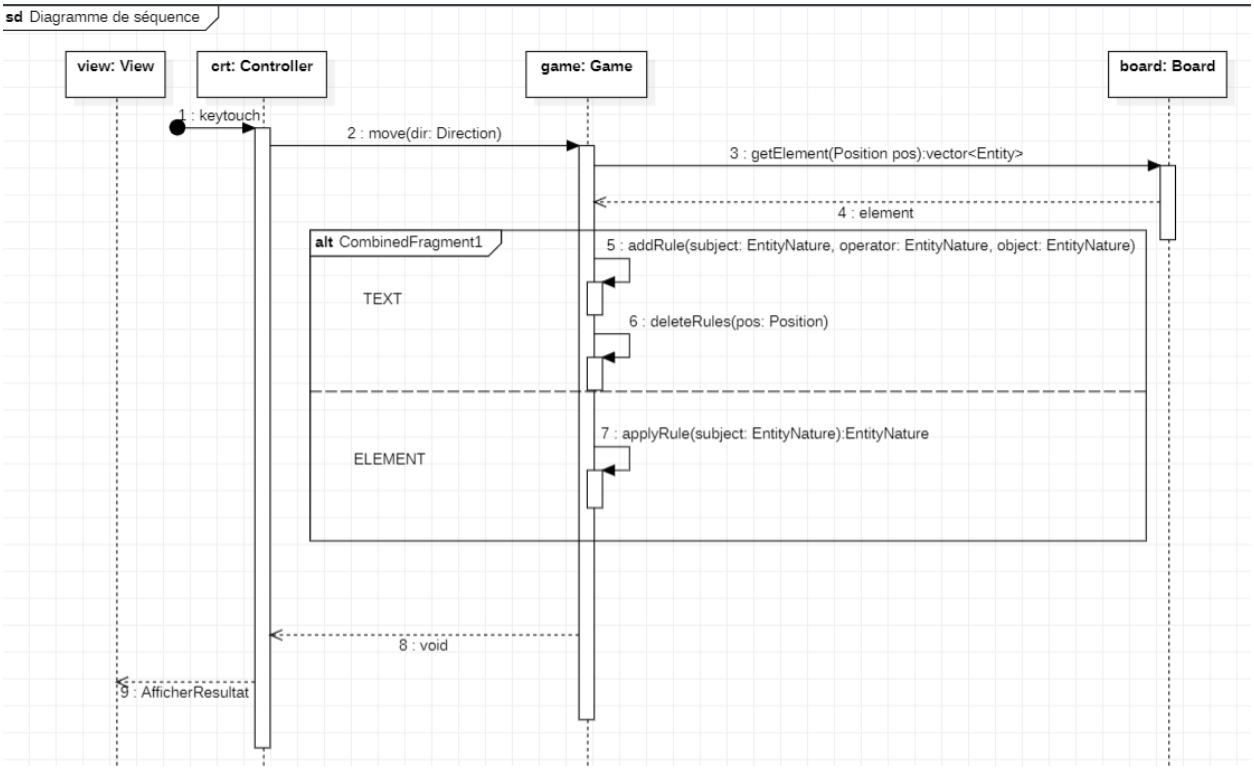
Sert à déterminer le comportement des objets Entity. L’attribut priority a été ajouté afin d’attribuer un ordre d’importance dans certains cas précis. Exemple: deux règles existent telles que “Wall is stop” et “Wall is win” quelle règle prend le dessus?

- GAME



Sert comme façade de notre projet. Elle a une liste de règles du jeu qui sont actives, les règles qui se font désactiver seront supprimées de cette liste. Cette liste est de type vector<Rule>. La construction du niveau se fait dans cette classe mais en appelant la classe utilitaire LevelLoader qui possède une méthode statique.

DESCRIPTION DIAGRAMME DE SEQUENCE



- L'utilisateur va appuyer sur une touche directionnelle du clavier
- Le contrôleur va appeler la méthode move() de son attribut 'game' dont l'argument correspond la touche directionnelle choisie.
- L'attribut 'game' va appeler la méthode getElement()
- qui va retourner un liste d'objet Entity qui se trouve à l'endroit où on veut se déplacer.
- En fonction du type des objets (TEXT ou ELEMENT), on va appeler une certaine méthode.
- Si le type est TEXT, on voit si on peut le pousser. Si on ne le pousse pas , on appelle pas de méthodes.
- Si on le pousse, on voit si on a créé une nouvelle règle ou si on en a détruite une → appel des méthodes addRule() ou deleteRulte().
- Si le type est ELEMENT, on appelle la méthode applyRule() en donnant la nature de l'objet retourné, il va vérifier si on règle existe et si oui , il va renvoyer quelle règle respecter. Donc on saura si on peut pousser, si on est tué , si on a gagné etc …