Documents Pour les SAEs S2 2024-2025

1 Auteurs

Garry Romain Quemener Nicolas, Meilleur Isidore.

2 Diagrammes de classe (15 points pour SAEs S2.01 S2.02)

La classe Entity est très importante car est reliée à plein d'autres classes. Cela montre qu'elle peut être considéré comme au centre du système, comme un point de rassemblement.

On peut voir qu'elle est directement liée à pas mal d'autres classes, ce qui veut dire elle peut donc prendre diverses formes, selon le contexte. Donc, Entity peut servir à représenter plusieurs types d'objets différents.

Ce genre de classe est pratique elle permet :

- de réutiliser le même modèle pour plusieurs types d'entités, sans avoir à tout recoder à chaque fois.

- de faire évoluer facilement le code, on peut rajouter de nouveaux types sans tout casser.

- de centraliser les infos, ce qui aide à gérer les relations et les données plus facilement.

3 Tests Junit : (15 points pour SAEs S2.01 S2.02)

Nous avons réalisés des jeux de tests pour les classes Character, Field et ItemStock car elles font parties des classes les plus importantes du projet et permettent un bon déroulement un fonctionnement du jeu. La classe ItemStock est plus là pour s'assurer qu'aucun problème ne sera rencontrer lors de la création et/ou utilisation d'un item.

4 Structures de données (15 points pour SAEs S2.01 S2.02)

Nous avons utiliser des LinkedHashMap pour réaliser l'inventaire dans les classes de création du modèle des personnage mais aussi dans la classe ItemStock afin de gérer correctement le craft de nos objet avec la classe CraftController.

Des listes d’énumération dans la classe ItemStocks pour les différents types d’item nous a permit de créer des constante et donc gagner du temps au niveau de l’utilisation des objets ce qui nous évite de devoir recréer un objet à chaque fois que l’on veut l’ajouter à l’inventaire.

5 algorithmique (30 points pour SAEs S2.01 S2.02)

Les algorithmes les plus intéressants que nous avons coder sont celui du *BFS* permettant que les mobs aillent attaquer le joueur en parcourant le chemin le plus efficace, il est présent dans sa propre classe.

Nous avons aussi utiliser un algorithme de *« time line »* nous permettant de réaliser des actions répétitives en évitant de provoquer des boucles infini comme si nous l’avions fait avec une boucle *while ou for.* Il est présent dans le contrôleur principal qui porte le même nom.

Enfin un dernier algorithme que nous pourrions citer même si de nombreux autres méritent d’être présent ici, l’algorithme permettant au logiciel de vérifier si le joueur se retrouve bien avec avec 0 points de vie et gère donc l’affichage de game over à l’aide d’un *listener* sur la barre de vie. Cet algorithme est situer à la fin du contrôleur principal