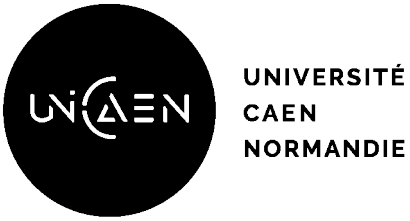

Rapport Méthode de Conception

Jeu de Stratégie

TROTOUX Charlie - 21716001

VU Nguyen Phuong Vy - 21911658

LE GUIRIEC Timothée

L3 Informatique - Année 2021-2022

9 décembre 2021

Table des matières

1 Introduction

Ce jeu de stratégie est un jeu de combat sur une grille d'une taille de dimension choisie entre 10 et 20 cases de largeur et de longueur, remplie par des murs, des bonus boucliers, des bombes et des mines. Chaque joueur dispose d'un nombre de points de vie, d'une arme avec une certaine portée et une quantité de dégats. Les joueurs peuvent se déplacer horizontalement et verticalement. Le dernier joueur à être en vie est le gagnant de la partie.

2 Utilisation du programme

Pour utiliser le programme, se placer dans un terminal dans le repertoire "livraison" et lancer la commande "ant", des paramètres d'initialisation seront demandés dans le terminal avant de lancer le jeu. Il faut entrer la dimension du plateau dans laquelle le joueur souhaite jouer, la quantité d'obstacles, un mode de jeu, et enfin le nombre de joueurs. Si le valeurs entrée sont valides la fenêtre du jeu s'affiche.

3 Conception du programme

3.1 Présentation

Le jeu à été réalisé selon le MVC pattern (Model-View-Controller) pour la gestion de l'interface graphique, le Proxy pattern pour permettre différente vue des objets dans l'interface graphique, et le Strategy pattern pour permettre différents mode de fonctionnement du jeu.

3.2 Diagrammes

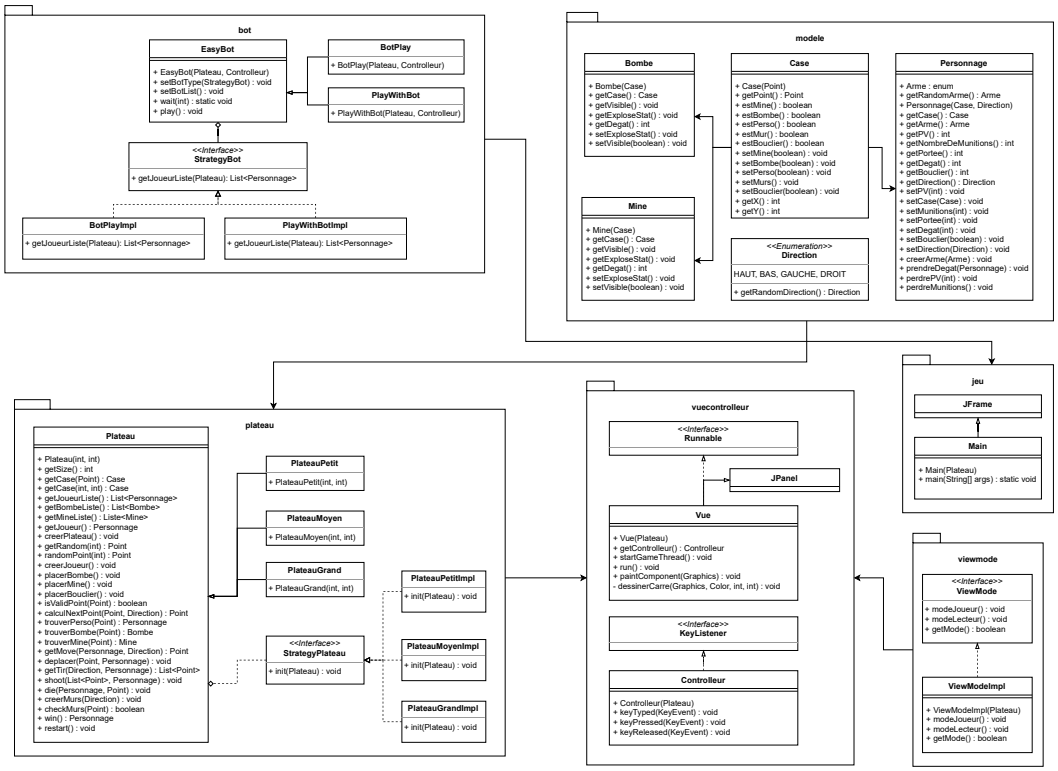

FIGURE 1 – UML

3.3 Proxy Pattern

Le but du Proxy Pattern est de ne faire voir qu'une partie des données réelles. Il permet de cacher les bombes et le mines pour le joueur. On a l'interface **ViewMode** avec les méthodes `modeJoueur(), modeLecteur(), getMode()`.

La méthode `modeJoueur()` permet de cacher l'affichage des obstacles au joueur et vice-versa avec `modeLecteur()`. Enfin, on ajoute cette interface au package **vue-controlleur** permettant de construire et appeler le mode vue joueur ou non avec un bouton prédéfinit sur le clavier.

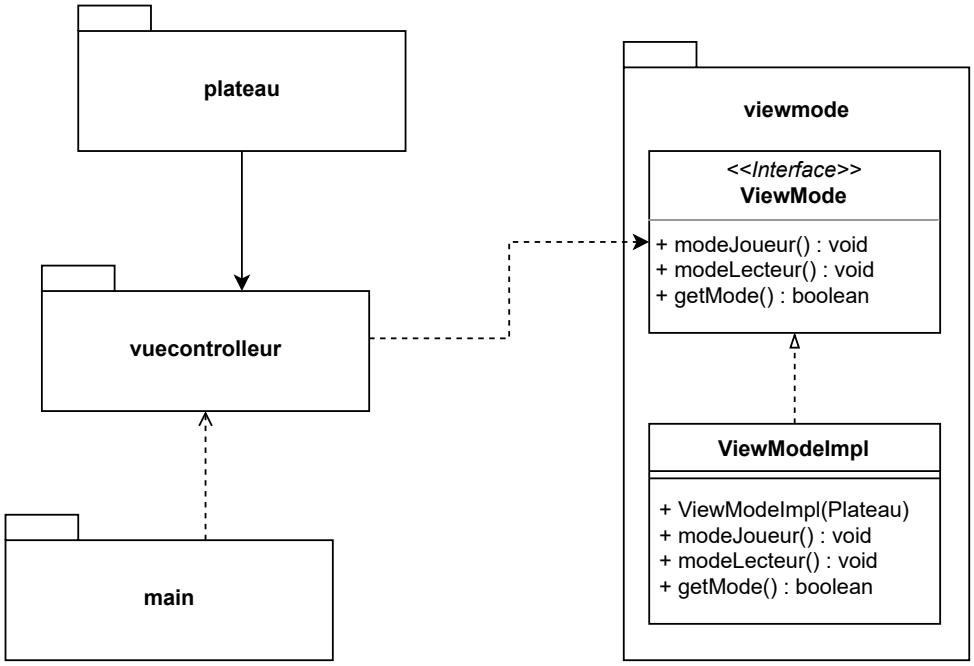

FIGURE 2 – Proxy Pattern

3.4 Strategy Pattern

Le but du Strategy Pattern est de faire différents modes de jeu. On développe simplement un joueur robot qui choisit alétoirement un point pour se déplacer, puis il tire les autres joueurs. Avec le Strategy Pattern, nous avons créé deux modes de jeu. L'un est "Jouer avec les bot" et l'autre est "Bot uniquement". De plus, nous configurons le nombre d'obstacles présents sur le plateau de jeu. Ainsi, nous pouvons choisir parmi trois modes avec ce pattern : avec peu d'obstacles, avec moyennement d'obstacles ou avec beaucoup d'obstacles.

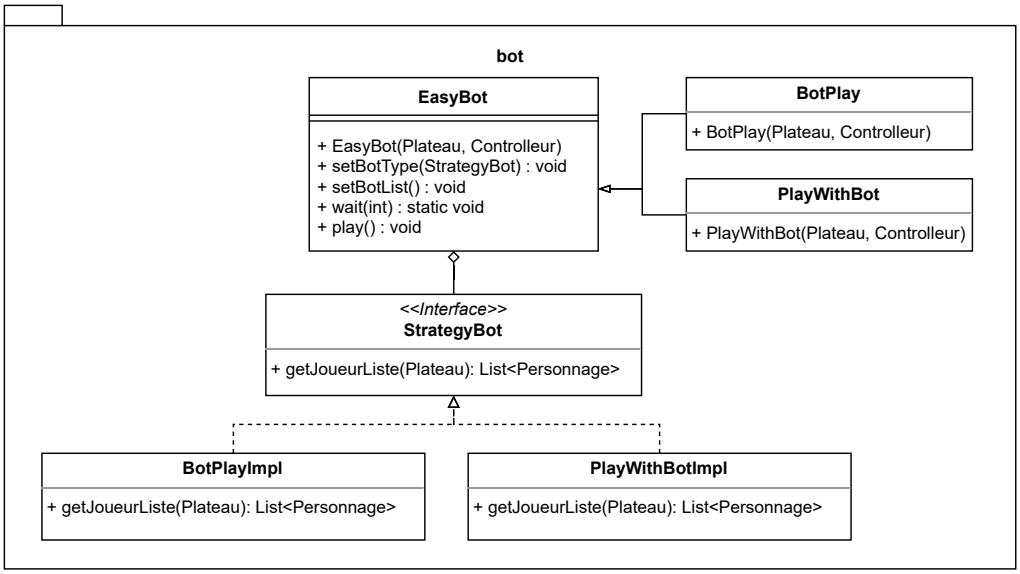

FIGURE 3 – Strategy Pattern pour joueur robot

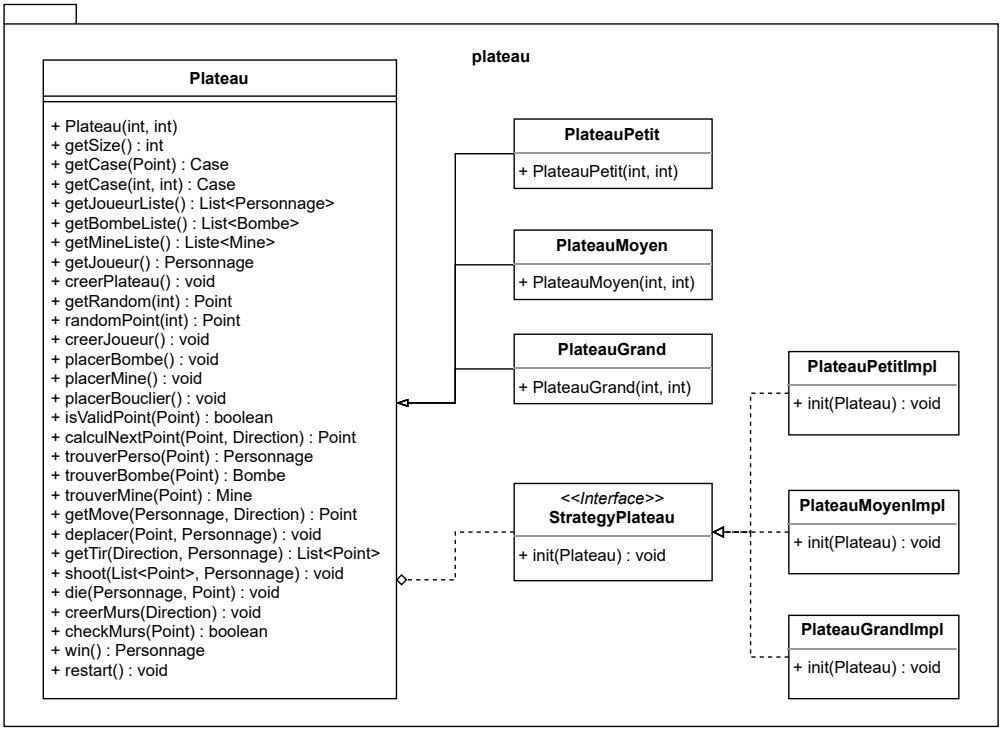

FIGURE 4 – Strategy Pattern pour le plateau

3.5 Modèle

Nous avons développé 5 modèles d'objets : **Bombe, Case, Direction, Mine, Personnage**.

3.6 Vue

La classe **Vue** utilise le **JPanel** pour faire l'interface. Puis, on hérite le **JFrame** pour appeler l'interface dans le **Main**. Pour ce jeu, nous avons choisi de faire un une vue dymanique et intéressante, c'est pourquoi nous avons choisit des images pour pour animer l'interface graphique.

3.7 Controlleur

Le contrôleur contient la logique concernant les actions effectuées par l'utilisateur. Pour faire les contrôleurs, nous avons implémenté l'interface **KeyListener**. Pour les déplacements du joueur, il faut utiliser **Z - Haut, S - Bas, Q - Gauche, D - Droit**. Pour tirer il faut utiliser les flèches de navigation. Pour rédémarrer le jeu, il faut utiliser la touche **espace**. Enfin pour changer mode de vue il faut utiliser **V**.

4 Capture d'écran

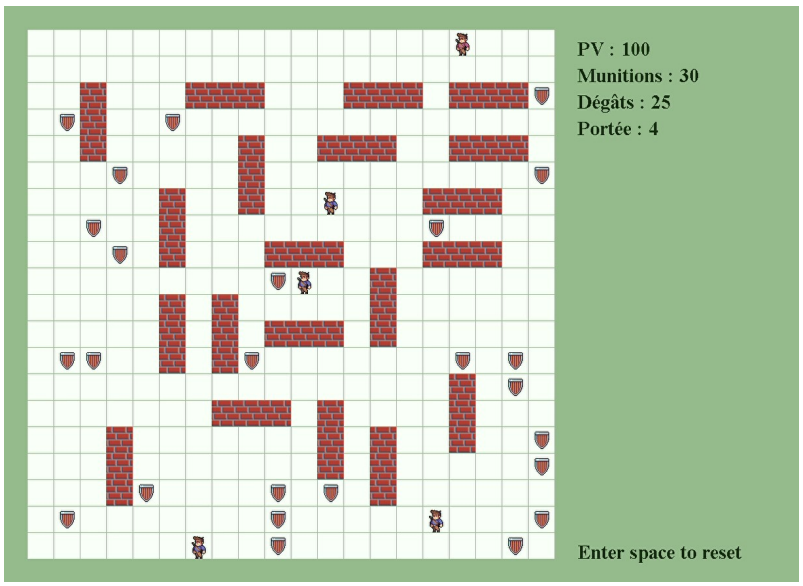

FIGURE 5 – Point de vue de joueur

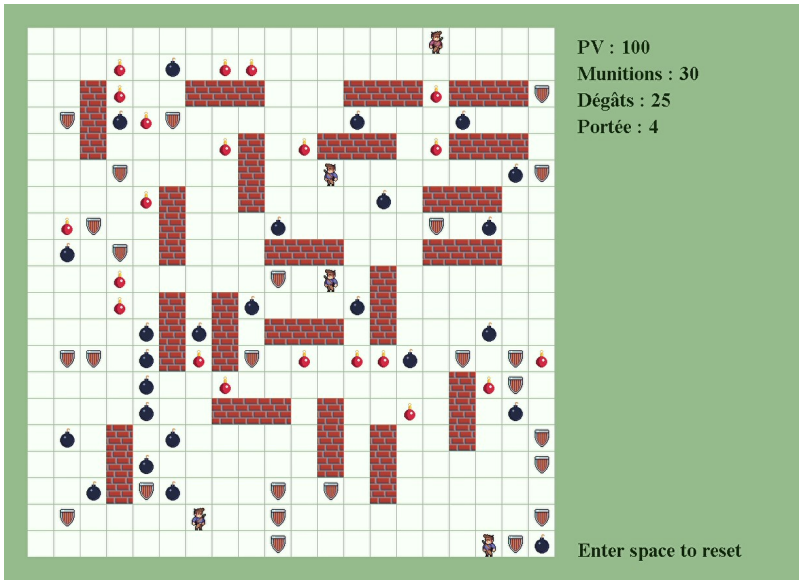

FIGURE 6 – Point de vue de viewer

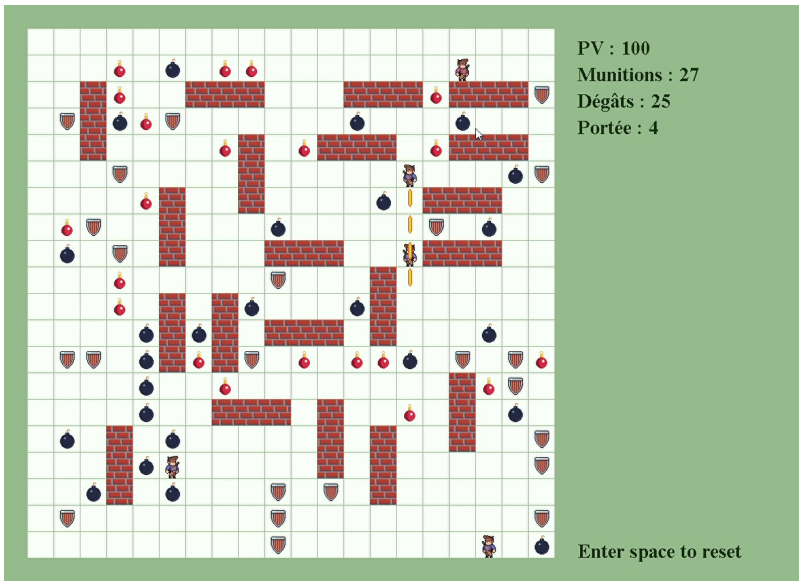

FIGURE 7 – Point de vue de viewer

5 Conclusion

Ce projet nous à facilement permit d'utiliser le MVC pattern, le Strategy pattern et le Proxy pattern grâce aux différents concepts qu'offre ce jeu de combat. En amélioration nous pourrions utiliser tous ces pattern de manière plus poussé.