Projet Design Pattern

Nom : TOURE
Prénom : Boubacar
Num-étud : 18.00.94.22
Date : 12/12/2021 à 00 h 45 min à Angers
Niveau : Master 1 Informatique
But : Le but de ce projet est principalement lié à la mise en place d'un jeu **BOMBERMAN** suivant les principes fondamentales du design pattern.

Ce projet est accessible sur mon compte github à l'adresse ci-dessous :
https://github.com/TBoubacar/projetJavaDesignPattern.git

université angers

SOMMAIRE

université angers

I. Introduction

◆ Règles du jeu

Les règles du jeu sont les suivantes :

- Le jeu se déroule dans un monde limité (Plateau) sur lequel évolue un certain nombre d'agents.
- Au début du jeu, tous les agents sont créés et placés sur le Plateau suivant une configuration définie par l'utilisateur.
- A chaque tour t du jeu, chaque agent réalise une action prédéfinie par un ensemble de règles qui impacte l'environnement (c'est-à-dire entraîne un changement d'état du Plateau de jeu car tous les agents seront mise en mouvement de manière simultanée).
- Une fois que le nombre de tours maximum est atteint ou qu'une condition spécifique de fin de jeu est atteinte, alors le jeu s'arrête.
- Seuls les agents capables de voler, peuvent se déplacer au-dessus des murs. Qu'elles soient destructibles ou indestructibles (les murs à l'intérieur du Labyrinthe).
- Si un agent **Bomberman** se trouve sur la même case qu'un agent **PNJ**, il est mangé et disparaît de l'environnement de jeu.
- Un agent **Bomberman** peut décider de mettre une bombe à son tour t au lieu de se déplacer. La bombe explose après un nombre de temps T-bombe. L'explosion se propage dans les 4 directions sur un certain nombre de cases C-bombes. Exemple après 5 secondes la bombe explose 4 cases à partir de sa position initiale vers le haut, bas, gauche et droite.
- A la destruction d'un mur (mur destructible), il y a des possibilités d'avoir un objet bonus (items) à la place du mur détruit. L'agent **Bomberman** peut s'emparer de cet objet en se positionnant dessus.
- Le jeu se termine si tous les **Bombermen** sont éliminés (Victoire des agents PNJ) ou si un agent **Bomberman** est le seul survivant (Victoire de cet agent **Bomberman**). Donc par conclusion, les agents **Bomberman** sont capables de se tuer entre eux.

L'environnement du jeu est constitué des agents Bomberman et PNG et d'un labyrinthe avec : des cases libres, des murs destructibles, des murs indestructibles et éventuellement des items.

N.B. Le labyrinthe est entouré par des cases indestructibles non franchissable.

◆ Objectifs

L'objectif de ce projet est d'implémenter le jeu Bomberman de manière interactive avec une interface visuelle de la façon la plus modulable et extensible possible en utilisant les principes du Design Patterns vus en cours. On implémentera plusieurs modes de jeu : en mode coopératif ou en mode affrontement avec un ou plusieurs agents **Bomberman** ainsi que des Intelligences Artificielles**.**

II. Développement

◆ Packages, Classes et Interfaces

Pour la réalisation de ce projet, j'ai eu à implémenter plusieurs classes et interfaces que j'ai pu stocker dans différents packages.
Le projet est constitué au total de sept (7) packages parmi lesquels nous avons :

- **agent :** contenant la liste des classes et interfaces liées à l'implémentation d'un agent.
- **controller :** contenant l'ensemble des différents controller de notre jeu.
- **etat :** contenant l'ensemble de classes et interfaces assurant l'implémentation du design pattern Etat.
- **model :** contenant les différentes classes utilisées comme modèle du jeu.
- **stratégie :** contenant l'ensemble de classes et interfaces assurant l'implémentation du design pattern Stratégie.
- **utils :** contenant l'ensemble des outils nécessaires pour le fonctionnement du jeu.
- **view :** contenant différentes classes ayant pour but d'assurer l'affichage de l'interface du jeu.

Le projet est tout aussi constitué de différents dossiers tels que : le dossier **src** contenant les fichiers java du projet, le dossier **layouts**, **images** et **icons** permettant d'assurer la gestion de l'affichage graphique du terrain de jeu ainsi que de l'interface de commande.
Le principale rôle de ces dossiers sont les suivants :

- **layouts :** contenant l'ensemble des différents panneaux du jeu bomberman.
- **images :** contenant les images représentatives des différents agents ainsi que des informations sur le terrain de jeu (murs, items, etc).
- **icons :** contenant l'ensemble des icônes utilisées pour l'affichage de l'interface de commandes.

◆ Les différents types de design pattern utilisés

Lors de l'implémentation de ce projet, j'ai eu à utiliser plusieurs principes fondamentales du design pattern parmi lesquels nous avons :

- Le design pattern **stratégie** : pour le déplacement des agents.
- Le design pattern **état** : pour l'activation du minuteur des bombes ainsi que pour la gestion des différents état du jeu (état pause, état play, état start, état restart, état fin, etc).
- Le design pattern **patron de méthode** : implémenter au sein de la classe Agent afin que les sous classes (agent bomberman, bird, etc) ne puissent pas modifier le fonctionnement de base de la méthode *moving() assurant le déplacement.*
- Le design pattern **fabrique abstraite** : pour la création des différents agents sans avoir à modifier les codes assurant la création. Pour ce faire, j'ai utilisé une usine de création d'agent, chose qui m'a permis de réduire de 90 % le nombre de lignes de codes pour l'instanciation des agents (Agent bomberman, ennemis, bird, raijon)
- Le design pattern **Singleton** : pour la création d'une unique usine de création d'agents au sein du projet. Grace à cette façon de faire, même si l'on souhaite ajouter un nouveau type d'agents au sein de notre jeu, notre code sera totalement fermé à la modification et il n'y aura qu'une unique usine qui se chargera de la création des agents.
- Le design pattern **MVC** : pour la création des modèles, vues et contrôleurs.

- Le design pattern **oberservateur** : pour assurer la mise à jour des interfaces dès qu'il y a une modification observée.

◆ Fonctionnalités supplémentaires et implémentation de l'Intelligence Artificielle du jeu Bomberman

Les différentes fonctionnalités supplémentaires que j'ai eu à implémenter au sein de mon projet sont les suivantes :

- Ajout des informations concernant le déroulement du jeu. Ceci pourrait-être directement observable à l'aide de la console (prise d'une bombe par un agent, prise d'un item en particulier, l'incapacité pour agent de mourir car il est protégé par un item ou son incapacité à poser des bombes, mise en pause du jeu, signalement de la fin d'une partie ou de la mort d'un agent en spécifiant sa position, etc)
- Ajout d'un bouton au sein de l'interface de commandes permettant d'assurer le changement de l'arène du jeu.
- Ajout d'un bouton permettant d'assurer le changement de mode du jeu (le fonctionnement du jeu en coopératif ou non coopératif). Cette fonctionnalité est accessible via l'interface de commandes comme toutes les autres fonctionnalités supplémentaires. Un clic sur le bouton permet de changer le mode du jeu et ceci est aussi valable au cours du déroulement du jeu.
- Amélioration de l'interface et de la gestion du jeu à l'aide de l'intégration d'un tableau de bord scrollable contenant ainsi l'ensemble des informations utiles sur le jeu en cours (le nombre de tours maximum, le nombre de bomberman en vie ainsi que les agents PNG, affichage d'un message de fin de partie « affichage du vainqueur de la partie »).
- Ajout de l'IA permettant aux agents **bomberman** de repérer les agents PNG à 2 mètres de leurs positions. Ce qui leur mettront en mode alerte et leur forceront à poser des bombes successivement autour d'eux pour se protéger tout en s'éloignant le plus possible des agents PNG en prenant des directions opposées à ces derniers. Par contre si le **bomberman** à des bombes de range > 2 alors il dévient plus confiant et attaque les agents PNG de front en se rapprochant de leurs positions (à voir lors de l'exécution du jeu). L'IA mis en place permet aussi aux agents **bird** de repérer les agents bomberman à 5 mètres de leurs positions. Ce qui leur permettront de s'éveiller de leur sommeil et de se lancer à l'attaque sur un agent bomberman repéré jusqu'à ce qu'il arrive à le manger.
- Ajout d'un bouton d'arrêt permettant à l'utilisateur de fermer automatiquement l'ensemble des différentes interfaces du jeu sans avoir à le faire manuellement.
- Ajout de boutons radios permettant à l'utilisateur de choisir l'UIManager sur les interfaces du jeu assurant ainsi un affichage plus adapté et ergonomique.

III. Conclusion

◆ Les valeurs ajoutées

Au cours de la réalisation de ce projet, je me suis rendu compte de la présence de nombreuses fonctionnalités manquantes parmi lesquelles nous pouvons considérer :

- Il aurait été très intéressant d'intégrer la possibilité de lier les déplacements d'un agent bomberman aux différentes touches de notre clavier (<gauche> ; <droite>;<haut> ; <bas> et <entrer> pour poser des bombes). Dû à un manque de temps et afin d'éviter tout problème étant proche de la date de remise du projet. Je n'ai pas pu intégrer cette fonctionnalité, mais je la réserve pour plus tard.
- Qu'avec le type d'implémentation choisi pour la gestion des interfaces graphiques (ViewCommand & viewSimple or BombermanGame), on n'est pas fermé à la modification lors de l'ajout d'une nouvelle vue. Pour une meilleure implémentation, il faudrait créer une classe mère View dont hériterait toutes les classes de types View. Et ainsi, on créera une ArrayList permettant de stocker l'ensemble des vues de notre jeu dans le contrôleur et ainsi les créer et les ajouter en tant qu'observateur de manière automatique à l'aide d'une méthode qui sera appelé au sein du constructeur de notre contrôleur. Cette méthode permettra d'ajouter une nouvelle vue à la demande sans avoir à modifier la classe du contrôleur.

  Néanmoins, vu que je suis en manque de temps et que je n'ai pas l'intention de risquer de modifier toute la structure de mon projet étant si proche de la date de remise du projet, j'ai préféré avancer comme cela afin de ne pas créer des bugs au sein de mon projet.

N.B. On supposera que notre jeu n'aura plus d'autres vues à implémenter pour l'instant.

◆ Fin du projet

Il existe encore de nombreuses fonctionnalités que nous pourrions améliorer ou intégrer au sein de notre jeu. Mais pour l'instant, je considère que pour une première version du jeu, elle est terminé à 100 %.

Le fichier java en charge du lancement de mon projet se situe au sein du package **utils.** Son contenu a été réduit le plus possible (une ligne de code permettant d'exécuter tout le projet en entier). L'utilisateur n'aura pas à y toucher n'y à le modifier. Toute l'interaction avec la machine concernant notre jeu se fera directement sur l'interface graphique.

Néanmoins, si l'envie vous tentes de vérifier le fonctionnement du SimpleGame, vous pourriez ouvrir le fichier test.java et enlever les commentaires empêchant son exécution.

N.B. Les deux interfaces peuvent être lancer simultanées, ceci est une petite subtilitée que j'ai mis en place afin que vous puissiez avoir le plaisir de faire une vérification rapide et pousser du bon fonctionnement de mon code.