Rapport du jeu de stratégie au tour par tour

Mouhamadou Ousmane MBAYE
Papa Farba NDOUR
Ibrahima DIOUF
Madina DIENG

Novembre 2018

Contents

Chapter 1

Objectif du projet

Ce projet a pour objectif de réaliser un jeu de combat qui oppose 2 à n joueurs sur une grille carrée de taille paramétrable. À chaque tour de jeu, un combattant peut réaliser une action et essaye de combattre ses autres adversaires dans la grille.

1.1 Description du concept derrière le jeu

Le jeu de stratégie au tour par tour à fait son apparition à la fin des années 1970. Comme son nom l'indique, les actions sont réalisées au tour par tour et le joueur doit donc attendre son tour pour jouer selon un ordre initialement défini. Le jeu se caractérise également par une gestion des énergies des joueurs et des armes en fonction des différents combattant. Ces derniers disposent en effet d'une quantité d'énergie limitée et variant d'un type de combattant à un autre.Chaque combattant possède un certains nombre d'armes avec ses minutions.

Chapter 2

Architecture du projet

2.1 Diagramme de classes

La figure 2.1 représente le diagramme de classes des entités utilisées dans notre jeu.

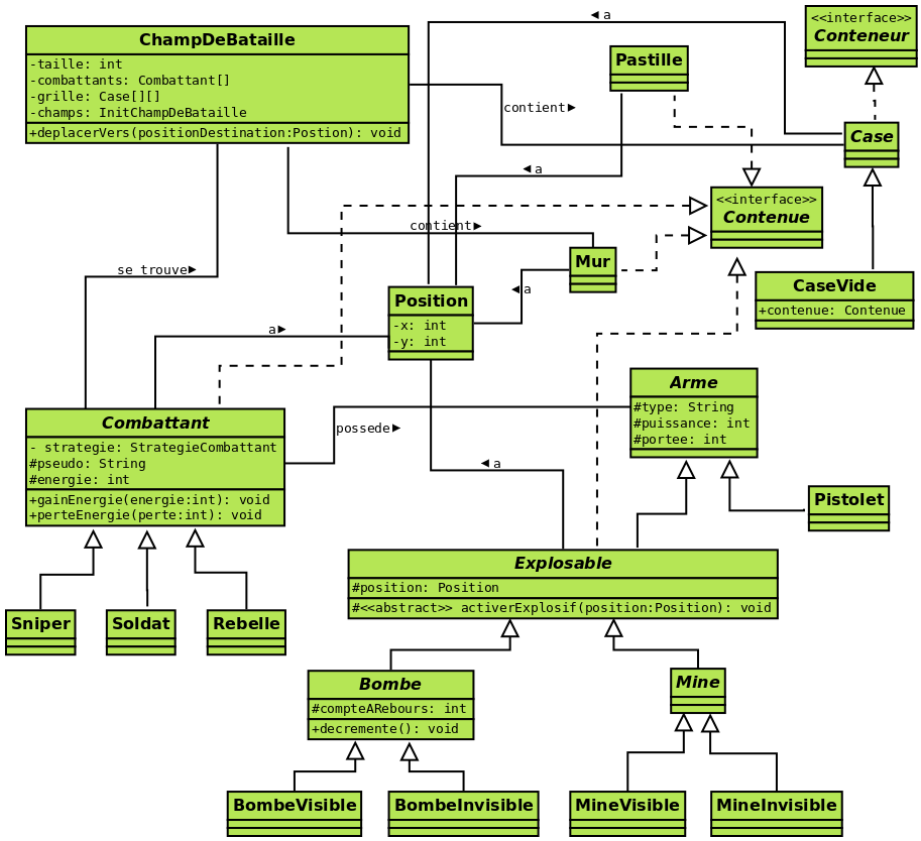

Figure 2.1: Diagramme de classes

On est partie d'une **CaseVide** qui est un **Conteneur** contenant une instance de **Contenue**. Sur cette dernière heritent les classes **Explosable**, **Mur**, **Pastille**, **Combattant**. Notre classe **ChampDeBataille** étant considérée comme 'principale' contient une matrice de **CaseVide** et un attribut champ qui permet d'initialiser notre matrice.

2.2 Diagramme de classes du pattern proxy

La figure 2.2 représente le diagramme de classes de la mise en œuvre du pattern proxy.

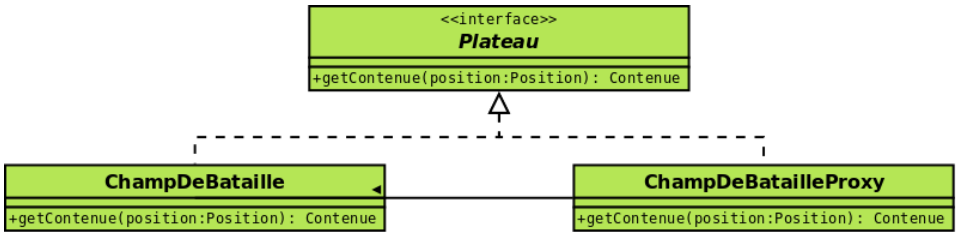

Figure 2.2: Diagramme de classes du pattern proxy

2.3 Diagramme de classes du pattern factory

La figure 2.3 représente le diagramme de classes de la mise en œuvre du pattern factory.

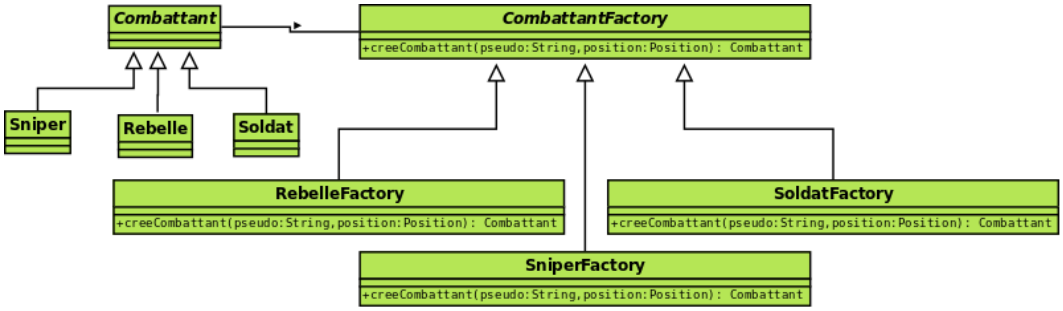

Figure 2.3: Diagramme de classes du pattern factory

2.4 Diagrammes de classes des pattern stratégies

La figure 2.4 représente le diagramme de classes de la mise en œuvre du pattern stratégie pour l'initialisation de la grille et La figure 2.5 représente celui de la mise en œuvre du pattern stratégie pour les strategies d'actions des combattants.

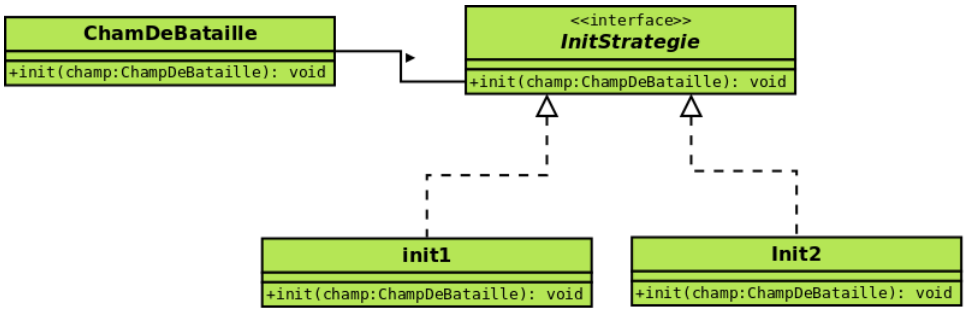

Figure 2.4: Diagramme de classes du pattern stratégie pour l'initialisation

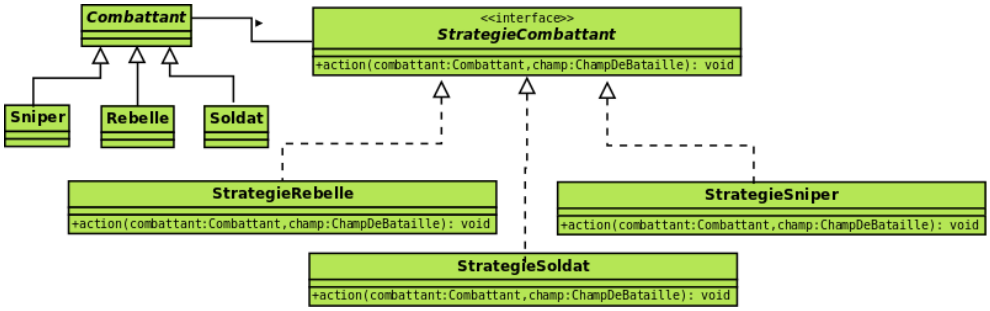

Figure 2.5: Diagramme de classes du pattern stratégie pour l'action des combattants

2.5 Diagramme de classes du pattern adapter

La figure 2.6 représente le diagramme de classes représentant la mise en œuvre du pattern adapter.

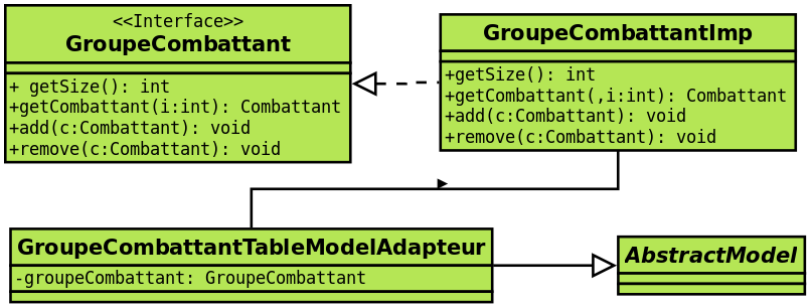

Figure 2.6: Diagramme de classes du pattern adapter

2.6 Diagramme de package

La figure 2.7 représente le diagramme de package qui reflète l'organisation de paquetages et de leurs éléments.

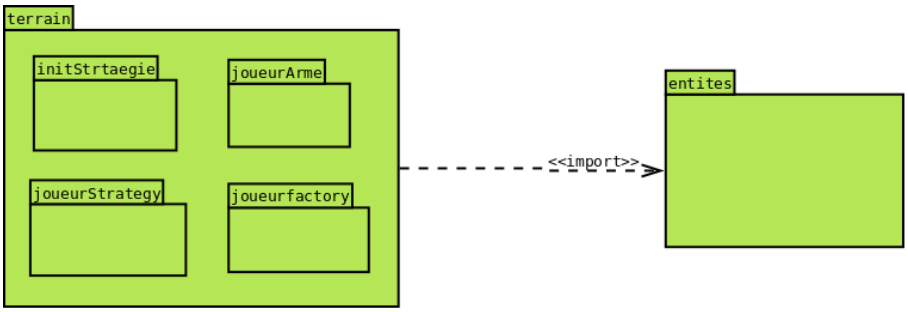

Figure 2.7: Diagramme de package

Chapter 3

Fonctionnalités implémentées

3.1 Description des fonctionnalités

Les patrons de conception (en anglais Design Pattern), appelés aussi modèles de conception ou motifs de conception, sont un recueil de bonnes pratiques de conception pour un certain nombre de problème récurrent en programmation orientée objet. Dans le cadre de notre projet, on a utilisé quelques patrons de conception à savoir:

- Modèle Vue Controlleur (MVC) : c'est un modèle destiné à répondre aux applications interactives en séparant les problématiques liées au différents composants au sein de leur architecture respective.

- Pattern proxy : ce pattern est utilisé pour donner à chaque combattant une vue partielle de notre jeu. En effet seul les explosifs marqués visibles par les combattants qui les ont déposés sont visibles par les autres combattants. Ainsi les explosifs marqués invisibles sont invisibles pour tous les combattants sauf ceux qui les ont déposés.

- Pattern stratégie : il est utilisé deux fois dans notre projet. La première fois pour l'initialisation de notre grille de jeu et la seconde pour définir les stratégies de déplacement et de combat de chaque combattant.

- Pattern factory : elle permet d'instancier des objets dont les types sont dérivés d'un type abstrait, dans notre application ce pattern a été utiliser pour déléguer la création des combattants à une autre classe.

- Pattern adapter : grâce à ce pattern nous avons pu gérer l'affichage d'un groupe de combattant dans un Jtable.

3.2 Organisation du projet

D'abord nous nous sommes retrouvés pour discuter du diagramme UML. Ensuite, on s'est partagé les tâches selon les compétences de chacun, et enfin chacun a travaillé sur sa partie qu'il met en ligne sur forge. On s'est beaucoup entre-aidé sur certaines parties qui font l'objet de réflexion. Ainsi nous avons pu continuer le projet de manière plus rassurant.

Chapter 4

Expérimentation et usages

Le jeu démarre avec une fenêtre de dialogue permettant d'indiquer le nombre de combattant. Après choix du nombre de combattant, une autre fenêtre permettant de choisir le mode d'execution souhaité s'ouvre.L'utilisateur a le choix entre deux modes : mode console et mode interface graphique. Une fois le mode déterminé le jeu commence automatiquement et à chaque tour de jeu, les joueurs jouent l'un après l'autre selon un ordre défini. Un combattant est éliminé du jeu lorsque son énergie est négative ou nulle. Il disparaît alors du jeu et le gagnant est le dernier survivant.

4.1 Capture d'écrans

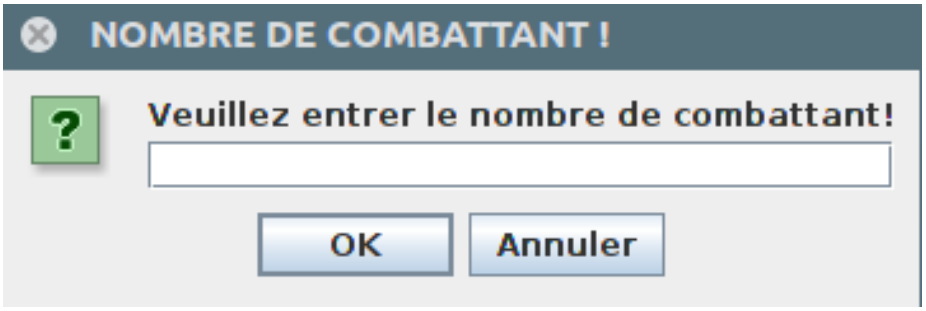

Figure 4.1: Choix du nombre de combattant

Ce schéma représente la fenêtre de dialogue qui nous permet de renseigner le nombre de combattant du jeu.

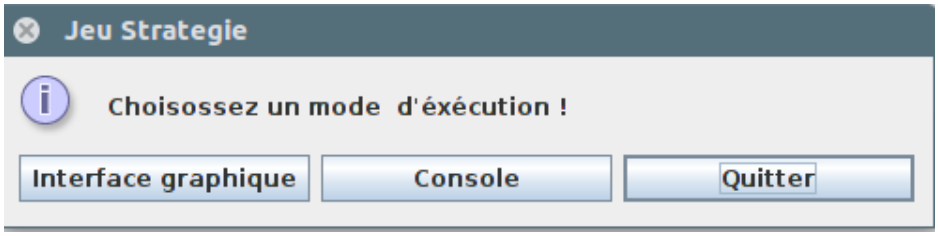

Figure 4.2: choix du mode d'exécution
Ce schéma montre le choix du mode d'exécution que l'on souhaite utilisé.

Figure 4.3: Déroulement du jeu
Ce schéma montre la phase de Déroulement du jeu.

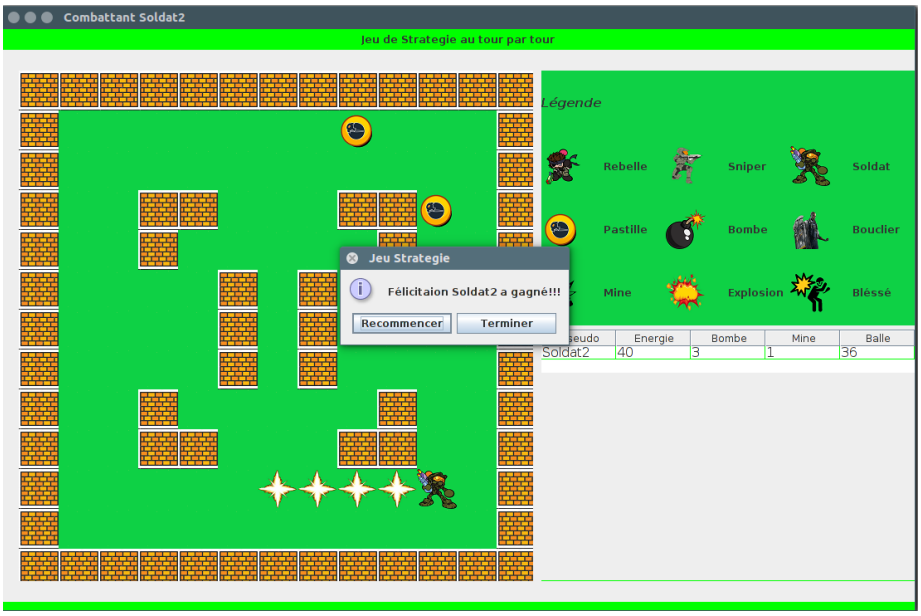

Figure 4.4: Fin du jeu

Ce schéma montre la phase final du jeu.

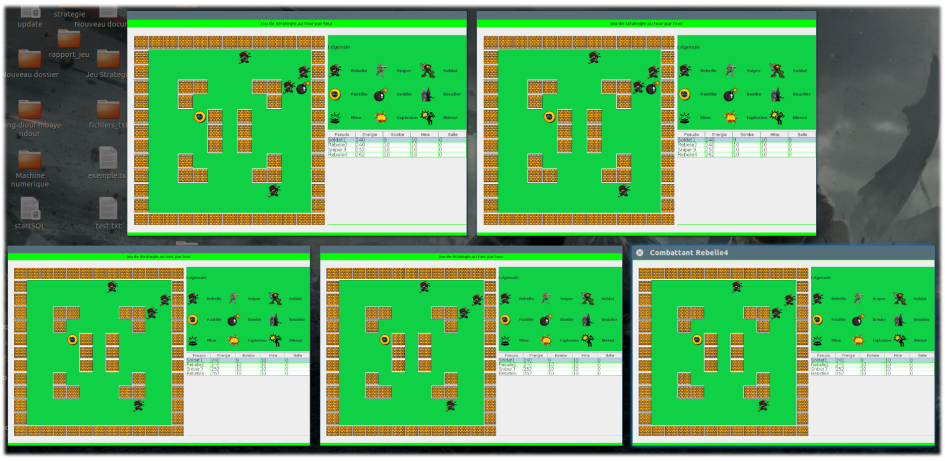

Figure 4.5: Mise en oeuvre du pattern Proxy

Chapter 5

Conclusion

L'implémentation des patterns (proxy, stratégie, factory et adapter) dans ce jeu nous a facilité non seulement la réalisation de certaines fonctionnalités qui paraissaient complexe, mais nous a permis aussi d'optimiser le code avec une possibilité d'extension dans le futur.
Toutefois l'intelligence artificielle pourrait être mis en œuvre pour une amélioration approfondit des stratégies des combattants, avec des actions bien plus réfléchit.