Rapport modélisation métiers

Nader Hadj Youssef - 54685
Mohamed Bentouhami Belhaj - 56387

Table des matières

1. Présentation

Dans un premier temps, nous allons essayer de décrire comment le projet va être implémenté de manière générale ensuite, on va justifier la pertinence de nos choix du diagramme de classe et finalement l'explication des algorithmes « non-triviaux ».

2. Description générale

Tout d'abord, notre projet stratego contiendra 6 sous-projets :

1. **controller** : qui va faire le lien entre la vue et le modèle.
2. **view_console :** qui va pouvoir lancer notre projet en console.
3. **test** : qui contiendra des tests unitaires pour s'assurer des fonctionnalités du jeu.
4. **gui** : dédié au lancement du jeu en interface graphique.
5. **model** : va contenir nos différentes classes.
6. **placement_pawn :** l'endroit pour y mettre la mise en place probable par fichier.

Notre projet Stratego suivra l'architecture MVC (Model-Vue-Controller) comme nous avons l'habitude de faire et implémentera le design pattern Observateur-Observé pour la partie graphique.

3. Nos choix de modélisation

a. Mise en place

Choix : Nous avons laisser le choix aux deux joueurs de soit faire la mise en place du plateau de du jeu par fichier ou soit de demander interactivement les pions selon leurs choix.

Motivation : Comme indiqué dans les consignes les joueurs auront le choix pour la mise en place.

⚠ **Attention** : le choix de la mise en place se fera de la même manière pour les deux joueurs.

b. Plateau de jeu

Choix : Nous avons choisi de représenté notre plateau comme étant un vector de vector de Pawn pour représenter nos cases. Ce qui implique qu'un objet pawn aura un attribut pawnType (MARECHAL, GENERAL, … mais aussi des UNUSED et SPECIAL).

UNUSED : représente une case vide est sera construite par un constructeur différent de celui des autres Pawn.

SPECIAL : représente la case eau au milieu du plateau et sera aussi construite par notre deuxième constructeur de la classe Pawn.

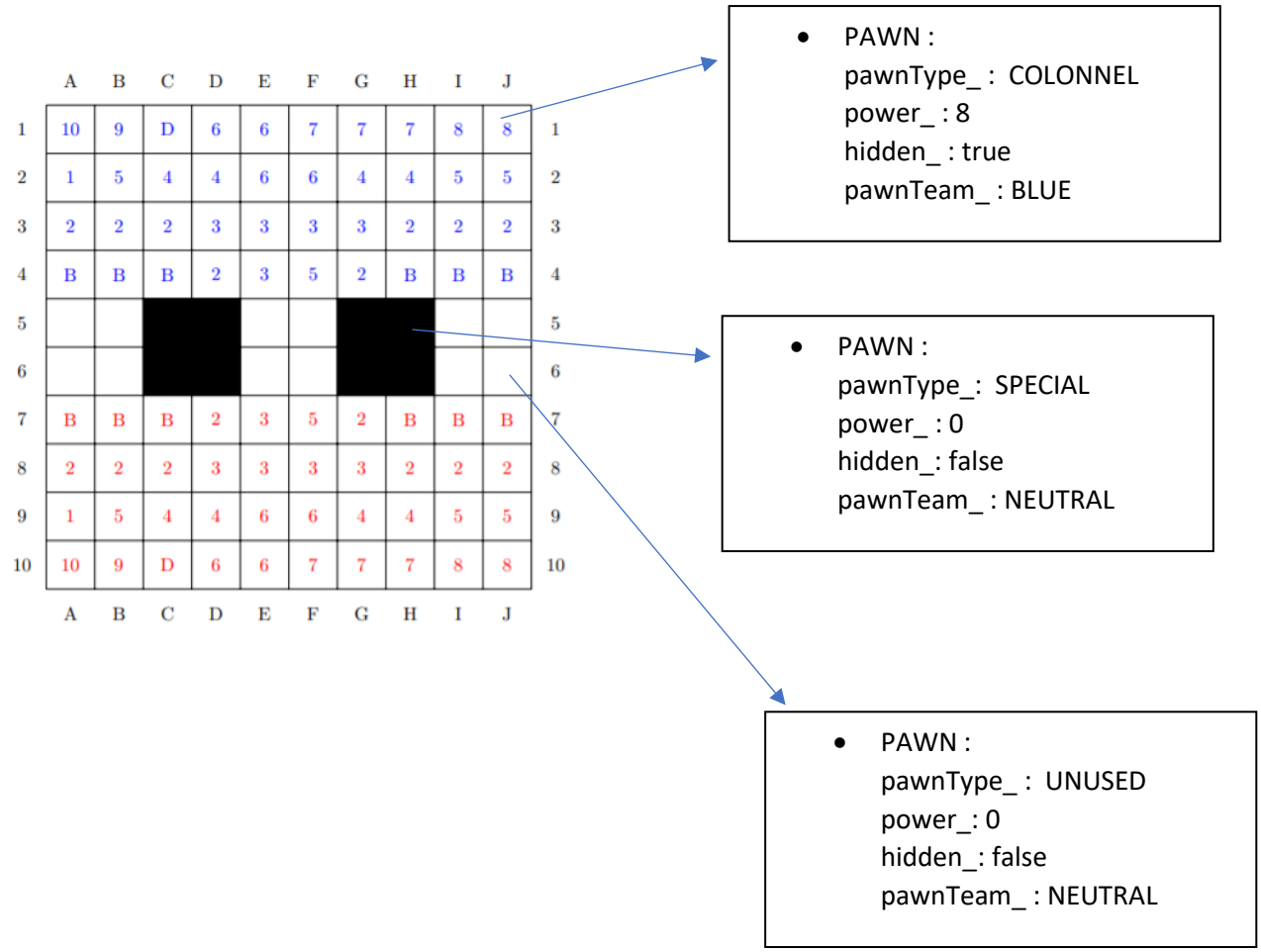
Motivation : Nous pensons que c'est plus intéressant d'avoir un vector de vector de Pawn avec un type commun (Pawn) au lieu de différencier les pions des cases vides et cases spécial (eau).

Pourquoi le vector ? Car c'est contenaires standard qui va gérer lui-même tout ce qui est mémoire plus avantageux que devoir la gérer nous-même.

⚠ **Attention** :

Faudra bien gérer le fait que les pions SPECIAL (eaux) sont inaccessibles donc pas déplacement de ces Pawn.

Schéma explicatif :



c. Distinction des deux joueurs

Choix : Pour distinguer les différents joueurs et pour que le joueur adverse ne puisse pas aller déplacer les pions de l'adversaire. Nous avons choisi de mettre un attribut PawnTeam dans la classe Pawn.

Motivation : C'est ce qui nous semble plus logique et ainsi pouvoir éviter la triche (déplacer les pions du joueur adversaire).

Attention : Faut bien vérifier lorsque le joueur choisi un pion afin de le déplacer que c'est bien le sien qu'il l'a choisi. Pour les pawn particulier comme les pions vides ou l'eau leurs pawn team sera neutral.

d. Présence d'énumération

Choix : Nous avons choisi d'avoir 3 énumération dans notre projet : PawnType, Direction, PawnTeam.

Motivation : Cela permet d'avoir une sécurité en plus le faite d'avoir des énumération. On contraint le faite d'avoir que les éléments des énumération et pas d'autres

Exemple : un Pawn ne peut avoir que comme PawnTeam que RED ou BLUE ou NEUTRAL pas d'autre valeur.

e. Le tour de chaque joueur

Choix : Nous avons choisi dans la classe Game d'avoir un attribut teamTurn_ qui va indiquer l'équipe a qui est le tour.

Motivation : Cela nous a semblé une bonne manière de géré le tour pour les deux joueurs.

4. Explication des algorithmes non triviaux

a. Méthode figth de la classe Board

prototype : void fight (Position pos, Direction direction, int déplacement);

Cette méthode va donc recevoir une position qui a été au préalable vérifier comme étant une position correcte avec une direction ainsi qu'un déplacement qui sera toujours de 1 excepté pour les espions qui eux peuvent se permettre d'en faire plus.

Sans oublié que cette méthode réglera les cas de combat exceptionnel. Voir exemple ci-dessous.

Exemple : espion (power = 1) bat maréchal (puissance = 10) donc maréchal meurt (devient un pion de type unused).

b. Méthode move de la classe Board

prototype: void move(Position pos, Direction direction, int déplacement =1).

Cette méthode va géré le faite que le Pawn à la position donnée selon ses caractéristiques son mouvement c'est-à-dire si la pièce va donc pouvoir faire un déplacement court ou déplacement long (l'espion) .

c. Méthode isOver de la classe Game

prototype : bool isOver()

Elle va géré le faite que la partie puisse continuée ou non. Comment ?

En regardant si un des deux drapeaux a été dévoiler (!isHidden) mais aussi en regardant si les pions restant peuvent bouger (bombe, drapeau ne peuvent pas bouger) .

d. Interdiction de déplacer certains de nos pions

En effet les pion FLAG, BOMB ne doivent en aucun être déplacer.

5. Vue d'ensemble

Voir annexe diagramme de classe.